Time-Lapse Image Acquisition

(Exercise #6 – Real-Time Software Systems)

ECEN – 5623: Real Time Embedded System - Summer 2016

- **Group Members (Individual):**
  Gayatri Mestry

- **Breakdown of the Report:**

1. **Introduction**
   1.1 Objective
   1.2 Components Used
   1.3 Description
   1.4 Implementation
   1.5 Analysis
2. **Functional (capability) requirements for system design**
3. **High level system and software design**
   3.1 Block Diagram
4. **Real-Time Services and Requirements**
   4.1 Services and descriptions
   4.2 Ci and WCET specification
   4.3 Ti and Di specification
5. **Real-Time Services Feasibility, Safety and Margin**
   5.1 Cheddar Worst-Case and Simulation
   5.2 Scheduling Point / Completion tests
6. **Time-stamp tracing**
7. **Submission details**
8. **Conclusion**
9. **References**

1. Introduction

1.1 Objective:

The time lapse analysis aims at capturing images at a lower frequency of 1Hz. The images are then compressed and send over Ethernet to store them. A video is then created for the images captured. Several experiments are performed to test the time lapse analysis.

1.2 Components used:

2. Capturing images – The image capturing is basically done using the V4L2 (Video for Linux)library for acquiring the images in a .ppm format.
   The pixels are processed to be converting from YUYV to RGBRGB format for this project.
3. Compressing images – The OpenCV library is used for compressing images and the functions for reading a Mat image file and writing to other are mostly utilized.
4. Sending images over Ethernet- The images are sent over the Ethernet using UDP sockets and sendto() and recvfrom() functions. Their role in the project is described below.
5. Creating Video – The ffmpeg library is used to create the required time lapse video.

1.3 Description:

- The time lapse analysis begins with capturing images from a Webcam C200 at a specified frequency using V4L2 library on Jetson TK1.
- The IO_METHOD_MMAP and xioctl functionality of V4L2 is used to get the frames. The captured images are then stored as .ppm files.
- Since these files are larger in size, they are compressed to a .jpg size which is around 30-40% the size of the original .ppm images.
- These .jpg images are then streamed over the Ethernet using UDP sockets.
- The .jpg file is first opened, then the file size is acquired and sent over the socket first.
- Then the file is read and sent to other machine using sendto() function.
- The receiver side first receives the file size using UDP packets. Then a new file is created on the receiver side. Whose contents are received from the sender using recvfrom() function and are written to the file immediately.
- The images thus captured are compiled in the form of a video using ffmpeg

1.4 Implementation:

- The main process first completes the function of initializing the device and starting the capture process.
- It then spawns two threads, one for capturing the images and the other for compressing the images using OpenCV's imread() and imwrite() and sending them over the Ethernet.
- Semaphores are used in the producer-consumer fashion to align and interleave the functionality of the two threads.
- Also client socket is created in the main function that can connect to server running on any Linux machine using UDP protocol.
- The send, receive, read and write of the bytes between the two programs is carried out in a loop.
- The ffmpeg library thus compiles the images by running the command – ffmpeg –i img%08d.png –c:v libx64 –r -30 –pixfmt yuv420 out.mp4

1.5 Analysis:

The primary analysis like jitter and latency analysis is done using the timestamps of the images and the timestamps obtained by the execution of the threads. The deadlines met or missed are also checked.

2. Functional (capability) requirements for system design:

The skeleton code is acquired from the course website and has been reworked to accomplish the desired results:-
http://mercury.pr.erau.edu/~siewerts/cs415/code/computer-vision/simple-capture/capture.c

The main requirements and goals implemented in the project as described in the page - http://ecee.colorado.edu/~ecen5623/ecen/labs/Linux/extended_lab_requirements.html

***R#1**: Resolution must be at least VGA (640x480) or Standard Definition TV (720x480 or 640x480)*

- The resolution of the image captured is fixed by setting the macros HRES to 640 and VRES to 480.
- The value of these macros is given to the pixel width and height value in the struct V4L2_format.

```
fmt.fmt.pix.width       = HRES;
fmt.fmt.pix.height      = VRES;
```

***R#2**: You must acquire INDIVIDUAL frames from your camera - NOT an MPEG encoded stream. The frames must be acquired at 1 Hz for verification purposes.*

- The code enables the Webcam to capture individual frames at a rate of 1Hz i.e. capture at a speed of 1 frame per second.
- The main functions, calls the init_device(), where the xioctl sets the required capabilities by accessing the device and setting the video input, standard, cropping and other low levl tasks.
- It then calls the start_capturing() function where a buffer is allocated and the video is captured, stored and enqueued for further processing.
- After this the frame is set to be captured at a rate of 1Hz using the struct timespec read_delay.

***R#3**: Image file format must be PPM "P3" or "P6" RGB format as documented for Netpbm format*

- The dump_ppm() function is used to create a .ppm image file and the contents in the buffer are written to it.
- The PPM header also contains timestamps and the hostname ‘uname –a’ output.

***R#4**: Image Acquisition verification using EXTERNAL WALL CLOCK*

- The code changes made is thus verified by recording the passing of time on a watch for 30 minutes.
- To overcome or get a work around for the camera initializing jitter, 1808 images are being captured and the images from number 8 to number 1808 are used for analysis purposes.
- This is done to because the first 7 images captured are dark and blurry and cannot be verified for their authenticity.

***R#5**: Observing a physical process like the melting of a cube of ICE as the second experiment.*

- The process of melting of ICE is also recorded for 30 minutes and the corresponding video is created using ffmpeg.

***R#6**: Compression of frames on your target so you can store more than 2000 frames and so that you have less to transfer over Ethernet.*

- The images are compressed using OpenCV’s library function imread() and imwrite() and are converted into .jpg image which are considerable smaller in size than the .ppm images.

***R#7**: Continuous download of frames over Ethernet*

- The .ppm images are compressed and sent over to the server running on a different machine using UDP socket functions.
- This functionality of compression of images and sending them takes place in the Compress thread.

***R#8**: Run at a higher frame rate of 10 Hz with continuous download*

- The process of capturing images, compressing them and sending them at the same time is also repeated at a higher frame rate of 10Hz.
- The passing of time on an analog clock is recorded and the video encoded.

3. High level system and software design:

3.1 Block Diagram

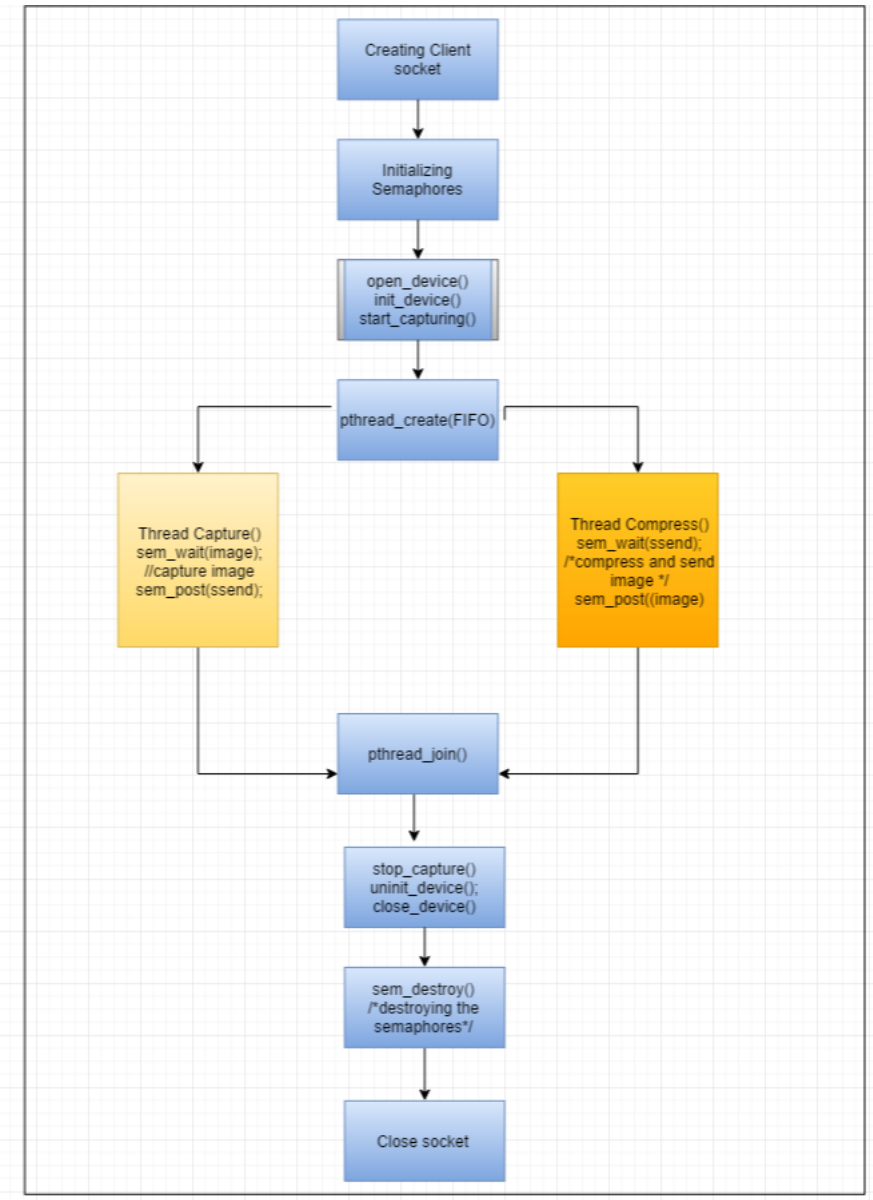

Fig 3.1 The Big picture

- The diagram gives the high level overview of the progress of the program and the flow of the main function.
- The socket created in the main function is the client socket for which only the socket() function is called to create it.
- Two semaphores are created, image and ssend and they are used in the producer consumer locking pattern so that the two threads used in the program get a sequence for their execution.

- Two threads are spawned from the main program thread_capture() and thread_compress() and are given core affinity so that they execute on the same core.
- The thread_capture() is the thread that does the functionality of capturing the image, processing it and creating the .ppm images.
- The thread_compress() does the functionality of compressing the image using OpenCV methods and sending the .jpg image over the client socket created.
- The functionality of the two threads take place as below.

Fig 3.2 Thread functionality

- The interleaving of the threads take place in a sequence which takes place because of the semaphores.
- The image semaphore is initialized to 1 and the ssend semaphore is initialized to 0.
- The way the semaphores are used is shown below:

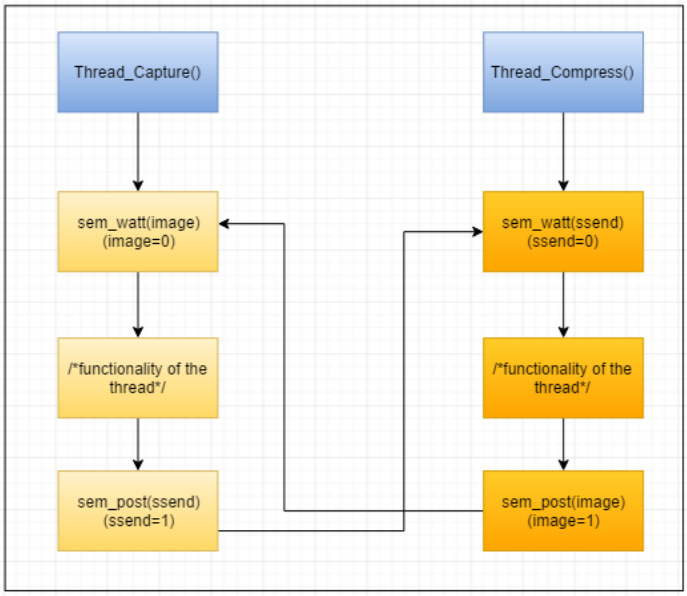

Fig 3.2 Interleaving of threads using semaphores

4. Real-Time Services and Requirements

4.1 Services and descriptions

- There are two services in the program which are the two threads.
- Since in the first part of the requirements, each thread should arrive at a frequency of 1Hz, both the threads have their execution time within that particular second.
- The execution time of the threads is recorded by taking the time difference of the struct timespec structure from the beginning and the end of the functionality of the threads and the log file is recorded.

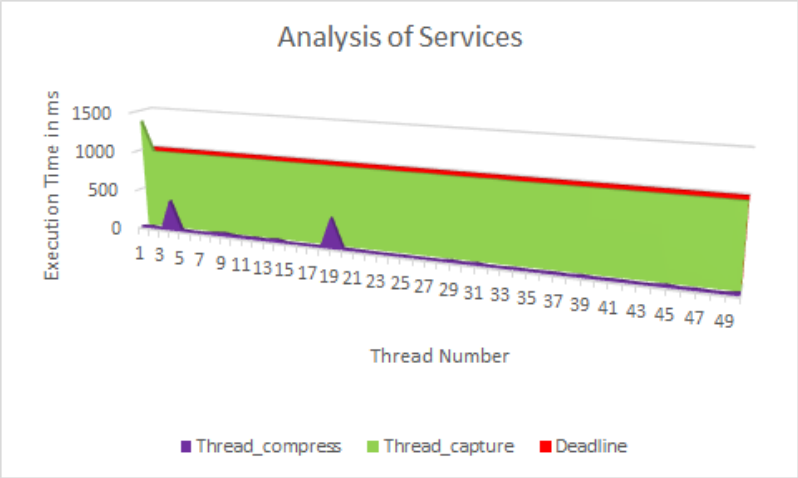

Fig 4.1 Analysis of Services

4.2 Ci and WCET specification

- The Execution times of the processes are recorded and plotted to check their deadlines.

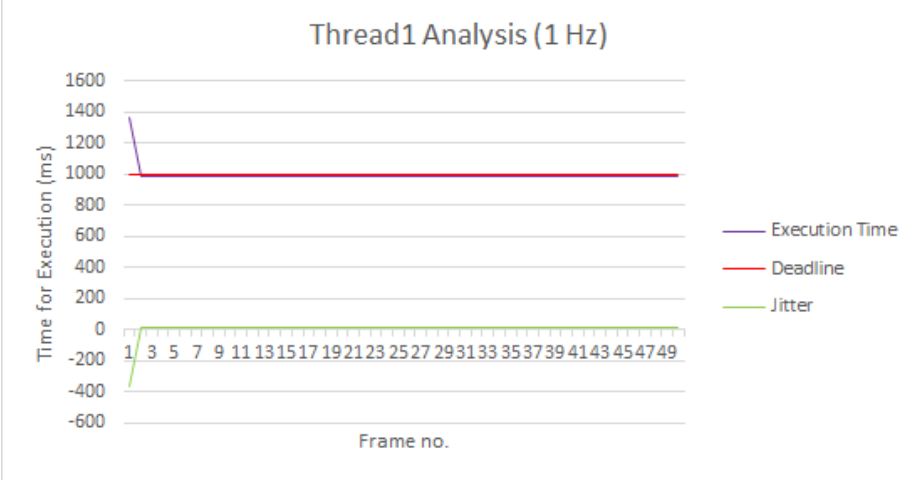

Fig 4.2 Thread 1 (Thread_Capture) jitter analysis

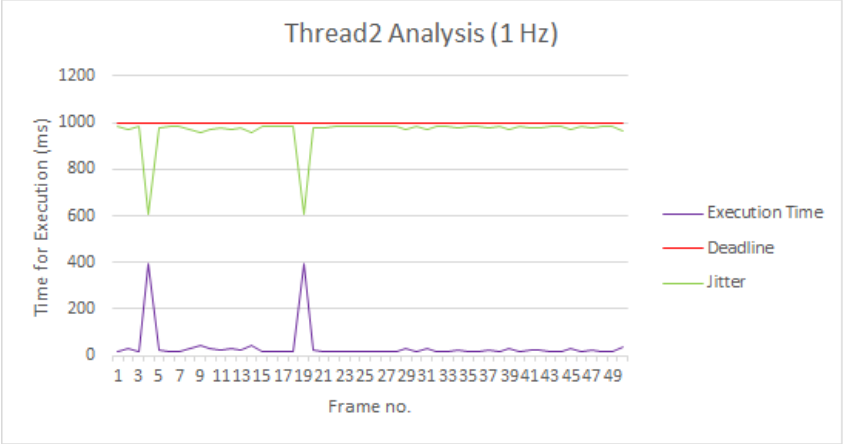

Fig 4.3 Thread 2 (Thread_Compress) jitter analysis

- The Thread analysis for Thread1 and Thread2 with a deadline for 1Hz indicates that both the threads have successfully maintained a positive jitter and that they have individually never crossed the deadline mark.

- The analysis for the two threads that run at 10Hz is also done and the Ci for Thread1 i.e. Thread_Capture() is observed to always miss the deadline of 100ms, thus giving a negative Jitter.
- This shows that the threads have missed their respective deadlines when running at a higher frequency of 10Hz.
- The Thread2 i.e the Thread_Compress thread meets its deadline and has a positive Jitter.

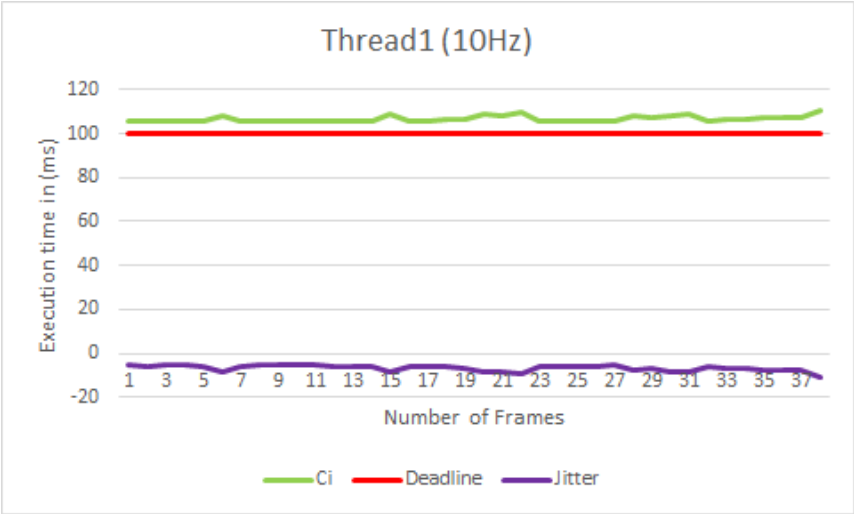

Fig 4.4 Thread 1 (Thread_Capture) jitter analysis

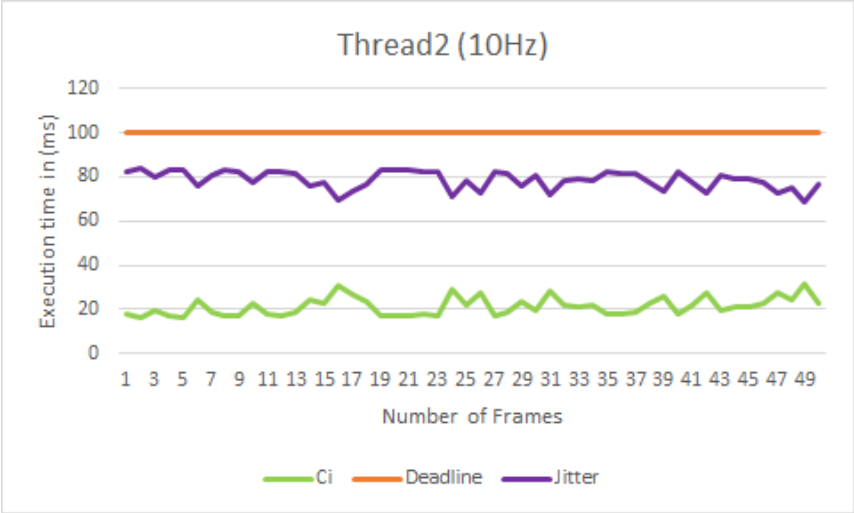

Fig 4.5 Thread 2 (Thread_Compress) jitter analysis

4.2.2 Worst Case Execution Time Analysis

- The worst case execution time of thread1 i.e. Thread_capture() for 50 frames at 1Hz is 985.18ms,and the graph plotted shows that the execution time of all other frames have values very near to the worst case execution time.
- This proves that the thread1 has better execution time and does not miss deadlines.

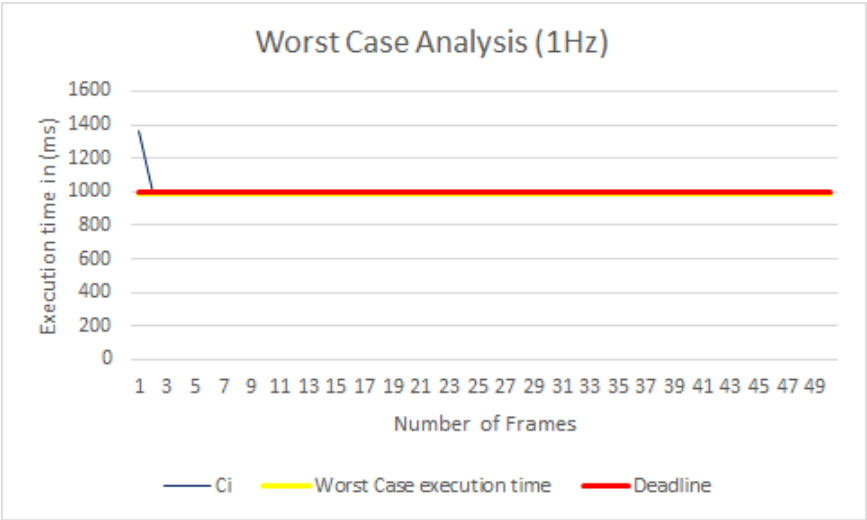

Fig 4.6 WCET analysis

- The Worst case execution time for 50 frames captured is 110.72ms and the execution time of the other threads is compared to deadline.
- The graph plotted shows the Ci of the Thread_Capture() along with the deadline and the longest Ci among the 50 threads.
- It can be thus analyzed that the thread always misses the deadline and the Ci of most of the threads is closer to the worst case execution time observed.

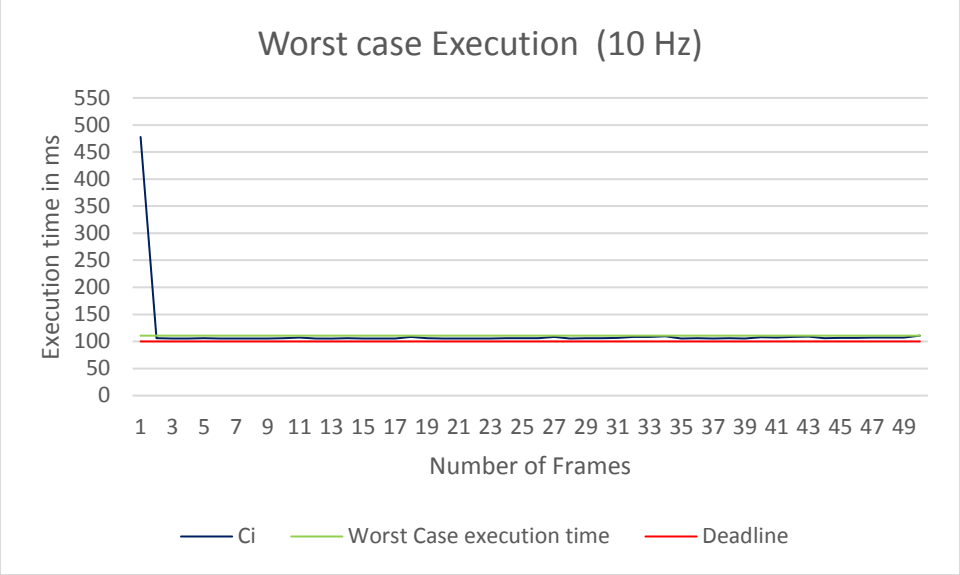

Fig 4.7 WCET analysis

4.3 Ti and Di Analysis

- The graph below shows the comparison between the summations of execution times required by the two thread versus the Deadline set for the project (1Hz).
- The time period and the deadlines is equal for this experiment.
- The summation of the execution times of the two services does miss the deadline in some scenarios but the overall functionality is intact as the services individually do not miss any deadline.

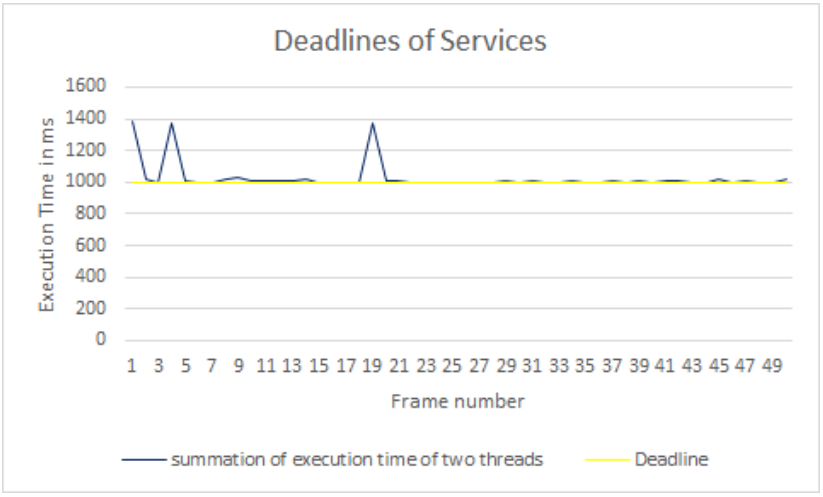

Fig 4.8 Deadline analysis

- The deadline for the threads running at 10Hz is always missed as seen in the graph below.
- Thus the summation of the execution time for both the threads is longer than the deadline set for the project.
- This proves that there is negative jitter and that the thread execution time cannot meet the required deadline at 10Hz.

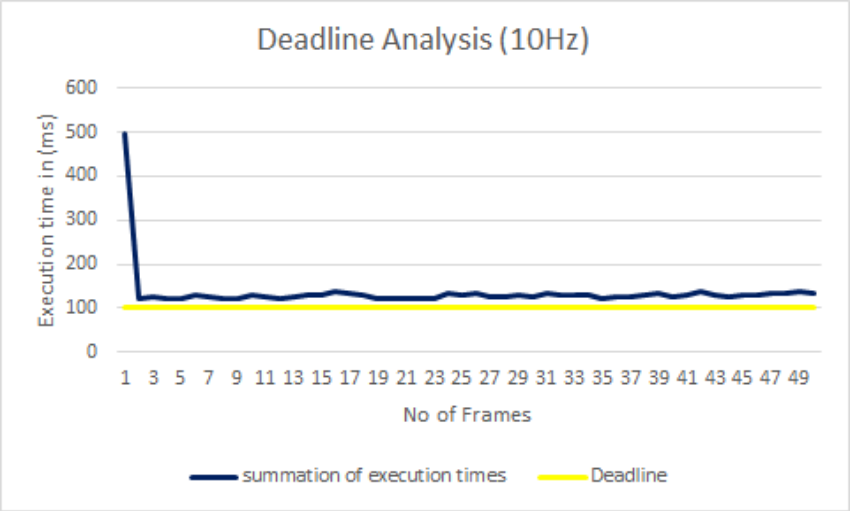

Fig 4.9 Deadline analysis

5. Real-Time Services Feasibility, Safety and Margin

5.1 Cheddar Simulation

- The Earliest deadline first scheduling algorithm was used to simulate the execution of the two threads in Cheddar.
- The Deadline Monotonic Analysis could not be used in this case as the period of the services is equal to their deadline, which results in an analysis similar to Rate Monotonic.
- Earliest Deadline Monotonic was used as the threads are run on a single core and earliest deadline first algorithm gives better analysis for the threads having a single core affinity.
- The threads are pre-emptible but are constrained to a desired sequence with the use of semaphores and thus are interleaved to get desired results.
- Also the utilization during EDF is maximum when the Ti and Di of the tasks are the same.
- The Cheddar Simulation for 1Hz frequency for frame 3 shows the proper simulation output in Cheddar and that the deadlines are not missed

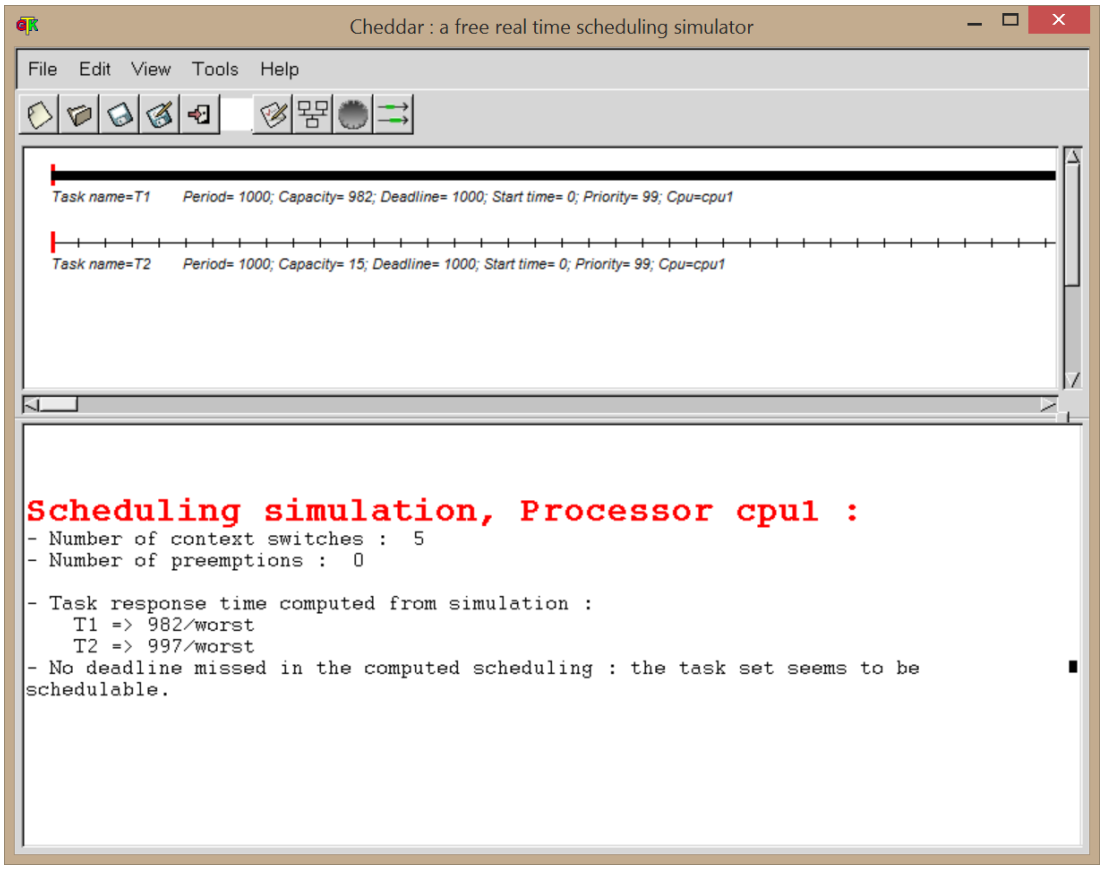

Fig 5.1 Cheddar simulation and analysis

5.1.1 Worst Case Execution

- The worst case execution time for Thread1 and Thread2 of deadline 1Hz were used to simulate the worst case analysis in Cheddar.
- Considering the Worst case Analysis shows that the deadlines are not met in any of the iterations of the frame capture, compress and send functionality.

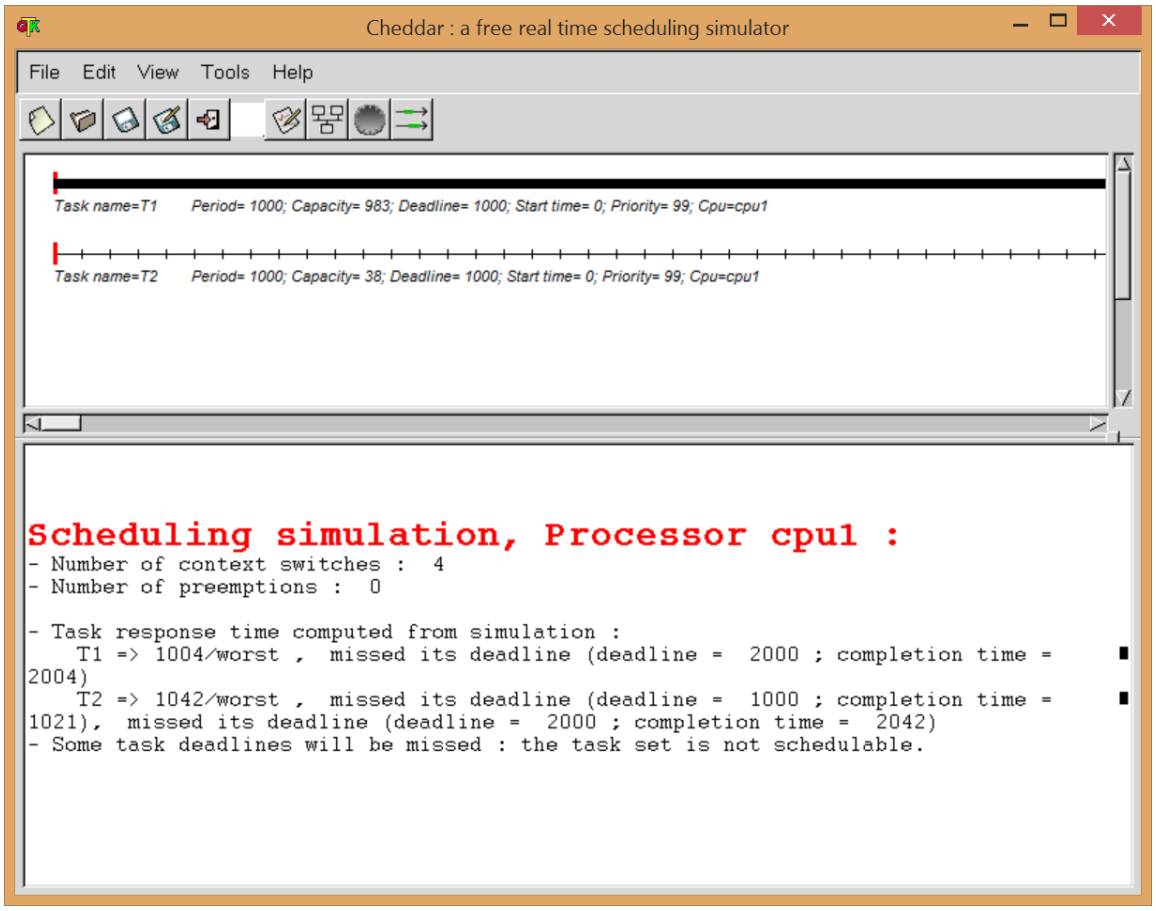

Fig 5.1 Cheddar worst case analysis

5.2 Scheduling Point / Completion tests

- The two threads are schedulable as they complete their functionality in normal execution times.
- Its only during the Worst case execution time the threads miss their deadlines.
- Overall the execution of threads in a loop of getting 1800 frames is successful and we get the frames correctly on the client as well as on the server side simultaneously with very little latency and jitter.

6. Proof-of-concept code and time-stamp tracing

- The images below show the PPM header output for the first and the last of the 1800 frames captured for a period of 30 minutes with 1Hz frequency.

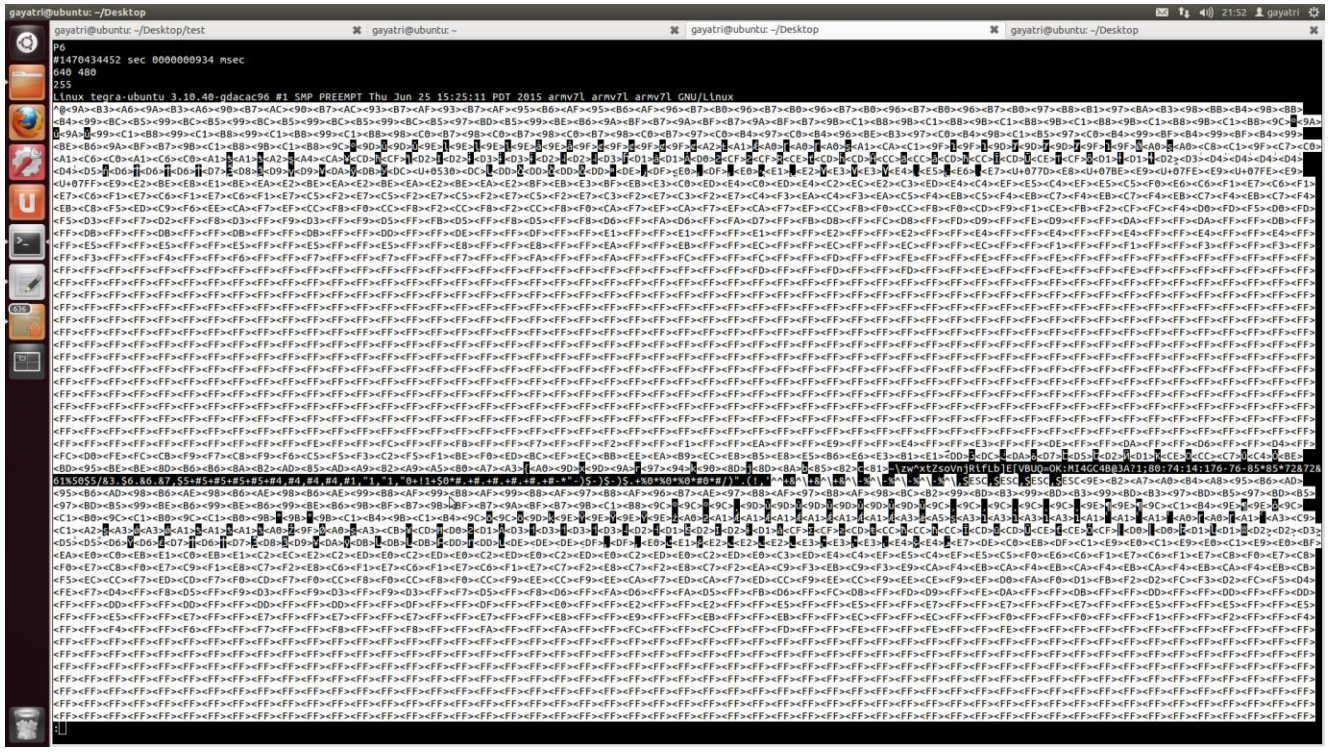

Fig 6.1 Time trace of PPM image 1

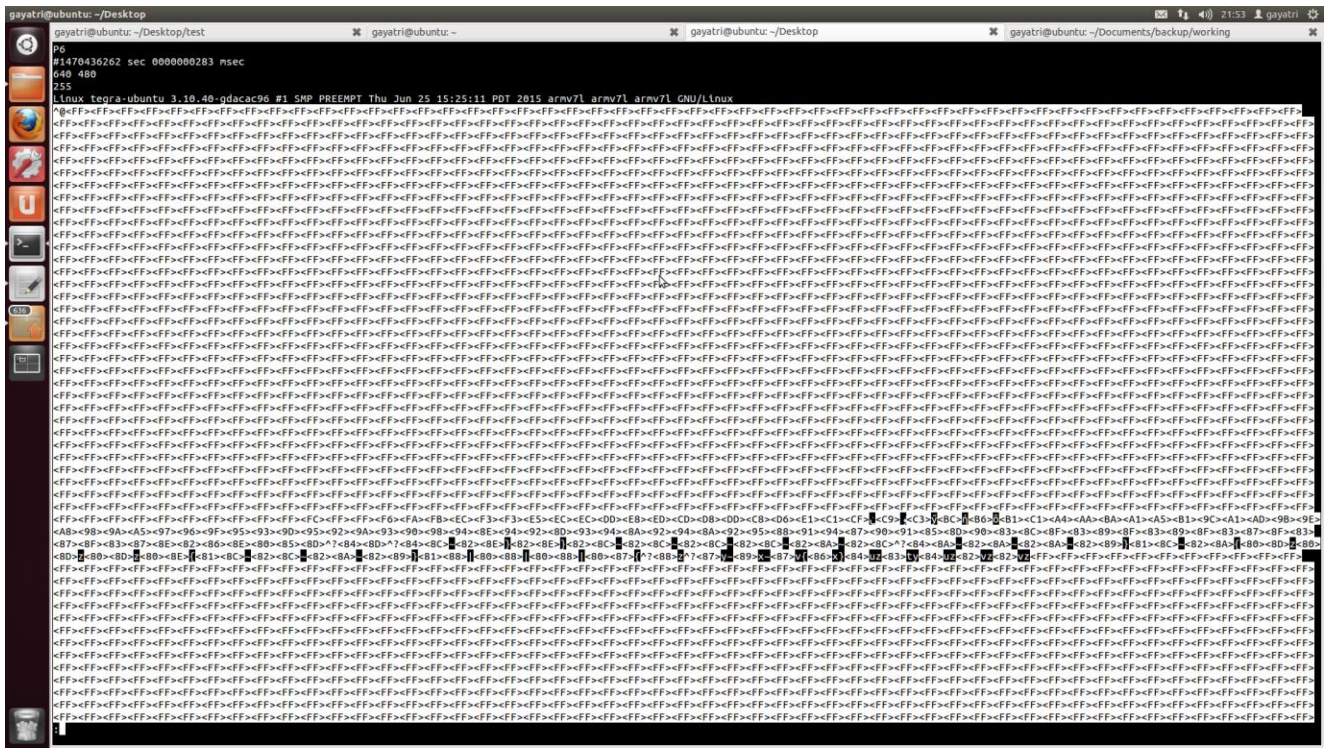

Fig 6.2 Time trace of PPM image 1800

7. Submission details

The project submission directory has the following directory orientation

1. Code_1hz folder contains the
   1.1 Client directory which has the code for capture and the PPM images and the video created named out.mp4
   1.2 Server directory that has the server code and the images acquired after receiving from the Ethernet.
2. Ice folder contains the output video after compiling the output from the images captured during the melting of Ice.
3. Freq_10hz folder contains the output ffmpeg video obtained after capturing frames at 10Hz.

8. Conclusion

The time lapse analysis gives an insight into the synchronization of two threads and analyzing whether the threads meet their deadlines in their iterations or not. The time lapse acquisition at 1Hz gives favorable results in terms of jitter and latency but the analysis of frame capture at 10Hz shows that the jitter increases with the increase in frequency.

9. References

[1] https://arxiv.org/ftp/arxiv/papers/1101/1101.0056.pdf
[2] https://github.com/alexanderkoumis/toy-opencv-mat-socket-server/blob/master/src/socket_client.cpp
[3] http://stackoverflow.com/questions/21738789/c-opencv-image-handling-after-get-it-through-sockets
[4] http://stackoverflow.com/questions/749964/sending-and-receiving-an-image-over-sockets-with-c-sharp
[5] Beej's Guide to Network Programming
[6] http://mercury.pr.erau.edu/~siewerts/cs415/code/computer-vision/