Decentralized Cloud Method For Multicasting Media Stream

D M B N Bandara, W M J I Wijayanayake

Abstract: With the advancement of Information technology the concept of idea sharing has advanced. Mostly on presentations; personal computer and projector have become essentials. But on most occasions for connecting these equipment cables and physical devices are used. This is inefficient and time consuming. If a problem occurs, someone with technical knowledge is necessary to solve the situation. The objective of this research is to use the wireless technology to reduce the manual configuration and build up a platform where one can easily share files, a visuals, media and feedback. A system has been developed to detect all the devices over a network and upon granted permission, will share video, audio, and access controls. Final outcome of the research was a collaborative software bundle which work together on a network. One part of the system is a Desktop Network Software. And other is a Mobile Application. Desktop application can detect all other devices in the network which provides the same facility and if required can allocate a group and share it's screen, files and have a message stream to each device using multicasting. Mobile application can act as a mobile remote to the host computer of the group which can detect any input from user and pass it to the system.

Index Terms: Network, Multicasting, Streaming, Cloud, Network Protocol, Data Transmit, Memory Efficiency

1 Introduction

In an era where the information technology has become the pioneer advance tool for organizations and institutes when it comes to knowledge and idea transferring, sharing information through visual aids increases the efficiency. For most institutions and organizations use hard wired manual connections to connect a personal computer with a projector or similar device and share the visuals. Or the other method is to bring the files in a mobile device and transfer it to a set computer at a projector and present through that. There was time where more than several people have been presenting with same projector. They had to switch with PCs once each other's turn came and they had to remove and connect the VGA cable to the projector for each PC the presenters have used. The wire was short and it was messed up because presenters had to be near the computer to navigate the presentation and have tripped the wire because of that and the presenter was stuck at the presenting PC and had to keep track with the screen by glancing back at the screen whether he was running ahead or falling behind with his verbal facts. Some conferences are held in a fashion among fewer people without a projector so the host had to share the same PC with all the others to show his work where many people flock together and peered at the same screen. But in a world where every device is networked, even by wireless connections setup through almost everywhere, in a world where mobile devices are becoming revolutionary, do we need to keep our work this much complicated and so low technical where we are to set up the connections manually? Do we have to endure these technical difficulties?

- *Nawanjana Bandara completed bachelor's degree program in management and information in University of Kelaniya, Sri Lanka. E-mail: buddhilanj@gmail.com*
- *Co-Author name is currently pursuing masters degree program in electric power engineering in University, Country, PH-01123456789.*

This research project will be addressing a solution to solve the following issues.

- Often plugging in the cable overtime can damage the ports/pins.
- Once a new PC is connected should be reconfigured which takes time.
- Using a difference device may not appear the same as the original since they might not be the same software wise.
- Presentation might fail due to software incompatibility.
- If the presenter is not able to direct his full attention to the crowd if he is to turn back and at the screen.
- Inability to share a file amongst a group of friends quickly

2 Background Study

For two devices to communicate with each other, they need to be connected from the network first and they need to know each other's IP addresses. McCabe [1] furthermore discusses how IP address and devices should arrange according to requirements. Then a communication protocol for it to transfer data should be chosen. TCP and UPD are the possible protocols choices. Oracle documentations [2] explains TCP provides a point-to-point channel for applications that require reliable communications. The Hypertext Transfer Protocol (HTTP), File Transfer Protocol (FTP), and Telnet are all examples of applications that require a reliable communication channel. The order in which the data is sent and received over the network is critical to the success of these applications. When HTTP is used to read from a URL, the data must be received in the order in which it was sent. Otherwise, end result would be a jumbled HTML file, a corrupt zip file, or some other invalid information. The UDP protocol provides for communication that is not guaranteed between two applications on the network. UDP is not connection-based like TCP. Rather, it sends independent packets of data, called datagrams, from one application to another. Sending datagrams is much like sending a letter through the postal service: The order of delivery is not important and is not guaranteed, and each message is independent of any other. Generally speaking, a computer has a single physical connection to the network. All data destined for a particular computer arrives through that connection. However, the data may be intended for different applications running on the

computer. For this different port numbers are used for different application. The computer detects to which application the data should be forwarded to. Data transmitted over the Internet is accompanied by addressing information that identifies the computer and the port for which it is destined. The computer is identified by its 32-bit IP address, which IP uses to deliver data to the right computer on the network. Ports are identified by a 16-bit number, which TCP and UDP use to deliver the data to the right application. Harold [3] explains, with lot of basics, normally a server runs on a specific computer and has a socket that is bound to a specific port number. The server just waits, listening to the socket for a client to make a connection request. In connection-based communication such as TCP, a server application binds a socket to a specific port number. This has the effect of registering the server with the system to receive all data destined for that port. A client can then connect with the server at the server's port. If everything goes well, the server accepts the connection. Upon acceptance, the server gets a new socket bound to the same local port and also has its remote endpoint set to the address and port of the client. It needs a new socket so that it can continue to listen to the original socket for connection requests while tending to the needs of the connected client. On the client side, if the connection is accepted, a socket is successfully created and the client can use the socket to communicate with the server.

2.1 Multicast

Harte [4] among explanations discusses multicasting is a techniques used for real time communication over an IP infrastructure to communicate efficiently in a one-to-many data transfer situation. It enables many peers to receive data from one peer, without any knowledge about the origin peer's details. It enables many receivers to get data while one node uploads to the network. Multicasting can be done when one group is allocated one specific address and other receivers join this address. Multicast uses UDP protocol. When explaining how to remotely manage the devices in the network, with basics explanations of Richardson et al.[5], we can derive there are 3 protocols which are current and commonly used.

2.2 Remote Frame Buffering (RFB) protocol

This simple protocol is used in all VNC software and available for all windowing systems(x11, Windows, OSX) on all major operating systems. It basically works as follows

- Host sends a picture of the desktop across the network
- Client sends keyboard and mouse events to the host
- Then wait for host to render image and send it back to client

Most of the free implementations are not integrated with session management. Therefor it is difficult to use without a user at the host. Since this protocol is pixel based, frames (pictures) are transferred via network so that VNC software consumes much bandwidth than other protocols. This would be very useful to share desktop simply with other users with in a LAN.

2.3 Remote Desktop Protocol (RDP)

This proprietary protocol is defined by Microsoft and specification has been released under Microsoft open specifications. Jang [7] mentions basically, the host machine sends a description of the window and the method to render an image to the client machine, where the client machine is responsible for rendering the image and displaying it. Other processing rather than GUI would be done at the host. Since the client computer "understands" the image it has created, it can perform simple operations like moving windows without sending all mouse responses to the host and wait for the response. It can just calculate and draw the results right away. This causes highly responsive remote desktop experience. This protocol is integrated well with Microsoft windows logons and sessions and data is transferred through a secure connection. Unfortunately usual Microsoft windows do not support multiple user logins (Windows server version supports up to 2or 3 logins) which restricts major benefits which could have been achieved via remote desktop.

2.4 X11

This is an abstraction of GUI from other components of an operating system (kernel) in UNIX like operating systems. This is specially designed to be used over network connections and provides the basic framework for building such GUI environments on other supported platforms (Windows Clients are available like Cygwin). Put in simpler terms; all graphics are rendered at client and host CPU and its programs are used remotely passing commands back and forth. Unlike other protocols this has an application abstraction layer which supports 2D and 3D operations to be fully accelerated on the remote X server. Since X network protocol which is used for communication is based on X command primitives (with GLX, OpenGL 3D primitives), very little band width would be used to transfer commands.

3 METHODOLOGY

3.1 Presence Broadcast

First step on a network oriented application is to announce its presence. For this project a new protocol will be opened using a new port, and a broadcast signal will be sent to ensure the presence of the device. For this application, a broadcast signal was sent to the multicast address 224.0.0.1, which sends the message to all the hosts over the network. The message contained "[deviceType]:[uniqueName]:[ipAddress]". Then upon request from each device, ports were supplied after verification of whether it's requested by the same software. uniqueName is a unique name for the device over the network which is selected by the user. If there is a broadcaster to signal other devices of the presence of the system there should be a listener to catch that broadcast. Therefor there should be a loop to check on communication receipt on the defined open port. This keeps track of availability of each device. It also links human readable name and the IP address which acts as a similar functionality as in DNS.

3.2 Data Transfer

Next important part of this system is the data transfer. It transfers two types of data. They are Real time data and Stored Data. Stored data can be transferred the traditional way. Peer-2-Peer file transferring can be categorized this way using a TCP connection. But an issue arises when transferring real time data and transferring stored data to a group. Here it is required to use a UDP connection. The issue was that an UDP datagram can only contain 64kB. No data was smaller than 64 KB unless it is a chat message. To send data over 64

KB through a UDP Datagram the best method was to break down the packet and rebuild the packet in the other side. For that a special method needed to be introduced. Let's take an image for an instance. This requires to send large data; divided into small parts. When divided it is understood each packet should contain data,

- Which part of the whole file
- What is the size of the whole file
- Who is sending the file
- Which Image

In order to identify how to reconstruct the original data from the client end. Therefore a special method is introduced to transmit the data. In each network packet transmitted data contained can be divided in to two types of data. One is Metadata related to the partial image that is sending and the partial image data.

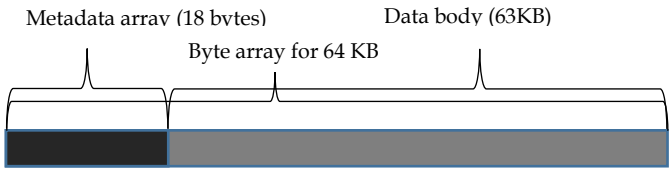

Fig 1. The graphical representation how one packet is divided

In this data packet for the metadata array, the 18 bytes are allocated in a certain pattern so that data encryption and decryption is unique, which is only identifiable by the cloud presenter application. The 18 bytes are allocated in a meaning full way (see Table 1).

TABLE 1

DATA REPRESENTATION OF BYTES IN METADATA ARRAY

Index of bytes	Data the bytes represent
1-8	timestamp of the full data (image)
9	Type of the data
10	Partial data part no. of the full data
11-14	Size of the partial data sent in this packet
15-18	The full data size (image size)

Shows how data was arranged in metadata array in a recognizable format

1) Get a screen recording session
2) Get the latest image
3) If no image in queue
4) Try again in given interval for stream time
5) Else
6) Get the last entered image
7) Get full size of the image
8) If the size is larger than 63 KB
   a) Until 0KB left
      i) Include the image timestamp in metadata
      ii) Include which part this is in metadata
      iii) Include the size of body data in metadata
      iv) Include the whole size of the image in metadata
      v) Include the next in line data Max of 63 KB in body data
      vi) Send data packet to receiver
      vii) Set for sending packet
      viii) Set the receiver
      ix) Identify whether a group transmission or individual transmission
      x) Set the receiver accordingly
      xi) Sending the packet

3.3 Group Broadcasting

Core functionality of this system is its ability to create a group in this decentralized cloud space and show the presentation at several devices. The specialty of this is that the system uses multicasting to make the data traffic of the network more efficient. Once a new group is created, a multicasting address is allocated and human understandable name is also bound. The device which created the group will be the administrator and all this data will be stored in each device and will be synchronized. Once it comes to group broadcasting all video transfer, data transfer and user input transfer, they will be converted to UDP communication and algorithms will be changed.

3.4 Mobile Remote

For user's convenience, a mobile part is also allocated to let the user control one device using one mobile device. It enables the user to input any key command or accelerate the pointer in any direction in any speed.

3.4 System algorithms

An algorithm used is to capture a recording of the screen. Next is the removal of useless old images in the recording session. Another process is sending the stream to another device Next Algorithm is the situation of receiving a Data package Then is the reconstruction of an image when a partial data is received from the imagestream

4 CONCLUSION

Final outcome of this project contained 2 parts: A desktop application and a mobile application. Both mobile and desktop applications can detect all the available devices in a single local network. It contains both a broadcaster and a receiver.

The Desktop software version can detect the other computers connected in the network and enable them to share screens, exchange messages, and exchange files. The other part, mobile solution which provides a remote control service to the devices connected to the network. A presenter can setup this mobile application on an Android based mobile device and simply connect to another device on the network and request for authorization for remote. Final outcome of this project was a platform where a person/presenter is provided with a platform where he can present using several resources in different devices linked together using a wireless network.

1) Start a new Session
2) Define a buffer time (How long the image will be kept in queue for transmission before discarding)
3) Define an interval between screen capturing.
4) Capture the image of a screen.
5) Tag the timestamp to identify the image.
6) Include it in the queue in ready state for transmission

1) Decode the metadata of the packet
2) Get the timestamp of the image
3) If the image partially sent before
   a) Get the partially sent image
4) Else
   a) Get the image size
5) Create a new partially received image to the identified size
6) Add the partial image to the relevant place
7) Mark that this part has been received
8) If all parts are received

1) Open a path to receive data
2) Wait for receivable
3) Identify from whom
4) If New sender
   a) Create new stream
   b) Return the newly created stream
5) Else
   a) Get existing stream
6) Send the part to the stream to add in a new Thread

REFERENCES

[1] J. D. McCabe, Network Analysis, Architecture, and Design. Morgan Kaufmann, pp. 250-293, 2007.

[2] Oracle, “Networking Basics,” Oracle Documentation, https://docs.oracle.com/javase/tutorial/networking/overview/networking.html 2015.

[3] E. R. Harold, Java Network Programming. O'Reilly Media, pp. 019-045, 2004

[4] L. Harte, Introduction to data multicasting. Althos Publishing, pp. 01-07, 2008.

[5] T. Richardson, Q. Stafford-Fraser, K. R. Wood, and A. Hopper. "Virtual network computing." Internet Computing, IEEE 2, no. 1, pp. 33-38. Jan. 1998.

[6] T. Richardson & K. R. Wood, “The RFB Protocol”, ORL, Cambridge., Version 3.3, January 1998.

[7] S.J. Jang, “Design of the Remote Management System in the Windows Operating System,” IJCSNS 11, no. 11, p.38, 2011.