CHAPTER 2

NETWORK CONTROL SYSTEM – A CONTROL PERSPECTIVE

2.1 INTRODUCTION

Traditional control system relies on analog data transmission for both sensing and actuation. Analog cables carry one signal per cable and are susceptible to contamination from a variety of noise sources. Even when the modulation schemes are used, significant noise susceptibility remains. These attributes of analog systems demand a power incentive for the use of networking in control. This incentive provides operational and capital cost aspects. Digital signals can have arbitrarily strong protection against noise and cost of that protection scales reasonably with the severity of the noise, whereas beyond certain practical limits, the cost of protecting analog signals escalates rapidly. Control systems, however, operate on short time scales and thus with proper design, digital signal with a very high degree of data integrity can be ensured. The most important property of a network in a control system is the delay or latency time.

The important attribute to be considered with network is throughput (i.e.,) the amount of information that can be passed through the network per unit of time. With the many networked components, delays cause errors in synchronizing different parts of the system thereby affecting the throughput. It is true that network with higher throughput will often have shorter latency times than networks with lower throughput, though the latency time depends
on many other factors like communication protocol. The overload associated with networking becomes critical when multiple components require control commands at high speed to maintain system stability.

This chapter discusses a fuzzy based controller to improve the system performance when the system is under the influence of network delay. Control methods that include a unique state estimator are suggested as an improved solution that reduces the communication overhead in a network. Also for the convenience, system analysis is done using the Single Input Single Output (SISO) model of DC motor which represents a single control loop. The fuzzy tuning rules for the PI controller are applied to this single control loop to reduce the effect of delay. Multi Input Multi Output (MIMO) system is considered to simulate distributed network model of many control loops. For the MIMO the state estimator design is extended to study the effect of overhead caused by control loops of the network in the system performance.

2.2 CLOSED LOOP CONTROL OF DC MOTOR

The DC Motor is a common actuator found in many industrial control applications like robotics. Since many applications use multiple DC motor for multivariable control, the analysis on DC motor performance in a multi looped networked system is essential. The effect of long time network delay on the DC motor performance and adverse effect on stability are stated as follows: When the DC motor turns on, the controller will rapidly increase the input voltage for the output speed to track the required reference speed to be achieved in an acceptable rise time with settling time criteria.

But as the output speed approaches the required reference speed, the controller will reduce the input voltage since the error signal between the
reference speed and DC motor speed is reduced. When network is used in the closed loop DC motor control, the input voltage command from the controller to the DC motor will be delayed due to the NID. If this delay is long enough, the DC motor will still run on the non reduced input voltage which causes high overshoot response that may lead to burning the DC motor.

Figure 2.1 shows the equivalent diagram of DC motor, from which state space model can be developed to analyse its performance under varied conditions.

From this the armature voltage equation can be expressed as,

\[ u(t) = e_a = R i_a + L \frac{di_a}{dt} + K_i \omega \]  

(2.1)

The mechanical torque balance based on Newton’s law is

\[ J \frac{d\omega}{dt} + B \omega + T_l = T_e = K_i \omega \]  

(2.2)

where \( u(t) = e_a \) is the armature input voltage, \( L \) is the armature winding inductance; \( i_a \) is the armature current; \( R \) is the armature winding resistance; \( J \) is the system moment of inertia; \( B \) is the system damping coefficient; \( K \) and
$K_b$ are the torque constant and the back emf constant, respectively; $T_l$ is the load torque in Nm; and $\omega$ is the rotor angular speed.

Assuming $T_l = 0$, the state variable $X$ can be expressed as $[i_a \ \omega]^T$, where $i_a$ and $\omega$ are the armature current and shaft rotational speed respectively. Then state space expression for the DC motor model is expressed as follows:

$$
\dot{x} = \begin{bmatrix} \dot{i}_a \\ \dot{\omega} \end{bmatrix} = \begin{bmatrix} \frac{-R}{L} & -\frac{K_b}{L} \\ -\frac{K}{J} & -\frac{B}{J} \end{bmatrix} \begin{bmatrix} i_a \\ \omega \end{bmatrix} + \begin{bmatrix} \frac{1}{L} \\ 0 \end{bmatrix} u 
$$

(2.3)

$$
Y = \begin{bmatrix} 0 & 1 \end{bmatrix} x
$$

(2.4)

The parameters of the DC motor used are given in the Table 2.1.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ra</td>
<td>Armature Resistance</td>
<td>6.43 $\Omega$</td>
</tr>
<tr>
<td>La</td>
<td>Armature Inductance</td>
<td>2.88e^{-3} mH</td>
</tr>
<tr>
<td>J</td>
<td>Moment of inertia</td>
<td>3.53125e^{-6} Nm-s</td>
</tr>
<tr>
<td>B</td>
<td>Damping Co-efficient</td>
<td>0.1e^{-3} Nm</td>
</tr>
<tr>
<td>K</td>
<td>Torque constant</td>
<td>2.55e^{-3} Nm/A</td>
</tr>
<tr>
<td>K_b</td>
<td>Back EMF constant</td>
<td>0.255e^{-4} V/(rad/s)</td>
</tr>
</tbody>
</table>
From the equations (2.3) and (2.4) and also from the DC motor parameters as shown in Table 2.1, the open loop transfer function of the DC motor is expressed as

$$G(S) = C (sI - A)^{-1} B$$

$$= 2.507 \times 10^5 / (s^2 + 2261s + 6.329 \times 10^4)$$

The uncompensated closed loop control of DC motor is shown in Figure 2.2. The step input with a reference speed of 50 rad/sec is applied, the output response is as shown in Figure 2.3. From this output it is observed that the speed is not settled at reference speed of 50 rad/s and the performance factors of this uncompensated closed loop control system is listed below:

- Rise Time $t_r = 14.8$ ms,
- Delay Time $t_d = 5.2$ ms,
- Settling time $t_s = 26.8$ ms,
- Maximum overshoot $M_p = 0$,
- Steady state error $e_{ss} = 10.1024$ rads/s.

![Diagram of uncompensated closed loop DC motor](image)

**Figure 2.2 Uncompensated Closed Loop DC motor**

![Graph showing step response of the uncompensated DC motor](image)

**Figure 2.3 Step response of the uncompensated DC motor**
For the uncompensated DC motor, it is calculated that the steady error \( e_{ss} \) is 10.1024 rad/s. For the DC motor, the controller is designed such that for the applied step input, it should provide fast response with a rapid rise time and settling time. The maximum overshoot should be less than 5% and the steady state error of the motor speed should be less than 1%.

The PI controller most commonly used in industrial applications is added in the forward path of the closed loop control system which reduces the steady state error without affecting the transient response of this closed loop control system as shown in Figure 2.4.

![Figure 2.4  PI controlled DC Motor](image)

From the basics of control system, the values for the controller parameters are calculated as \( K_p = 1.5 \) and \( K_i = 42.5 \) and the output response of the PI controlled DC motor is shown in Figure 2.5 and the output settles at the reference speed of 50 rad/s.

![Figure 2.5  Output response of PI controlled DC motor](image)
The performance factors of PI controlled DC motor is calculated as follows:

\[
t_r = 12.34 \text{ ms}; \ t_d = 4.29 \text{ms}; \ t_s = 21.52 \text{ms}; \ M_p = 0\%; \ e_{ss} = 0 \text{ rad/s}.
\]

The PI controller in the feed forward path of the closed loop control is thus reducing the steady state error to zero and the settling time of the system to 21.52 ms.

However, if the control system is closed through communication network, the performance factors obtained for the above designed PI controlled system will be applicable, only if lossless communication network is assumed. But the communication network imposes a random delay to the system, thereby affecting the performance factors. Figure 2.6 shows the network based PI controlled DC motor.

![Network based PI controlled DC Motor](image)

**Figure 2.6 Network based PI controlled DC Motor**

The control voltage signal \( u_c(t) \) is converted into an actuating signal to drive the DC motor and is mathematically expressed as

\[
u_R(t) = u_c(t-\tau_R)
\]

(2.18)

where \( \tau_R \) is the network delay time to transmit the control signal \( u_c \).

The measured signal \( y_R(t) \) is given back to the controller as

\[
y_c(t) = y_R(t- \tau_c)
\]

(2.19)

where \( \tau_C \) is the network delay time to transmit the measured signal \( y_R \).
The output response of the network based DC motor control, hereafter referred as NCS, is as shown in Figure 2.7. For the delay of 4ms, it is observed from Figure 2.7, the overshoot response and settling time has increased. Figure 2.8 present the output response of the NCS subjected to various delay. Due to the presence of Network Induced Delay (NID) the behaviour of network is unpredictable which may degrade the performance of the NCS.

![Figure 2.7 Output Response of the network based DC Motor](image)

**Figure 2.7** Output Response of the network based DC Motor

![Figure 2.8 Output responses with different random time delay](image)

**Figure 2.8** Output responses with different random time delay
2.3 NEED FOR DECISION BASED SYSTEM

In industries where large distributed process takes place the controlling unit takes a major role. In some chemical processes which are MIMO networked systems, the system components are to be controlled accurately in order to achieve the optimal performance. There are some cases where multiple sensors and actuators are operating time synchronously for the whole processes. So in order to operate the specific sensor or actuator in a networked system, a decision making system is needed. This could schedule the network utility by the system components so as to improve the system performance and also here the reduction in power consumption of communication network.

2.4 FUZZY BASED DELAY COMPENSATOR FOR NCS

Fuzzy logic is a convenient way to map an input space to an output space. Fuzzy system is the best way to implement human knowledge. Fuzzy variables are linguistic variables and could be described by membership functions. Fuzzy propositions have variables, adverbs and connectors. Fuzzy Inferences are same as the human does his conclusions with no determination.

Fuzzy control for NCS is an attractive method because the fuzzy control technique can be used for control of networked plant which is subjected to system uncertainties as delay and dropout in communication network.

This section suggests a self tuning fuzzy PI controller that improves the system performance subjected to NID. This control algorithm tunes the parameters of PI controller by incorporating fuzzy inference and realizing a
fuzzy adaptive PI controller that can be used to improve the performance of network based systems.

The fuzzy based controller is discussed here as a compensator for the network induced time delay effects.

2.4.1 Fuzzy Logic Controller

Figure 2.9 shows the basic configuration of a Fuzzy Logic Controller (FLC), which comprises four principal components: a rule base, a fuzzy inference system, an input fuzzification interface and an output defuzzification interface. The rule base holds a set of IF-THEN rules that quantify the knowledge which could solve a systems’ particular problem. The fuzzy rule acts as a resource to the fuzzy inference system. The fuzzy inference system makes successive decisions about which rules are most relevant to the current situation and applies the actions indicated by these rules. The input fuzzifier takes the crisp numeric inputs and converts them into a fuzzy membership form needed for the fuzzy inference system. At the output, the defuzzification interface combines the conclusions reached by the fuzzy inference system which converts them into crisp numeric values used as control variables.
Steps for designing the FLC:

Step 1: Design inputs and outputs for the FLC.

Step 2: Define frames for fuzzy variables.

Step 3: Assign membership values to fuzzy variables.

Step 4: Create a rule base.

Step 5: Choose scaling gains for the variables.

Step 6: Fuzzify inputs to the FLC.

Step 7: Determine which rule fire.

Step 8: Infer the output recommended by each rule.

Step 9: Aggregate the fuzzy outputs recommended by each rule.

Step 10: Defuzzify the aggregated fuzzy set to form crisp output from the FLC.

2.4.2 Fuzzy Decision Making System

In the networked process considered here, it is assumed to control different motors at different speeds according to the data given by the fuzzy decision making system. The NCS Considers here that four sensor nodes are to be present which gives four different data’s according to the density of data’s present at each node. These clusters of nodes are sensed and the data’s are given to the fuzzy decision making system that decide the speed of the motor. According to the density of data present at each node the FLC decides the control voltage which tunes the speed of the target motor in the network. This speed is controlled by the self tuning fuzzy PI controller. Figure 2.10 shows the fuzzy decision making system.
The input is fuzzified and in this case, four inputs and four outputs are considered. It can be converted into multiple inputs and multiple outputs as required. The general block diagram is shown in Figure 2.11.

Figure 2.11 explain about the fuzzy decision making of a distributed system in which the input is from four sensors S1, S2, S3 and S4 nodes given to a FLC through the network.
For simulation a single closed loop comprising of a motor speed sensor is considered with the sensor data given as input to the fuzzy decision making system. The input is then fuzzified by using trapezoidal membership function. The linguistic variables for input are Less (L), Medium (M) and High (H). In the output singleton membership functions are used. The rules are framed to achieve the rated motor speed on minimizing the error of the control voltage.

2.5 NETWORK BASED SELF-TUNING FUZZY PI CONTROLLER

As delay in the network based system increases, the system becomes unstable with overshoot and oscillations, a new approach of tuning the PI parameters are done using fuzzy logic. On the other hand, fuzzy control provides a formal methodology for representing, manipulating and implementing a human’s heuristic knowledge about how to control a system. This is also a convenient method for constructing nonlinear controllers by using heuristic information obtained from experience. The structure of self tuning fuzzy PI controller for network based DC motor is shown in Figure 2.12.

![Figure 2.12 Structure of self tuning PI controller](image-url)
In Figure 2.12, y(t) is the actual output of the controlled plant, r(t) is the reference input, e(t) = r(t) - y(t) is the error between r(t) and y(t), de(t) is the derivative of error. The PI controller parameters are tuned by fuzzy inference, which provides a non linear mapping from the error signal e(t) and the derivative of error de(t) to the PI parameters, K_P and K_I. These parameters are changed within the initial parameter boundary.

2.5.1 Structure of fuzzy inference system

Fuzzy system consists of two inputs and two outputs in which the inputs and the outputs are linked by Mamdani based fuzzy rules (Timothy J. Ross, 2002). The inputs are error and derivative of error. The outputs are K_P and K_I which are the parameters of PI controller.

2.5.1.1 Normalization

Consider that the variable ranges of K_P and K_I are [K_pmin, K_pmax] and [K_lmin, K_lmax] respectively. In order to obtain feasible rule bases with high inference efficiency the PI parameters must be normalized over the interval [0, 1] as follows:

\[
K'_P = \frac{K_P - K_{p\min}}{K_{p\max} - K_{p\min}}
\]

\[
K'_I = \frac{K_I - K_{i\min}}{K_{i\max} - K_{i\min}}
\]

(2.20)

(2.21)

2.5.1.2 Fuzzification

Figure 2.13 shows the membership functions of input variables. The maximum error and the maximum derivation of error ranges are chosen for membership functions.
In this thesis, the linguistic levels assigned to the input variables \( e(t) \) and \( de(t) \) are S: Small; M: Medium; L: Large.

VS: Very Small; S: Small; Z: Zero; L: Large; VL: Very Large are the fuzzy sets of \( K_p' \) and \( K_i' \) which are the output variables and the membership function of the fuzzy set are shown in Figure 2.14.

2.5.1.3 Fuzzy rules

Using the above fuzzy sets of the input and output variables fuzzy rules are composed as follows:

Rule1: If \( e(t) \) is small and \( de(t) \) is small then \( K_p' \) is Very Small and \( K_i' \) is Very Small, where small and Very small are the fuzzy sets of the input and output variables used in the fuzzy rules. Here nine rules are used for this
system. The other rules are formed from the Fuzzy Associative Memory (FAM) table as shown Table 2.2.

Table 2.2 FAM table

<table>
<thead>
<tr>
<th>e(t) \ de(t)</th>
<th>S</th>
<th>M</th>
<th>L</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>VS</td>
<td>S</td>
<td>M</td>
</tr>
<tr>
<td>M</td>
<td>S</td>
<td>M</td>
<td>L</td>
</tr>
<tr>
<td>L</td>
<td>M</td>
<td>L</td>
<td>VL</td>
</tr>
</tbody>
</table>

Generally the fuzzy rules are dependent on the plant to be controlled and the type of the controller. The Mamdani fuzzy inference method used here is MAX-MIN and the defuzzification method is centroid.

2.5.2 Fine-Tuning Fuzzy PI Controller

The self tuning fuzzy PI controller can be further improved by increasing the membership function and rules which makes a fine tuning. The fuzzy controller is a two-dimension fuzzy logic controller with two inputs and single output. Its function is to make the actual system output $y(t)$ track the reference input $r(t)$ through adjusting control variable.

There are seven linguistic values for each linguistic variable are defined on the fuzzy universe of discourses, such as Positive Big (PB), Positive Middle (PM), Positive Small (PS), Zero (ZE), Negative Small (NS), Negative Middle (NM), and Negative Big (NB). The choice of membership functions has definite influence on the control performance. Generally speaking, the steeper the membership function is, the higher the resolution is
and higher the control sensitiveness is. On the contrary, the flatter the membership function is, the better the stability is and the stronger the robustness is. Because triangular membership function has the advantages of simple computation, ease of realization and good control performance. It is adopted to represent the membership functions of the inputs and the output of the fuzzy controller, which are shown in Figure 2.15 and Figure 2.16.

This improves the system performance as shown in Figure 2.17. Here the membership functions for the input variables are increased.

![Figure 2.15 Membership functions of input variables for fine tuning](image)

The same membership functions are chosen for output variables. The rules have been increased to 49 in order to achieve a fine performance as shown in Table 2.3.

![Figure 2.16 Membership functions of output variables for fine tuning](image)
Table 2.3  FAM table for Fine tuning

<table>
<thead>
<tr>
<th>e(t) \ de(t)</th>
<th>NB</th>
<th>NM</th>
<th>NS</th>
<th>Z</th>
<th>PS</th>
<th>PM</th>
<th>PB</th>
</tr>
</thead>
<tbody>
<tr>
<td>NB</td>
<td>VS</td>
<td>VS</td>
<td>S</td>
<td>S</td>
<td>S</td>
<td>M</td>
<td>M</td>
</tr>
<tr>
<td>NM</td>
<td>VS</td>
<td>S</td>
<td>S</td>
<td>S</td>
<td>M</td>
<td>M</td>
<td>M</td>
</tr>
<tr>
<td>NS</td>
<td>S</td>
<td>S</td>
<td>S</td>
<td>M</td>
<td>M</td>
<td>M</td>
<td>M</td>
</tr>
<tr>
<td>Z</td>
<td>S</td>
<td>S</td>
<td>M</td>
<td>M</td>
<td>M</td>
<td>M</td>
<td>L</td>
</tr>
<tr>
<td>PS</td>
<td>S</td>
<td>M</td>
<td>M</td>
<td>M</td>
<td>M</td>
<td>L</td>
<td>L</td>
</tr>
<tr>
<td>PM</td>
<td>M</td>
<td>M</td>
<td>M</td>
<td>M</td>
<td>L</td>
<td>L</td>
<td>VL</td>
</tr>
<tr>
<td>PB</td>
<td>M</td>
<td>M</td>
<td>M</td>
<td>L</td>
<td>L</td>
<td>VL</td>
<td>VL</td>
</tr>
</tbody>
</table>

2.6  PERFORMANCE ANALYSIS OF FUZZY BASED NCS

Using the MATLAB/SIMULINK blocks the NCS system is modeled and simulation is carried out. The analysis of NCS performance with conventional PI and fuzzy based PI controller is done by subjecting the system to track a reference speed of 50(rad/s) under 4ms NID.

Figure 2.17 shows a comparison of self tuned, fine tuned and without tuning PI controller response. The fine tuning has better settling time of 0.035seconds with faster rise time and also has zero steady state error.

Figure 2.18 and Figure 2.19 shows the controlled output response of the motor speed at different reference speeds. The output response is obtained to simulate the effect of fuzzy PI controller to three control loops communicating through the network as modeled for the assumed decentralized system. It shows that self tuning fuzzy PI controller is used in order to control the motor speed at different references.
Figure 2.17 Comparison of PI controller with and without tuning

Figure 2.18 Output responses at different reference speeds

Figure 2.19 Controlled output response for fuzzy decision
The section presented simulated results for the self and fine tuning fuzzy based PI controller designed for speed compensation of effect of random delays on a networked motor. From the results it is observed that NCS has better performance when it is incorporated with fine tuning FLC, because the performance indices like overshoot and steady state error are close to zero with improved settling time.

2.7 STATE ESTIMATOR MODELING FOR COMMUNICATION REDUCTION

When a decision is taken the process can be controlled by the controller in which the plant, sensor and controller are in remote places and are interconnected through a shared communication medium which is termed as Networked Control Systems (NCS). This requires a sophisticated control technique to maintain the stability of the system.

When a control system is subjected to communication constraints, the policies that govern how communication medium is used can have a direct effect on the design of the control policy or vice versa. It is required to make the problem easier to solve by assuming, the communication policy (like protocol) fixed and designing an estimator to reduce the rate at which packets are transmitted. A networked decentralized system is in demand of control strategies that reduce data communication among control loops with maintaining system stability. Techniques like data compression, scheduling algorithms etc., can be a viable solution. This section presents the need for estimator at each node which could minimize the amount of actual communication required in the network. The proposal is a novel method that uses a state estimator giving improved performance of network by reducing data transmission rate.
The simulation is carried out to analyse the behavior of distributed system with state estimators in each node of the network. Each of the nodal estimators is considered as SISO. When communication delays are negligible, the performance of DCS should be same as that of a good centralized MIMO. It is necessary to design an estimator with Communication Logic (CL) to decide the condition at which the data can be transmitted from one node to other without affecting the system stability.

2.8 STANDARD MIMO SYSTEM

The modeling of state estimator for distributed MIMO system can be developed from the knowledge of a standard MIMO system. Consider a discrete time causal Linear Time Invariant (LTI) MIMO system as shown in Figure 2.20, whose state model can be described as follows:

![Figure 2.20 Standard MIMO closed loop system](image)

\[
X(k+1) = AX(k) + BU(k) + BD(k) \\
Y(k) = CX(k)
\]  
with the state feedback controller as

\[
U(z) = K(z) E(z) = K(z) [R(z) - (CX(z) + N(z))]
\]

where \( k \) is the time index associated with the sampling time \( T \) in discrete time domain, \( A \in \mathbb{R}^{n \times n}, B \in \mathbb{R}^{n \times m}, C \in \mathbb{R}^{r \times n} \) are system, input and output matrices respectively. The parameters \( R, D, N, \) and \( K \) are the reference, disturbance,
noise (sensor noise) and closed loop controller respectively. The states of the system can be expressed as

\[ X(z) = [zI - \psi(z)]^{-1}[BK(z)R(z) - BK(z)N(z) + BD(z)] \]  \hspace{1cm} (2.24)

where

\[ \psi(z) = (A - BK(z)C) \]

If the number of inputs and outputs is large and the system is physically distributed over a large area, the MIMO architecture will become complex, hence the MIMO implementation over a network is advantageous. The performance of the standard MIMO and distributed MIMO will remain same as long as there is no communication delay. Hence a modification in state model of standard MIMO has to be made for implementation of distributed MIMO.

2.8.1 Distributed Implementation of MIMO System and State Estimator Framework

Figure 2.21 give the distributed implementation of MIMO system. For the distributed configuration to achieve the same performance as the centralized case each error \( E_i \) must be communicated over the network to other nodes at every sample time.

The proposed state estimator is a centralized MIMO system which contains the nominal plant \( P \) with \( Ao, Bo, Co \) (which is used to design the controller) instead of the actual plant \( \Pi \) with \( A, B, C \). On the \( i \)th node, the input to the controller is composed of the actual error \( E^* \) for the output associated with that node, and the estimated errors \( E^*_j, j \neq i \) for all other nodes.
In addition, the proposed state estimator includes communication logic which compares the actual output of the $i^{th}$ node with the estimated output of the $i^{th}$ node and manages the communication from the $i^{th}$ node to the entire system. For example, if the difference between the actual and estimated output of the $i^{th}$ node is greater than a threshold value, either the actual output or the estimated states of the $i^{th}$ node are broadcast to the entire system. At that time, the estimator states representing the $i^{th}$ node are updated to reflect their current values in all estimators. The detailed idea of the proposed estimator for the subsystem is shown in Figure 2.22. This threshold communication logic is used.
The estimator dynamics between communication instants can be described by the following equations.

\[
\dot{X}(k+1) = A_\theta \dot{X}(k) + B_\theta \hat{U}(k) \\
\hat{Y}(k) = C_\theta \dot{X}(k)
\]

with

\[
\hat{U}(z) = K(z)\hat{E}(z) \\
= K(z)[R(z) - C_\theta \dot{X}(z)]
\]  

(2.25)

When there is a communication from the \(i^{th}\) node, the estimators in all nodes update their states to reflect the current actual value of the system output or state.
2.9 ANALYSIS OF STATE ESTIMATOR FRAMEWORK

An example of a networked two-agent system with state estimator and communication module is shown in Figure 2.23 and the schematic diagram of the Control/Communication module is depicted in Figure 2.24. The basic idea is to let one node use estimated states for control actions or otherwise broadcast its current states to other nodes if estimation is not possible by it.

![Figure 2.23 Networked two-agent systems with state estimator and communication module](image)

Each node is modeled to have one estimator which computes its states and the other nodes, based on difference of states using the estimation algorithm. The main functionality of the Control/Communication module is to compute the difference of the true and estimated states of the nodes, control the frequency of communication, and update the estimated states by the states of other nodes.
For example consider communication between two nodes in Figure 2.23. Estimator 1 computes $X_{1e}$ and $X_{2e}$ and Estimator 2 computes $X_{1e}$ and $X_{2e}$ as well. At a normal scenario, (i.e.,) no communication required, node 1 is operating based on its own state $X_1$ and the estimated state of node 2, $X_{2e}$. When Control/Communication module 1 receives new $X_2$, it informs node 1 to use the newly arrived $X_2$ instead of the estimated state $X_{2e}$. In addition, Control/Communication module 1 broadcasts $X_1$ to node 2 if $|X_1 - X_{1e}|$ is larger than a predefined threshold, say $H$, which is the principle used here to incorporate an event based communication logic. Similar estimation and communication mechanisms are designed at node 2 and other nodes. The same concept can be extended to an n-node system, where there are n estimators of states and communication modules. Utilizing the locally estimated states can save certain amount of communication cost/bandwidth and also achieve good control performance because communication is done only as an event based approach.

**Figure 2.24 Schematic diagram of the communication module**
2.9.1 MIMO System with Proposed Estimator

Consider the system described above. When this system is implemented with the proposed estimator the equations become as

\[ X^*(k+1) = AX^*(k) + BU^*(k) + BD(k) \]
\[ Y^*(k) = CX^*(k) \]

(2.26)

The controller output \( U^* \) differs from \( U \), since \( U^* \) is computed using both actual and estimated errors. The difference between actual and estimated outputs can be described as

\[ \Gamma = (Y^* + N) - \hat{Y} \]
\[ = (R - \hat{Y}) - (R - (Y^* + N)) \]
\[ = \hat{E} - E^* \]

(2.27)

The actual controller output in the system with the estimator \( U^* \) are discussed by (Yook et al 2002).

2.10 MASS-SPRING-DAMPER SYSTEM – A CASE STUDY

The utility of the proposed framework through simulation with experimental validation of the communication of estimates between nodes is done. A linear mass-spring-damper system is considered for analysis using MATLAB.

A two degree of freedom mass-spring-damper system is shown in Figure 2.25.
The dynamical system can be described by the following differential equations.

\[
\begin{bmatrix}
\dot{x}_1 \\
\dot{x}_1 \\
\dot{x}_2 \\
\dot{x}_2
\end{bmatrix} = A \begin{bmatrix}
x_1 \\
x_1 \\
x_2 \\
x_2
\end{bmatrix} + B \begin{bmatrix}
u_1 \\
u_2
\end{bmatrix}
\]

(2.31)

with

\[A = \begin{bmatrix}
0 & 1 & 0 & 0 \\
-k_1/m_1 & b_1/m_1 & k_1/m_1 & b_1/m_1 \\
0 & 0 & 0 & 1 \\
k_1/m_2 & b_1/m_2 & -k_1 + k_2/m_2 & b_1 + b_2/m_2
\end{bmatrix},
\]

\[B = \begin{bmatrix}
0 & 0 \\
1/m_1 & 0 \\
0 & 0 \\
0 & 1/m_2
\end{bmatrix},
\]

\[C = \begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 0 & 1 & 0
\end{bmatrix}
\]

(2.32)

Let the actual and nominal parameters be as follows:

\[k_1=1.1 \quad b_1=0.22 \quad m_1=1.1 \implies A, B, C \text{ (actual)}
\]

\[k_2=4.4 \quad b_2=0.11 \quad m_2=2.2
\]

\[k_{1o}=1 \quad b_{1o}=0.2 \quad m_{1o}=1.1 \implies A_o, B_o, C_o \text{ (nominal)}
\]

\[k_2=4.4 \quad b_2=0.11 \quad m_2=2.2
\]

where the units of k, b, m are N/m, Nm/s, and kg, respectively.
A suitable controller is designed using standard design method. The controller is given as

\[
K = \begin{bmatrix}
K_{11} & K_{12} \\
K_{21} & K_{22}
\end{bmatrix}
\]

(2.33)

The value of \(K_{11}, K_{12}, K_{21}, K_{22}\) is obtained as given below.

\[
K_{11} = \frac{40s^3 + 145.6s^2 + 178.9s + 71.23}{0.1s^4 + 2.46s^3 + 19.55s^2 + 48.74s} + (40s^3 + 145.6s^2 + 178.9s + 71.23)
\]

\[
K_{12} = \frac{-100s^3 - 520s^2 - 896s - 499.2}{s^5 + 24.15s^4 + 175.1s^3 + 85.35s^2 + 422.5s} + (-100s^3 - 520s^2 - 896s - 499.2)
\]

\[
K_{21} = \frac{-40s^3 - 145.6s^2 - 178.9s - 71.23}{0.5s^5 + 9.9s^4 + 51.2s^3 + 19.55s^2 + 48.74s} + (-40s^3 - 145.6s^2 - 178.9s - 71.23)
\]

\[
K_{22} = \frac{100s^3 + 520s^2 + 896s + 499.2}{0.1s^4 + 3s^3 + 29s^2 + 84.5s} + (100s^3 + 520s^2 + 896s + 499.2)
\]

This system is implemented in a two node distributed control architecture with the proposed estimator scheme.

2.10.1 Implementation of state estimator framework

The above MIMO system (mass-spring damper system) is a two input and two output systems. The system is modeled as two sub systems \((u_1, u_2)\) to \((x_1, x_2)\). The above designed controller is implemented in a decentralized manner. The controller part \([K_{11}, k_{12}]\) is placed at node1 (subsystem1) and \([K_{21}, K_{22}]\) is placed at node2 (subsystem2). For the simulation, the sampling rate is set as \(Ts = 0.01\) sec. The system is simulated with noise and disturbances.
The disturbance at each node is considered as $D_1 = D_2 = 0.1 \sin (10t)$ and white gaussian noise as $N_1 = N_2 = 0.001$. The simulations were run with different threshold values viz., 0.01, 0.04 and 0.07. The bound on the performance index is a function of $H$ hence the same threshold value is chosen for both outputs.

Each node is built in a personal computer (PC), configuring each PC as a subsystem. Hence, communication between subsystems is considered as communication between two PCs. The same method can be implemented for $n$-subsystems. Serial communication is implemented by connecting the systems by a null modem cable for data transmission and reception.

Hence the plant model and the controller design is done for the above system whose nodes communicate with each other through null modem cable and the simulation is carried out in MATLAB/SIMULINK and is shown in Figure 2.26.

The system is simulated without and with the controller designed above with standard step source as reference input.

![SIMULINK Model of plant and Controller](image)

**Figure 2.26** SIMULINK Model of plant and Controller

Figure 2.27 shows the output response of a plant without controller with step input is given to the plant. Figure 2.27 shows that the output $Y_1$ and $Y_2$ takes long time to settle at the required set point.
Figure 2.27  Step response of plant without controller

Figure 2.28  Step response of plant with controller

Figure 2.28 shows the output response of a plant with controller. For the given step input plant it is evident from the Figure 2.28 that the output Y1 and Y2 settles much faster at the required set point than the earlier one without controller.

For the above system to be implemented in a distributed fashion, the controller is decentralized where a part of a controller is placed at each node to take the required control action. The communication between nodes
which is essential for overall system performance is established through serial communication between nodes and simulation specifications are given below:

Sample time: 0.01sec  Data Format: Binary  Input Buffer size: 8
Precision     : 64 Bit Float  Interface       : Serial  Baud rate     : 9600bps

The two PCs are thus connected through Null modem cable which is the network for the two systems to communicate.

2.10.2 Distributed Implementation of System

Figure 2.29 shows the model of distributed implementation of the system.

![SIMULINK Model of distributed implementation of the system](figure)

Each subsystem is considered as a node and the controller output is given as input to the plant. Disturbance (D1 and D2) is added to each node and white Gaussian noise is added as measurement noise. The difference between the actual output (Y) and the reference input is calculated as $E^*$ and fed as input to the controller. This error is also fed into the network from the
current estimating node to the other nodes. There is simultaneous reception of data from other nodes through the same network. For this distributed configuration to achieve the same performance as the centralized case, each error $E_i$ must be communicated over the network to the other nodes at every sample time.

Figure 2.30 shows the SIMULINK model for the proposed estimator framework. An estimator is placed at each node compares the actual value of the state with the estimated value. If the difference between them is within bound i.e. within the predefined threshold value $H$, no data is transmitted through the network. Else, the actual state is transmitted through the network using which the estimators at all other nodes get updated.

![SIMULINK model for the proposed state Estimator scheme](image)

The above system is simulated under various threshold levels. Figure 2.31, 2.32 and 2.33 shows the output of the communication module of both the nodes. Magnitude 1 of CL (Communication Logic) denotes that the
difference between the actual and the estimated value exceeds the predefined threshold $H$ and thus indicates the instant of communication through the network between the subsystems. Magnitude 0 indicates that no communication occurs between nodes as the difference between the actual and the estimated value does not exceed the predefined threshold $H$.

**Figure 2.31** Instance of communication of subsystem 1 to 2 (Threshold $H=0.01$)

**Figure 2.32** Instance of communication of subsystem 1 to 2 (H=0.04)
A state updating algorithm is implemented in the Figure 2.29. If the difference between the actual and estimated output of the $i^{th}$ node is greater than a threshold value, either the actual output or the estimated states of the $i^{th}$ node are broadcast to the entire system. At that time, the estimator states representing the $i^{th}$ node are updated to reflect their current values in all estimators. Figure 2.34, 2.35 and 2.36 shows the outputs of communication after the updating of states.
The communication frequency depends on the behavior of $\Gamma = Y - \dot{Y}$. In other words, when $|\Gamma_i| \geq H_i$ is true, a communication from the i subsystem occurs and $\Gamma_i$ becomes a discontinuous variable bounded by the threshold value $H_i$. Between the communication instants, $\Gamma$ can be described...
as \( \Gamma = (Y^* + N) - \hat{Y} \). The communication logic is seen bounded by the Threshold Value (H=0.07).

**2.10.3 Performance Index**

The performance Index \((P_I)\) is the difference between outputs of the distributed system with the proposed estimator and the nominal centralized system. It is represented as \( P_I \). It is calculated for both the nodes of the system and is as shown in Figure 2.37.

\[
P_I(K) = Y^*(K) - Y(K)
\]  

(2.35)

![Figure 2.37 Performance Index (y*-y) of both subsystem 1 and 2.](image)

Thus the performance index is also bounded by the threshold value (H=0.07).
2.10.4 Network Usage

Network usage ($N_u$) is defined as ratio of number of messages sent to maximum number of messages. For a sampling time of 0.01 second, with the simulation time of 1 second, the maximum number of messages is 100. Thus, the network usage can be calculated for various time period and various threshold value, $H$.

Table 2.4 below gives the result details for the plant (A Mass-Spring - Damper system) considered. The network utilization and the communication time for various threshold value say $H=0.01, 0.04, 0.07$ is simulated.

**Table 2.4  Performance for various Threshold $H$**

<table>
<thead>
<tr>
<th>THRESHOLD</th>
<th>$H = 0.01$</th>
<th>$H = 0.04$</th>
<th>$H = 0.07$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sampling period</td>
<td>0.01s</td>
<td>0.01s</td>
<td>0.01s</td>
</tr>
<tr>
<td>Network Usage, $N_u$</td>
<td>69 – 75 %</td>
<td>14 – 16 %</td>
<td>4 – 5 %</td>
</tr>
<tr>
<td>Communication time utilization</td>
<td>57 - 63 %</td>
<td>12-13 %</td>
<td>3 – 4 %</td>
</tr>
</tbody>
</table>

The simulation is carried on a Mass-Spring-Damper system and the result is displayed. From the above table it is clear that network utilization and the communication time utilization depends on the threshold value, $H$.

2.11 CONTRIBUTION OF THIS CHAPTER

The performance of NCS degrades due to the presence of network induced delay. In real time systems either control policies or communication policies are kept constant. This chapter discussed the issues on the effect of communication disturbances on NCS performance which degrades due to the
network delay and is compensated by modeling a fuzzy based PI controller. The performance of fuzzy based system is simulated for various delay conditions the stability of NCS is verified with fuzzy self tuning and fine tuning.

As the scale of a distributed control system and its communication frequency increase, the performance of the system can suffer significantly due to network-induced delays. A state estimator framework is proposed for a Mass-Spring-Damper system. Using the states of the estimator designed above, communication logic has been designed. The communication logic compares the estimated values with the actual output. If it exceeds a pre defined threshold $H$, given by the design engineer, it transmits the estimated value to all other nodes for updating. Else, no communication occurs between other nodes. The system is divided into subsystems and the communication is established between them using a Null Modem cable. The system is simulated using MATLAB/SIMULINK for various threshold values and the corresponding Network usage and communication time utilization is calculated which proves that the communication between nodes is reduced to a great extent.