Available online www.jocpr.com

Journal of Chemical and Pharmaceutical Research, 2013, 5(12):118-122

Research Article

ISSN : 0975-7384
CODEN(USA) : JCPRC5

An independence display platform using multiple media streams

Zhang Gen-Yuan[1,2]

[1]*School of Information Engineer, Wuhan University of Technology, Wuhan, China*
[2]*School of Electronics and Information, Zhejiang University of Media and Communications Hangzhou, China*

ABSTRACT

Sharing of display content amongst different display device such as laptop, desktop, handheld computers and smart phones (IPAD) is needed for collaboration. Because of different architectural, sharing of display content is complicated and the performance is very low. By using special media stream formats, the complexities and performance bottlenecks is reduced or avoided. Display content is encoded by continuous media, and a media server processes and distributes customized or fixed rate streams to viewers by streaming it to the media server, and playing it back on up to 1000 clients distributed over 20 computers. The CPU load is below 40% on the server.

Keywords: Display Technique; Media Streams

INTRODUCTION

The number and diversity of personal computational devices (PCs, laptops, PDAs, etc.) is increasing. The number of individual users increases. At the same time, more users start using multiple devices. This increased dispersion of personal devices causes an increasing need for sharing information and computational resources dynamically as users move between different networked environments where they want both to interact and to make use of available devices. As a result, there are very light load in the network. However, applications can hardly support different hardware and software architectures[1]. Graphics library model needs the same graphics libraries in devices. Pixel model is simple, but it spends more network bandwidth. But there is no general display sharing solution that is adopted across the range of display devices that we address[2]. Different sharing protocols are applied between notebooks, display walls, and PDAs. For example, the virtual network computing protocol (VNC) may be used to share desktop content among PCs; while a display wall typically needs a modified VNC to distribute and share desktop content across tiles and externally[3,4]. For reasons of compute and power restrictions, PDAs require other solutions. The scenario also requires that differences in network performance capabilities, display size and resolution, and graphic processing capabilities are handled. There are many applications with real-time requirements, such as 3D rendering. VNC achieves poor 3D frame rates, and effectively only supports 2D applications[5,6]. In summary, achieving seamless sharing of display content across the range of devices ad-dressed raises several issues for which there is no established answer. The contribution of this work is to develop and demonstrate the applicability and performance of a new pixel based display sharing system, MultiStream, that uses media streams as a unified solution to transmit encoded display pixel areas across networks to a wide range of display devices. MultiStream supports 2D and 3D applications and is independent of any shared applications. Using well established formats for encoding pixel data also allows our solution to benefit from the availability of a diverse set of highly tuned media players all platforms.

Design and Implementation

To support multiple users with a range of different devices and applications, MultiStream must (1) capture visual content without requiring the shared applications to be modified and (2) provide for heterogeneous clients that may require different media resolutions and encodings. To support multiple platforms and reduce network bandwidth usage, MultiStream uses standard media streams as the communication protocol to share content. The implementation includes three components, shown in Figure 1, that solve each of the problems identified above: (1) live streaming producers, (2) streaming servers, and (3) live streaming consumers.

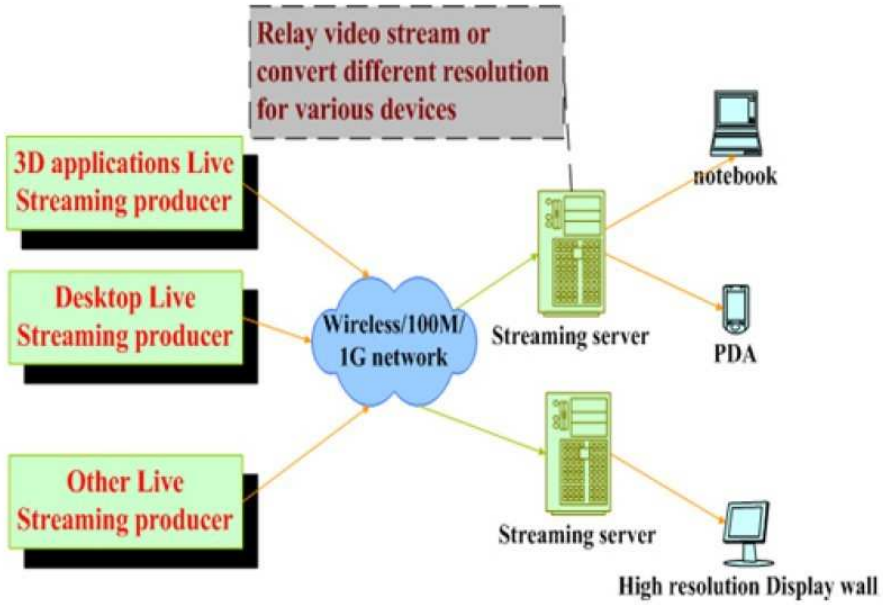

Figure 1. MultiStream Implementation

Capturing and Encoding

The goal of the live streaming producer is to produce media stream sources, which includes capturing pixels and encoding pixels. It requires the independence on shared applications in order to support various applications. The independence is based on the pixel model because the model reads pixels from frame buffers. Sharpshooter is one prototype of producers implemented by us, which is developed with visual C++. Sharpshooter supports to share display of desktop and 3D applications. Sharpshooter employs Windows hooking, which is a technique uses hooks to make a chain of procedures as an event handler. A hook is a point in the system message handling mechanism. When Sharpshooter starts, it registers the hook procedure which notifies Sharpshooter before activating, creating, destroying, minimizing, maximizing, moving, or sizing a window. When the system receives the window messages, Sharpshooter gains a chance to check whether the hooked application is needed to capture pixels. If Sharpshooter finds that the application is interesting, it modifies a few of the graphics library functions addresses. These functions are used to render the graphics into the frame buffer so that Sharpshooter reads the pixels data from the frame buffer when the hooked application updates the frame buffer each time. Sharpshooter sends the data to one thread, which is to encode pixels into media streams and send media over network. Sharpshooter uses FFmpeg library to deal with media/audio encoding.

Scale Media Server

Streaming servers will receive multiple media streams and provide them to consumers over network. At the same time, streaming servers also need to support some extra functions, such as converting between different resolutions of various devices and forwarding media streams between streaming servers in order to improve performance. We implement one scale media server with the FFmpeg library, which is a http media server. One consumer uses http proto-col to access data. The server also needs to provide flexible media qualities, such as different media formats,

resolutions and frame rates, in order to address devices polymorphism. When the server finds requests from one consumer is different in media quality, the server scales the media into the requested media quality and sends the scaled media to the consumer.

Performance

We have measured the frame rate which the 3DMark application can achieve with and without concurrently encoding a media of the frames. The media resolution was 320*200, 640*480, 800*600, 1024*768, and 1600*1200. Each experiment ran for 10 minutes. The result is shown in Figure 2. The five values on the X-axis is the number of kilobytes per frame for each of the five resolutions, respectively. Each pixel is encoded by four bytes. The y-axis is the frame rate of 3DMark. When 3DMark runs without encoding a media of its frames, the frame rate drops from about 50 to 18 with increasing resolution. When encoding is done concurrently with running 3DMark, the frame rate drops from about 38 to 4.

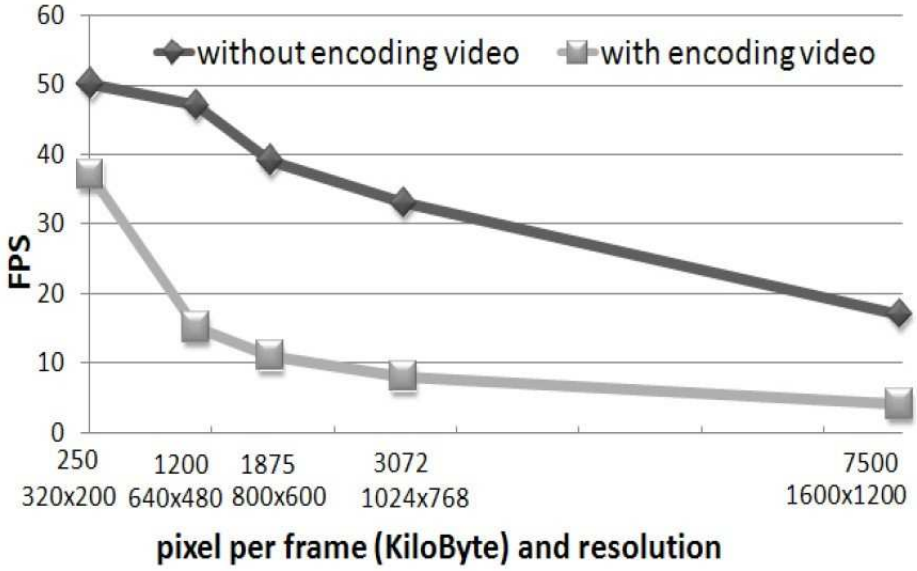

Figure 2. Performance impacts on producer

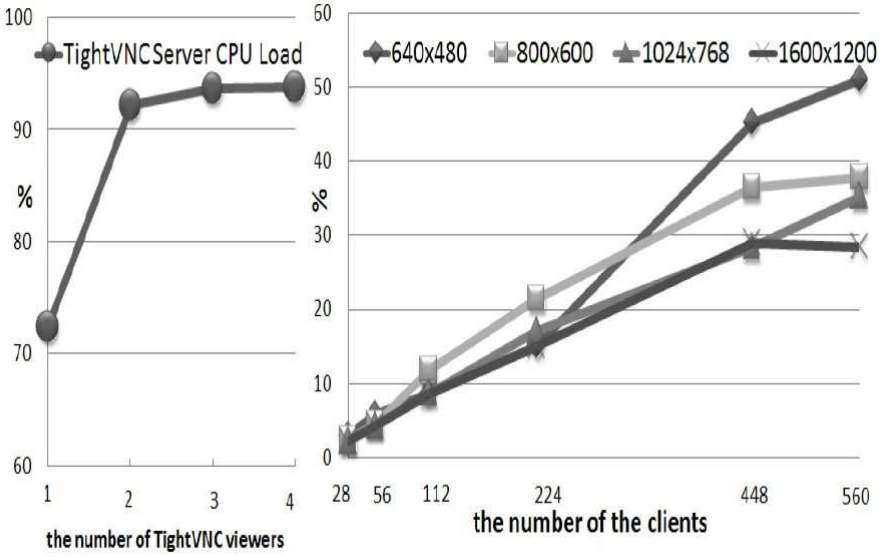

Figure 3. CPU Load on the server

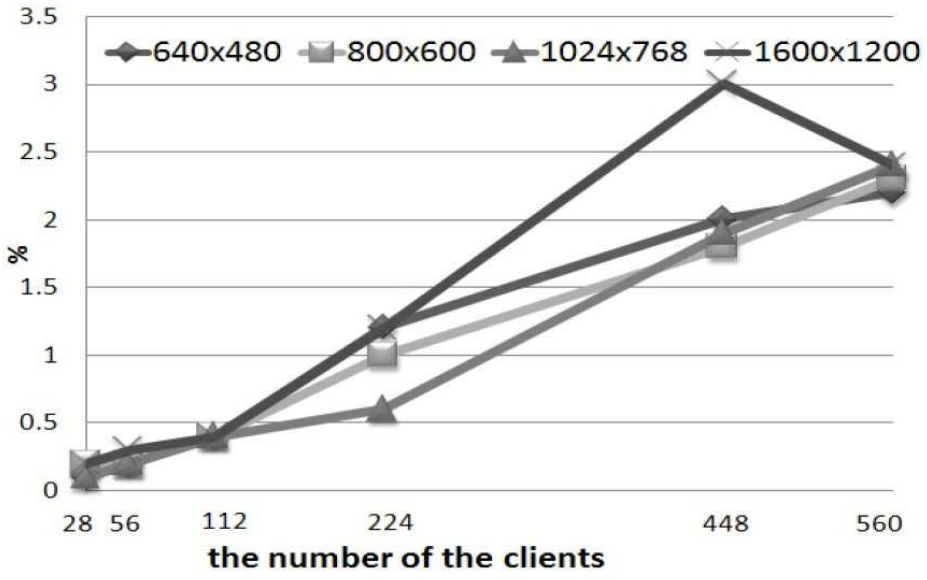

Figure 4. Memory Load on the server

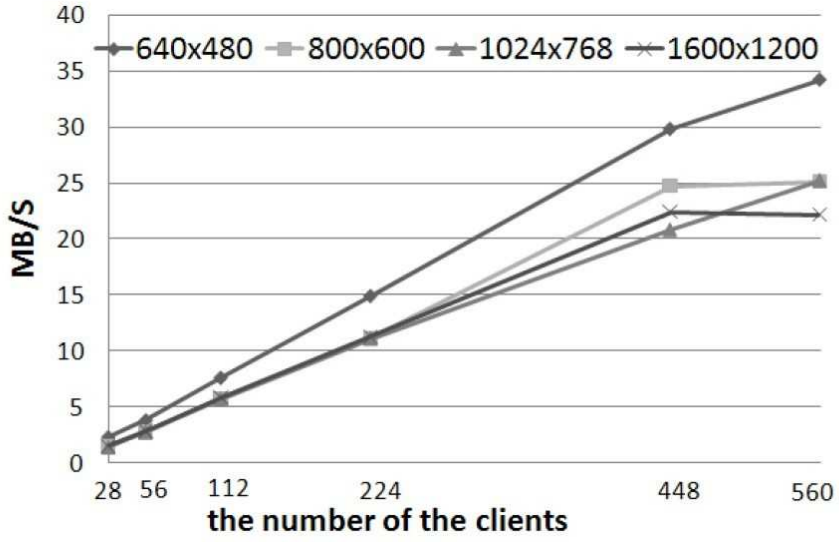

Figure 5. Network bandwidth usage

CONCLUSION

We have documented through a prototype and experiments with it, the performance characteristics of using live streaming medias to share display content. The obvious advantage of the architecture is to utilize media players as consumers of live display content since media players are supported by most devices and computers. The live streaming producer captures stream sources without modifications for the target. The MultiStream system scales significantly. It uses lower CPU and bandwidth, while supporting at least an order of magnitude more clients. In addition, the MultiStream architecture supports scalability as several streaming servers can be used. The drawback is that the encoding of the live media will impact the CPU load of the computer running the encoding software. This will impact the performance of applications on the computer.

REFERENCES

[1] G. Humphreys, M. Houston, R. Ng, R. Frank, S. Ahern, P. D. Kirchner, and J. T. Klosowski. Chromium: A stream processing framework for interactive rendering on clusters. SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques, July **2002**, pages 693–702.
[2] K. A. Perrine and D. R. Jones. Parallel graphics and interactivity with the scaleable graphics engine. Proceedings of the 2001 ACM/IEEE conference on Supercomputing, November **2001**, pages 5–5.
[3] R. Singh, B. Jeong, L. Renambot, A. Johnson, and J. Leigh. Teravision: a distributed, scalable, high resolution graphics streaming system..Proceedings of the **2004** IEEE International Conference on Cluster Computing, September **2004,**pages 391–400.
[4] B. Jeong, L. Renambot, R. Jagodic, R. Singh, J. Aguilera, A. Johnson, and J. Leigh. High-performance dynamic graphics streaming for scalable adaptive graphics environment. Proceedings of the **2006** ACM/IEEE conference on Supercomputing SC '06, November **2006,** page 24.
[5] Richardson, Q. Stafford-Fraser, K. R. Wood, and A. Hopper. *IEEE Internet Computing*, January **1998**, 2(1):33–38.
[6] SGI. Opengl vizserver 3.5: Application-transparent remote interactive visualization and collaboration. http://www.sgi.com.