Chapter 6

Multi-sensor Track Data Generator and Real-time Visualization System

6.1 Model-6: Multi-Sensor Track Data Generator Design and its Challenges

Overview

There exists anxiety among engineers as well as scientists before a test of defense equipment. This happened in the aviation sector too, where each time a flight took off, it puts on constant monitoring by the air traffic controller. The real-time test is expensive. And it is also dangerous if it didn't go well as per plan. So the need for a well-structured simulator is a necessity. The simulator provides several advantages and is known as the backbone for modern aviation training programs and the air-borne vehicle test programs. We have developed a Multi-Sensor Track Data Generator (MSTDG). The MSTDG calculates the tracking information that would be sent by the tracking sensors like electro-optical tracking system (EOTS), Radar and Telemetry, and sends it to the Data Processing Server. Those tracking information is calculated using various parameters like the nominal data of trajectory, chamber pressure and body rates, the tracking sensor coordinates and the launch point coordinates, and is as per the format of the tracking information sent by the tracking sensors and also at the rate at which tracking sensors send data. This information generated by Simulator is used for validation of many systems of the server, various real-time displays, and parameters related to the target object. The Simulator has an extensive GUI allowing configuration of necessary parameters and displaying nominal data as it is sent.

6.1.1 Introduction

One of the primary objectives of any test related to defense and flight operation is safety, data tracking and success. Those objectives should meet by all means. Simulation plays a pivotal role in the effectiveness of a mission. It is a tool for achieving mission success. It helps before an event occurs as well as in post-mission analysis. This benefits the organization to look into the desired target result before a final test. From the simulation, it can verify that the data displayed are pertinent, easy to analyze and function well with desired procedures. So it gives a prediction about the upcoming results. These importance of simulation motivated us to develop a well-structured simulator for the smooth operation of tracking data, data validation and video display data synchronization.

The electro-optical tracking system in short EOTS is an object detection or object targeting system. The term "electro-optic" refers to its two working components one is electronic another is optical, both combine works to locate a real object of interest. EOTS uses 3 types of tracking methods of an object like edge tracking, centroid tracking, and correlation tracking. It also uses a high-speed camera IR camera to detect an object. This device computes the coordinates of the object of interest by the Triangulation method.

Radar is the acronym of Radio Detection and Ranging. It is a device that uses radio waves to determine how far an object is present, velocity and proper positioning of the object. Radar consists of a transmitter that transmits the radio waves and there is a receiving antenna. A receiver used to determine the properties of the radio waves. Sometimes the same antenna is used both for transmitting and receiving. The Radar signals that are emitted touch the object and is reflected

back, but some of them gets penetrated. The ones that are reflected make the Radar work.

Radar is used to identify objects by using radio waves and that is to determine what range of unknown objects is and what angle is there and what is the velocity of that object. Radar can be simulated by the small inexpensive Arduino shown by Ghoghre et al. [24]. The main difference between the Radar the EOTS is that EOTS gives a high precision more accurate position of the object in a low range or height.

Telemetry is an automatic communication process by which some physical quantities can be measured and collected data from the remote or such a place where it is not possible to reach easily and then data can be transmitted to a receiver station for logging, monitoring and analyzing.

The working process of a telemetry system can be explained like this manner, sensors at source (remote location) measure electrical data (voltage, current) or physical data (temperature, humidity) and convert it to specified electrical data. Using a multiplexer all collected data are combined with timing data and send to a receiver. At the receiver station, the data stream is separated into original data. Then these data are processed as per need. Telemetry can be done using both analog and digital equipment.

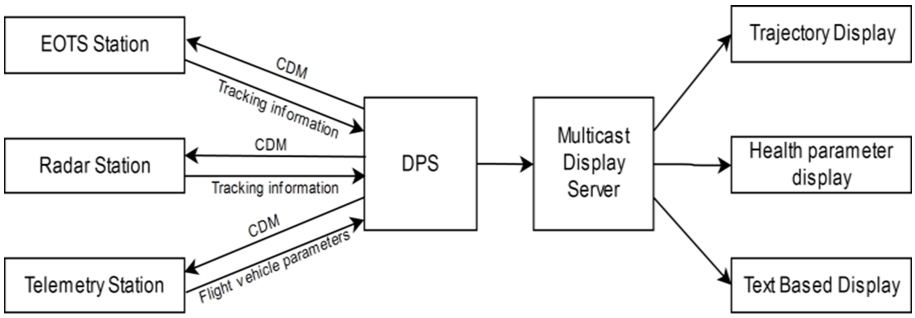

Figure 6.1: Block diagram of the existing system

CPUs	Minimum Quad Core CPU with 3.2 GHz
Memory	Minimum 8 GB of DDR3
Storage	Minimum 1 TB SATA
Networking	Onboard Gigabit Ethernet NICs
Video memory	Minimum 2 GB of GDDR5
LED Display	Minimum 1920x1080 resolution and 18” size

Table 6.1: Hardware Requirements

6.1.2 Requirements specifications

Hardware Requirements

The specification of hardware is shown in Table 6.1.

Software Non-functional Requirements

It can run in Windows 7 and also in Linux operating system. For software development, NetBeans IDE 8 was used. It required Java Development Kit 8.

Software Functional Requirements

- Edit settings, save settings and load settings on initialization
- Graphical representation of nominal data
- Visualization of flight parameters
- Create an EOTS packet
- Create Radar packet
- Create a Telemetry packet
- Send packets
- Control of data transmission

6.1.3 Context and Data Flow Diagram

Inputs required to create the tracking information sent by the MSTDG and the output packets which contain the simulated tracking information are shown in the context diagram Fig 6.2.

In the level 1 diagram shown in Fig 6.3, the conversions required for creating the packets from the nominal data available are shown. The trajectory coordinates, which are available in ENU format with respect to launch point, are first converted to ECEF format. Then using various input data like location of the tracking station, range, azimuth and elevation noise and bias, the packet sent by the tracking instrument is created.

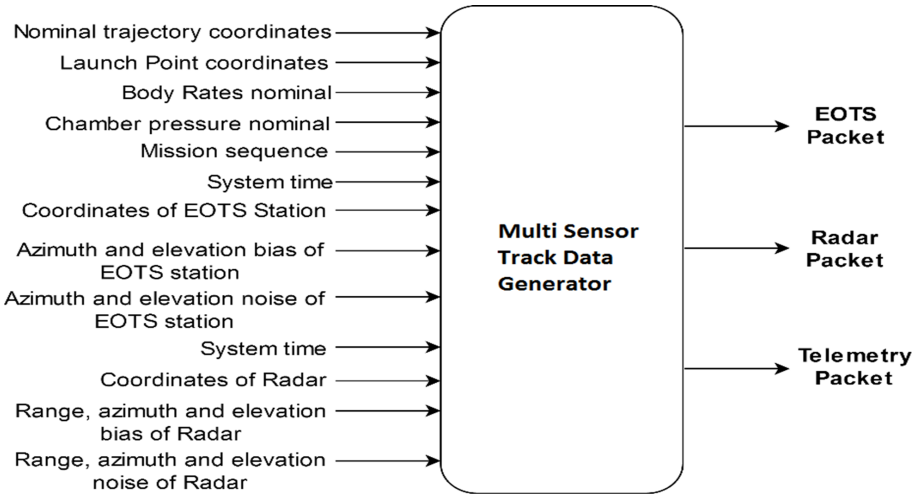

Figure 6.2: Context diagram

In the level 2 diagram shown in Fig 6.4, the conversions required are shown in more detail. The trajectory coordinates converted to ECEF format are converted to ENU format for the EOTS and Radar stations on the basis of the position coordinates of stations. These trajectory coordinates are used to calculate various input data like range, azimuth and elevation as applicable by conversion to the spherical coordinate system, also taking into consideration the range, azimuth and elevation noise and bias of each station. The packet created for telemetry stations contains various health parameters and positional information.

6.1.4 Detailed design

The NetBeans IDE is used for development for the software. It's free and open source. It is a powerful GUI builder and has good profiling and debugging tools. The application was developed using java programming language. As it is an object-oriented programming language, it has a great advantage over other lan-

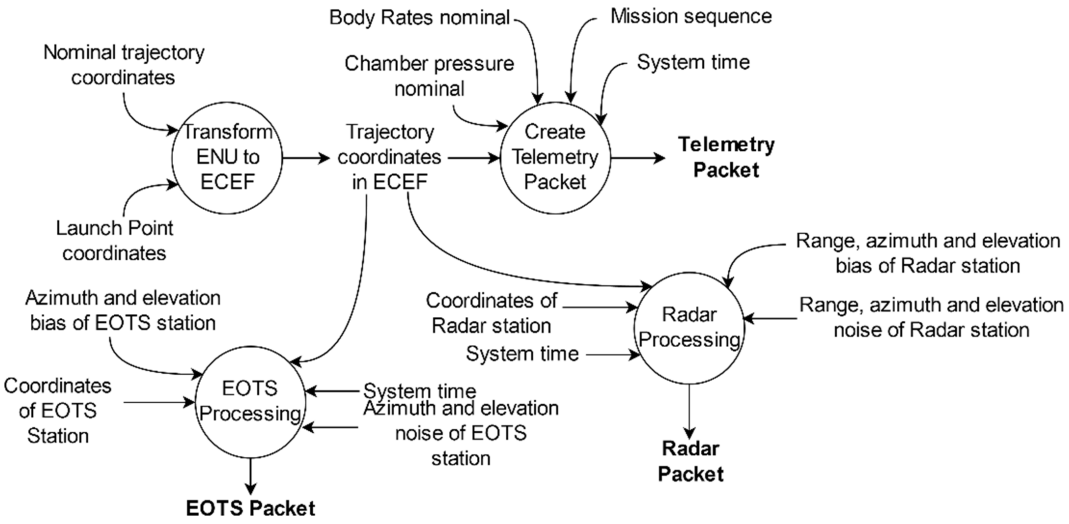

Figure 6.3: Level 1 Data Flow Diagram

guages. Code re-usability and software maintenance are better compared to other programming languages. The details design specification was described below.

Settings and initialization

Enable/disable sending packets: Each station can send packets containing tracking information to 4 servers. It is possible to enable/disable the sending of packets from each station to each server (Fig 6.5).

Edit mission settings: The Server IP is given in IPv4 format. The maximum value for starting port number is 300. It can be specified whether the servers are active or not (Fig 6.6). The position coordinates of the launch point are entered in a degree-minute-second format. The flight azimuth is given in degrees. The mission name can also be entered. The path for nominal files that contain data about trajectory, body rates and chamber pressure of the flight vehicle can be specified. The nominal files must have .txt or .dat format. It can also be specified

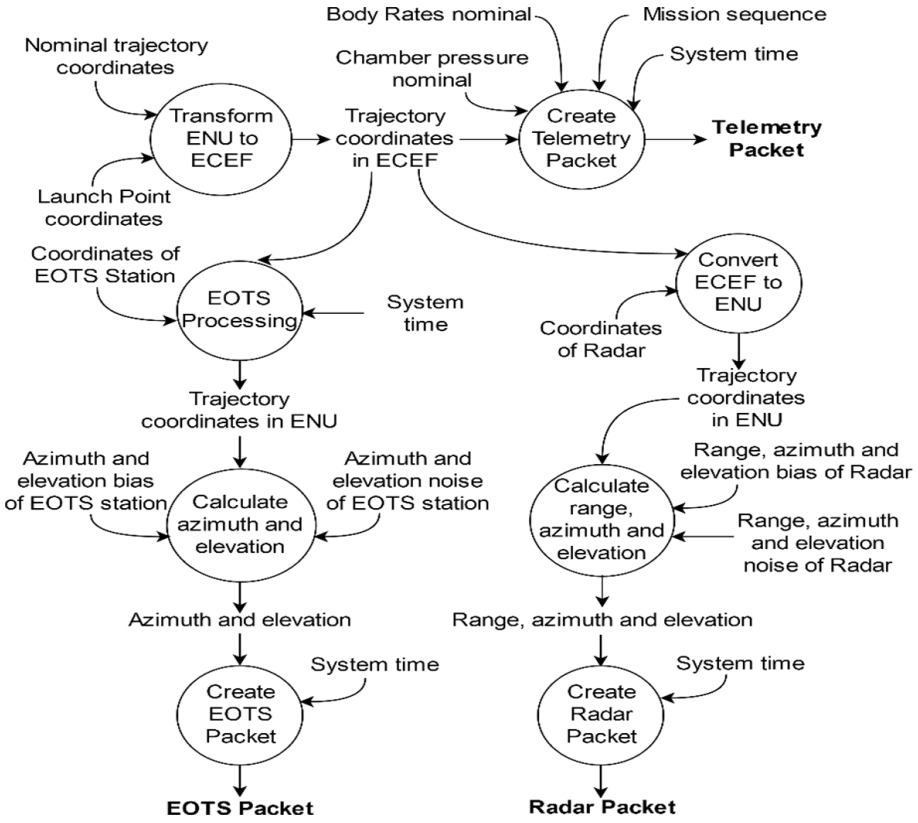

Figure 6.4: Level 2 Data Flow Diagram

whether the body rates and chamber pressure nominal files are to be read (Fig 6.7). The mission sequence events and physical events can also be entered (Fig 6.8 and Fig 6.9)

Edit site settings: The name, site ID and frame length of tracking sensors can be configured. The latitude and longitude are entered in degree-minute-second format. The range, azimuth and elevation noise, and range, azimuth and elevation bias are specified in radians.

Send packets at a specified rate: The rate at which packets are sent from stations can be specified. Packets can be sent every 25 ms, 50 ms, 100 ms or 250 ms. All processing must be completed before the specified time to ensure that

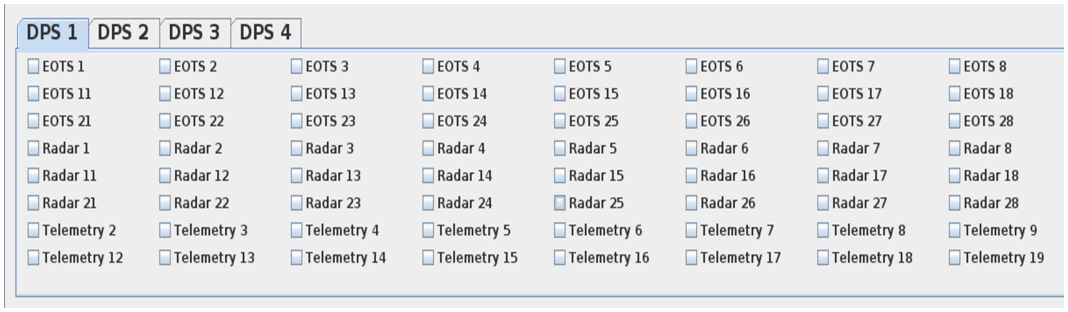

Figure 6.5: Choose whether to send data from each station

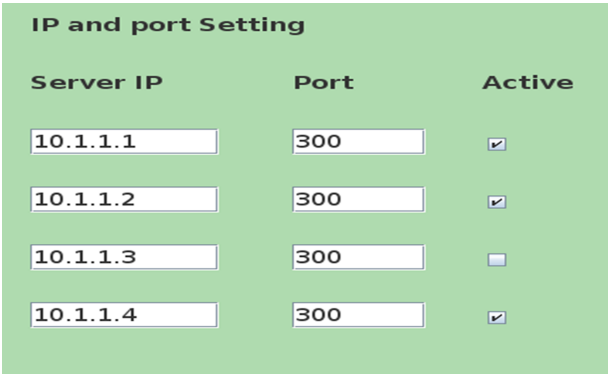

Figure 6.6: Configure Server IP and port

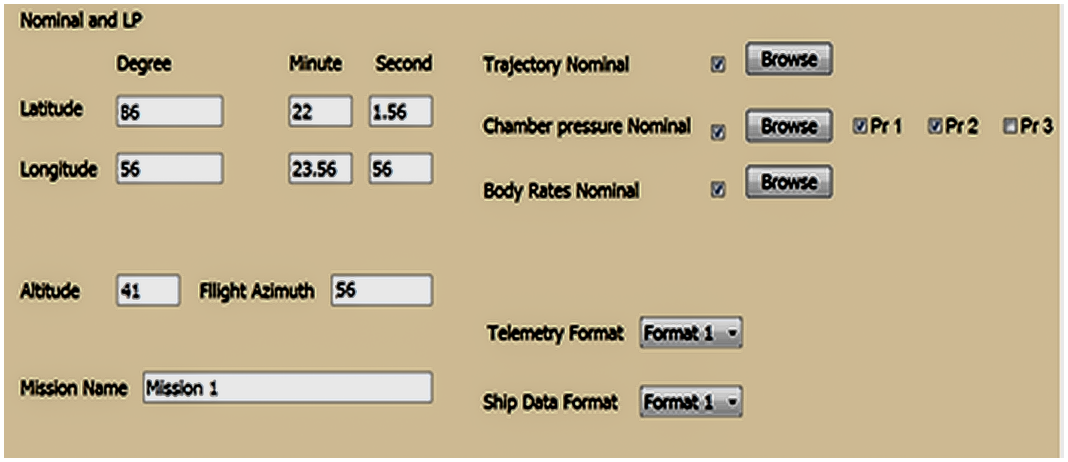

Figure 6.7: Set mission information

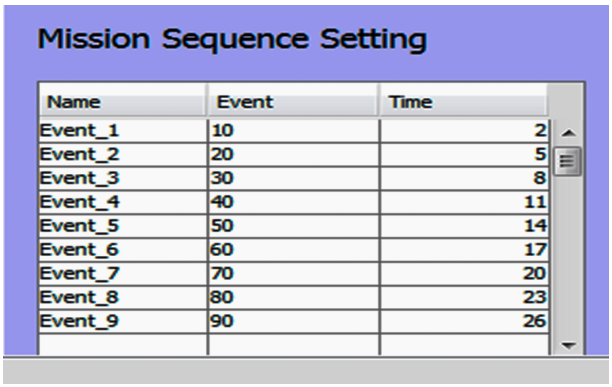

Figure 6.8: Set Mission events

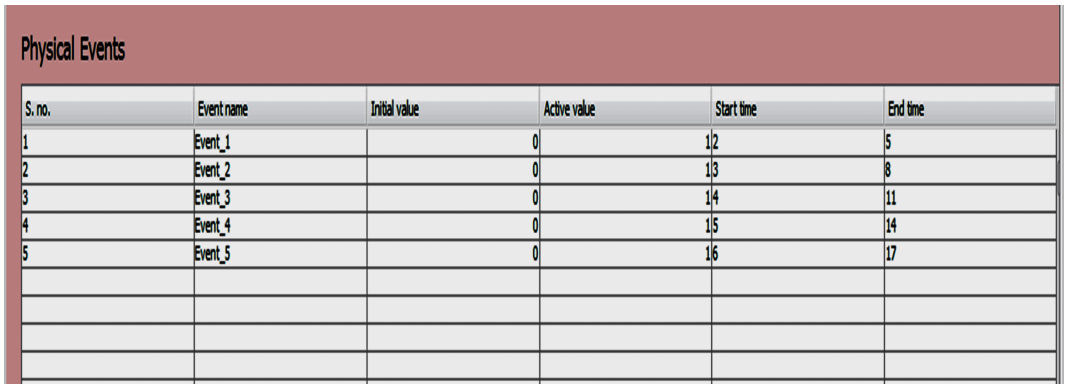

Figure 6.9: Set physical events

the packets are transmitted on time. The timer function used should use the CPU clock cycle.

Save settings: When settings are saved, they are stored in a predefined file. The file has .txt format.

Load settings on initialization: On initialization, if the settings file is not empty, the settings stored in the file are read and the values are displayed in the corresponding data entry fields.

Type of telemetry packet selection: The type of telemetry packet can be selected from a list of available types. The format of the telemetry packet will vary accordingly.

Graphical representation of nominal data

Group graphs into tabs: Nominal data is displayed using graphs. Grouped by the window, the parameters plotted are - downrange vs. altitude, downrange vs. cross-range; velocity vs. time, x, y, z coordinates of velocity vs. time; chamber pressures vs. time; roll, pitch, yaw vs. time; and quaternion angles vs. time (Fig 6.10).

Conditional display of graphs: If the nominal files for a parameter are not chosen, the corresponding graphs will not be drawn.

Update graphs at a specified rate: The graphs are updated at the rate at which packets are sent. At any instant of time, the last point plotted represents the data used to calculate the parameters of the packets sent at that instant.

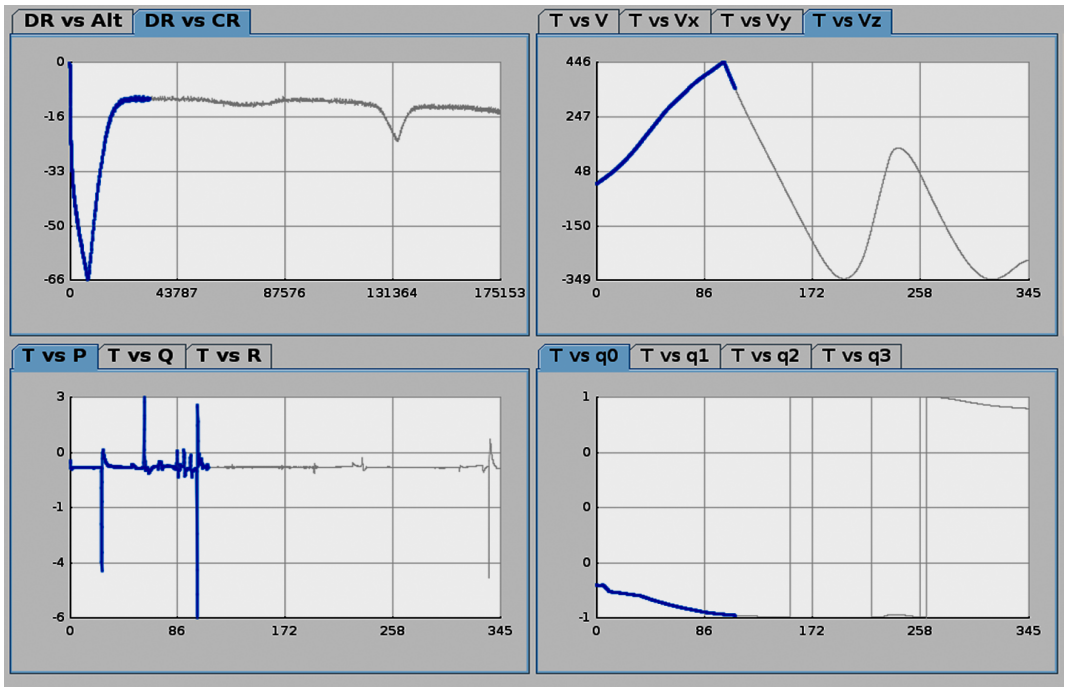

Figure 6.10: Display nominal data

Visualization of flight parameters

The range, altitude, and azimuth of the missile, the number of seconds elapsed after liftoff and the system time in hour: minute: second: millisecond format are displayed. This display is updated every second.

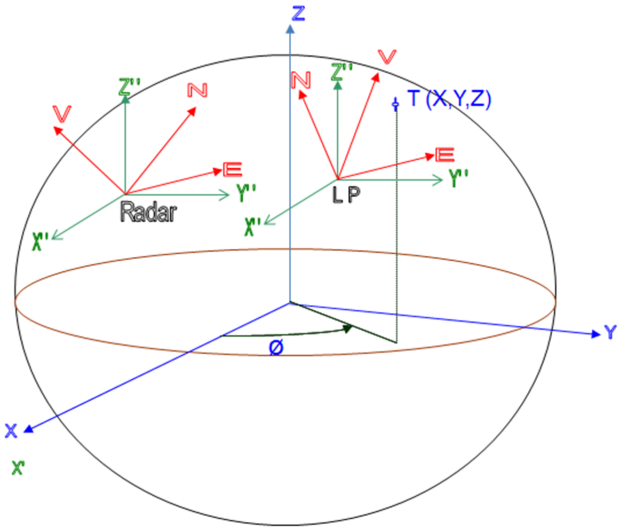

Figure 6.11: Diagram illustrating location of tracking station and launch point

Formulae for the creation of packets for EOTS, Radar and Telemetry

- Obtain coordinates in ECEF format from coordinates ENU format and location of launch point in Latitude-Longitude-Altitude format

$$\begin{pmatrix} x_lp" \\ y_lp" \\ z_lp" \end{pmatrix} = \begin{pmatrix} -sin\mu_lp & -sin\lambda_lp\cos\mu_lp & cos\lambda_lp\cos\mu_lp \\ cos\mu_lp & -sin\lambda_lp\sin\mu_lp & cos\lambda_lp\cos\mu_lp \\ 0 & cos\lambda_lp & -sin\lambda_lp \end{pmatrix} \begin{pmatrix} E_lp \\ N_lp \\ V_lp \end{pmatrix}$$

- Account for the ellipsoidal shape of the Earth

  a=6378137.0 $\qquad er = \frac{a}{1-e^2*\sin\lambda_lp}$

  $X_lp = (er+h_lp)*\cos\lambda_lp*\cos\mu_lp$

  $Y_lp = (er+h_lp)*\cos\lambda_lp*\sin\mu_lp$

  $Z_lp = ((1-e^2)*er+h_lp)*sin\lambda_lp$

  Similarly we get X_r, Y_r, Z_r

- Convert ECEF to ENU coordinates w.r.t. position coordinates of tracking sensor

$$\begin{pmatrix} E_r \\ N_r \\ V_r \end{pmatrix} = \begin{pmatrix} -sin\mu_r & cos\mu_r & 0 \\ -sin\lambda_r \cos\mu_r & -sin\lambda_r \sin\mu_r & cos\lambda_r \\ cos\lambda_r \cos\mu_r & cos\lambda_r \sin\mu_r & sin\lambda_r \end{pmatrix} \begin{pmatrix} X_r^{"} \\ Y_r^{"} \\ Z_r^{"} \end{pmatrix}$$

- Calculate range, azimuth, and elevation

  $R = \sqrt{(E_r^2 + N_r^2 + V_r^2)}$

  $Az = \tan^{-1}(E_r/N_r)$

  $EL = \tan^{-1} \frac{V_r}{\sqrt{E_r^2+N_r^2}}$

Create an EOTS/Radar/Telemetry packet

The following steps are used in each timer event for the creation of each sensor packet.

Step-1: Read the position of a flight vehicle in ENU format

Step-2: Read the position of Launch point in Latitude-Longitude-Altitude

Step-3: Rotate local axes and align it with ECEF axes

Step-4: Translate to ECEF axes with earth's center as origin

Step-5: Read the position of respective sensor station in Latitude-Longitude-Altitude format

Step-6: Translate ECEF axes w.r.t. respective sensor station

Step-7: Rotate translated axes to align with ENU axes w.r.t. respective sensor station

Step-8: Convert ENU axes to spherical axes to obtain azimuth and elevation

Step-9: Create respective sensor packet as per specified format which includes parameters like azimuth, elevation, system time and validity, etc.

Send packets

The packets will be sent to 4 servers using a UDP socket which will be created using the configured Server IP and starting port number. The port of the socket for a station will be the starting port number added to the station's site ID.

Control of data transmission

Start/stop sending data: When data sending is started, packets containing tracking information calculated from the first entry onwards in nominal files are sent from the stations to the servers.

Pause/resume sending data: When data sending is resumed after pausing, packets being sent contain tracking information from the next entry onwards in the nominal file.

Display/control of nominal file used for the creation of packet

The part of the nominal file currently being used for the creation of packets will be displayed. The part of the nominal file used for the creation of a packet can be

changed if desired.

6.1.5 Testing

Socket creation and data transmission

A benchmark client program is created which receives data from all the sockets of the MSTDG. The packets received by the client from each socket are logged in a file to check whether data is properly transmitted from each socket. The results are found to be satisfactory.

Rate of transmission

The time at which the packets are sent by the MSTDG to the client program is logged in file for all the possible intervals i.e. 25 ms, 50 ms, 100 ms, and 250 ms. The time of sending packets is monitored for a duration of one hour.

- Test case-1: Packets sent at an interval of 25 ms The overall begin and end times are as expected. The time interval between the sending of successive packets is also as desired.
- Test case-2: Packets sent at an interval of 50 ms The overall begin and end times are as expected. The time interval between the sending of successive packets is also as desired.
- Test case-3: Packets sent at an interval of 100 ms The overall begin and end times are as expected. The time interval between the sending of successive packets is also as desired.

- Test case-4: Packets sent at an interval of 250 ms The overall begin and end times are as expected. The time interval between the sending of successive packets is also as desired.

Reading nominal files

The format of the nominal files is fixed (.txt or .dat). The MSTDG shows an error message when a file with an unsupported format is chosen. The trajectory nominal file must always be read, however, the body rates and chamber pressure nominal files may or may not be read. The results of the following test cases are observed as given below.

- Test case-1: File is chosen, the file is to be read The files are read and stored in a local array where each element of the array has a format that is expected for each row of the file. Data from this local array is written to another file. It is observed that no errors occur when trying to store the contents of the nominal file in the local array, and the contents of the nominal file and the file into which data is written from the local array are the same, which verifies that data is read correctly from the nominal files.

- Test case-2: File is chosen/not chosen, the file is not to be read The files are not read in this case and the graphs corresponding to those files are not drawn. The local array is created and initialized to zero value to prevent errors in case it is accessed.

- Test case-3: File is not chosen, the file is to be read An error message is displayed and transmission of packets cannot be started until the file is chosen and/or the file is specified to not be read.

Transmission of the packet of tracking sensors

Frame length of the packet sent by tracking sensors: Frame length specifies the number of bytes the packet sent will contain.

- Test case-1: Frame length is greater than or equal to the sum of the size of parameters to be sent To prevent overflow of the byte array used for sending the packet whose size is determined by the frame length, the total size of the parameters to be sent is calculated beforehand and compared to the frame length given as input. If it is greater than or equal to the total size of the parameters being sent, a byte array having size as frame length is created and sent through the socket. The number of bytes received is verified using the client software to be the same as the frame length.
- Test case-2: Frame length is less than the sum of the size of parameters to be sent An error message is displayed stating that an overflow will occur if this frame length is used to create the packet for that tracking sensor

Position coordinates of tracking sensors

Position coordinates of tracking sensors are given in Latitude-Longitude-Altitude format. If the Latitude or Longitude is beyond the possible range, an error message is displayed and the conversion of nominal data to packet for tracking sensor is not done.

Conversion formulae used for creating a packet of tracking sensor

The correctness of the conversion formulae for creating the packets is verified using a software developed in Matlab which contains the formulae for obtaining nominal data from the packet received. Various test packets corresponding to different input classes are sent to the Matlab software for testing. It is observed that the nominal data obtained is the same as that used for creating the packet by the MSTDG, which verifies that the formulae used for the creation of the packets are correct.

Graphical representation and visualization of data

Nominal files are plotted with a benchmark software developed in Matlab and visually compared. Scales of each of the graphs are checked. Satisfactory results are observed. The flight parameters displayed are calculated manually and the values are observed to match.

GUI controls

There are some pushbuttons, checkboxes and slider controls that are operated in real-time. The functionality of all the controls has been checked and satisfactory performance has been found. Checkboxes for control of data transmission were checked using the receive module of the benchmark client software. It is tested for each of the sensors and 4 servers. Expected results are observed.

6.1.6 Finding and Discussion

A multi-sensor trajectory simulator is capable of simulating three different types of tracking sensors. Electro-optical and Radar can simulate only the trajectory information. The telemetry system simulates trajectory information along with body rates and health parameters. Its capability of simulating total eighty number of sensors along with data transmission to four different Data processing server is tested and working smoothly with GUI controls in real-time. It's live modeling of EOS, Radar and telemetry tracking systems. Individual EOS, Radar and Telemetry simulator is available with the respective sensor. Our simulator output is compared with these existing simulators separately and result is satisfactory.

The MSTDG considers the tracking sensors as a black box, i.e. it calculates the tracking information that would be sent by them given the nominal data. It does not simulate the internal working of the sensors. The simulator sends packets of all tracking sensors at the same rate. However different sensors may have different rate of transmission. The MSTDG can be modified to send data of tracking sensors at different rates. The telemetry station sends a large number of health parameters and positional information to the server. At present, the MSTDG can simulate a limited number of those parameters.

6.2 Model-7: Real-time Multi Channel Remote Monitoring System

Overview

We have presented a system that transmits a series of screenshots over Ethernet in real-time using UDP Protocol. This system comprises of two parts. The first part is to capture screenshots then cropped into tiles and compare each of the tiles with previously captured screenshots corresponding tiles. only mismatched tiles are transmitted with a sequence number tagging to a client in the form of the UDP packet. It is called a Real-time Display Sender (RTDS). The second part is at the viewer's end, it receives screenshot cropped tiles and forms the full image by matching the sequence number of the cropped images. It receives a multicast or unicast packet as per availability automatically. It can receive packets from ten different RTDS simultaneously. GUI interface is available to choose any one of the streams by a single mouse click. This system we call as Display on Demand System (DOD). There may be as many as ten numbers of RTDS system that can be deployed for streaming ten different screens as video over Ethernet for DOD system. There is no limit to deploy the DOD system. We have tried with fifty numbers of DOD systems and ten numbers of RTDS systems running simultaneously in the Ethernet network and all systems functions smoothly. The proposed system here is designed based on multi-socket programming, threading and image processing. Here image processing is used to crop the image and make it into tiles and compared with previously captured images so that only mismatched tiles

are transmitted. Other end tiles are used to make full images thus it improves bandwidth utilization and efficiency of the whole system.

6.2.1 Introduction

Display on Demand application package is a client/server software package allowing Real-time data share to all Display on Demand client application over Ethernet that is sent by different-different Servers (Real-time display sender) applications. The Real-time display sender Server transmits data packets by taking screenshots of real-time display over Ethernet to selected IP and port no. It shows a Display setting panel to select or de-select Multicast and Unicast transmission. It also gives the facility to select or de-select the IP address to which data to be sent. The panel gives the facilities of adding, removing and updating IP, Port no, update rate of transmission and format of the screenshot image that is to be transmitted.

At the client end, Display on Demand system client application use a multi-channel panel in which each channel receives different data at real-time based on multi-threading through port number transmitted by the server GUI from different sites. If there is no reception of data, it shows a welcome screen with multicast IP and port no and Unicast IP and port no. The Display on Demand system client application shows ten channel displays that receive data simultaneously. The active channel is displayed by green and all available channels which are receiving data are displayed by yellow. The channels with no data are inactive and displayed by gray color. We can switch-over to other available channels by a single click to that channel. The default display which is shown to help men at the right side of the application is used to open that channel view always whenever we run the client

application.

To make this system more efficient in terms of bandwidth management image processing used. Here image captured in a predefined frame rate and cropped in K x K tiles and compared with previous captured corresponding cropped tiles. Finally, only mismatched tiles are transmitted. Tiles are split further to make the packet size less than the MTU limit of a router to overcome the delay caused by the routers for splitting, sequencing and rearranging the packets of length more than 1500 bytes.

Motivation and Objective

The maximum packet length of UDP is 64 KB. Communication switches and routers use default Maximum Transmission Unit (MTU) as 1500 bytes. Screenshot captured for full HD resolution in jpg or png image format is of the order of 150KB. Here we are cropping the image into K x K tiles so the size of each tile is less than the limit of UDP packet size but normally more than 1500 bytes. If packet size is more than 1500, the packet is split into two or more packets by the communication switch and routers as per the MTU limitation and it produces a delay in data transmission. Here we adopted a technique to limit that delay by making all packets below the MTU size.

Our main objective is to allow real-time data transmission using a display on demand application package. And our aim is to receive data from different servers to a single client without any interruption.

6.2.2 Real-time Display Sender (RTDS) System

Capturing the screen

Here java.awt.Robot class is used to capture pixels of the screen. It has a method called createScreenCapture which captures the current screen and returns as a BufferedImage object. To save in PNG image format ImageIO is used.

Reading and writing the image

Image reading and writing methods are defined in File class. This class represents file and directory path names. Error handling is done using the IOException class. The captured image stored in the BufferedImage object. This object is used to store an image in RAM. The captured image is read and the height and width of the image are found using the predefined functions.

Packet Management and Transmission

Packet management activity is carried out at both the transmitter and receiver switch level. The transmitter uses 1500 bytes packets as MTU (Maximum Transmission Unit) and splits in two or more packets if higher data bytes are found. This splitting procedure is done in a manner that switches uses. At the receiver side, the fragments along with the fragment table are received. The switch uses the fragment table to recombine fragments that are split are sender site switch.

The display on Demand server GUI application transmits packets after taking a snapshot of the display screen and preparing a packet of maximum 1500

bytes. It splits into two or more packets if it is found greater than 1470 bytes. Due to the wide area network, the old configuration switch is found at some remote locations. These switches can take more time in splitting and combining the packets. So 1470 bytes of data packets is used instead of a 64 kb data packet for a minimum delay during transmission. The packet is prepared and transmitted every second. The transmission of the package is done by the sender by creating a multicast and Unicast socket for connection for receiver application.

Image Cropping

Image cropping can be done in two ways. The first method is to crop the image based on the size(dimensions) means getting the sub-image from the original image can be done as follows. BufferedImage img1 = img.getSubimage(i, j, w, h); Here img1 is the first cropped part of the original image and img is the original image and i, j are the co-ordinates of one end of the image and w,h are the width and height of the image required to be cropped.

Image Comparison

Two images can be compared only if they have the same dimensions. If the dimensions match then we need to find the RGB values and then the difference in two corresponding pixels of three color components. The procedure is repeated for each pixel. Now the percentage can be calculated by dividing the sum of differences with the number of pixels, to obtain the average difference per pixel. If the percentage difference is not zero then two images are different.

Cropping, comparing and transferring the image – Server-side

The screenshot is split into K x K tiles and keeps a copy of the tiles in a Buffered-Image[] buff. Compares tiles of each new frame with its respective tile from buff[] and updates the buff[] content if its different and passes it to send function. Send function encodes the image passed to it and essential parameters for decoding into packets and sends through UDP sockets(Both Unicast and Multicast). This function calculates the required tiles to be sent compulsorily to the receiving side to recover missing tiles and takes those tiles from buff[] and passes them to send(img, b, k).

Java DatagramSocket class is used for sending and receiving datagram packets. In this case, we are sending multiple packets so it may arrive in any order. Those cropped parts are given numbers for purpose of receiving in the same order and then with some time delay next screenshot is captured and cropped and each cropped image is compared with the previous cropped part and if the percentage difference is non zero then that image is sent through the socket from server-side. For sending a screen shot server can be decided using the IP address of the server and the port number used.

Image Transmission

Data transmission is initialized to selected IP after clicking on the START button on the server setting panel. public void sendImage():- This function split the screen contents into different datagram packet and send them to the receiver. Using Java Datagram Socket class a connection-less socket object is created for sending and receiving datagram packets.

Fragmentation

Fragmentation is the process of breaking up a single packet into multiple packets of smaller size. The fragmentation is done at Real-time Display Sender to minimize time delay during packet transmission.

The configuration of switches uses a default packet size of 1500 bytes as the Maximum Transmission Unit (MTU) for minimum delay. The Real-time Display Sender uses UDP packets which takes a maximum size of 64 KB for transmission. For minimum delay of packets, the Real-time Display Sender uses a fragmentation technique to fragment UDP packets into 1472 bytes packets and takes 2 bytes for storing sequence no of packets. Another 2 bytes are used to store no of packets generated. The generated packets along with sequence number and number of packets are sent to Display on Demand Systems.

Configuration

The Design view of Real-time Display Sender GUI has three panels. The first panel displays the activation of Multicast and Unicast transmission. It also keeps a start button for the initialization of the transmission of data packets. The second panel shows a list of IP, port number and remarks status along with checkbox for active selection. We can add, delete and update IP, port and remarks section.

This panel uses six columns in which checkbox, IP, Port number, remarks, add button and delete button are displayed. If the checkbox is clicked, the IP and port number in the row is selected for data transmission. Add and Delete buttons are used for removing particular IP and port from when not necessary. The third panel displays the update rate of transmission and format of the snapshot as png,

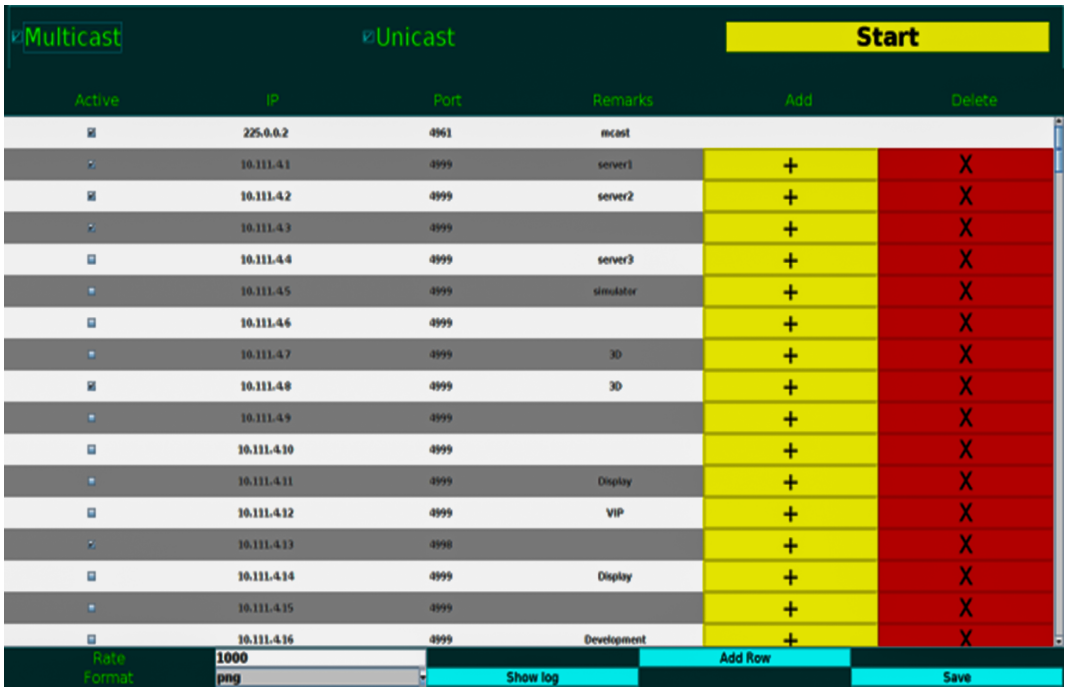

Figure 6.12: Real-time Display Sender Interface

jpg, pngencoder, etc.

It has three buttons named show log, Add row and save button. The show log function displays log file data after transmission. The Add row button adds a Row after the last Row of the file and updates the Interface of RTDS. The save button is used to save the data after editing any details of IP, Port, and Remarks.

6.2.3 Display on Demand (DoD) System

Receiving and Displaying the image – Client-side

The display on demand system GUI is developed in java. It uses independent threads for all its ten-channel architecture. Every channel uses Multicast and Uni-

Figure 6.13: Display on Demand interface

cast sockets for data reception.

The Unicast priority is higher than Multicast. The receiver application work in a loop to check either multicast or Unicast is present or not. If anyone is present, it shows the data to a specific channel on display. If both are present, it prefers Unicast socket data to display on screen rather than the multicast. The transmitted packet searches distinct IP in a wide area network and enters into the network through port no matching. The packet is delivered to the receiver application after matching port no. In this way, the packet is delivered.

The images sent on the server-side are received on the client-side. If a data packet is being received by Display on the Demand system, it collects the data into the buffer as soon as the image is received that image is displayed on the JFrame window. The only images that differed from the previous image are received.

Image is received in the form of packets and that packet is then converted into byte array and displayed using the repaint method. The image is taken in the byte array format and is read as a bufferedimage.

If c is the variable on the client side the c.repaint displays the image received using the draw canvas method with some time delay. All the coding part is done in java using the Netbeans IDE. For GUI, Swing widget toolkit is used.

Recombination

The Recombination of packets at Display on Demand system is done after the arrival of the data packet along with sequence number and number of packets. The Display on Demand system arranges the packets in sequence and recombines it to prepare a final packet.

Configuration

The Design view of the Display on Demand system has three panels. The first panel is used to show ten-channel architecture that uses buttons to switch from one channel to another. This panel shows a help menu also with ten-channel buttons. The default display button color is green as the active display. The other available channels show yellow if they are receiving data simultaneously. The Help menu in the first panel is located on the right side of the display. The second panel of the DOD system shows a welcome screen with active IPs and port numbers and it will be transformed into the real-time display when the data packet is received. It takes a large size of the screen, as it shows the real-time view of the server screen. The third panel shows the Help Section of different settings.

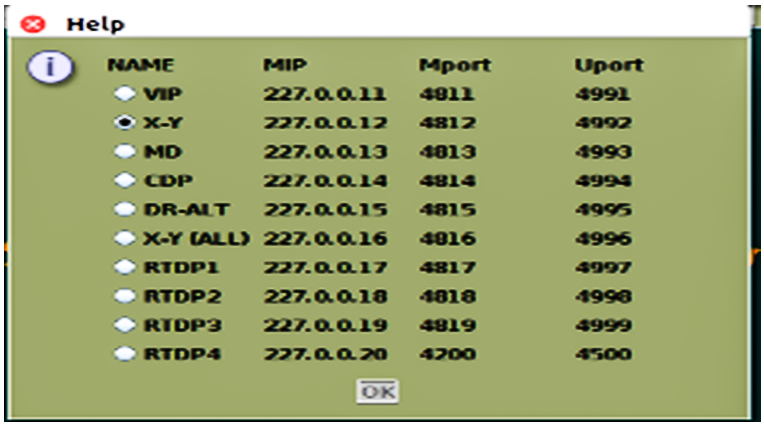

Figure 6.14: Help menu for selecting default display

6.2.4 Implementation

Image Processing

An image can be an array or a matrix of square pixels that are arranged in row and columns. Images can be stored as 2D pixel array in memory and each pixel value controls the color. Different formats of image are .jpg, .bmp, .tiff, .png , .gif, .jpeg, etc.

Image processing means performing some operations on the images to extract some useful information. Image processing basically includes the following three steps.

The first stage, Importing the image with any capturing technique.

In the second stage, the input image is analyzed and manipulated. It includes data compression, image enhancement or spotting patterns.

In the last stage, the output can be altered image or report that is based on image

analysis.

Socket Programming

Socket programming is used for transferring packets over Ethernet. As it is a real-time data transfer, the UDP protocol is followed. While UDP socket can transfer images up to 64 kb. So to reduce the size of the image we will crop the images and then transfer through the socket. We are implementing socket programming including images in the java programming language.

Implementation of Image Processing Techniques

- The Current screen of the Sender (in my case Server) is captured using Robot().
- The image is then divided into K x K tiles where K is an even integer less than 20 and all the tiles are transferred for the first time when the server started running.
- A copy of the transferred tile is stored in a BufferedImage array for future comparisons.
- For the next screenshot onwards the tiles are compared with the respective reference tiles and only the different tiles are sent.
- While sending the image through socket image is converted into ByteArrayOutputStream.

Length of the byte array is found and then another new byte array is created with the length one more than the previous array in the created new array the first in-

dexed element is the number that is the identification for the image while receiving. That new byte array with number is now converted into packets and transformed. Those packets are sent through the socket and received on the client-side. While receiving a new client-side variable is created and a new datagram packet is created for receiving and that received is packet is again converted to byte array form and saved.

Reconstructing the Image

As each image is received it is joined with all the images received and each image is placed in its particular place. Joining is done in the following way. Each image height and width are found and the next image height and width are found.

Then those two images are joined and it is repeated for each cropped image. Each received image is joined to the previous joined image and after receiving the last part and joining it the original image is formed.

The reassembly of packets is performed by switch at the receiver end. The fragment table maintains information about already received fragments of the packet. Each packet is uniquely identified by the source IP, Destination IP, and a unique identification number. The fragment table is used to find all the fragments and reassembled to a packet.

Displaying the image

Image display can be done in two ways. The first method is to display the image after receiving all the parts and joining the parts then the final image is displayed. The second method is to display the image as soon as it is received and each

image is displayed in a similar way with some time delay. For drawing the image on JFrame Drawcanvas is used. Repaint is used to draw the image and d is the canvas variable and f is the JFrame variable and 1366,768 is the size of the JFrame displaying. repaint() function is called after all image parts are joined.

6.2.5 Testing

Here system testing is done to test the overall performance of the software and evaluated the system's compliance with all specified requirements.

- Test case-1: Only unicast transmitted from one sender and received by one receiver
  - Data packet transmitted successfully.
  - Data packet received successfully.
  - Image displayed successfully.
- Test case-2: Unicast and multicast transmitted from one sender and received by one receiver
  - Data packet transmitted successfully.
  - Data packet received successfully
  - Image displayed successfully.
- Test case-3: The receiver starts running after sender
  - Data packet transmitted successfully.
  - Data packet received successfully

  - Image rearranged displayed successfully.

- Test case-4: Only unicast transmitted from one sender and received by 5 receivers
  - Data packet transmitted successfully.
  - Data packet received successfully
  - Image displayed successfully.
- Test case-5: Unicast and multicast transmitted from 10 senders and received by 10 receivers
  - Data packet transmitted successfully.
  - Data packet received successfully
  - Image displayed successfully.
  - Data available status correct.
- Test case-6: Ideal Delay Testing to reduce Network traffic as well as reducing packet loss
  - Data packet transmitted and received successfully without loss.
  - Minimal Compromise with Frames per second.
  - Data rate is improved.

6.2.6 Finding and Discussion

It is a very challenging task to provide multiple displays on a single screen. This project was aimed at building an effective system to monitor real-time displays

that are deployed at remote locations with an acceptable frame rate and HD resolution. It handles all the exceptions and provides easy to use reliable GUI so that any novice person should be able to view the desired display and switch over of his choice in real-time.

As of now after completion coding and all testing system is running smoothly. It is streaming HD resolution with an average frame rate of 7 fps using the average data rate of 0.6 Mbps. This project is still open to changes though it can be modified in various ways and a lot of other things can be added and improved but for now it can provide Real-time Display on Demand with ten different views in locally connected systems.

Here, tiles are used as the means for comparing and sending, but a tile contains more than 6000 pixels from which if a single pixel is changed although it looks similar to the human eye still the entire tile is sent. So, for future work Machine learning and fragmentation techniques can be applied to resolve the issue and get a better promising frame-rate along with a steep improvement in data rate.