International Journal on Recent and Innovation Trends in Computing and Communication ISSN: 2321-8169
Volume: 3 Issue: 4 2398 - 2403

Video Streaming Using Message Accelerator

Luvish Yadav, Prof. A. J. Jadhav, Chaitanya Pol, Akshay Teke, Saurabh Yeramwar
Department Of Information Technology, Savitribai Phule Pune University
luvishyadav@gmail.com, polchaitanya92@gmail.com, saurabhyr@gmail.com

Abstract:- Virtual Network Computing or VNC is a largely used client application for accessing files and applications on remote computers. When there is high latency between the client and the server, VNC can undergo from major losses in throughput. These losses become obvious in the case of video, where updates are both large and continuous. Message Accelerator proxy for VNC is simple but highly effective solution for video performance while maintaining the advantages of a client-pull system. By operating on the server, it sends updates to the client at a rate corresponding to proxy-server interactions which are quicker than client-server interactions. When testing using video, Message Accelerator's results are more superior to VNC under high latency condition. Message Accelerator uses the pipelining system for updating the frames, which increases its performance to a great extent. Message Accelerator here is not a hardware part but software that we have to just apply in our video streaming program.

Keywords*: Clients, VNC, Video, Latency, Networking, Message Accelerator.*

*****

I. INTRODUCTION

Thin client systems, have ability to connect multiple users to same computer system. Users can connect to multiple sessions on the same computer and can share the data and application or can have many numbers of connections to the same desktop allowing them to share the same virtual screen. The functionality to run multiple desktop sessions on a single sever offers other more system related advantages having all of the data and processes running on central server means that a business can update just software on one computer instead of hundreds or thousands of computers, a saving money and time on maintenance. Storing data on a central server provides security from lost PCs of the employees. Using thin clients electricity cost can be reduced to great extent.

The problem with thin client system is that they may suffer from poor performance. There can be computational overhead on the server side of tracking and encoding updates and of decoding them on client side. The limiting factor of thin client is bandwidth or how much data can be sent across the network in given time. The most significant limiting factor in thin client system is work latency. Latency can affect every message, irrespective of its size, sent between the client and server.

Performance is important to all tasks, it is more important for videos. Highly interactive task has less update rates as compared to video. The applications, with their infrequent screen updates are less impacted by high network latency. Video application require frame update every 30 to 60 milliseconds and update usually involves change in a large number of pixels. Due to its frequent updates, video is impacted by high latency condition.

VNC has been one of the popular systems for thin client research. Research on VNC has included adapting it for high resolution displays, to control home appliances and for optimal viewing on cell phones. VNC is one of the most ubiquitous thin client systems.

Message accelerator works with VNC to mitigate the effects of network latency. The message accelerator runs on server machine and requests the server in client-pull fashion. It then forwards these updates to clients as soon as possible. The message accelerator is not affected by high network latency because it does not wait to get a request from client before sending. Message accelerator with VNC can be used for receiving updates in high latency situations.

II. LITERATURE SURVEY

Details of **VNC**:

The VNC focuses on sending of message between VNC client and VNC server.

a. VNC CLIENT

The VNC client is simple program, it is not multithreaded program and it blocks on reads and writes. After the initialization phase in which VNC exchange setup information with server, it falls into while (1) loop. In this loop, VNC client waits to receive an update from server. Then it processes the update and redraws display. Then it issue request from new update. The client just read the calls

throughout processing of updates, rather than reading the entire update after that it will process it.

The update contains general header containing information about message, series of rectangles and its encoding. There are times when client has processed as much of update as it has received, but cannot read more from server. When this situation happens, client uses idle time to collect user input and sends proper message to server.

b. VNC SERVER

VNC server is more complex than client. As server runs, it notes when the frame buffers each change. It stores internal representation of area modified and new modifications made to it; this is called as modified region. Client sends request to the server, the first request is not incremental and all the requests afterwards are incremental. If request received by server is not incremental then the server immediately sends all of the frame buffer information to the rectangle. If the request is incremental then server compares the rectangle to the modified region. If they overlap then server sends update with modifications.

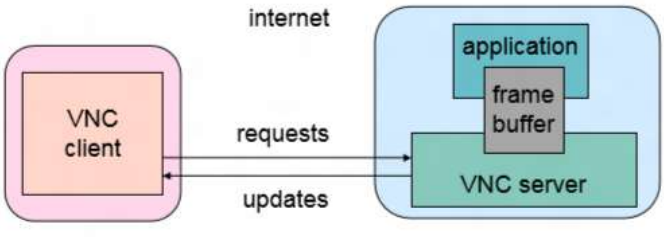

Figure 1. The VNC Client/Server Architecture[1]

c. VNC AS A CLIENT-PULL SYSTEM

All VNC systems use client-pull, in high latency situations the inner-update time is dominated by times spent on network, rather than compute time. Because of this our results are generalised to any VNC system.

d. AMES FRAMEWORK

AMES cloud framework includes the Adaptive Mobile Video Streaming and Efficient Social Video Streaming. The whole video storing and streaming system in the cloud is called video cloud (VC). In VC, there are video bases which stores video clips for video service providers. Temporal video base is used to cache new candidates for the videos. VC stores the videos into tempVB first. The cloud service may come across different places or even continents, so in the case of video delivery between different data centers, and transmission will be carried out, which is called “copy”. Because of optimal deployment of data centers, the “copy” of large video file takes small delay.

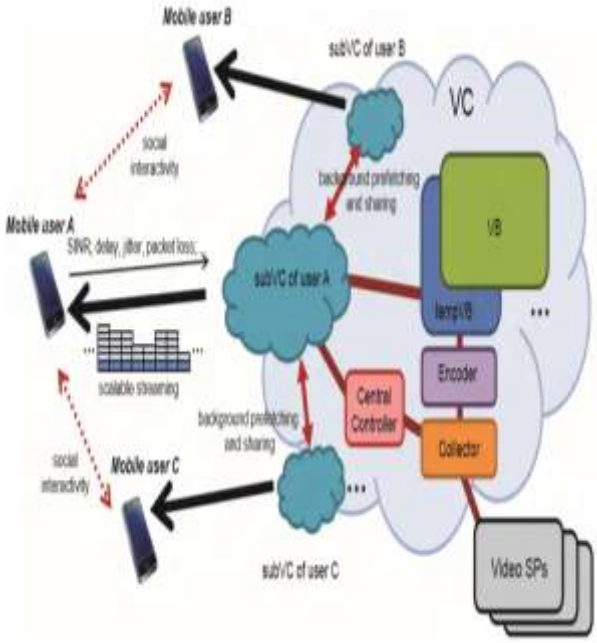

Figure 2. An illustration of the AMES-Cloud framework [1]

III. Proposed System

In the proposed system we are going to try to improve the streaming quality of the video. We have seen that users face saviour problem while streaming. While desktop streaming through internet, there we send large amount of data packets or heavy data. These data are of large size and which takes more time to send as well as receive the data.

While capturing the data or frame of any video and sending it we not only send the frames, the data is attached with additional data and noise. These data are not useful and also increase the size of the frames.

In our proposed system we are going to remove the noise and data which are not at all useful. We are going to send only the actual frame that is the output we required. We also apply compression technique to reduce the size of the frames. Due to this it is possible to send and receive the frames faster than that we used to do previously.

For compression we use CS Video Encoder (CSV)

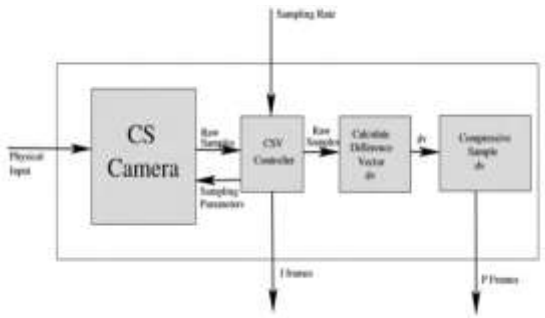

Figure 3.Block diagram for CS video encoder.[3]

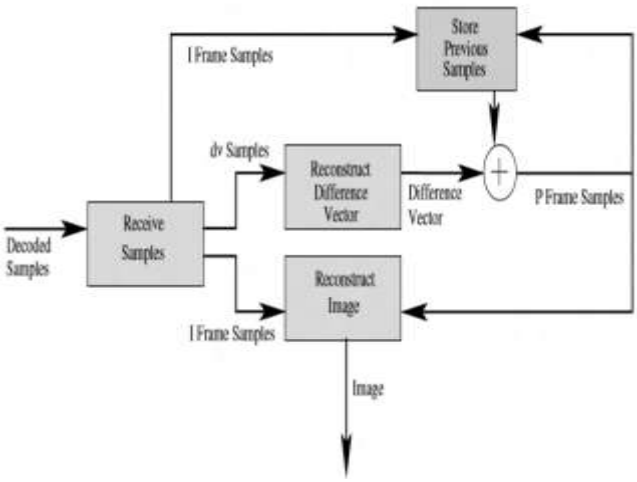

Figure 4. Block diagram for CS video decoder.[3]

We are going to use the Message Accelerator. Message Accelerator simply forwards everything sent to it by the client to the server, and everything sent to it from the server to the client. This period is where the client sends the server information such as the user's password, what encoding the client would like to use, and other parameters. It is necessary for the Message Accelerator to simply forward this information, since there is no way for it to know these parameters ahead of time.

The Message Accelerator sends requests to the server at the rate the client is processing them, and quickly receives updates from the server. This lets the Message Accelerator adjust for latency between the client and server

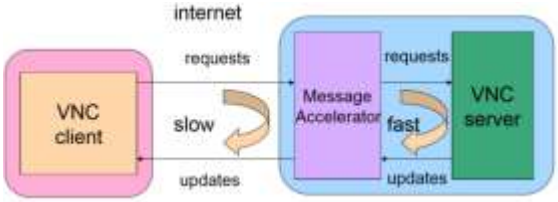

Figure 5. VNC with Message Accelerator[8]

As we know in video streaming the most important part is updating the frames. The quality of the video depends on how fast the frames are updated. So the Message Accelerator uses the pipelining system. Due to the pipeline update, the proxy sends requests to the client at the rate the client is processing, without waiting for a request.

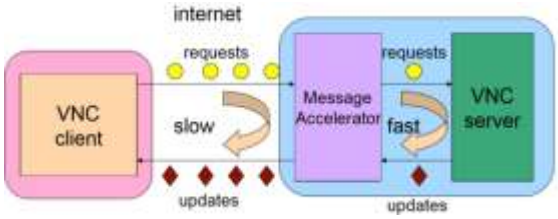

Figure 6. Pipelining used in Message Accelerator.[8]

Using VNC system under high latency we get the update rate up to 2 -3 frames per sec, which is very less and will make our effect our streaming.

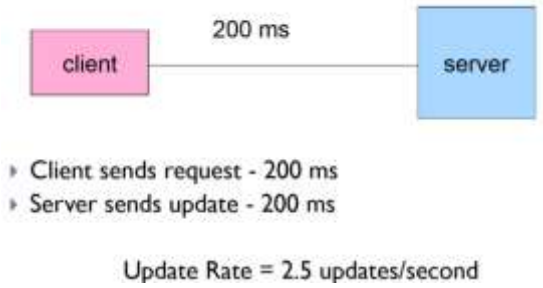

Figure 7. VNC w/o message accelerator in high latency[8]

Solution to this is using the message accelerator. Message Accelerator increases the frame update rate which makes the video streaming more efficient.

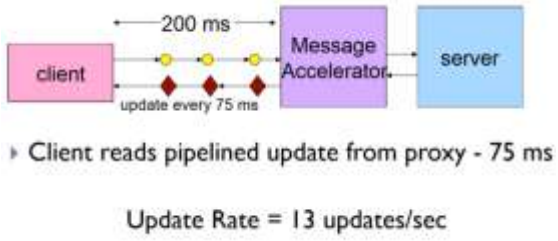

Figure 8. VNC w/o message accelerator in high latency[8]

We can improve VNC performance by having a Message Accelerator mediate the update rate over network latency. By using the Message Accelerator, we do not have to modify an existing code, avoiding issues of parallel code maintenance and source code availability.

III. IMPLEMENTATION

Streaming is mostly referred as a delivery system for media content or dynamic data where it is beneficial to begin processing while data is being delivered. In reality, HTTP was not designed for data streaming. HTTP communications are stateless, and they take place over TCP/IP where there is no continuous connection between the ends. Usually, HTTP responses are buffered rather than streamed. HTTP 1.1 added support for streaming through keep-alive header so data could be streamed, but yet for performance proposes, the majority of implementations including ASP.NET tend to buffer the content, send it, and close the connection. As a result, there are few real world applications that use HTTP for streaming data, and normally, an additional protocol is built on top of HTTP for reconnection and error detection. However, this does not pose a problem because there are other UDP-based protocols that can be used for streaming where it is needed.

ISSN: 2321-8169
2398 - 2403

So, why would we need data streaming over HTTP? Because, we build our web applications over HTTP. Playing video clips, displaying RSS fields, and updating time sensitive data are considered common features of a webpage nowadays, but yet, we are bounded to HTTP capabilities. Here is where the browsers make use of plug-ins to overcome these boundaries, and also add new troubles!

Plug-ins are executed outside of your application's context. Unlike Hyper Text Markup Language or JavaScript, plug-ins are mainly compiled binaries and they are difficult to customize. This is not to mention security, accessibility, platform independency, and web standards issues that are involved in pages that use plug-ins. While using plug-ins seems to be inevitable for pages with rich contents, AJAX has created high hopes in my opinion. Even though JavaScript language, as the version of today (the latest 1.7 in Firefox 2), is not fully capable of performing the tasks associated with plug-ins, I believe the future versions can offer enough browser integration and supporting libraries which eliminate the need of using complied plug-ins. I know this sounds very abstract, that is why I decided to write an AJAX application that does the most common task associated with plug-ins: video player. This AJAX video player is a scripted prototype-based video player that runs in JavaScript enabled HTML browsers that support Base64 encoded images (almost all modern browsers but IE). The AJAX video player can broadcast live (using an XML service) or cached video streams (XML file) to a variety of users on different platforms and browsers.

a. BACKGROUND

A video is a sequence of framed images that are displayed at a rate one after another. If we had all frames of a video clip in our browser, we could display them one after another at a frame rate and there we had our video playing! This sounds like a plan, let us see how we can translate this into an actual web application. From what we planned, we divide our efforts into smaller steps:

Step 1: Getting the frames, frame rate and other necessary information from a video file or a live stream

Step 2: Transport our frames over HTTP to the client's browser.

Step 3: Animate the frames at the client, response to user interaction and request for more frames if needed.

We have used the JMF API here to show the video streaming between the 2 computers .In the process we are giving the user the option to create the session with another computer directly by a search or by giving the appropriate IP address followed by PORT number.

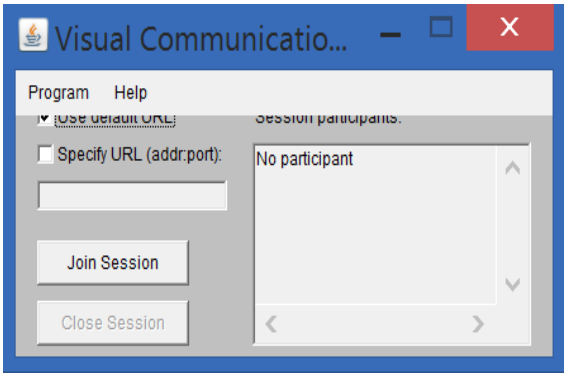

Fig.9 Video Stream creating session.

After joining the session directly or by giving the IP address we will receive the following output.

Fig 10. Without Message Accelerator

Here we can clearly see that the frame rate is 15 fps connected over wired LAN.
But our aim is to increase the FPS and the quality of the video. This can be achieved by adding MESSAGE ACCELERATOR.

Adding Message Accelerator

We have used the JMF API here to show the video streaming between the 2 computers .In the process we are giving the user the option to create the session with another

 ISSN: 2321-8169
2398 - 2403

computer directly by a search or by giving the appropriate IP address followed by PORT number.

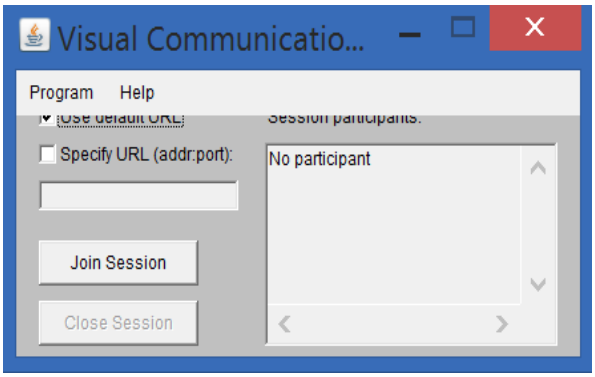

Fig. 11. Video Stream creating session.

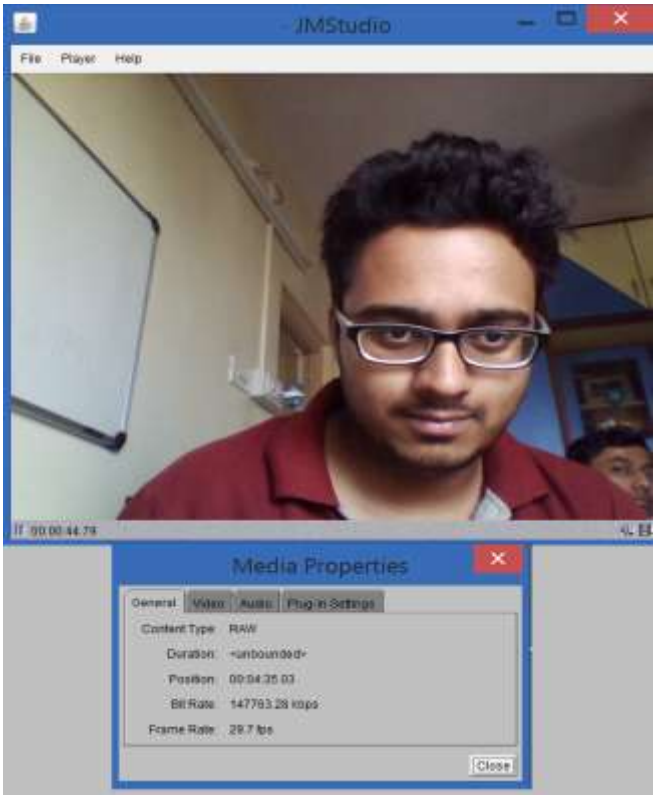

Fig 12. With Message Accelerator.

We can clearly see that after adding MESSAGE ACCELERATOR the FPS and the Quality of the video has been changed to a great extent.

FPS change - 15 fps to 29.7 fps.

In the same way we can also use our message accelerator to stream the video in mobile, when the system and the mobile are connected to the same network.

The message accelerator part of our program is over the server side that is our system where the video is captured. The client side contains our mobile phone which will display our video.

There is no special application required to run the video. Only requirement here is the java enabled web browser.

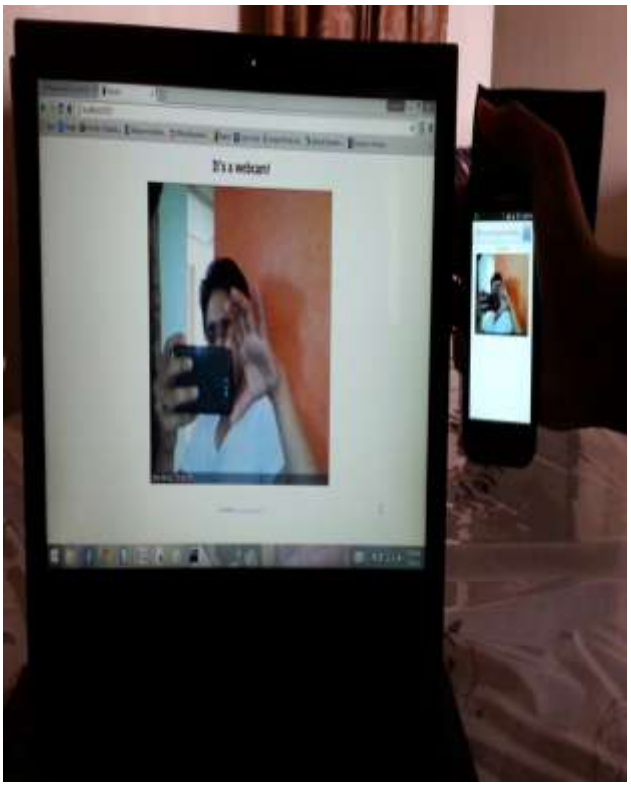

Fig 13. Mobile Video Streaming.

IV. FUTURE WORK

Using an adaptive message-accelerating proxy we can build other systems. The message modifying program offers a simple way to improve performance or add additional features, is easy to deploy, and works with existing binaries. It does not suffer from problems of parallel code maintenance, and will continue to work as long as the message format between the client and server remains the same. The proxy can even be deployed to a different machine from the server, and still offer performance advantages. There are additional ways that a modifying proxy could be used with thin client systems. The proxy could dynamically tightly compress updates when network speeds were low, and uncompress when the client device had low batteries or other computational issues. With an additional client application, the server application could encrypt updates, and the client application could decrypt them either on the client machine, or on a machine with a trusted network connection to the client. The server application could also perform machine vision tasks such as object detection or face recognition. Clearly, adding an adaptive proxy to a client-pull system offers any number of ways to improve performance or trans- form data. Updates can be buffered, information can be cached, or messages can be modified in a wide variety of ways. While some systems

have taken some of these approaches, we are not aware of any other system that uses a proxy for message acceleration.

We are also looking forward to use this technique and implement it efficiently for distributed Bandwidth allocation. So the users will not suffer from unequal bandwidth allocation and can stream in ease working in LAN.

VI. CONCLUSION

Adding a Message Accelerator proxy to a VNC system is a very simple but highly effective way of improving video performance with VNC under high latency conditions. Even with small amounts of network latency, video performance is as good as or better than an unmodified VNC system. Installing is easy, requiring no recompilation of client or server code. A video displayed using the Message Accelerator system will look almost the same, in terms of frame rate, with a 300 ms network latency as it does with a 3 ms network latency, while a video playing on an normal system will take ten times as long to display updates.

REFERENCES

[1] Cynthia Taylor and Joseph Pasquale Department of Computer Science and Engineering, University of California, San Diego {cbtaylor,pasquale}@cs.ucsd.edu "Improving Video Performance In VNC Under High Latency Conditions".

[2] Sergey Smirnov, Atanas Gotchev "A DISPARITY RANGE ESTIMATION TECHNIQUE FOR STEREO-VIDEO STREAMING APPLICATIONS", IEEE TRANSACTIONS ON CLOUD COMPUTING, VOL. 9, NO. 3, MARCH 2014.

[3] Xiaofei Wang, Student Member, IEEE, Min Chen, Senior Member, IEEE, "AMES-Cloud: A Framework of Adaptive Mobile Video Streaming and Efficient Social Video Sharing in the Clouds", IEEE TRANSACTIONS ON CLOUD COMPUTING VOL:15 NO:4 YEAR 2013.

[4] Scott Pudlewski, Tommaso Melodia, Arvind Prasanna "Compressed-Sensing-Enabled Video Streaming for Wireless Multimedia Sensor Networks".

[5] Maxim Claeys, Student Member, IEEE, Filip De Turck, Senior Member, IEEE "Design and Evaluation of a Self-Learning HTTP Adaptive Video Streaming Client".

[6] Q. Yang, L. Wang, and N. Ahuja, "A constant-space belief propagation algorithm for stereo matching," in Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on, June 2010, pp. 1458–1465.

[7] V. Menkovski and A. Liotta, "Intelligent control for adaptive video streaming," in 2013 IEEE International Conference on Consumer Electronics.

[8] Improving Video Performance in VNC under High Latency Conditions. Cynthia Taylor, Joe Pasquale University of California, San Diego

[9] http://cseweb.ucsd.edu/~pasquale/Research/Papers/cts10.pdf

[10] https://occs.oberlin.edu/~ctaylor/cts.pdf