ON SIMPLIFIED PREDICTIVE CONTROL VIA IP NETWORKS

by

Samer E. Mansour

Submitted in partial fulfillment of the
requirements for the degree of

DOCTOR OF PHILOSOPHY

Major Subject: Engineering Mathematics

at

DALHOUSIE UNIVERSITY

Halifax, Nova Scotia June 12th , 2006

© Copyright by Samer E. Mansour, 2006

Library and Archives Canada

Published Heritage Branch

395 Wellington Street
Ottawa ON K1A 0N4
Canada

Bibliothèque et Archives Canada

Direction du Patrimoine de l'édition

395, rue Wellington
Ottawa ON K1A 0N4
Canada

Your file Votre référence
ISBN: 978-0-494-27646-4
Our file Notre référence
ISBN: 978-0-494-27646-4

NOTICE:
The author has granted a non-exclusive license allowing Library and Archives Canada to reproduce, publish, archive, preserve, conserve, communicate to the public by telecommunication or on the Internet, loan, distribute and sell theses worldwide, for commercial or non-commercial purposes, in microform, paper, electronic and/or any other formats.

The author retains copyright ownership and moral rights in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

AVIS:
L'auteur a accordé une licence non exclusive permettant à la Bibliothèque et Archives Canada de reproduire, publier, archiver, sauvegarder, conserver, transmettre au public par télécommunication ou par l'Internet, prêter, distribuer et vendre des thèses partout dans le monde, à des fins commerciales ou autres, sur support microforme, papier, électronique et/ou autres formats.

L'auteur conserve la propriété du droit d'auteur et des droits moraux qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

In compliance with the Canadian Privacy Act some supporting forms may have been removed from this thesis.

While these forms may be included in the document page count, their removal does not represent any loss of content from the thesis.

Conformément à la loi canadienne sur la protection de la vie privée, quelques formulaires secondaires ont été enlevés de cette thèse.

Bien que ces formulaires aient inclus dans la pagination, il n'y aura aucun contenu manquant.

Canada

DALHOUSIE UNIVERSITY

To comply with the Canadian Privacy Act the National Library of Canada has requested that the following pages be removed from this copy of the thesis:

Preliminary Pages
- Examiners Signature Page
- Dalhousie Library Copyright Agreement

Appendices
- Copyright Releases (if applicable)

To the soul of my grandmother Nadime.

We really miss you and forever will, dearest "Tayta".

Table of Contents

List of Tables

List of Figures

Acknowledgements

Now that I am writing the final words in this thesis, I feel humbled more than ever. The more I learn the more I discover how much I do not know. And so I realize that the journey I started four and a half years ago did not end. In fact it will never end. No doubt the achievement of the degree is great but greater is the journey with both its bitter and sweet memories and greater are the people around me who made the journey so far possible and enjoyable. To those people I will always be grateful.

- First and foremost I would like to express my deepest gratitude to Dr. Kember who has helped with each step of the research. And there are no words that can express my gratitude to the Kembers. Guy, Suzi, Michaela, and, Erika, you were and will always be a true family to me.
- Dear parents, Nada, Abir, Manal, and Lellou: thank you for your unconditional support and sacrifices, and for always believing in me. I will always be indebted to you.
- I would like to thank my supervisors Dr. Phillips and Dr. Robertson for granting me the opportunity to pursue a Ph.D. degree.
- I would also like to take this opportunity to thank those relatives who provided sincere support especially my cousins Nadine, Hady, and Chady . A special thank-you goes to aunt Marie Toulany for her encouragement and the sincere support that emanates from the heart.
- I would like to thank all the friends who provided support, encouragement, and valuable advice: Alona, Maen and Muna, George, Elisabeth, Walid, Marcel, Jalal, Carlos, Abboudi, Joe and Pam, Maikel, Father Saikali, Serguei, Dr. Dubay, and Dr. Fenton.
- I would like to express my gratitude to Canada, this great nation that embraced me first as a foreign student and which I now proudly call my homeland.

Abstract

Online adaptation of the Simplified Predictive Control (SPC) from [14] to be used via IP networks is developed. No knowledge of network delay distributions, delay bounds or stationarity is assumed. Five protocols are proposed for scheduling the control tasks and communicating the control data and its usage in the additive feedback correction of the predictions. A generic, semi-discrete, analytical study of SPC is developed for each protocol. Stability analysis of the resulting delay differential equations (DDEs) allow for an understanding of the twin effects of measurement timestamping and assumptions regarding the timing of move application. To enhance the stability and robustness to random delay of the best performing protocol, a two-parameter variant of the SPC is formulated and Fuzzy-sensing of the network delay state is used to Fuzzy-schedule the two parameters of the SPC and adapt to future delay. A PI-SPC equivalence is developed in both cases of conventional and networked controls. It allows commonly-used PI control to provide equivalent control to SPC via a move-to-move PI parameter adaptation. All control methods are implemented as an Application layer protocol in the IP stack with the User Datagram Protocol (UDP) functioning as a Transport protocol. The implementation is based on the *ns-2* simulator which is modified here to include a controller and necessary plant agents.

Chapter 1

Introduction

1.1 Networked Control Systems: NCS

Industrial control networks, also known as Networked Control Systems (NCS) [34] or Integrated Control and Communication Systems (ICCS) [15], are attractive because of cost efficiency and flexibility especially in large scale systems that are geographically spread out and require extensive cabling. NCSs are also useful in environments that are harsh or inaccessible to human operators. A summary of control and communication integration prospects and future applications over networks is described in the report of the *"Panel on Future Directions in Control, Dynamics, and Systems"* involving industrial, educational, military, and governmental bodies [40].

Interestingly enough, Paul Baran, one of the researchers involved in the 1960's ARPANET project, the origin of the Internet, mentioned that the principle of "*redundancy of control and connectivity*" was the motivation behind the first packet switched network project

"The ARPANET stemmed from the need for a resilient control and communications system that would still provide after surprise attacks by giving connectivity and control access to military facilities through redundant routes and control centers" [3].

Despite this focus, packet switched networks gained more popularity as a means of information data exchange and veered from the control redundancy path. Hence, data communication networks in general were built initially for information data exchange without *hard-real-time* constraints in mind: as the network load increases the quality of service (QoS) of the network decreases and as a result the *random communication delay* and *packet loss* increases degrade the real-time quality of the network.

Random communication delay causes the closed-loop system to become time-varying, thus it may, depending on the size and nature of the stochastic delays, degrade the control performance and destabilize the system [34]. Since the real-time requirement is essential in control systems many specialized dedicated networks were built specifically for connecting sensors, actuators, controllers, and process monitors.

These local dedicated digital industrial networking technologies, dubbed generically *Fieldbuses*, are meant to replace old analogue technologies long used in industrial instrumentation and automation [41]. Industrial devices such as sensors, actuators, and controllers connected via Fieldbuses are thereafter dubbed *Field Devices.* A large number of Fieldbus technologies are available: AS-Interface, CAN, DeviceNet, Interbus, ARCNET, ControlNet, Profibus, LonWorks, SwiftNet, WorldFIP, HART ... [41]. Some LAN technologies used originally for information data communication were adapted for industrial networking: e.g., Industrial Ethernet and the wireless Bluetooth technology.

Fieldbuses, no different than any shared communication networks, still show communication delay and packet loss. However, the effect of such QoS imperfections on the real-time quality of the overall system is bounded since the industrial networking technologies were designed as dedicated systems. As such, they provide the quality of service required for real-time control with little alteration of the basic control formalism. The effect of the QoS imperfections is bounded in industrial networks mainly by

- employing centralized link layer medium access methods like token passing or a master station for polling agents. Furthermore, a transmission priority scheme can be assigned to packets or nodes. All these techniques are used to schedule transmissions. This guarantees deterministic upper bounds on inter-transmission times [18].

- having the industrial network under one authority. The number of field devices, their bandwidth requirements, access privileges, and any other factors that shape the traffic patterns, are predetermined in the design phase. Network resources are thus assigned according to predetermined specifications. This results in highly deterministic traffic or at least deterministic bounds on the traffic hence on the QoS [39] [18] [37] .

While Fieldbus networks are widespread, especially in the automobile industry, the more generally available and cheaper networks such as the Internet have seen less use in automatic control because of lower quality of service in comparison to that found in local, dedicated networks.

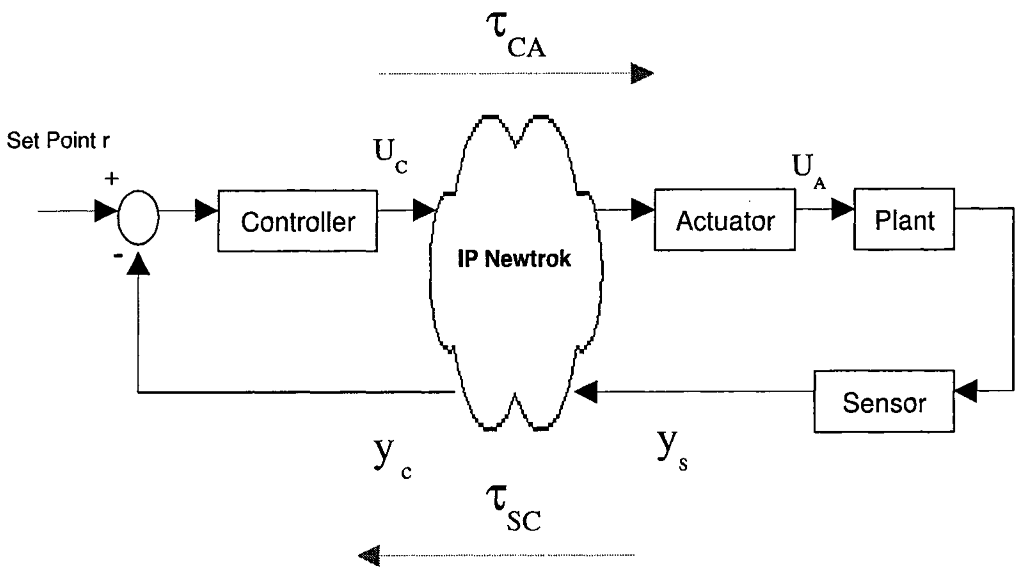
A chief downside of local industrial networks is their constrained distance of application. A partial solution to this problem is the use of the widespread and cheaply available Internet. It is particularly useful for applications where toleration of a lower quality of service is coupled with a need for control over a widespread geography. Hence, research into control strategies that are able to adapt to a lower quality of service is a rich area that is driving the current development of a range of control strategies.

In this thesis the focus is on Internet-based (IP-based) NCS. IP networks are *best effort* packet-switched shared networks. In other words **there are no guarantees on the QoS**. So the assumptions of bounded QoS parameters available in the case of the dedicated Fieldbuses are unavailable in basic IP networks. Of great interest for NCS in general and the work in this thesis, is that the characteristics of the communication delays of the Internet are different from those experienced over Fieldbuses [12]. Note here that IP networks that can provide QoS guarantees by using QoS and signaling protocols like DiffServ, IntServ, or MPLS for example, are higher priced than best effort and are not considered here.

In the remainder of this chapter the focus of this thesis is further discussed within the context of the current literature. Therefore, a summary of the related work on Fieldbus-based NCS and Internet-based NCS is presented Sections 1.2-1.5. The justification of the focus on SPC rather than on matrix-based predictive methods is detailed in Section 1.6. The equivalence between SPC and PI control is demonstrated in the absence of communication delays along with the rapid degradation of PI control relative to SPC control when the delays are present in Section 1.7. The thesis goals, contributions and organization are outlined in Section 1.8.

1.2 Illustration of NCS

A generic feedback NCS is illustrated in Figure 1.2. The delay from sensor to controller is τ_{SC} and from controller to actuator is τ_{CA}. The total delay through the closed-loop which can be looked at as a Round Trip Time (RTT) from and to the controller is $RTT = \tau_{CA} + \tau_{SC}$.



The measurement available at the controller site at a time t is $y_C(t) = y_S(t - \tau_{SC})$ and the control move available at the plant site is $u_A(t) = u_C(t - \tau_{CA})$.

1.3 What can NCS offer?

NCS is useful in

- industrial facilities. The field devices can be widespread over a large geographic area while the control can be centralized in a single place (control center). In this way, NCS cuts down on the costs of cabling and later in the maintenance of the systems.
- manufacturing. Machinery manufacturers can provide centralized control as part of their maintenance package. In fact control solutions provided by the manufacturers would be more beneficial since they have detailed knowledge of their machines.

- Building Automation, Control and Management (BACM). It helps provide automation of the HVAC, water , boiler rooms, and power systems. For example, the BACM LonWorks technology (ANSI standard EIA-709) is already gaining popularity. With the EIA-852 standard, LonWorks can be extended over IP networks.
- unreachable areas like space, or harsh environments (deep oceans, polar areas, or regions devastated by nuclear attacks).
- transportation networks. Vehicular Ad-hoc networks (VANET) are the subject of extensive researches. One of the main applications of VANETs is the automation and control of moving vehicles. An example is air transportation systems that already rely on networked monitoring facilities.
- distance learning, deployed military facilities (nuclear facilities), or remotely controlled moving robots (spy planes etc ...)

In general, NCS can provide

- centralized control to distributed plants (one-to-many).
- distributed control to a central plant (many-to-one). This can be useful for having backup controllers in case the main one fails or if control necessitates parallel processing.
- Simplified control equipment and computational power at the plant side. The plant will act as a server with the bare minimum requirements of receiving control moves,measuring, and transmitting measured output.
- The Controller side can be as complex as needed. The controller would be implemented as a client with high computation power and hence complex control algorithms.

The Internet is an attractive medium for NCS for the following reasons

- It is the most widely deployed packet switched internetwork. It is an ubiquitous medium, that is relatively cheaper than other forms of communication.

- IP networks in general and the Internet, in particular, provide a generic rather than a dedicated service that can be useful for a wide range of applications.
- TCP/IP internetWorking protocol suite has undergone and is still undergoing many improvements in QoS, real-time applications, and communication security. Although the main driving force behind the real-time requirements are the ever growing multimedia applications (like video conferencing, voice over IP, Internet gaming, online TV), the Internet-based NCS can still benefit from such improvements.

1.4 Related work in the case of Fieldbus-based NCSs

As mentioned in the previous section, Fieldbuses provide bounds on the QoS parameters of the network, thus all related studies assume bounds on the QoS. The QoS bounds of a control LAN or Fieldbus depend on the characteristics of its Physical and Data Link layers as seen in the Open Systems Interconnection (OSI) model. The Physical layer characteristics are mainly represented by the bandwidth and the number of nodes. The Link layer characteristics are mainly defined by the medium access protocols and their network resources sharing or transmission scheduling techniques. These characteristics can be predetermined at the NCS design stage, and after the deployment stay maintained and monitored by the appropriate authority in charge of the dedicated LAN or Fieldbus. Such networks are dedicated for the purpose of industrial control and not for information data exchange like e-mail, HTTP, or FTP applications, etc $\cdots$ Since control applications are subject to far more rigorous predetermined scheduling than typical Internet applications, such dedicated control networks have more deterministic characteristics. As an example of such dedicated networks is the CAN fieldbus deployed in automobiles for communication between sensors, actuators, and a central computer used for control and monitoring.

Some examples of related network control studies are

- In [15] an augmented state space description of the closed-loop system to account for periodic delays. Both the controller and the sensor are time-driven.
- In [24] buffers are introduced at both the controller and the actuator sides.

The sizes of the buffers are obtained from the maximum (worst-case) delays to be experienced which depends on the assumption of bounded delay. The introduction of buffers transforms the random delays into deterministic ones, thus making the system a time-invariant transport lag system. Buffering is used in multimedia applications over the Internet. This method can adapt to delays that are longer than the process sampling period but it always assumes longer delays than necessary which affects control performance. Both the controller and the sensor are time-driven. An LQG-Optimal controller compensates for the artificial deterministic delay.

- In [22] the probability of certain delay exceeding a preset maximum value, is assumed to be known. Both the controller and the sensor are time-driven. The controller uses an estimator to predict the state of the process when no measurement is available at the control calculation time. TimeStamping is used in the estimation. An LQ-Optimal controller is used.

- In [17] the process sampling period is optimized to be just large enough so that delays have minimal effect on the closed-loop control. It assumes that the delay is bounded.

- In [25] the NCS design is treated as an optimal stochastic control problem in discrete time domain. In this methodology the sensor is time-driven, the controller and the actuator are event-driven. Three types of delay distributions are considered: constant delay, independently random delays with fixed distributions, independent random delays where the distributions are varying according to Markovian jumps. In all three cases, the delay distributions and the Markovian properties are assumed to be exactly known. This methodology also assumes that the RTT is smaller than the sensor's sampling period. The methodology emphasizes the importance of TimeStamping and clock synchronization in knowing the history of the process state.

The stability of Fieldbus-based NCSs is emphasized in

- [39]. The total closed-loop delay τ is assumed constant the sensor is taken to be time-driven with sampling period h. The study is done both analytically and via monte-carlo simulations to determine stability regions as a function of the ratio of τ and h. The analytical study used a hybrid (continuous and discrete) scheme.

- [37]. This study was interested in the scheduling of the network resources among many control nodes (Sensors, actuators , controllers). A transmission scheduling protocol "Try-Once-Disregard" (TOD) was developed: the node tries to send its control data, but when new data is available the old one is discarded. It also advocated the avoidance of queues in control systems: "In real-time control systems, the newest data is the best data". The study of the NCS stability in both the TOD scheduling and the conventional static scheduling is based on the assumption of a Maximum Transmission Interval (MTI) (equivalent to maximum intertransmission interval): as long as the transfer time is within the MTI the system is stable. In [37] the study is expanded to investigate the effect of perturbations on NCS assuming the existence of the MTI.

1.5 Related work in the case of Internet-based NCSs

As stated earlier, the characteristics of the QoS in the Internet are different from those of dedicated control networks. The latter possess more deterministic properties. The QoS characteristics of the Internet depend mainly on the Network layer functionalities of routing. The communication delay over the Internet results mainly from waiting in the routing queues. The waiting delay nature of a certain Internet link depends on the size and the shape of the overall traffic passing through that link. The sources and characteristics of delay over the Internet will be discussed in more details later.

The Internet has a best effort delivery: there are no guarantees about the size of the communication delay or the shape of its distribution. Moreover, all best effort traffic is treated with the same preference. Thus control applications implemented via the Internet must be designed taking into account probable large delay jumps and thus expect a certain rate of failure. Large delay jumps mean that one has to live with the fact that, so far, Internet communication delays are not fully controlled or predictable [30].

Note that the Internet can offer differentiated services and guaranteed QoS as opposed to the regular best effort service for a price.

The following is a summary of some studies and methodologies of Internet-based NCSs based upon the assumption of differentiated services

- In [32] an event-based methodology is proposed to decrease the dependence of the closed-loop on time. In this methodology, intermediate set-point references are scheduled according to network events or the occurrence of certain error sizes until the process reaches the desired set-point.

- In [30] a useful introduction is provided on the topic of IP-based NCS. The main focus is on defining the QoS parameters that affect control and it classifies control applications into different categories depending on their QoS requirements and tolerance. It discusses the two main philosophies used to integrate Fieldbus technologies with IP networks: i) the gateway approach where all nodes in the dedicated control network have native IP implemented on them in addition to the Fieldbus protocols ii) the tunneling approach where only selected nodes have IP implemented and these nodes encapsulate the control data into IP datagrams to connect local and remote field devices via the IP channels. The simulation example in this study uses the EIA-852 standard regulating the IP tunneling for the LonWorks Fieldbus (EIA-709) networks. Note that the IP tunneling is mainly used in the Building Automation, Control, and Management (BACM) applications like boilers and heating systems control, air quality control, water systems control... Such applications are classified as 'soft real-time'. The study concludes with the importance of control packets sequencing.

- In [26] two techniques are simultaneously used to adapt the force-reflecting teleoperation system: i) the buffering technique where the temporal length of the buffer is 3 times the delay standard deviation ii) the network state is constantly monitored and the sampling period of the control system is set accordingly. It suggests using a state predictor to compensate for lost packets, and a backup local controller at the plant site to guarantee the safety of the system. Note that in this research it was assumed that the delay standard deviation is small and that the additional delay caused by buffering is negligible.

- In [1] the gains of a PI controller are fuzzy scheduled. This is done by multiplying the output of the PI controller by a factor β. β is obtained from a fuzzy system that is tuned via an offline optimization of some cost functions that reflect the overshoot, undershoot, and oscillatory behavior under different delay distributions. The multiplication by β of the PI output is equivalent to scheduling the PI parameters.

- In [33] and [35] a study based on the same notion of PI gains scheduling as in [1] is presented. However, offline simulations, under different generalized exponential distributions of delay, are used to build a lookup table for the parameter β instead of fuzzy logic. Network delay statistics are gathered, over a chosen window width, and matched as close as possible to one of the simulation distributions for looking up the appropriate β. Both the controller and the actuator are time-driven: when new data is not available for either of the two, the latest data is used for computation. Out-of-order packets are ignored in this control scheme.

- In [12] a similar combination of buffering and prediction as in [26] is used. But in this case two buffers are setup at the controller side and one on the actuator side. It is assumed that the delay in both directions is bounded: $\tau_{SC} < \overline{\tau}_{SC}$ and $\tau_{CA} < \overline{\tau}_{CA}$. The controller, the sensor, and the actuator are time-driven with respective periods Δ_c, Δ_p, and Δ_p. Δ_c is chosen to be a multiple of Δ_p in such a way to guarantee that enough information is available at the controller side before a control move is calculated. Received measurements are accumulated in a buffer at the controller for every period Δ_c and then used to calculate the same number of control moves that would be sent to the actuator in one packet. This packet is sent repeatedly to guarantee that it will reach the plant side. The actuator puts the received moves in a buffer and applies them one by one at dates equidistant by Δ_p, following a FIFO policy.

- In [19] a quality-controlled predictive control method, suitable for control of fast, remote systems subject to significant communication delays, is developed.

Each move is quality controlled in that it independently satisfies a risk-based control performance criterion. The method is found to be capable of good control performance for mild non-stationarity. It is demonstrated on simplified predictive control (SPC) of a single-input, single-output process. The delay distribution is known and assumed to be Lognormal.

1.6 Model Predictive Control

Simplified Predictive Control (SPC) belongs to the general class of Model Predictive Control (MPC) developed by Cutler and Ramaker in the late 70's [11]. To illustrate MPC the next subsection discusses the Dynamic Matrix Control (DMC) method first introduced in [11].

1.6.1 Dynamic Matrix Control

Suppose a ***linear and time-invariant*** plant is excited by a step input (open-loop test) and normalized. An example is shown in Figure 1.1. Let the normalized continuous open-loop response be denoted $p(t)$.

If $p(t)$ is sampled using a sampling period T_s the result can be viewed as a control moves input vector ΔU that resulted in a plant output vector p

$$\begin{array}{ccccccccc} \Delta U & = & 1 & 1 & 1 & \cdots & 1 & 1 \\ p & = & p_0 & p_1 & p_2 & \cdots & p_{N-1} & p_N \end{array} \tag{1.1}$$

Assuming linearity and time-invariance of the plant then it can be predicted that a step input to the plant of size Δu_0 will give the following input-output vector pair

$$\begin{array}{ccccccc} \Delta U_0 & = & \Delta u_0 & \Delta u_0 & \cdots & \Delta u_0 & \Delta u_0 \\ p & = & \Delta u_0.p_0 & \Delta u_0.p_1 & \cdots & \Delta u_0.p_{N-1} & \Delta u_0.p_N \end{array} \tag{1.2}$$

Moreover a shifted input to the plant of size Δu_1 would yield the following pair of input-output vectors

$$\begin{array}{cccccccccc} \Delta U_1 & = & 0 & 0 & 0 & \cdots & \Delta u_1 & \Delta u_1 & \cdots & \Delta u_1 & \Delta u_1 \\ p & = & 0 & 0 & 0 & \cdots & \Delta u_1.p_0 & \Delta u_1.p_1 & \cdots & \Delta u_1.p_{N-1} & \Delta u_1.p_N \end{array} \tag{1.3}$$

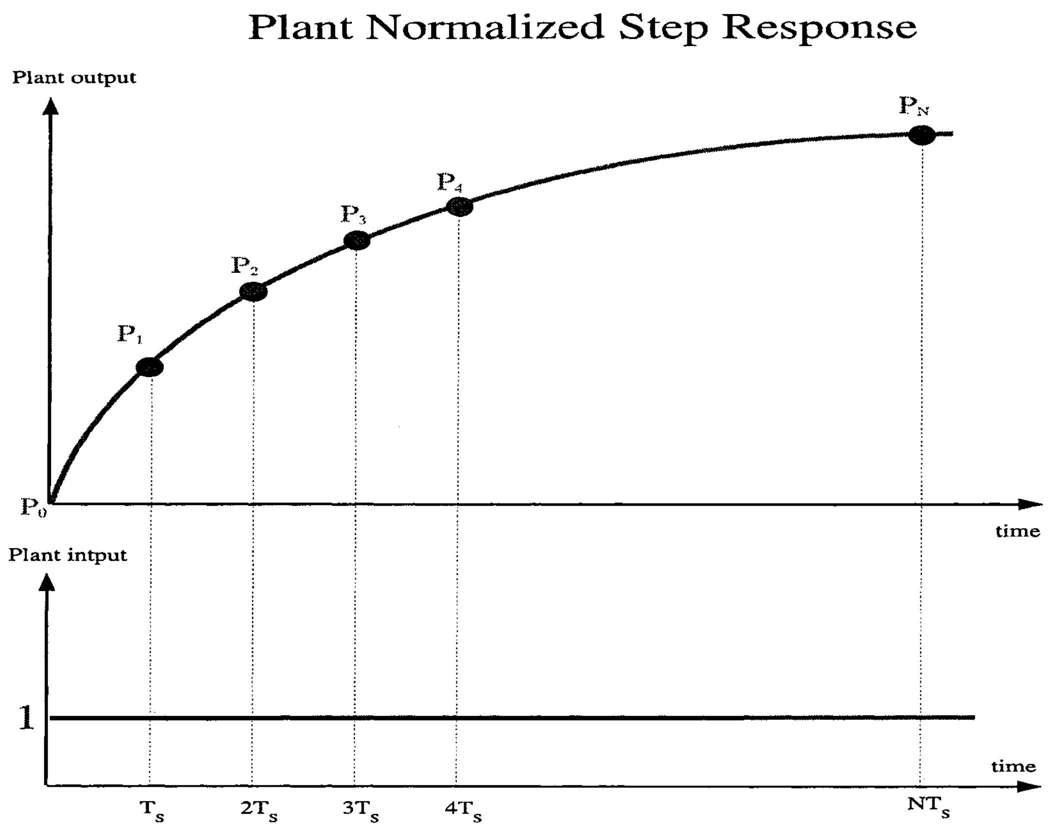


Figure 1.1: Normalized Step Response

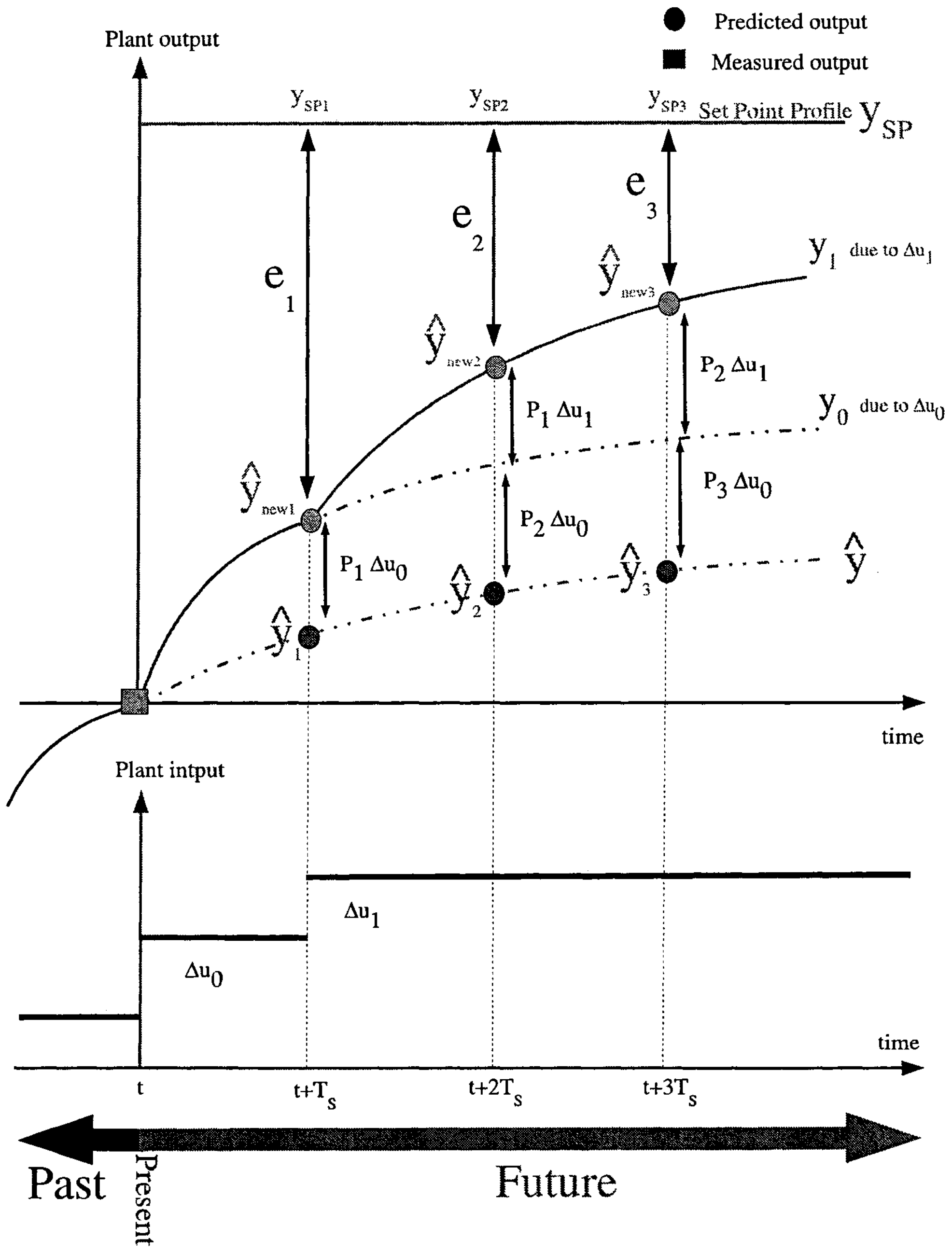


Figure 1.2: MPC scheme

The previous relations will be useful to clarify a basic example given in Figure 1.2, to illustrate the DMC theory.

In Figure 1.2 suppose that at sampling instant t, $\hat{y}$ is the plant output that will follow if the controller's output stays at the value it had before instant t. Also suppose that at instant t the controller is to make a control move Δu_0. Note that if at instant t no action was taken by the controller (i.e. no change in controller's output $\Delta u_0 = 0$) then the process will follow the graph $\hat{y}$. However, if the controller takes action and imposes a step change control move of size $\Delta u_0 \neq 0$, then by time-invariance and linearity the predicted plant output at instant $t + iT_s$ (i=1,2...) will be the previous prediction $\hat{y}_i$ plus the contribution of the step Δu_0. For example, at time instant $t+T_s$ the plant output is predicted to be $\hat{y}_1 + p_1.\Delta u_0$, and at $t+T_s$ it is $\hat{y}_2 + p_2.\Delta u_0$. Now assume that the controller knows the control move that is going to take place in the future at instant $t + T_s$ (Δu_1). In this case the controller can predict what the plant output will be in the future by predicting the effect of both Δu_0 and Δu_1 on the process output. At instant $t + T_s$ the effect of Δu_0 is $p_1.\Delta u_0$ and the effect of Δu_1 is zero. At instant $t + 2T_s$ the effect of Δu_0 is $p_2 \Delta u_0$ and the effect of Δu_1 is $p_1 \Delta u_1$. Similarly, at at instant $t + 3T_s$ the effect of Δu_0 is $p_3 \Delta u_0$ and the effect of Δu_1 is $p_2 \Delta u_1$ and so on. The predicted effect of Δu_0 is the graph y_0 and that of Δu_1 is the graph y_1. The predicted plant output is as follows

$$\begin{array}{ll} \text{At } t & \hat{y}_{new_0} = \hat{y}_0 = \text{current sampled value of the plant output} \\ \text{At } t+T_s & \hat{y}_{new_1} = \hat{y}_1 + p_1.\Delta u_0 \\ \text{At } t+2T_s & \hat{y}_{new_2} = \hat{y}_2 + p_2.\Delta u_0 + p_1.\Delta u_1 \\ \text{At } t+3T_s & \hat{y}_{new_3} = \hat{y}_3 + p_3.\Delta u_0 + p_2.\Delta u_1 \end{array} \tag{1.4}$$

The general form of prediction can be viewed as a convolution of the control moves and the normalized open-loop response $p(t)$. For example the prediction, $\hat{y}(t_S)$ of the output of the plant at some time t_S, starting from 0 initial conditions, if the control moves Δu_j ($j = 0....n$) were applied respectively at times t_{A_j} would be

$$\hat{y}(t_S) = \sum_{j=0}^{n} \Delta u_j \; p(t_S - t_{A_j})$$

Note that there are two key values in DMC:

- **The prediction horizon N.** This is how many sampling instants in the future the controller predicts the plant output. This predicted control occurs at sampling instants $t,\ t+T_s,\ t+2T_s,\ \cdots,\ t+NT_s$. This means that the controller predicts the output over N sampling instants into the future. In the set of equations 1.4 the plant output was only predicted 3 sampling instants in the future and the **prediction horizon N=3.**

- **The control horizon** n_u. This is how many future control move step changes are calculated by the controller. These future step changes are denoted Δu_0, Δu_1 $\cdots$ Δu_{n_u-1}. The controller predicts the plant output with n_u control move step changes happening at sampling instants $t,\ t+T_s,\ t+2T_s,\ \cdots\ t+(n_u-1)T_s$. For example in figure 1.2 and the set 1.4 the controller only predicted two future control moves Δu_0 and Δu_1, so in this case the **control horizon** $n_u = 2$. Note that Δu_0 is still a future move because it is yet to be sent to the plant.

Part of the process of DMC tuning involves tuning N and n_u. N is generally chosen from an open loop test as the location where the plant output reaches 95% or more of the steady state value.

Now an error vector can be calculated $[e_1,\ e_2,\ e_3,\ \cdots,\ e_N]$ as shown in Figure 1.2 where $e_i = y_{sp_i} - \hat{y}_{new_i}$. The curve y_{sp} is known as the **set-point profile or trajectory.** It is a desired trajectory that the plant output should follow while going towards the set-point; y_{sp} can be a straight line equal to the desired set-point (most aggressive form) or an exponential trajectory with a steady state equal to the set-point. For simplicity consider the set-point and Y_{sp} to be equal to the set-point r.

DMC controller decision making is based on making the best effort so that the predicted plant output would follow the set-point trajectory as closely as possible. The basis of DMC decision making will be explained using the simple example of Figure 1.2.

The prediction of errors, e_1, e_2 and e_3, are as follows

$$e_1 = y_{sp_1} - (\hat{y}_1 + p_1\Delta u_0 + p_0\Delta u_1) \tag{1.5}$$

$$e_2 = y_{sp_2} - (\hat{y}_2 + p_2\Delta u_0 + p_1\Delta u_1) \tag{1.6}$$

$$e_3 = y_{sp_3} - (\hat{y}_3 + p_3\Delta u_0 + p_2\Delta u_1) \tag{1.7}$$

Let the **Performance Index I** be

$$I = e_1^2 + e_2^2 + e_3^2 \tag{1.8}$$

To have the plant output follow the set-point trajectory as close as possible, I in 1.8 should be minimized. In other words at instant t the DMC controller should choose Δu_0 and Δu_1 so that I is minimum. That can be done by solving for $\frac{\partial I}{\partial \Delta u_0} = 0$ and $\frac{\partial I}{\partial \Delta u_1} = 0$. Using equations (1.5, 1.6, and 1.7) to calculate I then partially differentiating it, leads to the least squares formulation

$$\Delta U = (P^T \cdot P)^{-1} \cdot P^T \cdot [y_{sp} - \hat{y}] \tag{1.9}$$

where

$$P = \begin{bmatrix} p_1 & 0 \\ p_2 & p_1 \\ p_3 & p_2 \end{bmatrix} ; \quad \Delta U = \begin{bmatrix} \Delta u_0 \\ \Delta u_1 \end{bmatrix} ; \tag{1.10}$$

$$\hat{y} = \begin{bmatrix} \hat{y}_1 \\ \hat{y}_2 \\ \hat{y}_3 \end{bmatrix} ; \quad y_{sp} = \begin{bmatrix} r \\ r \\ r \end{bmatrix} . \tag{1.11}$$

The above is an example for a DMC controller with a control horizon $n_u = 2$ $[\Delta u_0,\ \Delta u_1]$ and a prediction horizon $N = 3$ $[p_0, p_1, p_2, p_3]$. The more general derivations for any N and nu immediately follow.

To summarize, an DMC controller decides at an instant t what control move to make by finding n_u future control move step changes (a current and $n_u - 1$ future moves) that result in a predicted function $\hat{y}_{new}$ that has minimum distance (measured by the performance index) to the set-point profile. Finding the n_u control move step changes is accomplished using a least squares formulation of equation 1.9, where

$$P_{(N\times n_u)} = \begin{bmatrix} p_1 & 0 & 0 & \cdots & 0 \\ p_2 & p_1 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ p_N & p_{N-1} & p_{N-2} & \cdots & p_{N-n_u+1} \end{bmatrix} ; \Delta U_{(n_u\times 1)} = \begin{bmatrix} \Delta u_0 \\ \Delta u_1 \\ \vdots \\ \Delta u_{n_u-1} \end{bmatrix} ; \quad (1.12)$$

$$\hat{y}_{(N\times 1)} = \begin{bmatrix} \hat{y}_1 \\ \hat{y}_2 \\ \hat{y}_3 \\ \vdots \\ \hat{y}_N \end{bmatrix} ; \quad y_{sp(N\times 1)} = \begin{bmatrix} r \\ r \\ r \\ \vdots \\ r \end{bmatrix} . \quad (1.13)$$

The matrix (P^TP) is ill-conditioned because the columns of P are nearly equal, i.e. they become equal in the limit of zero sampling time where the determinant of P^TP vanishes. For this reason the DMC formulation ill-conditioned to small changes in sampling time. The ill-conditioning may be reduced by a heuristic multiplication of the diagonal elements of P^TP by a constant or weighted values w_i all marginally greater than 1. This has the blackbox effect of increasing the determinant and reducing the ill-conditioning. So equation 1.9 becomes

$$\Delta U = (P^T \cdot P + W)^{-1} \cdot P^T \cdot [y_{sp} - \hat{y}] \quad (1.14)$$

where W is the matrix formed only by the elements of the diagonal of (P^TP) each multiplied by $(w_i - 1)$.

Tuning a DMC involves tuning 3 parameters: the control horizon n_u, the prediction horizon N, and the move suppression factor w_i. The parameter w_i is the trickiest parameter to tune. The discovery of w_i becomes increasingly difficult as the desired rate of control increases. It must also be re-tuned whenever the sampling time is changed.

1.6.2 Simplified Predictive Control

SPC [14] is MPC without matrices. In SPC, one move is predicted so that $n_u = 1$. Besides the simplicitiy of the method it is well-conditioned with respect to changes in sampling time. The performance index in equation 1.8 only involves one error e_α which is the error t_α seconds into the future, i.e. at $t + t_\alpha$. The one control move Δu_n, calculated at time $t = nT_s$ to minimize e_α, is

$$\Delta u_n = \frac{e_\alpha}{p_\alpha} \tag{1.15}$$

where p_α is the value of the normalized open-loop at time t_α. The value e_α is the predicted error (predicted distance from the set-point) t_α seconds into the future. Note that SPC depends on one parameter [21], β (or equivalently $\alpha = 1 - \beta$), which is equal to the fraction of the normalized open-loop steady-state at t_α, i.e.

$$\beta = 1 - \alpha = \frac{p_\alpha}{\text{Steady-state}}$$

A two-parameter SPC arises when in Δu_n, the term p_α is replaced by a new gain parameter k so that

$$\Delta u_n = \frac{e_\alpha}{k} \tag{1.16}$$

and the gain k is independent of the prediction horizon $exp(-t_\alpha) = 1 - \beta$.

Continuous approximations for DMC and SPC were developed respectively in [20] and [21]. In [21] a surprising finding was that the simple SPC algorithm generalizes the more complicated DMC: for each DMC it is possible to find an equivalent SPC. Moreover, the SPC is well conditioned on the whole range of the process sampling period while DMC is not [21]. In this thesis, it is proved that a two-parameter version of SPC is capable of duplicating Proportional-Integral control (PI). This makes SPC a PI alternative that does not suffer so greatly from the effects of noise due to communication delays: subsection 1.7.2 shows that an SPC is more robust to the change in the closed-loop communication delay than an equivalent PI controller.

1.7 Why SPC?

The recourse to model prediction fits naturally with the NCS problem. For example when the controller receives old information on the state of the plant because of the delay, model prediction is useful to estimate the most recent state. Missing measurements can also be estimated with model prediction [26]. Many NCS methodologies have recourse to state predictor/estimator techniques that are based on model prediction independent of the control method [26] [12]. From this point of view, the advantage of the MPC is that the prediction and control method are joined. SPC itself is an MPC method, but one which possesses the following main advantage

- Simplicity/Well-conditioning. SPC has a well-conditioned dependence on its sole parameter [20] and [21]. This makes its tuning independent of the sampling time which is of prime importance in the presence of communication delays. It is also less computationally demanding than DMC.

- Robustness. This term is taken here to mean that the control performance does not suffer greatly under changes in the sampling time. The SPC method is robust to variations in communication delay because of its well-conditioned dependence on changes in sampling time. This makes it a better candidate for NCS control than PI or DMC. This point is detailed in Chapter 8.

It should be stated as a footnote that all forms of predictive control rely upon an approximate process model. In contrast, PID and its variants do not require such a model since knowledge of the system is derived from instantaneous estimates of the error derivatives. Furthermore, predictive methods rely upon linear and time invariance in order to form predictions in an additive fashion using convolution. When a process cannot be approximated in any fashion as a linear or time-invariant system the PID methods are likely to be more useful than predictive methods. Given the assumption of an approximate linear and time-invariant process then predictive methods become available along with their robustness to changes in sampling times. The assumption of approximate linearity and time-invariance is also predominant in industrial implementations of PI control in any case. Hence, the study of SPC and its equivalence to PI will be shown later in this work to allow the movement of PI control over to control over IP which would otherwise be more difficult.

The robustness of SPC and the corresponding lack of robustness of PI control with respect to communication delays are demonstrated in the next subsections.

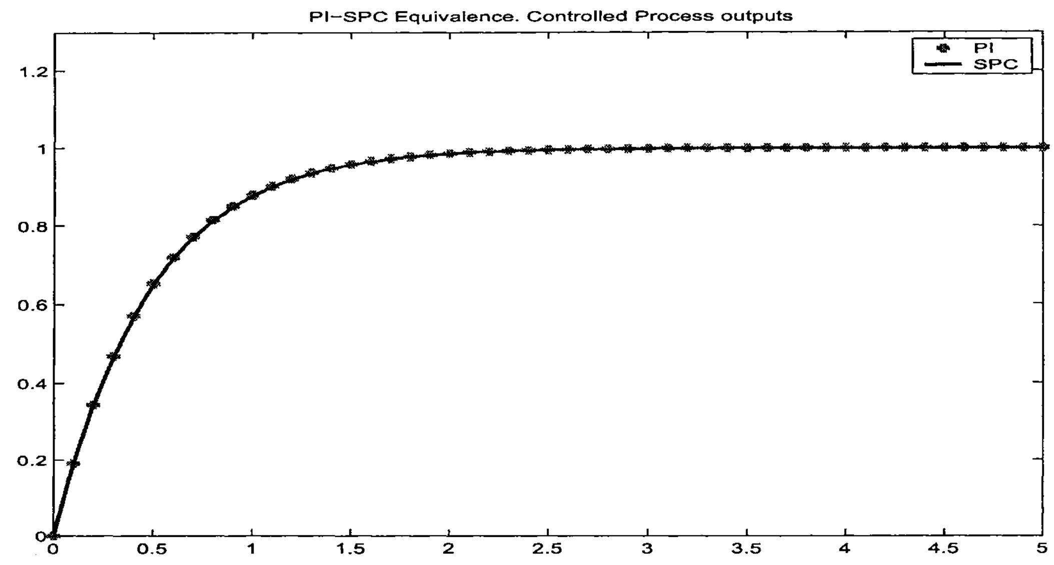


Figure 1.3: PI-SPC Equivalence. No delay in closed-loop. Controlled process outputs.

1.7.1 Simulation example of equivalent PI and SPC controllers

In Chapter 8, Section 8.2, an exact equivalence between SPC and PI control is developed. Using the equivalence, two controllers, a one-parameter SPC and a PI are tuned to yield equivalent control of the first order plant

$$P(t) = 1 - e^{-t}$$

In both cases the sensor sampling period is $T_s = 0.1$ *seconds* and there is no communication delay. Choosing the SPC parameter to be $\beta = 0.5$ yields, using equation 8.16, an equivalent PI controller with $K_p = 9.5083$ and $T_i = 0.4754$. Figures 1.3 and 1.4 show equivalent resulting control of the two methods.
In the next subsection, communication delays are introduced into the closed-loop control scheme and the robustness to delay is compared.

1.7.2 SPC vs. equivalent PI: delay robustness

The controllers in the previous subsection (1.7.1) yielded equivalent control in the absence of any delay in the closed-loop. In order to test their robustness to delay,

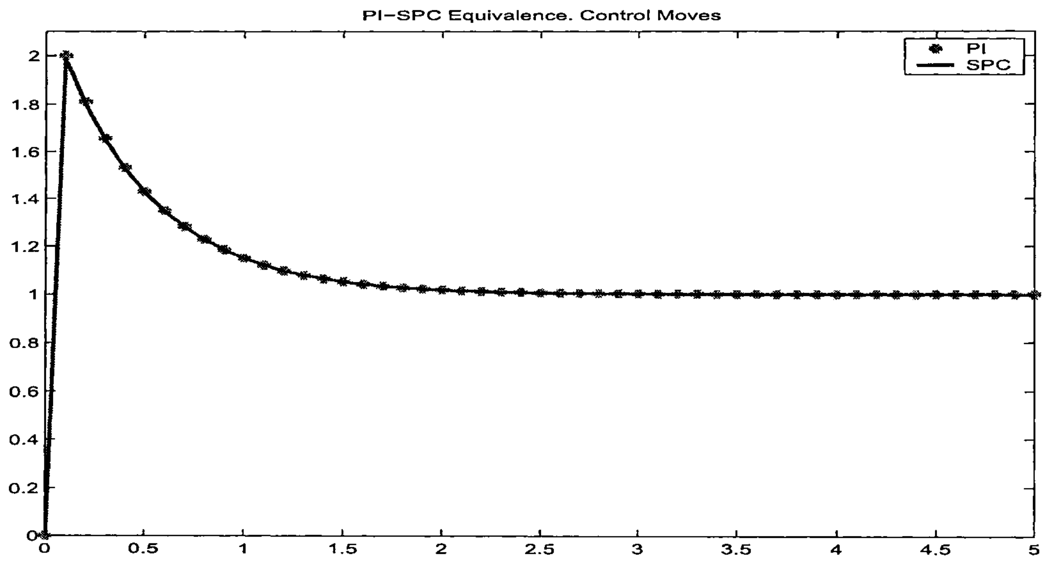


Figure 1.4: PI-SPC Equivalence. No delay in closed-loop. Control moves.

a constant communication delay is introduced. The modified *ns-2* simulator that will be introduced later was used and the constant delays were produced using large bandwidth links and manipulating the propagation delay along the links. Figure 1.5 shows that the SPC possesses excellent robustness to closed-loop delay in comparison with PI. One can see how the two controllers are still equivalent at the start of the control after which the PI controller starts to deviate and produce undesirable oscillations with a period at the order of the plant time constant. The SPC was actually able to accommodate constant delays up to 300 ms before producing even small oscillations at this timescale while maintaining acceptable control performance. In the case of random delays, their variations cause the PI robustness to degrade at shorter delays.

1.8 Thesis goal, contributions, and organization

The *goal* of this thesis is to adapt SPC [14] for usage via IP networks without knowledge of delay distributions or delay bounds.

The *contributions* of this thesis can be summarized as follow

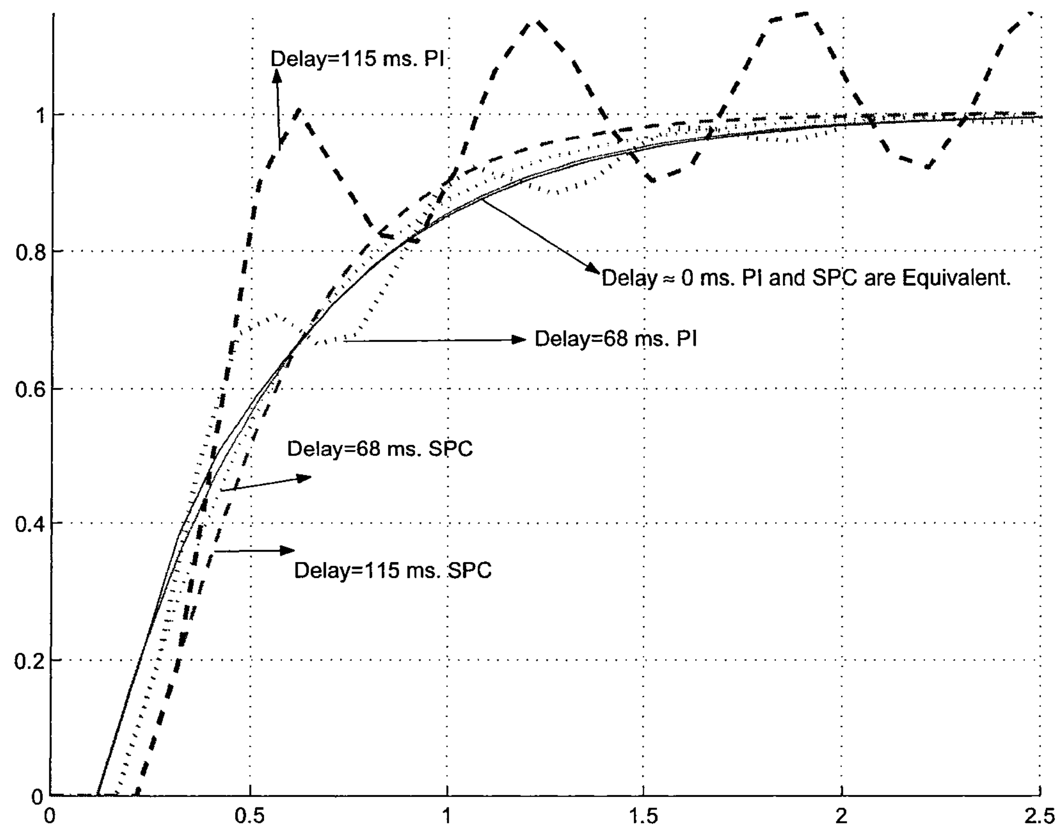


Figure 1.5: Behavior of equivalent PI and SPC controllers under different values of constant delay.

- Two series of protocols are implemented for control tasks scheduling and control data communication and its usage in the predictions. The first series deals with Measurement TimeStamping and its usage for Additive Feedback Correction of the prediction convolutions. The second series deals with Control Move Acknowledgement and its usage in the timing of the prediction convolutions. (Chapter 3).

- A generic semi-discrete analytical study of the SPC, for each protocol of the two series and in the basic case of constant communication delays (Transport Lag), is developed and yields a series of Delay Differential Equations (DDE). (Chapter 4).

- The method of Matched Asymptotic Expansions (MAE) is used to asymptotically approximate the solutions of the DDEs and to compare the performance and stability of the SPC in the case of each protocol. Both the analytical study and simulation results show that the knowledge of the timing of process measurements, or Measurement Timestamping, was more efficient for providing good control than the knowledge of the exact moves application times (Chapters 5, 6).

- The stability and robustness to random delay of the best performing protocol are expanded by formulating a two-parameter variant of the SPC. A fuzzy decision system is constructed that senses the state of the network and schedules the two SPC parameters accordingly (Chapter 7).

- A PI-SPC equivalence is developed in both cases of conventional control and NCS. This allows the most widely used form of control, the PI control, to benefit from the stability of SPC (Chapter 8).

- SPC is implemented as an Application layer protocol of the IP stack using the *ns-2* simulator. The *ns-2* network simulator is used as an experimental testbed and modified to include two agents: (i) a controller agent to simulate the controller application (PI or SPC), (ii) a plant agent to simulate the plant application that also include a sensor and an actuator. All the protocols in the two series are tested in an *ns-2* setup that provides random communication delays and packets loss as well as alternate routes (Chapter 2).

Chapter 2

The *ns-2* Network Simulator and Experimental Setup

2.1 Introduction

IP networks in general and the Internet in particular, are complex macrosystems constituted of interconnected autonomous heterogeneous microsystems. Each microsystem has its own governing authority and access privileges, and may be subjected to different types of inputs that would affect the network as a whole. RTT over an IP connection is only one of the many system state variables. It is also a random process state variable whose statistical properties remain the subject of much study. Although RTT is the main output state variable affecting the control, it is more realistic and informative to involve the network system as a whole in the testing of networked control methodologies, as opposed to approximating the RTT random process by some probability distribution. The reason for this is that the RTT state variable results, itself, from the highly coupled interaction of all the systems input and output state variables (Queue lengths, bandwidths, bit error rates, communication protocols, time of the day, etc ...) Moreover, independent access privileges and administration policies and the temporal randomness of IP network systems, make it impossible to reproduce the same environment conditions for experimental comparison of different networked control methodologies. For all the above reasons a need for a good network simulator arises [16].

2.2 The network simulator *ns-2*

The Network Simulator *ns* in its current version *ns-2.29* [42] is adopted as the IP networks simulator in this thesis. *ns-2* is a free open source discrete event network simulator. It is widely used in academia for researches pertaining to TCP/IP networks protocols and applications.

As one of the contributions in this thesis, two communication agents were added to *ns-2* and are listed as NCS tools on the "Networked Control Systems Repository" website [44]

- Plant agent. This can be used to simulate plants of different orders capable of receiving inputs and sending measurements via the underlying *ns-2* simulated IP network. The simulation of the plant necessitates knowledge of its transfer function and is based on solving the related differential equation for each sampling time t_{S_n}.
- Controller agent. This can be used to simulate different types of controllers capable of receiving measurements and sending control moves via the underlying *ns-2* simulated IP network.

The main advantage of using *ns-2* is providing a freely available networked control systems simulator that can be used as a common testbed for different networked control methodologies. Note that different LAN technologies (Ethernet, wireless IEEE 802.11 ...) are simulated in *ns-2*. Thus it can also be used for LAN based networked control systems.

2.3 Experimental setup

The experimental setup used throughout the thesis is shown in figure 2.1. It consists of 8 IP nodes and 8 full-duplex links. The controller agent (application) runs on node n_0. The plant is attached to node n_5. Simulated competing traffic can be generated by running different applications and agents on different nodes. Such traffic as well as the properties of each physical link, e.g. bandwidth, propagation delay, queues lengths, QoS management schemes..., can be set individually to provide different experimental network conditions creating different distributions of communication delays for packets in the network. Note that this setup is chosen because it provides the possibility of alternate routes between the controller and the plant when dynamic routing is enabled allowing measurement and control packets to arrive out of order at their destinations. Dynamic routing is enabled in the following tcl code segment that should appear in the beginning of the *ns-2* tcl script

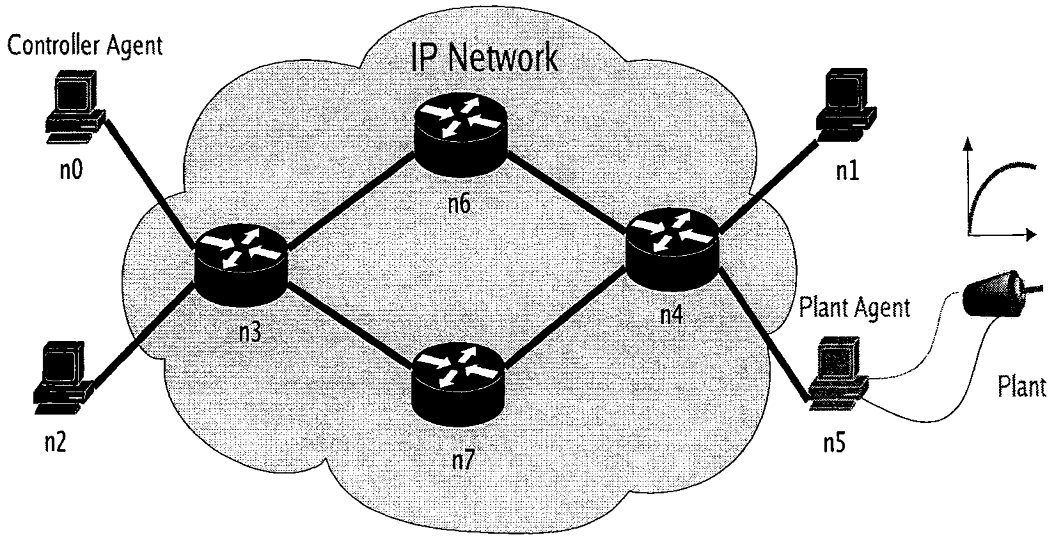


Figure 2.1: Experimental setup used in simulations.

Chapter 3

Scheduling of Control Tasks and Communication of Control Data

This chapter deals with the scheduling and communication technical aspects of SPC control via IP networks. Timing and scheduling notations are detailed. Five protocols are proposed for scheduling the control tasks, communicating control data via IP networks, and the usage of such data in the SPC predictions.

3.1 Introduction

Conventional control was aptly summarized by Liu and Layland [23] as:

"A process control computer performs one or more control and monitoring functions ... Each of the tasks must be completed before some fixed time has elapsed following the request for it. Service within this span of time must be guaranteed, categorizing the environment as 'hard-realtime' ..."

The primary difference between conventional feedback control and Networked Control Systems (NCS) is that communication between the plant and the controller is essentially deterministic in conventional control. In other words, stochastic communication delays, any data losses, and variation of sampling times have a negligible impact on the control results. Thus, conventional controllers are designed under the limiting assumption that each control move will be actuated at an *a priori* known time instant and for a known duration. Such assumptions of determinism in the timing are valid only if the controller and the plant are either directly connected (this encompasses the case when the delay, though stochastic, is negligible compared to the time constant of the plant to be controlled) or that the communication delay between them is fixed or known.

Application of conventional control methods in NCS implemented for example over the Internet, leads to unacceptable control results since the conventional control assumption of deterministic communication is violated. The main difficulties are due to packet loss caused by signal corruption, transmission collision, network buffer overflow, etc., and generally non-stationary network delays that extend over several orders of magnitude in time. As far as this thesis is concerned packet loss is equivalent to an infinite delay for the lost packet.

The way the control entities communicate affects seriously the control performance. This point was raised by previous studies [38] [26] stating that Networked Control is a multidisciplinary problem that must adapt both the control and the networking and communication methodologies. This chapter proposes different protocols that might be used to govern the control data communication and usage. This will also involve the scheduling of the control tasks since the latter is inherently a part of and affects the communication protocols among control entities.

3.2 Control tasks

In any control system there are *Three Control Tasks* to be performed

- Sensing. The plant output is sampled and sent to the controller. It is also known as the *Measurement* task.

- Control move calculation. Based on the process feedback, the controller calculates a control move and sends it to the actuator.

- Actuation. The actuator inputs a control move to the plant.

Any of the *Three Control Tasks* can be scheduled using one of the following *Two Scheduling methodologies*

- Time-Driven scheduling. A control task is periodically applied.

- Event-Driven scheduling. A task is applied if one or more events occur. This implies inter-tasks constraints.

3.3 Notation

A set of notations is needed to differentiate among variables and related times. For clarity, and without loss of generality to the extent that nonlinear plants can be locally approximated as LTI, the plant of choice to be controlled is a first order LTI plant with a unit step response (open-loop response)

$$p(t) = 1 - e^{-t} \tag{3.1}$$

The discrete normalized open-loop response $p(n)$ is obtained from the sampled unit step response with a sampling period T_s

$$p(n) = p(nT_s) = 1 - e^{-nT_s} \tag{3.2}$$

Three time variables are defined as follow

- t_{S_j}. Denotes the sensing time of the j^{th} measurement.
- t_{C_j}. Denotes the time of calculation of the j^{th} control move.
- t_{A_j}. Denotes the time of actuation of the j^{th} control move.

Notation details are illustrated in the timing diagram of Figure 3.1.

The n^{th} measurement of the plant output, sensed and relayed to the controller at time t_{s_n}, is denoted y_m^n where the subscript m is used to emphasize that it is measured. y_m^n takes τ_{SC_n} seconds (Backward delay) to get to the controller and is received by the controller at t_{c_n}. At this same time t_{c_n}, the controller calculates the control move Δu_n and sends it directly to the actuator. It takes Δu_n a period of τ_{CA_n} seconds (Forward delay) to get to the actuator which applies it directly at time t_{A_n}. It can be seen that $t_{c_n} = t_{s_n} + \tau_{SC_n}$ and $t_{A_n} = t_{C_n} + \tau_{CA_n}$.

The delay variables are highlighted here

- τ_{SC_n}. Sensor to Controller or Backward delay. The time it takes the measurement y_m^n to get to the controller.
- τ_{CA_n}. Controller to actuator or Forward delay. The time it takes the move Δu_n to get to the actuator.

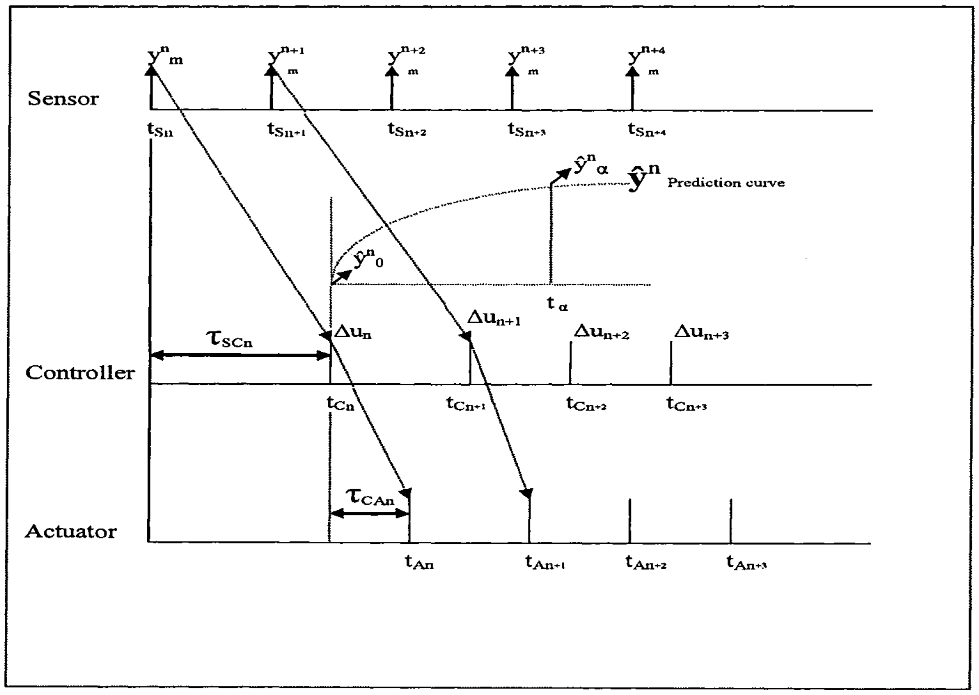


Figure 3.1: Notation and timing diagram.

The curve $\hat{y}^n$ denotes the prediction of the process performed by the SPC node at time t_{C_n} in order to calculate the move Δu_n. This curve is predicted using all of the previous moves that the controller thinks were actuated, i.e. some or all of $\Delta u_0 \cdots \Delta u_{n-1}$, depending on the data available to the controller in order to determine when and which moves were actuated. This will be explained in more detail later.

On this curve, two values are of interest to the SPC

- $\hat{y}_0^n$ which is the prediction value of the plant output state at the prediction time t_{C_n}.
- $\hat{y}_\alpha^n$ which is the prediction value, t_α seconds into the future, i.e. the prediction at time $t_{C_n} + t_\alpha$, where t_α is the time when the unit step response of the plant reaches $\beta \times 100 = (1 - \alpha) \times 100$ percent of its steady state value. Note that β, or equivalently α or t_α, is the only SPC parameter [21]. We have

$$0 \leq \alpha \leq 1.0$$

$$p(t_\alpha) = 1 - e^{-t_\alpha} = 1 - \alpha = \beta$$

Thus

$$\alpha = e^{-t_\alpha}$$

Additive correction is used for feedback correction of prediction [11]. The Additive Feedback Correction ϕ_n is the difference between a feedback value and a prediction value. The most commonly used feedback value is normally the most recent received measurement. The prediction value depends on which protocol is being used.

3.3.1 Timestamping

Timestamping is a methodology used in communication protocols to uniquely associate a certain data packet with a certain time [31]. Timestamping has also been used in NCS [12][25] as follow

- Measurement timestamping. The Measurement Packet would carry in addition to the measurement y_m^n the corresponding sensing time t_{S_n} as a timestamp to indicate when y_m^n was produced. This can be used by a predictive controller for feedback correction of the prediction.

- Control move timeStamping. The control move packet would carry, in addition to the control move Δu_n, the corresponding control calculation time t_{C_n} as a timestamp to indicate when Δu_n was produced.

- In both cases the timestamping is useful for detecting out-of-order packets.

3.3.2 Acknowledgement of reception

Acknowledgment of reception, or ACKing for short, is a methodology used in reliable communication protocols like the Transmission Control Protocol (TCP) [31]. Upon the successful reception of a packet, the destination node sends back to the source an acknowledgment of a reception (ACK). In NCS this can be used by the actuator to send back to the controller an ACK of the reception of a control move. A timestamped ACK would be useful for a predictive controller to know which moves were actuated and when they were actuated to enhance prediction. Measurement ACKing seems of no usage to the controller.

3.4 Scheduling of control tasks

3.4.1 Scheduling of the sensing task

The sensing task is the act of making a measurement of the plant output by the sensor and forwarding it to the controller. The sensor implemented in the simulations in this thesis is capable of both time-driven and event-driven sensing

- Time-driven. A measurement y_m^n of the plant output is regularly taken at sensing time t_{S_n} and sent to the SPC. Sensing times are equidistantly spaced with $t_{S_{n+1}} - t_{S_n} = T_s$ termed the sampling period.

- Event-driven. A measurement of the plant output is taken and sent every time a control move is actuated.

3.4.2 Scheduling of the control move calculation

This is the main task of the controller and consists of calculating the control move based on the latest information about the plant state, and then sending the newly calculated move to the actuator. The fact that this task depends on the plant state information favors an event-driven policy. Some Networked Control methodologies implemented a hybrid event-time-driven scheduling relying on offline simulations or buffered information whenever the up-to-date information was unavailable [24]. Along with the work presented in this thesis, a hybrid event-time-driven scheduling of SPC using an offline model was implemented but the results were not encouraging and thus omitted.

Event-driven SPC can be implemented under two strategies

- A control decision is calculated every time a new measurement arrives (adopted in this thesis).
- A control decision is calculated only when the timestamped acknowledgment of the reception of the last move sent is received (adopted in a research related to this thesis [19]).

3.4.3 Scheduling of the actuation task

The actuation task is the act of actuating or applying a received control move by the actuator. This implies that actuation can only be event-driven. Thus, each time the actuator receives a *"new"* control move it applies it.

The *"new"* move requirement demands that the actuator have a mechanism to distinguish packets carrying out-of-order moves as well as copies of packets of moves already actuated. Such packets of moves already actuated may be reproduced as a result of

- An acknowledgment timeout in the case of using control move acknowledging [19].
- Network malfunction. IP networks have been reported to reproduce unnecessary duplicates of IP datagrams [31].

3.5 SPC as an IP application

In this thesis, SPC control via IP networks is implemented as an Application layer protocol. The IP protocol suite offers two choices for the Transport Layer protocols: the User Datagram Protocol (UDP) and the Transmission Control Protocol (TCP).

The TCP header size is at least 20 bytes and this transport protocol provides

- Full-Duplex communication channel.
- Connection oriented communication. A connection must be established via the TCP three way handshake [31] before any data exchange is possible.
- Reliable transmission. TCP guarantees the transfer of the data complete and error free. This is done via the complementary roles of checksums and acknowledgments.
- Flow control. The TCP protocol adapts the data flow of a connection it manages to the condition of the network. Packet loss on a TCP connection is considered a sign of congestion which makes the TCP protocol slow down the data flow from the source, e.g. Slow Start, Sender's TCP window shrinkage ... .

On the other hand UDP uses a fixed length 8 bytes header and provides

- Simplex communication channel.
- Connectionless communication. No connection must be established. Data is sent without any guarantee that the receiver is up and willing to receive.
- Unreliable transmission. UDP has no guarantee that each datagram will be delivered free of errors. Note that the UDP header contains a field for a checksum that covers the UDP header and its data. This helps the receiver establish whether a received packet contains any errors, but it does not provide any mechanism, e.g. Acknowledgments, to solicit retransmission of the erroneous packets.

Despite all the features TCP offers, it is not suitable for control purposes for the following reasons

- Full reliability as offered by TCP is not needed by control applications in either direction. For example, although an erroneous packet should not be used by the control algorithm, it is not efficient nor stable to resend a measurement since the control should depend on the most recent data and it is better to send a new measurement instead. Thus full reliability is not useful in the Sensor→Controller direction.

  On the other hand, predictive control methods performance rely on their accuracy of prediction which is inherently dependent on knowing if and when each previous move was actuated. Thus full reliability in the SPC→Actuator direction enhances the prediction accuracy and this may improve the control performance.

- The control of data flow among the NCS entities must be governed by the control requirements and not by the underlying transport layer. When used, the TCP flow control introduces external irregularities in the scheduling of the control tasks thus adding another stochastic factor.

- Although establishing a connection before starting any control session is important, the responsibility for this task is best conferred to the control application.

Hence, UDP is the transport layer protocol of choice for control purposes in this thesis since

- it has smaller header overhead than TCP (8 bytes versus 20 bytes at least).

- it has a checksum capability to detect any corrupted data.

- functionalities, like connection establishing, ACKs, and retransmit timeouts, beyond the capability of UDP can be simply implemented in the control application itself and only where and when it's needed.

- flow control should be governed only by the control application.

Figure 3.2 shows the control application in the four-layer TCP/IP architecture and figure 3.3 shows the encapsulation of the control data in the IP datagram.

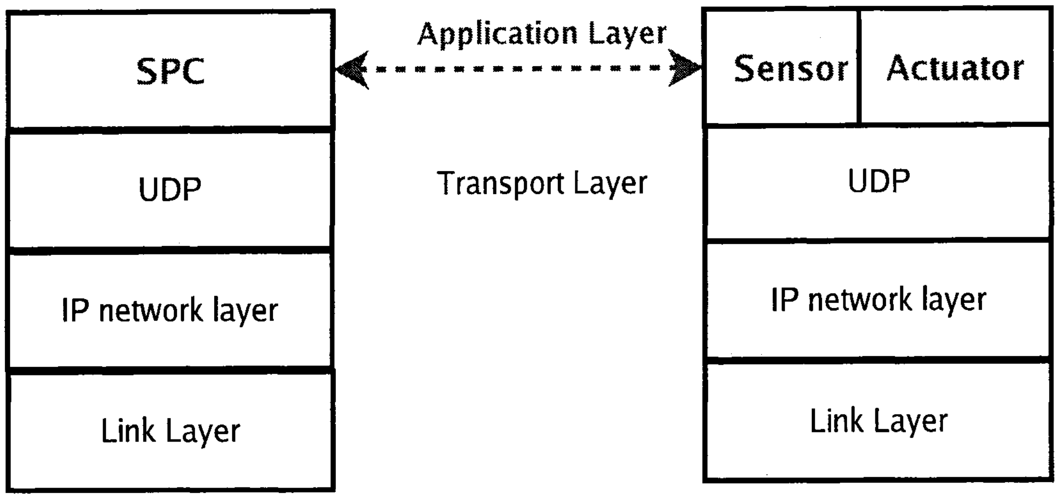


Figure 3.2: SPC as an application in the TCP/IP architecture.

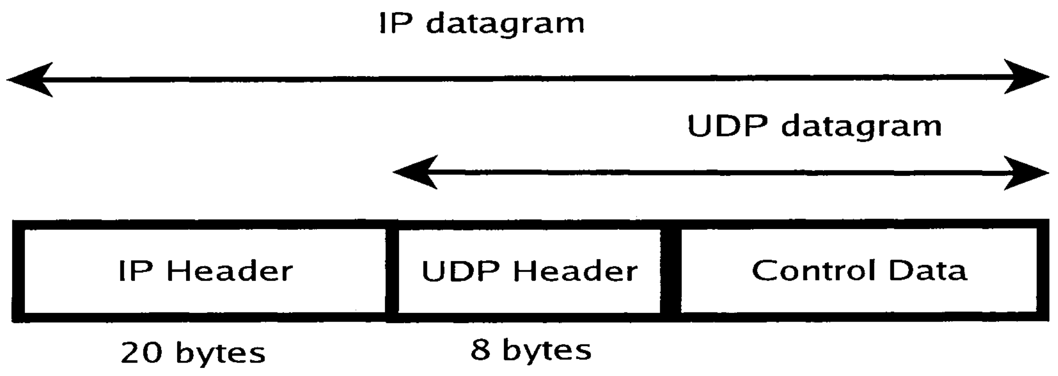


Figure 3.3: Encapsulation of the control data within the IP datagram.

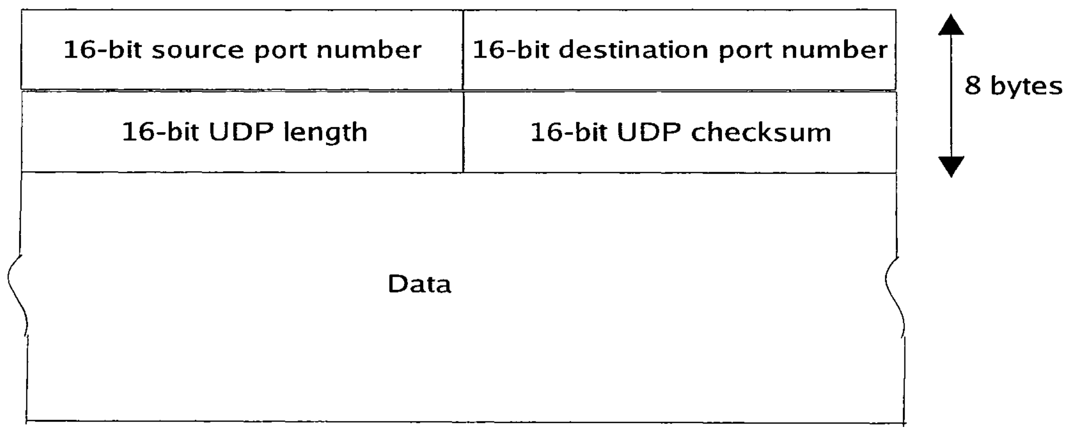


Figure 3.4: UDP header.

Figure 3.4 shows the UDP header. The UDP checksum covers the UDP header and data, and it is optional unlike TCP.

3.6 Proposed communication protocols

Five communication protocols, broken up into Series 1.x and 2.x, for the purpose of SPC control via IP are proposed and tested. The main difference among them lies with the rules governing the usage of timestamping and acknowledgment for the predictions. They are

- *Protocol 1.0.* Control moves are not acknowledged and the measurement timestamp is not used in prediction. All previous moves are assumed being applied at their calculation times.

- *Protocol 1.1.* Control moves are not acknowledged but the measurement timestamp is used in prediction. All previous moves are assumed being applied at their calculation times.

- *Protocol 2.0.* Control moves are acknowledged and the measurement timestamp is not used in prediction. Only ACKed moves are used in prediction while unACKed moves are ignored.

- *Protocol 2.1.* Control moves are acknowledged and the measurement timestamp is not used in prediction. Both ACKed and unACKed moves are used in prediction.
- *Protocol 2.2.* Same as *Protocol 2.1* except that the measurement timestamp is used in prediction. It is a combination of *Protocols 1.1* and *2.1.*

Different sensor scheduling schemes can be implemented with each protocol. The protocols will be detailed in the next sections.

Some common notation is introduced here

- $L\tau_{CA}$. A variable kept on the plant side to denote the last measured τ_{CA} by the plant node.
- $L\tau_{CA_c}$. A variable kept on the controller side to denote $L\tau_{CA}$ sent by the plant node.
- $L\tau_{SC}$. A variable kept on the controller side to denote the last measured τ_{SC} by the controller node.
- St_p. Start time of the control session at the plant node side to be used as reference for later timestamps generated at the plant side.
- St_c. Start time of the control session at the controller node side to be used as reference for later timestamps generated at the controller side.
- Lt_S. A variable kept on the controller side to denote the sensing time (t_S)of the last in-order measurement received, relatively to St_p.
- Lt_C. Kept by the plant node to indicate the generation time (t_C) of the last actuated move, relatively to St_c.
- Lt_A. A variable kept on the plant side to denote the actuation time, relative to St_p, of the last actuated move.
- ct_p. To indicate the current time at the plant side.
- ct_c. To indicate the current time at the controller side.

3.7 Common functionalities and common basic packets format

This section describes the common functionalities among all five proposed protocols. It also introduces the basic formats of the packets common to all protocols. If a certain protocol requires additional fields to be added to the basic packets, the resulting packets will be described in the next sections detailing each protocol separately.

3.7.1 Common functionalities

Some functionalities shared by the proposed protocols are presented here

- When the control session is being established initially, the SPC and the plant nodes exchange temporal reference points (St_p and St_c) to be used as references for the timestamps used in the protocols. Moreover, they both agree on a session identification number. This is useful to detect packets from previous control sessions still lingering in the network.
- Measurements and control moves timestamping is used in all the protocols to detect out-of-order and duplicate packets. Thus
  - The controller drops any received measurement that has a timestamp smaller than that of the last measurement received (Lt_S).
  - The actuator drops any received control move that has a timestamp smaller than that of the last received move (Lt_C).

  An important point to make here is that ***whether the measurement timestamp is used in prediction or not depends on the protocol.***
- The actuator is event-driven. Upon reception, a move is actuated only if it has a newer timestamp than that of the last received move (Lt_C). Otherwise the move is considered as out-of-order and is dropped.

3.7.2 Common basic measurement packet format

Figure 3.5 shows the **basic fields** contained in the Measurement Packet sent by the sensor for all the proposed protocols. Additional fields pertaining to control moves

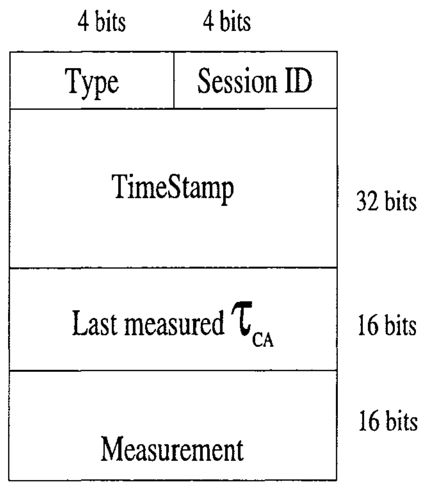


Figure 3.5: Measurement Packet format showing the basic fields common to all protocols.

acknowledgement will be described in the sections related to to the protocols that use control moves acknowledgements, namely *2.x* series protocols.

A Measurement Packet contains the following basic fields

- *Type.* This 4-bit field is common to all packets and indicates the type of the packet. It can accomodate 16 types. Only 3 types are used so far
  - 0. Indicates a pure Measurement Packet.
  - 1. Indicates a control packet.
  - 2. Indicates a compound Measurement Packet carrying an acknowledgement to a control move. This type appears only in the *2.x* series protocols.
  - The rest of the values up to 15 can be used to indicate different signaling packets for negotiating the control session, the time origins, plant resetting etc ...
- *Session ID.* This 4- bit field indicates the ID of the current control session. The importance of this field was outlined earlier.
- *TimeStamp.* This 32-bit field indicates the time at which the measurement was taken and sent, in reference to the start of the control session at the plant end.

It is calculated by the plant node when performing a sensing task as follow

$$T\text{imeStamp} = ct_p - St_p$$

Note that this field corresponds to t_{S_n}. The main usage of this field by all protocols is for the controller to detect out-of-order Measurement Packets. But whether this field will be used by the controller to calculate the Additive Feedback Correction or not depends on the protocol in use and will be detailed in later sections.

It contains two subfields.

- The first 16-bit word indicates the integer value. This yields a time span of approximately 18 hours.
- The second 16-bit word indicate the fraction value. This can give a precision of up to $\frac{1}{2^{16}} \approx 15micro$.

This accuracy is generally more than enough for many applications.

- *Last measured* τ_{CA}. This is an *optional* field. It is the last measured delay from the controller to the actuator. It is used when the controller node needs to estimate the RTT. It is set to the delay $L\tau_{CA}$ available at the time the Measurement Packet is being produced. The calculation and usage of this field will be detailed in Section 7.4.

- *Measurement.* This field holds the measurement y_m^n of the controlled process output taken and sent at *TimeStamp.* This is the feedback from the plant to be used for control decisions.

3.7.3 Common control move packet format

Figure 3.6 shows the control move packet common to all protocols.

The control move packet contains the following fields

- *Type.* It has a value 1 to indicate a control move packet.

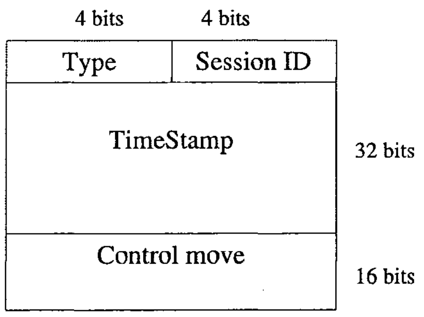


Figure 3.6: Control Move packet common to all protocols.

- *Session ID.* Indicates the current control session ID agreed upon between the plant and the SPC and it must be the same as the one in the Measurement Packet.

- *TimeStamp.* This 32-bit field indicates the date at which the control move was calculated and sent in reference to St_c.

$$T\text{imeStamp} = ct_c - St_c$$

  It has the same subfields and corresponding accuracies as the Measurement Packet. This field corresponds to t_{C_n} and is used by the plant node only to detect out-of-order moves. Such moves should be discarded without being actuated.

- *Control Move.* This is the control move value Δu_n.

3.8 *Protocol 1.0*

The characteristics of this protocol are

- The event-driven SPC calculates a new move every time it receives a new in-order measurement.

- Move ACKing is not used. All sent moves are assumed to have been actuated at the time of calculation and hence are used in the prediction. The application time of a move Δu_j is therefore assumed to be its calculation time t_{C_j}.

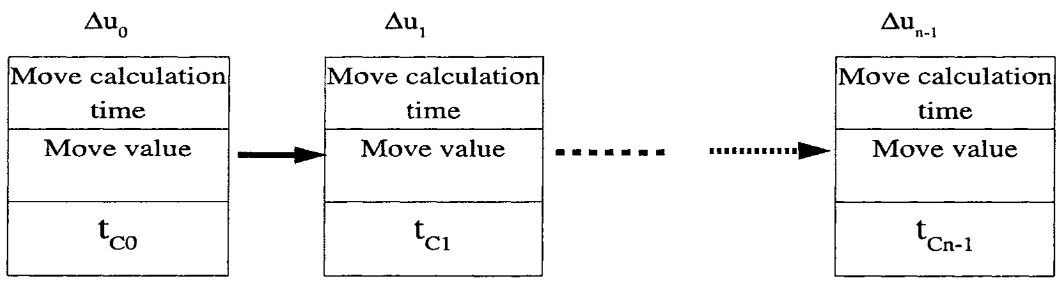


Figure 3.7: *Protocol 1.0.* Linked list of all the moves calculated and sent.

- The Measurement TimeStamping is not used in calculating the Additive Feedback Correction.

- The sensor can be either time-driven or event-driven. But in the case of an event-driven sensor any lost packet in either direction can lead to deadlock and breaking of the communication cycle between controller and plant. Thus, for this protocol it is best to use a time-driven sensor.

Protocol 1.0 measurement and control move packets are the basic packets shown resepctively in figures 3.5 and 3.6. It can be seen that *Protocol 1.0* is equivalent to closing the control loop of a conventional SPC via an IP network without changing anything in the SPC algorithm except for the out-of-order data detection.

The controller keeps a linked list of all the moves calculated and sent. The list is shown in Figure 3.7. Every time a move Δ_{u_i} is calculated by the controller, a node carrying the current time (ct_c) which corresponds to t_{C_i}, and the value of the control move (Δu_i) is added to the moves list.

Figure 3.8 shows the flow chart of the receive function of the controller node when it receives a Measurement Packet. If the received Measurement Packet *Session ID* does not match the current session or the measurement is out-of-order, this packet is discarded. Otherwise, the SPC uses the *Measurement* field to calculate the new move. In order to estimate the RTT, it measures the delay τ_{SC} as follow

$$L\tau_{SC} = ct_c - (St_p + TimeStamp) \tag{3.3}$$

and it updates its $L\tau_{CA_c}$ to the Measurement Packet *Last measured* τ_{CA}. Now the RTT can be estimated to be

$$RTT = L\tau_{CA_c} + L\tau_{SC} \tag{3.4}$$

Equation 3.3 necessitates that the plant and the controller nodes have synchronized clocks ($St_p \equiv St_c$). Many algorithms and implementations are used for clock synchronization of different nodes via the Internet. The Newtork Time Protocol is mostly used. Its latest version (NTP v4) is described in [43] and can provide an accuracy to within 10 *millisec* over the Internet and 200 *microsec* over LANs.

The controller node also updates Lt_s to the *TimeStamp* field value.

Figure 3.9 shows the flow chart of the receive function of the plant node when it receives a Control Move Packet. If the received control packet *Session ID* does not match the current session or the measurement is out-of-order, this packet is discarded. Otherwise, the actuator actuates a change in the plant input equal to the *Control move* field value.

The plant node updates its estimation of $L\tau_{CA}$

$$L\tau_{CA} = ct_p - (St_c + TimeStamp)$$

Again, the last equation necessitates that the plant and the controller nodes must have synchronized clocks ($ct_c \equiv ct_p$ and $St_c \equiv St_c$). The plant node updates Lt_C to the *TimeStamp* field value so that any control packet received with a *TimeStamp* field that is less than Lt_C, is considered out-of-order.

3.9 *Protocol 1.1*

This protocol duplicates *Protocol 1.0* except that the SPC uses the Measurement TimeStamp in the Additive Feedback Correction ϕ_n. Thus the measurement and control packet formats, as in the case of *Protocol 1.0*, are the basic packets shown respectively in figures 3.5 and 3.6. The SPC receive and plant receive functions flowcharts are the same as those of *Protocol 1.0* shown respectively in figures 3.8 and 3.9. The change is actually only in how the SPC algorithm calculates the Additive Feedback Correction ϕ_n and this is detailed in the next chapter.

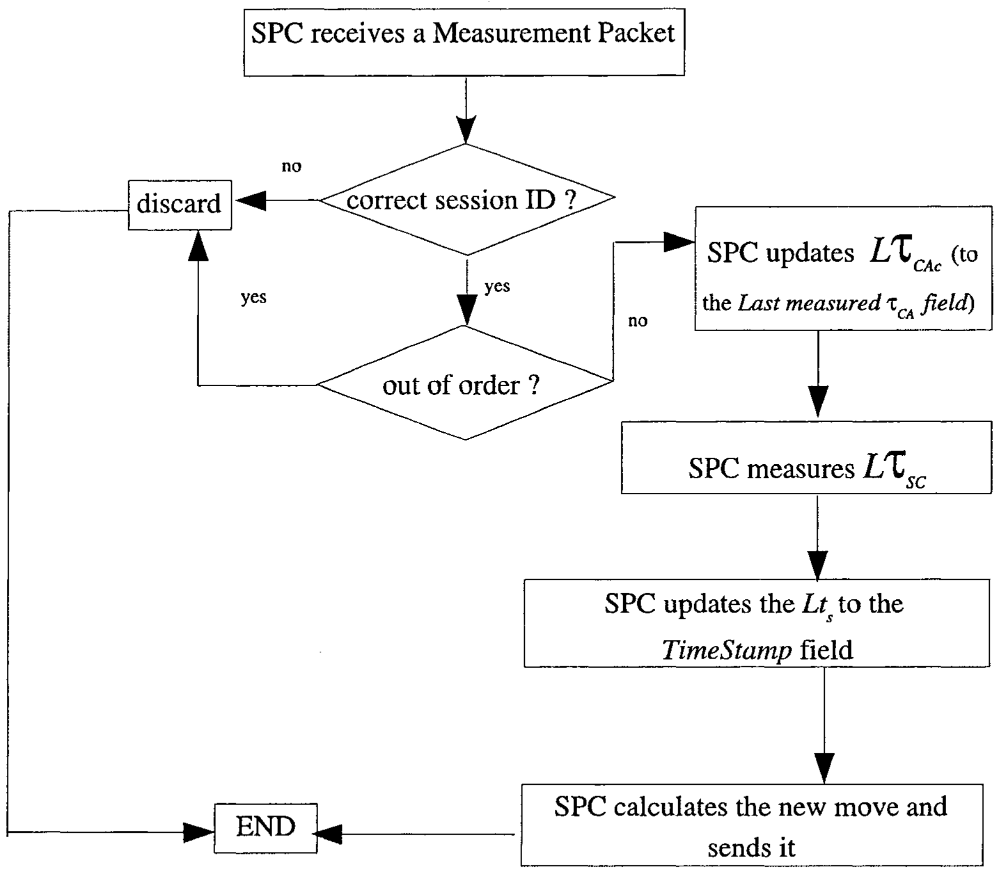

SPC receives a Measurement Packet
correct session ID ?
no
discard
yes
out of order ?
yes
no
SPC updates $L\tau_{CAc}$ (to the *Last measured* τ_{CA} *field*)
SPC measures $L\tau_{SC}$
SPC updates the Lt_s to the *TimeStamp* field
SPC calculates the new move and sends it
END

Figure 3.8: *Protocol 1.0.* Flow Chart of the SPC receive function.

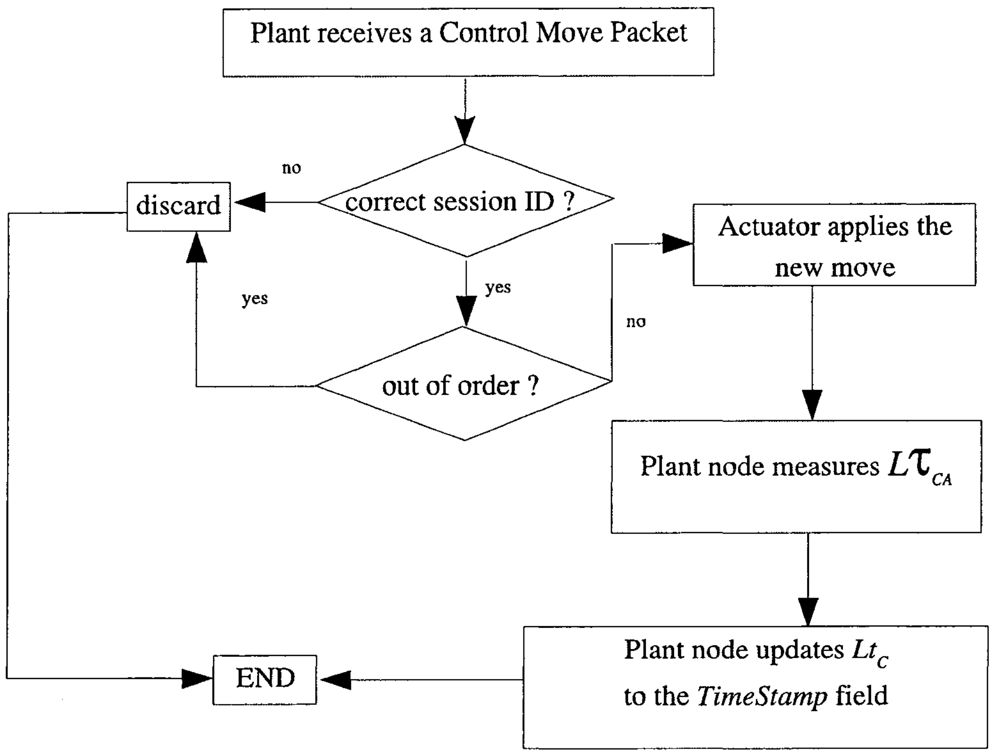

Plant receives a Control Move Packet
correct session ID ?
no
discard
yes
yes
out of order ?
no
Actuator applies the new move
Plant node measures $L\tau_{CA}$
Plant node updates Lt_C to the *TimeStamp* field
END

Figure 3.9: *Protocol 1.0.* Flow Chart of the plant receive function.

3.10 *Protocol 2.0*

The characteristics of this protocol are

- The event-driven SPC calculates a new move every time it receives a new in-order measurement.
- Control Moves are acknowledged by the actuator. To each Measurement Packet sent, an acknowledgement for the most recent received move is added. This is known as *Acknowledgement Piggybacking.* Thus the application time t_{A_j} is known by the SPC for each ACKed move.
- At the time t_{C_n} of calculating a new move Δu_n, only the previous moves that were ACKed are considered having been actuated and are used in the prediction convolutions. Previous unACKed moves are omitted from the prediction convolutions.
- No timeout is utilized along with the acknowledgement policy. There is no point resending a move after a certain timeout especially in the cases where the RTT is larger or equal to the process sampling period T_s, since a newer move is likely to have already been calculated.
- The sensor can be either time-driven or event-driven. But in the case of event-driven sensor any lost packet in either direction can lead to deadlock and breaking of the communication cycle between controller and plant. Thus for this protocol it is recommended to have a time-driven sensor.

3.10.1 *Protocol 2.0.* Measurement packet format

If no acknowledgements are available to be sent then the Measurement Packet format is just the basic one shown in figure 3.5. This only happens at the start of the control session. But when a move acknowledgement needs to be sent, it is sent piggybacking on top of the Measurement Packet by means of some additional fields which results in a Compound Measurement Packet. Figure 3.10 shows the format of the Compound Measurement Packets that carry piggybacking acknowledgements. A value of 2 in

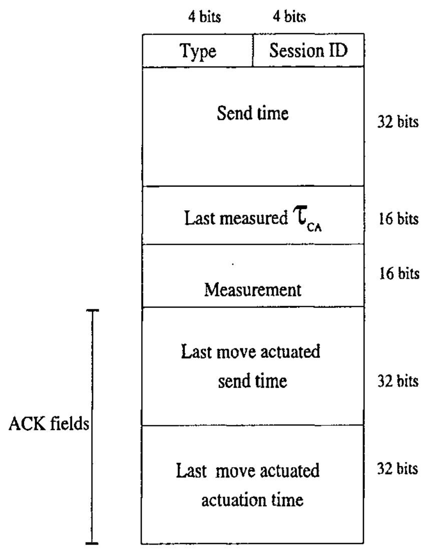


Figure 3.10: *Protocol 2.0.* Compound Measurement Packet carrying a piggybacking move ACK.

the *Type* field indicates a compound Measurement Packet. The fields related to acknowledgement are

- *Last move actuated send time.* This field indicates the time at which the last move actuated was calculated at the controller side relative to St_c. The plant node has this time recorded in the variable Lt_C.

- *Last move actuated actuation time.* This field indicates the time at which the last move was actuated in reference to St_p. The plant node has this time recorded in the variable Lt_A.

The controller keeps a linked list of all the moves calculated and sent. The list is shown in figure 3.11. Every time a move Δ_{u_i} is calculated by the controller, a node carrying the current time (ct_c), the value of the control move (Δu_i) is added to the moves list. The field *Move actuation time* is left blank. The *ACKed flag* is set to 0. When the SPC receives a Compound Measurement Packet carrying an ACK for a move Δu_i, the corresponding node in the moves list is updated as follow

- The *ACKed flag* is set to 1.

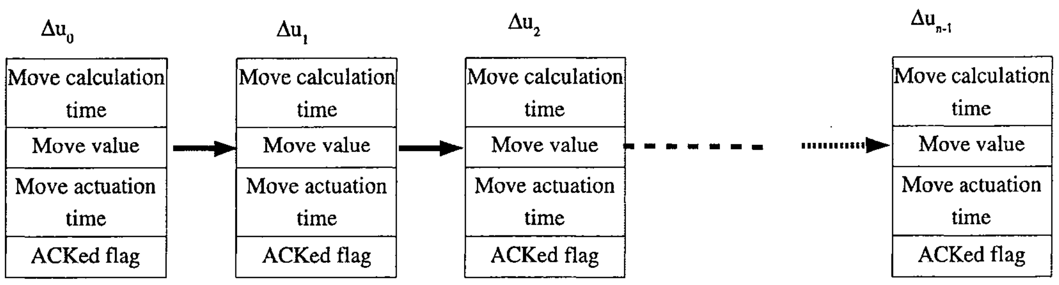


Figure 3.11: *Protocol 2.0.* Linked list of all the moves calculated and sent.

- The *Move actuation time* is set to the actuation time t_{A_j} that arrived in the Compound Measurement Packet.

Figure 3.12 shows the flow chart of the receive function of the controller node when it receives a Measurement or Compound Measurement Packet. If the received Measurement Packet *Session ID* field does not match the current session this packet is discarded.

Otherwise, the SPC looks in the moves list for the move that was calculated at the time indicated in the field *Last move actuated send time* and updates its actuation time using the value in the field *Last move actuated actuation time* if it was not already updated (the *ACKed flag* would be 1 if the move was previously acknowledged). After this step the packet is treated the same way as in protocol 1 starting with the process of checking if it is out-of-order.

3.10.2 *Protocol 2.0.* Control move packet format

Protocol 2.0 Control Move Packet has the same format as that of *Protocols 1.0,1.1* which is the basic format shown 3.6. It is also treated the same way by the plant receive method regarding out of order packets, times and delay updates ... The difference in *Protocol 2.0* is that for the plant node, with each measurement sent, the last move actuated is acknowledged via piggybacking. Thus the plant receive function flowchart is the same as that of *Protocols 1.0, 1.1* shown in Figure 3.9.

For the same reasons, as in the case of *Protocols 1.0, 1.1*, *Protocol 2.0* necessitates that the plant and the controller nodes have synchronized clocks.

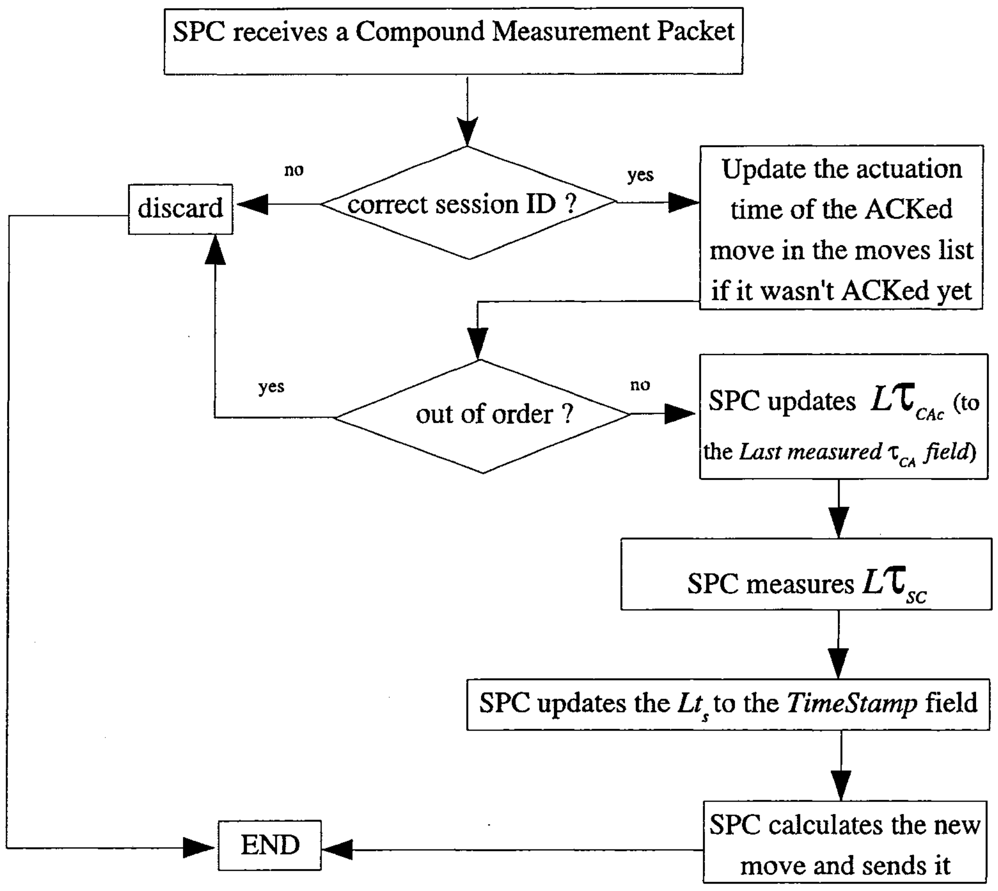

SPC receives a Compound Measurement Packet
correct session ID ?
no
yes
discard
Update the actuation time of the ACKed move in the moves list if it wasn't ACKed yet
out of order ?
yes
no
SPC updates $L\tau_{CAc}$ (to the *Last measured* τ_{CA} *field*)
SPC measures $L\tau_{SC}$
SPC updates the Lt_s to the *TimeStamp* field
SPC calculates the new move and sends it
END

Figure 3.12: *Protocol 2.* Flow Chart of the SPC receive function.

3.11 *Protocol 2.1*

This protocol is the same as *Protocol 2.0* except that in this protocol all previous moves are considered actuated including the unACKed ones. Thus, in this protocol all the previous moves are used in the prediction convolution as follows

- For ACKed moves the actuation times are known via acknowledgements.
- For unACKed moves the actuation time is assumed to be the move calculation time t_{C_n}.

Here also the controller keeps a linked list of all the moves calculated and sent. The list is the same as that of *Protocol 2.0* shown in Figure 3.11. Every time a move Δ_{u_i} is calculated by the controller, a node carrying the current time (ct_c), the value of the control move (Δu_i) is added to the moves list. The field *Move actuation time* is set to ct_c. Note that ct_c corresponds to the control move calculation time ,t_{C_i} , of the move Δu_i. The *ACKed flag* is set to 0.
When the SPC receives a Compound Measurement Packet carrying an ACK for a move Δu_i, the corresponding node in the moves list is updated as follow

- The *ACKed flag* is set to 1.
- In the *Move actuation time* field, the real actuation time t_{A_j} that arrived in the Compound Measurement Packet replaces t_{C_i}.

The SPC receive and plant receive functions flowcharts are the same as those of *Protocol 2.0* shown respectively in figures 3.12 and 3.9.

3.12 *Protocol 2.2*

This protocol combines the functionalities of both *Protocols 1.1* and *2.1*

- Moves are ACKed. All previous moves are considered as actuated including unACKed ones. Thus all the previous moves are used in the prediction convolution as follows
  - For ACKed moves the actuation times are known via acknowledgements.

  - For unACKed moves the actuation time is assumed to be the move calculation time t_{C_n}.

- The Measurement TimeStamp is used to calculate the Additive Feedback Correction ϕ_n.

It can be easily seen that this protocol is the same as *Protocol 2.1* regarding the scheduling and communication technicalities. Thus

- It keeps the same Control Moves linked list as in *Protocol 2.1* shown in Figure 3.11. This list is also built and updated in the same way as in *Protocol 2.1.*

- The SPC receive and plant receive functions flowcharts are the same as those of *Protocols 2* and *2.1* shown respectively in figures 3.12 and 3.9.

The only difference from *Protocol 2.1* is in calculating ϕ_n and the effect of this change ~~will become clearer in the next chapter.~~

Chapter 4

Analytical Study of the Suggested Communication Protocols

4.1 Introduction

In this chapter the performances of each suggested communication protocol from Chapter 3 are studied analytically.

4.2 Analytical study of SPC characteristics under communication delay

This section extends the continuous analytical study of conventional SPC published in [21] to probe SPC behavior under constant delay without packet loss. Although IP networks communication is subject to random delays and packet loss, analytical studies have leaned towards the study of controllers subjected to constant delay without packet loss [39] for two main reasons:

- Analytical study of controllers under random delays yields stochastic Differential Equations that are in most cases impossible to solve and are themselves based, in the first place, on constrained assumptions that veer from reality.

- Analytical studies with constant delays are easier to address and at the same time yield fair insight into the controller's behavior under random delays. This insight does help in proposing a control approach for random delays that are verified via simulation.

- Introducing packet loss into the control formulations is difficult. It is practical to consider packet loss as an infinite delay contribution to the control formulations.

Thus it is assumed that both forward and backward delays are constant and that $\tau_{SC_n} = \tau_{CA_n} = d$ seconds where d is a *Real* constant. This results in the following

$$t_{S_n} = n\ T_s \tag{4.1}$$

$$t_{A_n} = t_{S_n} + 2d = t_{C_n} + d \tag{4.2}$$

$$\Delta t_{A_n} = t_{A_{n+1}} - t_{A_n} = T_s \tag{4.3}$$

Hence, independent of the value of d, each move Δu_n lasts for a constant period $\Delta t_{A_n} = t_{A_{n+1}} - t_{A_n} = T_s$.

4.3 *Protocol 1.0.* Continuous approximation of SPC behavior under constant communication delay

A continuous approximation to SPC, subjected to constant communication delays and using *Protocol 1.0* for control data communication, is developed in this section.

Recall that *Protocol 1.0* features are

- No measurement timestamping or acknowledgements of timing of implementation of moves.
- The SPC controller assumes that all previous moves Δu_j had been applied at their corresponding calculation times t_{C_j}.

It can be seen that SPC control using *Protocol 1.0* involves the application of conventional SPC in an NCS without alteration of the SPC features.

At time t_{C_n} the SPC node, before it calculates Δu_n, uses the feedback y_m^n to adjust the prediction curve. Therefore, let the Additive Feedback Correction, ϕ_n, be the difference between the expected value $\hat{y}_0^n$ and the measured value y_m^n then

$$\phi_n = \hat{y}_0^n - y_m^n \tag{4.4}$$

The prediction value at t_α, $\hat{y}_\alpha^n$ is adjusted by $(-\phi_n)$ to give

$$\hat{y}_\alpha^n = \hat{y}_\alpha^n - \phi_n \tag{4.5}$$

4.3.1 *Protocol 1.0* The actual value of y_m^n

The measured value of the process, y_m^n, received at t_{C_n} results from all of the moves that were applied before t_{s_n}. And those would be all the moves Δu_j with corresponding application times $t_{A_j} \leq t_{S_n}$. Thus

$$jT_s \leq nT_s - 2d$$

Assuming an LTI plant, y_m^n is then

$$y_m^n = \sum_{j=0}^{jT_s \leq nT_s - 2d} \Delta u_j \; p(t_{S_n} - t_{A_j}) \tag{4.6}$$

4.3.2 *Protocol 1.0* Calculation of the control move Δu_n

Assume for now that the SPC node does not know the application times t_{A_j} of moves already calculated and sent. The SPC assumes that all those previous moves were applied and that the application time of a previous move Δu_j , calculated and sent at t_{C_j}, is t_{C_j}. As a result, the prediction curve $\hat{y}^n$ involves all previous moves. Thus

$$\hat{y}_0^n = \sum_{j=0}^{n-1} \Delta u_j \; p(t_{C_n} - t_{C_j}) \tag{4.7}$$

and $\hat{y}_\alpha^n$

$$\hat{y}_\alpha^n = \sum_{j=0}^{n-1} \Delta u_j \; p(t_{C_n} + t_\alpha - t_{C_j}) \tag{4.8}$$

Applying the Additive Feedback Correction ϕ_n, Δu_n is

$$\Delta u_n = \frac{1 - [\hat{y}_\alpha^n - \phi_n]}{\beta} \tag{4.9}$$

where the reference set-point is taken to be 1 for simplicity and this is without any loss of generality.

Recalling that $\beta = 1 - \alpha$, equation 4.9 is equivalent to

$$\beta \, \Delta u_n = 1 - \sum_{j=0}^{n-1} \Delta u_j \, p(t_{C_n} + t_\alpha - t_{C_j}) + \sum_{j=0}^{n-1} \Delta u_j \, p(t_{C_n} - t_{C_j}) - \sum_{j=0}^{jT_s \leq nT_s - 2d} \Delta u_j \, p(t_{S_n} - t_{A_j}) \tag{4.10}$$

Let $t = n\ T_s$, $z = j\ T_s$ thus $\Delta t = \Delta z = T_s$. Thus $t_{C_n} = t + d$, $t_{C_j} = z + d$. An alternative form of equation 4.10 is

$$\begin{aligned}
\beta\ \Delta t \frac{\Delta u_n}{\Delta t} = 1 \quad & - \sum_{j=0}^{n-1} \frac{\Delta u_j}{\Delta t} p((n-j)T_s + t_\alpha)\Delta t \\
& + \sum_{j=0}^{n-1} \frac{\Delta u_j}{\Delta t} p((n-j)T_s)\Delta t \\
& - \sum_{j=0}^{jT_s \le nT_s - 2d} \frac{\Delta u_j}{\Delta t} p((n-j)T_s - 2d)\Delta t
\end{aligned} \tag{4.11}$$

In the limit $\Delta t = T_s \to 0$ a continuous form of equation 4.11 is

$$\begin{aligned}
\beta\ \Delta t \frac{du(t)}{dt} = 1 \quad & - \int_0^t \frac{du(z)}{dz} p(t + t_\alpha - z)dz \\
& + \int_0^t \frac{du(z)}{dz} p(t - z)dz \\
& - \int_0^{t-2d} \frac{du(z)}{dz} p(t - 2d - z)dz
\end{aligned}$$

$$\begin{aligned}
&= 1 + \int_0^t \frac{du(z)}{dz} [p(t-z) - p(t + t_\alpha - z)]dz - \int_0^{t-2d} \frac{du(z)}{dz} p(t - 2d - z)dz \\
&= 1 + \int_0^t \frac{du(z)}{dz} [e^{-(t+t_\alpha - z)} - e^{-(t-z)}]dz - \int_0^{t-2d} \frac{du(z)}{dz} [1 - e^{-(t-2d-z)}]dz
\end{aligned} \tag{4.12}$$

Letting $q = \beta\ \Delta t$, and noticing that $\alpha = e^{-t_\alpha}$, 4.12 becomes

$$q\ \dot{u} = 1 - u(t-2d) - \beta \int_0^t \dot{u}(z) e^{-(t-z)} dz + \int_0^{t-2d} \dot{u}(z) e^{-(t-2d-z)} dz \tag{4.13}$$

Let

$$I(t) = \int_0^t \dot{u}(z) e^{-(t-z)} dz$$

4.13 becomes

$$q\ \dot{u} = 1 - u(t-2d) - \beta I(t) + I(t-2d) \tag{4.14}$$

Noticing that

$$I(t) = e^{-t}\int_0^t \dot{u}(z)e^z dz$$

Differentiating $I(t)$ gives

$$\dot{I}(t) = \left[e^{-t}\right]'\int_0^t \dot{u}(z)e^z dz + e^{-t}\left[\int_0^t \dot{u}(z)e^z dz\right]' = -I(t) + \dot{u}(t)$$

It is easy to verify

$$\dot{I}(t-2d) = -I(t-2d) + \dot{u}(t-2d)$$

Further differentiating 4.14 gives

$$q\,\ddot{u} = -\dot{u}(t-2d) - \beta(-I(t) + \dot{u}(t)) - I(t-2d) + \dot{u}(t-2d) \tag{4.15}$$

Eliminating $I(t)$, $I(t-2d)$ and its derivatives from 4.14 and 4.15 yield the LTI delay differential equation (DDE) in $u(t)$ as

$$q\,\ddot{u}(t) + (\beta + q)\dot{u}(t) = 1 - u(t-2d) \tag{4.16}$$

The effect of the delay d is now apparent on the right-hand-side of the DDE.

4.4 ***Protocol 1.1.*** Continuous approximation of SPC behavior under constant communication delay

Protocol 1.1 is similar to *Protocol 1* except for the addition of Measurement Timestamping.

Recall the *Protocol 1.1* features

- Measurement Timestamping is used. Thus the SPC knows t_{S_j} for each measurement y_m^j.
- As in *Protocol 1.0* no move acknowledgements are used.
- As in *Protocol 1.0*, the SPC controller assumes that all previous moves Δu_j had been applied at their corresponding calculation times t_{C_j}.

Since the measurement time, t_{S_n}, is known, the Additive Feedback Correction, ϕ_n, can be the difference between the measured value y_m^n and the estimation, by the controller, of the past plant state at t_{S_n}. In other words, let

$$\phi_n = \hat{y}_m^n - y_m^n \tag{4.17}$$

where $\hat{y}_m^n$ is the state that the controller thinks the plant output had at t_{S_n}. Then

$$\hat{y}_m^n = \sum_{j=0}^{jT_s \le nT_s - d} \Delta u_j \ \ p(t_{S_n} - t_{C_j}) \tag{4.18}$$

The prediction value at t_α, $\hat{y}_\alpha^n$ is adjusted to

$$\hat{y}_{\alpha c}^n = \hat{y}_\alpha^n - \phi_n \tag{4.19}$$

As in all protocols, the measurement y_m^n value is given by equation 4.6.

4.4.1 *Protocol 1.1.*Calculation of the control move Δu_n

The prediction of the process state, $\hat{y}_\alpha^n$, still follows

$$\hat{y}_\alpha^n = \sum_{j=0}^{n-1} \Delta u_j \ p(t_{C_n} + t_\alpha - t_{C_j}) \tag{4.20}$$

Now, we have

$$\beta \ \Delta u_n = 1 - (\hat{y}_\alpha^n - \phi_n)$$

thus

$$\beta \ \Delta u_n = 1 - \sum_{j=0}^{n-1} \Delta u_j \ p(t_{C_n} + t_\alpha - t_{C_j}) + \sum_{j=0}^{jT_s \le nT_s - d} \Delta u_j \ p(t_{S_n} - t_{C_j}) - \sum_{j=0}^{jT_s \le nT_s - 2d} \Delta u_j \ p(t_{S_n} - t_{A_j}) \tag{4.21}$$

and equivalently

$$\begin{aligned} \beta \ \Delta u_n = \quad & 1 - \sum_{j=0}^{n-1} \Delta u_j \ p((n-j)Ts + t_\alpha) + \sum_{j=0}^{jT_s \le nT_s - d} \Delta u_j \ p((n-j)Ts - d) \\ & - \sum_{j=0}^{jT_s \le nT_s - 2d} \Delta u_j \ p((n-j)Ts - 2d) \end{aligned} \tag{4.22}$$

As in *Protocol 1.0*, let $t = n\ T_s$, $z = j\ T_s$ thus $\Delta t = \Delta z = T_s$. $t_{C_n} = t + d$, $t_{C_j} = z + d$. Also let $q = \beta\ \Delta t$.

In the limit $\Delta t = T_s \rightarrow 0$ the continuous form of equation 4.22 would be

$$\begin{aligned} q\ \frac{du(t)}{dt} = 1 \quad & - \int_0^t \frac{du(z)}{dz} p(t + t_\alpha - z) dz \\ & + \int_0^{t-d} \frac{du(z)}{dz} p(t - z - d) dz \\ & - \int_0^{t-2d} \frac{du(z)}{dz} p(t - z - 2d) dz \end{aligned}$$

Proceeding as in the case of *Protocol 1.0* yields the LTI DDE in $u(t)$

$$q\ \ddot{u}(t) + (\beta + q)\dot{u}(t) + u(t) = 1 + u(t - d) - u(t - 2d) \tag{4.23}$$

Again, the effect of the delay d is seen on the right-hand-side as occurred with Protocol 1.0 but now there are two terms due to Measurement Timestamping.

4.5 *Protocol 2.0.* Continuous approximation of SPC behavior under constant communication delay

A continuous approximation of SPC response, subjected to constant communication delays and using *Protocol 2.0* for control data communication, is developed in this section. *Protocol 2* features are highlighted here

- No Measurement Timestamping is used.
- The actuator sends back Move Acknowledgements, or ACKs, for each control move it applies.
- The SPC controller assumes that only the previous control moves that were ACKed were applied. In its move predictions, it excludes all the other previous moves that were not ACKed.

The Additive Feedback Correction, ϕ_n, is the same as that in *Protocol 1.0* where the difference between the expected value $\hat{y}_0^n$ and the measured value y_m^n is

$$\phi_n = \hat{y}_0^n - y_m^n \tag{4.24}$$

4.5.1 *Protocol 2.0* Calculation of Δu_n

The SPC assumes that only moves that were ACKed were actually applied. So the SPC knows that the application time of an ACKed move Δu_j is t_{A_j}. As a result, the prediction curve $\hat{y}^n$ involves only previous ACKed moves. Such moves would be the moves that have their application times

$$t_{A_j} \leq t_{S_n}$$

or

$$jT_s + 2d \leq nT_s$$

Hence,

$$\hat{y}_0^n = \sum_{j=0}^{jT_s \leq nT_s - 2d} \Delta u_j \; p(t_{C_n} - t_{A_j}) \tag{4.25}$$

and the predictions are then $\hat{y}_\alpha^n$ is

$$\hat{y}_\alpha^n = \sum_{j=0}^{jT_s \leq nT_s - 2d} \Delta u_j \; p(t_{C_n} + t_\alpha - t_{A_j}) \tag{4.26}$$

We have the move as

$$\beta \; \Delta u_n = 1 - (\hat{y}_\alpha^n - \phi_n) = 1 - \hat{y}_\alpha^n + \hat{y}_0^n - y_m^n$$

and thus

$$\begin{aligned} \beta \; \Delta u_n = \quad & 1 - \sum_{j=0}^{jT_s \leq nT_s - 2d} \Delta u_j \; p(t_{C_n} + t_\alpha - t_{A_j}) + \sum_{j=0}^{jT_s \leq nT_s - 2d} \Delta u_j \; p(t_{C_n} - t_{A_j}) \\ & - \sum_{j=0}^{jT_s \leq nT_s - 2d} \Delta u_j \; p((n-j)Ts - 2d) \end{aligned} \tag{4.27}$$

or equivalently

$$\begin{aligned} \beta \; \Delta u_n = \quad & 1 - \sum_{j=0}^{jT_s \leq nT_s - 2d} \Delta u_j \; p((n-j)Ts + t_\alpha - d) + \sum_{j=0}^{jT_s \leq nT_s - 2d} \Delta u_j \; p((n-j)Ts - d) \\ & - \sum_{j=0}^{jT_s \leq nT_s - 2d} \Delta u_j \; p((n-j)Ts - 2d) \end{aligned} \tag{4.28}$$

As in the previous protocols let $t = n\ T_s$, $z = j\ T_s$ thus $\Delta t = \Delta z = T_s$. Note that $t_{C_n} = t + d$ and $t_{C_j} = z + d$. Also let $q = \beta\ \Delta t$.

In the limit $\Delta t = T_s \to 0$, the continuous form of equation 4.28 follows as

$$\begin{aligned} q\ \frac{du(t)}{dt} = 1 \quad & - \int_0^{t-2d} \frac{du(z)}{dz} p(t + t_\alpha - z - d) dz \\ & + \int_0^{t-2d} \frac{du(z)}{dz} p(t - z - d) dz \\ & - \int_0^{t-2d} \frac{du(z)}{dz} p(t - z - 2d) dz \end{aligned}$$

Proceeding as in the previous protocols would yield the LTI DDE in $u(t)$

$$q\ddot{u}(t) + q\dot{u}(t) = 1 - u(t - 2d) - \beta\ e^{-d}\ \dot{u}(t - 2d) \tag{4.29}$$

This time, the effect of the delay d involves not only the moves evaluated in the past but also their rate of change. The latter has occurred because of Move Acknowledgements.

4.6 *Protocol 2.1*. Continuous approximation of SPC behavior under constant communication delay

A continuous approximation of SPC response, subjected to constant communication delays and using *Protocol 2.1* for control data communication, is developed in this section. *Protocol 2.1* is similar to *Protocol 2.0* except that in *2.1* the controller uses t_{C_j} as the application time of an unACKed move Δu_j (i.e when t_{A_j} is not known).

The *Protocol 2.1* features are

- No Measurement Timestamping is used.
- The actuator sends back Move Acknowledgements for each control move it applies.
- The SPC controller assumes that all previous moves ($\Delta u_j \quad j = 0....n - 1$) were applied. Acked moves are assumed to have been applied at t_{A_j} while unAcked moves are applied at t_{C_j}. Thus, no previous moves are excluded from prediction.

Since no Measurement Timestamping is used, the Additive Feedback Correction ϕ_n is akin to that in *Protocols 1.0* and *2.0* where the difference between the expected value $\hat{y}_0^n$ and the measured value y_m^n is

$$\phi_n = \hat{y}_0^n - y_m^n \tag{4.30}$$

4.6.1 *Protocol 2.1.*Calculation of Δu_n

The prediction curve $\hat{y}^n$ involves both previous ACKed and unACKed moves. The ACKed moves are those with application times

$$t_{A_j} \leq t_{S_n} \qquad \text{or} \qquad jT_s + 2d \leq nT_s$$

The unACKed moves are those previous moves $(t_{C_j} < t_{C_n})$ with application times

$$t_{A_j} > t_{S_n} \qquad \text{or} \qquad jT_s + 2d > nT_s$$

In this case $\hat{y}_0^n$ and $\hat{y}_\alpha^n$ are calculated by the SPC as follows

$$\hat{y}_0^n = \underbrace{\sum_{j=0}^{jT_s \leq nT_s - 2d} \Delta u_j \; p(t_{C_n} - t_{A_j})}_{ACKed \;\; moves} + \underbrace{\sum_{jT_s > nT_s - 2d}^{jT_s \leq nT_s} \Delta u_j \; p(t_{C_n} - t_{C_j})}_{unACKed \;\; moves} \tag{4.31}$$

$$\hat{y}_\alpha^n = \underbrace{\sum_{j=0}^{jT_s \leq nT_s - 2d} \Delta u_j \; p(t_{C_n} + t_\alpha - t_{A_j})}_{ACKed \;\; moves} + \underbrace{\sum_{jT_s > nT_s - 2d}^{jT_s \leq nT_s} \Delta u_j \; p(t_{C_n} + t_\alpha - t_{C_j})}_{unACKed \;\; moves} \tag{4.32}$$

We have

$$\beta \; \Delta u_n = 1 - (\hat{y}_\alpha^n - \phi_n) = 1 - \hat{y}_\alpha^n + \hat{y}_0^n - y_m^n$$

thus

$$\begin{aligned}
\beta \; \Delta u_n = \quad & 1 - \sum_{j=0}^{jT_s \leq nT_s - 2d} \Delta u_j \; p((n-j)T_s + t_\alpha - d) - \sum_{jT_s > nT_s - 2d}^{jT_s \leq nT_s} \Delta u_j \; p((n-j)Ts + t_\alpha) \\
& + \sum_{j=0}^{jT_s \leq nT_s - 2d} \Delta u_j \; p((n-j)T_s) + \sum_{jT_s > nT_s - 2d}^{jT_s \leq nT_s} \Delta u_j \; p((n-j)T_s) \\
& - \sum_{j=0}^{jT_s \leq nT_s - 2d} \Delta u_j \; p((n-j)Ts - 2d)
\end{aligned} \tag{4.33}$$

As in the previous protocols let $t = n\ T_s$, $z = j\ T_s$ thus $\Delta t = \Delta z = T_s$. Again, $t_{C_n} = t + d$, $t_{C_j} = z + d$. Also let $q = \beta\ \Delta t$.

In the limit $\Delta t = T_s \to 0$ the continuous form of equation 4.28 is

$$\begin{aligned} q\ \dot{u}(t) = 1 \quad & - \int_0^{t-2d} \dot{u}(z)p(t-d+t_\alpha - z)dz - \int_{t-2d}^{t} \dot{u}(z)p(t-d+t_\alpha - z)dz \\ & + \int_0^{t-2d} \dot{u}(z)p(t-d-z)dz + \int_{t-2d}^{t} \dot{u}(z)p(t-d-z)dz \\ & - \int_0^{t-2d} \dot{u}(z)p(t-z-2d)dz \end{aligned}$$

Proceeding as before yields the LTI DDE for $u(t)$ as

$$q\ddot{u}(t) + (q+\beta)\dot{u}(t) = 1 - u(t-2d) + \beta\ (e^{-2d} - e^{-d})\ \dot{u}(t-2d) \tag{4.34}$$

The effect of the communication delay d appears as a linear combination of terms involving the past moves and their derivatives.

4.7 *Protocol 2.2.*Continuous approximation of SPC behavior under constant communication delay

Protocol 2.2 is a combination of *Protocols 1.1* and *2.1.* Hence, it implements Measurement Timestamping and Move Acknowledgements while considering ACKed and unACKed moves in process predictions.

The *Protocol 2.2* features are

- Measurement Timestamping is used, so that t_{S_n} is known.
- The actuator sends back acknowledgements for each control move it applies.
- The SPC controller assumes that all previous moves ($\Delta u_j\ \ j = 0....n-1$) were applied. Acked moves are assumed to have been applied at t_{A_j} while unAcked moves are assumed to have occurred at t_{C_j}. In this way, no previous moves are excluded from prediction.

Since Measurement Timestamping is used, the Additive Feedback Correction, ϕ_n is, as in *Protocol 1.1*, the difference between the expected value $\hat{y}^n_m$ and the measured value y^n_m. Hence,

$$\phi_n = \hat{y}^n_m - y^n_m \tag{4.35}$$

4.7.1 *Protocol 2.2.* Calculation of Δu_n

Recall, that the prediction curve $\hat{y}^n$ involves both previous ACKed and unACKed moves. The ACKed moves would be the moves that have their application times

$$t_{A_j} \leq t_{S_n} \quad \text{or} \quad jT_s + 2d \leq nT_s$$

The unACKed moves would be the previous moves ($t_{C_j} < t_{C_n}$) that have their application times

$$t_{A_j} > t_{S_n} \quad \text{or} \quad jT_s + 2d > nT_s$$

In this case $\hat{y}^n_m$ and $\hat{y}^n_\alpha$ are calculated by the SPC following

$$\hat{y}^n_m = \underbrace{\sum_{j=0}^{jT_s \leq nT_s - 2d} \Delta u_j \; p(t_{S_n} - t_{A_j})}_{ACKed \;\; moves} + \underbrace{\sum_{jT_s > nT_s - 2d}^{jT_s \leq nT_s} \Delta u_j \; p(t_{S_n} - t_{C_j})}_{unACKed \;\; moves} \tag{4.36}$$

$$\hat{y}^n_\alpha = \underbrace{\sum_{j=0}^{jT_s \leq nT_s - 2d} \Delta u_j \; p(t_{C_n} + t_\alpha - t_{A_j})}_{ACKed \;\; moves} + \underbrace{\sum_{jT_s > nT_s - 2d}^{jT_s \leq nT_s} \Delta u_j \; p(t_{C_n} + t_\alpha - t_{C_j})}_{unACKed \;\; moves} \tag{4.37}$$

We have then that

$$\beta \; \Delta u_n = 1 - (\hat{y}^n_\alpha - \phi_n) = 1 - \hat{y}^n_\alpha + \hat{y}^n_m - y^n_m$$

thus

$$\begin{aligned}
\beta \; \Delta u_n = \quad & 1 - \sum_{j=0}^{jT_s \leq nT_s - 2d} \Delta u_j \; p((n-j)T_s + t_\alpha - d) - \sum_{jT_s > nT_s - 2d}^{jT_s \leq nT_s} \Delta u_j \; p((n-j)Ts + t_\alpha) \\
& + \sum_{j=0}^{jT_s \leq nT_s - 2d} \Delta u_j \; p((n-j)T_s - 2d) + \sum_{jT_s > nT_s - 2d}^{jT_s \leq nT_s} \Delta u_j \; p((n-j)T_s - d) \\
& - \sum_{j=0}^{jT_s \leq nT_s - 2d} \Delta u_j \; p((n-j)Ts - 2d)
\end{aligned} \tag{4.38}$$

As in the previous protocols let $t = n\ T_s$, $z = j\ T_s$ thus $\Delta t = \Delta z = T_s$ and note that $t_{C_n} = t + d$, $t_{C_j} = z + d$. Defining $q = \beta\ \Delta t$ and taking the limit $\Delta t = T_s \to 0$, the continuous form of equation 4.28 is

$$\begin{aligned} q\ \dot{u}(t) = 1 \quad & - \int_0^{t-2d} \dot{u}(z)p(t-d+t_\alpha - z)dz - \int_{t-2d}^{t} \dot{u}(z)p(t-d+t_\alpha - z)dz \\ & + \int_{t-2d}^{t} \dot{u}(z)p(t-d-z)dz \end{aligned} \tag{4.39}$$

Proceeding as in the previous protocols would yield the LTI DE in $u(t)$

$$\begin{aligned} q\ddot{u} + (q+\beta)\dot{u} - [(1-\beta)(e^{-d} - e^{-2d}) + e^{-d} - 1]\dot{u}(t-2d) & \\ +u(t) - u(t-d) + u(t-2d) \quad = \quad 1 & \end{aligned} \tag{4.40}$$

This time a fairly complicated dependence upon the past moves and their derivative occurs.

4.8 Two-Parameter SPC

All the results presented in this chapter were obtained for the case of the one-parameter SPC [14][21]. These results can be extended in a very simple way to the case of the two-parameter SPC 1.6.2 by just letting $q = kT_s$ instead of $q = \beta T_s$. All the DDE's remain exactly the same.

Chapter 5

Matched Asymptotic Expansions of the Protocols DDEs

5.1 Introduction

The discrete protocols were approximated in Chapter 4 using semi-discrete, constant coefficients, second order, ordinary delay-differential equations (DDEs). These equations are semi-discrete because the time between samples of the process, $T_s > 0$, appears in the continuous form. Also recall that the delay d is the time required for information from the controller to reach the plant. Similarly, a delay d is required for process measurements, taken at the plant, to return the controller. In other words the round-trip-time, or RTT, is double the communication delay. In practice, decreasing the sampling duration T_s will reduce the RTT since the return delay from the plant to the controller is reduced. However, a point is reached where further reductions in T_s have little impact, and T_s and d may be assumed to be independent. This limit is considered here for control over the web since control data packets add little to the total web traffic.

The semi-discrete DDEs developed in the previous section are "singular" because in the limit of zero sampling duration, the order of the DDE changes and the coefficient of the second order derivative term vanishes. The term "singular" means that one then "loses" a condition because the order of the differential equation has been reduced. These DDEs are also termed "stiff". The "stiffness" of the DDEs corresponds with an increasing gap between the rate constants, or eigenvalues, of the DDEs that increases as the sampling duration, T_s, decreases. The rate constants may be characterized as a "pair" in that they either have real parts that are large or small in absolute value. This feature allows the decomposition of the closed-loop control response into a linear combination of a "fast timescale" or "fast response" and a "slow timescale" or "slow response". Those rate constants having a large absolute real part give the "fast response" while the those with the smaller absolute real part give the "slow response".

An apt method to analyze these singular DDEs is the Method of Matched Asymptotic Expansions (MAEs). The method involves the discovery of a so-called "inner solution" which corresponds to the "fast control response", and an "outer solution" corresponding to the "slow control response". It was historically used as a method to approximate viscous flow behavior. At a surface, the viscous fluid must satisfy a "no-slip", or zero velocity condition. The effect of this boundary condition is to typically introduce a "boundary layer" wherein the fluid speed rapidly changes from zero to the free stream velocity outside the boundary layer.

The control analogy with fluid flow immediately follows from noting, for example, that the "fast timescale" or "fast control response" is a boundary layer in time driven by the initial conditions analogous to the no-slip condition. The time-wise boundary layer thickness is determined by the sampling duration T_s in the control problem. This is analogous to the dependence of the boundary layer thickness in the fluid flow on the inverse of the square root of the so-called Reynolds number. Now, the "slow control response" or "slow timescale" is *independent*, to a first approximation, of the sampling duration just like the free stream flow which knows nothing about the existence of the boundary layer. This separation of behaviors is consistent with good control performance [21].

Therefore, in this chapter the Method of Matched Asymptotic Expansions is used to characterize the set of Protocols introduced in Chapter 3. These equations possess key features that allow for a reduction of the entire set to a generic description. That description is first described and then the network details of each Protocol are inserted for a clear comparison of each Protocol strengths and weaknesses.

5.2 Analysis of Zero Delay Limit

The zero delay limit provides the desired generic mathematical form while also giving insight into the description provided by the Matched Asymptotic Expansion without the intrusion of networking details.

In the limit of a zero delay the DDEs developed in Chapter 4 reduce to

$$q\ddot{u}(t) + (q+\beta)\dot{u}(t) + u(t) = 1 \tag{5.1}$$

with $u(0) = 0$ and $\dot{u}(0) = 1/q$ where $q = kT_s$. Note that the "dot" notation is convenient to imply differentiation with respect to the time variable t.

Control performance improves as the sampling duration, T_s, or equivalently q, is reduced since the fast timescale response boundary layer is correspondingly reduced in size. In other words the control response becomes more like a first order control response as the sampling duration is reduced. Therefore, (5.1) is analyzed for small q.

5.2.1 Outer Solution

The outer expansion of (5.1) is a power series in the small parameter q and the first order approximation is just

$$\beta\dot{u}_0(t) + u_0(t) = 1 \tag{5.2}$$

where the initial conditions, $u(0) = 0$ and $\dot{u}(0) = 1/q$, are dropped since this expansion is invalid at small times because $\ddot{u}(t)$ cannot be neglected at early times.

Solving gives the outer expansion $u(0)(t)$ as

$$u(0)(t) \approx u_0(t) = 1 + Ae^{\lambda t} \tag{5.3}$$

and the correction term is $O(q)$ ("O" means "the size of" and read "big oh of") because the next term in the power series will involve q^1.

The rate constant satisfies

$$\lambda + 1/\beta = 0 \tag{5.4}$$

Now, there is an unknown constant A in the outer solution and this is found by functionally matching, *not* value patching at some arbitrary location, the outer and inner solutions within a zone of overlapping validity.

5.2.2 Inner Solution

The inner solution is found by proposing the existence of a boundary layer of width q near $t = 0$. Therefore, the variable $s = t/q$ is defined where the assumption is that the inner solution is valid for $s = O(1)$ and that zone corresponds to the boundary layer $t = O(q)$. Substitution to (5.1) yields

$$u_{ss}(t) + (q+\beta)u_s(t) + qu(s) = q \tag{5.5}$$

with $u(0) = 0$ and $u_s(0) = 1$.

It is clear that for vanishingly small q that the second derivative term is now order one and cannot be neglected as occurred in the outer solution. However, to first order, the $q = 0$ inner solution knows nothing about the steady state and is only concerned with the interaction of the control response with the initial conditions.

As before, the inner expansion of (5.5) is a power series in the small parameter q and the first order approximation is

$$u_{0ss}(s) + \beta u_{0s}(s) = 0 \tag{5.6}$$

where the initial conditions are applied to the first approximation and $u_0(0) = 0$ and $u_{0s}(0) = 1$.

Solving (5.6) gives the first order approximation to the inner solution $u^{(i)}$ as

$$u(i)(s) \approx u_0(s) = (e^{\tilde{\lambda}s} - 1)/\tilde{\lambda} \tag{5.7}$$

where $\tilde{\lambda}$ satisfies

$$\tilde{\lambda} + \beta = 0 \tag{5.8}$$

and the $\lambda = 0$ has been incorporated, for brevity, into the constant term in u(i)(s).

Note that there are no free constants left in the inner solution so that when this solution is functionally matched with the outer solution the free constant in the latter will be determined.

5.2.3 Matching

The inner and outer solutions are matched following the procedure advocated in [36]. The matching procedure involves: (i) placing each solution in the complementary variable, (ii) Taylor expanding for small q, (iii) returning the solutions to one of the variables, and (iv) functionally equating the expansions to various orders in the parameter q [21].

Thus the outer solution placed in the inner solution variable is

$$u(0)(t) \approx 1 + Ae^{\lambda_s\, q} \tag{5.9}$$

and the inner solution placed in the outer solution variable is

$$u(i)(s) \approx (e^{\tilde{\lambda}t/q} - 1)/\tilde{\lambda} \tag{5.10}$$

Next, each of these are expanded for small q and to first order

$$u^{(0)}(t) \approx 1 + A \tag{5.11}$$

while

$$u^{(i)}(s) \approx -1/\tilde{\lambda} \tag{5.12}$$

Equating each of these forms yields

$$A = -(1 + 1/\tilde{\lambda}) \tag{5.13}$$

5.2.4 Composite Solution

The inner solution valid for within the boundary layer $t = O(q)$ and the outer solution valid outside the boundary layer for $t = O(1)$ are used to construct a "composite" solution valid to $O(q)$ for all time t. This composite solution is assumed to be the "union" of the outer and inner solutions where the region of overlapping validity wherein they were matching is the "intersection" of the two solutions taken as either of $1 + A$ or $-1/\tilde{\lambda}$. Thus the composite solution, $u^{(c)}(t) = u^{(o)}(t) + u^{(i)}(t) - (1 + A)$, gives

$$u^{(c)}(t) = A(1 - e^{\lambda t}) + \frac{e^{\tilde{\lambda}t/q} - 1}{\tilde{\lambda}} + O(q) \tag{5.14}$$

where the "O(q)" means that the correction to this composite solution is the size of q.

5.2.5 Discussion

The composite solution shows that the control may be broken up into a fast and a slow response. The fast response is the order of $t = O(kT_s)$ while the slow time scale is $t = O(1)$. The wide separation of these timescales means that the control performance, as represented by the slow response, is approximately first order. The behavior of the control response as the sampling duration shrinks to zero is also well-conditioned since the composite solution is bounded for $q \to 0$. Note, that this contrasts with least-squares based, matrix predictive control methods which have unbounded moves in the limit of zero sampling duration and are ill-conditioned to small changes in T_s [21]. The well-conditioned nature of SPC is a basic reason for its superiority in networked control environments with time-varying delays.

The SPC control algorithm considered here generalizes SPC as introduced in [14] to dependence upon two parameters: β and k. It is clear from the above analysis that the parameter β is the slow control timescale. The fast timescale involves the width of the boundary layer equal to $O(q) = O(kT_s)$. If one also inserts the correction $\tilde{\lambda}$ then the fast timescale, or boundary layer, is kT_s/β. Note that the choice made by [14] is to choose the aggressive case $k = \beta$ from which the boundary layer width is approximately the sampling duration T_s.

SPC control becomes more aggressive as β is reduced since the slow timescale is shortened. On the other hand, the fast timescale kT_s/β is made wider for decreased β giving correspondingly less aggressive control at this timescale. For this reason, k is useful to modify the fast timescale independently of the slow.

While the control is globally stable in the absence of delays, the choices of k and β that are consistent with widely separated control response timescales are those that satisfy $kT_s/\beta \ll \beta$ or $\beta \gg \sqrt{kT_s}$. Finally, it is useful to note that global stability of (5.1) is guaranteed if $q + \beta > 0$ and that this stability may be increased by either widening the boundary layer width via the sampling duration T_s or k and/or by reducing the slow timescale by increasing β.

In the Introduction to this chapter the zero delay expansion was said to provide a generic description for the generalization to DDEs with nonzero delays. This is true because the DDE solution structure is unchanged as a function of the delay magnitude and shows the same separation of control response timescales. In fact, the nontrivial delays simply modify the polynomial characteristic equations, $\lambda = -1/beta$ and $\tilde{\lambda} = -\beta$ to a transcendental form.

A final point is that the analysis in this chapter has stayed in the continuous time domain. Analyzing the control in the time domain has given the benefit of a direct understanding of the control dynamics on the control algorithm parameters. This feature will prove to be useful in understanding various features of the more complicated networked control problem.

5.3 Protocol Analysis

The Protocols are in two series: (i) Series 1 has Protocols 1.0 and 1.1 and, (ii) Series 2, has Protocols 2.0, 2.1, and 2.2. The protocols are distinguished by assumptions regarding the time of application of moves, "Application Assumption", and the time of sensing of the process state, "Measurement Timestamping". These assumptions are detailed at the beginning of each section devoted to each series.

All of the Protocols satisfy the initial conditions $u(0) = 0$, $\dot{u}(0) = 1/q$ and for $t \leq 0$ the solution is assumed to be identically zero.

Each Protocol DDE is characterized by: (i) a $q\ddot{u}(t)$ term, (ii) a linear combination of $\dot{u}(t)$ terms which may or may not be delayed and which approach $(q+\beta)\dot{u}(t)$ in the limit of of zero delay $d = 0$, and (iii) a linear combination of $u(t)$ terms which may or may not be delayed that approach $u(t)$ in the limit of zero delay. The features in (i), (ii) and (iii) guarantee that the MAE analysis is analogous to that developed for the zero delay case. The presence of delayed terms in $u(t)$ and $\dot{u}(t)$ simply lead to transcendental characteristic equations for λ and $\tilde{\lambda}$ that reduce to polynomials in the zero delay limit.

In the following sections each Protocol is described in turn and the results are stated in terms of the analysis already worked out for the zero delay case by noting formulae for λ, $\tilde{\lambda}$.

The introduction of nonzero delays disrupts the globally stable control and increases the approximately first order behaviour enjoyed in the zero delay case to second order. Therefore, the stability of each protocol is found and the region within which the slow response is critically damped is found where appropriate.

5.4 Series 1 Protocols

In Series 1, all moves are computed on the Application Assumption that *past moves were applied at the time they were found.* In this case there is no Measurement Acknowledgement, or Acking.

Predictive errors due to this assumption are corrected by future sensing of the process state. Since they are used as corrections they are termed Additive Feedback Corrections. Specifically, the difference between the most recently received process state measurement and the prediction of the current state using the Application Assumption, is used as an additive correction to the predicted future state required in the calculation of the next move. The effect of neglecting the difference between the time when the process state measurement was made and the time of the predicted current state, used in this Protocol's Additive Feedback Assumption, differentiates Protocol 1.0 and 1.1.

5.4.1 *Protocol 1.0*

As was stated in the previous section, the difference between the timing of the process state measurement and the predicted current state is neglected in this Protocol's Additive Feedback Assumption.

The DDE for *Protocol 1.0* is

$$q\ddot{u}(t) + (q+\beta)\dot{u}(t) + u(t-2d) = 1 \tag{5.15}$$

The characteristic equations for λ and $\tilde{\lambda}$ satisfy

$$\beta\lambda + e^{-2\lambda d} = 0 \tag{5.16}$$

$$\tilde{\lambda}(\tilde{\lambda}+\beta) = 0 \tag{5.17}$$

The first approximations to λ and $\tilde{\lambda}$ are

$$\lambda \approx \frac{-1+\sqrt{1-8d/\beta}}{4d} \tag{5.18}$$

$$\tilde{\lambda} = -\beta \tag{5.19}$$

where the slow and fast response timescales respectively are $1/\lambda$ and $kT_s/\tilde{\lambda}$. Note that the approximation for λ was based upon $\exp(-2\lambda d) \approx 1/(1+2\lambda d)$ for small λd.

Control Performance

Good control performance occurs when the control is at least critically damped and from the outer solution rate constant in (5.19), this requires $d/\beta < 1/8$ where $\lambda d \ll 1$. Note that this constraint *only* applies to $\lambda d \ll 1$ since the control is stable for the limiting cases of $d = 0$ (Section 5.2) and $d \to \infty$. This point is detailed in the next section on global stability.

The tightness of the constraint on d/β demonstrates the difficulty of networked control. It is quite restrictive since $\beta \approx 1$ requires $d \approx 1/8$ of the process time constant. Yet the control at these levels of β is no faster than the process time constant.

However, a factor of two improvement in control performance can be had by reducing β from near unity to $\beta \approx 0.5$ since the slow control response timescale is proportional to β at small λd. Reaching this level is possible for delays equal to approximately 5% of the process time constant.

There is also the relationship between the slow and fast control timescales to consider. The slow response timescale is assumed to be much larger than the sampling duration T_s to allow for good separation of the slow and fast control timescales. This can be seen by noting that the fast timescale is kT_s/β and choosing $k = O(\beta)$ yields a fast timescale at the order of the sampling duration. This choice of the parameter k also gives a well-conditioned dependence on changes in the sampling duration since the fast timescale is no faster than the sampling duration. Adjustment of k to be larger than $O(\beta)$ will be used in later chapters to expand the stability envelope when the network delay characteristics make this necessary.

Global Stability

The range of values of β and k for which the SPC algorithm is globally stable under *Protocol 1.0* are found.

The full rate constant equations are found by substituting $u = exp(\Lambda t)$ to (5.17) from which

$$q\Lambda^2 + (q + \beta)\Lambda + e^{-2\Lambda d} = 0 \tag{5.20}$$

It is clear that at large delay, $d \to \infty$, the rate constants approach $\Lambda = 0$ and

$\Lambda = -(q+\beta)$ and the control is stable. Similarly, as was shown in the zero delay analysis in Section 5.2, the control is stable for $d = 0$.

However, control at small and finite delays associated with $\Lambda d \ll 1$ *can be unstable.* This can be seen by expanding $\exp(-2\Lambda d) \approx 1 - 2\Lambda d + 2\Lambda^2 d^2$ in (5.20)

$$q\Lambda^2 + (q + \beta - 2d)\Lambda + 1 = 0 \tag{5.21}$$

Hence, a minus real part of Λ requires $\beta > 2d - q$ for stability.

A somewhat surprising observation is that control at, or near, $\beta = 1$ may also be unstable in the presence of delays since $2d - q < 1$. This means that even control that is operating near the open-loop response dynamics *still* requires control. This point emphasizes the danger of attempting to generalize zero delay results to those where delays are present.

This difficulty is easily alleviated in the two-parameter SPC used here by simply *expanding* the zone of stability using the second parameter k. Recall that $q = kT_s$ and the zone of stability is then $\beta > 2d - kT_s$. In this case it is possible to choose k such that $2d - kT_s < \beta$. This action is only taken to stabilize control and is used in the following chapters as part of the real-time adaptation of the two SPC parameters β and k.

5.4.2 *Protocol 1.1*

In the discussion of *Protocol 1.1*, only differences between this protocol and Protocol 1.0 are considered. The remaining discussion of Protocol 1.0 applies to Protocol 1.1 and is not repeated.

Protocol 1.1 is based on the same Application Assumption as *Protocol 1.0.* However, unlike *Protocol 1.0*, the Additive Feedback Correction used here also respects the time that the process measurement was made. Respecting the process measurement timing is termed "Measurement Timestamping". In other words, the difference between measured process state and the predicted state is evaluated at the same time instant.

The resulting DDE for *Protocol 1.1*, developed in Section 4.4 is

$$q\ddot{u}(t) + (q + \beta)\dot{u}(t) + u(t) - u(t-d) + u(t-2d) = 1 \tag{5.22}$$

Qualitative comparison of this DDE with the DDE (5.15) for *Protocol 1.0* shows that the Measurement Timestamping involves an additional term $u(t) - u(t-d)$ while the terms multiplying $\ddot{u}(t)$ and $\dot{u}(t)$ remain unchanged. Hence, the inner, or fast control timescale, solution in *Protocol 1.1* is unaltered from that in *Protocol 1.*

The $u(t)-u(t-d)$ correction does change the outer solution characteristic equation for the rate constant λ from the seen in *Protocol 1.* Hence, λ and $\tilde{\lambda}$ satisfy

$$\beta\lambda + 1 - e^{-\lambda d} + e^{-2\lambda d} = 0 \tag{5.23}$$

$$\tilde{\lambda}(\tilde{\lambda} + \beta) = 0 \tag{5.24}$$

The first approximations to λ and $\tilde{\lambda}$ are

$$\lambda \approx \frac{-(1 + d/\beta) + \sqrt{(1 + d/\beta)^2 - 8d/\beta}}{4d} \tag{5.25}$$

$$\tilde{\lambda} = -\beta \tag{5.26}$$

where the approximation for λ is again based upon $\exp(-2\lambda d) \approx 1/(1 + 2\lambda d)$.

Control Performance

As was observed for *Protocol 1.0*, the control performance in *Protocol 1.1* depends on the slow response timescale $1/\lambda$. The approximation of the λ rate constant in (5.26) is a function of d/β and shows a critically damped behaviour when

$$(1 + d/\beta)^2 = 8d/\beta \tag{5.27}$$

or $d/\beta < 1/6$ for overdamped control dynamics. This constraint is a slight relaxation of that seen $d/\beta < 1/8$ seen in *Protocol 1.0* and is due to Measurement Timestamping.

A useful comparison between *Protocol 1.0* and *1.1* can be found by comparing the characteristic equations for each protocol, i.e.

$$2d\lambda^2 + \lambda + 1/\beta = 0 \tag{5.28}$$

$$2d\lambda^2 + \frac{\beta + d}{\beta}\lambda + 1/\beta = 0 \tag{5.29}$$

where the first equation is for *Protocol 1.0* and the second for *Protocol 1.1.* It is clear that *Protocol 1.1* results in an increased damping seen in the coefficient $(\beta+d)/\beta$. The effect of this damping vanishes in the limit of zero delay and large delays where the two protocols become identical. Hence, *Protocol 1.1* is somewhat more conservative for delays where the control can become unstable. This point is considered in more detail in the next section.

Stability

The global characteristic equation for *Protocol 1.1* is

$$q\Lambda^2 + (q+\beta)\Lambda + 1 - e^{-\Lambda d} + e^{-\Lambda d} = 0 \tag{5.30}$$

This equation shows that in the limit of zero delay and large delays the control is stable. Therefore, it is useful to expand for $\Lambda d \ll 1$ so that $\exp(-2\Lambda d) \approx 1 - 2\Lambda d + 2\Lambda^2 d^2$ from which

$$q\Lambda^2 + (q+\beta-d)\Lambda + 1 = 0 \tag{5.31}$$

Comparison of this characteristic with the same for *Protocol 1.0* in (5.21) shows that the coefficient of Λ indicates stable control for $\beta > d-q$. This is a little less restrictive than the requirement $\beta > 2d-q$ in *Protocol 1.0.*

5.5 Series 2 Protocols

In *Protocols 1.0* and *1.1*, the Application Assumption is that *past moves were applied at the time they were found.* This assumption is modified in the Series 2 protocols to be: *past moves were applied at the time they were applied.* In other words, the time that a move is applied is noted by the plant and passed back to the controller. This is termed "Move Acknowledgement" or Acking.

It is clear that, because of communication delays, not all moves will have been acknowledged when the next move is being computed. Therefore, the Series 2 protocols are differentiated by two features: (i) whether or not moves that are unacknowledged, or unAcked, are assumed to be applied at the time they were found, and (ii) whether, or not, Measurement Timestamping is applied in the Additive Feedback Assumption:

- *Protocol 2.0*: (i) unacknowledged moves are ignored, and (ii) Measurement Timestamping is not applied in the Additive Feedback Assumption

- *Protocol 2.1*: (i) unacknowledged moves are not ignored and are assumed, as in the Series 1 protocols, to have been applied at the time they were computed and, (ii) Measurement Timestamping is not applied in the Additive Feedback Assumption

- *Protocol 2.2*: (i) unacknowledged moves are not ignored as in *Protocol 2.1* and, (ii) Measurement Timestamping is applied in the Additive Feedback Assumption

5.5.1 *Protocol 2.0*

The *Protocol 2.0* DDE, developed in Section 4.5 is

$$q\ddot{u}(t) + q\dot{u}(t) + \beta e^{-d}\dot{u}(t-2d) + u(t-2d) = 1 \tag{5.32}$$

Comparison with the *Protocol 1.0* DDE in (5.15) indicates that move acknowledgement has modified the term $(q+\beta)\dot{u}(t)$ to $q\dot{u}(t) + \beta e^{-d}\dot{u}(t-2d)$ while the terms $u(t-2d)$ and the $q\ddot{u}(t)$ remain the same.

In this protocol, λ and $\tilde{\lambda}$ satisfy

$$e^{-d}\beta\lambda + 1 = 0 \tag{5.33}$$

$$\tilde{\lambda}(\tilde{\lambda} + \beta e^{-(1+\frac{2\tilde{\lambda}}{q})d}) = 0 \tag{5.34}$$

The first approximations to λ and nontrivial $\tilde{\lambda}$, valid for small λd, satisfy

$$\lambda \approx \frac{e^{-d}}{\beta} \tag{5.35}$$

$$\tilde{\lambda} = \frac{-1 + \sqrt{1 - 8d\beta e^{-d}}}{4d} \tag{5.36}$$

where it is assumed that $\exp(-2\lambda d) \approx 1/(1+2\lambda d)$.

Control Performance and Discussion

The control performance in this protocol is poor for two reasons: (i) the rate constant in the outer, or slow response, solution (5.36) shows increasingly aggressive control as the delay increases and, (ii) the rate constant in the inner, or fast response, shows less aggressive control as the delay is increased. Hence, the slow response timescale is speeding up while the fast response timescale is slowing down for increased delays. Such control is clearly not desirable and stands in contrast to the Series 1 protocols where the fast control timescale is *independent* of the delay to first order while the slow response timescale becomes less aggressive as the delay is increased. This is quite surprising since one would consider using the acknowledgements that include the actual time of application of a move should improve the control performance. However, it is important to remember that it is the *balance* between the Application Assumption and the Additive Feedback Assumption that affects control performance.

Poor control performance in this protocol, in comparison with the Series 1 protocols, may stem from unacknowledged moves ignored in this protocol. Therefore, *Protocol 2.1* is considered where unacknowledged moves are assumed to have been applied at the time they were computed as in the Series 1 protocols.

Stability

The global characteristic equation for *Protocol 2.0* is

$$q\Lambda^2 + q\Lambda + \beta e^{-d} e^{-2d\Lambda}\Lambda + e^{-2d\Lambda} = 0 \tag{5.37}$$

Assuming $e^{-2d\Lambda} \simeq 1 - 2d\Lambda$ 5.37 becomes

$$(q - \beta e^{-d} 2d)\Lambda^2 + (q + \beta e^{-d} - 2d)\Lambda + 1 = 0 \tag{5.38}$$

The coefficient of Λ^2 becomes more positive when d increases while that of Λ becomes more negative. A stability zone is obtained by requiring the coefficient of Λ (the damping) to be positive

$$d < \frac{q+\beta}{\beta+2} \tag{5.39}$$

5.5.2 *Protocol 2.1*

In *Protocol 2.1* the Application Assumption of *Protocol 2.0* is updated to include unacknowledged moves as having occurred at the time of computation.

The *Protocol 2.1* DDE developed in Section 4.6 is

$$q\ddot{u}(t) + (q + \beta)\dot{u}(t) - \beta(e^{-2d} - e^{-d})\dot{u}(t - 2d) + u(t - 2d) = 1 \tag{5.40}$$

The main feature of interest in this DDE is the change from the *Protocol 2.0* DDE in (5.32) to include the term $(q+\beta)\dot{u}(t)$ seen in the Series 1 protocols. This term appears here because the unacknowledged moves are now included as having occurred at the time of computation.

The *Protocol 2.1* characteristic equations λ and $\tilde{\lambda}$ are

$$2d\lambda^2 + (1 - e^{-2d} + e^{-d})\lambda + 1/\beta = 0 \tag{5.41}$$

$$\tilde{\lambda}[\tilde{\lambda} - \beta(e^{-2d} - e^{-d})e^{-\frac{2d}{q}\tilde{\lambda}} + \beta + q] = 0 \tag{5.42}$$

Assuming $e^{-\frac{2d}{q}\tilde{\lambda}}$, the first approximations to λ and $\tilde{\lambda}$ are

$$\lambda = \frac{-(1+d) + \sqrt{(1+d)^2 - 8d/\beta}}{4d} \tag{5.43}$$

$$\tilde{\lambda} = -\beta - 1 \tag{5.44}$$

where the approximation for λ was constructed from $\exp(-\lambda d) \approx 1/(1+\lambda d)$ and $d \ll 1$.

Control Performance

This protocol is a definite improvement over Protocol 2.0 and approximates the results for *Protocol 1.0.* Specifically, respectively comparing the *Protocol 1.0* results in (5.17) and (5.19) with their *Protocol 2.1* counterparts in (5.42) and (5.44) shows that *Protocol 2.1* is somewhat more damped over *Protocol 1.0. Protocol 2.1* also has a slightly better restriction to obtain overdamped control than seen in *Protocol 1.0.* These results show that *Protocol 2.1* possesses slightly better control performance than seen in *Protocol 1.0.*

Control Stability

Substitution of $u(t) = \exp(\Lambda t)$ to the characteristic equation for λ in (5.42) and expanding $\exp(-\lambda d) = 1 - \lambda d + \lambda^2 d^2$ and $\exp(-d) = 1 - d + d^2$ yields

$$q\Lambda^2 + (q + \beta(1+d) - 2d)\Lambda + 1 = 0 \tag{5.45}$$

which yields an approximate stability zone $d < \frac{q+\beta}{2-\beta}$. Comparison with the stability results for Protocol 1.0 in Section 5.4.1 shows that Protocol 2.1 has a somewhat broader zone of stability for a given set of parameters β and k (recall that $q = kT_s$). However, such a small increased zone of stability has little effect since k can be independently adapted to increase the stability zone where necessary.

Discussion

The results for *Protocol 2.1* compare very closely with that for *Protocol 1.0.* The tighter stability constraint in this protocol in comparison with *Protocol 1.1* (Section 5.4.2) may be improved by including an Additive Feedback Assumption where Measurement Timestamping is included. Therefore, *Protocol 2.1* is further updated to *Protocol 2.2* where Measurement Timestamping, used in *Protocol 1.1*, is also included.

5.5.3 *Protocol 2.2*

In *Protocol 2.2* the Additive Feedback Assumption in *Protocol 2.1* is modified to include Measurement Timestamping.

The DDE for this protocol, developed in Section 4.7, is

$$\begin{aligned} q\ddot{u} + (q+\beta)\dot{u} - [(1-\beta)(e^{-d} - e^{-2d}) + e^{-d} - 1]\dot{u}(t-2d) & \\ +u(t) - u(t-d) + u(t-2d) &= 1 \end{aligned}$$

The characteristic equations λ and $\tilde{\lambda}$ are

$$2d^2\lambda^2 + (\beta - 2d)\lambda + 1 = 0 \tag{5.46}$$

$$\tilde{\lambda}(\tilde{\lambda} + [1 - (1-\beta)(e^{-d} - e^{-3d}) - e^{-d}]) = 0 \tag{5.47}$$

where the equation for λ has been simplified for small d.

The first approximations to λ and $\tilde{\lambda}$ are

$$\lambda = \frac{-(\beta - 2d) + \sqrt{(\beta - 2d)^2 - 8d^2}}{4d^2} = 0 \tag{5.48}$$

$$\tilde{\lambda} = -\beta + d \tag{5.49}$$

where the approximation for λ was constructed from $\exp(-\lambda d) \approx 1/(1 + \lambda d)$ and $d \ll 1$.

Control Performance

The outer solution rate constant (5.49) indicates an instability in the outer solution occurs when $\beta < 2d$ and this agrees with stability found using the full equation (see the next section).

Control Stability

Substitution of $u(t) = \exp(\Lambda t)$ to the characteristic equation for λ in (??) and expanding $\exp(-\lambda d) = 1 - \lambda d + \lambda^2 d^2$ and $\exp(-d) = 1 - d + d^2$ yields

$$q\Lambda^2 + (q + \beta - (1 - \beta)d)\Lambda + 1 = 0 \tag{5.50}$$

An approximate stability zone can be obtained by setting $d < \frac{q+\beta}{1-\beta}$. Comparison with the stability results for *Protocol 2.1* (5.45) shows that *Protocol 2.2* has an identical zone of stability.

5.6 Discussion Of All Protocols

In this chapter the Series 1 and Series 2 protocols were analyzed for control performance and stability. The surprising result is that the best SPC control performance and stability of a first order LTI process, lie with a control regime possessing two features: (i) moves are made under the assumption that they will be applied at the time of computation and, (ii) the error made in (i) is corrected through timestamped process sensing.

The first feature leads to good control performance since the $\dot{u}(t)$ terms are unmodified from those for zero delay, while the second leads to enhanced control stability through the modification of the $u(t)$ in the zero delay DDE. A rule-of-thumb is that it is best to simply make moves under a *constant* assumption of when they will be applied while relying on precise timestamped feedback of the process state to stabilize control move assumption errors.

Stated another way, the rule-of-thumb implies that it may be best to be "sloppy" on the computational side since "precise" process state measurements will continually correct any errors incurred by the sloppy computation. Hence, relying on sensing for control stability rather than complex algorithms intended to compensate for unknown delays may provide an unexpected route to stable and effective network control.

Having said the above, it must be remembered that these results apply to predictive control of a first order, LTI process. The multitude of variations from such a process may lead to completely different conclusions. However, the results are compelling for their simplicity.

Chapter 6

Protocols Control Performance Comparison via Simulations

6.1 Introduction

Control performance is quantified by three measures

- Maximum Percent Overshoot (MPO):Largest positive relative error.
- Rise Time (RT): taken here as the time for the process to reach 90% of the set point.
- Settling Time (ST): taken here as the time for the process to reach and stay within 2% of the set point.

The *ns-2* setup shown in Figure 2.1 is used in the simulation. The five suggested protocols performances are to be compared in the case of random communication delays on a high traffic network. All five protocols will be subjected to the same network and delay conditions.

The links bandwidths are set as shown in the following tcl script block of code

```
# Connect the nodes with a star graph
$ns duplex-link $n0 $n3 10Mb    1ms  DropTail
$ns duplex-link $n2 $n3 10Mb    1ms  DropTail
$ns duplex-link $n3 $n6 0.015Mb   1ms  DropTail
$ns duplex-link $n3 $n7 0.015Mb   1ms  DropTail
$ns duplex-link $n6 $n4 0.015Mb   1ms  DropTail
$ns duplex-link $n7 $n4 0.015Mb   1ms  DropTail
$ns duplex-link $n4 $n1 10Mb    1ms  DropTail
$ns duplex-link $n4 $n5 10Mb    1ms  DropTail
```

```
$ns queue-limit $n3 $n6 10
$ns queue-limit $n6 $n3 10

$ns queue-limit $n3 $n7 10
$ns queue-limit $n7 $n3 10
$ns queue-limit $n4 $n6 10
$ns queue-limit $n6 $n4 10
$ns queue-limit $n4 $n7 10
$ns queue-limit $n7 $n4 10
```

The bit error rate is not set on any of the links. The packet loss results from dropping packets by routers with full queues and not from data corruption.

Competing IP traffic is generated using *ns-2* traffic generators as follow

- A variable bit rate (VBR) two-way UDP traffic between nodes *n*0 and *n*1. The VBR On, Off periods are drawn from Pareto distributions as shown in the following tcl script block of code

  ```
  set e_n0n1 [new Application/Traffic/Pareto];
  $e_n0n1 attach-agent $src_n0n1;
  $e_n0n1 set packetSize_  200;
  $e_n0n1 set burst_time_ 150ms;
  $e_n0n1 set idle_time_  150ms;
  $e_n0n1 set rate_ 35.0kb;
  $e_n0n1 set shape_ 1.5;

  set e_n1n0 [new Application/Traffic/Pareto];
  $e_n1n0 attach-agent $src_n1n0;
  $e_n1n0 set packetSize_ 400;
  $e_n1n0 set burst_time_ 150ms;
  $e_n1n0 set idle_time_ 300ms;
  ```

```
$e_n1n0 set rate_ 35.0kb ;
$e_n1n0 set shape_ 1.3;
```

- A VBR one-way UDP traffic between nodes $n6$ and $n5$. The VBR On, Off periods are drawn from Exponential distributions as shown in the following tcl script block of code

```
set e_n6n5 [new Application/Traffic/Exponential];
$e_n6n5 attach-agent $src_n6n5;
$e_n6n5 set packetSize_  250;
$e_n6n5 set burst_time_  0ms;
$e_n6n5 set idle_time_  500ms;
$e_n6n5 set rate_ 60Kb;
```

In the simulation, each closed loop control session lasts for 10 simulated seconds. The simulation is run in batches: 20 batches in total and 50 closed loop control runs per batch. Thus a total of 1000 closed loop 10-second control sessions are simulated. For each batch the random number generators use different random streams. Moreover for each batch, the start times of the traffic generators are drawn randomly from a uniform distribution between 0 and 1 second, which provides different, random overlap between the random RTT time series and the closed loop sessions.

The plant is a first order plant with a unit step response

$$P(t) = 1 - e^{-t}$$

In all the protocols simulations it is assumed that

- The SPC parameter β is set to a medium-aggressive value of 0.5.
- The sensor is time-driven with a sampling period $T_s = 0.1$ *seconds* chosen to be small compared with the time constant.

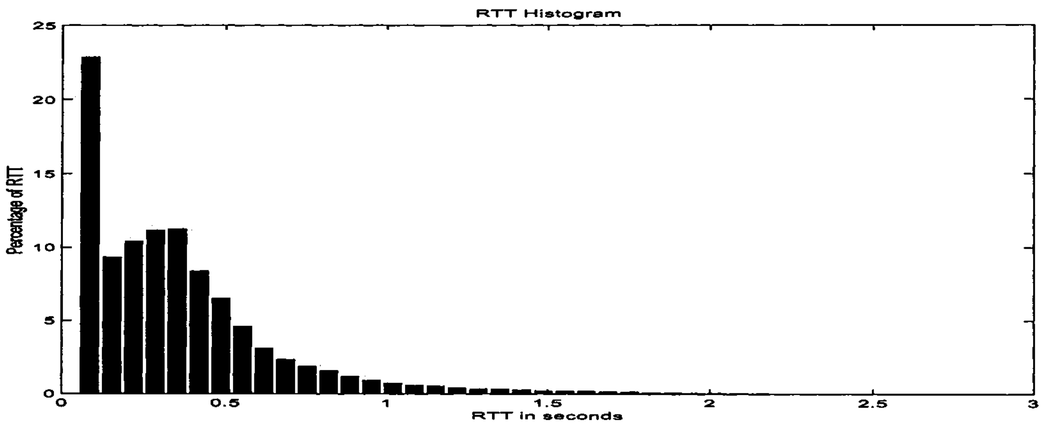


Figure 6.1: Large Load Network. RTT histogram for all 5 protocols.

6.2 Control performance histograms: large load network

Figure 6.1 shows the histogram of the RTT between the controller and the plant nodes resulting from a fairly large load traffic on the network. These histograms are the same for all protocols since the control data traffic is small in all cases and the size of the control data packets was made the same for all the protocols. This is realistic since the control data traffic is small compared with other traffic on the real Internet and thus the control traffic should not affect the Internet traffic conditions. The average RTT is 0.3625 *sec* and its standard deviation is 0.2959 *sec*. On average, the RTT is about 3.6 times the process sampling period T_s.

Control performance measures for *Protocols 1.0, 1.1, 2.0, 2.1, 2.2* are shown, in the form of histograms, respectively in figures 6.2, 6.3, 6.4, 6.5, 6.6 .

Table 6.1 presents statistical summaries of the control performance indices for the usage of each protocol. The seventh column, denoted by Number of Excessive Overshoots (NEO), shows the number of realizations that had an MPO larger than 20% .

6.3 Control performance histograms: smaller load network

Figure 6.7 shows the histogram of the RTT between the controller and the plant nodes resulting from a smaller load traffic than that of the previous section. The average

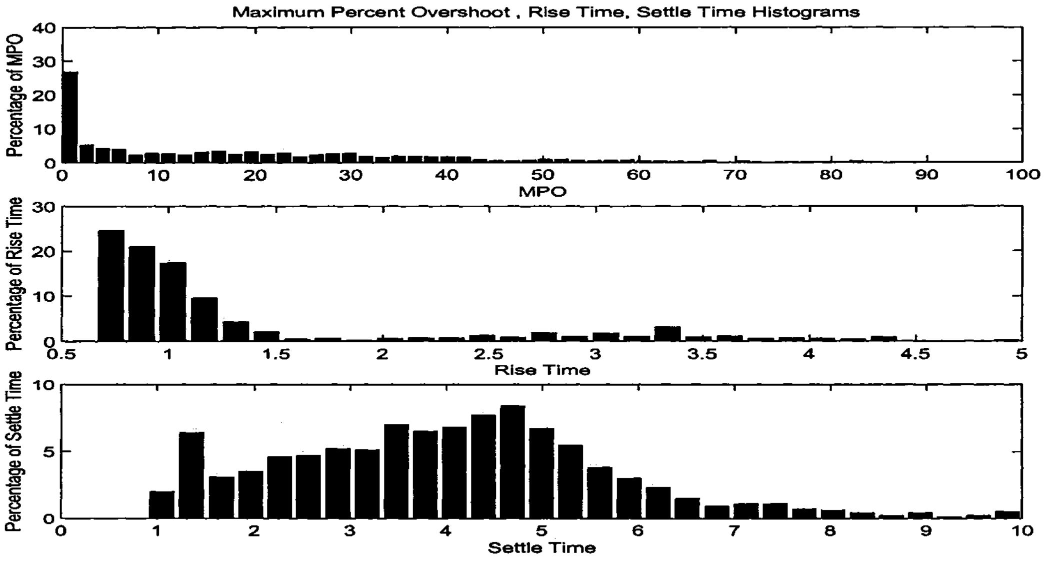


Figure 6.2: *Protocol 1.* MPO, RT, and ST histograms.

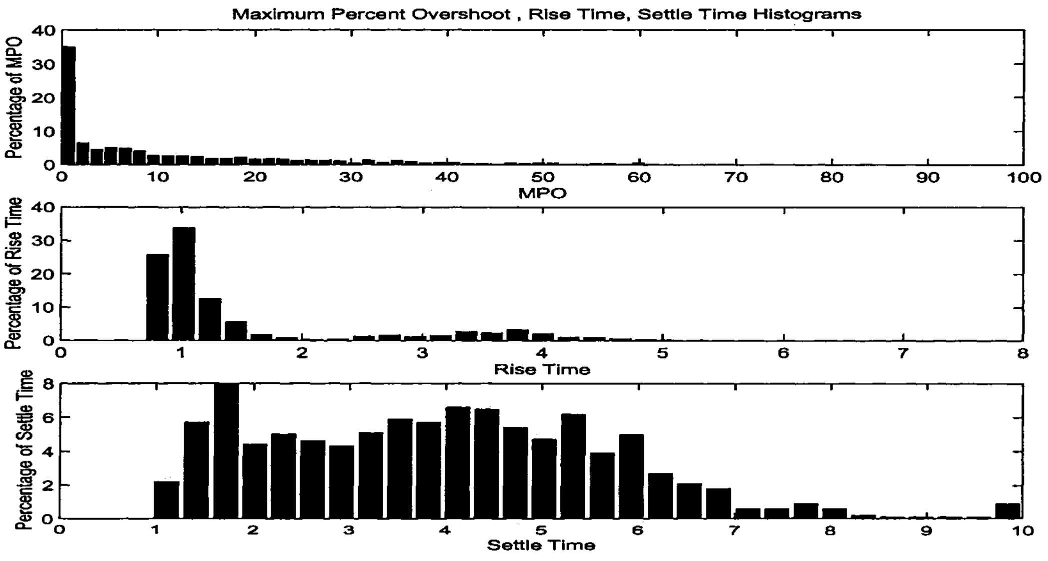


Figure 6.3: *Protocol 1.1.* MPO, RT, and ST histograms.

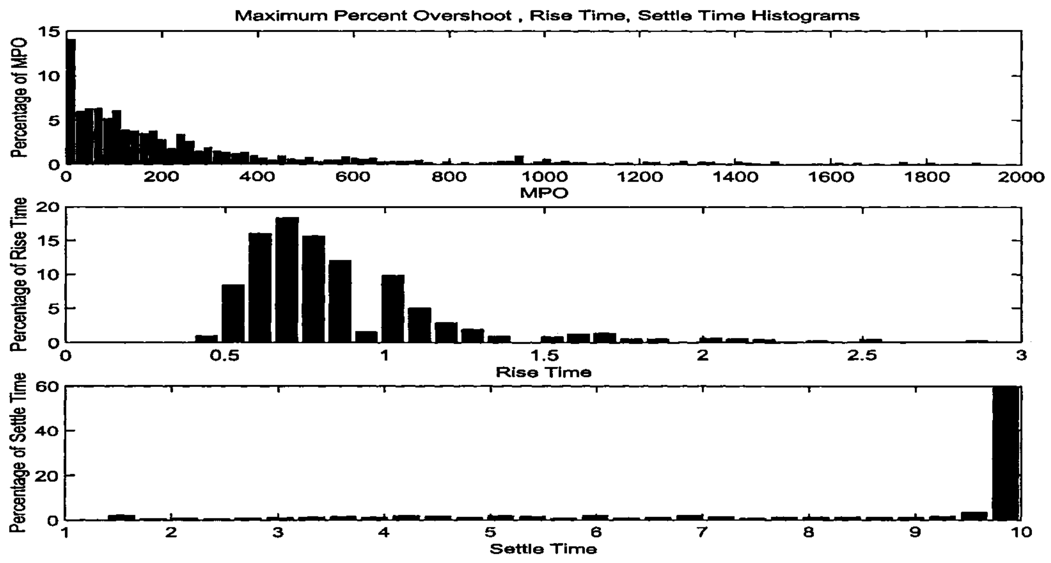


Figure 6.4: *Protocol 2.* MPO, RT, and ST histograms.

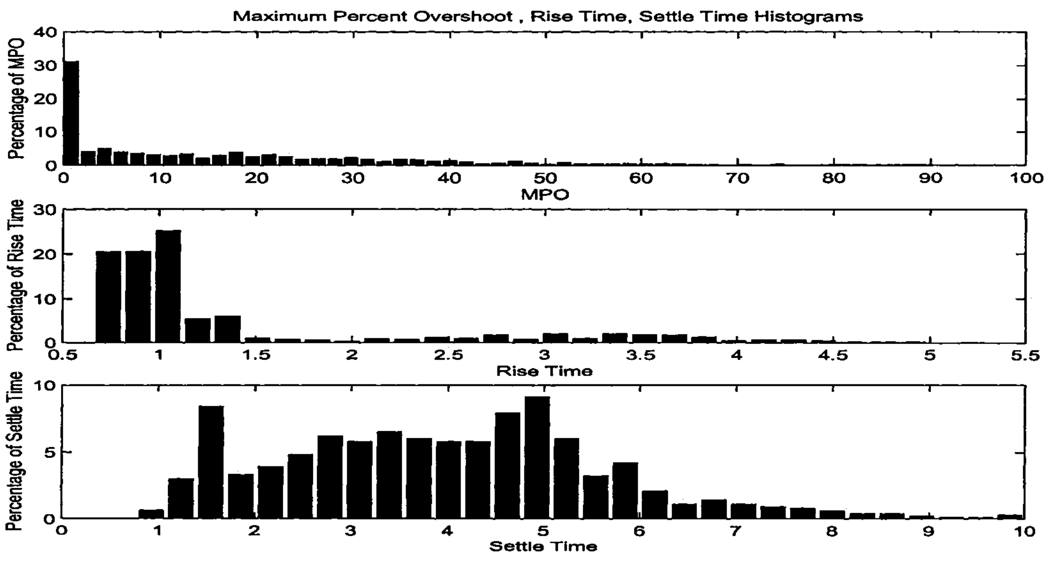


Figure 6.5: *Protocol 2.1.* MPO, RT, and ST histograms.

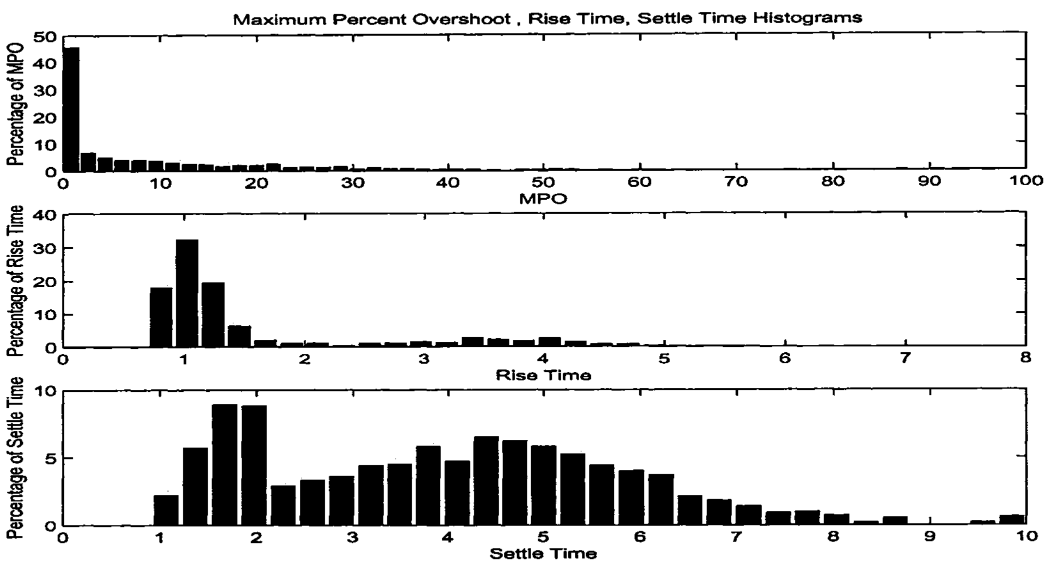


Figure 6.6: *Protocol 2.2.* MPO, RT, and ST histograms.

Table 6.1: Case of Large Load Network.Performance Indices Sample Statistics for 1000 realizations. (Sample Average Rise Time: $\overline{RT}$, Sample Average Settling Time: $\overline{ST}$, Sample Average Maximum Percent Overshoot: $\overline{MPO}$, and their corresponding Sample Standard Deviations: S_{RT}, S_{ST}, S_{MPO}). NEO is the nb. of realizations with $MPO > 20\%$.

$\beta = 0.5$	$\overline{RT}$	S_{RT}	$\overline{ST}$	S_{ST}	$\overline{MPO}$	S_{MPO}	NEO
Protocol 1.0	1.401	0.957	4.021	1.686	19.261%	21.719%	391
Protocol 1.1	1.562	1.083	3.942	1.784	12.046%	16.733%	222
Protocol 2.0	0.865	0.347	8.327	2.538	281.630%	365.840%	855
Protocol 2.1	1.437	0.981	3.948	1.678	16.637%	20.184%	333
Protocol 2.2	1.638	1.128	3.943	1.857	10.026%	16.127%	184

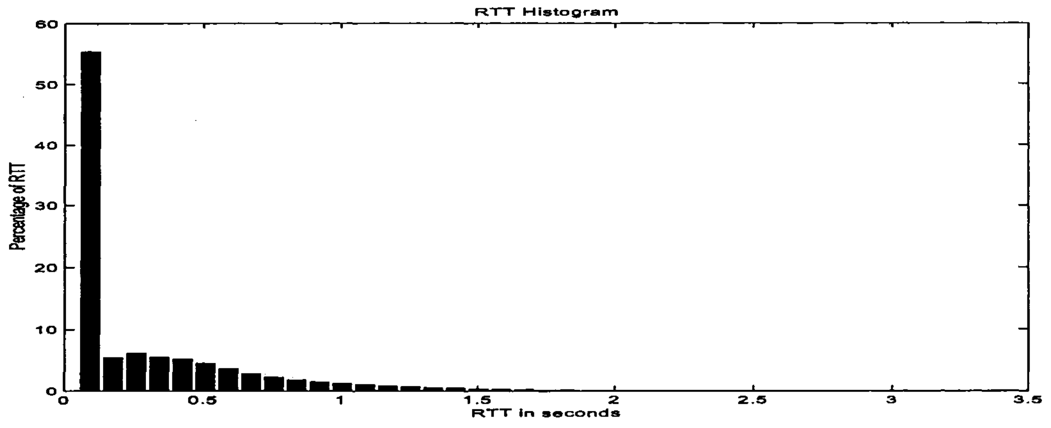


Figure 6.7: Smaller Load Network. RTT histogram for all 5 protocols.

RTT is 0.306 *sec* and its standard deviation is 0.344 *sec*. On average, the RTT is about 3 times the process sampling period T_s.

Control performance measures for *Protocols 1.0, 1.1, 2.0, 2.1, 2.2* are summarized in table 6.2.

6.4 Discussion

These simulations were constructed for a medium-aggressive value of $\beta = 0.5$ in order to compare protocol performance for a constant β.

The results indicate that the mean protocol performance is fairly close between all of the protocols while the analysis showed that Protocol 1.1 had a small edge over the remaining protocols. Recall that in Protocol 1.1 the Application Assumption was that each move was applied at the time of computation and the Additive Feedback Correction involved Measurement Timestamping.

This edge becomes significant in light of the rule-of-thumb that "sloppy" move computation followed up with "precise" sensing should provide control as good as that seen in more "precise" move computation based on Move Acknowledgement. The claim in the analysis is borne out to the extent that Protocol 1.1 fares well in comparison with the other in an average sense even though it relies on "sloppy" computation. However, it is important to note that the standard deviation blurs any

Table 6.2: Case of Smaller Load Network.Performance Indices Sample Statistics for 1000 realizations. (Sample Average Rise Time: $\overline{RT}$, Sample Average Settling Time: $\overline{ST}$, Sample Average Maximum Percent Overshoot: $\overline{MPO}$, and their corresponding Sample Standard Deviations: S_{RT}, S_{ST}, S_{MPO}). NEO is the nb. of realizations with $MPO > 20\%$.

$\beta = 0.5$	$\overline{RT}$	S_{RT}	$\overline{ST}$	S_{ST}	$\overline{MPO}$	S_{MPO}	NEO
Protocol 1.0	2.040	1.153	3.811	1.643	6.448%	13.373%	128
Protocol 1.1	2.159	1.222	3.992	1.636	6.410%	12.885%	113
Protocol 2.0	1.409	1.320	9.489	1.974	666%	667%	804
Protocol 2.1	2.072	1.154	3.808	1.627	5.641%	12.018%	104
Protocol 2.2	2.203	1.219	4.014	1.600	5.588%	11.592%	93

average difference so that they are all closely related for time-varying delays. The sole exception to this observation is Protocol 2.0 which fares badly due to the exclusion of unAcked moves from the process prediction used in the move computation.

Given the near equivalence of all the Protocols it would seem that for constant β either Protocol 1.0 or Protocol 1.1 should be sufficient. The computational simplicity of Protocol 1.1 coupled with its reliance upon sensing for stability improvement make it a natural choice.

Chapter 7

Fuzzy Network State Sensing and Parameter Scheduling

7.1 Introduction

In earlier chapters, a rule-of-thumb analytical study was performed to gain overall insight into major features governing control over IP. It was found that "sloppy" computation followed up by good sensing for feedback correction leads to ease of implementation without sacrificing control capability. This generic point was borne out in simulation and between the simulation and analysis it appears that *Protocol 1.1* is most useful.

In this chapter a fuzzy logic decision system is designed to sense the state of the network. It is used for two things: (i) to sense the delay state of the network, and (ii) to schedule the two SPC parameters k and β (see Sections 1.6.2). In the next section some general characteristics of the Internet delay relevant to the fuzzy logic decision system are presented. These characteristics are gathered from studies that are done by different researchers and that are not part of this thesis.

7.2 Internet delay characteristics

7.2.1 Introduction

Many studies have been done on the end-to-end dynamics of the Internet. RTT adopted as a measure of end-to-end delay, constitutes the main metric considered in those studies. This is no surprise since it is a comprehensive measure that implicitly gives indications about other metrics like Loss, Congestion..., while most Internet applications and protocols are explicitly affected by it.

Many sources contribute to the Internet communication delay and include

- Propagation delay which is the sum, along the physical path, of the times it takes to put a bit on and propagate it through the physical links. This is related mainly to the bandwidth and the physical properties of all the physical links and is deterministic for each link but is stochastic for a logical channel since the physical path of a logical communication channel may change from one packet to another.

- Processing delays that are the amount of time needed to process information along the path. It is stochastic because it depends on the size of the packet and on processing power along the physical paths taken.

- Queueing delays equal to the delays encountered by packets while waiting in queues to be processed along the way to the destination. These delays are stochastic since they depend on the size of the traffic and are highly coupled with the two types of delay mentioned.

Researches have concentrated on approximating RTT distributions or performing predictions of an RTT time series. Unfortunately the conclusions reached are different from one research to another and are sometimes contradictory. For example, while some concluded that the RTT distribution is unimodal [2] others have directly refuted that conclusion [7].

Such disparity is not surprising since end-to-end Internet dynamics depend on many coupled and highly stochastic processes. The Internet itself is also subject to ongoing transformations. What is true for one link may not be true for another and even the behavior of the same link is not the same from one time of day to another. Delay size and variability can change from one continent to another, from one period of the day to another, and from one ISP (Internet Service Provider) to another.

7.2.2 General characteristics of the Internet communication delay

Some points from the different researches relevant to this thesis are highlighted here with assumptions first stated followed by references:

1. Inherent RTT (minimum RTT on a connection) in most cases appears frequently [2][29][7][9][5].

2. There are large geographical and temporal variations in RTT. i.e, RTT changes from a geographical area to another and from a time of the day to another [2][29][7][9][5].

3. RTT distributions have long tails and in most cases have narrow peaks [2] [29] [7][9][5]. They can be approximated by Gamma distributions[7][9].

4. RTT distributions are skewed with fairly spaced means and modes and the mode is a better characteristic of such skewed distributions [2][29].

5. Spikes or excessive RTTs are isolated rare events [2][29][7][9][5].

6. RTTs are in most cases clustered within 10% of the mean , or the mode, or the Inherent delay [2][29] [7][9].

7. RTT distributions change fairly slowly [2][7]. Periods before substantial change vary from 50 minutes [2] down to 2 seconds [7].

8. The jitter in RTT observations is relatively small [2][29]. Over a window of 100 seconds, the range of RTT is within 10 ms [2]. In other words the RTT stays relatively steady for a period of about 100 seconds.

With the admission of the researchers, none of the RTT properties mentioned here are generally valid. However, experience with the Internet and the research conclusions mentioned earlier in this section have yielded a general property of the RTT random process that can be stated with some degree of confidence:

- RTT jitter is small. Within a certain time window, termed a Jitter-Window, equal to the order of 1 second RTT may be considered steady with its variation confined within the order of milliseconds. Such steadiness is occasionally perturbed every now and then in by large delay spikes. These spikes may follow a "jump diffusion" [6] description seen in financial markets and is the subject of ongoing research.

It should be emphasized here that the methods employed in this thesis are independent of the validity of any of the above-mentioned properties. Recall that the goal of this thesis is to build an SPC algorithm and a related communication protocol giving acceptable control performance in the presence of highly unpredictable and variable delays. Therefore, the above discussion on RTT properties serves to justify the straightforward approach taken here to predict RTT in the following chapter.

A fuzzy logic approach is developed in this chapter to sense the state of the RTT and to adapt the SPC parameters to the ever-changing network state. The proposed method will function the best when the following assumption is met

- A causal RTT window exists that yields information about the average RTT in the near future, such that the control parameters may be adjusted in a timely fashion to provide acceptable control performance.

This means then that when a large spike in RTT appears, the controller and the plant lose communication for a time long compared with the process time constant and the controller is assumed to enter a sleep mode [27] wherein no control is attempted.

7.3 Summary of two-parameter SPC

A two-parameter SPC was presented in the Section 1.6.2. This form of SPC has the usual parameter β and an additional parameter k. It was shown Chapter 5 and specifically for *Protocol 1.1* in Section 5.4.2 of the same chapter, that this protocol is stable for $\beta > d - q$ or equivalently $\beta + q > d$. Recall that $q = kT_s$ so that the stability criterion is $\beta + kT_s > d$. Clearly, the zone of stability can be increased by simply increasing the value of k as detailed in Section 5.4.1 for *Protocol 1.0.*

Now, the round trip time between the controller and the sensor is $RTT = \tau_{CA} + \tau_{SC} = 2d$. Rewriting the stability criterion in terms of the RTT gives

$$\beta + q > \frac{RTT}{2} \tag{7.1}$$

and substituting $q = kT_s$

$$\frac{2}{T_s}\,\beta + 2k > \frac{RTT}{T_s} \tag{7.2}$$

Thus the approximate inequality in (7.1) implies that the RTT value after which the closed loop control becomes unstable is the order of $2(\beta + q)$. It is also clear that increasing the second SPC parameter k will expand the zone of stability. This extra degree of freedom is exploited in the fuzzy adaptation to communication delay developed below.

7.4 Fuzzy system design

The fuzzy system is designed to detect the state of the Network delay and suitably adapt the SPC parameter β. In this approach a two dimensional fuzzy logic system that schedules both of k and β is avoided. This is achieved by using the stability analysis to set k such that the control dynamics are inside the zone of stability.
The basic assumption made here to adapt the SPC parameters is

The relative size of the communication delay and/or its variability compared with the sampling period is a sensitive measure of NCS controllability.

Therefore, the state of the network delay is quantified by a relative RTT measure J that represents the ratio of the RTT relative to the sensing, or sampling, period T_s. Note that the sampling period also reflects the plant time constant through the requirement it satisfy $T_s << \textit{Plant Time Constant}$ equal to unity in this thesis. Thus, J can be simply stated as the following three possibilities

- $$J = \frac{RTT_{MAX}}{T_s} \tag{7.3}$$

- $$J = \frac{\overline{RTT}}{T_s} \tag{7.4}$$

  where the Round trip Time is $RTT_n = \tau_{CA_n} + \tau_{SC_n}$, RTT_{MAX} and $\overline{RTT}$ are respectively the largest and the average "Round Trip Times" over an RTT window of size M.

- $$J = \frac{\overline{RTT} + \sqrt{var[RTT]}}{T_s} \tag{7.5}$$

  $var[RTT]$ is the RTT variance over a window of size M.

Note that an RTT sample is estimated every time a measurement packet is received by using the *TimeStamp* and the *Last measured* τ_{CA} fields of the received packet (see section 3.7.2). If J is small, the delay and/or its variability are small compared to T_s and the controller can function in an aggressive mode. If J is medium, the delay and/or its variability are the order of T_s and the controller should function in a mild mode. If J is large, the delay and/or its variability is large compared to T_s and the controller should function in a slow mode.

The plain language statement "how big" is written in term of the three input fuzzy sets

- *Small*: $J \ll \frac{1}{T_s}$.
- *Medium* : $J \simeq \frac{1}{T_s}$.
- *Large*: $J \gg \frac{1}{T_s}$.

The fuzzification membership functions quantify the degree of membership of the input J in each of the input fuzzy sets

$$\mu_{small}(J) = \begin{cases} -\frac{5}{2T_s}J + 1 & \text{if } J \in [0, \frac{2}{5T_s}] \\ 0 & \text{otherwise} \end{cases} \tag{7.6}$$

$$\mu_{medium}(J) = \begin{cases} 5T_sJ - 1 & \text{if } J \in [\frac{1}{5T_s}, \frac{2}{5T_s}] \\ -\frac{5T_s}{4}J + 9 & \text{if } J \in [\frac{2}{5T_s}, \frac{6}{5Ts}] \\ 0 & \text{otherwise} \end{cases} \tag{7.7}$$

$$\mu_{large}(J) = \begin{cases} 0 & \text{if } J < \frac{2}{5T_s} \\ \frac{5T_s}{8}J - \frac{1}{4} & \text{if } J \in [\frac{1}{T_s}, \frac{2}{Ts}] \\ 1 & J \geq \frac{2}{Ts} \end{cases} \tag{7.8}$$

The choice $\frac{2}{T_s}$ as an upper limit for J is associated with a choice of $\beta = 1$ (maximal allowable β) and $k = 0$ (limiting k with no effect on stability) in (7.2).

The input membership functions are depicted in figure 7.1.

The heuristics behind the fuzzy system mentioned earlier can be put into three rules that constitute the fuzzy rule base to be used for inference

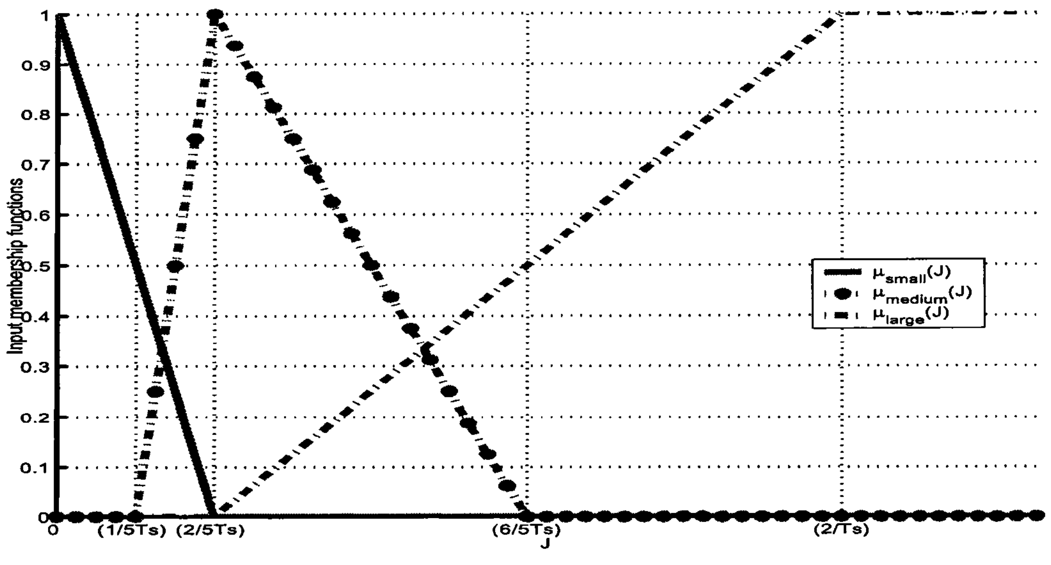


Figure 7.1: Input Membership Functions: $\mu_{smaller}(J)$, $\mu_{equal}(J)$, and $\mu_{larger}(J)$.

- Rule 1: if J is small then β is small. *(Aggressive control).*
- Rule 2: if J is medium then β is mild. *(Mild control).*
- Rule 3: if J is large then β is large. *(Slow control).*

The fuzzification membership functions used to quantify the degree of membership of the output β to each of the output fuzzy sets (linguistically labeled "aggressive", "mild", and "slow" in the rule base) are

$$\mu_{aggressive}(\beta) = \begin{cases} -2.5\beta + 1 & \text{if } \beta \in [0, 0.4] \\ 0 & \text{otherwise} \end{cases} \tag{7.9}$$

$$\mu_{mild}(\beta) = \begin{cases} 5\beta - 1.5 & \text{if } \beta \in]0.3, 0.5] \\ -4\beta + 3 & \text{if } \beta \in]0.5, 0.75] \\ 0 & \text{otherwise} \end{cases} \tag{7.10}$$

$$\mu_{slow}(\beta) = \begin{cases} \frac{10}{3}\beta - \frac{7}{3} & \text{if } \beta \in]0.7, 1.0] \\ 0 & \text{Otherwise} \end{cases} \tag{7.11}$$

The output membership functions are depicted in figure 7.2.

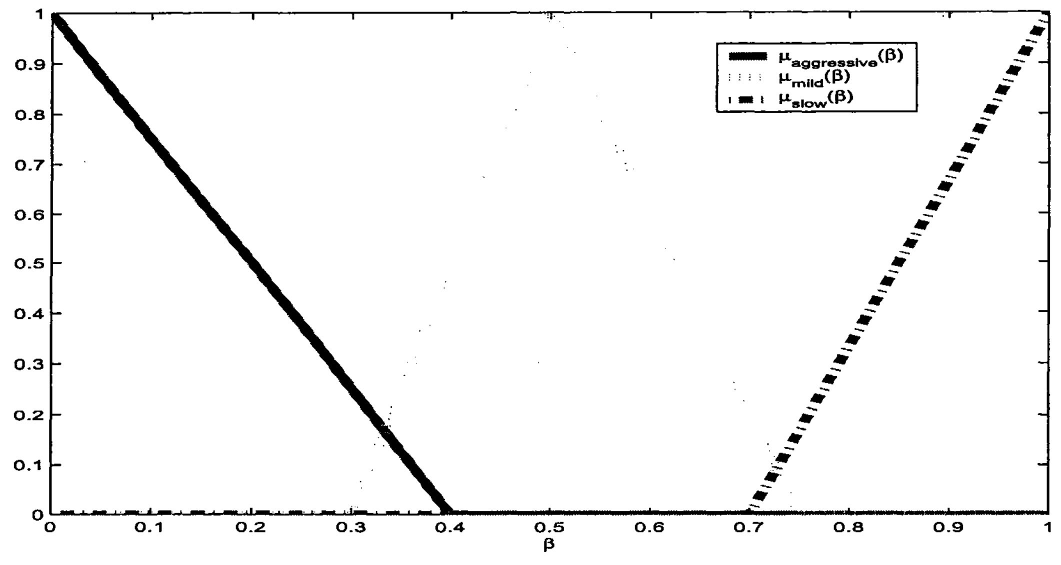


Figure 7.2: Output Membership Functions: $\mu_{small}(\beta)$, $\mu_{mild}(\beta)$, and $\mu_{large}(\beta)$.

The Center-Of-Gravity (COG) [28] method is used in the defuzzification of the output. The values of the output area COGs are chosen empirically to be 0.25, 0.5, and 1.0. This choice guarantees that the most aggressive value of β would be 0.25 in order to avoid bangbang and quasi-bangbang control associated with $\beta \rightarrow 0$. Note that the value of $\beta = 1.0$ (infinite control prediction horizon) corresponds to closed-loop dynamics equivalent to the constant-input open-loop response dynamics.

7.4.1 Adapting the Parameter k

The value of k is calculated using equation (7.2). For stability

$$J < 2(\frac{\beta}{T_s} + k)$$

Given $\beta \leq 1$, the following two rules are used for calculating k and maintain stable control

$$k = \begin{cases} \beta & \text{if } J <= \frac{2}{T_s} + 2 \\ (\frac{J}{2} - \frac{1}{T_s}) & \text{if } J > \frac{2}{T_s} + 2 \end{cases} \tag{7.12}$$

A more conservative control can be obtained by scheduling k more conservatively as follow

$$k = \begin{cases} \beta & \text{if } J <= \frac{2}{T_s} \\ (J - \frac{2}{T_s}) & \text{if } J > \frac{2}{T_s} \end{cases} \tag{7.13}$$

The conservative scheduling of k in equation 7.13 is adopted in the simulations to follow.

7.5 Simulation results

In the next subsections simulation results are presented. Gains in control performance obtained from the fuzzy adaptation developed above and the two-parameter SPC are examined.

7.5.1 Simulation results: large load network

The same simulation scheme used in *Chapter 6* and introduced in *Section 6.2* is used here. Thus, all the results presented here are subjected to the same RTT distribution shown in Figure 6.1. The same RTT distribution is also shown in Figure 7.3 along with the corresponding J measure distribution with J chosen to be

$$J = \frac{\overline{RTT}}{T_s}$$

where $\overline{RTT}$ corresponds to the mean RTT over a window of 20 RTT samples (which corresponds roughly to 2 seconds window width).

The average RTT is 0.3625 *sec* and its standard deviation is 0.2959 *sec*. On average, the RTT is about 3.6 times the process sampling period T_s.

Note that *Protocol 1.1* is chosen as the protocol for control tasks scheduling and control data communication. Also note that the SPC utilized in *Chapter 6* has $\beta = 0.5$ and $k = \beta$ which corresponds to the one-parameter SPC introduced in [14].

Fuzzy Scheduled Two-Parameter SPC

Control performance indices are shown in Figure 7.5. The histograms of the scheduled parameters β and k are shown in Figure 7.4. The average of β is 0.7101 and k is 1.0378.

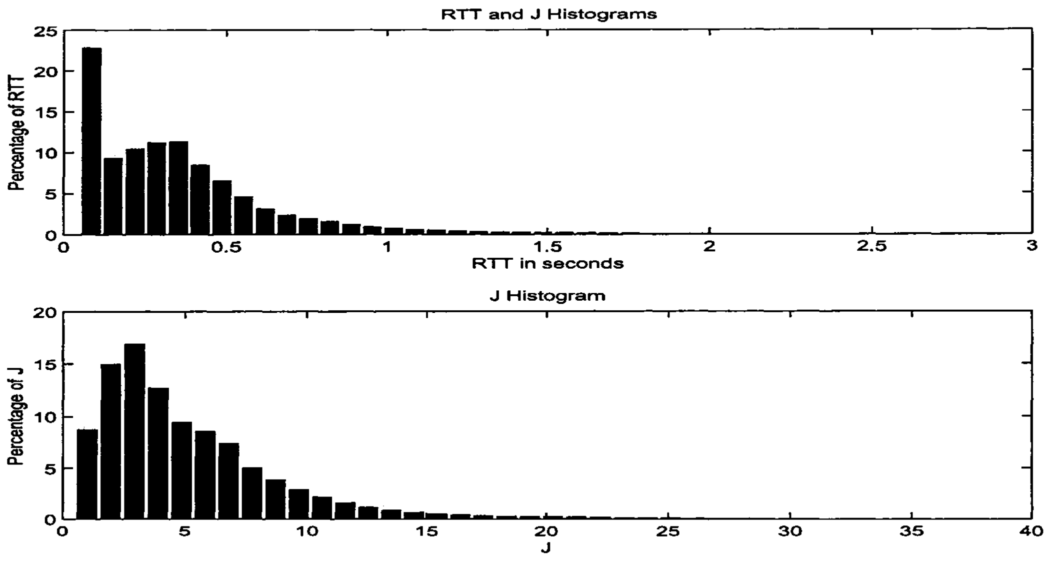


Figure 7.3: RTT and J histograms. The RTT distribution is the same as the one in Figure 6.1.

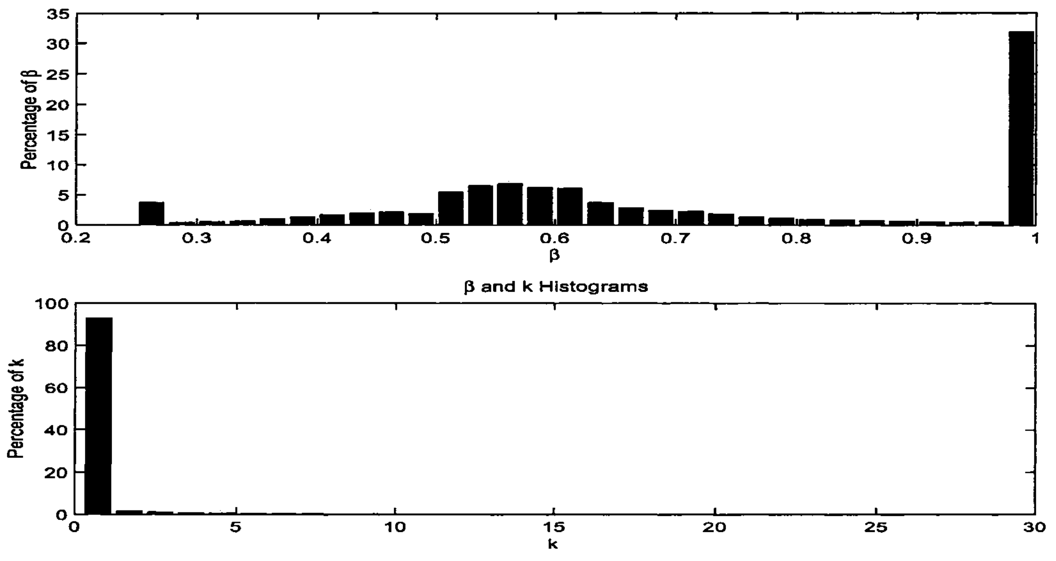


Figure 7.4: Fuzzy scheduled two-parameter SPC. β and k histograms under RTT distribution of figure 7.3.

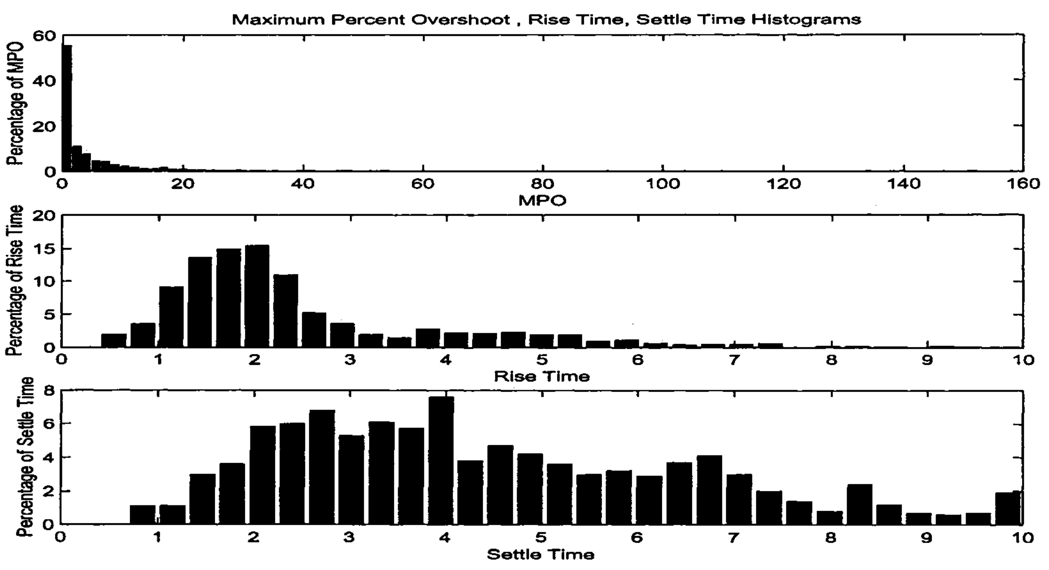


Figure 7.5: Fuzzy scheduled two-parameter SPC. Control Performance Indices under RTT distribution of figure 7.3.

Fuzzy Scheduled One-parameter (β) SPC

In this subsection the results are presented for one-parameter SPC [14]. Its parameter β is scheduled via the fuzzy system introduced in this chapter. Control performance indices are shown in Figure 7.7. The scheduled one parameter β has its histograms shown in Figure 7.6. Since the RTT distribution is the same as the previous subsection β average is still 0.7101.

Average β

This subsection shows results when one-parameter SPC is used while its parameter β is kept constant and equal to the average β from the previous two subsections ($\beta = 0.7101$). The average β control performance indices are shown in Figure 7.8.

Summary table

Table 7.1 shows a summary of the performance indices.

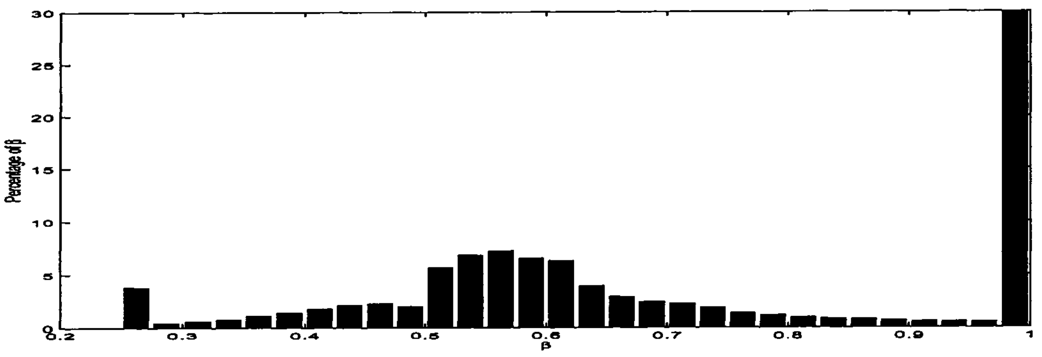


Figure 7.6: Fuzzy scheduled one-parameter SPC. β histogram under RTT distribution of figure 7.3.

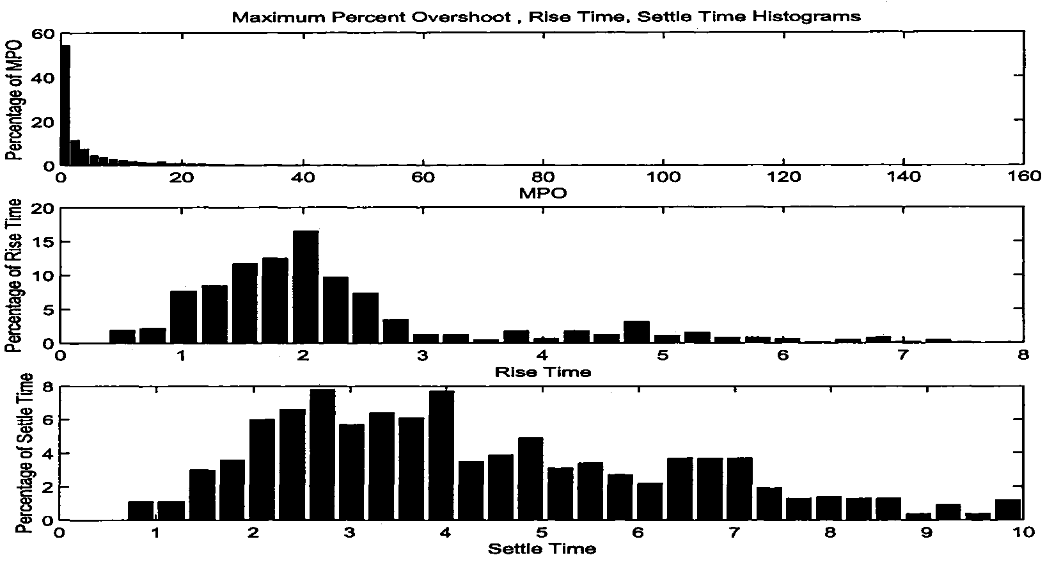


Figure 7.7: Fuzzy scheduled one-parameter SPC. Control Performance Indices under RTT distribution of figure 7.3.

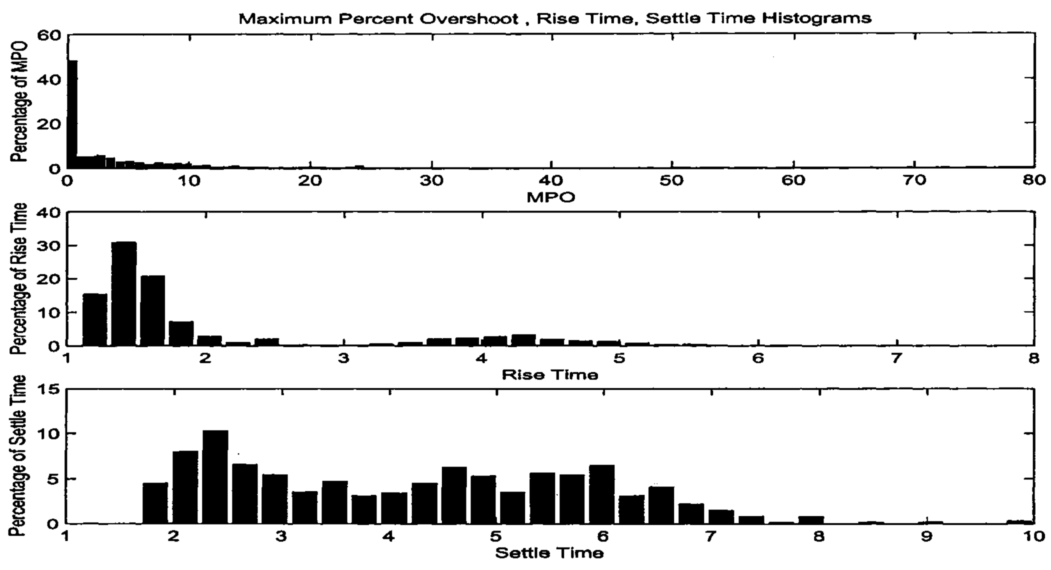


Figure 7.8: One-parameter SPC:$\beta = 0.7101$. Control Performance Indices under RTT distribution of figure 7.3.

7.5.2 Simulation results: smaller load network

The simulation scheme used in *Chapter 6* and introduced in *Section 6.3* is used here. The results presented in this subsection is subject to the same RTT distribution shown in Figure 6.7. The same RTT distribution is also shown in Figure 7.9 along with the corresponding J measure distribution where J is chosen to be

$$J = \frac{\overline{RTT}}{T_s}$$

where $\overline{RTT}$ corresponds to the mean RTT over a window of 20 RTT samples (This corresponds roughly to 2 seconds).

The average RTT is 0.306 *sec* and its standard deviation is 0.344 *sec*. On average, the RTT is about 3 times the process sampling period T_s.

The histograms of the scheduled β and k for the two-parameter SPC are shown in figure 7.10. The average of the scheduled β is 0.56 and that of k is 0.73. Note here that the histogram of the 1-parameter SPC β and its average is the same as the two-parameter SPC, thus there is no need to show that histogram.

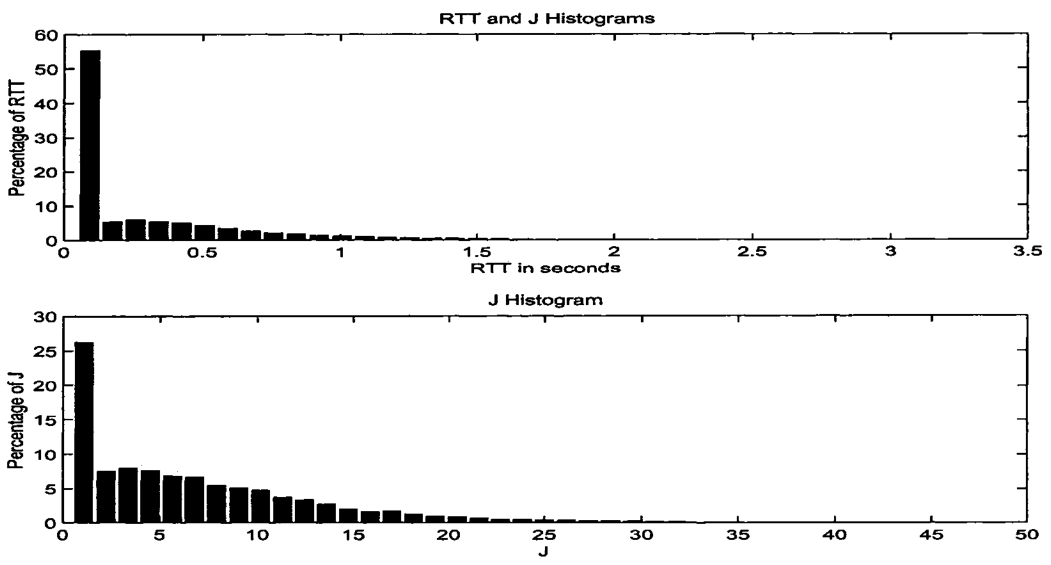


Figure 7.9: RTT and J histograms. The RTT distribution is the same as the one in Figure 6.7.

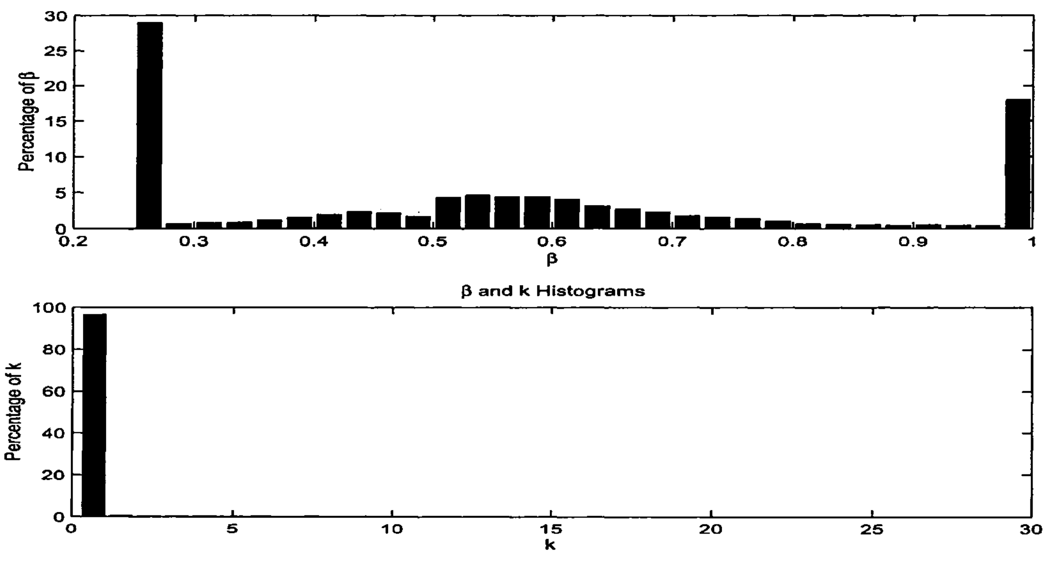


Figure 7.10: β and k histograms under the RTT distribution of Figure 7.9.

Table 7.1: Large Load Network. RTT distribution of figure 7.3. Performance Indices Sample Statistics. (Sample Average Rise Time: $\overline{RT}$, Sample Average Settling Time: $\overline{ST}$, Sample Average Maximum Percent Overshoot: $\overline{MPO}$, and their corresponding Sample Standard Deviations: S_{RT}, S_{ST}, S_{MPO}). NEO is the nb. of realizations with $MPO > 20\%$.

Protocol 1.1	$\overline{RT}$	S_{RT}	$\overline{ST}$	S_{ST}	$\overline{MPO}$	S_{MPO}	NEO
$\beta = 0.5$	1.562 sec	1.08 sec	3.942 sec	1.78 sec	12.04%	16.733%	222
Fuzzy 1-Par SPC	1.765 sec	1.23 sec	4.026 sec	1.95 sec	9.68%	15.65%	154
Fuzzy 2-Par SPC	1.760 sec	1.23 sec	4.014 sec	1.94 sec	9.66%	15.65%	153
$\beta = 0.7101$	1.625 sec	1.09 sec	3.967 sec	1.72 sec	10.97%	15.54%	202

Only a summary of the simulations Performance Indices under the RTT distribution of Figure 7.9 is presented in Table 7.2.

7.6 Discussion of results

Tables 7.1 and 7.2 are statistical summaries of the control performance of the previous subsections. The former involves control within a heavily loaded network with a delay mean and standard deviation about four times the sampling period. The latter models a less heavily loaded network with delay mean and standard deviation about three times the sampling period.

The first row of these tables corresponds with the results in Sections 6.2 and 6.3 for the case of *Protocol 1.1*. The "benchmark" choice of $\beta = 0.5$ was chosen because it provides good control performance over a wide variety of zero delay control applications including multi-input/multi-output (MIMO) and nonlinear processes applications [14].

The improvements made by the fuzzy adaptation can be seen from the pair of tables by comparing the constant β results in rows 1 and 4 with the fuzzy scheduled one and two-parameter SPC presented in rows 2 and 3. It is clear from the tables that the most sensitive parameter to measure the effect of fuzzy parameter adaptation is

Table 7.2: Smaller Load Network. RTT distribution of figure 7.9. Performance Indices Sample Statitics, (Sample Average Rise Time: $\overline{RT}$, Sample Average Settling Time: $\overline{ST}$, Sample Average Maximum Percent Overshoot: $\overline{MPO}$, and their corresponding Sample Standard Deviations: S_{RT}, S_{ST}, S_{MPO}). NEO is the number of realizations with $MPO > 20\%$.

Protocol 1.1	$\overline{RT}$	S_{RT}	$\overline{ST}$	S_{ST}	$\overline{MPO}$	S_{MPO}	NEO
$\beta = 0.5$	2.159 sec	1.22 sec	3.992 sec	1.63 sec	6.41%	12.88%	113
Fuzzy 1-Par SPC	2.596 sec	1.02 sec	3.439 sec	1.19 sec	1.63%	7.61%	14
Fuzzy 2-Par SPC	2.644 sec	1.06 sec	3.456 sec	1.20 sec	1.49%	7.52%	13
$\beta = 0.56$	2.292 sec	1.21 sec	4.058 sec	1.62 sec	4.93%	10.58%	87

the number of extreme overshoots (NEOs show > 20% overshoot). These extreme events are significantly reduced for the fuzzy scheduled SPC in both of the network simulations. What is particularly interesting is the order of magnitude reduction of the NEO for only a 25% change in the delay characteristics between the simulations. The advantage of two-parameter SPC appears in the less heavily loaded network as a factor of two reduction of the MPO coefficient over 1-parameter SPC. This is due to the adaptation of the second parameter k influencing the *local* histogram spread around the mean overshoot. The local nature of this gain has no effect on the number of events in the tails of the distribution (NEOs) which is essentially unchanged.

The results constructed here have been found for the case where the delays significantly exceed the sampling time in contrast with the literature where it is assumed that the delays are the order of the sampling time. The effect of lengthening the delays is apparent from the control performance which approximates the zero delay open loop response dynamic. Thus, the control performance at these longer delays seen in the tables represents *good control performance.* Such a statement can be made since it is difficult to even attain the zero delay open loop response dynamics in the presence of delays that are much longer than the sampling time. It is clear from the constant delay global stability Section 5.4.1 where the counterintuitive result showed that even a choice of $\beta = 1$ may lead to instability in the presence of delays.

The main goal of this chapter was to show that a very simple fuzzy adaptation scheme is able to show good improvement of control performance solely based on a short window of past RTT delay. The significance of this goal is that the fuzzy adaptation is distribution independent while remaining simple. The independence from RTT distributions means that the method is capable of adaptation to nonstationary RTT distributions which are the norm rather than the exception.

Chapter 8

Derivation of Proportional-Integral (PI) control and SPC Equivalence

8.1 Introduction

Proportional-Integral (PI) control is widely used in industry. In fact more than 85% of industrial controllers continue to use PI control [10]. Gain scheduling of PI control for networked control purposes was, and still is, the subject of many studies [1][33] [35]. In those studies empirical scheduling based on offline simulations was implemented.

The ability of SPC to adapt to networked systems is due to its robustness to delays stemming from its usage of prediction. Furthermore, the continuous model analysis presented in Chapters 4 and 5 led to an understanding of the dependence of the stability zone on the two parameters and made their adaptation possible.

A question that arises is then, "Can other linear control methods, like PI, benefit from the power of model prediction and be used with equal capability in control over networks?" In other words, can the results found here for SPC network control of an LTI plant be extended to existing PI installations?

The transferral of the SPC results to PI control is made possible by a general PI-SPC *discrete* equivalence. A relationship between the PI parameter K_p and T_i and SPC parameters k and β, and communication delays can be found such that the SPC and PI moves are *identical*. This leads to a time-varying equivalence due to the variation between moves of the communication delay. It is important to remember that this equivalence is really based on the existence of a model from which the SPC parameters and ultimately the PI parameter are found. Hence, PI is being improved through the presence of a plant model utilized to determine the SPC parameters.

8.2 PI-SPC equivalence - Conventional control case.

This section shows the derivation of a PI-SPC discrete equivalence in the case where control entities are directly connected with no delay among them. The general case of a two-parameter ($\beta = 1 - \alpha$ and k) SPC controller would be considered first. The one-parameter case follows in a straightforward way. Figure 8.1 shows the input and output of the process under SPC control up to the present time $t_n = n \times T_s$. All the moves $[\Delta u_j \ , \ j = \ 0 \ \cdots \ n-1]$ are calculated and applied, and the SPC and the move Δu_n are in the process of being computed.

Now, the trajectory $\hat{y}$ would be the output of the plant resulting from applying all the moves $[\Delta u_j \ , \ j = \ 0 \ \cdots \ n-1]$. Assume that there exists a PI controller that would have had all its moves ΔQ_j equivalent to the moves Δu_j already taken by the SPC. In other words, assume that there exists a PI controller that was exactly equivalent to the SPC up to the current time t. This means that all old moves are as follows

$$\Delta Q_j = \Delta u_j \ \ , \ j = \ 0 \cdots n-1$$

The SPC and PI controllers are used to calculate the moves Δu_n and ΔQ_n respectively. These moves would be calculated as

$$SPC : \Delta u_n \ = \ \frac{e_\alpha^n}{k} \tag{8.1}$$

$$PI \ : \Delta Q_n \ = \ \Delta t K_p [\frac{e^n - e^{n-1}}{\Delta t} + \frac{1}{T_i} e^n] \tag{8.2}$$

where $\Delta t = T_s$ is the sampling period and

$$e_\alpha^n \ = \ r - \hat{y}_\alpha^n \tag{8.3}$$

$$e^{n-1} \ = \ r - y^{n-1} \tag{8.4}$$

$$e^n \ = \ r - y^n \tag{8.5}$$

and r is the set-point reference.

The question the arises now is: "Given that we know β and k, is it possible to choose K_p and T_i so that there is an equivalence between the PI and the SPC moves?"

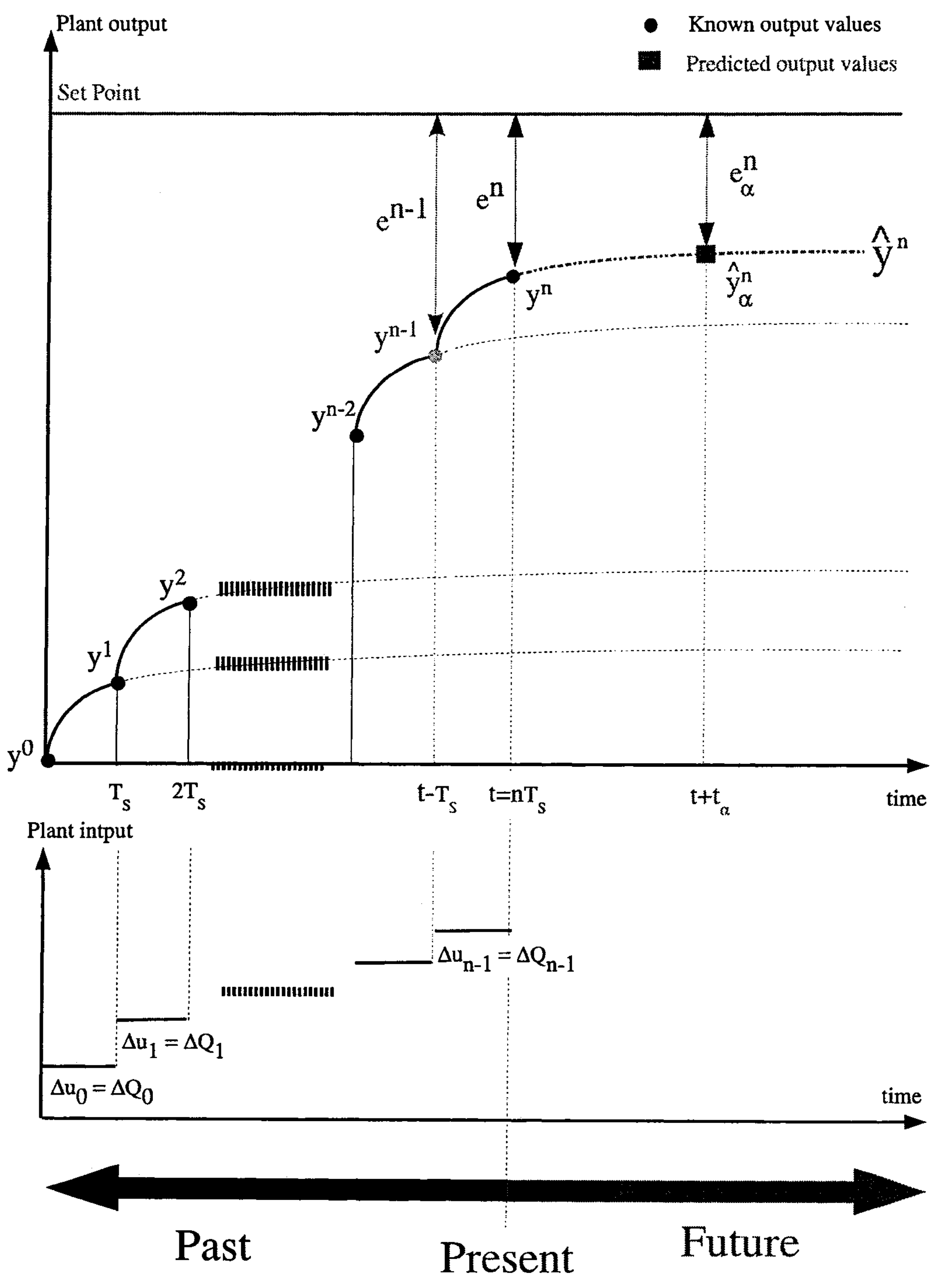


Figure 8.1: Notation and timing diagram. PI-SPC equivalence. No delay.

This means what values of K_p and T_i would yield

$$\Delta Q_n = \Delta u_n$$

or alternatively

$$K_p[e^n - e^{n-1} + \frac{\Delta t}{T_i} e^n] = \frac{e^n_\alpha}{k} \tag{8.6}$$

Using convolution gives the future prediction

$$\hat{y}^n_\alpha = \sum_{j=0}^{n-1} \Delta u_j p(t + t_\alpha - t_{A_j}) \tag{8.7}$$

$$y^n = \sum_{j=0}^{n-1} \Delta u_j p(t - t_{A_j}) \tag{8.8}$$

$$y^{n-1} = \sum_{j=0}^{n-2} \Delta u_j p(t - \Delta t - t_{A_j}) \tag{8.9}$$

Thus

$$\begin{aligned}
e^n - e^{n-1} &= y^{n-1} - y^n \\
&= \sum_{j=0}^{n-2} \Delta u_j p(t - \Delta t - t_{A_j}) - \sum_{j=0}^{n-1} \Delta u_j p(t - t_{A_j}) \\
&= \Big[\sum_{j=0}^{n-2} \Delta u_j [p(t - \Delta t - t_{A_j}) - p(t - t_{A_j})]\Big] - \Delta u_{n-1} p(t - t_{A_{n-1}}) \\
&= \Big[\sum_{j=0}^{n-2} \Delta u_j [1 - e^{-(t-\Delta t - t_{A_j})} - 1 + e^{-(t-t_{A_j})}]\Big] - \Delta u_{n-1} p(\Delta t) \\
&= \Big[\sum_{j=0}^{n-2} \Delta u_j e^{-(t-t_{A_j})}(1 - e^{\Delta t})\Big] - \Delta u_{n-1}(1 - e^{-\Delta t}) \\
&= \Big[e^{\Delta t} \sum_{j=0}^{n-2} \Delta u_j e^{-(t-t_{A_j})}(e^{-\Delta t} - 1)\Big] - \Delta u_{n-1} e^{\Delta t} e^{-\Delta t}(1 - e^{-\Delta t}) \\
&= \Big[e^{\Delta t} \sum_{j=0}^{n-2} \Delta u_j e^{-(t-t_{A_j})}(e^{-\Delta t} - 1)\Big] \\
&\quad + e^{\Delta t} \Delta u_{n-1} e^{-(t-t_{A_{n-1}})}(e^{-\Delta t} - 1) \\
&= -e^{\Delta t}(1 - e^{-\Delta t})\Big[\sum_{j=0}^{n-1} \Delta u_j e^{-(t-t_{A_j})}\Big]
\end{aligned} \tag{8.10}$$

Now

$$
\begin{aligned}
\Delta Q_n &= K_p[e^n - e^{n-1} + \frac{\Delta t}{T_i} e^n] \\
&= -K_p e^{\Delta t}(1 - e^{-\Delta t})\Big[\sum_{j=0}^{n-1} \Delta u_j e^{-(t-t_{A_j})}\Big] + \frac{K_p \Delta t}{T_i}\Big[r - \sum_{j=0}^{n-1} \Delta u_j p(t - t_{A_j})\Big]
\end{aligned}
$$

or

$$
\begin{aligned}
\Delta Q_n &= [-K_p e^{\Delta t}(1 - e^{-\Delta t}) + \frac{K_p \Delta t}{T_i}]\Big(\sum_{j=0}^{n-1} \Delta u_j e^{-(t-t_{A_j})}\Big) \\
&\quad + \frac{K_p \Delta t}{T_i}\Big(-\sum_{j=0}^{n-1} \Delta u_j \; + \; r\Big)
\end{aligned}
\tag{8.11}
$$

It is easy to show that

$$
\Delta u_n = [\frac{\alpha}{k}]\Big(\sum_{j=0}^{n-1} \Delta u_j e^{-(t-t_{A_j})}\Big) + \frac{1}{k}\Big(-\sum_{j=0}^{n-1} \Delta u_j \; + \; r\Big) \tag{8.12}
$$

An equivalence between equations (8.11) and (8.12) becomes possible when

$$
K_p = \frac{\beta}{k}\frac{1}{e^{\Delta t} - 1} \tag{8.13}
$$

$$
T_i = \frac{\beta \Delta t}{e^{\Delta t} - 1} \tag{8.14}
$$

It is straightforward to find that for the one-parameter SPC the equivalence necessitates

$$
K_p = \frac{1}{e^{\Delta t} - 1} \tag{8.15}
$$

$$
T_i = \frac{\beta \Delta t}{e^{\Delta t} - 1} \tag{8.16}
$$

Note that T_i is the same in both cases. This could have been anticipated knowing that k, is a gain term.

8.3 PI-SPC equivalence - Networked Control Systems case.

The analysis in the previous section is generalized to the nonzero delay case.

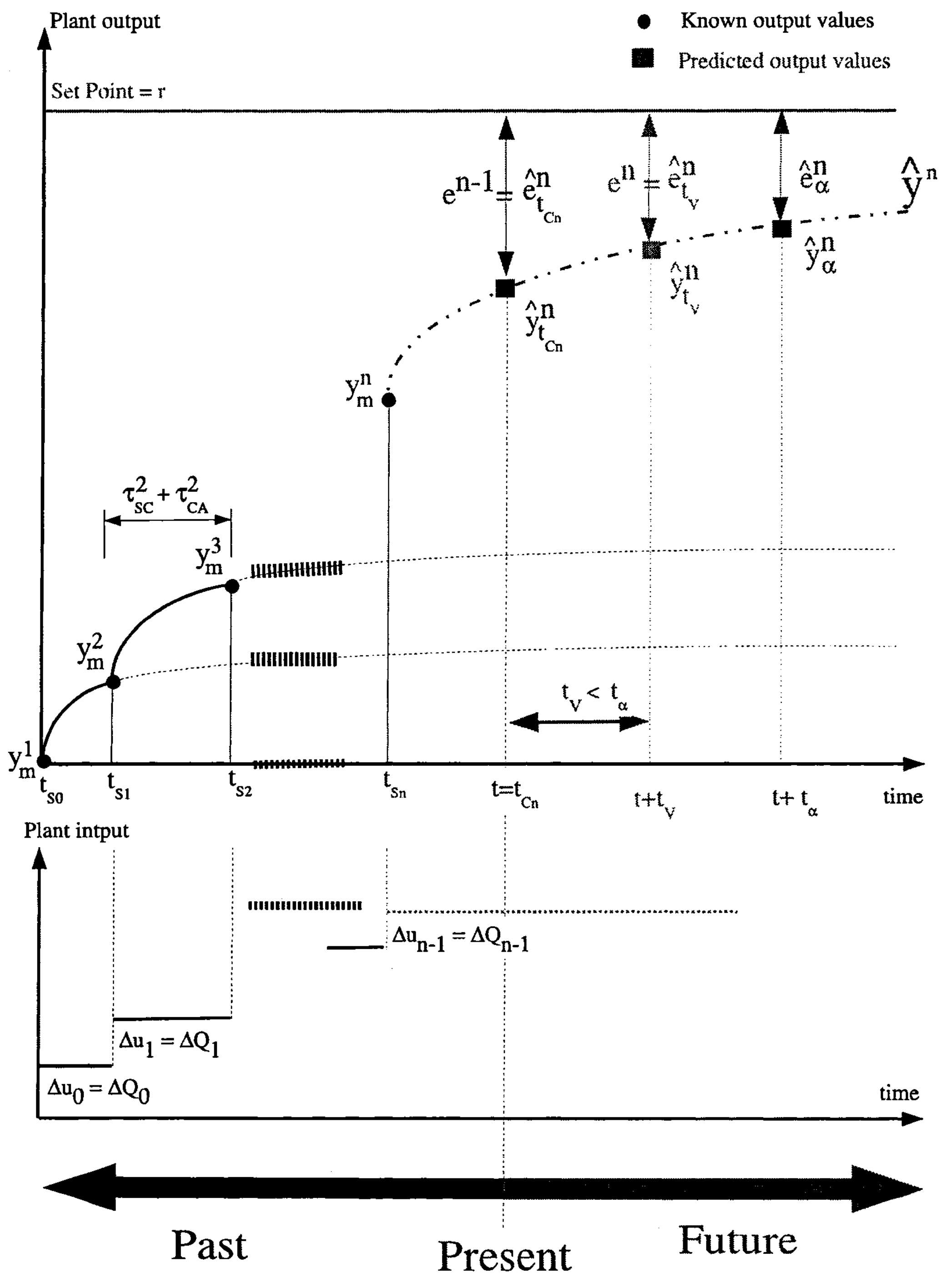


Figure 8.2: Notation and timing diagram. Delay case. The trajectory $\hat{y}^n$ would be the output of the plant resulting from applying all the moves $[\Delta u_j\ ,\ j =\ 0\ \cdots\ n-1]$.

In Figure 8.2 the input and output of the process under SPC control up to the present time $t = t_{S_n}$ is shown. The controller has just received measurement y_m^n. So the SPC controller assumes that all the moves $[\Delta u_j \ , \ j = \ 0 \ \cdots \ n-1]$ were calculated and applied and it is assumed that the respective actuation times are the corresponding calculation times t_{C_j}. These assumptions are true for *Protocols 1* and *1.1*. Nevertheless the derivations to be presented in this section can be easily reproduced for the other protocols.

Assume that the SPC controller is in the process of calculating the move Δu_n. The trajectory $\hat{y}^n$ would be the output of the plant resulting from considering all the moves $[\Delta u_j \ , \ j = \ 0 \ \cdots \ n-1]$ being applied at the respective times t_{C_j}.

The two-parameter SPC calculates Δu_n as follow

$$SPC : \Delta u_n \ = \ \frac{\hat{e}_\alpha^n}{k} \tag{8.17}$$

where $\hat{e}_\alpha^n = r - \hat{y}_\alpha^n + \phi_n$ is the predicted error at the future date $t + t_\alpha$ as illustrated in Figure 8.2.

Assume that there exists a PI controller that would have had all its moves ΔQ_j equivalent to the moves Δu_j already calculated by the SPC. In other words, assume that there exists a PI controller that was exactly equivalent to the SPC *up to the current time t*. This means that all old moves are

$$\Delta Q_j = \Delta u_j \ \ , \ j = \ 0 \cdots n-1$$

The PI, like the SPC, is now trying to calculate the move ΔQ_n. The calculation of a PI control move necessitates "two error values" (e^n and e^{n-1}) and would be done as follow

$$PI \ : \Delta Q_n \ = \ \Delta t \ K_p \ [\frac{e^n - e^{n-1}}{\Delta t} + \frac{1}{T_i} \ e^n] \tag{8.18}$$

where Δt is not necessarily the sampling period but it depends on the choice of the errors e^n and e^{n-1} and their timing.

Notice that the SPC uses the prediction of the future output and that the error $\hat{e}_\alpha^n$ used by the SPC is based on the estimation of the future . Thus an equivalent PI controller must use the output prediction. In other words the two errors must be predicted future errors and must lie on the $\hat{Y}$ prediction curve. A choice of two predicted errors that complies with this last point is

$$e^{n-1} = \hat{e}^n_{tC_n} = r - \hat{y}^n_{tC_n} + \phi_n \tag{8.19}$$

$$e^{n} = \hat{e}^n_{t_V} = r - \hat{y}^n_{V} + \phi_n \tag{8.20}$$

r being the set-point reference. Note that ϕ_n must be consistently used with predictions. These error values are illustrated in Figure 8.2. $\hat{e}^n_{tC_n}$ is the predicted error at the current time. $\hat{e}^n_{t_V}$ is the error t_V seconds into the future, i.e, the predicted error at time $t + t_V$. t_V is some chosen fraction of t_α. This choice leads to Δt being

$$\Delta t = t_V \tag{8.21}$$

After choosing e^{n-1} and e^n, the question is: "Given β and k, is it possible to choose K_p and T_i to provide an equivalence between the PI and the SPC moves?"

Once again, we want K_p and T_i such that

$$\Delta Q_n = \Delta u_n$$

or alternatively

$$K_p \, [\hat{e}^n_{t_V} - \hat{e}^n_{tC_n} + \frac{t_V}{T_i} \, \hat{e}^n_{t_V}] = \frac{\hat{e}^n_\alpha}{k} \tag{8.22}$$

Using convolution to predict the future state

$$\hat{y}^n_\alpha = \sum_{j=0}^{n-1} \Delta u_j \, p(t + t_\alpha - t_{C_j}) \tag{8.23}$$

$$\hat{y}^n_{tC_n} = \sum_{j=0}^{n-1} \Delta u_j \, p(t - t_{C_j}) \tag{8.24}$$

$$\hat{y}^n_V = \sum_{j=0}^{n-1} \Delta u_j \, p(t + t_V - t_{C_j}) \tag{8.25}$$

From which

$$\begin{aligned}
e^n - e^{n-1} &= \hat{y}^n_{t_{C_n}} - \hat{y}^n_V \\
&= \sum_{j=0}^{n-1} \Delta u_j \left[p(t - t_{C_j}) - p(t + t_V - t_{A_j})\right] \\
&= \sum_{j=0}^{n-1} \Delta u_j \left[e^{-(t+t_V-t_{C_j})} - e^{-(t-t_{C_j})}\right] \\
&= \sum_{j=0}^{n-1} [\Delta u_j \; e^{-(t-t_{C_j})} \; (e^{-t_V} - 1)] \\
&= (e^{-t_V} - 1) \sum_{j=0}^{n-1} [\Delta u_j \; e^{-(t-t_{C_j})}] \qquad (8.26)
\end{aligned}$$

and

$$\begin{aligned}
\Delta Q_n &= K_p \, (\hat{e}^n_{t_V} - \hat{e}^n_{t_{C_n}}) + \frac{K_p \, t_V}{T_i} \, \hat{e}^n_{t_V} \\
&= K_p \, (e^{-t_V} - 1) \sum_{j=0}^{n-1} [\Delta u_j \; e^{-(t-t_{C_j})}] \\
&\quad + \frac{K_p \, t_V}{T_i} \left[r - \sum_{j=0}^{n-1} \Delta u_j \; p(t + t_V - t_{C_j}) \; + \phi_n \right] \\
&= \left[K_p \, (e^{-t_V} - 1) + \frac{K_p \, t_V \, e^{-t_V}}{T_i} \right] \sum_{j=0}^{n-1} [\Delta u_j \; e^{-(t-t_{C_j})}] \\
&\quad + \frac{K_p \, t_V}{T_i} \left(r + \phi_n - \sum_{j=0}^{n-1} \Delta u_j \right) \qquad (8.27)
\end{aligned}$$

It is easy to develop Δu_n in the same way to get It is easy to develop Δu_n in the same way to get

$$\Delta u_n = \left(\frac{\alpha}{k}\right) \sum_{j=0}^{n-1} [\Delta u_j \; e^{-(t-t_{C_j})}] + \frac{1}{k} \left(r + \phi_n - \sum_{j=0}^{n-1} \Delta u_j \right) \qquad (8.28)$$

An equivalence between (8.27) and (8.28) is possible if

$$\begin{aligned}
K_p &= \frac{(e^{-t_V} - 1 + \beta)}{k(e^{-t_V} - 1)} \\
T_i = k \; K_p \; t_V &= \frac{(e^{-t_V} - 1 + \beta) t_V}{(e^{-t_V} - 1)} \qquad (8.29)
\end{aligned}$$

As before with zero delays, it is straightforward to find that for the one-parameter SPC the equivalence necessitates

$$\begin{aligned} K_p &= \frac{(e^{-t_V} - 1 + \beta)}{\beta(e^{-t_V} - 1)} \\ T_i = \beta\, K_p\, t_V &= \frac{(e^{-t_V} - 1 + \beta))t_V}{(e^{-t_V} - 1)} \end{aligned} \tag{8.30}$$

Again note that T_i is independent of the SPC second parameter k which functions as a gain.

Figure 8.3 shows equivalent two-parameter SPC and PI controls under the same RTT conditions. The two SPC parameters are scheduled using the fuzzy scheme of Chapter 7 and the equivalent PI parameters are scheduled according the set of formulae 8.29.

As was stated in the preamble to this section, the real practical relevance of these formulae for K_p and T_i is that they form an *adaptive equivalence* in the presence of time-varying communication delay. In other words, the SPC can use an adaptive scheme like the Fuzzy scheme developed in this thesis and the PI adapts similarly by being equivalent to the SPC.

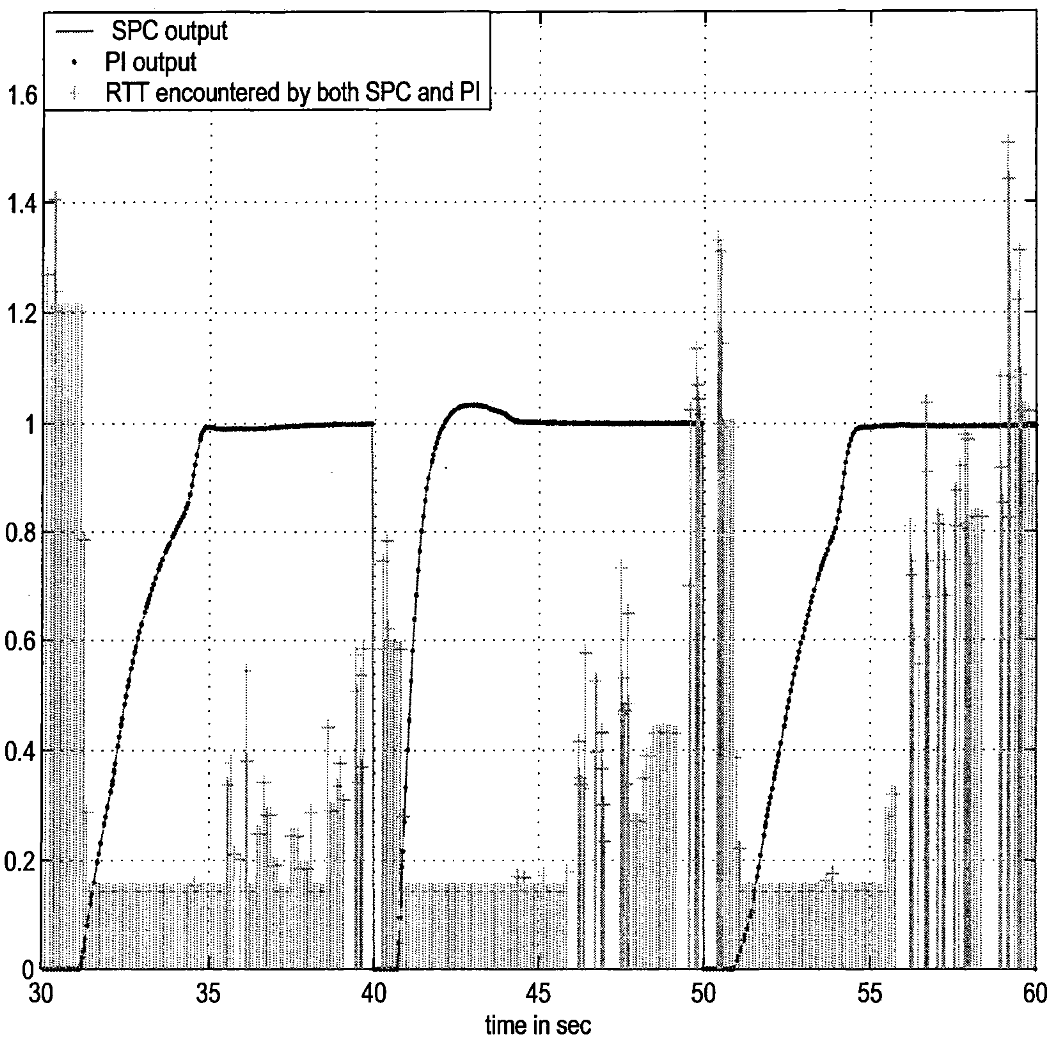


Figure 8.3: Three realizations of equivalent (PI, 2-par SPC) controls under the same RTT conditions.

Chapter 9

Conclusion and Future Work

9.1 Conclusion

The main *goal* of this thesis was to adapt SPC [14] for usage via IP networks without prior knowledge of delay distributions or delay bounds. The central issue was the assessment of the impact of Measurement Timestamping and Move Acknowledements. It was found that Measurement Timestamping enhanced stability through more accurate feedback of the process. On the other hand, Move Acknowledgment regarding the timing of application of moves was found to have little effect. The overall rule-of-thumb that emerged from these observations were that simple PI and SPC control algorithms coupled with accurate sensing for feedback are sufficient to supply good control performance and stability. This result is especially useful in control via IP networks since it was found for the case where the network delays are much longer than the time between process samples. It should be noted that these results have been found to be true for predictive control of a first order, LTI plant.

The direct extension of the SPC results to PI control of the same plant was made possible by the discovery of a PI-SPC equivalence for control of an LTI plant. The analytical equivalence gives an exact timewise adaptation of the PI parameters and identical moves to those found through SPC. While PI control is model independent, the linkage between the SPC and PI parameters allows for the simple transferral of the SPC model dependence to the PI control algorithm.

A fuzzy online adaptation was used to schedule parameters in the SPC algorithm. The primary effects of the fuzzy adaptation are twofold: (i) the control can be made more or less aggressive depending upon the delay size, and (ii) the zone of stability can be maintained in the presence of longer delays. This last point was found to predominate in this work since the delay length is long compared with the plant sampling time. In order to increase the zone of stability of one-parameter SPC [14], a two-parameter SPC generalization was introduced and analyzed.

This study of network control and online adaptation via fuzzy logic of a well-conditioned method like SPC has led to the introduction of a PI control scheme capable of control in the presence of noisy communication delays. The methods of analysis and overall approach involving direct linkages between seemingly disparate control schemes should provide avenues to control in more complex situations.

9.2 Future work

The research in this thesis can be furthered in three directions

- Further adaptation of networked SPC.
  - The adaptation of SPC in this thesis did not assume any delay distributions or bounds. It would be of benefit to exploit the specific characteristics of the communication delays of a specific Internet connection. Such exploitation would make control parameters online scheduling methodologies, like the fuzzy network state sensing scheme presented in this thesis, more efficient. This evidently requires studying the end-to-end characteristics of an Internet connection. The "jump diffusion" theory [6] utilized in financial markets might be useful in explaining some aspects of the Internet delay.
  - The stability zones established in this work can be useful in establishing negotiation schemes between the SPC controller and QoS protocols for setting higher bounds on the communication delay and its jitter.
- Investigation of the applicability of the conclusions reached in this thesis to other control methodologies.
  - Although the conclusions reached were specific to networked SPC control and were extended by equivalence to networked PI control, it would be beneficial to explore how well such conclusions apply to other control schemes, especially the conclusion pertaining to simple control and accurate sensing.
- Investigation of linkages and equivalences among seemingly disparate control schemes and related applications.

- By using the LTI plant model as was done in developing the PI-SPC equivalence, further linkages can be sought. A PID-SPC linkage is of great interest since the PID contains three forms of control (Proportional, Integral, and Derivative) but does not account for the future nor does it depend on any model like the SPC.
- The PI-SPC equivalence and any other linkages that might be developed, can be used to provide control that is based on different control schemes working in symbiosis: switching between such control schemes can be done conveniently depending on the closed-loop system state . For example a future PID that uses a model, like an SPC does, can be developed to be used along a conventional PID. The future PID should be switched on in the control zones where the model is accurate and the sensing might be noisy, while the conventional PID should be switched on in the control zones where the model is not accurate enough.
- The PI-SPC equivalence can be used to develop a method for PI tuning.

Bibliography

[1] Almutairi N. B., Chow M. Y., Tipsuwan Y. "Network-based controlled DC motor with fuzzy compensation." *The 27th annual conference of the IEEE industrial electronics society (IECON 01)*, **3**, 1844-1849. Denver, CO. (2001).

[2] Acharya A., Saltz J. A study of Internet round-trip delay, Technical report, University of Maryland, College Park 20742, 1997.

[3] Baran P., "The beginnings of packet switching: Some underlying concepts", *IEEE Communications Magazine,* July 2002.

[4] Beldiman O. ,Bushnell L., Walsh G.,Wang H., Hong Y., "Perturbations in networked control systems", Proceedings of *ASME-IMECE'01,* New York, NY, USA, 2001.

[5] Bi J., Wu Q., Li Z., "Packet Delay and Packet Loss in the Internet", *In proceedings of the IEEE ISCC'02,* 2002.

[6] Blanco C., Soronow D.,"Jump diffusion processes- Energy price processes used for derivatives pricing and risk management",*Commodities now,* Sep. 2001.

[7] Bovy C. et al., "Analysis of end-to-end delay measurements in the Internet", RIPE 41, Amsterdam, Jan. 2002.

[8] Chan H., Ozguner U. "Closed-loop control of systems over a communications network with queues". *International Journal of Control*, 62(3), 493-510 (1995).

[9] Corlett A. et al. "Satistics of one-way Internet packet delays", 53rd IETF, Minneapolis, March 2002.

[10] Corripio A.B., "Tuning of industrial control systems—2nd edition," *ISA*, 2001.

[11] Cutler C. R., Ramaker B. L. "Dynamic matrix control - a computer control algorithm", *Proc. AIChE 86th Meeting*, Apr. 1979.

[12] Fang Y., Liu G., Wu B., "On Internet-based control system design with clock synchronization", *Proceedings of the 2004 IEEE International Conference on Robotics and Biometrics*, Shenyang, China, Aug. 2004.

[13] Goktas F. Distributed control of systems over communication networks. Ph.D. dissertation, University of Pennsylvania (2000).

[14] Gupta, Y.P., "A simplified predictive control approach for handling constraints through linear programming." *Computers in Industry*, **21**, 255-265, (1993).

[15] Halevi Y., Ray A., "Integrated communication and control systems: Part I Analysis," *Journal of Dynamic Systems, Measurement, and Control,* vol. 110, pp. 367-373, 1988.

[16] Hartman J., "Networked control system co-simulation for co-design: theory and experiments.", M.Sc. Thesis, Case Western Reserve University, Jun. 2004.

[17] Hong S. H., Kim W. H. "Bandwidth allocation scheme in CAN protocol". *IEEE Proceedings-Control Theory and Applications*, 147(1), 37-44 (2000).

[18] Ishii H., Francis B. A., "Stabilization with control networks", *Automatica,* vol. 38, pp. 1745-1751, 2002.

[19] Kember G. C. , Mansour S. E. , Dubay R. "Risk aversion predictive control." *ISA Transactions: Journal of Science and Engineering of Measurement and Automation,* to appear in 2006.

[20] Kember G. C. , Dubay R. , Mansour S. E. , "Continuous analysis of Move Suppressed and Well Conditioned DMC," *ISA Transactions: Journal of Science and Engineering of Measurement and Automation* , 2004.

[21] Kember G. C. , Dubay R. , Mansour S. E. , "On Simplified Predictive Control as a generalization of Least Squares Dynamic Matrix Control," *ISA Transactions: Journal of Science and Engineering of Measurement and Automation,* vol. 44, pp. 345-352, 2005.

[22] Liou L., Ray A., "A stochastic regulator for integrated communication and control systems: Part I- Formulation of control law ," *Journal of Dynamic Systems, Measurement, and Control,* vol. 113, pp. 604-611, 1991.

[23] Liu C., Layland J. , " Scheduling algorithms for multiprogramming in a hard-real-time environment", *Journal of the ACM*, vol. 20, nb. 1, pp. 46-61, 1973.

[24] Luck R., Ray A., "An observer-based compensator for distributed delays ," *Automatica,* vol. 26, pp. 903-908, 1990.

[25] Nilsson J. Real-time control systems with delays. Ph.D. dissertation, Lund Institute of Technology (1998).

[26] Oboe R., "Web-interfaced, force-reflecting teleoperation system," *IEEE Transactions on Industrial Electronics,* vol. 48, pp. 1257-1265, Dec. 2001.

[27] Overstreet J., Tzes A., "An Internet-based real-time control engineerig laboratory", *IEEE Control Systems Magazine*, pp. 19-34, 1999.

[28] Passino K.M., Yurkovitch S., Fuzzy Control,*Addison-Wesley, Addison Wesley Longman,Inc.,* 1998 .

[29] Sanghi D., Agrawala A., Gudmundsson O., Jain B. "Experimental assessment of end-to-end behavior on Internet". In proceedings of IEEE Infocom, 1993.

[30] Soucek S., Sauter T., "Quality of service concerns in IP-based control systems," *IEEE Transactions on Industrial Electronics,* vol. 51, No. 6, Dec. 2004.

[31] Stevens W. R. , *TCP/IP Illustrated, Volume 1, The Protocols.* Addison Wesley, Seventh Printing, Mar. 1996.

[32] Tarn T., Xi N. "Planning and control of Internet-based teleoperation." *Proceedings of SPIE: Telemanipulator and Telepresence technologies V* , vol. 3524, pp. 189-193. Boston, MA, 1998.

[33] Tipsuwan Y., Chow Y. "Gain adaptation of networked mobile robot to compensate QoS deterioration." *The 28th annual conference of the IEEE industrial electronics society (IECON 02)* **4**, 3146-3151, 2002.

[34] Tipsuwan Y., Chow Y. "Control methodologies in networked control systems". *Control Engineering Practice,* **11**, pp. 1099-1111, 2003.

[35] Tipsuwan Y. ,Chow M.Y. "On the gain scheduling for networked PI controller over IP network." *IEEE/ASME transactions on mechatronics*, vol. 9, No. 3, Sep. 2004.

[36] Van Dyke M., Perturbation methods in fluid mechanics, Parabolic Press, 1975.

[37] Walsh G., Ye H., Bushnell L. "Stability analysis of networked control systems", Proceedings of *American control conference,* pp. 2876-2882, Jun. 1999.

[38] Wittenmark B., Trungren N. , "Timing problems in real-time control systems", *American Control Conference, Seattle, Washington.* 1995.

[39] Zhang W., Branicky M. S., Phillips S. M., "Satbility of Networked Control Systems," *IEEE Control Systems Magazine*, February 2001, pp. 84-99, 2001.

[40] "Control in an Information Rich World", Final report of the Panel on Future Directions in Control, Dynamics, and Systems. *IEEE Control Systems Magazine,* vol. 23, number 2, pp. 20-33, Apr. 2003.

[41] The read-out instrumentation signpost [Online]. Available: http://read-out.net/signpost/fieldbus.html [2006, 5 May].

[42] The Network Simulator - ns-2. [Online]. Available: http//www.isi.edu.nsnam/ns/[2006, 5 May]

[43] The Network Time Protocol Version 4 Protocol Specification, [Online] Available: http://www.ietf.org/internet-drafts/draft-ietf-ntp-ntpv4-proto-01.txt [2006, 5 May]

[44] Networked Systems Repository. [Online] Available: http://filer.case.edu/org/ncs/links.htm [2006, 5 May]