Bonjour-based Collaboration Service in a Remote Collaborative Environment

Choon Jin NG[1], and Masahiro Takatsuka[2]

[1] ViSLAB, The University of Sydney, masa@vislab.usyd.edu.au
[2]ViSLAB, University of Sydney, choon.jin.ng@gmail.com

Abstract. Virtual Network Computing (VNC) is a widely used remote desktop technology. However, it provides only one collaborative function; shared desktop viewing. Interaction with the desktop is limited to a single user. In this paper, we propose a new collaboration framework using a proxy architecture, which assists dynamic integration of collaboration services using Bonjour technology. Regardless of their platforms, services can virtually stem from any existing software or a proprietary created one, and be deployed into the VNC-based collaborative environment. The prototype created demonstrates the new capabilities of underlying architecture that makes this possible, namely, the Service-Oriented Remote Collaboration framework.

Keywords: Virtual Network Computing, Bonjour, remote collaboration architecture, service proxy, computer-supported cooperative work, CSCW.

1 Introduction

In this highly globalised borderless world, communication plays a significant role in collaboration. Furthermore, the marked advances of computer and networking have brought on the opportunity to enhance cooperative work through the use of computer technology. The concept of "Computer-Supported Cooperative Work" (CSCW), a widely researched field, was first defined by Irena Greif and Paul Cashman, back in 1984 (Schmidt et al. 1992).

In the field of application sharing, there are two approaches for distributing the "shared" application state: (1) Centralised and (2) Replication (Abdel-Wahab et al. 1999). 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 (Richardson et al. 1998). It is based on the idea of exchanging screen images/framebuffers representing a current state of a system. The other approach involves running replicating instances of an individual application. These instances then synchronise locally executed applications/desktops via synchronization of events. This can be achieved in either client/server or peer-to-peer fashion. 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. Moreover, this approach is more susceptible to condition of a network such as any packet losses and delay.

With the emergence of the Service-Oriented Architecture (SOA), we have noticed a possibility to bridge the centralised/replicated architectural gap. A few characteristics of the SOA are:

- Loosely coupled services.
- Service encapsulation and composability.
- Services discoverability.

In this paper, we present the Service-Oriented Remote Collaboration (SORC) framework, a Service-Oriented Architecture, which provides a CSCW environment by means of desktop/application sharing combined with the provision of remotely distributed services. The core of this framework is a proxy architecture named VNC Proxy, which exploits the existing VNC technology. The SORC framework can minimize the limitation of VNC, and yet acquire further advanced functionalities while maintaining VNC legacy support.

By utilising VNC technology, we are able to form an environment composed of loosely coupled services, each of which can be located at different sites. Each service encapsulates certain functions and they can be combined to compose a new set of services. To illustrate this, we provide a few examples of collaboration functions being deployed as

collaboration services: an annotation service called DrawTop, and a white board drawing service called the WhiteBoard.

In order to integrate all the composable services, we have selected Apple's Bonjour technology as the core service deployment mechanism of the SORC framework. It enables services to be discovered and plugged in seamlessly at run-time.

Through this framework, we are also able to provide true multi-user support. This will certainly enhance remote CSCW as participants can now work together and provide simultaneous feedback to the system.
The application of the SORC framework is wide. It can be used in many remote CSCW scenarios such as those in the sector of military, education, emergency services, and medicine.

2 Related Works

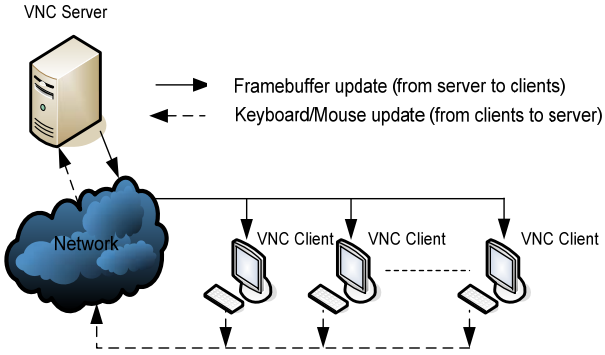


Figure 1. The VNC RFB protocol.

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. As illustrated in

Figure **1**, client inputs are piped to the server. Those parts of the screen that are changed are then sent to the clients as framebuffer updates, an operation based on a single graphics primitive (Richardson et al. 1998):

"Put a rectangle of pixel data at a given x,y position."

However, one major disadvantage of VNC technology is its limitation on the number of server it can control. On any instance, only one server/service can be controlled at a moment, typically a single desktop or a single service such as the CD player control panel illustrated by *Richardson T. et al.* (Richardson et al. 1998). Moreover, the system operates on the basis of a single-user interface environment. To elaborate further, VNC assumes there is only one mouse cursor on the server and so only one client can control the server at any one time. This does not create a true multi-user environment where more than one user can participate actively at a given time.

A similar remote collaboration based CSCW system is the X TerminalView (XTV) prototype created by *Abdel-Wahab et al.* (Abdel-Wahab et al. 1991). Operating on the principle of an event-based synchronization between applications, it allows a group of users to join in a collaborative session for viewing and interacting with X-based single user application by exploiting the X display protocol. The benefits of doing so are its flexibility in controlling how the collaboration session behaves and the possible one-to-one private communication. This is made possible by the fact that the contents of a screen such as windows position, text boxes and size are known in an X Window system. Even though no modification of the single-user application is necessary, XTV relies only on the X-Window system which is not only non cross-platform but also requires programs to be specially created to use the synchronisation mechanism defined.

In another study, Steven *et al.* has created an architecture that utilises an approach, namely the *transparent adaptation* (TA), to leverage a single-user application for multi-user collaboration (Steven et al. 2004). A Microsoft Word (MS

Word) editor has been transformed into a real-time collaborative word processor, called CoWord, which supports multiple users to view and edit any objects in the same Word document at the same time. This approach employs a technique called the *operational transformation* (OT), which intercepts user inputs, analyses and then converts them into the appropriate MS Word API call to edit the document. In this way, user inputs will be made in sync with the changes made. However, the disadvantage of this approach is that the single-user application must have the appropriate APIs available which can fulfil the OT requirements. With this constraint, only a well-designed single-user application which contains a rich set of control APIs can be converted to a multi-user application. This is one of reasons we have avoided event-based synchronisation approach and chosen the more generic RFB approach.

On the RFB approach, *Li et al.* has created an integrated synchronous and asynchronous collaboration system built on top of VNC (Li et al. 2000). The existing VNC architecture was being extended by placing a proxy server in between the VNC server and the clients. The asynchronous proxy exploits the fact that VNC works by having frame buffer updates sent from a server to a client (Li et al. 2000). Therefore, by placing an asynchronous proxy in between the VNC server and the clients, frame buffer updates can be captured and stored into the server. By having this, any late comers who wish to join a VNC collaboration session can now retrieve an archive to keep an update on what had happened before.

The significant gain from this exploited VNC architecture is the ability to add a service while maintaining the legacy support of VNC client and server. This implies neither the VNC servers, nor the VNC clients, has to be modified to accommodate the new service. However, this extended architecture is tightly coupled and only supports one service, the recording of frame buffers to support asynchronous collaboration. On the other hand, we have found this to be an excellent foundation for us to build the SORC framework.

Collaborative VNC supports multi-users with the RFB approach (Levitt 2005). Each user receives its own mouse cursor and this is done by modifying existing VNC protocol. However, the trade-off for is the loss of legacy VNC support.

Besides defining how the system works, it is also important to define a good user interface. In a paper by Hidekazu *et al*. (Hidekazu et al. 1999), the authors devised a framework where shared information is managed under more than two levels or better known as the “pseudo-three-dimensional graphics”. These layers can be categorised into personal layer, group layer and common layer. This would enable users to acquire maximum workspace instead of a screen being split into personal space, public space, or even further fragmentation of the public space to host different activities. Therefore, if there were many activities hosted in a screen, the actual work space could be reduced significantly since a user could only work on one activity at any time.

On the web market segment, web services play a crucial role in realising the SOA. Typically, web services revolve around three protocols, namely (Curbera et al. 2002):

- Simple Object Access Protocol (SOAP) for messaging between services.
- Web Service Description Language (WSDL) for describing the web service’s interface.
- The Universal Description, Discovery, and Integration (UDDI) protocol for the discovery of service providers.

These protocols are built on top of HTTP in conjunction with XML. Together, they allow developers to write services independently of any environment, deploy, and integrate them. Currently, there are two major frameworks that implements these standards and they are the Microsoft .NET framework and the Sun J2EE framework. These frameworks however are unsuitable to be used as they only provide high-level functionalities that are used to implement business logic. It does not support low-level functionality such as network management.

In the field of service discovery for CSCW, most studies have been directed towards providing discovery services for dynamic and portable multi-device Web-based computing and business-to-business framework (Richard et al. 2000; Bartram et al. 2003). These discovery services are designed to provide automated end-to-end connection between clients and services.

With WebSplitter (Richard et al. 2000), as a client or service is plugged into the network, it will register itself to the discovery service. This will, in-turn, lead them to connect to the appropriate collaboration proxy. The benefit of such a service is the provision of “plug-and-play” facility. In fact, only two APIs are needed in WebSplitter to discover a

service, register() and query() (Richard et al. 2000). However, the downside of the discovery service is that it only supports local subnet and not wide-area network (WAN) or enterprise size network.

In Portable Collaborative Networks (Bartram et al. 2003), a hybrid approach is used instead to overcome the problem of WebSplitter. It involves the combination of a local discovery service and a remote presence server. Clearly, such a model will impose more challenges to the user of the service. Users will now have to maintain their own databases of saved peers presentities while synchronising another one with the server (Bartram et al. 2003).

The shortcomings associated with WebSplitter and Portable Collaborative Networks are that they are unscalable and as such, are unable to support WAN. Moreover, it should be able to discover services without incurring a high overhead cost on network bandwidth. For example, a discovery service that floods a network will consume a lot of bandwidth while a discovery service that uses broadcasting method will only work on its own subnet. These systems are therefore only limited to small home or office environment.

Upon close scrutiny (Adrian et al. 2004) on the support of service discovery, querying, and interaction, several widely used discovery protocols have been identified, including Jini (Ken 1999), HAVi (Teirikangas 2001), Service Location Protocol (SLP) (Guttman et al. 1999) and Universal Plug and Play (UPnP) (Glässer et al. 2002). A comparative analysis was made on these services and it was concluded that scalability was the primary design goal for SLP (Adrian et al. 2004). Currently, there are a few implementations of SLP such as Zeroconf (Erik 2001) and Apple's Bonjour technology (Johns 2002).

3 The SORC Framework

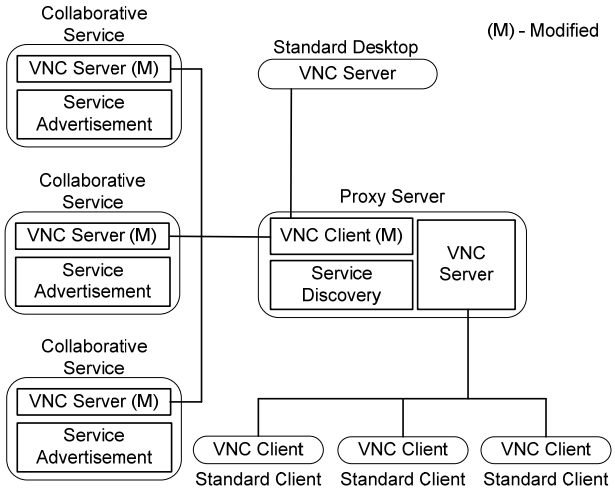


Figure 2. The SORC framework.

To overcome the limitations and shortcomings of the previous works, we have created a new framework called the Service-Oriented Remote Collaboration (SORC) framework shown in Figure 2. This framework contains four components, namely: (1) Standard VNC desktop, (2) Standard VNC clients, (3) Proxy server, and (4) Collaboration services. 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.

3.1 Standard Desktop

To create a desktop, a standard VNC server that can be easily acquired off the Internet such as TightVNC (Kaplinsky 2007) can be used. As VNC server is supported across various platforms, we have the choice to use any desktop/windowing system such as Gnome, KDE, OS X and Windows. The main purpose of this component is to instantly convert any existing application into a collaboration tool.

3.2 Standard Clients

On the user side, an unmodified VNC client is used. As VNC viewers are widely available on many platforms such as OS X, Linux, UNIX and Windows, we have the advantage of a wide existing user base. Furthermore, since VNC clients are essentially thin-clients, we have the benefit of delivering computational intensive applications to user. For example, a collaborative service can be executed on a supercomputer modelling complicated 3D graphics such as the structure of molecules. If these data are to be retrieved and restructured on the client side, intensive computational power and bandwidth are required. However, with the help of the VNC and SORC framework, such computationally heavy services can be deployed and executed remotely.

3.3 Proxy Server

Sandwiched between the VNC clients and the VNC server is a VNC proxy server which enables legacy VNC support plausible. The proxy server is in charge of manipulating client's screen and to multiplex client inputs to the appropriate collaborative services. As such, the VNC server, which interfaces with the clients, must remain unmodified. To maintain legacy support, modification to the RFB protocol can only be made between the proxy server and the collaboration services.

In order to ease the configuration of new services and establish new connections between the proxy server and the collaborative services, the proxy server must have a discovery service. The collaborative service will then have to advertise its service to the proxy server.

3.4 Collaboration Services

Each collaborative service must have a built-in VNC server to communicate with the proxy. To exploit the full potential of the SORC framework, custom services can be written and plugged in. As the proxy supports multi-user, the service can be written to take advantage of it.

In any collaboration, we understand it is essential for a user to be able to communicate with each other with regard to the content of the collaboration. In many cases, an overlay of videos/graphics/texts on top of existing information can be very useful (Cox et al. 2000; Ellis et al. 2004; Kirk et al. 2005; Forlines et al. 2006). As human are good at perceiving patterns and inductive reasoning (Carver et al. 2007), an overlay is a convenient place for them to add additional information without requiring knowledge of the internal data representation. In the SORC framework, an overlay layer is provided such that services or participants can provide information regarding the layer (desktop/standard services) below it.

Regardless of desktop, services or overlay, all of them are treated as services. Particularly, a desktop does not have a service advertisement mechanism and must be connected manually when the proxy server is launched. This is done in order to maintain legacy VNC support to the desktop server. For all services, the proxy server categorise them into three distinct categories as desktop, standard and overlay as shown in Table 1. Overview of the types of services..

Figure 3 illustrates how user's screen on the VNC is composed of the three distinct layers. In order to traverse these layers, users can use a specific hot-key. At the middle layer, standard collaborative services that are discovered are automatically displayed as individual windows and are therefore managed by a window manager built into the proxy server. The window manager will manage the positioning and overlapping of windows.

Table 1. Overview of the types of services.

Service Type	Attribute
DESKTOP	Always at the lower most layer and connects to the VNC server desktop. Only one instance which connects to a standard VNC server.
STANDARD	For normal collaborative services. Typically, they are application window like. These are grouped in the second layer.
OVERLAY	For collaborative services that needs to overlay on top of all other services.

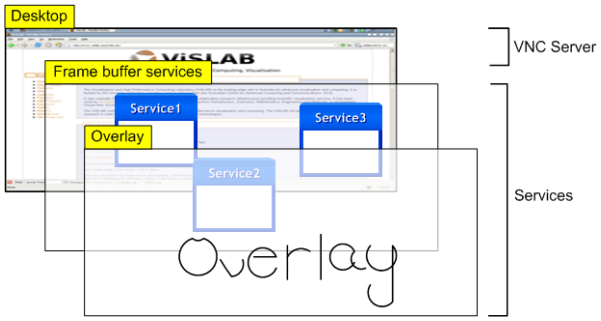


Figure 3. Three distinct layers of screen.

4 Prototype

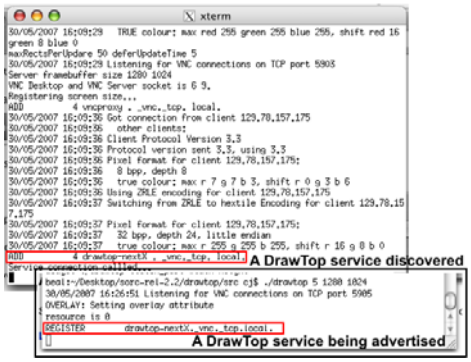


Figure 4. DrawTop advertising its service using Apple's Bonjour and is discovered by VNCProxy.

The SORC framework is demonstrated through the development of a prototype, called the VNCProxy. Two complimentary services, DrawTop annotation overlay service and WhiteBoard drawing service, were also created.

Screenshots of the prototype in action are shown in Figure 4, Figure 5, Figure 6, and Figure 7. In Figure 4, the log indicates that the DrawTop service was advertised. Upon discovery by VNCProxy, the service was added to the collaboration session automatically as an overlay. The prototype showed a few different users drawing on the screen simultaneously with each line colour representing a user.

4.1 VNCProxy

Libraries used to program VNCProxy consist of LibVNCServer, LibVNCClient and mDNSResponder. MDNSResponder is Apple's implementation of their Bonjour service discovery technology.

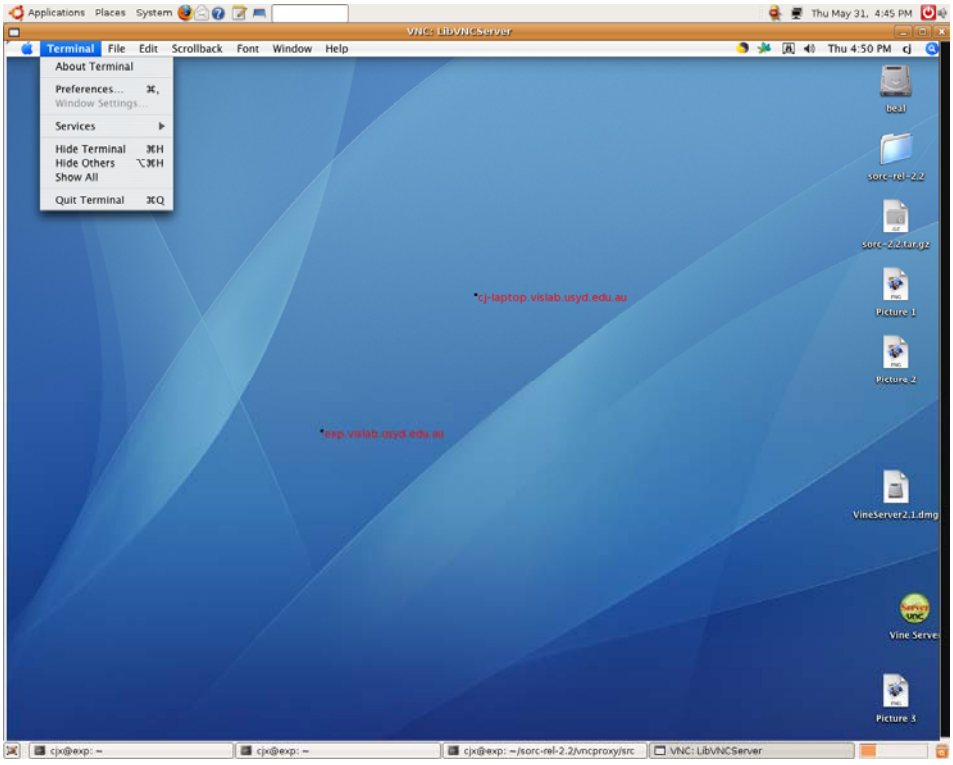


Figure 5. Desktop service with multi-pointer support.

In order to facilitate true multi-user support, the RFB protocol was modified to send mouse co-ordinate in a triplet of (x, y, id) instead of (x, y) as in the original client. With this change, a new RFB version (3.9) was assigned to the protocol. This is done so that the VNCProxy can send the correct form of message to the desktop server (uses current protocol) and to the collaborative services (uses the modified protocol).

VNCProxy has a simple window manager. It allows users to select which service to control (standard mode/overlay mode), move the services window into different desktop location, and hide/show standard services window. Standard mode refers to the control of desktop/standard services, while overlay mode refers to the control of overlay services.

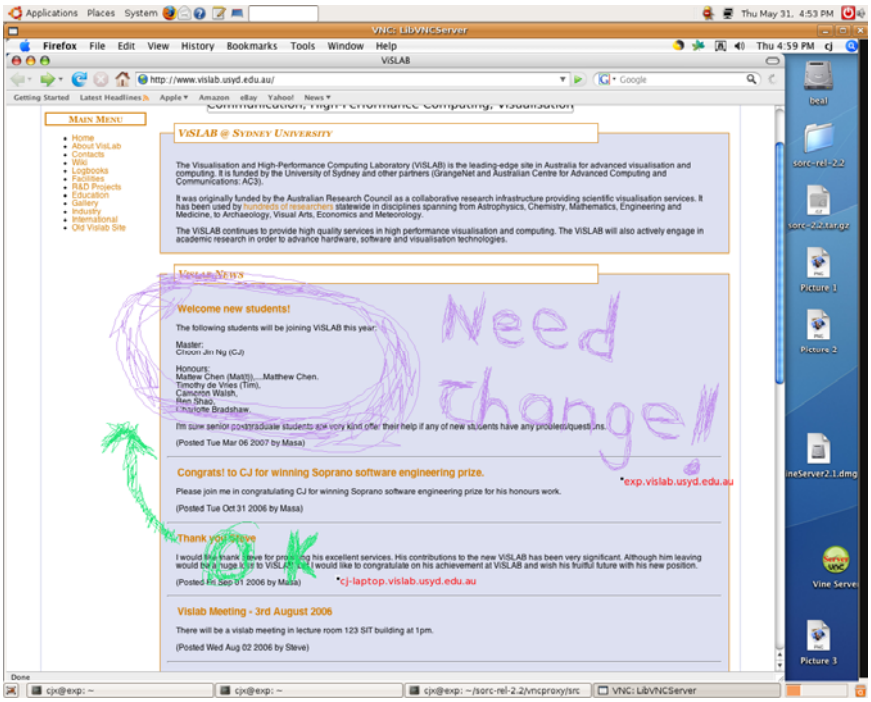
As the desktop itself is a single user interface, conflict of actions between multiple users must be avoided. For example, if multiple users simultaneously move their mouse, the mouse cursor on the desktop would move erratically. To avoid this, we have introduced floor control to VNCProxy in which only one user can grab the desktop floor at a time.

With multi-user support, each individual user is assigned a mouse cursor and it is overlayed onto the screen. As illustrated in Figure 5, there are two users identified by their hostname collaborating on VNCProxy.

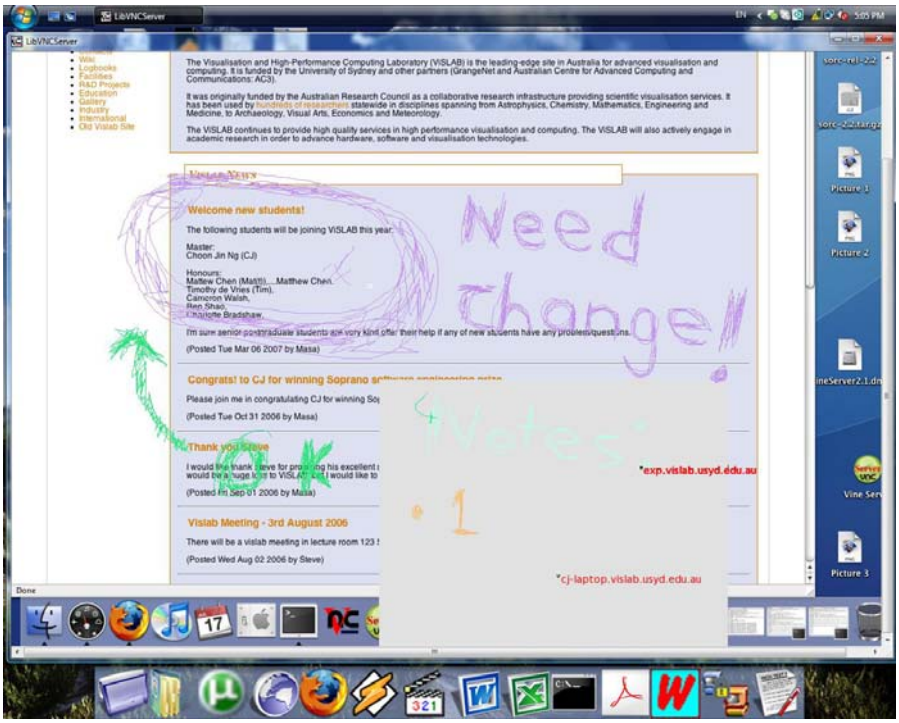
Current implementation supports the following commands:

- F2: Switch control between desktop/standard service and overlay services.
- F3: Move a standard service window to the location pointed by the mouse cursor.
- F4: Hide all standard services.
- F5: Show all standard services.
- F6: Grab desktop floor control.

4.2 Example Services: DrawTop and WhiteBoard



(a) DrawTop overlay service. VNCViewer running on Ubuntu Linux.



(b) WhiteBoard standard service and DrawTop overlay service. VNCViewer running on Windows Vista.

Figure 6. VNCProxy with (a) DrawTop service, and (b) Whiteboard standard service.

As illustrated in Figure 6, the SORC framework major benefit is the availability of cross-platform clients. Shown in the figure are two participants, using two services, collaborating in a Mac OS X system from Ubuntu Linux and Windows Vista.

DrawTop is an overlay service which enables user to annotate on a screen so they can collaborate on the same piece of work. As shown in Figure 6(a), two users are collaborating. Using the DrawTop overlay service, the user *exp.vislab.usyd.edu.au* signalled that a certain portion of the web site needs to be changed. The user *cj-laptop.vislab.usyd.edu.au* then annotate on the screen to acknowledge the former user.

In addition, we have also created WhiteBoard which allows user to sketch notes and ideas on it. It is shown in Figure 6(b) with the grey square area close to the lower right corner of the screen. Since this is a standard service, it behaves like a normal window application which can be closed/open and moved.

Our current implementation for both services has the following commands:

- Left-click + Mouse drag: Draw your annotation on the screen.
- Page Up key: Clear the overlay.

For both services, each client is assigned a random colour when he/she joins the collaboration session. These services uses a modified VNC server to enable them to receive the (x, y, id) triplet of the mouse co-ordinate.

5 Service Integration

As stated previously, the Apple's Bonjour is a very scalable technology. It can be used not only in local area network (LAN), but also in wide area network (WAN). To make automatic service discovery possible, Bonjour has three levels of services, they are (1) Link-local addressing, (2) Multicast DNS, and (3) DNS Service Discovery (Cheshire et al. 2006).

5.1 Link-Local Addressing

The implementation of link-local addressing component is the responsibility of the operating system itself. In an IP network, each device is assigned at least one unique IP address. There are three ways in which an IP address can be determined:

- Statically: Manually entered network information.
- Dynamically: Assigned by DHCP server.
- Self-assigned: When IP address is not available statically and dynamically, the device will randomly assign its own IP address. Using the Address Resolution Protocol (ARP), it then determines if this address crashes with any other devices.

5.2 Multicast DNS

Having an address assigned to a device alone is not sufficient. It can be difficult for users to locate a host using an IP address since they are meaningless and can change any time. As such, in order to identify a host easily, an IP address should be mapped to a meaningful name. This can be done by using Domain Name System (DNS).

However, usual DNS system requires a central name server used for keeping track of naming records. Bonjour exploit this DNS system by using multicast technology. By implementing multicast DNS (mDNS), Bonjour is able to send DNS query using IP Multicast which broadcasts to the entire network. Each client will then have a small DNS server (mDNSResponder) to listen for these queries. If the client knows the answer to the query, then it will reply.

Bonjour uses *.local.* domain name as a pseudo-top-level domain (TLD) (Cheshire et al. 2006). This is done to prevent collision with public domain names.

5.3 DNS Service Discovery

With mDNS in place, Bonjour service discovery can be realised with a DNS-based service discovery (DNS-SD) component. This step is usually implemented in the user application.

DNS-SD exploits a record type used in DNS, called the SRV record. Defined in RFC 2782, the SRV record is intended to enable service discovery in a given domain name lookup. The lookup name takes the form of *_serviceName._connectionType.domainName*. For example, a DNS SRV query with the name *_vnc._tcp.local* will return a list of VNCProxy services. In effect, this query no longer asks for a specific host on the network but rather a list of hosts with VNCProxy service.

However, in reality, different VNCProxy service can exists and we need to categorise this into different "instance name". To achieve this, another DNS record type called the pointer (PTR) record is used. Each PTR record actually points to a SRV record. The format of a PTR lookup is *ServiceType.Domain*, and this will returns a list of instance name of that service specified. For example, a PTR query of *_vnc._tcp.local* may return the service instance names *drawtop._vnc._tcp.example.com* and *whiteboard._vnc._tcp.example.com*. From here, a client can select the service instance and a SRV query for this instance is sent. A list of SRV record is then sent back to the client showing all hosts hosting the instance of this server.

Then, each service may need to describe in details regarding the properties of the service offered. This is done through exploiting the DNS-SD TXT record. All values returned are in the form of "key=value" pair.

In the SORC framework, both VNCProxy and the collaborative services advertise themselves. The "key=value" pair advertised includes:

- VNCProxy: *screenWidth* and *screenHeight* provide the screen resolution (pixels) used by VNCProxy. These attributes are needed by the collaborative services to ensure its screen resolution does not exceed the one used in VNCProxy.

- Collaborative services: *overlay* allows VNCProxy to determine whether to treat the service as an overlay service or a standard application service.

The browsing and resolution of services takes place in VNCProxy. As services are discovered, users can choose whether to add the services (Figure 7). Upon addition of the service, VNCProxy retrieve the service IP address and port number in order to create a new connection to it.

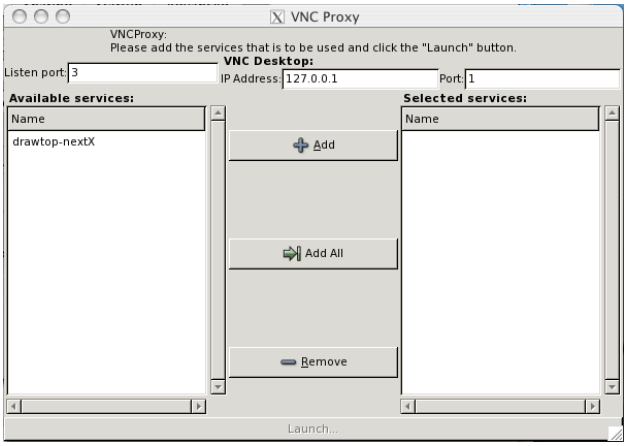


Figure 7. VNCProxy GUI with a service discovered.

Finally, with the choice to select the specific service and compose them together, this completes our SOA architecture.

6 Conclusions

In this work, we introduced a new framework based on VNC technology. It provides seamless integration of distributed collaboration services. The outcome of our current implementation shows that the SORC is a very promising architecture. It provides a method to not only offer proprietary written generic application but also integrates all possible existing services and applications to a single user desktop without any modification.

The current preliminary work indicates there are much more research that needs to be done to further enhance the potential capability of the SORC framework. These include:

- Implementation of a highly personalized screen for each client.
- Creation of more attractive services to further enhance collaboration experience.
- Improvement to the window manager using OpenGL to offer a more complex windowing system.
- Improve the scalability of the framework. Due to the use of TCP protocol, current framework can support no more than six participants.

7 Acknowledgement

We would like to thank the Apple University Consortium for making this project possible through its sponsorship.

References

1 Abdel-Wahab, H., P. Kabore, et al. (1999). "Replication Management of Application Sharing for Multimedia Conferencing and Collaboration." Networking and Information Systems Journal **2**(1): 63-73.
2 Abdel-Wahab, H. M. and M. A. Feit (1991). XTV: a framework for sharing X Window clients in remote synchronous collaboration. Proceedings of TRICOMM '91., Chapel Hill, NC.
3 Adrian, F., D. Nigel, et al. (2004). "Supporting service discovery, querying and interaction in ubiquitous computing environments." Wirel. Netw. **10**(6): 631-641.
4 Bartram, L. and M. Blackstock (2003). "Designing portable collaborative networks." Queue **1**(3): 40-49.
5 Carver, L. and M. Turoff (2007). Human-computer interaction: the human and computer as a team in emergency management information systems. Communications of the ACM. **50:** 33-38.
6 Cheshire, S. and D. H. Steinberg (2006). Zero Configuration Networking: The Definitive Guide. Sebastopol, CA, USA, O'Reilly Media.
7 Cox, D. and S. Greenberg (2000). Supporting collaborative interpretation in distributed Groupware. CSCW '00: Proceedings of the 2000 ACM conference on Computer supported cooperative work, Philadelphia, Pennsylvania, United States, ACM Press.
8 Curbera, F., M. Duftler, et al. (2002). "Unraveling the Web services web: an introduction to SOAP, WSDL, and UDDI." IEEE Internet Computing **6**(2): 86-93.
9 Ellis, S. E. and D. P. Groth (2004). A collaborative annotation system for data visualization. AVI '04: Proceedings of the working conference on Advanced visual interfaces, Gallipoli, Italy, ACM Press.
10 Erik, G. (2001). "Autoconfiguration for IP Networking: Enabling Local Communication." IEEE Internet Computing **5**(3): 81-86.
11 Forlines, C., A. Esenther, et al. (2006). Multi-user, multi-display interaction with a single-user, single-display geospatial application. UIST '06: Proceedings of the 19th annual ACM symposium on User interface software and technology, Montreux, Switzerland, ACM Press.
12 Glässer, U., Y. Gurevich, et al. (2002). High-Level Executable Specification of the Universal Plug and Play Architecture. Proceedings of the 35th Hawaii International Conference on System Sciences - 2002, Hawaii, Microsoft Research, Redmond, WA.
13 Guttman, E., C. Perkins, et al. (1999). "Service Location Protocol, Version 2." Internet proposed standard RFC 2608.
14 Hidekazu, S., O. Ken-ichi, et al. (1999). Perspective layered visualization of collaborative workspaces. Proceedings of the international ACM SIGGROUP conference on Supporting group work. Phoenix, Arizona, United States, ACM Press.
15 Johns, H. (2002). Understanding Zeroconf and Multicast DNS, O'Reilly Media.
16 Kaplinsky, C. (2007). "TightVNC: VNC-Based Free Remote Control Solution." from http://www.tightvnc.com.
17 Ken, A. (1999). The Jini architecture: dynamic services in a flexible network. Proceedings of the 36th ACM/IEEE conference on Design automation. New Orleans, Louisiana, United States, ACM Press.
18 Kirk, D., A. Crabtree, et al. (2005). Ways of the Hands. ECSCW 2005: Proceedings of the Ninth European

19 Conference on Computer-Supported Cooperative Work, Paris, Springer.
20 Levitt, B. (2005). Collaborative VNC, http://benjie.org/software/linux/vnc-collaborate.html.
21 Li, S. F., Q. Stafford-Fraser, et al. (2000). "Integrating synchronous and asynchronous collaboration with virtual network computing." Internet Computing, IEEE **4**(3): 26-33.
22 Richard, H., P. Veronique, et al. (2000). WebSplitter: a unified XML framework for multi-device collaborative Web browsing. Proceedings of the 2000 ACM conference on Computer supported cooperative work. Philadelphia, Pennsylvania, United States, ACM Press.
23 Richardson, T., Q. Stafford-Fraser, et al. (1998). "Virtual network computing." Internet Computing, IEEE **2**(1): 33-38.
24 Schmidt, K. and L. Bannon (1992). "Taking CSCW Seriously: Supporting Articulation Work." Computer Supported Cooperative Work (CSCW): An International Journal **1**(1): 7-40.
25 Steven, X., S. David, et al. (2004). Leveraging single-user applications for multi-user collaboration: the CoWord approach. Proceedings of the 2004 ACM conference on Computer supported cooperative work. Chicago, Illinois, USA, ACM Press.
26 Teirikangas, J. (2001). HAVi: Home Audio Video Interoperability. Espoo, Helsinki University of Technology.