Simulating Multimedia IOT

Arvind Vishnubhatla

***Abstract:** While working with IOT projects with camera sensor ,the cost of hardware becomes prohibitive as it is not possible to duplicate the hardware for a large number of sensors. Here we need to have lots of multimedia sensors which send image data. The image data so generated is to be compressed at the client node suitably using some transform. To make a cost effective client we simulate it as a client process in linux. .Here a random number generator is used to generate an image sample. In this implementation we use a wavelet transform to compress the generated image. This is then sent over a socket to the server process. On the server side a stream socket receives the image data which is then suitably decompressed. Here a broad range of image processing algorithms are applied to enhance the image, threshold and segment it. Here different morphological operations are applied to isolate the region of interest. There could be multiple clients generated in several windows. This way a cost effective client and server is implemented to simulate the IOT architecture.*

***Keywords:** Simulate, Multimedia , IOT, Linux , Client ,Server*

I. INTRODUCTION

An IOT contains a set of interconnected sensors attached to an embedded board used for industrial applications like surveillance, automation etc[1]. A sensor detects changes in ambient conditions and converts them to electrical signals. A node contains the power supply, processor unit, sensing unit and transceiver [2]. Multimedia sensors require larger bandwidth and consumes high battery power. There are issues related to prioritization, mobility management and security. One of the methods is to process the image at the camera node. This saves the data to be sent to the remote server. However the power consumption at the camera node is high [3]. To alleviate this feature extractions are done at edge node. In this paper network congestions due to multi-hop transmissions have not been modelled.

A distributed architecture where workloads are simulated and given to a server is envisaged. To save space both client and server reside on the same machine. The server awaits incoming requests from clients and suitably processes them[4].The total number of processes is limited. As this is pnly used to demonstrate proof of concept, the denial of service attack is not considered.

Revised Manuscript Received on November 30, 2019.

* Correspondence Author
Arvind Vishnubhatla*,ECE,Gokaraju Rangaraju Institute of engineering and technology,Hyderabad,India,vainfo66@gmail.com

While working with IOT projects with camera sensor, the cost of hardware becomes prohibitive as it is not possible to duplicate the hardware for a large number of sensors [5]. Here we need to have lots of multimedia sensors which send image data. The image data so generated is to be compressed at the client node suitably using some transform. To make a cost effective client we simulate it as a client process in linux. Here a random number generator is used to generate an image sample. In this implementation we use a wavelet transform to compress the generated image. This is then sent over a socket to the server process[6].

On the server side a stream socket receives the image data which is then suitably decompressed[7].Here a broad range of image processing algorithms can be applied to enhance the image ,threshold and segment it .Here different morphological operations can be applied to isolate the region of interest[8].There could be multiple clients generated in several windows .This way a cost effective client and server is implemented to simulate the IOT architecture.

II. PROJECT METHODOLOGY

A. Block diagram

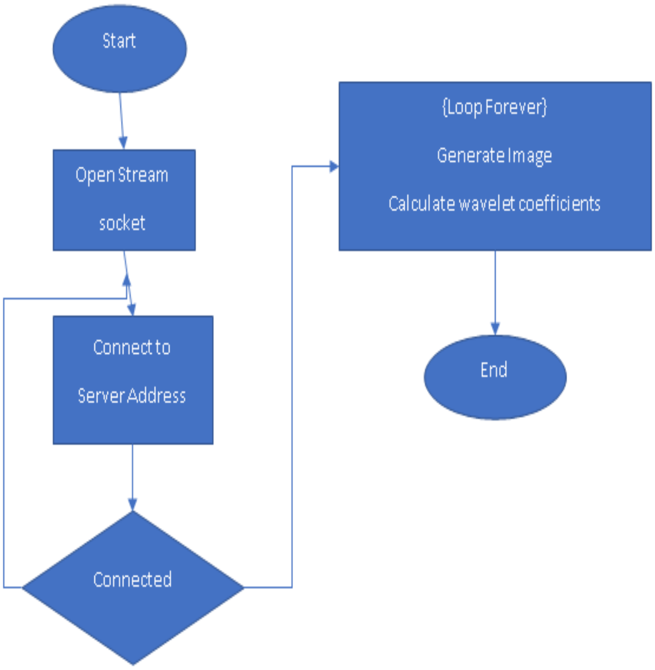


Figure 1 The client program flow in Linux

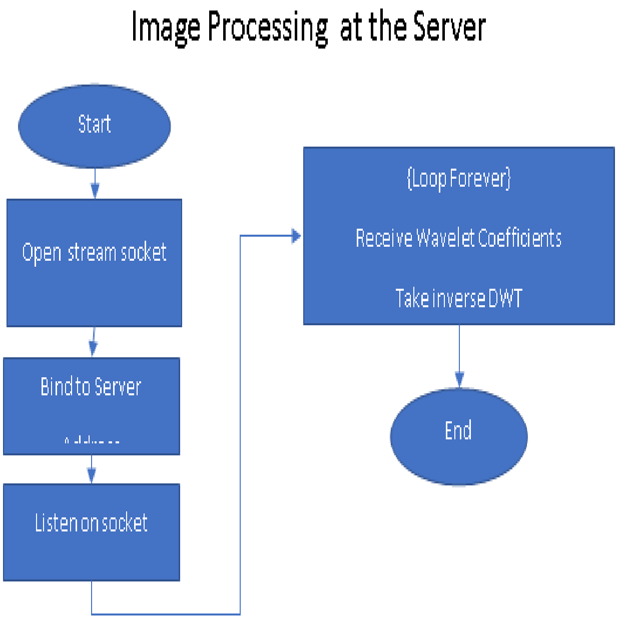


Figure 2 The Server program flow in Linux

B. Algorithm

Two C programs tcpClient.c and tcpServer.c are written in Linux.

For compilation we use the commands gcc -c tcpClient.c tcpServer.c to get object files.

For linking we use commands gcc -o client tcpClient.o gcc -o server tcpServer.o to get executables **client** and **server** in bold. In one of the windows the server is first run as ./server. Now several clients are executed from different linux terminals using ./client.. Log files are used to see if feature extraction has been properly made and image processing runs are successful.

C. Flowchart

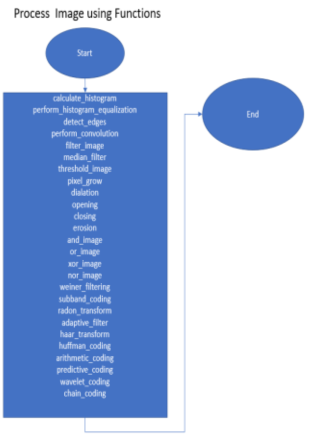


Figure 3 Process the image using Functions

Figure 4 DWT 2D transform

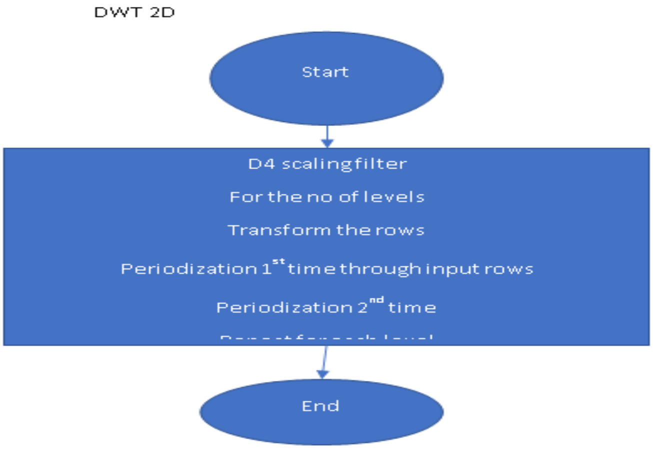


Figure 5 Inverse DWT 2D transform

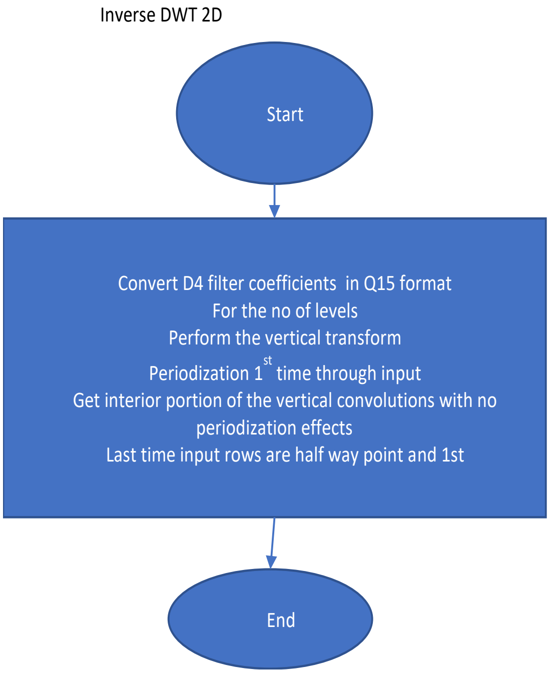
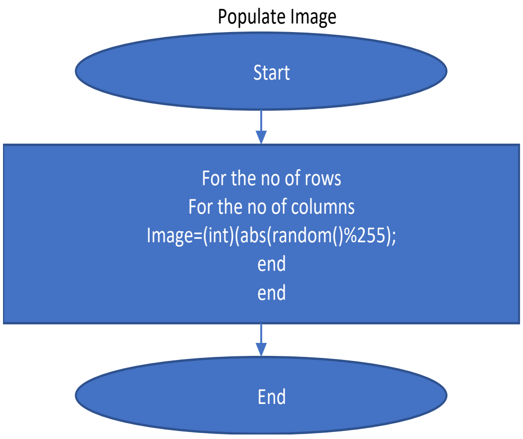

Inverse DWT 2D
Start
Convert D4 filter coefficients in Q15 format
For the no of levels
Perform the vertical transform
Periodization 1st time through input
Get interior portion of the vertical convolutions with no periodization effects
Last time input rows are half way point and 1st
End

Figure 6 Populating the image using a random number generator



III. RESULT ANALYSIS

The image generation and client programs were tested for wavelet encryption. At the server the received image was properly decrypted by using an inverse discrete wavelet transform. At this stage image processing functions were properly used to equalize the histogram and the image was enhanced by adaptive filtering. Now different masks were applied to segment the image. Morphological operations were correctly applied to extract features. Thus the required objectives were properly met in the implementation

IV. CONCLUSION

A sample implementation of the client server has been made by writing C programs for the client and server in Linux. The image processing is implemented using specific functions written in C. The execution results show that the simulator has been properly implemented.

REFERENCES

1. Benatallah, B.; Casati, F.; Toumani, F. (2004). "Web service conversation modeling: A cornerstone for e-business automation". *IEEE Internet Computing*. 8: 46–54. doi:10.1109/MIC.2004.1260703.
2. Dustdar, S.; Schreiner, W. (2005). "A survey on web services composition" (PDF). *International Journal of Web and Grid Services*. 1: 1. CiteSeerX 10.1.1.139.4827. doi:10.1504/IJWGS.2005.007545
3. Harper, Douglas. "server". *Online Etymology Dictionary*. Retrieved 30 November 2013.
4. Tolia, Niraj; Andersen, David G.; Satyanarayanan, M. (March 2006). "Quantifying Interactive User Experience on Thin Clients"
5. Otey, Michael (22 March 2011). "Is the Cloud Really Just the Return of Mainframe Computing?". *SQL Server Pro*. Penton Media. Retrieved 1 December 2013.
6. Barros, A. P.; Dumas, M. (2006). "The Rise of Web Service Ecosystems". *IT Professional*. 8 (5): 31. doi:10.1109/MITP.2006.123
7. Nieh, Jason; Yang, S. Jae; Novik, Naomi (2000). "A Comparison of Thin-Client Computing Architectures". *Academic Commons*. doi:10.7916/D8Z329VF. Retrieved 28 November 2018
8. "Distributed Application Architecture" (PDF). Sun Microsystem. Archived from the original (PDF) on 6 April 2011. Retrieved 2009-06-16.

AUTHORS PROFILE

Arvind Vishnubhatla Areas of Interest: Signal Processing , Image Processing , Real time Embedded Systems , Photonics
Education

- Ph.D., Jawaharlal Nehru Technological University, Hyderabad
- MS, Northern Illinois University,Dekalb IL
- BE, Thapar institute of Engineering and Technology, PATIALA

Academic/Industrial Experience

1. Professor, ECE, Gokaraju Rangaraju institute of Engineering and Technology , 2009-Present
2. Associate Professor,ECE, Gokaraju Rangaraju institute of Engineering and Technology , 2002-2009
3. Project Manager, Ericsson Communications India Limited, 2000-2002
4. Senior Software Engineer, Motorola, Location (Network Solution Sector, Schaumburg, IL), 1999-2000
5. Senior Software Engineer, Motorola, Location(Bangalore), India, 1993-1997
6. Software Engineer, Bharat Heavy Electricals Ltd(Electronics Division. Bangalore) 1989-1992