A Multi-Threaded Communication Architecture for Networked Control Systems

Raazi Muhammad Khaliq-ur-Rahman Syed [1],* Muhammad B. Kadri [2]

[1]Department of Computer Systems Engineering, Faculty of Computing and Engineering, Mohammad Ali Jinnah University, Karachi, Pakistan

[2]Department of Mechatronics Engineering, College of Engineering, Karachi Institute of Economics and Technology, Karachi, Pakistan.

*Corresponding author: phone: e-mail: raazi.m.syed@ieee.org;

Abstract

Advancement in communication technology has paved the way for geographically dislocating controllers from the plants they are controlling. Establishing a secure and reliable communication is an essential component to achieve robust control performance. Myriad network control schemes have been proposed but they are incapacitated due to a lack of reliable software paradigm. This highlights the need of a distributed system, which provides platform for smooth communication between a plant and its controller. In this work, we propose CASAPAC, which is a multi-threaded communication architecture designed to ensure reliable and in-order delivery of information between different modules of a network control system. Any control algorithm can be tested and employed over any network using CASAPAC. An adaptive fuzzy controller and a network-based gain scheduled PI (Proportional Integral) controller have been tested on different networks using CASAPAC. In both cases, tests were carried out on a real plant of a coupled tank system. CASAPAC was able to handle all the communication efficiently in different scenarios and good control performance was achieved in both cases.

Keywords: Adaptive controls, Client-server systems, Communication architecture, Distributed system, Multithreading, Network delay, Networked control systems, Plant-controller communication

1. Introduction

Notion of using software in control systems has been around since we have had software systems. However, they had reduced applicability due to connection constraints and hardware limitations. This notion slowly transformed into reality as more sophisticated hardware evolved with the passage of time. However, software for automation systems is still bespoke as each system is unique and automated systems have varied characteristics and format [1-2]. Software has been developed separately for each type of system [3].

Although each control system has specific input and output parameters, attempts have been made to outline generalized software architectures [4-5]. Generalized software architecture can resolve commonly occurring issues and provide a development platform for future automation software [6-7] current attempts for generalized software architecture do not target control systems distributed across network/internet.

Control algorithms have three major components named as sensor module, actuator module and controller module. Actuator and sensor modules are responsible for pre and post processing of the data and communication with the plant. Control law is implemented in controller module. More recently, researchers have focused on distributing the control system software over the network. In this setting, sensor and actuator modules reside alongside the plant and communicate with the controller module over a network [8-9]. Consequently, many researchers have focused on issues and challenges of network control systems [10-14]. However, these researches lack actual implementation of a networked control system over a network/internet.

In this era of full-time connectivity and endless internet bandwidth, it is essential to have a distributed software architecture, which can measure performance of any control system distributed over network/internet. In that regard, network control systems require communication modules with separate sequences of execution. Also, data needs to be produced and consumed between these modules in a specific sequence. This outlines the need of a platform, using which control system engineers can run their control systems over network/internet. According to our knowledge, this is the first attempt to develop a system, which can be used to test the performance of any controller over the network/internet.

https://doi.org/10.24949/njes.v12i1.403

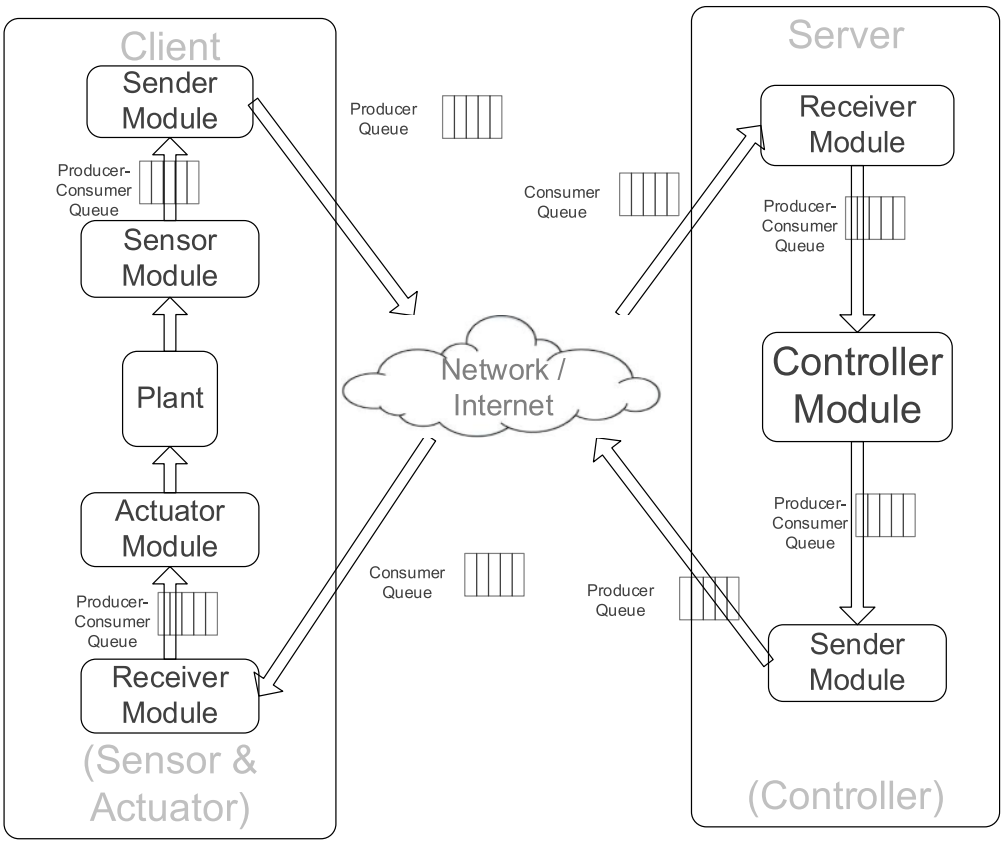

Figure 1. Network model

In this paper, we present CASAPAC, which is a multithreaded software architecture designed to support control systems over any type of computer network including the internet. CASAPAC uses the computing power of .net platform to run each module in a separate thread. Also, it ensures data consistency and seamless transfer of data between modules by implementing FIFO (First-In-First-Out) version of the classical producer-consumer problem [15]. Using CASAPAC, performance of any control system can be tested over the network/internet.

This paper is organized as follows: Related work is presented in section 2. In section 3, system and network model is outlined along with the assumptions taken into consideration while developing CASAPAC. In section 4, CASAPAC is presented comprehensively. We tested some existing control systems over the network using our proposed architecture. Results of those experiments, along with the discussion, are presented in section 5. Finally, conclusions are drawn in section 6.

2. Related Work

When discussing related work for control system software, it is important to mention efforts made to develop bespoke software, which incorporated control techniques specific to a system [3, 16]. Also, discussion about communication architecture for control systems will be incomplete without mentioning FASA [6]. FASA is a recent research, which proposes software architecture for flexible, distributed automation systems. Focus of FASA is on better usage of system resources to increase performance of any control system rather than providing generic platform for any control system distributed across computer networks.

A lot of research work has been focused on the effects (such as delays, information loss) of distributing control systems over networks [17-22]. Also, many techniques have been developed to mitigate network induced effects on control systems and to reduce load of control systems on computer networks [23-25]. However, all these researches have been based on specific communication models and simulations based on those models. Therefore, need for a generic platform is still not fulfilled.

Some researchers have focused on simulating existing controllers over networks [26]. In pursuit of a generic architecture for networked control systems, only some primitive efforts have been made [27]. Tools such as TrueTime and Jitterbug [28] provide platform to analyze performance of control systems with introduction of delays. However, they are just MATLAB/Simulink based simulation tools, in which delays are only simulated. On the other hand, CASAPAC is a generic software architecture, which works on actual delays and takes into account real computer network / internet.

3. Model and Assumptions

The network model, for which this architecture was developed, is shown in Fig. 1. The architecture consists of a client program and a server program, which communicate through network/internet. It is assumed that client and server programs are reachable to each other and no network errors exist. Also, this architecture cannot affect network induced delay. It is important to note that apart from communication over computer networks, there are others factors, which contribute towards delays [29-31]. Such delays should not be confused with network induced delays when testing performance of a control system over a computer network or the internet.

CASAPAC has been developed for testing any controller over computer networks. It is assumed that the target controller can understand the readings sensed from its plant and produce actuating values, which can be forwarded to its plant without further processing. CASAPAC does not process values transferred between controller and the plant.

4. Methodology

This architecture has been developed using C# language for any controller written in MATLAB. It ensures that the data from plant is fed into the controller in order and the data from controller is fed into the plant in order. Maintaining order of data is important in preventing malfunctioning of a control system. Also, it ensures correctness of results of any control algorithm tested across a network. The architecture consists of two programs, which reside across network or the internet. On server side, user has to select the MATLAB file, in which the subject control algorithm is written. On client side, user has to provide emergency shutdown command for the plant and information about the serial port and its parameters to connect with the plant. In addition to that, IP address and port numbers are required for communication between client and server programs.

On server side, MATLAB instance is initialized while serial port object is created at the client side. Apart from that, separate threads for sending and receiving are created at both client and server. There are three threads of execution on server side: Receiver, Controller and Sender. On client side, there are four threads: Sensor, Sender, Receiver and Actuator. Communication between threads of the same program is implemented keeping in mind producer-consumer problem with FIFO (First-In-First-Out) ordering. In producer-consumer problem with FIFO ordering, following issues are important: -

- A producer can't produce if producer-consumer buffer is full. We have kept buffer size sufficiently large, so that none of the values are missed out.
- A consumer can't consume if producer-consumer buffer is empty.
- A value, which is produced first, should be consumed first. This is important to ensure smooth controlling functionality.
- Producer and consumer should not access the producer-consumer buffer at the same time. This is used to ensure data consistency.

There are two producer-consumer buffers at server and client side each. These buffers hold data at different stages of processing. On client's side, 'sensedValue' buffer holds the values sensed from plant and 'valFromServer' holds processed values received from controller over a network. Sensor thread inserts sensed values into 'sensedValue' buffer while client's Sender thread retrieves them and sends them to the server. Client's Receiver thread inserts received values from server into 'valFromServer' thread while Actuator thread retrieves them and sends them to the plant. On server's side, 'valFromClient' buffer holds values received from plant and 'valAfterComputation' buffer holds processed values received from the controller implemented in MATLAB. Server's Receiver thread inserts values into 'valFromClient' buffer while server's Controller thread retrieves them and sends them as an argument to the controller function implemented in MATLAB. Controller thread inserts processed values into 'valAfterComputation' thread while server's Sender thread retrieves them and sends them to the client. Variable 'bufSize' indicates the maximum number of values that can be stored in the producer-consumer buffer. Buffer size can be configured to any value before running simulations. Inbuilt functionality of connection-oriented socket programming is used to ensure ordering between the following pairs of modules: client's Sender thread and server's Receiver thread and server's Sender thread and client's Receiver thread.

In order to implement in order delivery in every pair of producer-consumer modules, following variables, with suffixes, are used for every producer-consumer buffer: 'empty', 'full', 'latest', 'earliest' and 'mutex'. Variable 'empty' indicates the number of empty slots in a buffer, 'full' indicates the number of filled slots in a buffer, 'latest' indicates index of the newest value inserted into a buffer, 'earliest' indicates index of the oldest value inserted into a buffer and 'mutex' ensures that access to the buffer by producer and consumer are mutually exclusive. Suffix SV (or Sensor) indicates that the variable is used for 'sensedValue' buffer, suffix VFC indicates that the variable is used for 'valFromClient' buffer, suffix VAC indicates that the variable is used for 'valAfterComputation' buffer and suffix VFS indicates that the variable is used for 'valFromServer' buffer. All of these variables are declared and made public in main classes on both sides i.e. in class 'sensorActuatorForm' on client's side and in class 'ControllerDialog' on server's side.

In all cases, initial value of variable 'empty' is set equal to 'bufSize' and initial value of variable 'full' is set to zero, which means that all slots in the buffers are empty and no slot in the buffers is full in the beginning. Since the buffer is empty in the beginning, values of 'latest' and 'earliest' are set to zero. Starting at index zero, programs insert/retrieves values in/from producer-consumer buffers in circular manner i.e. indexes becomes zero whenever they become equal to 'bufSize' after increments. Lastly, all 'mutex' variables are set to one initially i.e. they are available. In order to access a buffer, its producer

or consumer module decrements its 'mutex' so that it becomes zero i.e. not available to the other module. After accessing the buffer 'mutex' is incremented so that it becomes one again i.e. available. If 'mutex' is not available to a module, it waits until it becomes available. While initializing variables at the server's side, an instance of MATLAB is made ready so that controller function can be invoked whenever required.

Client-server connections are established using connection-oriented socket programming and a pair of Sender and Receiver threads are started at both the server's side and the client's side. We refer to the main thread of execution at server's side as Controller thread and the main thread of execution at client's side as Actuator thread.

On client side, a serial port is initialized and opened. Also, a handler function is specified for receiving data from plant. As soon as the Sensor thread senses a value from plant, it waits for an empty slot i.e. a slot to become available in the 'sensedValue' buffer. As soon as an empty slot becomes available, it increments 'latest' variable, decrements 'empty' variable and waits on 'mutex' for the 'sensedValue' buffer. After acquiring 'mutex', it inserts value into the 'sensedValue' buffer, releases 'mutex' (increment it to one) and increments 'full' variable associated with the 'sensedValue' buffer.

In order to extract value from 'sensedValue' producer-consumer buffer and send it to the server using network/internet, client's Sender thread waits for a filled slot. Once a filled slot is available, it increments 'earliest' variable, decrements 'full' variable and waits on 'mutex' for the 'sensedValue' buffer. After acquiring 'mutex', it extracts value from the 'sensedValue' buffer, releases 'mutex' (increment it to one) and increments 'empty' variable associated with the 'sensedValue' buffer. The extracted value is sent to server using sockets and the client's Sender thread waits for a filled slot again.

As soon as the server's Receiver thread receives a value, it inserts it into 'valFromClient' buffer using the same above described procedure, which is used by client's Sensor thread to insert values into 'sensedValue' buffer. Server's Controller thread extracts values from 'valFromClient' buffer using the same above described procedure, which is used by client's Sender thread to extract values from 'sensedValue' buffer. After extracting a value, server's Controller thread invokes the target MATLAB function, passes the extracted value as a parameter to the function, gathers result and inserts the result into 'valAfterComputation' buffer using the same above described procedure, which is used by client's Sensor thread to insert values into 'sensedValue' buffer.

Results from the MATLAB function are the actuating values that are to be sent to the plant. Server's Sender thread extracts actuating values from 'valAfterComputation' buffer using the same above described procedure, which is used by client's Sender thread to extract values from 'sensedValue' buffer. After extracting an actuating value from 'valAfterComputation' buffer, the server's Sender thread sends it to the client using sockets and waits for the next actuating value from the Controller thread.

Client's receiver thread always waits for actauting values from the server. As soon as it receives a value, it inserts it into 'valFromServer' buffer using the same above described procedure, which is used by client's Sensor thread to insert values into 'sensedValue' buffer. Client's Actuator thread extracts actuating values from 'valFromServer' buffer using the same procedure, which is used by client's Sender thread to extract values from 'sensedValue' buffer. After extraction, it sends the value to the plant through the serial port and waits for the next actuating value to be forwarded to the plant.

In addition to the above functionality, CASAPAC ensures graceful shutdown of both programs in case of interruption. In case of interruption, all threads are joined and all ports and sockets are closed. At client's side, emergency shutdown command is sent to the plant before closing the port.

5. Experiments and results

According to our knowledge, this is the first attempt in developing generic software for distributing any control systems across computer networks / internet with actual delays. Comparison of our architecture with bespoke software or software, which has simulated delays, cannot provide meaningful results. Therefore, we are providing results achieved from testing state-of-the-art controllers over computer network / internet using CASAPAC.

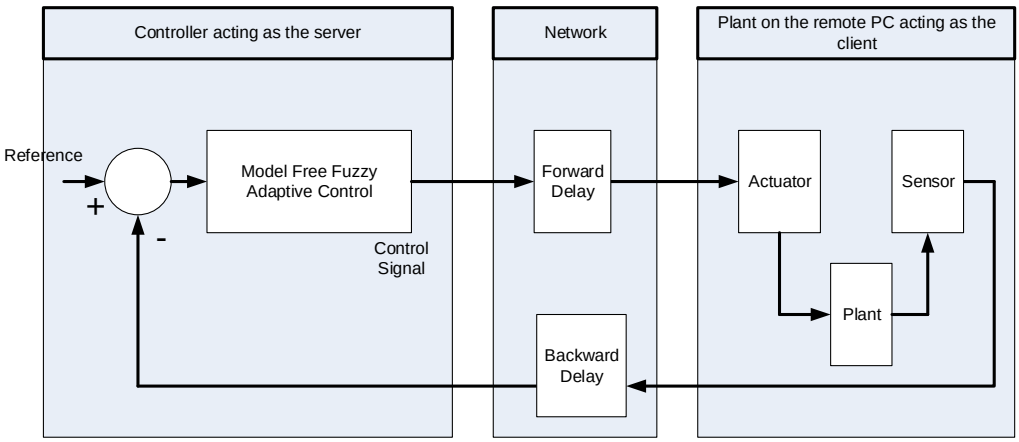

Figure 2. Block diagram of the proposed control setup

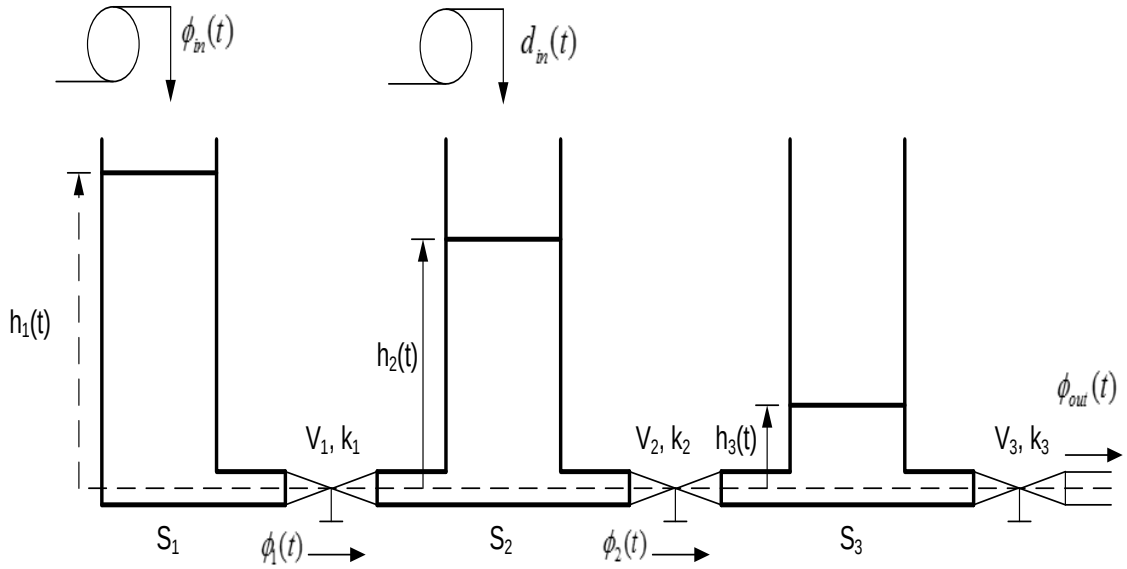

Figure 3. Block diagram of the coupled tank system

Two different controllers were successfully tested on CASAPAC with 'bufSize' configured to '50'. The general block diagram of the control setup is shown in Fig. 2. An adaptive fuzzy controller (AFC) was tested in three different scenarios i.e. over a LAN (Local Area Network), over the internet and over a Wi-Fi network. A network based PID controller [32] was also tested using the proposed network control architecture. In all the cases the plant is a coupled tank system (CTS), which is considered to be standard test equipment for evaluating the performance of control algorithms. The control objective was to maintain the liquid in the middle tank at a desired level. CTS was modeled as a single input single output (SISO) system. The inlet was in the first tank S1 whereas the liquid drained out from the right most tank labelled S3. There was no disturbance in the system hence din(t) i.e. water flow rate in the middle tank was zero. The block diagram of the CTS is given in Fig. 3.

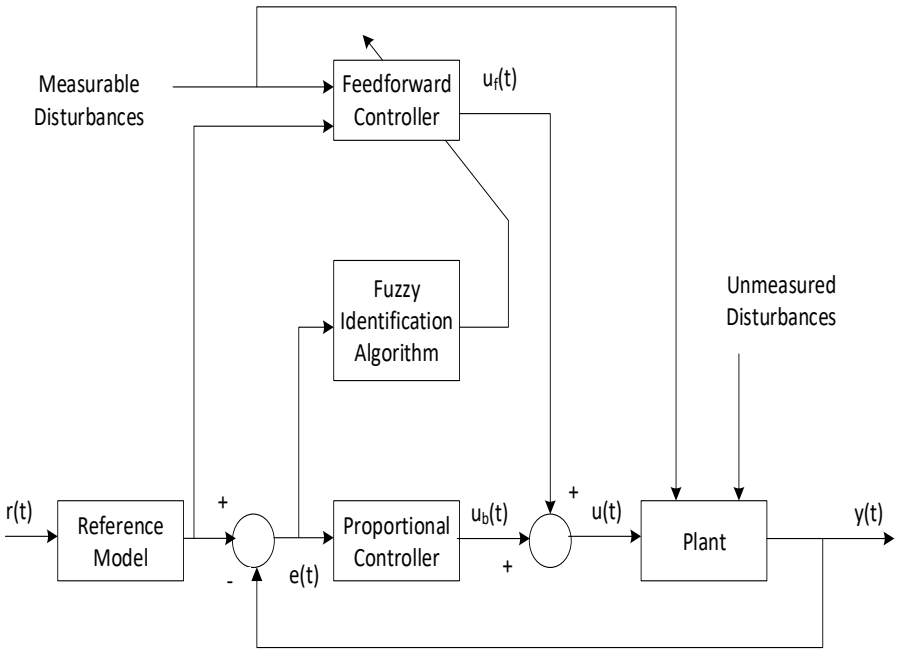

Figure 4. Block diagram of the adaptive fuzzy controller

5.1 Network based Adaptive Fuzzy Controller

The block diagram of the AFC is shown in Fig. 4. A complete overview of the AFC can be found in the works of Kadri et al. [33-34].

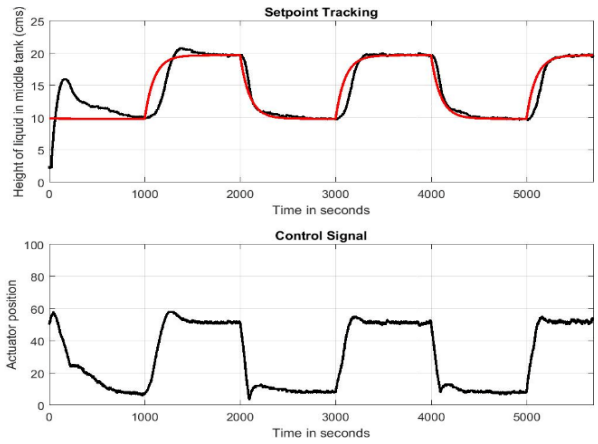

Figure 5. Control performance over local area network

Using CASAPAC, control systems can be implemented on any type of network. Three different scenarios in which the AFC was tested are discussed below: -

- The plant and the controller are on a local area network with Ethernet connectivity (Fig. 5).
- The plant and the controller are connected via internet using fixed IP (Fig. 6).
- The plant and the controller are connected using Wi-Fi (Fig. 7).

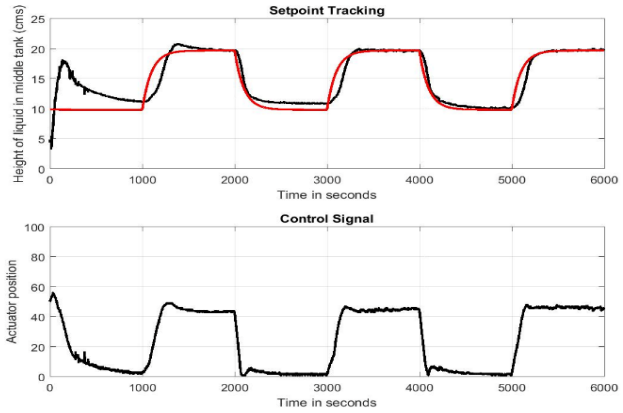

Figure 6. Control performance over the internet

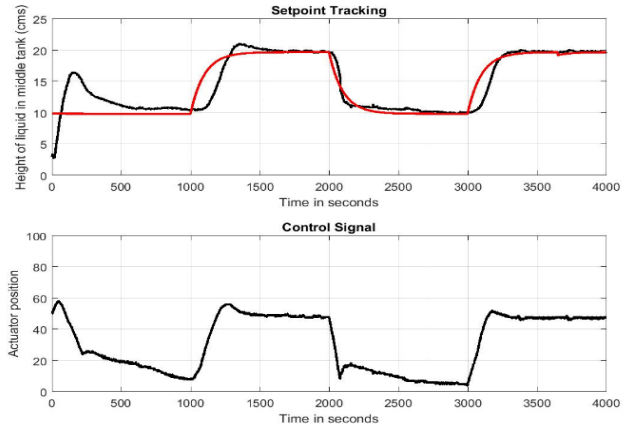

Figure 7. Control performance over wireless network

In all the three cases all the communication between the plant and the controller was handled seamlessly by CASAPAC. The control performance in all the three cases was satisfactory. The controller and the plant were unaware of the intermediate network layer. Network induced delays were compensated by the control scheme.

5.2 Network based PI Controller

As another example a network-based PI controller was also tested with CASAPAC. In order to test the network-based PI controller no software support was proposed by the researchers. Network based PI was implemented using CASAPAC. The plant is modelled as first order plus dead time (FOPDT). The plant and controller equations are given below: -

$$G_c(s) = k_c\left[1 + \frac{1}{T_i s}\right] - - - - - - - - - - - (1)$$

$$G_p(s) = \frac{k_p}{1+T_s} e^{-sL} - - - - (2)$$

The controller parameters shown in the Table are derived based on the gain margin (GM), phase margin (PM), network induced delay, gain of the system and time constant specifications: -

$$\frac{\pi}{2} + arctan\omega_p T_i - arctan\omega_p T - \omega_p(\tau_n + L) = 0 - (3)$$

$$\frac{1}{A_m} = \frac{k_c k_p}{\omega_p T_i}\sqrt{\frac{1+\omega_p^2 T_i^2}{1+\omega_p^2 T^2}} - - - - - - - - - - - - (4)$$

$$1 = \frac{k_c k_p}{\omega_g T_i}\sqrt{\frac{1+\omega_g^2 T_i^2}{1+\omega_g^2 T^2}} - - - - - - - - - - - - (5)$$

$$\varphi_m = \frac{\pi}{2} + arctan\omega_g T_i - arctan\omega_g T - \omega_g(\tau_n + L) - - - - - - - - - - - - - - - - - - (6)$$

Table 1: Controller parameters for the network-based PI controller

	PI_1	PI_2
Pre-specified Parameters		
A_m	3db	3db
φ_m	50°	50°
k_p	19.981	28.065
L	5.047s	6.2s
T	73.995s	98.97s
τ_n	3s	3s
Computed Values of PI Controller		
ω_p	0.1865	0.1636
ω_g	0.065	0.0568
T_i	38.7179	48.047
k_c	0.2287	0.1912

Where ω_p is the phase crossover frequency, ω_g is the gain crossover frequency, A_m is the specified gain margin, φ_m is the specified phase margin, τ_n is the total network delay, L is the plant dead time, T is the plant time constant, k_p is the plant gain, T_c is the integral time constant and k_c is the controller gain. Since CTS is nonlinear, it was linearized at two different operating points. The two linearized FOPDT models at the different operating points are given in following equations: -

$$Gp(s) = \frac{18.981}{73.995s+1} e^{-5.07s} - - - - - - - - - (7)$$

$$Gp(s) = \frac{28.065e^{-6.2s}}{98.97s+1} - - - - - - - - - - - - (8)$$

A gain scheduled network-based PI controller was designed with two different sets of gains for the two operating levels. A comparison of the control performance for the gain scheduled network-based PI and AFC on CASAPAC is shown in Fig. 8.

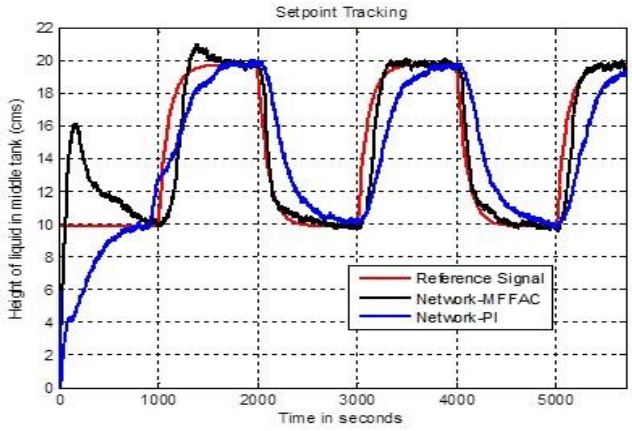

Figure 8. Comparison of control performances

6. Conclusions and future work

A software architecture (CASAPAC) has been proposed for networked control system. The controller and the plant

communicate over a network and the machines can be geographically distributed. The proposed architecture provides an interface for the plant and the controller so that they can be configured for any type of network connectivity e.g. LAN, internet or Wi-Fi. The network architecture ensures that the data can be transferred between plant and controller over network smoothly and in order. On the client side, separate threads are used for processing sensor information and subsequently sending actuation signal to the plant. On the server side, sending and receiving threads, communicates with the main thread. Main thread invokes the MATLAB controller function, which processes all the information and generates a control signal to be relayed to the client. The user is completely oblivious of the threading model.

References

1. M.B. Kadri, "Disturbance rejection in information poor systems using model free neurofuzzy control", Ph.D. Thesis, University of Oxford, Oxford, United Kingdom, 2009.
2. K.M. Vijaya, S. Sundaram, S.N. Omkar, G. Ranjan and S. Prasad, "A direct adaptive neural command controller design for an unstable helicopter", *Eng. Appl. Artif. Intel*, vol. 22, 2009, pp.181-191.
3. T.M. McPhillips, S.E. McPhillips, H.J. Chiu, A.E. Cohen, A.M. Deacon, P.J. Ellis, E. Garman, A. Gonzalez, N.K. Sauter, R.P. Phizackerley et al., "BluIce and the Distributed Control System: software for data acquisition and instrument control at macromolecular crystallography beamlines", *J. Synchrotron Radiat.*, vol. 9, 2002, pp. 401-405.
4. D.L. Rogerio, G. Holger, A.M. Hausi, S. Mary, A. Jesper, L. Marin, S. Bradley, T. Gabriel, M.V. Norha, V. Thomas et al., "Software engineering for self-adaptive systems: A second research roadmap", *Proceedings of Software Engineering for Self-Adaptive Systems II*, Berlin, Heidelberg, Germany: Springer, pp. 1-32, 2013.
5. V. Valeriy, "Software engineering in industrial automation: State of the art review", *IEEE T. Ind. Inform.*, vol. 9, 2013, pp. 1234-1249.
6. W. Michael, G. Thomas, K. Atul and O. Manuel, "FASA: A software architecture and runtime framework for flexible distributed automation systems", *J. Syst. Architect.*, vol. 61, 2015, pp. 82-111.
7. E. Raphael, S. Thanikesavan, M. Aurelien and L. Jun, "Real-time network traffic handling in FASA", *Proceedings of 10th IEEE International Symposium on Industrial Embedded Systems (SIES)*, NY, USA, pp. 1-10, June 8-10, 2015.
8. G.R. Ashok and Y.C. Mo, "Networked control system: overview and research trends", *IEEE T. Ind. Electron.*, vol. 57, 2010, pp. 2527-2535.
9. A. Panos and B. John, "Special issue on technology of networked control systems", P. *IEEE*, vol. 95, 2007, pp. 5-8.
10. Z. Lixian, G. Huijun and K. Okyay, "Network induced constraints in networked control systems: A survey", *IEEE T. Ind. Inform.*, vol. 9, 2013, pp. 403-416.
11. Z. Hui, S. Yang and M.A. Saadat, "Robust H∞ PID control for multivariable networked control systems with disturbance noise attenuation", *Int. J. Robust Nonlin.*, vol. 22, 2012, pp. 183-204.
12. W. Xiaofeng and M.D. Lemmon, "Event triggering in distributed networked control systems", *IEEE T. Automat. Contr.*, vol. 56, 2011, pp. 586-601.
13. M.C.F. Donkers, W.P.M.H Heemels, V.D.W. Nathan and H. Laurentiu, "Stability analysis of networked control systems using a switched linear systems approach", *IEEE T. Automat. Contr.*, vol. 56, 2011, pp. 2101-2115.
14. Y. Rongni, S. Peng, P.L. Guo and G. Huijun, "Network based feedback control for systems with mixed delays based on quantization and dropout compensation", *Automatica*, vol. 47, 2011, pp. 2805-2809.
15. A. Silberschatz, P.B. Galvin and G. Greg, Operating system concepts". Ninth ed., NY, USA: Wiley, 2013.
16. J.L. Yan and T. Shaocheng, "Adaptive fuzzy control for a class of unknown nonlinear dynamical systems", *Fuzzy Set. Syst.*, vol, 263, 2015, pp. 49-70.
17. C. Seunghwan, N.S. Kiong and W. Wenqin, "Robust H∞ fuzzy control of discrete nonlinear networked control systems A SOS approach", *J. Frankl. Inst.*, vol. 351, 2014, pp. 4065-4083.
18. Y. Huaicheng, Y. Sheng, Z. Hao and S. Hongbo, "L2 control design of event triggered networked control systems with quantizations", *J. Frankl. Inst.*, vol. 352, 2015, pp.332-345.
19. G. Yang, W. Jingcheng, Z. Langwen and L. Chuang, "Robust H∞ control of multi systems with random communication network accessing", *J. Frankl. Inst.*, vol. 352, 2015, pp. 1693-1721.
20. L. Lu, P. Feng and X. Dingyu, "Fractional order optimal fuzzy control for network delay", *Optik*, vol. 125, 2014, pp. 7020-7024.
21. H. Fei, F. Gang, W. Yong, Q. Jianbin and Z. Changzhu, "A novel dropout compensation scheme for control of networked TS fuzzy dynamic systems", *Fuzzy Set. Syst.*, vol. 235, 2014, pp. 44-61.
22. Z. Changzhu, F. Gang, Q. Jianbin, A.Z. Wen, "TS fuzzy model based piecewise H∞ output feedback controller design for networked nonlinear systems with medium access constraint", *Fuzzy Set. Syst.*, vol. 248, 2014, pp.86-105.
23. Z. Dawei, L.H. Qing and J. Xinchun, "Network-based output tracking control for TS fuzzy systems using an event-triggered communication scheme", *Fuzzy Set. Syst.*, vol. 273, 2015, pp. 26-48.
24. W. Huijiao, S. Peng and Z. Jianhua, "Event-triggered fuzzy filtering for a class of nonlinear networked control systems", *Signal Process.*, vol. 113, 2015, pp. 159-168.
25. H. Songlin, Y. Dong, P. Chen, X. Xiangpeng and Y. Xiuxia, "Event triggered controller design of

nonlinear discrete time networked control systems in TS fuzzy model", *Appl. Soft. Comput.*, vol. 30, 2015, pp. 400-411.
26. D.T. Hoang, H.G. Zhi, K.D. Xuan and M.C. Xin, "FuSY. A normalized PID controller in networked control systems with varying time delays", *ISA T.*, vol. 52, 2013, pp. 592-599.
27. K. Kirsanov, "The Architecture of Robotics Control Software for Heterogeneous Mobile Robots Network", *Proceedings of 24th International Symposium on Intelligent Manufacturing and Automation (DAAAM 2014)*, London, UK, pp. 216-221, October 23-26, 2013.
28. C. Anton, H. Dan, L. Bo, E. Johan and E.A. Karl, "How does control timing affect performance", *IEEE Contr. Syst. Mag.*, vol. 23, 2003, pp. 16-30.
29. Z. Lai, P. Wu and D. Wu, "Application of fuzzy adaptive control to a MIMO nonlinear time delay pump valve system", *ISA T.*, vol. 57, 2015, pp. 254-261.
30. J. Li and H. Yue, "Adaptive fuzzy tracking control for stochastic nonlinear systems with unknown time varying delays", *Appl. Math. and Comput.*, vol. 256, 2015, pp. 514-528.
31. A. Arunkumar, R. Sakthivel and K. Mathiyalagan, "Robust reliable H∞ control for stochastic neural networks with randomly occurring delays", *Neurocomputing*, vol. 149, 2015, pp. 1524-1534.
32. H. Tran, Z. Guan, X. Dang, X. Cheng and F. Yuan, "A normalized PID controller in networked control systems with varying time delays", *ISA T.*, vol. 52, 2013, pp. 592-599.
33. M.B. Kadri and A.L. Dexter, "Disturbance rejection using fuzzy model free adaptive control FMFAC with adaptive conditional defuzzification threshold", *Int. J. Uncertain. Fuzz.*, vol. 22, 2014, pp. 243-261.
34. M.B. Kadri, "Fuzzy relational control of uncertain systems", J. Frankl. Inst., vol. 351, 2014, pp. 3013-3031.