International Journal of Enhanced Research in Science Technology & Engineering, ISSN: 2319-7463
Vol. 4 Issue 2, Feb.-2015, pp: (184-193), Impact Factor: 1.252, Available online at: www.erpublications.com

Gain Scheduling PID Controller For Networked Control System with Random Delay

Isra'a L. Salim[1], Osama A. Awad[2]
[1]Networks Engineering Department, College of Information Engineering
[2]Systems Engineering Department, College of Information Engineering
Al-Nahrain University, Baghdad, Iraq

Abstract: Networked Control System (NCS) has recently been a hotspot in the research fields of control theory and control engineering application, where control system components (controller and controlled system or plant) are connected via a shared network. Taking a third-order DC motor as the plant and using the TrueTime toolbox in MATLAB, a NCS model is designed with Ethernet network as the communication channel. Through the simulation, different traffic scenarios on Ethernet network are studied. Different load conditions induce time-varying delays between measurements and control. To face these variations a delay-dependent gain scheduling Proportional - Integral – Derivative (PID) controller has been designed.

Keywords: Ethernet, gain scheduling, NCS, PID controller, TrueTime toolbox.

I. Introduction

One of the major challenges faced by modern control systems is how to integrate control, communication, and computing aspects into different levels of operations [1]. The current motivations is to implement a distributed control systems with a computer-based control system [2]. A Networked Control System (NCS), is one type of distributed control system, is defined as a control system where the control loops are closed via a network, this network is shared with other nodes outside the control system [3, 4]. A closed loop NCS general structure is shown in fig. 1[5]. NCS comprises (1) at local side: sensor, actuator, and plant, (2) at remote side: a controller, all communicated and sharing data (control action and sensor measurement packets) via a communication network. NCS has many advantages comparing with traditional control systems: improve the flexibility, simpler, efficient, quick and easy maintenance, and systems are remotely controlled. Thus, NCS reduces installation and maintenance costs [6] [7]. Nowadays, there are many cheap network technologies and the controllers could be implemented with a cheap microcontrollers, this availability of a shared networks results in flexibility and lower costs [8].

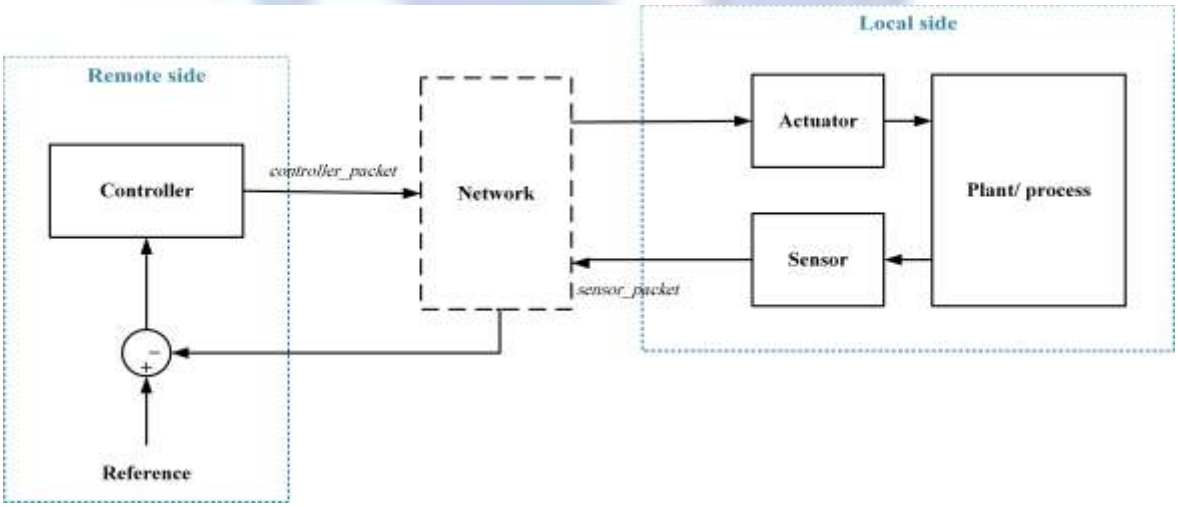

Figure 1: General structure of NCS

However, there are many problems along with NCS itself because of the uncertainties in network communication. The obvious problem is the network-induced time delay, which has a great impact on the stability and the performance of the NCS. Moreover, due to the uncertainty of the network transmission and the restriction of communication bandwidth and data packet, there exist some problems such as data packet dropout. Regardless of the type of the network, time delay naturally brings negative effect on NCS stability and performance, and even results in unstable control system [9, 3]. The delayed packets may be more harmful to system stability than packet loss [4]. There are three types of time delay in NCS:

1. Time delay from sensor to controller τ_{sc}
2. Time delay from controller to actuatorτ_{ca}
3. Processing time delayτ_p

Hence, the total time delay in NCS can categorized by the following:

$$\text{RTT} = \tau_{sc} + \tau_{ca} + \tau_p \quad \ldots (1)$$

Where RTT is the round trip time or the total time delay to send the sensor packet from sensor node, until the controller packet has received at the actuator node.

The problem under focus in this work is the loss of Quality of Performance (QoP) arising when conventional controllers (particularly: Proportional Integral Derivative (PID)) are implemented in NCS subject to time-varying delays between measurements and actuation. A delay-dependent Gain Scheduling PID (GSPID) controller structure is to be designed and compared with the conventional PID controller based on a specific performance analysis criterion.

In order to apply delay-dependent controller methodologies, the time delay measurement must be carried out for each sample. One option is to synchronize nodes [10, 11]. It's known that high accuracy in clock synchronization between network nodes will lead to high network bandwidth consumption, i.e. effecting Quality of Service (QoS) of network, and consequently resulting in more time delay because the network channel not idle most of the time. The synchronization protocol needs to send a huge quantity of special messages. Synchronization is a complicated process and requires additional packets to send the synchronized clock signals over the network, this will generate further traffic on network [12]. So in the control structure in this paper, another option is used to measure the RTT directly at the local side. By assuming sensor and actuator are in shareto the same local clock, hence no synchronization is required.

The layout of this paper is as follows: Section 2 describes the TrueTime toolbox. Section 3 presents the designed NCS model as well as PID and GSPID controllers. Section 4 presents the simulation results, and a comparison between PID controller and the designed GSPID controller is done. The conclusion and suggestions for future work are presented in section 5.

II. TrueTime Toolbox

Since NCS field is a combination of control system and network system, both aspects must be taken into consideration while designing NCS model or testing a controller algorithm. NCS lies in the co-simulation field of control and communication network design. The candidate co-simulation tool for this paper is the TrueTime toolbox [13] available at [14]. TrueTime is a MATLAB-based simulation toolbox that has been developed at department of Automatic Control, Lund University, Sweden since 1999, for designing and simulation of embedded control systems and distributed control systems. TrueTime toolbox library (version 2, 2010) showed in fig. 2 [15], consists of TrueTime Kernel, TrueTime Network (wired network), TrueTime Send, TrueTime Receive, TrueTime Battery, TrueTime Wireless Network, and TrueTime Ultrasound Network. These blocks are variable-step, discrete, MATLAB S-functions written in C++.

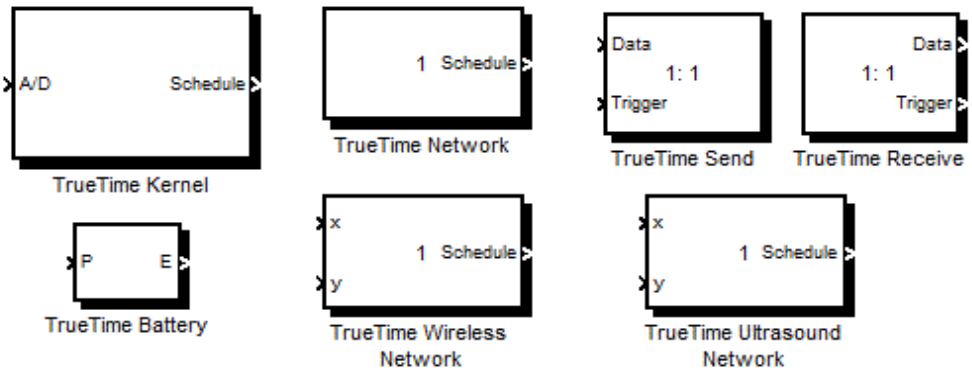

Figure 2: TrueTime toolbox library

TrueTime Kernel block simulates a computer node and is initialized by call code functions written in m-file or C++ code, this block executes a user defined tasks and interrupts handlers similar to a real time operating system kernel representing: input/output tasks, control algorithms, timers, and network interfaces. TrueTime kernel can communicate with other MATLAB blocks by A/D input port and D/A output port. TrueTime Kernel supports three scheduling policies: prioFP (Fixed-Priority scheduling), prioDM (Deadline-Monotonic scheduling), or prioEDF (Earliest-Deadline-First

scheduling). Each TrueTime Kernel is associated with independent local memory for storing the necessary data for each task.

TrueTime Network and TrueTime Wireless Network blocks distribute messages between computer nodes according to a chosen network model. TrueTime Network supports seven network protocols: Carrier Sense Multiple Access/Collision Detection (CSMA/CD), Carrier Sense Multiple Access/Arbitration Message Priority (CSMA/AMP), Round Robin, Frequency Division Multiple Access (FDMA), Time Division Multiple Access (TDMA), FlexRay, and Switched Ethernet. TrueTime Wireless Network works in the same way as the wired one, but takes into account the location of the nodes by specifying x and y inputs. TrueTime Wireless Network supports two network protocols: IEEE 802.11b/g (WLAN) and IEEE 802.15.4 (ZigBee). The main steps of build a NCS using TrueTime is as follow [16]:

1. Using TrueTime and Matlab/Simulink module to establish a real-time NCS model.
2. Design of nodes: adopt TrueTime Kernel nodes to build each node (such as sensor, controller, etc.).
3. Initialized all nodes with m-file code functions, and set appropriate parameters.
4. Design of network: set parameters of network module, includes: type of network, data rate, number of nodes, packet loss probability, etc.

III. The Designed NCS Model and Controllers

A. SummaryDescription of The Designed NCS Model

For designing embedded control system and NCS model, the TrueTime Kernel and TrueTime Network blocks are used, with the help of other MATLAB blocks. With the decreasing price, high data rate, widely use, increasing software and applications, and well-established infrastructure, Ethernet network has been widely used in industrial for controlling applications [17]. Therefore, the network setup considers the parameters shown in table 1. The designed NCS model represented with MATLAB and TrueTime toolbox is shown in fig. 3.

Table 1. Setup of network parameters, taken from [18, 19]

Item	Network parameter	Value
1	Type of network	Ethernet
2	Data rate	10 Mbps
4	Minimum packet size	72 byte
5	Maximum packet size	1518 byte

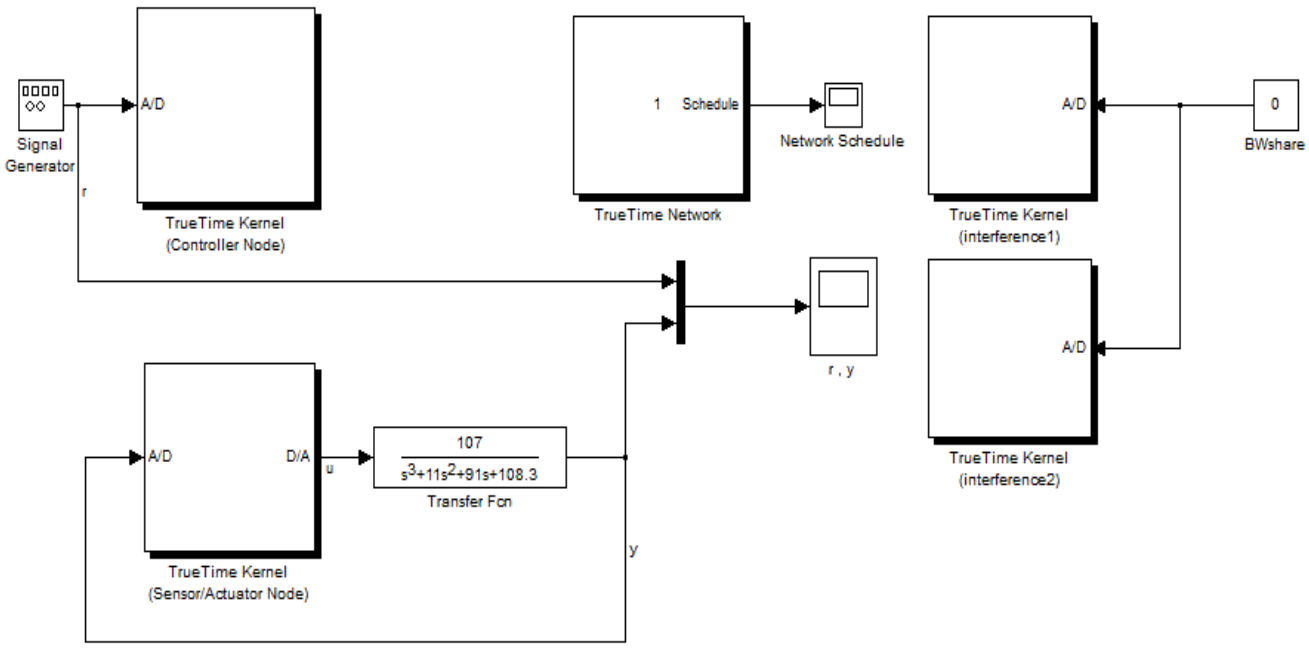

Figure 3: The designed NCS model represented with TrueTime and MATLAB

The DC motor studied in [20] is used as the controlled plant with the following transfer function:

$$G(s) = \frac{107}{s^3+11s^2+91s+108.3} \quad \ldots (2)$$

The open loop step response of this process is shown is fig. 4. Processrise time and settling time are equal to 1.54 sec and 2.92 sec respectively.

The local side consists of a TrueTime Kernel (Sensor/Actuator node) connected directly with the plant via A/D and D/A ports. The scheduling policy for this kernel is chosen as 'prioDM'. This kernel implement two tasks: sensor and actuator tasks, both share the same local memory storage. The sensor task is a time-base mode of operation, i.e. sensor will sense the plantevery 0.1 sec directly from A/D input port, and storing the time of sensing, named as *sensingtime*, along with a packet identifier (ID) for this *sensor_packet*. This *sensor_packet*is designed to contain: packet ID, plant measurement, and the pervious measured RTT. When the preparation of *sensor_packet*is done, the sensor will send it to the controller node over network with 72-byte packet size.The 0.1 sec sampling time is chosen for the reasons: (1) best for both QoP of control and QoS of network, (2) the process will be sensed about 15 times during its rise time (i.e. rise time/sampling time=15).

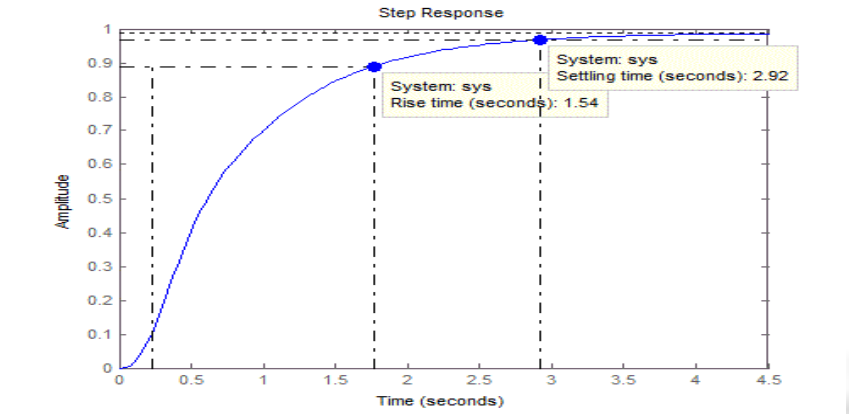

Figure 4: Open loop step response of process in (2)

The actuator task is responsible of doing D/A conversion of control signal to actuate the plant with the new received value. The actuator task is an event-based mode of operation, i.e. an interrupt is set to listen to network interface for any receiving *controller_packet*. When the actuator task is triggered, it will capturing the time of receiving *controller_packet*, named *responsetime*, and examine the *controller_packet* ID. Actuator task will search the localmemory for matching IDs. When is found, the associated *sensingtime* is retrieved and its field is cleared. Then the calculation of a new RTT value is done by:

$$\text{RTT} = responsetime - sensingtime\ (in\ sec) \quad \ldots (3)$$

The remote side consists of a TrueTime Kernel (Controller node) connected directly with a reference or set point input via A/D port. The reference input is chosen as a square wave form with frequency equals to 0.05 Hertez. The scheduling policy for this kernel is chosen as 'prioFP. This kernel implement the controller task which is either implement the PID controller algorithm or the proposed GSPID controller algorithm. The controller task is event-based mode of operation, i.e. an interrupt is set to listen to network interface for any receiving *sensor_packet*. When the controller task is triggered, it will do the following steps sequentially:

1. Reading or pulling the *sensor_packet* from network interface, and reading reference value from A/D input port.
2. Examine the packet ID and the RTT value.
3. Implement the desired controller algorithm, to produce control signal.
4. Preparing the *controller_packet*. This packet is designed to contain: packet ID which is the same received sensor_packet ID, and the control signal.
5. Sending the *controller_packet* to the actuator node over network with 72-byte packet length.

Because the controller node will receive an old value of RTT, the controller will use a forecasting technique for predication of the next RTT value, the Forecasted RTT (FRTT). The Moving Average (MA) is one type of the time series models [MA2], it takes the average of a defined window size as a forecast. The simplest form of MA, for example forecasting the RTT, can be represented by the following formula:

$$\text{RTT}_{t+1} = (\text{RTT}_t + \text{RTT}_{t-1} + \text{RTT}_{t-2} + \cdots + \text{RTT}_{t-ws-1})/ws \quad \ldots (4)$$

Where t is the time at which the forecasting is implemented and ws is the window size.RTT_{t+1} is the FRTT.
In order to generate load on the Ethernet network, Two TrueTime Kernel blocks (Interference1 and Interference2 nodes as showed in fig. 3) are used to send packets over network with different length (with algorithm similar to that used in

[19]) and hence a time-varying delay will induced in the communication channel. Four parameters define the operation mode for each interference node:

1. *maxP*: the Maximum packet size.
2. *BWshare*: a fraction of the occupied bandwidth by each node.
3. *w*: weighting index for the parameter *BWshare*, w is random number between 0 and 1.
4. nodes: number of computer nodes emulated by the interference node, each interference node represents 249 computer node, so the total nodes in this network will be 500 (including control system nodes).

In this way, each interference node will send a packet with size *maxP* × *BWshare* × *w* and repeated for *nodes* time. Obviously, when a high value of *BWshare* is used, high network traffic will be generated and vice versa.

B. PID Controller

PID controllers are commonly used, over 95% of the controllers in industrial applications are PID controller (or some form of PID controller like P, PI, or PD controller) [22], large factory may have thousands of them, in instruments and laboratory equipment. The PID controller is delay-independent, and therefore the MA technique is not taken into considering while designing a PID controller. Because PID controller is implemented as a computer-based, the continuous time PID controller has to be approximated by a discrete time PID controller described as [23]:

$$u(t_K) = P(t_K) + I(t_K) + D(t_K) \quad \dots (5)$$

Where:

$P(t_K) = Kp\, e(t_K)$

$(t_{K+1}) = I(t_K) + K_i T_s e(t_K)$

$D(t_K) = -K_d \dfrac{y(t_K) - y(t_{K-1})}{T_s}$

$e(t_K) = r(t_K) - y(t_K)$

Define $P(t_K)$ as the proportional term, $I(t_K)$ the integral term, and $D(t_K)$ the derivativeterm.t_K denotes the sampling instants,t_{K-1} is the previous state, $T_s$is the sampling period, $u(t_K)$is the control signal output, $e(t_K)$is the error, $r(t_K)$is the reference, and $y(t_K)$ is the received measurement.The PID tunable gains are: proportional gain K_p, integral gain K_i, and derivative gainK_d.

C. GSPID Controller

Gain scheduling controller is one of the most simple and popular nonlinear controllertechniques.The GSPID controller can be viewed as a multi-structure PID controller, which the PID controller switch itsparameters based on the scheduling variable. The scheduling variable for NCS is usually the time delay of the network, where the time delayis continuously changed based on the current data traffic in the network. To design a GSPID controller the following steps are considered:

1. Dividing the network time delays into groups at selected operating points, from 1 to n.
2. A separate PID controller is designed for each operating point, i.e. determine K_{pn}, K_{in},and K_{dn} for the n^{th}operating point.
3. The final control action is computed fromthese local PID controllers, as given below:

$$u(t_K) = \sum_1^n K_n\, P_n(t_K) + I_n(t_K) + D_n(t_K) \quad \dots (6)$$

Where K_n is the scheduling variable, which is defined by:

$$K_n = \begin{cases} 1 & \tau_n \le \tau_c < \tau_{n+1} \\ 0 & otherwise \end{cases}$$

τ_n is the n^{th} network time delay, and $\tau_c$is the current or the actual network time delay. Simply, when the current time delay is belong to a specific group, switch to the PID controller for that group. During online operation, the GSPID controller make an automatic switching between these stored local PID controllers. In this way, to implement the GSPID controller within a computer-based control system, a look-up table is used to stores and recalls these PID controllers.

IV. Simulation Results and Controllers Evaluation

By altering the *BWshare* value, different load scenarios can be obtained. Three scenarios for Ethernet network are studied and RTT is calculated online inside Sensor/Actuator node based on (3):

A. Unloaded network scenario: the control system nodes (Controller and Sensor/Actuator nodes) are the only active nodes in the NCS, in this case the *BWshare* value is set to zero and no packets will be send from interference1 and interference2 nodes. The RTT plot and its histogram for this scenario with 200 sec simulation run are showed in fig. 5 and fig. 6 respectively.

B. Medium loaded network scenario: the *BWshare* value is set to 0.4, the packet length send from interference1 and interference2 nodes will be (0.4 × 1518 × *w*) byte. The RTT plot and its histogram for this scenario with 200 sec simulation run, are showed in fig. 7 and fig. 8 respectively.

C. High loaded network scenario: the *BWshare* value is set to 0.9, the packet length send from interference1 and interference2 nodes will be (0.9 × 1518 × *w*) byte. The RTT plot and its histogram for this scenario with 200 sec simulation run, are showed in fig. 9 and fig. 10 respectively.

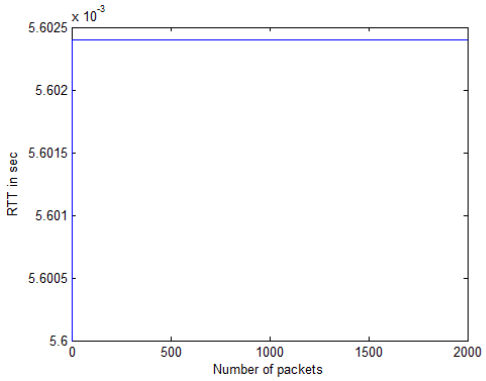

Figure 5: RTT values for unloaded Ethernet networkscenario

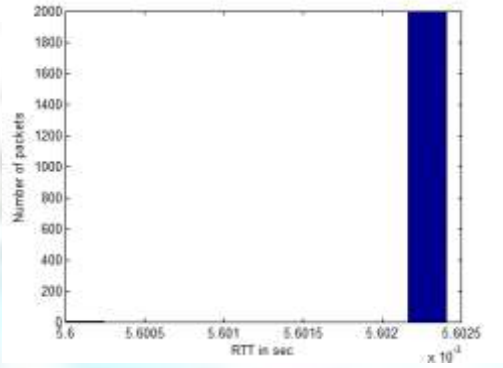

Figure 6: RTT histogram for unloaded Ethernet networkscenario

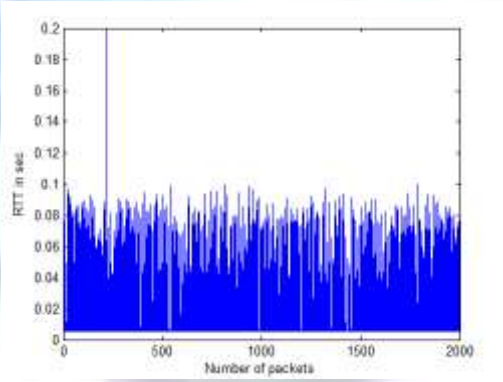

Figure 7: RTT values for medium loaded Ethernet network scenario

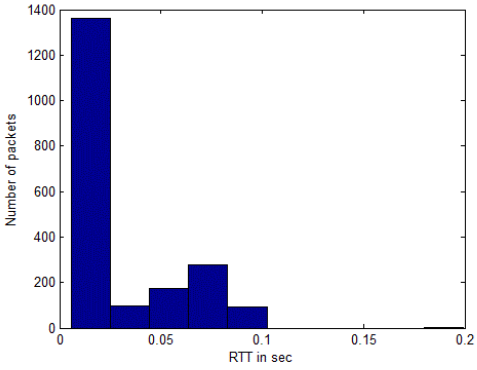

Figure 8: RTT histogram for medium loaded Ethernet network scenario

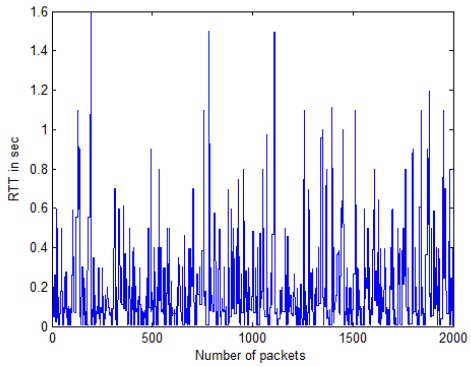

Figure 9: RTT values for high loaded Ethernet network scenario

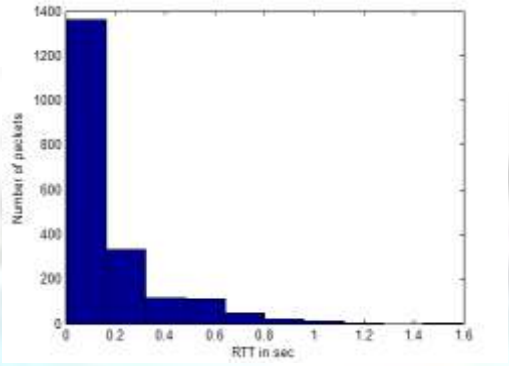

Figure 10: RTT histogram for high loaded Ethernet network scenario

The scenarios features can be summarized as follows: for unloaded Ethernet network, the time delay is almost constant at 5.6152 ms which is only the processing time at the controller node plus the transmission time to send a minimum length of an Ethernet packet (*sensor_packet* and *controller_packet*). In this case, Ethernet network is an ideal network for transferring control system packets with a very low time delay because of its high data rate. For medium loaded Ethernet network, except the *sensor_packet* and *controller_packet* there are other packets will be send over network, and one or more nodes will try to transmit at the same time which results in transmission collision and the nodes will be backed off and retransmit later. This will results in time-varying delays. The RTT for medium load could be bounded as less than or equal to 0.1 sec with a possibility of random high spikes.The last scenario shows that RTT values for high loaded Ethernet network are random and could not be bounded, 1.6 sec as a maximum value.

For comparison between PID and GSPID controllers, the PID controller of (5) is first tested with these scenarios. For the plant in (2) , PID gains are tuned automatically using *pidtune* MATLAB function without considering the network time delay in the closed loop control system. The obtained PID gains:$K_p = 2.114, K_i = 3.504,$ and $K_d = 0.083$ .The response of PID controller for 150 sec simulation run, along with control signal for unloaded (from 0 to 50 sec), medium loaded (from 50 to 100 sec), and high loaded (from 100 to 150 sec) network traffic scenarios, with respect to the square wave form input reference, is shown in fig. 11.

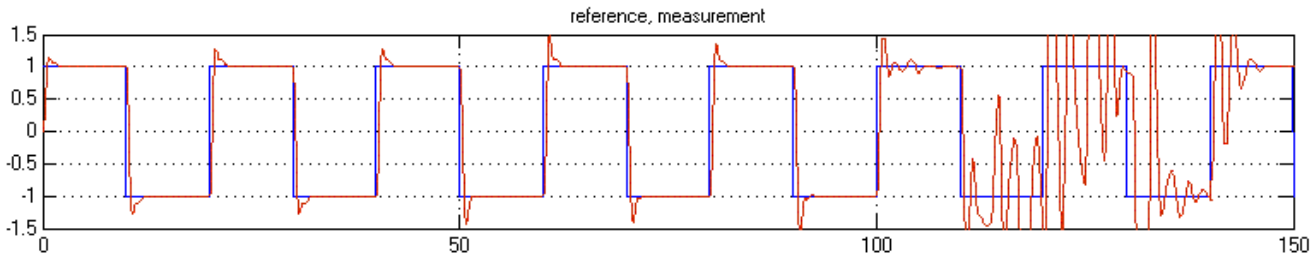

Figure 11: PID controller response

Because GSPID controller of (6) is delay-dependent, the MA technique is implemented inside controller node for calculating the FRTT. From the graphical analysis of the RTT values for different traffic scenarios for Ethernet network, it is found that spikes and high values of RTT does not last for more than five samples. The MA model of (4) with ws equal to five is used here for modeling RTT time series.

In order to implement the gain scheduling process, the FRTT values are divided into six groups:

1. Group one: (0.005 ≤ FRTT <0.1)
2. Group two: (0.1≤ FRTT <0.2)
3. Group three: (0.2≤ FRTT <0.3)
4. Group four: (0.3≤FRTT <0.4)
5. Group five: (0.4 ≤ FRTT <0.5)
6. Group six: (0.5 ≤ FRTT<0.6 and more)

The time-varying delays are assumed to be constant or have a small variations within each group. Table 2 shows the PID gains (automatically obtained using *pidtune* MATLAB function) for the median value of each group of the FRTT.

Table 2: PID gains for the six FRTT groups

FRTT value	PID gains		
	K_p	K_i	K_d
0.05	1.211	2.010	0
0.15	1.174	1.468	-0.081
0.25	0.843	1.185	0
0.35	0.861	0.904	-0.201
0.45	0.319	0.644	0
0.55	0.181	0.501	0

The response of GSPID controller for 150 sec simulation run, along with control signal for unloaded (from 0 to 50 sec), medium loaded (from 50 to 100 sec), and high loaded (from 100 to 150 sec) network traffic scenarios, with respect to the square wave form input reference, is shown in fig. 12.

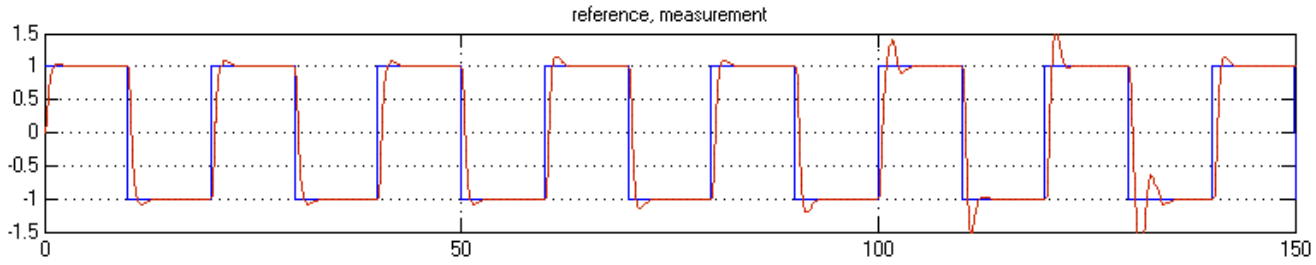

Figure 12: GSPID controller response

Generally the goal of any controller algorithm is simply achieving good reference tracking, thus the analysis of controller performance could depends on the observation of the plant measurement with respect to the input reference. Besides this criterion, many other performance analysis criteria are chosen to test the controllers' performance:

(1) ITAE (Integral Time Absolute Error): the accumulative sum of the absolute value of error multiplied by time over time, ITAE could be defined as: ITAE $= \int_0^t t|e(t)|\, dt$.

(2) MO (Maximum Overshoot): the maximum positive value of the system response.

(3) MU (Maximum Undershoot): the maximum negative value of the system response. The performance analysis criteria for PID and GSPID controllers are summarized in table 3.

As shown the PID controller has high values of IAE, MO, and MU. With PID controller the stability cannot be guaranteed. While GSPID controller significantly improves the closed loop system response with significantly decreases IAE, MO, and MU, and guarantee stability.

Table 3: Performance analysis criteria for PID and GSPID controllers

Controller type / Criteria	PID	GSPID
ITAE	4.639×10^3	1.572×10^3
MO	4.266	1.574
MU	-5.537	-1.811

V. Conclusion and Suggestions for Future Work

In this paper, a NCS model is built using TrueTime toolbox and ordinary MATLAB blocks. The network setup considers a remote side (Controller node) and a local side (Sensor/Actuator node connected with the plant). The Ethernet network is chosen as communication channel, the network is shared with other nodes that sending packets with randomlyvarying delays. The simulation shows the results for different Ethernet load scenarios. As a conclusion for the designed NCS, the use of intelligent components in control systems can significantly improves the performance. Since the modern controllers are implemented as a computer-based, the sensor, actuator, and controller can be implemented as tasks in real time operating systems, which can take advantage of the huge memory provided by the computer. Assuming that sensor and actuator of NCS are implemented within one node and both have a common local clock has the advantage to eliminate the need for synchronization between controller and sensor nodes. The calculation of RTT is done by the cooperation between sensor and actuator tasks.

The delay-independent PID controller is no more suitable for NCS field. A delay-dependent GSPID controller structure is proposed. Its performance is compared with the traditional delay-independent PID controller based on specific performance analysis criteria. The proposed GSPID controller has proved its effectiveness with varying time delays in NCS. The designed NCS model only eliminates the assumption of no packet loss in the communication channel. As a suggestion for future work, an appropriate solution could be designed when a packet is lost. Also the implementation of the GSPID controller for other networks, such as wireless networks, ad hoc networks, and the internet technologies, and taking into consideration the new changes. The authors are now conducting a work to use a fuzzy gain scheduling approach to improve the system performance while considering the effect of network delays and the packet loss under variable and width share and network load.

References

[1]. Hussein A. Abdullah and Chris R. Chatwin, “Distributed C3 Environment for Small to Medium-sized Enterprises”, Integrated Manufacturing Systems, Vol. 5 No. 3, MCB University Press Limited, Dhaka, Bangladesh, 1994.

[2]. Karl J. Astrom and Bjorn Wittenmark, “Computer-ControlledSystems 3rd edition”, Tsinghua University Press Prentice Hall, China, 1997.

[3]. Fei-Yue Wang and Derong Liu, “Networked Control Systems”, London, 2008.

[4]. Johan Nilsson, “Real-Time Control Systems with Delays”, Department of Automatic Control, Lund Institute of Technology, Lund, Sweden, 1998.

[5]. Lejiang Guo and Zhou Liu, “The Theory and Architecture of Network Control System”, International Conference on Intelligent Computing and Cognitive Informatics, Wuhan, China, 2010.

[6]. Joao P. Hespanha and et., “A SURVEY OF RECENT RESULTS IN NETWORKED CONTROL SYSTEMS”, Department of Electrical and Computer Engineering, Univ. of California, Santa Barbara, CA, USA,2007.

[7]. WEI ZHANG, “STABILITY ANALYSIS OF NETWORKED CONTROL SYSTEMS”, Department of Electrical Engineering and Computer Science, Case Western Reserve University, OH, USA, August 2001.

[8]. Bo Lincoln, “Dynamic Programming andTime-Varying Delay Systems”, Department of Automatic Control, Lund Institute of Technology, Lund, Sweden, 2003.

[9]. Yodyium Tipsuwan and Mo-Yuen Chow, “Control methodologies in networked control systems”, NC, USA, February 2003.

[10]. Svein Johannessen, “time synchronization in local area network”, IEEE Control Systems Magazine , Billingstad, Norway, April 2004.

[11]. Nikolai Vatanski and et., “Networked control with delay measurement and estimation”, Control Engineering Practice 17 (2009).

[12]. Li Zhang and et., “Clock Synchronization Algorithms for NetworkMeasurements”, IEEE INFOCOM, 160 - 169 vol.1, NY, USA, 2002.

[13]. Anton Cervin and et., “How Does Control Timing Affect Performance”, IEEE Control Systems Magazine, Lund, Sweden, June 2003.

[14]. TrueTime toolbox home page, http://www.control.lth.se/truetime.

[15]. Anton Cervin and et., “TRUETIME 2.0 beta—Reference Manual”, Department of Automatic Control, Lund University, Lund, Sweden, June 2010.

[16]. Y. Wanga and L. Heb, "ANALYSIS AND SIMULATION OF NETWORKED CONTROL SYSTEMS DELAY CHARACTERISTICS BASED ON TRUETIME", Computer Modelling and New Technologies, Vol.17, No. 4, Guizhou University, Guiyang, China, 2013.

[17]. Gadi Kaplan, "Ethernet's Winning Ways", IEEE SPECTRUM, January 2001.

[18]. Feng-Li Lian and et., "Performance Evaluation of Control Networks", IEEE Control Systems Magazine, University of Michigan, USA, February 2001.

[19]. Ángel Cuenca and et., "A Delay-Dependent Dual-Rate PID Controller Over an Ethernet Network", IEEE TRANSACTIONS ON INDUSTRIAL INFORMATICS, VOL. 7, NO. 1, February 2011.

[20]. L. Farkas and J. Hnat, "SIMULATION OF NETWORKED CONTROL SYSTEMSUSING TRUETIME", TechnicalComputing Prague 09, Prague, 2009.

[21]. Peter J. Brockwell and Richard A. Davis, "Introduction to Time Series and Forecasting" Second Edition, USA, 2002.

[22]. Donald Christiansen and et., "Standard Handbook of Electronic Engineering" Fifth Edition, USA, 2004.

[23]. Karl Johan Astrom and Richard M. Murray, "Feedback Systems", Princeton Univeristy Press, USA, September 2012.