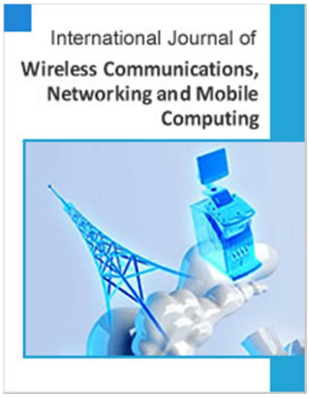
International Journal of Wireless Communications, Networking and Mobile Computing
2017; 4(2): 16-23
http://www.aascit.org/journal/wcnmc
ISSN: 2381-1137 (Print); ISSN: 2381-1145 (Online)

American Association for Science and Technology



Keywords
Raspberry-Pi,
Wi-Fi,
Remote Frame Buffer,
Virtual Network Computing,
Web Socketing

Received: March 8, 2017
Accepted: May 3, 2017
Published: August 8, 2017

Wireless Data Transmission Through VNC and RFB over Web Socketing

Sachin Ruikar, Manoj R. Jagdamwar

Department of Electronics Engineering, Walchand College of Engineering, Sangli, Maharashtra, India

Email address

ruikarsachin@gmail.com (S. Ruikar), mk5491@gmail.com (M. R. Jagdamwar)

Citation

Sachin Ruikar, Manoj R. Jagdamwar. Wireless Data Transmission Through VNC and RFB over Web Socketing. *International Journal of Wireless Communications, Networking and Mobile Computing*. Vol. 4, No. 2, 2017, pp. 16-23.

Abstract

This research paper consists of video data transfer in the using raspberry pi. The wireless transmission module is used as a tool for communication media between the devices. It use VNC and RFB protocols over web socket to transfer and access the PC desktop remotely with the help of Raspberry- pi. Now days the popular way to access a specific system is by using VNC (Virtual Network Computing). VNC offers user to share desktops over available network. VNC basically allows user to view and control the interface of another available computer remotely and can be thought of as the GUI (graphical user interface) equivalent to Telnet. In this work VNC server is used for front end system and for back end connection at server side TCP is used. There is need to develop client side which is open in browser. This requirement of opening browser for support web socketing.

1. Introduction

Wireless transmission refers communication system without any physical medium unlike wired transmission system. In modern communication system, information transfer via wireless medium is very crucial aspect. In wireless communication system, data or information transfer is done with the help of electromagnetic waves. EM waves include use of radio frequencies, infrared, satellite communication. This type of system is set and air is the medium used for data transmission. The end of 19th century is the dawn of wireless transmission system. The relevant technology has been developed over next century. Today, wireless technology is present in every aspect of communication systems ranging from Bluetooth devices to smart phones. Also, for data transfer in laptops, computers, printers and numerous other electronic devices wireless communication system is used. In today's world, space and accuracy are important aspect of data transfer. Hence, It is important achieve high data rate. Wireless devices using technologies like ZigBee, Bluetooth, Wi-Fi works with different data rates. IN smart phones, xender, shareit these types of applications are used for data transfer [1].

System consider an application in which multiple computer desktops are constructed on a server and streamed to remote display devices over an IP network. Applications executing on the server are interactive and, therefore, latency sensitive. Whereas the server typically includes high-end general computing resources, specialized graphical processing units (GPUs), and possibly specialized hardware for compression and streaming, the remote display devices (clients) are typically limited in processing capabilities. Our design objective is to deliver good video quality at low latency, while

minimizing processing requirements on the client. The server must be able to encode a large number of streams in real time. If implemented in hardware, the encoder must be a compact and scalable circuit. Software implementations must have modest processing requirements and should be able to use readily available hardware accelerators such as GPUs and multicore CPUs. Along with the encoder, it design a streaming protocol that is optimized for interactive desktop applications. In contrast to traditional video streaming protocols, the decoder cannot afford to have a large jitter buffer. Still, streaming must be robust in the face of packet losses and delays [2].

This paper is organized as follows. Section I describe introduction to communication. Section II illustrates proposed embedding system. Section III explain algorithm for protocols. Thorough representations of experimental results are presented in section V.

2. Proposed Embedding System

In this proposed system main focus is transmission of data with high speed. For this many systems are present but they won't provide required speed and not efficient. In proposed system it will transfer data between two platform (PC and Raspberry-pi) through wireless and for so that it will implement an algorithm. The proposed system will contain following major components as shown in Figure 1

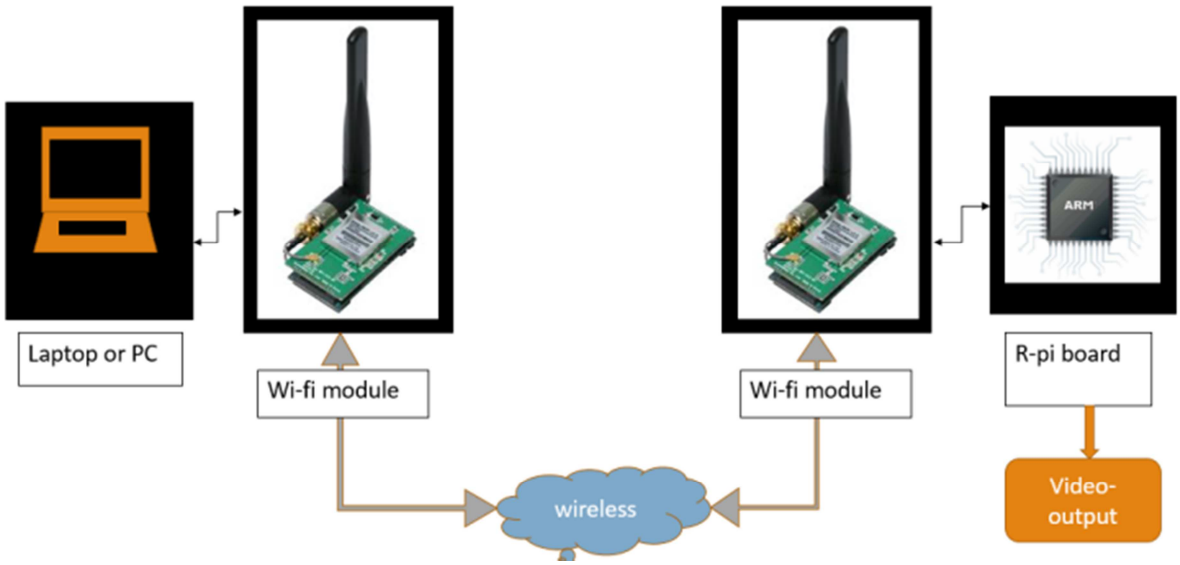


Figure 1. *Block diagram of proposed embeding system.*

2.1. Hardware

One single board computers (e.g. Raspberry Pi or any other relevant board) and PC or Laptop, Wi-Fi Module, Monitor, Power cable.

2.2. Software

System implement algorithm for high speed data transfer for that it use web socketing which allows full- duplex communication. System use VNC (Virtual Net- work Computing). VNC lets user share desktops over any network. It essentially allows user to view and control the interface of another computer remotely and can be thought of as the GUI (graphical user interface) equivalent to Telnet. It can also think of VNC as a long, virtual cable that enable user to view and control another desktop with its mouse, keyboard, and video signals. It use VNC with the Remote Frame buffer protocol, The Remote Frame buffer protocol (RFB) is an informational specification from the IETF (RFC 6143). A frame buffer is an array containing all of the pixel values displayed by a graphical computer system, and is the lowest common denominator model of a desktop computer. RFB is therefore a way to remotely access a frame buffer. For any system with a keyboard, mouse and screen, there is probably a way to access it with RFB [3] [4].

3. Software Implementation

The system use VNC server and RBF client protocol over the WebSocket for data streaming. WebSocket is an application level function, which run on transport layer and TCP is work as transport layer. It use TCP as a proxy server, which placed between VNC server and RFB Client as shown in Figure 2. TCP proxy server takes messages as WebSocket messages from server and passes as TCP messages to the Client.

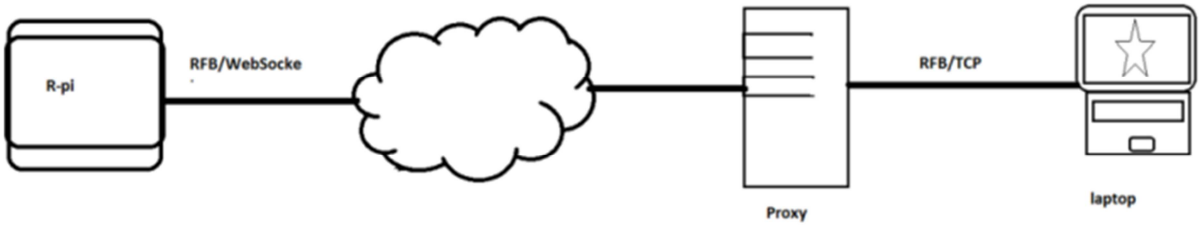


Figure 2. *Connecting with RFB over WebSocket.*

3.1. WebSocket Protocol

In the networking to communicate with server or client there are many issues, to eliminate these issues HTML5 includes WebSocket. WebSocket contain bidirectional, full-duplex, single-socket connection. After the WebSocket connection is created the server can send messages as the client available because of that the latency will reduces. After establishing the connection server send first message to the client, the server will not wait for the client request. WebSocket provide the real-time communication much more efficiently [5]. The WebSocket communication take place as shown in the Figure 3. The WebSocket protocol is a network protocol which define how server and clients communicate over the Web network. With HTTP request every WebSocket connection begins.

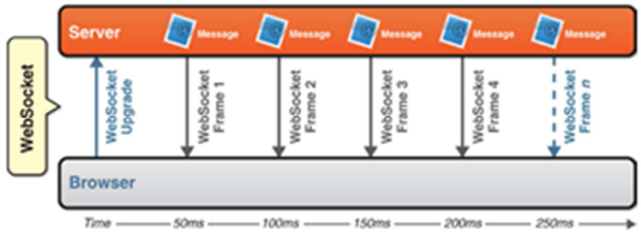


Figure 3. *The way WebSocket Handshake and communication take place between server and client.*

Ones the connection open between server and client, either server or client can send message to each other. The message represent as a binary syntax over the network, it also called as Frame. The Figure 4 show the structure of the WebSocket frame.

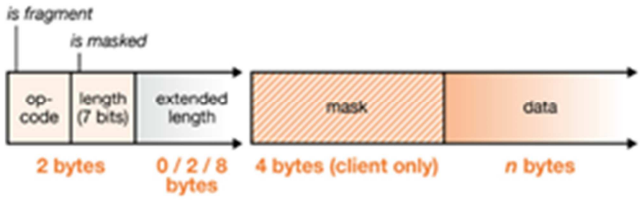


Figure 4. *WebSocket frame header.*

WebSocket Framing code is responsible for Opcodes, Length, Decoding Text, Masking, Multi-frame message

3.1.1. Opcodes

Each and every WebSocket message contain opcode which specifying the type of the message payload. The last four bits in the first byte of the frame consists opcode. Opcode contain numerical value as shown in table 1, only five opcodes define by the WebSocket Protocol.

Table 1. *Ready state attribute, values and status description.*

Attribute Constant	Value	Status
WebSocket.CONNECTION	0	The connection is in progress but has not been established.
WebSocket.OPEN	1	The connection has been established. Messages can flow between the client and server.
WebSocket.CLOSING	2	The connection is going through the closing handshake.
WebSocket.CLOSED	3	The connection has been closed or could not be opened

3.1.2. Length

The WebSocket Protocol uses variable number of bits to encodes frame length, which allows to use a compact encoding while still allowing the protocol to carry medium-sized and even very large message.

3.1.3. Decoding Text

With UCS Transformation Format 8 bit it encoded the Text WebSocket messages. For Unicode UTF-8 is a variable-length encoding, that is also backward compatible with seven-bit ASCII.

3.1.4. Masking

WebSocket frame are masked to unclear their contents which sent upstream from browsers to server. The purpose of masking is not to just prevent eavesdropping, but also to intend for an unusual security resone and to improve compatibility with existing HTTP proxies. To know whether the frame is masked or not is define by first bit of the second byte. The WebSocket Protocol requires that clients mask every frame they send.

3.2. VNC Server (Virtual Network Computing)

VNC is a popular way to access a specific system. VNC allow user to share desktop over any network. It specifically allows user to view and control the interface of another pc remotely equivalent to Telnet. VNC work as virtual cable which enables users to view and control another PC with it mouse, keyboard and video signals. There are several protocols for remotely accessing desktop. It use X11 for UNIX and Linux as a VNC Server [6] [7] [8] [9]. It is an Open source technology that is based on the RFB protocol. The Flow of the server is shown in following figure 5.

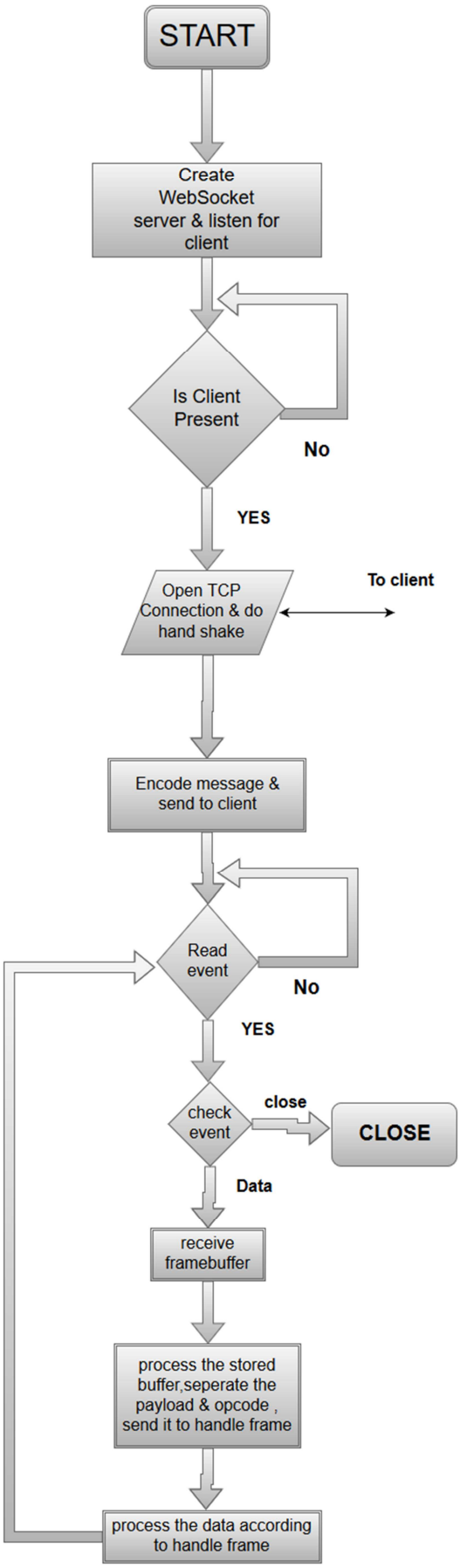


Figure 5. *Server side Flow chart.*

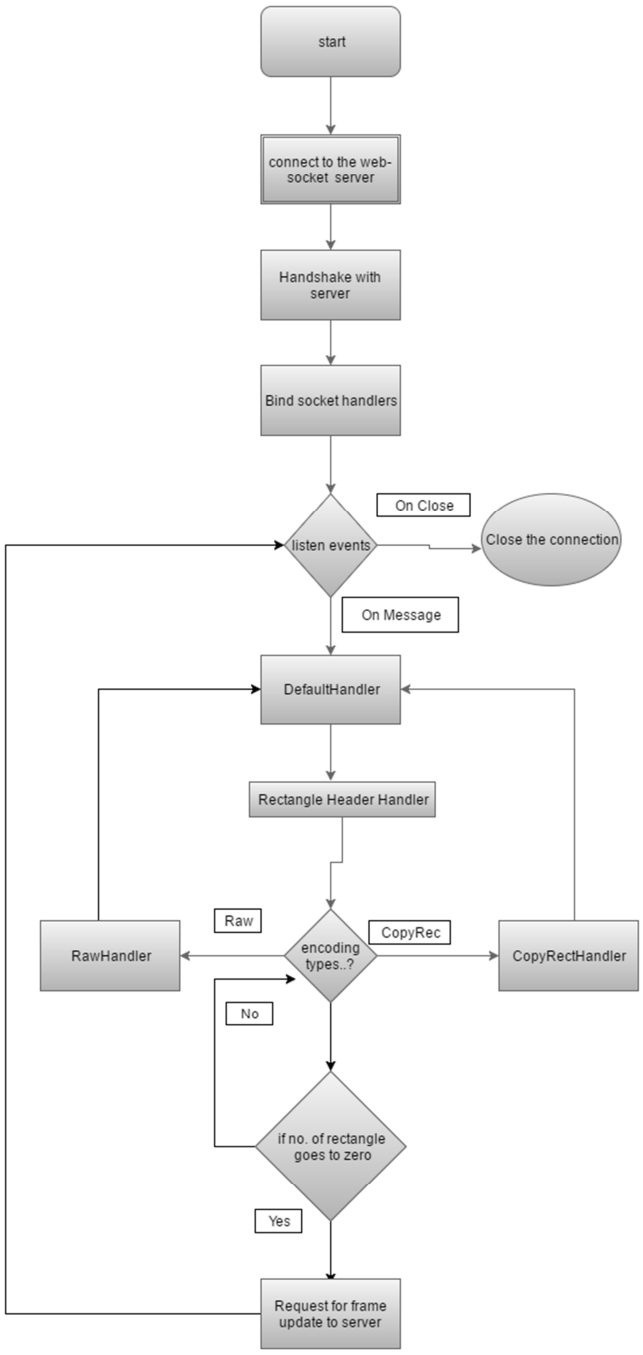


Figure 6. *Client side Flow chart.*

First system start the VNC server i.e. x11 on server, the server is Linux PC. System run WebSocket server program on server side. The server create the WebSocket variable with specific IP address and port address, the server listen the connection on this port address. So server continuously listen the client response until any client want to connect to server. At back end the TCP server is on, the TCP server take WebSocket Message and transfer as TCP message it work as transport layer. Ones the connection initiate first hand shake process will take place, in first phase they exchange the protocol ones it's happen it forward to authentication. The VNC server send the array of bytes that indicate the types of authentications it supports. After this server send a desktop

message which contain that if the client will share and allow other VNC viewer to connect to the same desktop, this message is only 1 byte in length. By this message handshake will complete, the server has to send first message to client which is server in it message, which contain screen dimension, bits per pixel, depth, big endian flag and true color flags. Ones it send message it send frame buffer update request, client send frame buffer when message event occur it check the opcode if it is data it process further else it close the web socket or error the message depending on the opcode. If it's a data it passes to process Buffer function, here it check the fin bit for checking the last frame, it check the frame length if it's to large then it send error message and it close the connection, it also separate the opcode and payload and send it to handle frame function. handle Frame function process the opcode and payload, opcode contain that the message contain text or binary data or pig or pong or close action if its close action then server close the WebSocket connection by giving the resone.

3.3. RFB Client

System can remotely access the graphical user interface by using RFB (Remote Frame Buffer), RFB works at framebuffer level. Windowing system like x11, windows and Macintosh use the same protocol. The protocol has good ability to exchange and use information so that the protocol is widely used. The RFB clients where the user sits typically with display, keyboard and printer and the RFB server is where changes to frame buffer originate. While designing the RFB protocol there are very few requirements of clients so clients can run on the widest range of hardware and implementation of client is also very simple. The state of user interface is resumed i.e. automatic reconnection is done to the same server after disconnection from given server. Same server can serve different client end points. User can see the same graphical user interface at two different end points, so the application becomes completely mobile. When user get network connectivity he can access his own application because of this, wherever user goes he finds familiar and uniform view of computing infrastructure [10] [11] [12].

RFB protocol can be based either on bytes stream transport or on message based transport. Protocol uses TCP/IP connection for its operation. In handshaking phase protocol version and the type of security are decided. In second phase i.e. initialisation phase clientInit and serverInit messages are exchanged by client and server. In the final stage of normal protocol interaction client can communicate with server through message and can receive reply message from the server. All these messages start with message type byte followed by intended data. Protocol messages use the basic types U8, U16, U32, S8, S16, and S32. Here U stands for unsigned integer and S stands for signed integers. Message format can also be having padding bits or bytes. For effective communication messages should be generated with padding value equals to zero and at the same time message recipient should not consider padding as any particular value. Figure 6 shows the flow of RFB client [13] [14] [15].

When system run the RFB Client on client side first it will connect to the given URL. Server listen the connection on this URL. Ones server know that client present then it initiate the connection by sending messages. Before Sending frames hand-shake will happen between server and client. Server send the protocol version message which contain small stream of byte. Client response for version will send to server. After Version exchange the readHandler control goes from VersionHandler to NumSecurityHandler. The NumSecurityHandler define number of security, from this type of security it has to choose one security type. For this the readHandler passes the control to securityTypesHandler. Here it select the particular type of security which it want in the communication throughout the connection. It can send back '1' to select non security. After selecting Security the readHandler passes to authSuccessHandler. Which allow the desktop sharing to server for this it send back '1' to server. By this the Handshake will complete and it listen the events, Ones the message event happen the read handler passes to DefaultHandler.

DefaultHandler passes the control to RectangleHandler rectangle has a count of rectangles which send by server. The RectangleHandler separate the opcode and payload from message depending upon the opcode the action on payload happen. If opcode contain binary data depending upon the encoding type RectangleHandler passes the control to one of the encoding type handler i.e. RawHandler or CopyRecHandler. Both of this handler paint the binary data on the screen using vnc.ss which allow the 2d painting on the screen. After executing the encoding handler the readHandler control passes to default handler. It also send the frame buffer to server in the response of framebuffer update request, with this frameBuffer client send framebuffer update to server. If the opcode contain close action then client close the connection with resone. Resone is for in which condition the connection will close.

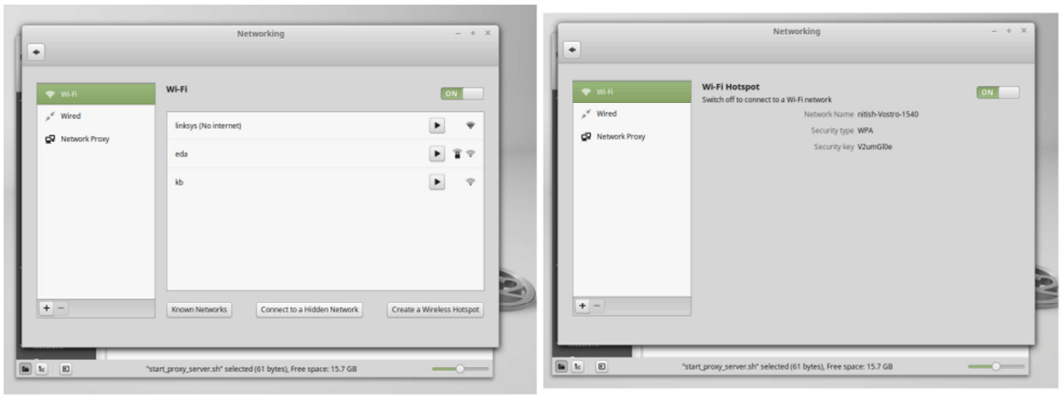
4. Hardware Implementation

For hardware implementation system required one Laptop with Linux operating System installed and have Wi-Fi connectivity, also with good processor. It also required one Raspberry- pi with Raspbean operating system installed and with up-graded version of browser. For wireless communication system use LeoSys Wi-Fi dongle for Raspberry-pi side. User has to start laptop and Raspberry-pi with monitor attach to the Raspberry-pi through HDMI cable, it also have to connect keyboard and mouse to raspberry-pi through USB cable. First system have to run x11vnc server on laptop after this it just run shell script file which also initiate the TCP server. In client side put all files in one folder and run the html file. User get the Server GUI on the client browser user can access whole server desktop from client browser.

5. Experiment and Result

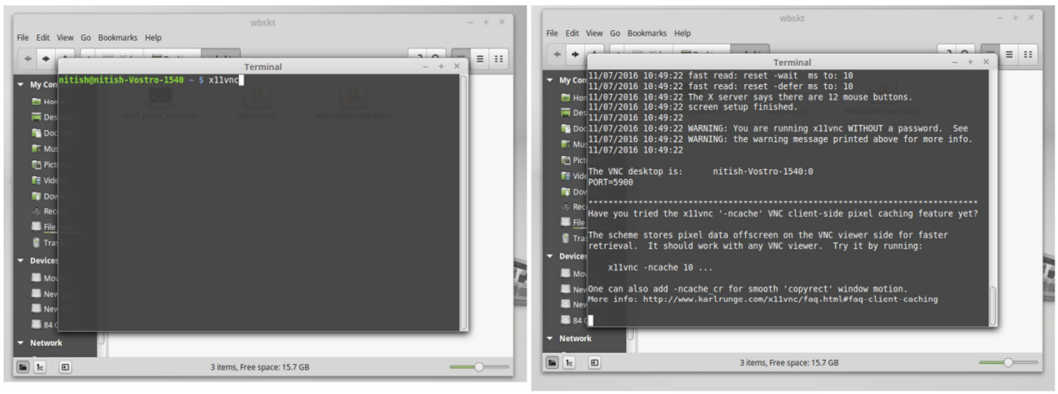
In this experiment system use PC and Raspberry pi as a two platform. At PC it create Access Point or Hotspot and R-

pi is station point as shown in figure 7. First it connect station point to access point so that it get the ip address, put that ip address in the URL so that it can connect to web socket server. Run the VNC server as shown in figure 8 followed by executing TCP proxy server as shown in figure 9. Opening and closing of TCP proxy connection shown in figure 10. Run client side web page user get the desktop control on the web page of server as shown in figure 11. Each event will transfer to the server i.e. keyboard and mouse and it get the update of event happen on the server.



(a) (b)

Figure 7. *(a) & (b) Execution of Wi-fi connection*



(a) (b)

Figure 8. *(a) & (b) Execution of the VNC server.*

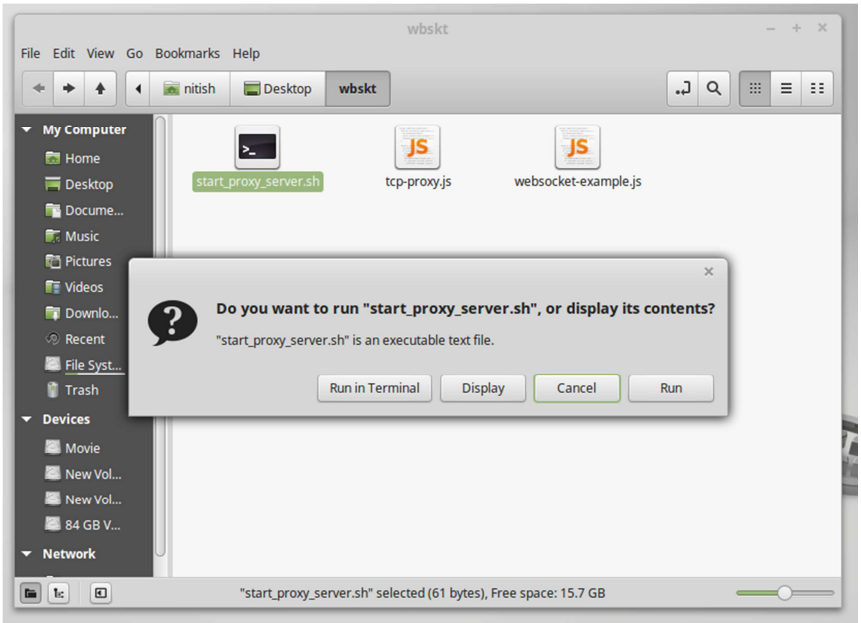
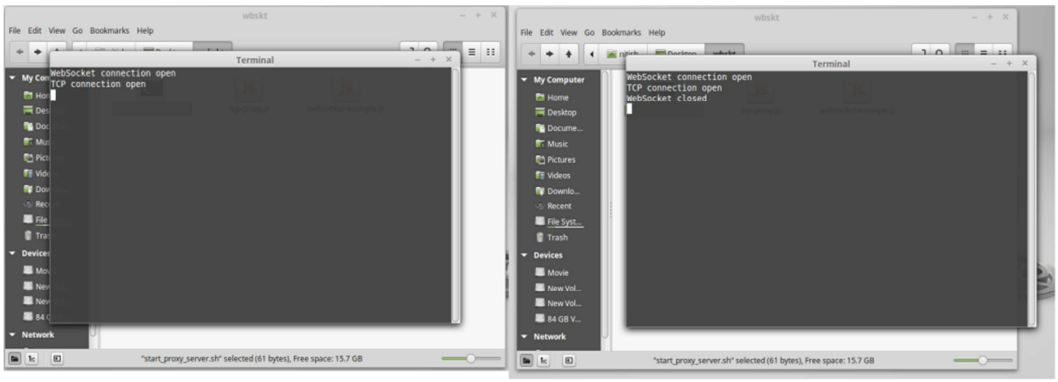


Figure 9. *Execution of WebSocket server.*



(a) (b)

Figure 10. *(a) & (b) Execution of TCP connection.*

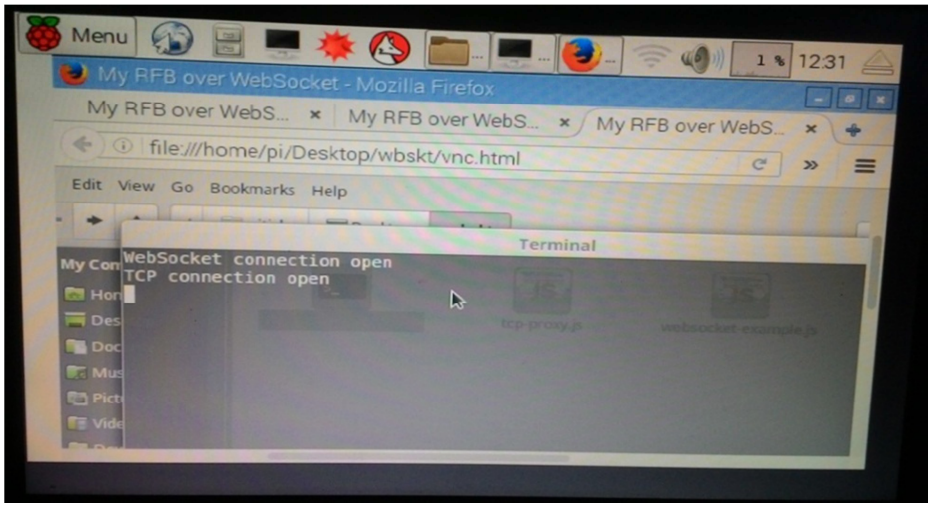


Figure 11. *Client side Screen.*

6. Conclusion

In this paper RFB (RemoteFrameBuffer) protocol with VNC (virtual network communication) server over the web socket. For implementation of this protocol JavaScript language is used. This protocol is tested for wire and wireless communication. Live desktop streaming is tested between two computers and between computer and R-pi. This protocol is tested specifically on Linux operating system.

References

[1] B. Eom, C. Lee, H. Lee and W. Ryu, An adaptive remote display scheme to deliver mobile cloud services, in IEEE Transactions on Consumer Electronics, vol. 60, no. 3, pp. 540-547, Aug. 2014.

[2] Jiewei Wu, Jiajun Wang, Zhengwei Qi and Haibing Guan, SRIDesk: A Streaming based Remote Interactivity architecture for desktop virtualization system, 2013 IEEE Symposium on Computers and Communications (ISCC), Split, 2013, pp. 000281- 000286.

[3] I. Hadic, H. C. Woithe and M. D. Carroll, A Simple Desktop Com- pression and Streaming System, Multimedia (ISM), 2013 IEEE International Symposium on, Anaheim, CA, 2013, pp. 339-346.

[4] Kyoung-ill Kim and Kyu-chul Lee, Game service platform based on the real-time streaming, Computing and Convergence Technology (ICCCT), 2012 7th International Conference on, Seoul, 2012.

[5] T. D. Nguyen, S. Choe and E. N. Huh, An efficient mobile thin client technology supporting multi-sessions remote control over VNC, Computer Science and Automation Engineering (CSAE), 2012 IEEE International Conference on, Zhangjiajie, 2012, pp. 155-159.

[6] D. Zinca, Design of a modified RFB protocol and its implementation in an ultra-thin client, Electronics and Telecommunications (ISETC), 2010 9th International Symposium on, Timisoara, 2010, pp. 157-160.

[7] K. J. Tan et al., A remote thin client system for real time multimedia streaming over VNC, Multimedia and Expo (ICME), 2010 IEEE International Conference on, Suntec City, 2010, pp. 992-997.

[8] G. Lai, H. Song and X. Lin, A Service Based Lightweight Desktop Virtualization System, 2010 International Conference on Service Sciences, Hangzhou, 2010, pp. 277-282.

[9] C. Taylor and J. Pasquale, Improving video performance in VNC under high latency conditions, Collaborative Technologies and Systems (CTS), 2010 International Symposium on, Chicago, IL, 2010, pp. 26-35.

[10] A. Skurski and B. Swiercz, VNC-based remote control for Symbian OS smartphones, Mixed Design of Integrated Circuits Systems, 2009. MIXDES 09. MIXDES- 16th International Conference, Lodz, 2009, pp. 171-174.

[11] Xiaolin Lu, WSFRB protocol and virtual program computing, Computer Supported Cooperative Work in Design, 2004. Proceedings. The 8th International Conference on, 2004, pp. 475-480 Vol. 1.

[12] H. Koriyama, K. Shimizu, T. Kawano, T. Ogura and M. Maruyama, Real-time processing method for an ultra-high-speed streaming server running PC Linux, Advanced Information Networking and Applications, 2004. AINA 2004. 18th International Conference on, 2004, pp. 441-446 Vol. 2.

[13] T. Richardson, the RFB Protocol, Version 3.8, August 2008, http://www.realvnc.com/docs/rfbproto.pdf.

[14] G. Lai, H. Song, and X. Lin, "A service based lightweight desktop virtualization system," in Service Sciences (ICSS). 2010 International Conference on. IEEE, 2010, pp. 277-282.

[15] A. Kivity, Y. Kamay, D. Laor, U. Lublin, and A. Liguori, "kvm: the linux virtual machine monitor," in Proceedings of the Linux Symposium, vol. 1, 2007, pp. 225-230.