A High Speed Multimedia Collaboration Framework

Choon Jin NG
ViSLAB, The School of IT,
The University of Sydney
National ICT Australia
cjng@vislab.usyd.edu.au

Masahiro Takatsuka
ViSLAB, The School of IT,
The University of Sydney
National ICT Australia
masa@vislab.usyd.edu.au

Alex Krumm-Heller
CSIRO
alex.krumm-heller@csiro.au

Abstract

Remote collaboration technology allows participants to communicate and share information across network, resulting in greatly improved tasks such as collaborative decision making process and creative thinking processes. The Virtual Network Computing (VNC) is a What-You-See-Is-What-I-See (WYSIWIS) tool that is widely used to facilitate remote desktop sharing. Utilising VNC, the Service-Oriented Remote Collaboration (SORC) framework was introduced to allow variable and dynamic service insertion, hence allowing rich collaboration experience that encompasses a variety of situation. However, as computing capability increases in line with broadband infrastructure, users are now demanding more real-time services such as video conferencing and sharing. Unfortunately, the Remote Frame Buffer (RFB) protocol used by VNC is not even capable of handling a single video stream. This prohibits the realisation of a multimedia intensive WYSIWIS collaboration. This paper introduces an alternative architecture which provides high-performance multimedia transmission capability, making an intensive WYSIWIS collaboration possible.

1. Introduction

The concept of “Computer-Supported Cooperative Work” (CSCW), a widely researched field, was first defined by Irena Greif and Paul Cashman, back in 1984 [1]. There are various different forms of a computer or technology enabled support. Among those, sharing what is displayed on the remote screen is routinely requested and possibly the most frequently used form. In the field of application sharing, there are two approaches for distributing “shared” application state: (1) Centralised and (2) Replication [2]. For the first approach, the Virtual Network Computing (VNC) technology, which is essentially a remote desktop sharing technology, has been the most commonly used tool [3]. It is based on the idea of distributing a system’s screen images/frame buffers. The other approach involves replicating instances of an individual application across all clients. These instances then synchronise locally executed applications/desktops via synchronization of events. The drawback of the latter approach is that the synchronization (consistency checking) mechanism often imposes complex system architecture and implementation. In addition, it is costly to write or rewrite an application for the proprietary environment to acknowledge and process the synchronization and replicated data.

As a result, the Service Oriented Remote Collaboration (SORC) framework was proposed to provide a more global generic collaboration environment [4]. Its application is wide and can be used in many remote CSCW scenarios such as those in the sector of military, education, emergency services, and medicine.

In a previous work, the SORC framework was prototyped as the VNCProxy. However, stemming from the fact on the lack of performance in the VNC, VNCProxy makes it impossible to support real-time multimedia collaboration.

The low performance issue in VNC is due to its design catering for use in low bandwidth environment. An example of such environment is in the general application of general office productivity. If a video content is to be transmitted via VNC, the client-side will very likely not being able to view a single complete frame due to time synchronisation issue. The root of this problem is twofold: (1) The encodings used by the RFB protocol and (2) The architecture of VNC itself.

The contributions this paper provides are:

- Creation of a remote thin-client architecture capable of high frame rate transmission on multimedia contents.
- Integration of the new architecture into the SORC framework to enable true multimedia collaboration.

2. Related work

The VNC technology defines a set of Remote Frame Buffer (RFB) protocol to control another computer remotely. A VNC server is the server in which its desktop is to be shared while a VNC client is a thin-client controlling the server. Client inputs are piped to the server. Those parts of the screen that are changed are then sent to the clients as frame buffer updates, an operation based on a single graphics primitive of "putting a rectangle of pixel data at a given x, y position" [3].

The VNC technology has limited functionality which is to control a server, typically a single desktop, at anytime. The system operates on the basis of a single-user interface environment.

The concept of remote frame buffering such as VNC is not new. This function is commonly built into collaboration suite such as Skype, Adobe Connect and MSN Messenger. However, it is worth to note that in this case, the remote frame buffering is solely used as a basic point-to-point desktop sharing only. They do not encompass any possibility that a collaboration activity can be far more sophisticated than that. Performance wise, they are similar to that of VNC.

Utilising the Service Oriented Architecture (SOA) [5], the SORC architecture was introduced to provide sophisticated collaboration space on top of VNC technology. The SORC framework contains four components, namely: (1) Standard VNC desktop, (2) Standard VNC clients, (3) Collaborative services, and (4) Proxy server. Collaborative services and desktop are aggregated into a proxy server in order to provide a common workplace for participants (clients) to collaborate.

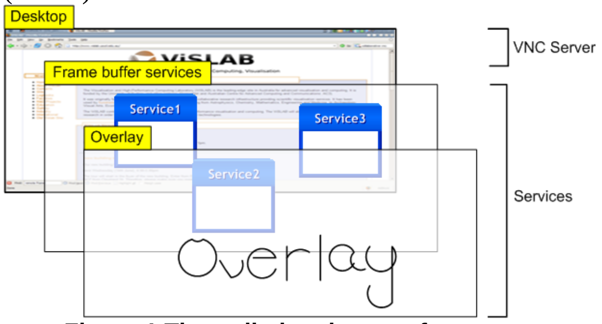


Figure 1 Three distinct layers of screen.

The advantages gained from this framework include:

- Service can be inserted easily at run-time.
- Legacy VNC support is maintained.
- True multi-user interaction is supported at service-level.
- Existing applications can be shared without any modification.

In SORC, the collaborative services are interfaced as desktop, standard application, and overlay. This is illustrated in Figure 1 which forms the VNCProxy prototype.

Currently, there are already several prototypes created to address the problem of having a multimedia collaboration over the RFB protocol. Among them are:

- THINC [6] is a thin-client remote display architecture which allows the sending of both video and raw screen content. However, its architecture design is closely linked to the X Window system, requiring interception of commands in the X video device abstraction layer. Moreover, video transmission over the line requires applications to connect to the XVideo interface. Hence, this system is not cross-platform and can have compatibility issue with programs that do not interface with XVideo. Besides, this system is complex as it needs to differentiate video from raw screen content and handle them differently.
- TurboVNC [7] is a modification from TightVNC which applies lossy compression scheme such as JPEG on transmission data. This solution is only suitable for video transmission but not desktop/application collaboration where visual elements such as texts need to have complete pixel data.

3. The Virtual Terminal architecture

The forthcoming architecture, named the Virtual Terminal (VT), is designed to address the RFB protocol limitations in a multimedia collaboration environment. It differs from existing solutions by providing:

- A much simpler system that can handle both raw screen content and video using single compression and encoding scheme.
- Scalable into many clients without penalizing screen content and video quality.
- Its protocol is cross-platform with minimal port into operating system specific API.
- Integrated into the SORC architecture, the system is transformed into a high-performance collaboration venue.

VT architecture has the following core elements:

- Scalable proxy expansion.
- Dual communication channel.
- Low communication overhead.
- Delta encoding.
- Independent of transport layer protocol.

3.1. Scalability

In terms of scalability, the VNC can only support regular desktop sharing operation with a maximum of

15 clients connected before the start of performance degradation. Furthermore, VNC cannot transmit video content smoothly even with only one client. One reason for this is the stress applied to a single server.

To address this problem, data traffics are channelled between the server and the clients through a proxy. This enables scalability in terms of the number of clients. If required, additional proxies can be added to reduce stress on the server, thus allowing more clients to interact with the server. The VT components are illustrated in Figure 2.

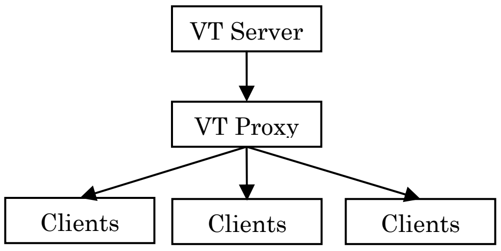


Figure 2 Virtual Terminal (VT) architecture.

3.2. Push model

With the RFB client operating based on the "pull" model [6], both the command/control stream and the frame buffer stream are highly synchronous. In order to retrieve a frame buffer update, the RFB client needs to send a request to the server to pull the update. In a low-bandwidth environment, the RFB clients have the benefit of controlling the frame buffer update rate. Essentially, participants can have enhanced collaboration experience when bandwidth is inadequate.

However, such benefit will no longer hold true in a multimedia collaboration environment. Since a video transmission often has high frame-rate, it is impractical for an RFB client to send so many requests for frame buffer updates. Not only is the transaction slow, but also too much of a communication overhead is imposed.

Due to the fact of high bandwidth consumption in multimedia collaboration, the major assumption of having a low bandwidth requirement is discounted in this architecture.

As opposed to the "pull" model, VT operates based on the "push" model. Communication overhead can be minimised as frame buffers are pushed to the clients based on the frame rate required. By doing so, clients can save the round trip time and processing time of having to send a frame buffer update request and waiting for the server to reply.

3.3. Dual channel

Another limitation in the RFB protocol is its single channel TCP communication. For this reason, the command/control traffic is intermixed with the frame buffer update traffic. This unnecessarily creates a long queue when frame buffer updates delay should be kept minimal.

Following the "push" model employed by the VT architecture, update requests from the client is eliminated. To further elaborate, the command/control channel is now solely used for sending keyboard/mouse commands and is no longer involved in controlling screen updates. Thus the command/control stream can now be fully decoupled from the frame buffer update stream, allowing it to operate in an asynchronous manner.

Taking advantage of this, a separate dual-communication channel is created for VT to further lower communication delay between the client and server. The dual-communication channel consists of the following:

- One down-stream channel for carrying frame buffer data.
- One up-stream channel for carrying command/control data.

With the dual channel communication system, frame buffer update can be retrieved as soon as possible without queuing behind the command/control data stream in a network buffer, and vice versa.

3.4. Encodings

Due to the characteristics of the RFB protocol, VNCProxy is only suitable for low-bandwidth type of collaboration such as web browsing and office applications. Its performance is influenced by several factors:

- Size of the rectangle being updated.
- Number of rectangles being updated.
- Encodings used.

Table 1. Commonly used encoding scheme for the RFB protocol.

Encodings	*Description*
CopyRect	An (x,y) position can be applied to allow only specific area to be copied.
RRE	CopyRect with run-length compression.
Hextile	Rectangles are split into 16x16 tiles. Each tile is encoded raw pixel or RRE encoding.
ZRLE	A variation of Hextile with Zlib data compression.
Tight	Uses ZLib library to compress the pixel data, but prepossesses data to maximise compression while minimising processing time. Techniques used include JPEG compression and reducing colour depth.

In the RFB protocol, each frame buffer update is encoded with one of the encodings listed in Table1 [8, 9]. Out of all the encodings listed in Table 1, the most bandwidth saving methods are the ZRLE encoding and the Tight encoding. In terms of computational complexity, the ZRLE is straight-forward, while the Tight encoding is more CPU intensive, requiring pre-processing such as colour-depth reduction.

In contrast to a full featured multimedia collaboration operating in a high-speed network, the RFB protocol encodings are designed from ground up to provide simple desktop activities over a low-bandwidth environment. If changes on the screen are large, there will be a significant number of rectangles that needs to be updated, leading to communication overhead.

Even though the RFB protocol has efficient frame buffer compression scheme, its technique of applying frame buffer update over a rectangular area can result in retransmission of the same screen data. Compression based on rectangles is less efficient since a rectangle does not necessary implies that the entire image content is new but rather only parts of it. Therefore, part of the “old image” states can still be stored in the new rectangle.

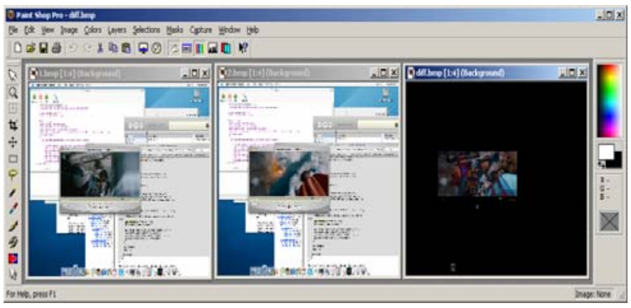


Figure 3. Delta compression: The third frame shows the difference between the first frame and the second frame.

To further enhance encoding efficiency, another aspect of the VT architecture is to make use of delta encoding. Illustrated in Figure 3, a delta encoding performs a differential on the first and second frames, producing a third one. The resulting frame is then compressed and sent to the clients. This is a more efficient compression scheme compared to the one used by the RFB protocol.

However, if only differential frames are sent throughout the entire session, then there can be possibly two problems. During a collaboration session, there can be late-joiners which have not yet retrieved the initial frame state. Moreover, since VT is designed for use regardless of its underlying transport protocol, packet loss can occur in unreliable protocol which translates into frame buffer state lost. To resolve these problems, there are two types of frames used in the VT architecture. Namely, they are the I-frame and P-frame. An I-frame is a key frame in which the entire screen content (not a differentiated frame) of the server is sent once at a defined rate. After an I-frame is sent, a series of P-frames which uses delta encodings on the screen content are sent. In this way, clients are given chance to recover to the full state of the screen at a given time interval.

3.5. Packing and transportation

The encoded data are further compressed with the zlib compression scheme, which is then packed into an AVInfoHeader structure that defines its properties such as data size, slice width and slice height. The entire packet is then wrapped into a RTP header to so that the VT protocol can be used regardless of reliable or unreliable transport protocol. The RTP (real-time transport protocol) protocol is a TCP/UDP independent transport protocol that is designed for applications sensitive to data packet delays and can tolerate data loss. In a scenario where protocol such as UDP data packets can be missing, the RTP contains timestamp and checksum to help keep track of the data packets. Figure 4 shows the final payload format of a packed data ready for transportation.

RTP Header	AVInfoHeader	Compressed data

Figure 4. Compressed frame buffer payload format.

Each data packet should not pack the entire screen content into it. Instead, the notion of slices is used with the screen broken into several horizontal slices, packed into the payload packet, and delivered to the clients. This is to ensure in the case of a packet loss, the entire screen content is not resent again. Otherwise, bandwidth utilisation would be inefficient as a small error in the packet causes an entire screen data to be discarded.

3.6. Architecture summary

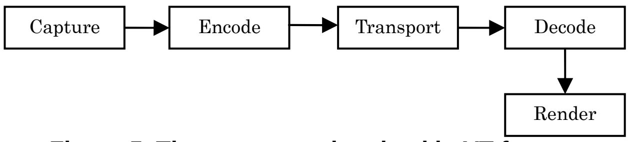


Figure 5. The processes involved in VT for capturing the screen and sending them to the clients.

Figure 5 describes the operations of VT:

- Capturing screen.
- Encoding: Making a screen differential and compressing the differential in a lossless compression such as zlib.
- Transport: Data is packed and transported through proxy for large scale distribution.
- Decoding: Data is unpacked and decoded.
- Rendering: The decoded data is then rendered onto the screen.

4. The VT-SORC architecture

With a multimedia high-speed capable infrastructure in place, the SORC can take advantage of the VT architecture for multimedia collaboration. This allows collaboration involving video production where high resolution videos are streamed across participants.

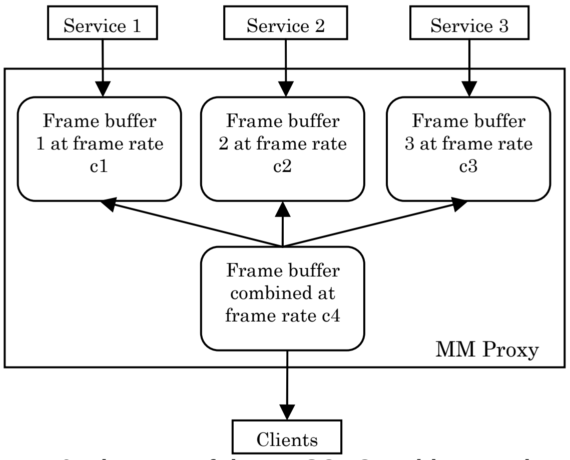
The merging of VT with the SORC architecture is called the VT-SORC architecture. Similar to the VT architecture illustrated in Figure 2, the VT-SORC architecture has its proxy server replaced by a more sophisticated one called the Multimedia Service (MM) Proxy. The primary difference is that instead of one VT server, multiple VT servers are now designated as collaborative services which are injected into the MM Proxy. Hence, the MM Proxy is a collaboration venue that plays a central role in mediating and aggregating all the services and presenting them to the clients.

The working principle behind the MM Proxy is different from VNCProxy. In the VNCProxy, the RFB protocol's "pull" model makes it the duty of the client to request frame buffer update. On the other hand, with the "push" model, services will push the frame buffers at their own respective frame rate. This actually enables each service to define its own bandwidth requirements, including the independent rate at which the proxy pushes the frame buffer to the clients. The mechanism of MM Proxy is shown in Figure 6.

As streams from all services converge at the two respective channels at the proxy, the data packets are intermix. To isolate these packets in the proxy, it is crucial for each stream originating from the services to identify themselves. Logically, each packet is identified with a unique stream ID.

The MM proxy contains a user interface (UI) manager which is in charge of frame buffer manipulation. Its purpose is to merge all service's frame buffer so participants can work on them. For each service that delivers its frame buffers to the MM proxy, a source store buffer is created in the UI Manager to store the frame buffer. An image processor will then collect the source buffers, merge and translate them into a single frame buffer for presentation. The end result is stored in a target frame buffer ready for transportation and distribution by a broadcast module located in the MM proxy.

Also built into the proxy is a signal multiplexer which decide the target service in which the commands dispatched by the clients are channeled to. In some circumstances, the commands dispatched can be directed to more than one service. The VT-SORC implementation defined a few types of services which in turn defines how the signals are routed. Beside the service type DESKTOP, STANDARD WINDOW and OVERLAY being defined in the VNCProxy, another new service type is included in the VT-SORC implementation. The new service defined is the INTERCEPTOR OVERLAY service. It intercepts all commands sent to the proxy even though the commands are targeted specifically for a certain service such as DESKTOP.



Service 1
Service 2
Service 3
Frame buffer 1 at frame rate c1
Frame buffer 2 at frame rate c2
Frame buffer 3 at frame rate c3
Frame buffer combined at frame rate c4
MM Proxy
Clients

Figure 6. The core of the VT-SORC architecture is the MM Proxy. Inside it, each individual service pumps frame buffers at rate c1, c2, c3 and c4 respectively.

To demonstrate the functionality of VT-SORC, a service called WordReplay is created. It is anticipated when an interactive teaching is in progress, participants would want to witness what each other have typed on the keyboard. For a solution, WordReplay offer the service of intercepting these keys and displaying them on the screen to all participants.

5. Results and discussion

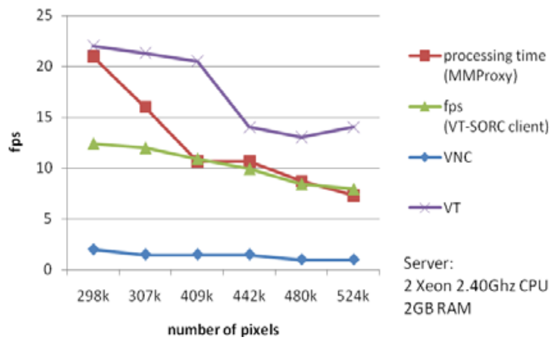


Figure 7. Performance comparison of VT, VT-SORC and VNC.

5.1. Performance of VT

VNC can only display 1 to 2 fps at best. As resolution increases, VNC cannot render a single frame properly. Each frame is rendered one quadrant at a time. Hence, the frame rate is near to null, making VNC an unsuitable medium to transport video content.

The VT architecture can achieve frame rate up to 15fps on a screen resolution of 1024x768, 22 fps on PAL (512x582) and 21 fps on VGA (640x480). This is shared across 8 clients with a full screen video playing.

5.2. Performance of VT-SORC

Figure 8 illustrates a VT client connected to the MM Proxy prototype. While the video is running, a WordReplay service intercepts keyboard commands and displays them on top of the video. Theoretically, the highest performance limit of the VT-SORC architecture cannot exceed that of the VT architecture. This is due to the heavy image processing involved at the proxy level. Compared to the standalone VT architecture, for N services, the VT-SORC proxy needs to perform N decoding and deflation, 1 encoding and compression, and image processing for frame buffer manipulation. Figure 7 shows the performance of the MM prototype. The implementation is capable of transporting full screen PAL and VGA video at the frame rate of 13 fps and 12fps respectively.

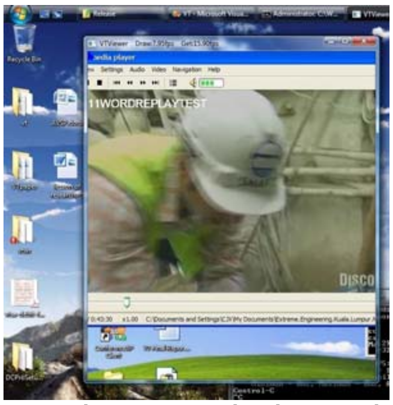


Figure 8. The VT-SORC implementation with one desktop layer and a WordReplay overlay service.

5.3. Limitations

As resolution increases, the output frame rate is lower. The processing time in Figure 8 shows how many frames the MM proxy can process per second. It was found that the processing of 2D frame buffer image in the proxy can easily overwhelm the CPU. In order to keep a constant frame rate output of 15fps, the proxy will need to generate a frame in no less than 66.7ms. If the processing time is longer than 66.7ms, then the frames will start accumulating in the buffer for processing, hence resulting in a lagged video playback. To eliminate this problem, there are several options which can be performed:

- Drop the lagged frames.
- The current algorithm for encoding/decoding and image processing is not efficient. In fact, there are a few redundant memory copies of the entire frame buffer several times. Reducing this can help shorten image processing time.
- Since the GPU is very fast and efficient for image processing, offloading processing from the CPU to the GPU will certainly improve speed. The CUDA (Compute Unified Device Architecture) is one tool that can realize this.

6. Conclusion

In this work, we have introduced the VT and VT-SORC architecture. Compared to the VNC implementation that utilizes the RFB architecture, VT has surpassed frame buffer transmission performance in multimedia transmission. It is able to carry high-bandwidth low-latency demanding multimedia contents. In addition, the VT-SORC encourages easy service injection, crucial for creating a flexible collaboration environment, hence allowing adaptation to different conferencing situation and is a viable alternative collaboration platform compared to the traditional RFB architecture.

7. References

[1] K. Schmidt and L. Bannon, "Taking CSCW Seriously: Supporting Articulation Work," *Computer Supported Cooperative Work (CSCW): An International Journal,* vol. 1, pp. 7-40, 1992.

[2] H. Abdel-Wahab, P. Kabore, O. Kim, and J. P. Favreau, "Replication Management of Application Sharing for Multimedia Conferencing and Collaboration," *Networking and Information Systems Journal,* vol. 2, pp. 63-73, 1999.

[3] T. Richardson, Q. Stafford-Fraser, K. R. Wood, and A. Hopper, "Virtual network computing," *Internet Computing, IEEE,* vol. 2, pp. 33-38, 1998.

[4] C. J. Ng and M. Takatsuka, "Bonjour-based Collaboration Service in a Remote Collaborative Environment," *Apple University Consortium,* pp. 97-106, 2007.

[5] M. P. Papazoglou and D. Georgakopoulos, "Service-oriented computing: Introduction," *Communications of the ACM,* vol. 46, pp. 24-28, 2003.

[6] R. Baratto, J. Nieh, and L. Kim, "THINC: A Remote Display Architecture for Thin-Client Computing," in *Technical Report CUCS-027-04*: Department of Computer Science, Columbia University, 2004.

[7] L. Deboosere, J. D. Wachter, P. Simoens, F. D. Turck, B. Dhoedt, and P. Demeester, "Thin Client Computing Solutions in Low- and High-Motion Scenarios," in *Proceedings of the Third International Conference on Networking and Services*: IEEE Computer Society, 2007.

[8] T. Richardson, "The RFB Protocol," AT&T Labs Cambridge 2007.

[9] K. V. Kaplinsky, "VNC tight encoder-data compression for VNC," in *Proceedings of the 7th International Scientific and Practical Conference of Students, Post-graduates and Young Scientists,* Tomsk, Russia, 2001, pp. 155-157