Manageable and Extensible Video Streaming Systems for On-Line Monitoring of Remote Laboratory Experiments

doi:10.3991/ijoe.v5i3.898

Yuan-Cheng Lai[1] and Jian-Wei Lin [2]
[1] National Taiwan University of Science and Technology, Taipei, Taiwan
[2] Ching Yun University, Taoyuan, Taiwan

***Abstract*—To enable clients to view real-time video of the involved instruments during a remote experiment, two real-time video streaming systems are devised. One is for the remote experiments which instruments locate in one geographic spot and the other is for those which instruments scatter over different places. By means of running concurrent streaming processes at a server, multiple instruments can be monitored simultaneously by different clients. The proposed systems possess excellent extensibility, that is, the systems can easily add new digital cameras for instruments without modifying any software. Also they are well-manageable, meaning that an administrator can conveniently adjust the quality of the real-time video depending on system load and visual requirements. Finally, some evaluation concerning CPU utilization and bandwidth consumption of the systems have been evaluated to verify the effectiveness of the proposed solutions.**

***Index Terms*—COM Component; Electronic Instrumentation; .NET Remoting; Remote Control; Video Streaming.**

I. INTRODUCTION

Due to the popularity of computer-controlled instrumentation and the wide diffusion of networks, remotely conducting laboratory experiments or remotely controlling instrumentation via the Internet has been realized extensively [1-20]. Such remotely-controlled applications have been popularly applied in both academic and industrial areas. Through virtual instrument control panels, students and engineers can remotely practice and control real apparatus at any time and from any location.

The applications of remote laboratory experiments play an important role in distance learning since it can provide students to learn instrument control and operation in a guided or self-explanatory way. Engineers also have chances to test new electronic prototypes or products at distance for verifying the compliance to parameters regulated by standards and norms.

The majority of remote laboratory experiments published so far in the literature focus on describing specific experiment setups, the scenario of system design, and the lifelike user interface of a real instrument. However, in order to let users have better realistic feeling and more consciousness of the current conditions of instruments, the best way is showing a live video of the real display panel of instruments. In contrast to other tools that send data and plot results in a scope-like chart using graphic resources at the client side, the real-time video is a powerful auxiliary tool to give users better percept of the current experimental status and operation of instruments as if they are in front of the instruments.

A handful of works of monitoring remote instrumentation with live video have been presented. Ko et al presented a Web-based virtual laboratory on an oscilloscope experiment, using only one real-time video capturing of the actual oscilloscope display [1]. A Web-based virtual laboratory on radiofrequency (RF) and baseband video measurement using two real-time videos which individually capture the involved two instruments was also proposed [2].

Ranaldo presented a real-time video system, enabling clients to monitor the instruments of the remote laboratory distributed on a geographical network [3]. He used an instrument-control server equipped with a USB webcam which only delivers one video stream for an instrument at a time. Thus, the server, which can be regarded as the streaming server, is also in charge of encoding and streaming processes. There is a central server acting as a dispatcher responsible for redirecting encoded video content to the clients. However, to monitor more instruments under this structure, more streaming servers are required in order to capture more instrumental videos. To avoid this problem, Ranaldo further used an off-the-shelf IP camera with built-in MPEG-4 encoding and streaming capability to replace the streaming server [4]. Consequently, the main benefit of this method is that clients can adjust the characteristics of video streams according to current available bandwidth, since the used IP camera provides the capability of adjustment.

This paper presents a video streaming solution for monitoring remote experiments that facilitate remote clients to better understand real instrumental status or experimental activities. Although we also adopt a computer as a streaming server as Ranaldo's first method, noticeably, our proposed design allows the server to be able to handle multiple concurrent encoding and streaming processes and each stream can have its distinguished video characteristics, such as video codec, frame rate, video size. Thus, multiple instruments can be monitored simultaneously by remote clients as long as the corresponding digital cameras are connected with the server. The proposed solution possesses excellent extensibility, that is, it can easily add new video devices without modifying any software. Also the live video are easily controlled and managed by an administrator in accordance with system load and visual requirements.

Based on the types of applied environments, two systems using the video streaming solution are individually developed. One is for the remote experiments whose apparatus centrally locate in one place. The other is for the remote experiments whose apparatus scatter over different places. Finally, after the two systems have been developed, some investigations concerning CPU utilization and bandwidth consumption have been manifested to evaluate the effectiveness and usability of the both systems.

Next section will briefly introduce the two general types of remote laboratory experiments. Section III and IV will elaborate the proposed design of the both video stream systems. Also their advantages are stated and some typical GUIs are exhibited. The testing results of system performance are reported in section V and the conclusion is given in section VI.

II. REMOTE LABORATORY EXPERIMENT

In general, the involved instruments of the remote laboratory experiment are either concentrated in one geographic place or scattered over different places. For brief, herein we call the former as a central-location experiment (CLE) and the latter as a distributed-location experiment (DLE).

The works belonging to CLE mostly offers remote control via a portal server, as shown in Fig. 1 [1-2], [5-13]. The server also acts as a device controller accessing the physical devices (e.g. electronic instrumentation) via communication bus such as GPIB, USB, RS-232. Each prominent device should be equipped with one digital camera to exhibit the live video of real panel of its corresponding device.

The works belonging to DLE mostly consists of a portal server and several Instrument Control Servers (ICSs), as shown in Fig. 2 [14-20]. An ICS connecting with several nearby apparatuses can be regard as a branch-experiment platform. The portal server acts as a dispatcher that directs users to the desired platform to control instruments scattered over a wide area network. Thus, a user can remotely and transparently perform far distributed experiments on accessible instruments.

Most of the previous researches elaborate on investigating distributed technologies between the portal server and an ICS. These distributed technologies include Virtual Instrument Transfer Protocol [15], CORBA [16], Web Services [18] and Microsoft .NET Remoting [19].

For the two types, CLE and DLE, we individually develop one real-time video system, allowing users to monitor all the involved instruments of the experiment. The inner software designs and some interfaces of the two systems will be elaborated in the next two sections. The system for CLE is developed by means of the component-based technology, Microsoft COM; while the one for DLE is based on the object-oriented distributed technology, Microsoft .NET Remoting.

III. THE VIDEO STREEAMING SYSTEM FOR CLE

A. The Software Structure

For video streaming on CLE, we propose the following software structure which comprises two elements, Video Stream Component and Web Application, as shown in Fig. 3. The former is a Microsoft COM component developed in Visual C++ while the latter is implemented by HTML, JavaScript, and ASP.NET. Under this structure, a computer simultaneously plays two roles: Portal Web Server and Video Control Server (VCS).

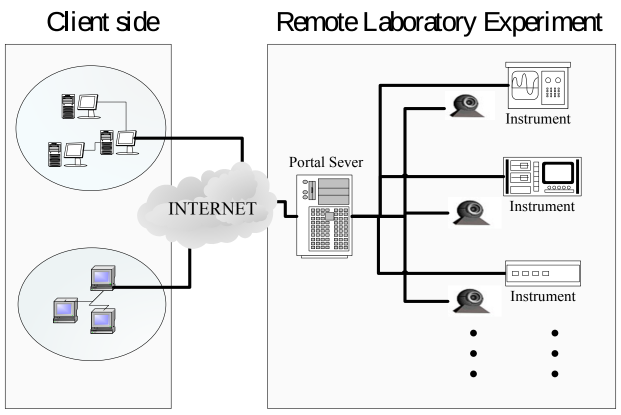


Figure 1. The typical hardware architecture of CLE

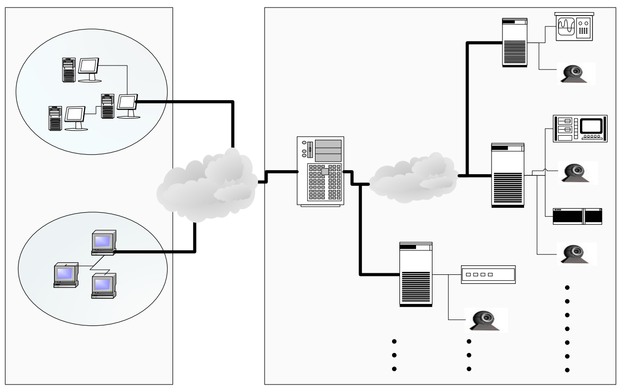


Figure 2. The typical hardware architecture of DLE

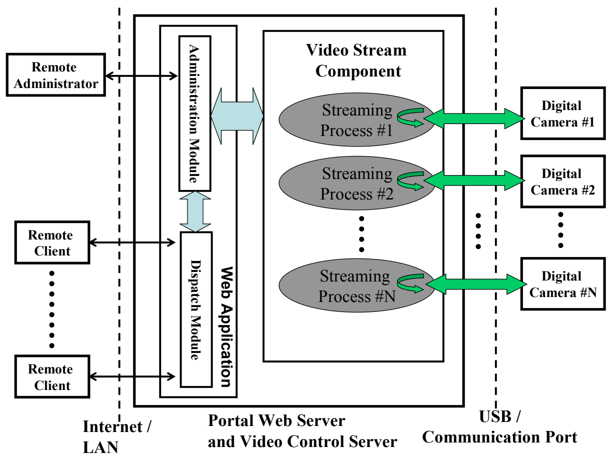


Figure 3. The proposed software structure for CLE

The Video Stream Component elaborates on constructing and distributing video streaming for electronic instrumentation. Thus there are several important works, mainly including detecting all connected video acquisition devices (e.g. USB camera, IP camera, or capture card), capturing video content from the devices, encoding (compressing) the video contents with a proper encoder (e.g., MPEG4, H. 263, and Window Media Video), and sending the compressed video content over the Internet.

To provide individual live video for each instrument, the system has to create individual streaming process for viewing each instrument. Also an extra data structure for each stream process is declared to store its own encoding profile, which characterizes a video stream. A encoding profile includes video device name, network protocol and port number, encoding mode, e.g. Constant Bit Rate (CBR) or Variable Bit Rate (VBR), encoder, frame size, frame rate, CBR output bit rate, and VBR quality level. Accordingly, each streaming process will capture an instrument video, encode it with its own profile, and then deliver it through used network protocol.

This component is intentionally developed to providing a uniform interface (API) and call flow in order to facilitate the development of Web Application. As a Web application launches, it calls the Initialization() function, which detects and returns all video acquisition devices attached to the server. In the meantime, the individual memory space is dynamically allocated for each attached video device. To create a new stream process, calling Register() with the desired video acquisition device and desired encoding profile will register this new stream process with these provided information in the COM component. A corresponding *handle* of the stream process is then returned. A handle represents the registered information of a stream process, and is also used as an indicator for the consequent processing. Then, calling Operation() with the corresponding handle will launch the stream process to encode and deliver the stream over Internet. Calling Unregister() will stop a stream process and further release the corresponding registration and handle for recycling. As the application terminates, it calls Close(), which disconnects all video devices and frees the allocated memory space.

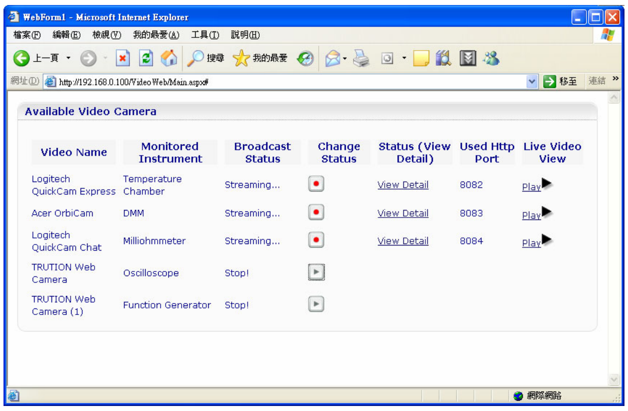
When several stream processes are employed in sequence, Register(), Operation(), and Unregister() functions are called iteratively. Accordingly, such design ensures each stream process can run independently. Moreover, it enables launching a new stream process dynamically for a new apparatus without modifying any code in Video Stream Component. The only thing to do is simply adding a new video device (i.e. plugging-in a USB webcam) for this new apparatus. Thus, the developed system is well-extensible.

In practical, we built the Video Stream Component with the Microsoft Windows Media Encoder Software Development Kit (SDK), a multimedia framework for media creation and distribution within Microsoft Windows [21]. The SDK provides useful API to facilitate encoding and delivering video media.

The Web Application contains two parts: Dispatch Module and Administration Module. The former directs a user to receive a streaming video of an instrument. The latter offers the system administrator to control and manage live video services depending on visual requirements and system load. For example, the administrator can enlarge the frame size or upgrade video quality for a clearer view of an instrument's display panel.

B. *Operation and Some Interfaces*

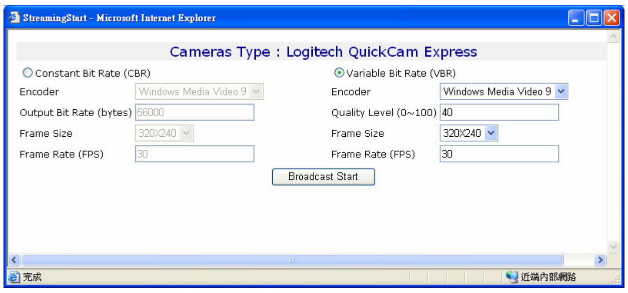
As shown in Fig. 4, the main administrative page lists all attached video devices for managing the video services. The remote administrator can launch or stop a streaming video, look the statistics of a live stream, and preview a stream.


WebForm1 - Microsoft Internet Explorer
檔案(F) 編輯(E) 檢視(V) 我的最愛(A) 工具(T) 說明(H)
上一頁 搜尋 我的最愛
網址(D) http://192.168.0.100/VideoWeb/Main.aspx#
移至 連結
網際網路

Available Video Camera

Video Name	Monitored Instrument	Broadcast Status	Change Status	Status (View Detail)	Used Http Port	Live Video View
Logitech QuickCam Express	Temperature Chamber	Streaming...	●	View Detail	8082	Play▶
Acer OrbiCam	DMM	Streaming...	●	View Detail	8083	Play▶
Logitech QuickCam Chat	Milliohmmeter	Streaming...	●	View Detail	8084	Play▶
TRUTION Web Camera	Oscilloscope	Stop!	▶			
TRUTION Web Camera (1)	Function Generator	Stop!	▶			

Figure 4. The main administrative Web page for CLE



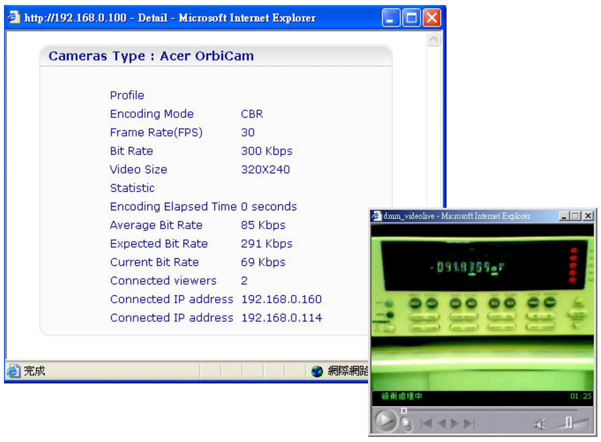
Cameras Type : Logitech QuickCam Express

○ Constant Bit Rate (CBR)		◉ Variable Bit Rate (VBR)	
Encoder	Windows Media Video 9	Encoder	Windows Media Video 9
Output Bit Rate (bytes)	56000	Quality Level (0~100)	40
Frame Size	320X240	Frame Size	320X240
Frame Rate (FPS)	30	Frame Rate (FPS)	30

Figure 5. The setup of encoding profile

Before an attempt to launch a video stream associated to an instrument, an encoding profile (e.g. CBR or VBR mode) has to be set. Thus, when pressing a launching button, i.e., an gray arrow icon below the Change Status column in Fig. 4, a frame for setting the encoding profile pops up. As shown in Fig. 5, encoder, output bit rate, frame size, and frame rate need to be settled down in a CBR mode, while encoder, quality level ranging from 0 to 100, frame size, and frame rate need to be settled down in a VBR mode. The quality level supported in SDK is used to represent the quality of viewing a VBR video. Thus, all clients receiving video from the same stream will view the identical video (e.g. the same video quality and frame size).

Fig. 6 illustrates the detailed status and the statistics of a live stream (e.g. current bit rate, the number of connected viewers, and their source IP addresses), and the video preview of an instrument.


http://192.168.0.100 - Detail - Microsoft Internet Explorer
完成
網際網路
dmm_videolive - Microsoft Internet Explorer
緩衝處理中

Cameras Type : Acer OrbiCam

Profile	
Encoding Mode	CBR
Frame Rate(FPS)	30
Bit Rate	300 Kbps
Video Size	320X240
Statistic	
Encoding Elapsed Time	0 seconds
Average Bit Rate	85 Kbps
Expected Bit Rate	291 Kbps
Current Bit Rate	69 Kbps
Connected viewers	2
Connected IP address	192.168.0.160
Connected IP address	192.168.0.114

Figure 6. The detailed statistics and preview of a streaming video

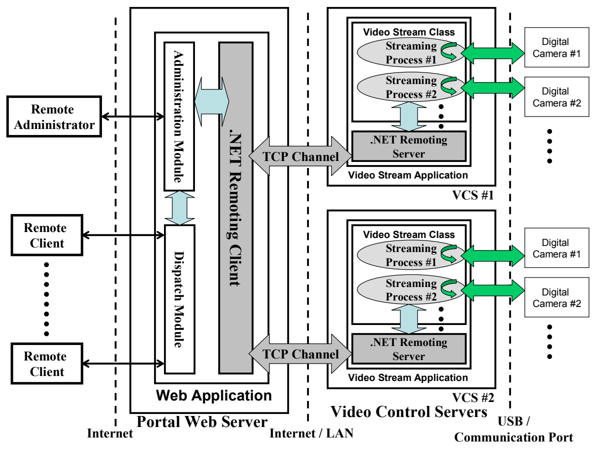


Figure 7. The purposed software architecture for DLE

IV. THE VIDEO STREAMING SYSTEM FOR DLE

A. *The Software Structure*

For DLE, we propose a software structure, which is also composed of two major parts: Video Stream Application and Web Application, as shown in Fig. 7. The former is a Microsoft .NET Application developed in Visual Studio .NET while the latter is implemented with HTML, JavaScript, and ASP.NET.

Differing with CLE, for applying to the distributed environment, Web Application is located in the Portal Web Server and one Video Stream Application is deployed in each Video Control Server (VCS). We adopted .NET Remoting as the distributed technique between Web Application and Video Stream Application. NET Remoting provides a powerful way for objects (or applications) located in different computers to communicate with each other. It also provides programmers with an object-oriented model for distributed computing as close as possible to programming with ordinary local objects. Remote objects are accessed through .NET channels, which are used to convey messages to and from remote objects. All called methods with their associated parameters are passed to remote objects and afterward the results of their execution return via these channels.

A .NET Remoting client and a .NET Remoting server are involved within the Web Application and Video Stream Application respectively for establishing a .NET channel. Via established .NET channels, the Web Application (.NET Remoting client) is able to access the objects provided by the remote Video Stream Applications (.NET Remoting servers).

The benefits of adopting .NET Remoting are: (1) The Web server can continuously interact with remote video objects deployed in VCSs, as if they were local objects. This feature hides the complexity of a VCS implementing a video object. (2) This solution possesses excellent extensibility, which can allow users to easily expand newly available VCSs and video devices. Actually, in our previous study [19], the .NET Remoting technique has revealed its superiority at developing distributed instrument control system.

A Video Stream Application consists of two classes. One class is aiming at constructing the .NET Remoting server on a VCS, intending for accepting connection from the .NET Remoting client. This class adopts a server activation mode [22] and activates only one singleton object instance for every connected video device. A singleton video object instance in a VCS acts as a listener to handle all requests from the Web server. The other class, namely Video Stream Class, is aiming at wrapping the functions related to video streaming control on a VCS, intending for providing the interfaces for the remote Web Application accessing through .NET channel. Consequently, the digital cameras and video stream services in different VCSs spreading over the Internet/LAN can be uniformly utilized by the Web Application with calling these interfaces via .NET channels. Similarly, the inner implementation of this Class resembles the designs inside the Video Stream COM in CLE. That is, it offers the same functions (i.e. APIs): Initialization(), Register(), Operation(), Unregister(), and Close() for the development of Web Application and therefore ensures that each stream process on a VCS can run independently.

The Web Application also contains two parts: Dispatch Module and Administration Module. Differing from CLE, the former redirects the users to receive the streaming videos stemming from different VCSs. The latter allows the system administrator to transparently manage all scattered streaming video.

In order to establish .NET channel, a VCS has to launch its Video Stream Application first so that a .NET Remoting server can also be launched. Then the .NET Remoting client of the Web Application is able to connect the .NET Remoting server according to the VCS's IP address and its listening TCP port. At this moment, the establishment of a .NET channel between two sides is completed. Through the channel, the Web Application can further create a corresponding remote Video Stream object to utilize the provided APIs on that VCS.

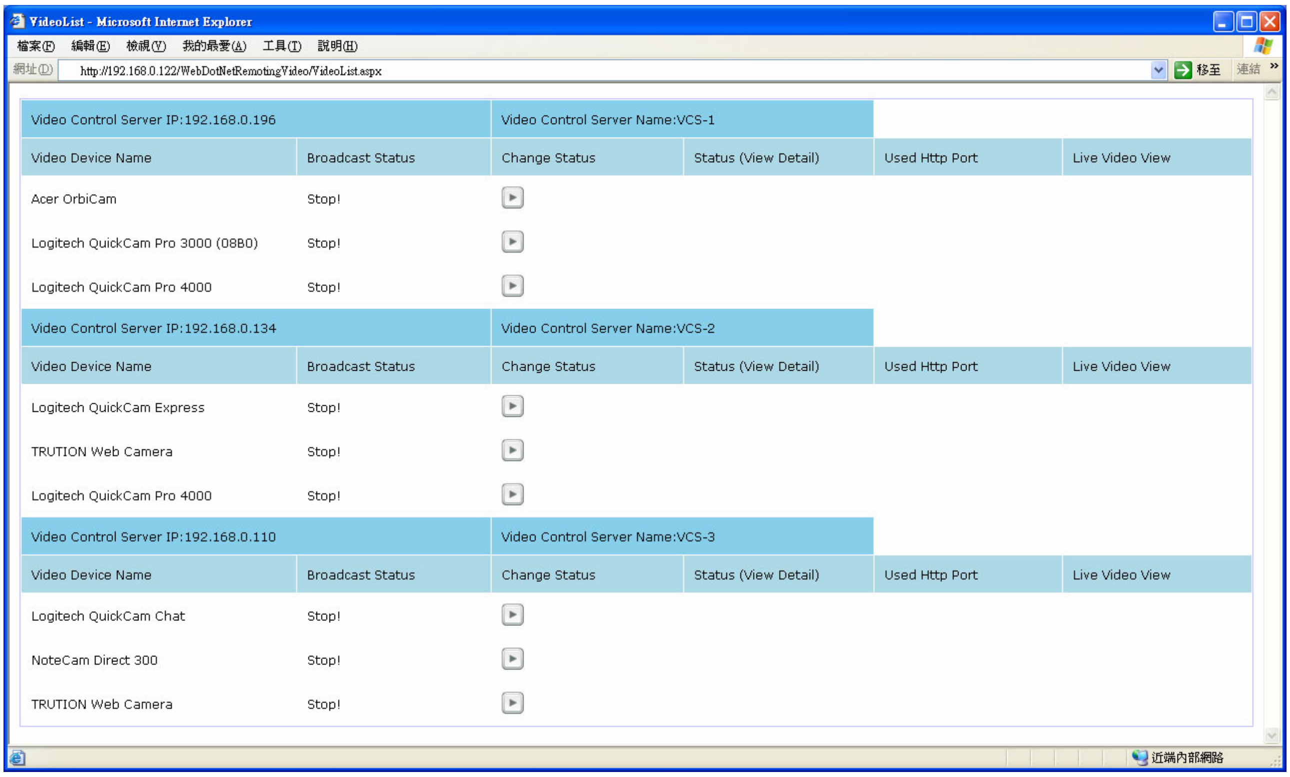
Such distributed design is also well-extendable because the two reasons: (1) a new VCS can be integrated immediately into this system after installing the Video Stream Software ;(2) an existing VCS can dynamically launch a new live video for a new apparatus by simply adding a new video device (e.g. plugging-in a USB webcam). Both above two conditions can be achieved easily without modifying any code.

B. *Operation And Some Interfaces*

The main administrative Web page of our system is shown in Fig 8, connecting three VCSs, called VCS1, VCS2, and VCS3, via LAN. Each VCS equips with three USB webcams. However, the offered operations for the administrator in this system are identical to the prior CLE system, including starting or stopping a stream, looking the statistics of a live stream, and previewing a stream which can refer to Fig. 5 and Fig.6.

V. SYSTEM PERFORMANCE EVALUATION

Multiple concurrent encoding and streaming processes running at a server certainly demand significant system resources and further impose this server loading. Note that under our proposed structures, such a streaming server precisely refers to the Web server in the CLE and a VCS in DLE, that is, where the streaming processes physically run. Thus, performance evaluations concerning CPU utilization and bandwidth consumption with respect to a streaming server are measured. For evaluation, a computer equipped with Intel Core Duo 1.66 GHz, 1GB RAM, and Window XP Pro, is acted as this streaming server.


VideoList - Microsoft Internet Explorer
檔案(F) 編輯(E) 檢視(V) 我的最愛(A) 工具(T) 說明(H)
網址(D)
移至
連結
近端內部網路

http://192.168.0.122/WebDotNetRemotingVideo/VideoList.aspx

Video Control Server IP:192.168.0.196		Video Control Server Name:VCS-1			
Video Device Name	Broadcast Status	Change Status	Status (View Detail)	Used Http Port	Live Video View
Acer OrbiCam	Stop!	▶			
Logitech QuickCam Pro 3000 (08B0)	Stop!	▶			
Logitech QuickCam Pro 4000	Stop!	▶			
Video Control Server IP:192.168.0.134		Video Control Server Name:VCS-2			
Video Device Name	Broadcast Status	Change Status	Status (View Detail)	Used Http Port	Live Video View
Logitech QuickCam Express	Stop!	▶			
TRUTION Web Camera	Stop!	▶			
Logitech QuickCam Pro 4000	Stop!	▶			
Video Control Server IP:192.168.0.110		Video Control Server Name:VCS-3			
Video Device Name	Broadcast Status	Change Status	Status (View Detail)	Used Http Port	Live Video View
Logitech QuickCam Chat	Stop!	▶			
NoteCam Direct 300	Stop!	▶			
TRUTION Web Camera	Stop!	▶			

Figure 8. The main administrative Web page for DLE

Among these investigations, two different encoding modes, CBR and VBR, are adopted in order to individually analyze their impacts. In a VBR mode, a desired quality level, ranging from 0 to 100, is required to specify. Under a given quality level, the output bit rate after encoding fluctuates depending on the complexity of the stream. That is, high motion will result in a high bit rate and slow motion will result in a lower bit rate. The benefit of quality-based VBR encoding is that the video quality is consistent among all streams. The drawback is that bandwidth requirement of the encoded content can not be known.

In a CBR mode, an output bit rate is required to specify. Accordingly, the video quality is not constant for content which complexity varies. The benefit of CBR encoding is that the output bit rate of the content is known before encoding, and thus the bandwidth requirement of the encoded content can be predicted in advance.

A. *CPU Utilization:*

In this evaluation, five different USB webcams are attached to the computer that launches five distinct stream processes. Additionally, our developed program is installed in the server to acquire CPU utilization mainly caused by these running processes.

The relationship between CPU utilization and the number of concurrent stream processes in CBR and VBR modes are shown in Fig 9 and Fig. 10, respectively. The concurrent stream processes is varied from 1 to 5. The output bit rate in the CBR mode is varied in the range [56, 128, 256, 512, 1024] Kbps while the quality level in the VBR mode is varied in the range [20, 40, 60, 80, 100]. The rest parameters are set as follows:

- Frame Rate: 30 frames per second
- Frame size: 320x240
- Encoder: Microsoft Media Video 9

Figure 9 manifests that the CPU utilization has positive correlation with the number of activated processes and the CBR output bit rate. Similarly, Figure 10 also exhibits that the CPU utilization is raised when the number of processes and the VBR quality level are increased. However, the trend of Fig. 10 varies more dramatically, comparing to Fig. 9, since keeping consistent highest quality of all streams will heavily consume CPU computing. For example, the CPU utilization of three concurrent processes already reaches almost 100% for the VBR quality level of 100.

B. *Bandwidth Consumption:*

This evaluation aims at observing the relationship between bandwidth consumption and the number of viewers under the condition where only one streaming process runs. The results of CBR and VBR modes are shown in Fig 11 and Fig. 12, respectively. The number of concurrent viewers is varied from 1 to 5. The output bit rate in the CBR mode, the quality level in the VBR mode, and the rest parameters are same to the previous evaluation.

Figure 11 illustrates that bandwidth consumption is nearly proportional to the number of concurrent viewers and CBR output bit rate. Similarly, Figure 12 also exhibits that the bandwidth consumption is raised when the number of viewers and the VBR quality level are increased. However, the trend of Fig. 12 varies more dramatically, comparing to Fig. 11, since sustaining consistent highest quality of all streams will output the vast encoded bit rate, so that considerable bandwidth are required.

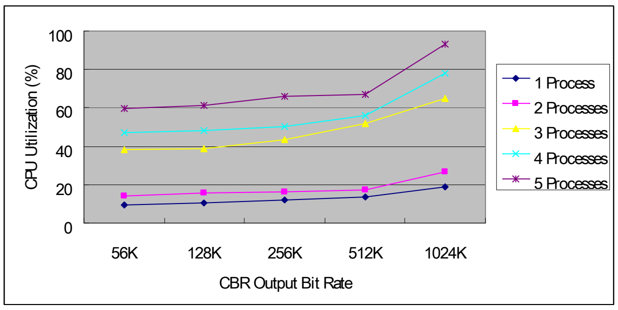


Figure 9. CPU utilization vs. the number of concurrent stream processes in CBR mode

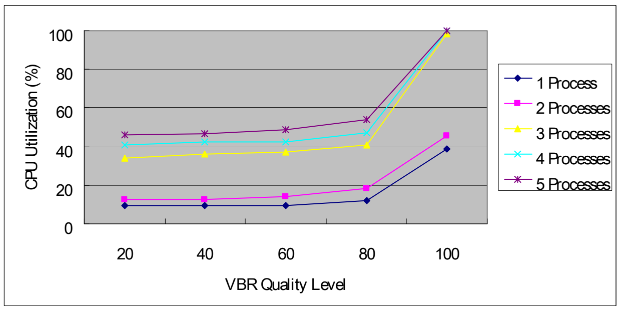


Figure 10. CPU utilization vs. the number of concurrent stream processes in VBR mode

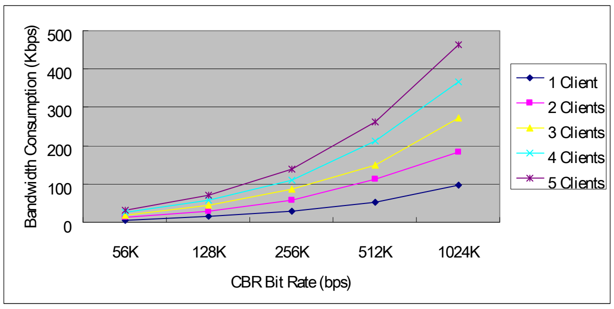


Figure 11. Bandwidth consumption vs. the number of viewers in CBR mode

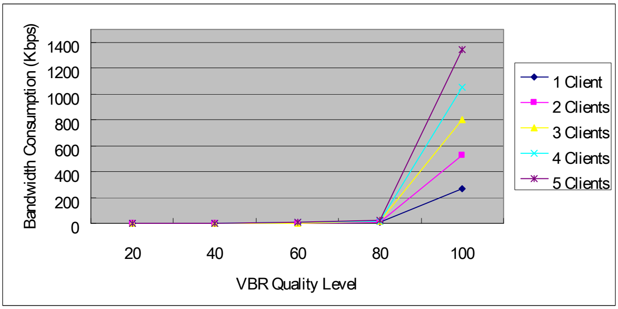


Figure 12. Bandwidth consumption vs. the number of viewers in VBR mode

In addition, based on our tests, bandwidth requirement can be predicted when several stream processes using CBR mode are running concurrently. This is reasonable because these stream processes output constant bit rate. For example, if one streaming process encoded with the CBR mode output 30Kbps to 3 receivers (viewers), leading totally 90Kbps consumption, Such three streaming processes running simultaneously in a server will occupy nearly 270 Kbps bandwidth.

Since the video playing may be slightly delayed owing to the overhead of media handling and network transmission, the experiments regarding waiting time that a viewer has to wait before playing are also conducted herein. However, our testing results exhibits that network conditions, such as available bandwidth and congestion, have much more impact on viewer's waiting time, compared to the number of concurrent stream processes and the CBR output bit rate (or the VBR quality level).

VI. CONCLUSION

This paper presents video streaming solutions for monitoring remote experiments to facilitate user's better understanding of real instrumental status or experimental activities. The proposed two systems deal with these remote experiments whose apparatus either locate in one place or scatter over different places.

Among them, noticeably, a streaming server is capable of running multiple concurrent streaming processes so that multiple instruments can be monitored simultaneously by multiple clients. Each stream has its own video characteristics so that the viewers requesting the same video stream will have the identical video, i.e. the same video quality. The proposed systems possess excellent extensibility since the systems can easily add new digital cameras for instruments without modifying any software. In addition, the live video services of both systems can be easily managed via administrative Web pages since our systems provide friendly interfaces with a transparent manner.

Finally, some evaluation concerning CPU utilization and bandwidth consumption have been conducted. Overall, the CPU utilization has positive correlation with the number of concurrent processes and CBR output bit rate (VBR quality level). The bandwidth consumption grows linearly with the number of concurrent viewers when using CBR mode. However, using VBR quality level has more dramatic influences on the bandwidth consumption. With respect to client's waiting time, these factors are negligible when compared to changeful network conditions.

REFERENCES

[1] C. C. Ko, B. M. Chen, S. H. Chen, V. Ramakrishnan, R. Chen, S. Y. Hu, and Y. Zhuang, "A Large-scale Web-based Virtual Oscilloscope Laboratory Experiment," IEE Engineering Science and Education Journal, Vol.9, No.2, pp.69-76, April 2000.

[2] A. A. P. Pohl, G. A. Michelon, S. M. Vidal, H. S. Filho, and K. V. O. Fonseca, "Remote RF and Baseband Video Measurement Laboratory Based Upon Open-Code Software," IEEE Transactions on Instrumentation And Measurement, Vol.57, No.3, pp.556-564, Mar. 2008.

[3] N. Ranaldo, S. Rapuano, M. Riccio, and F. Zoino, "On The Use of Video-Streaming Technologies for Remote Monitoring of Instrumentation," IEEE Instrumentation and Measurement Technology Conference, Italy, pp.861-866, April 2006.

[4] N. Ranaldo, S. Rapuano, M. Riccio, and F. Zoino," Remote Control and Video Capturing of Electronic Instrumentation for Distance Learning," IEEE Transactions on Instrumentation And Measurement, vol.56, no.4, pp.1419-1428, Aug. 2007

[5] A. Ferrero, and S. Salicone, "Towards a Hypertext of Electric Measurement: Different Approaches for an On-line, Remote, Didactic Lab," IEEE Transactions on Instrumentation And Measurement, vol.56, no.1, pp.89-94, Feb. 2007.

[6] A. Bagnasco, G. Parodi, D. Ponta, and A. M. Scapolla, "A modular and extensible remote electronic laboratory," iJOE Int. J. Online Eng., vol. 1, no. 1, 2005.

[7] R. Safaric, M. Truntic, D. Hercog, and G. Pacnik, "Control and robotics remote laboratory for engineering education," iJOE Int. J. Online Eng., vol. 1, no. 1, 2005.

[8] J. W. Lin, Y. C. Lai, H. H. Peng, and W. Wu, "A Web-based Environmental Experiment System Using Microsoft .NET and COM," IEEE Instrumentation and Measurement Technology Conference, Italy, pp.1558-1563, May 2006

[9] J. Guo, D. Kettler, M. Al-Dahhan, "A chemical engineering laboratory over distributed control and measurement systems," Computer Applications in Engineering Education, Vol. 15, NO.2, 2007, pp.174-184

[10] L. Wang, P. Orban, A. Cunningham, and S. Lang, "Remote real-time CNC machining for web-based manufacturing," Robotics and Computer-Integrated Manufacturing, Vol.20, NO.6, 2004, pp.563-571.

[11] H. J. Kim, W. S. Chu, S. H. Ahn, D. S. Kim, and C. S. Jun, " Web-Based Design and Manufacturing Systems for Micromachining: Comparison of Architecture and Usability," Computer Applications in Engineering Education, Vol.14, No.3, 2006, pp. 169-177.

[12] H. C. Lin," A Remote Monitoring and Control-Based Precise Multilocation Riveting System," Computer Applications in Engineering Education, Vol.13, No.4, 2005, pp. 316-323.

[13] H. C. Lin," Web-Based Remote Online Maximum Wind Power Monitoring and Control System," Computer Applications in Engineering Education, Vol.15, NO. 2, 2007, pp. 155-165.

[14] L. Benetazzo, M. Bertocco, F. Ferraris, A. Ferrero, C. Offelli, M. Parvis, and V. Piuri, "A web-based distributed virtual laboratory," IEEE Transactions on Instrumentation And Measurement, vol. 49, pp. 349–356, Apr. 2000.

[15] M. Chirico, A. M. Scapolla, and A. Bagnasco, "A New and Open Model to Share Laboratories on the Internet," IEEE Transactions on Instrumentation And Measurement, vol.54, no.3, pp.1111-1117, June 2005.

[16] G. Canfora, P. Daponte, and S. Rapuano, "Remotely accessible laboratory for electronic measurement teaching," Comput. Stand. Interfaces, vol. 26, no. 6, pp. 489–499, 2004.

[17] R. Kumar, K. Sridharan, and K. Srinivasan, "The Design and Development of a Web-Based Data Acquisition System," IEEE Transactions on Instrumentation And Measurement, Vol.51, No.3, 2002, pp.427-432.

[18] Y. Yan, Y. Liang, X. Du, H. S. Hasane, and A. Ghorbani, "Put Labs Online with Web Services", IEEE IT Professional, pp.37-44, Mar/Apr 2006.

[19] J. W. Lin, Y. C. Lai, A. Lin, H. H. Peng, and Y. C. Szu, "An Object-Oriented Approach for Developing Distributed Measurement Laboratory Using Microsoft .NET technology," IEEE Instrumentation and Measurement Technology Conference, Poland, pp.1-5, May 2007.

[20] A. Sand, H. Slinde, and T. A. Fjeldly, "A Secure Approach to Distributed Internet-Enabled Metrology," IEEE Transactions on Instrumentation And Measurement, vol.56, no.5, pp.1979-1985, Oct. 2007.

[21] Microsoft Inc., "Windows Media Encoder 9 Series SDK Help," http://www.microsoft.com.

[22] http://msdn.microsoft.com/webservices/remoting/default.aspx?pull=/library/en-us/dndotnet/html/hawkremoting.asp

AUTHORS

Yuan-Cheng Lai is with National Taiwan University of Science and Technology, Taipei, Taiwan (e-mail: laiyc@cs.ntust.edu.tw).

Jian-Wei Lin is with Ching Yun University, Taoyuan, Taiwan (e-mail: D9109104@mail.ntust.edu.tw).

Submitted, April, 7, 2009. Published as resubmitted by the authors on July, 2, 2009.