VNCMeeting -
Cross-Platform VNC and Web Technologies

MINT 709 Project Report
Jared Zheng, University of Alberta. July, 2011

READERS
Dr. Paul Lu, University of Alberta

DIRECTOR OF MASTERS OF INTERNETWORKING
Dr. Mike MacGregor, University of Alberta

VNCMeeting: Cross-Platform VNC and Web Technologies

MINT 709 Project Report

Jared Zheng, University of Alberta. July 2011

1. Introduction

Virtual Network Computing (VNC) is a technology for remote desktop sharing. It allows a user's entire desktop environment to be accessed remotely from any network-connected computer. As the internetworking technology has been rapidly advancing and has become ubiquitous, VNC technology can be applied commonly to make communications between network-connected computers more efficient.

One widely use of VNC technology is for technical troubleshooting of customers' problems remotely by large businesses' help desks. It works better than other kinds of communication because it allows a technician to control the client's computer as if it is in front of him.

Another common use of VNC technology is in educational context, where students and professors can share computer screens to demonstrate specific information or to diagnose computer problems. Dr. Paul Lu's VNCMeeting application is designed for such purpose. In a classroom environment, by using VNCMeeting, a student can conveniently share his/her computer screen with a professor, and the professor can perform actions such as viewing the information in the shared screen, controlling the student's computer through the shared screen, or projecting the shared screen to the entire class through a projector.

VNCMeeting has a simple and efficient design. With the help of SSH tunnelling and a well-known server component, VNC Meeting adds security and convenience to the usage of existing VNC server and client applications. Nonetheless, there are limitations.

1. Operating system dependent - VNCMeeting is designed for Linux-based operating systems. Users of VNCMeeting, both the professor and the student, are expected to use Linux computers. VNCMeeting adds new features to traditional VNC applications, but its fundamental usage does not change. Therefore, it still requires the availability of a VNC server application and a VNC client application in the system. To put VNCMeeting into more practical use, it is desirable to make the application compatible with more operating systems and less dependent on external applications.

2. Manually specifying the address of the well-know server - VNCMeeting has three components: the server, the professor and the student. The server needs to be a well-known server, that is, before the professor and the student can communicate with the server, the network address of the server needs to be specified. This is inconvenient when the server needs to be redeployed on a machine with a different network address.

While there are difficulties to eliminate these two limitations entirely, this project presents methods to partially solve the problems, without losing the functionalities and the level of security from VNCMeeting.

1. Avoid the need for platform-specific VNC client by using browser-based technology

   In VNCMeeting, the student component uses a VNC server application to share the screen. It is difficult to generalize this process so that it can be operating system independent. Nonetheless, the professor component that uses a VNC client can be replaced by web-based VNC technology, and allows the system to avoid the need for a traditional platform-specific VNC client. By using Guacamole, an open source HTML5 + JavaScript VNC viewer/client, it is possible to connect to the VNC server through a web browser. By integrating this technology into the existing VNCMeeting system, the professor component can run in a web-browser, and thus provides a platform independent solution.

2. Enhance the usability of the system by applying a service discovery protocol

   Traditionally, in order to use a service on the network, the application needs to know the host name or network address of the machine that provides the service. It is desirable to have the client application discover the service location automatically within the network, which will avoid the administrative effort of ensuring the correctness of the manually specified service location. Applying a service discovery protocol between the well-known server and the student's machine can solve this problem. OpenSLP, an open-source Service Location Protocol, is used in this project.

2. Related Technologies and Background

2.1 Virtual Network Computing

Virtual Network Computing (VNC) is a graphical desktop sharing system that uses the Remote Framebuffer (RFB) protocol to remotely control another computer [1]. It allows a user's entire desktop environment to be accessed from any network-connected machine. The RFB protocol, a simple protocol for remote access to graphical user interfaces, works at the framebuffer level and therefore applies to all operating systems [2]. Besides the RFB communication protocol, a VNC system consists of a VNC server that shares the screen, and a VNC client that connects to and interacts with the VNC server. VNC system is platform-independent because of the design of the communication protocol; therefore, a VNC client can interacts with a VNC server running on a different operating system.

2.2 VNCMeeting 1.0

Dr. Paul Lu created the original VNCMeeting. For the rest of this paper, the original version of VNCMeeting will be referred to as VNCMeeting 1.0.

VNCMeeting 1.0 is designed for students to share their computer screens with the professor conveniently and securely in the classroom environment. It consists of three components: the well-known server, the professor, and the student.

To use VNCMeeting, a student launches the student component of VNCMeeting, which communicates with the well-known server to retrieve an available port number, creates an SSH tunnel to the well-known server through port-forwarding, and launches a VNC Server. This puts the student to a ready state for incoming VNC connections. The professor acquires the port number that the student is using, usually by verbal communication, establishes an SSH tunnel to the well-known server through port-forwarding with that specific port number, and launches a VNC client. Figure 1 shows the basic architecture of VNCMeeting 1.0.

The main advantage of VNCMeeting 1.0 over traditional VNC application is the enhanced security from using SSH tunnels, and the avoidance of manually specifying the network address of the VNC server. Nonetheless, the overall experience of using VNCMeeting 1.0 is still similar to that of traditional VNC applications, and it still involves multiple manual steps. To make VNCMeeting more user-friendly and feature-rich, Roxanna Wong and Mengtao Ye have developed new versions of VNCMeeting.

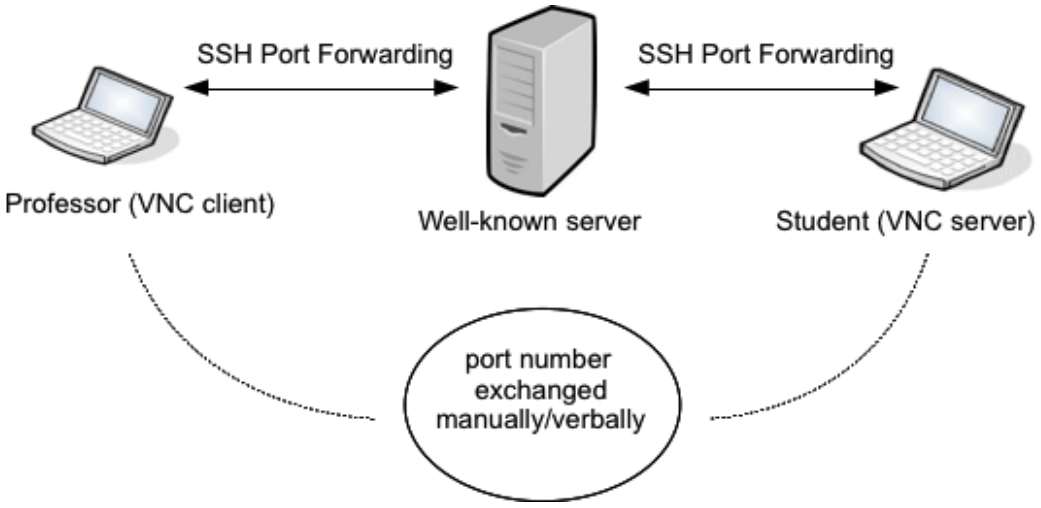


Figure 1. Basic architecture of VNCMeeting 1.0

2.3 VNCMeeting 2.0

Roxanna Wong has developed an enhanced version of VNCMeeting by adopting a serverless design and implementing automation of information exchange [3]. In this version, the information exchange required for establishing the VNC connection is automated. A front-end GUI for the professor facilitates the automation as well as provides user friendly operations. After the preparation is finished, the professor connects to the student directly through an SSH tunnel, without the need for the VNC data going through the well-known server. This is different from VNCMeeting 1.0, in which the VNC data connection between the professor and the student involves two SSH tunnels through the well-known server.

Mengtao Ye has developed another enhanced version of VNCMeeting based on VNCMeeting 1.0 [4]. This version mainly aims at adding new features: raising window and switching session. It expands VNCMeeting 1.0 to support multiple professors instead of a single professor. It adds functionalities to allow students to switch their VNC sessions from one professor to another, and raise a VNC session window remotely.

These two enhanced versions of VNCMeeting are also later merged into a more complete version that provides all the new functionalities. For the rest of this paper, this combined version of VNCMeeting will be referred to as VNCMeeting 2.0, and this is the version of VNCMeeting this project is based on. Figure 2 show the basic architecture of VNCMeeting 2.0.

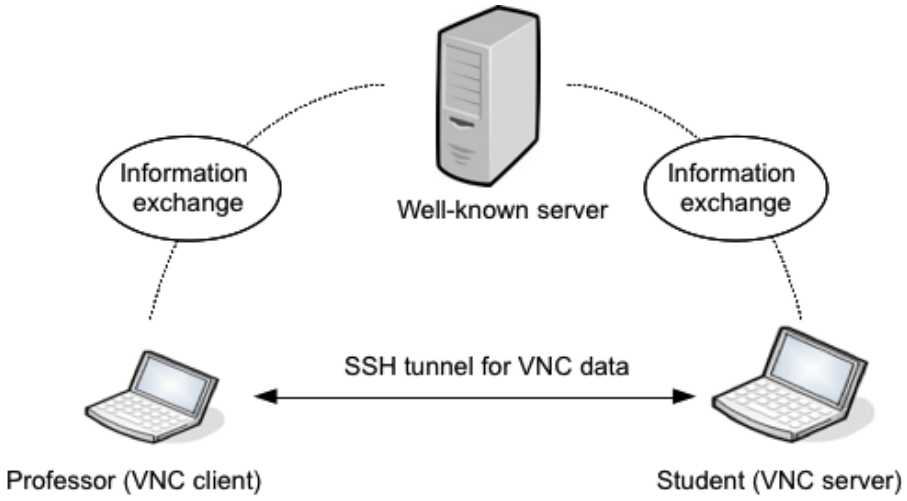


Figure 2. Basic architecture of VNCMeeting 2.0

2.4 Guacamole - HTML5 + JavaScript VNC viewer

Guacamole is an HTML5 + JavaScript (AJAX) viewer for VNC, which makes use of a server-side proxy written in Java [5]. The server-side half of Guacamole thus requires a servlet container like Apache Tomcat, while the client-side requires nothing more than a web browser supporting HTML5 and AJAX.

There are other web-based VNC viewers/clients. Some are commercial software, and some require other components such as Java, ActiveX installed for the web browser. Guacamole, which is an open source project, is a better candidate for this project because of its non-intrusive design, which allows Guacamole to be integrated into VNCMeeting in its entirety without code modification. The version of Guacamole used in this project is 0.3.0rc1.

2.5 Apache Tomcat

Apache Tomcat is an open source software implementation of the Java Servlet and Java Server Pages technologies [6]. It provides an HTTP web server environment for Java code to run. It can also serve as a general HTTP web server for HTML and non-Java CGI scripts. Apache Tomcat is used in this project to host the server-side VNC client proxy of Guacamole, as well as the web application component that replaces the GUI in VNCMeeting 2.0. The version of Apache Tomcat used in this project is Tomcat 6.0.

2.6 OpenSLP, Service Location Protocol (SLP)

Service discovery protocols are network protocols that allow computers to detect the network addresses of devices and services on a network. Service Location Protocol (SLP) is a service discovery protocol that provides a flexible and scalable framework for the discovery and selection of network services in enterprise networks. OpenSLP is an open-source implementation of SLP that provides a complete and important feature-set suitable for enterprise network.

3. Use Cases

This section describes the major use cases of the new VNCMeeting system implemented in this project. The environment is assumed to be a classroom environment that has multiple students as well as multiple professors. Students would like to share the screens so that the professors can either view the shared information or control the students' computers.

3.1 One Professor, One Student

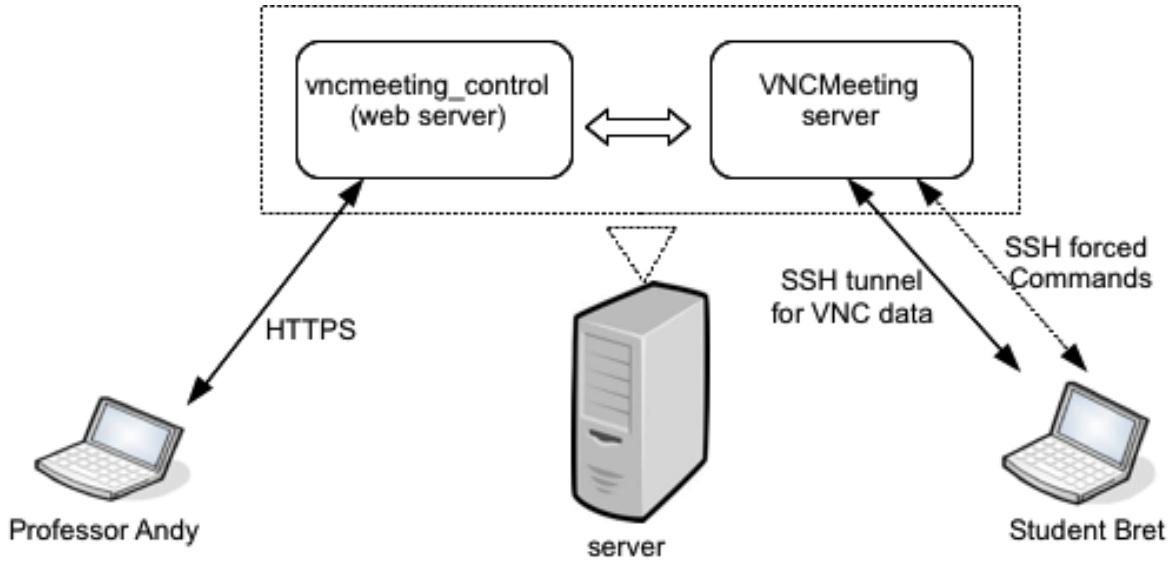


Figure 3. Use Case 1: one Professor, one Student

Scenario:
Student Bret shares his computer screen with Professor Andy.

Procedure:

1. Professor Andy opens a web browser and opens the VNCMeeting control page for professors with URL https://[well-known-server-address]/vncmeeting_control/. Professor Andy is warned about the self-signed SSL certificate. He accepts and proceeds. Professor Andy is prompted for a general username/password to access the page. He enters the username/password and proceeds (the username/password is distributed to all Professors who wish to use this page). Professor Andy is prompted for a professor username. He enters "Andy", and proceeds to see a menu with an empty list of Students.

2. Student Bret starts VNCStudent on his computer. Upon seeing a list of available menu options, Bret selects "1. Make Available to Professor". Bret selects "1.Andy", the only available Professor, from the Professor list. Bret enters "Bret" when prompted for a username. A message then indicates that Bret has successfully made available to Professor Andy under the name "Bret".

3. After the student list is refreshed, Professor Andy notices that Student Bret is now available for connection. He clicks the "Connect" button next to the Student name. A rectangular frame that displays Bret's computer screen replaces the white space next to the menu. Professor Andy is now able to view and control Bret's screen.

4. Professor Andy finishes with Bret's shared screen. He clicks the "Disconnect" button next to the Student name and the "Connect" button. The rectangular frame that displays Bret's screen next to the menu disappears.

5. Professor Andy is ready to leave the page. He tries to close the page, but is warned to "Logout" first. Professor Andy clicks the "Logout" button, and a message indicates that he has logged out gracefully.

3.2 One Professor, Two Students, Raising window

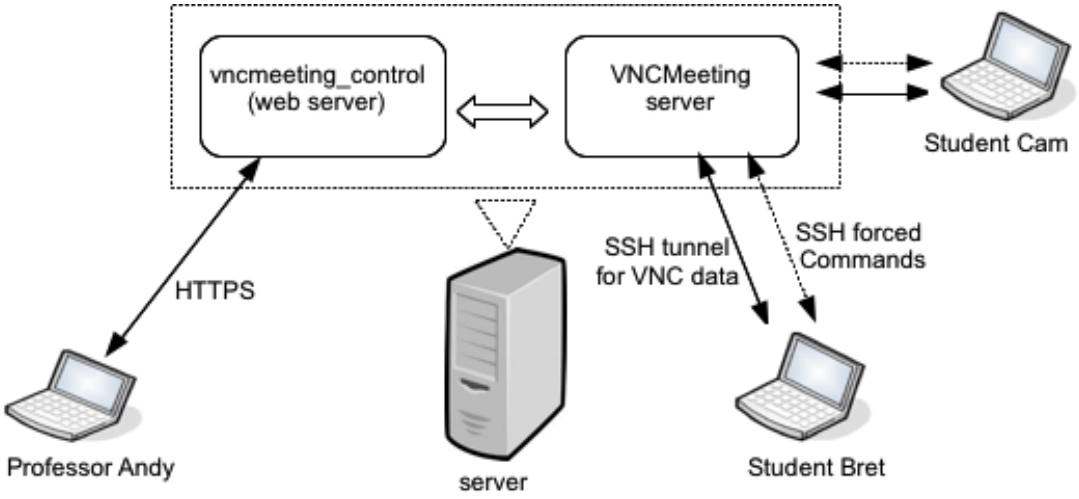


Figure 4. Use Case 2: one Professor, two Students

Scenario:
Student Bret and Student Cam share computer screens with Professor Andy. Professor Andy successfully connects to both Students' shared screens simultaneously. He is able to easily switch between the two shared screens. When Professor Andy is working on Bret's shared screen, Student Cam sends a "gotop" message to bring his shared screen to foreground.

Procedure:
1. Professor Andy opens the VNCMeeting control page in a web browser and logs in with username "Andy" (see use case 3.1 Procedure 1).

2. Student Bret and Student Cam start VNCStudent and both make available to Professor Andy for connection.

3. After the student list is refreshed in the webpage, Professor Andy sees that two students, Bret and Cam, are available for connection. Professor Andy clicks the "Connect" button next to the name Bret to connect to Bret's screen. Bret's computer screen is displayed next to the menu on the right.

4. Professor Andy also clicks the "Connect" button next to the name Cam. Cam's computer screen is displayed next to the menu, on top of Bret's screen.

5. Professor Andy decides to work on Bret's screen, so he again brings up Bret's screen by clicking the Student name Bret.

6. Student Cam would like to get Professor Andy's attention, so he selects the "2. Bring window to top." menu option in VNCStudent.

7. Professor Andy sees that Student Cam's shared screen is automatically brought to top. A message also pops up in the top-left corner indicating that a "gotop" request has been received from Student Cam.

8. Professor Andy finishes working with both Students' shared screen, and logs out.

3.2 Two Professors, One Student, Switching session

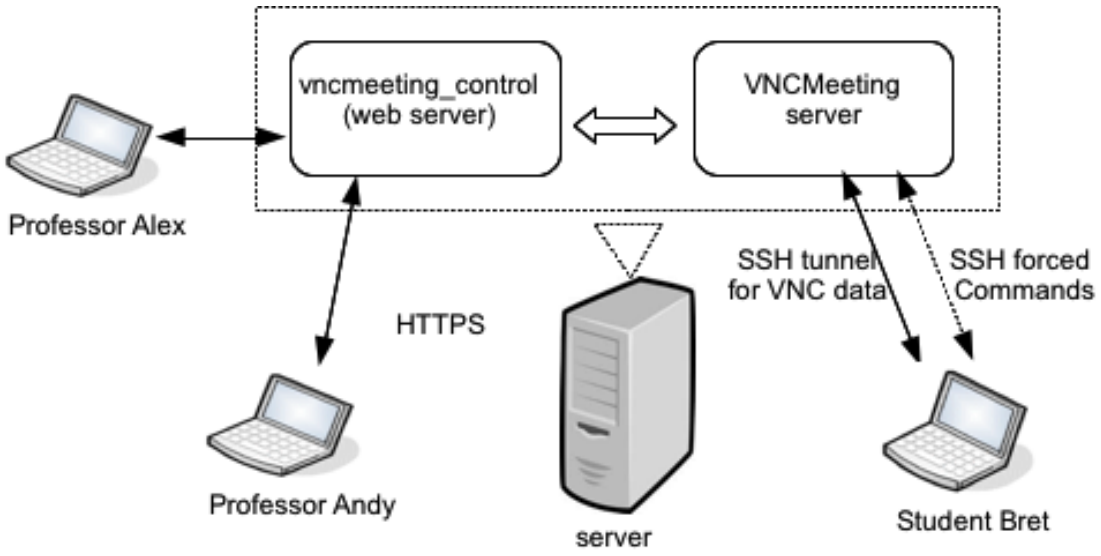


Figure 5. Use Case 3: two Professors, one Student

Scenario:
Professor Andy and Professor Alex have logged in to the system. Student Bret first makes available to Professor Andy for connection, and then switches to Professor Alex.

Procedure:

1. Professor Andy and Professor Alex both open the VNCMeeting control page in a web browser and logs in with their corresponding usernames, "Andy" and "Alex". (See use case 3.1 Procedure 1).

2. Student Bret starts VNCStudent and makes available to Professor Andy by selecting Andy from the list of available Professors.

3. Professor Andy connects to Student Bret, and works on the shared screen.

4. Student Bret now wants to share the screen with Professor Alex, regardless of whether or not Professor Andy has finished with the screen sharing session. Bret selects the "1. Switch Professors." menu option in VNCStudent, and again enters username "Bret", which may or may not be the same as the one previously entered.

5. After the Student list is refreshed, Professor Andy notices that Student Bret is no longer available, and a message also pops up in the top-left corner indicating that Student Bret has been removed from the list.

6. Professor Alex now sees that Student Bret is available for connection. He proceeds to connect to Student Bret's shared screen.

7. Both Professors log out respectively when finish.

4. Technical Overview

The architecture of the VNCMeeting system in this project is similar to that of VNCMeeting 2.0, it consists of three components: the Server, the Professor and the Student. Most of the functionalities in the Server and the Student components remain unchanged, but The Professor component has a different design and is deployed differently. This section describes the role of each component and how the components interact with each other in the system.

4.1 The Student

The Student component remains mostly the same as in VNCMeeting 2.0. It is an executable program that can be installed on any compatible operating systems to interact with the other components of VNCMeeting.

The Student's main functionality is to start a VNC server to make the computer screen available for sharing. It communicates with the Server component to provide the necessary information for the Professor to connect to, as well as to cooperatively establish the SSH tunnel.

4.2 The Server

The Server component remains mostly the same as in VNCMeeting 2.0. Its main functionality is to facilitate the information exchange between the Professor and the Student, to store the information required for establishing the connection, and to enhance the security by maintaining an SSH tunnel for the VNC data stream.

The Server component is deployed on a machine that is network-reachable from the Student and the Professor. It communicates with the Student and the Professor through two executable programs. The communications are based on the typical request-response mechanism. The Student and the Professor executes the programs actively to request for a piece of information or to request to have a task done, and the response is returned when the programs finish.

4.3 The Professor

The Professor component maintains most of the functionalities from VNCMeeting 2.0, but the design and architecture has changed. The Professor component is deployed as a separate web application (named vncmeeting_control) on the same machine that the Server component is deployed, instead of an executable program on the user's machine.

As a typical web application, the Professor component can be further divided into two sub-components: the server-side management scripts and the client-side HTML pages. A user (professor) accesses this web application in a web browser, and communicates with the server-side management scripts, which communicate directly with the Server component.

The following diagram shows the architecture of the current version of VNCMeeting, compared to VNCMeeting 2.0.

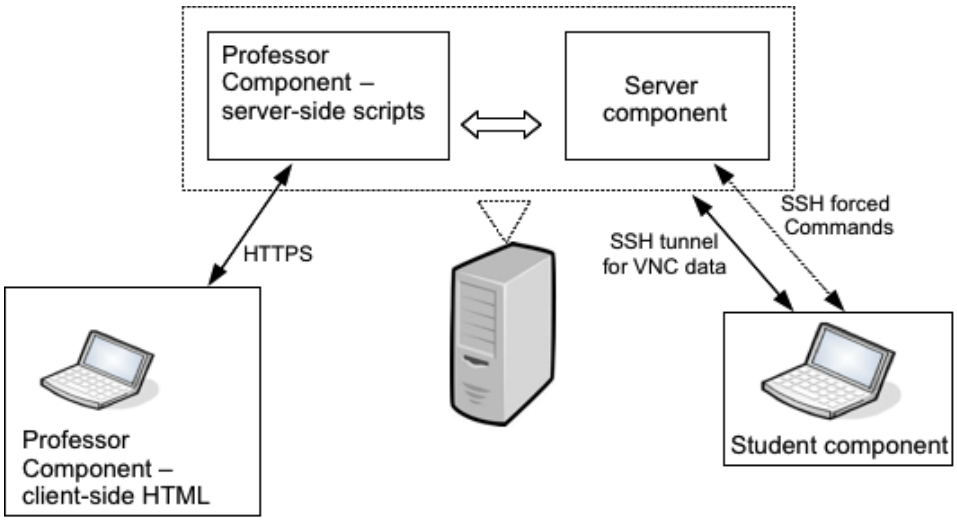


Figure 6. Architecture of the current version of VNCMeeting

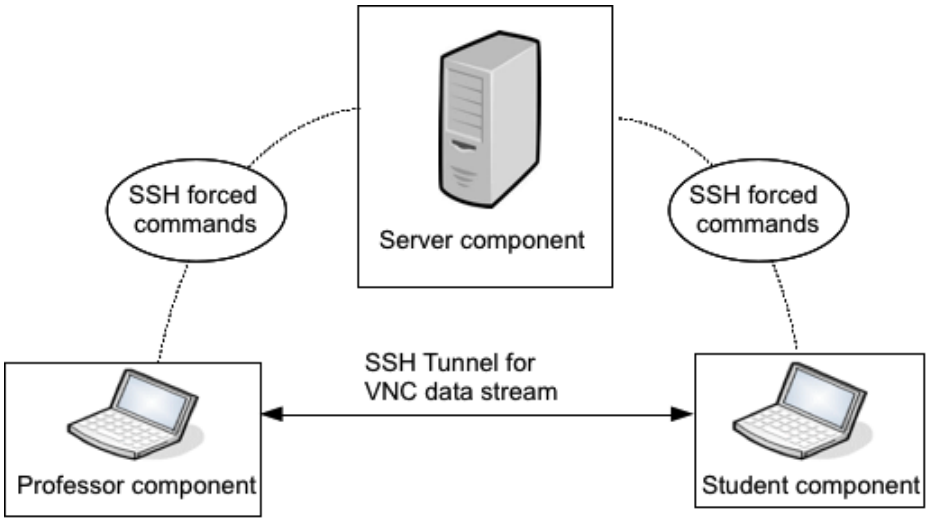


Figure 7 Architecture of VNCMeeting 2.0

5. System Requirements

This section lists all the external dependencies the VNCMeeting system requires.

5.1 The Student

- x11vnc - the VNC server application that delivers framebuffer data
- OpenSLP API - the API for using the Service Location Protocol

5.2 The Server

- OpenSLP - the OpenSLP package that includes the OpenSLP daemon, the API and the testing tool

5.3 The Professor

- Apache Tomcat - the web server for hosting vncmeeting_control (i.e. the Professor component) and Guacamole
- Guacamole - the HTML5 + JavaScript VNC viewer
- Ruby - the CGI scripts used in vncmeeting_control (i.e. the Professor component)
- HTML5 supported web browser - a compatible web browser to access vncmeeting_control (i.e. the Professor component, the web application)

6. Design & Implementation

This section describes all the major design and implementation details of the VNCMeeting system.

6.1 Applying OpenSLP, the Service Location protocol, to VNCMeeting

One of the goals in this project is to enhance the usability of the system by applying service discovery protocol. In the previous versions of VNCMeeting, after the Server component is deployed on a machine, the Professor and the Student components need to know the network address of the machine in order to communicate. The network address is either specified directly in the programming code, or in a configuration file. Even though the configuration file provides a flexible way of changing the address of the well-known Server, it still requires manual intervention every time the Server is redeployed on a different machine. OpenSLP is used in this project to avoid this administrative effort.

6.1.1 Service Location Protocol Overview

The Service Location Protocol (SLP) is an Internet Engineering Task Force (IETF) standards track protocol that provides a framework to allow networking applications to discover the existence, location, and configuration of networked services in enterprise networks [8].

The most fundamental and important concept of SLP is the existence of SLP agents, which are software entities that process SLP protocol messages. There are three types of SLP agents: User Agent (UA), Service Agent (SA) and Directory Agent (DA) [8]. User Agent is a software entity that is looking for the network location of one or more devices or services. Service Agent is a software entity that advertises the location of one or more services. Directory Agent is a software entity that acts as a centralized repository for service location information.

To find a service within the network, the User Agent issues a request that is multicast to all Service Agents, and the Service Agents that advertise the requested service unicast a reply to the User Agent. Alternatively, if one or more Directory Agents are present in the network, the service request is unicast to the Directory Agents, and the Directory Agents unicast the reply to the User Agent.

In the context of the VNCMeeting system, the Server component is a Service Agent, and it provides the service to handle VNCMeeting requests from both the Student and the Professor. The Student Component is a User Agent, and it queries the network location of the service in order to make VNCMeeting requests. There is only one type of service in the VNCMeeting system, so it is not necessary to use a Directory Agent, whose functionality is to enhance the performance and the scalability of SLP by centralizing service information.

6.1.2 OpenSLP overview

OpenSLP is an open-source implementation of the Service Location Protocol. It provides developers with tools and API to add SLP based features to existing applications.

slpd - The slpd daemon is an executable program provided by the OpenSLP package to provide the Service Agent and the Directory Agent functionality. It is expected to be running from the time that a service is advertised to the time that the service becomes unavailable. In other words, it accepts the registration request for a service from the Service Agent, maintains the service information and provides it to the User Agent when queried.

OpenSLP API - The OpenSLP API allows programmers to add SLP based functionalities to their applications to communicate with other SLP Agents within the network. The major functionalities offered by the API include finding services as well as registering/deregistering services. Specifically, programmers include the SLP library (libslp.sl) and call the corresponding SLP functions in their programs.

slptool - slptool is an executable program provided by the OpenSLP package. It implements most of the functionalities from the SLP API, and can be used as a standalone program to communicate with other SLP Agents to find services as well as register/deregister services.

6.1.3 Using OpenSLP in VNCMeeting

The goal is to allow the Server component in the VNCMeeting system to advertise its service (with the network address), and the Student component is able to query the network and find the service advertised by the Server.

The slpd daemon is the only tool from the OpenSLP package that can be used to advertise a service, so it needs to be running on the same machine the Server component runs on. The Server component also needs to register and deregister the service to the running slpd daemon. The slptool is used for these operations. The combination of the slpd daemon and the slptool makes the Server component a fully functional Service Agent in the SLP model.

While the Server component is functioning correctly as a Service Agent, the Student component uses the OpenSLP API function calls to query the location of the service. The Student component only queries the service location, so the SLP API alone is sufficient and the slpd daemon is not required.

The following diagram describes how OpenSLP is used in the VNCMeeting system.

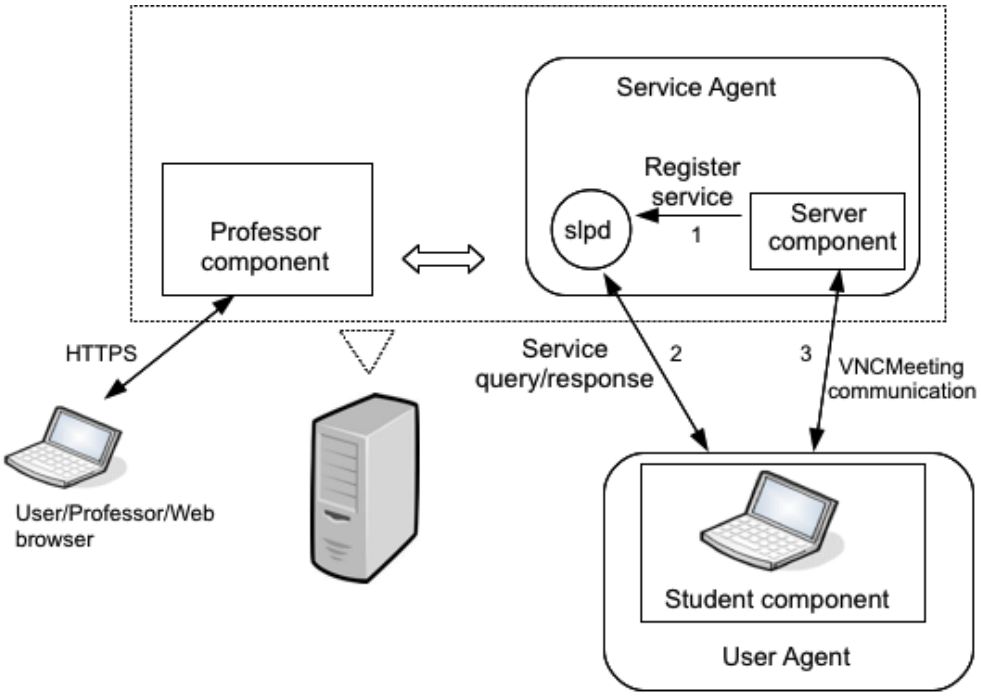


Figure 8. VNCMeeting integrated with OpenSLP

6.1.4 What OpenSLP cannot solve in VNCMeeting

Using OpenSLP can eliminate the need for the Student component to know the network address of the Server. In VNCMeeting 2.0, OpenSLP can be used in the Professor component to solve the same problem, so that neither the Student nor the Professor is required to know the network address of the Server. Nonetheless, in this new version of VNCMeeting, the Professor component is deployed differently. When a user/professor accesses the web application through a web browser, he/she needs to specify in the URL the hostname/address of the machine that the Professor component (i.e. the web application) is hosted on.

It is possible to use an external tool (e.g. the slptool) to detect the service location before accessing the web application, but this creates a new dependency on using the VNCMeeting system, and diminishes the purpose of this project, which is to use a web browser solely to access the Professor functionalities. This issue remains unsolved in this version of VNCMeeting. The users always need to know the correct URL to access the web application.

6.2 The Server

The Server component remains mostly the same as in VNCMeeting 2.0, except for the added support for SLP. The original functionality of the Server component can be separated completely from the SLP operations, so it only requires little modification to the original design.

The Server component can be divided into three sub-components: *studentcommands*, *professorcommands* and *cleanup*. *studentcommands* and *professorcommands* are executable programs that are executed by the Student and the Professor respectively, on a per request-response basis. The only ongoing process is the *cleanup* program, which checks and cleans up out-dated user files at a constant interval. The running time of the cleanup process can be considered the active period of the Server component. Therefore, the SLP operations can be wrapped around the start and the end of the cleanup process.

Since there is no relation between the cleanup process and the SLP operations, the SLP operations are done in a separate bash scripts instead of being mixed into the cleanup program. Two bash scripts are used for starting and stopping the Server component: *vncmeetingServerStart.sh* and *vncmeetingServerStop.sh*.

vncmeetingServerStart.sh first decides the network address that is to be used for advertising the service. Administrators are allowed to manually set the IP address as well as let it be automatically detected. The two variables that are configurable in the script are IP_ADDRESS and ETH_INTERFACE. The following table describes the logics for determining the final network address for the advertised service. It is recommended to configure using option 1 or 2. The method of automatically determining the network address (option 3) may be too general for a more complex network setting, and needs to be refined for more practical use.

Option	**IP_ADDRESS**	**ETH_INTERFACE**	**The final network address used for the advertised service**
1	A valid IP (e.g. 172.16.60.158)	EMPTY	The value of IP_ADDRESS (i.e. 172.16.60.158)
2	EMPTY	The name of a network interface (e.g. eth0)	The network address bound to the network interface specified by ETH_INTERFACE
3	EMPTY	EMPTY	The network address bound to the first network interface listed by ‘ifconfig’

After the network address is decided, *vncmeetingServerStart.sh* starts the slpd daemon and register the service by using the slptool. The registered service has the form *service:ssh.vncmeeting://172.16.60.158*, with "ssh.vncmeeting" being the description of the service, and the address that follows being the network address decided earlier in the script. The description string is what is used by the User Agents to search for this service.

After all the SLP related operations are finished, *vncmeetingServerStart.sh* starts the *cleanup* process, which is manually started in VNCMeeting 2.0.

vncmeetingServerStop.sh does the opposite of vncmeetingServerStart.sh. It is to be executed when the Server becomes unavailable. It kills the cleanup process and the slpd daemon. It is not necessary to deregister the service from the slpd daemon.

6.3 The Student

Similar to the Server component, the Student component mainly differs from VNCMeeting 2.0 on the support for SLP.

After start-up, VNCStudent first finds the location of the service provided by the Server component. By using the SLPFindSrvs() function from the SLP API, VNCStudent searches for the service with the description "ssh.vncmeeting", and receives the response in the same form the service is registered e.g. service:ssh.vncmeeting://172.16.60.158. The network address is abstracted from the response and saved. Alternatively, VNCStudent can execute an external tool (i.e. slptool) to search for the service and get the same response from the output of slptool.

There are other changes in VNCStudent that are related to the new design of the Professor component, and they are further discussed in the later sections of this paper.

6.4 The Professor

The Professor component is what differs mostly from VNCMeeting 2.0. The main goal of this project is achieved by integrating the original Professor functionalities with other external tools and technologies.

6.4.1 Separating the GUI unit from the controller unit

In VNCMeeting 2.0, the Professor component is a complete application that consists of the controller unit and the GUI unit. The component runs in one active process with multiple threads. The GUI thread monitors user actions and performs controller actions based on the events triggered. Each running instance of the Professor component represents an active user/professor. Since the GUI unit and the controller unit are within the same process, they can share the same resources, which means that any stateful information about the active user/professor is available to both the units.

This is not the same in the new version of VNCMeeting. The Professor component is deployed as a web application contained in Apache Tomcat. After the Professor component becomes a part of an active VNCMeeting session, the GUI, which is the client-side HTML/JavaScript, is loaded onto the user/professor's web browser, and the controller unit, which is the server-side CGI scripts, resides in the server machine that hosts the Apache Tomcat. The GUI unit and the controller unit become separate processes running on different machines, and the communication mechanism between the two units must be

different from that in VNCMeeting 2.0. The following diagrams show the difference between the designs of the Professor components in the two versions.

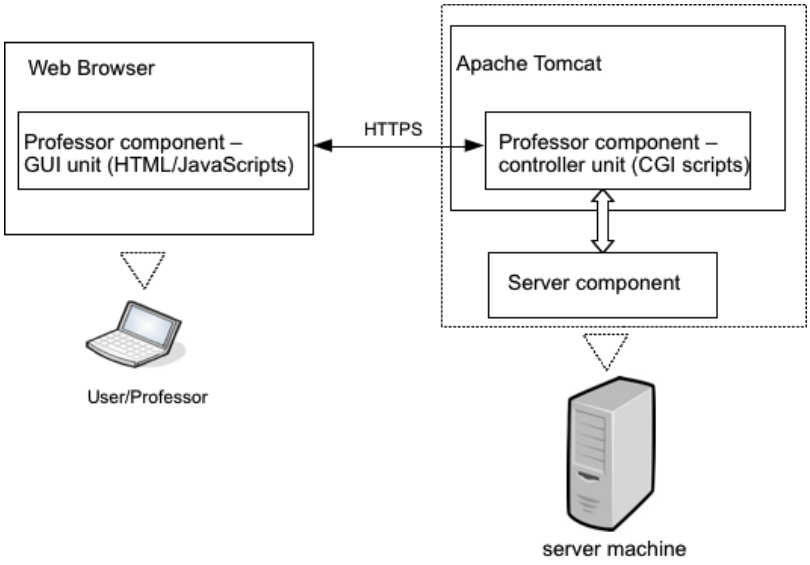


Figure 9. Professor Component in new VNCMeeting

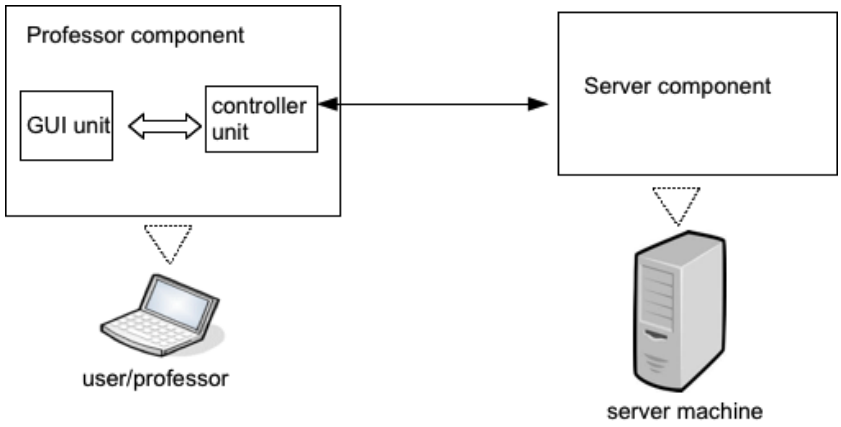


Figure 10. Professor Component in VNCMeeting 2.0

6.4.2 A stateless controller unit in the Professor component

In VNCMeeting 2.0, the GUI unit and the controller unit in the Professor component exist within one process. The whole process is running during the time the user/professor joins the system to the time the user/professor exits. Within this period of time, the running process represents the active user/professor, and maintains a state of information for the user/professor. In the new version of VNCMeeting, the GUI unit, which exists in the user/professor's web browser, also maintains a state, but the controller unit, which exists in the server machine in the form of CGI scripts, is stateless.

Every time the GUI unit communicates with the controller unit, the GUI unit sends an HTTP request to the Apache Tomcat, and the Apache Tomcat spawns a process and executes the corresponding CGI script. When the script finishes, it returns an HTTP response to the user/professor's web browser. The process for the controller unit for a specific user/professor exists only on a request-response basis. When there is no ongoing request-response, the controller unit is simply at rest, that is, no processes are running.

The controller unit is stateless, in the sense that, each time a process is spawned to run the CGI script, it has no knowledge of the previous interactions and the request is handled entirely based on information that comes with it.

6.4.3 Communication between the Professor and the server

In VNCMeeting 2.0, the communication between the Professor component and the Server component is in the form of SSH forced commands. After the SSH connection is established, the executable program (i.e. professorcommands) is executed remotely on the Server machine to handle the request from the Professor.

The Professor component is now deployed on the same machine the Server component runs on, the communication between the Professor component, more specifically the controller unit, and the Server component can be more direct. The SSH connection between the Professor component and the Server component can be removed, and the Professor component executes the executable program directly. The overhead of establishing the SSH connection is thus avoided.

6.4.4 Integration with Guacamole

One of the main features of this project is to replace the traditional VNC client with the web-based technology, so that the VNC data can be delivered to and rendered in a web browser. This is achieved by using Guacamole. Guacamole provides the functionality of a web-based VNC client and has a clean and non-intrusive interface.

A Context is what Tomcat calls a web application [9]. A Context Descriptor is an XML file that contains Tomcat related configuration for a Context. An important configuration in the Context XML file is the 'docBase' attribute, which contains the path of a web application resource (WAR) file, which is a compressed file that contains all the resources of a web application, including the programming codes.

To use Guacamole to connect to a running VNC Server, it is necessary to deploy a Context XML file with the 'docBase' attribute pointing to the guacamole.war file that is installed somewhere on the machine. The Context XML file also needs to contain connection-specific parameters. A connection can then be made from the web browser by pointing to the web application specified by the Context XML file (usually the name of the Context XML file). Upon success, the shared screen is reflected directly on the web browser filling the majority of the window space, and is ready for further user interactions.

Guacamole allows multiple simultaneous instances. This is achieved by deploying multiple Context XML files, with each containing different connection-specific parameters, e.g. the VNC Server's address, port, etc. Users are then able to connect to different VNC Servers simultaneously by pointing the web browser to the corresponding web applications.

The interception point between Guacamole and the VNCMeeting system is the Context XML file. After the Professor component acquires all the information required for connecting to the Student's VNC Server, it creates a Context XML file with the corresponding attributes, and copies the Context XML file to the appropriate directory under Apache Tomcat. The user/professor is then able to connect to the Student's shared screen via the GUI unit of the Professor component in the web browser.

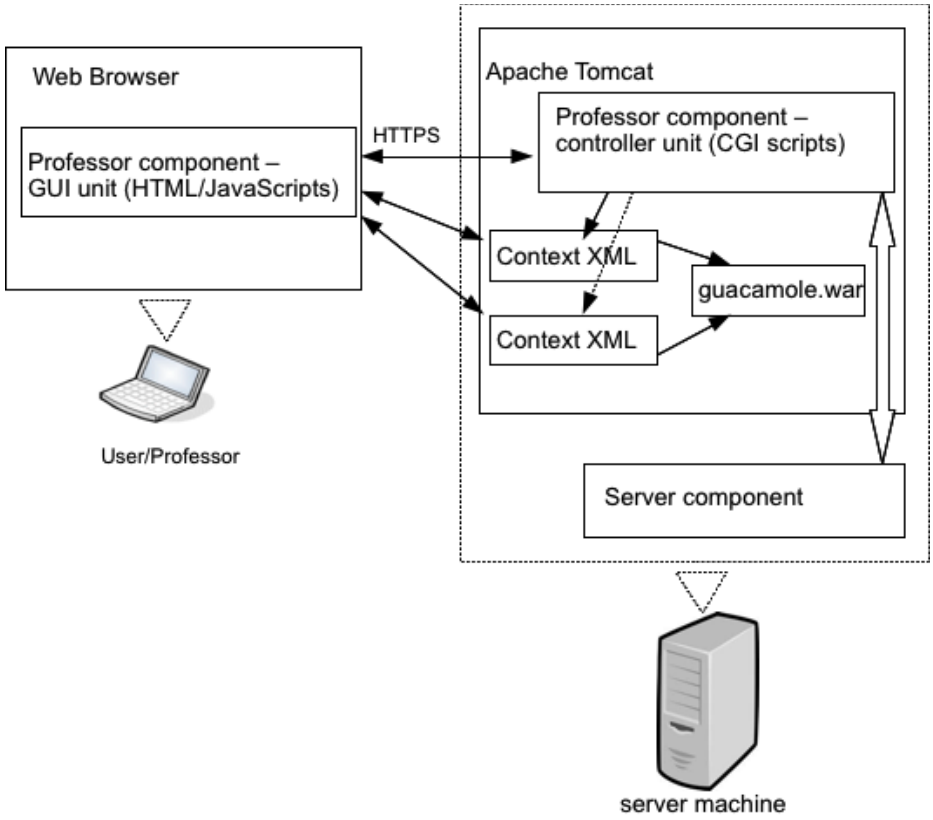


Figure 11. Integration with Guacamole

6.4.5 Implementation reversion for the VNC data traffic

The serverless feature is implemented in VNCMeeting 2.0, and the result is a direct VNC data traffic between the Professor and the Student, without involving the Server. In that implementation, the Professor initiates the SSH connection.

It is possible to keep the implementation in the new version and leave the Student component unchanged. That is, when the Professor component and the Student component have finished exchanging information and are ready for delivering the VNC data, the

Professor component sets up an SSH tunnel by port-forwarding a local port to the VNC server listening port on the Student's machine, and the VNC viewer (i.e. the Guacamole VNC viewer proxy) connects to the local port.

This seems feasible, but in fact the system will be very limited compared to VNCMeeting 2.0. The reason this implementation works in VNCMeeting 2.0 is that, the Professor and Student are assumed to be in the same classroom, and thus in the same IP subnet. The Professor can initiate an SSH connection to the Student directly. In the new version of VNCMeeting, the controller unit of the Professor component resides on the Server machine. An SSH connection can be established from the Server machine to the Student machine only if the Student machine is reachable. Unfortunately this is not true in most situations. A typical setting of the VNCMeeting system consists of a Server machine that is reachable by both the Professor and Student machines while the Professor and Student machines are within a more specific subnet behind a router and the IPs are not routable from the Server machine.

Therefore, this implementation of the Student component needs to be reverted back to VNCMeeting 1.0, in which the Student component initiates the SSH tunnel to the Server machine, not the other way around. The following diagram shows the establishment of the VNC data traffic in the new version.

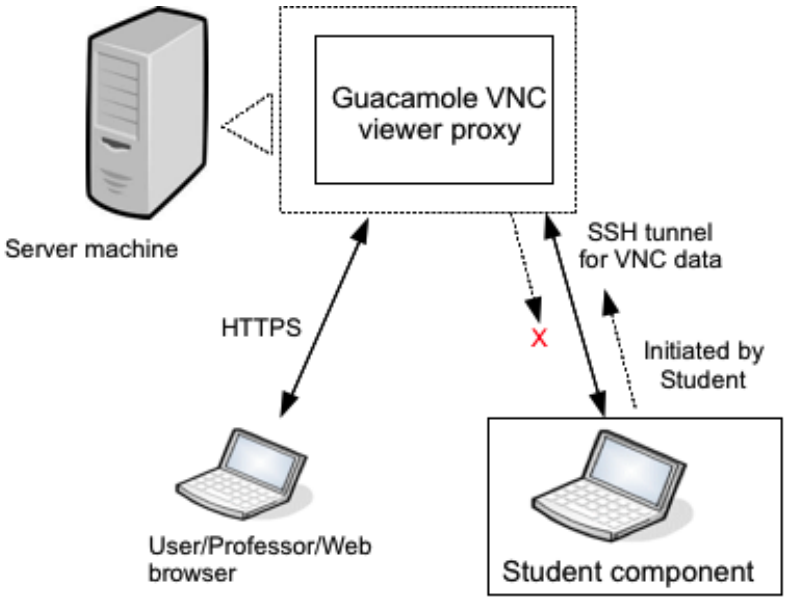


Figure 11. VNC data traffic

6.4.6 Implementation of the controller unit

The controller unit of the Professor component exists in the form of stateless server-side CGI scripts (see 6.4.1, 6.4.2). Its functionality includes parsing requests from the GUI unit, communicating with the Server component and setting up the connection for the VNC data traffic.

Commands.cgi is the entry point for communicating with the controller unit. It handles HTTP post requests, performs operations based on the type of the request, and returns the responses. The expected types are: 'create' for creating a Professor, 'remove' for removing a Professor, 'prepare' for setting up the connection for VNC data traffic, 'teardown' for tearing down the connection for VNC data traffic, 'list' for listing the available students for the Professor.

ServerCommand is a module in the controller unit that is responsible for the communication with Server component. It provides interfaces to interact with the professorcommands program to perform operations on or request information from the Server component: create a Professor, send a touch pulse for a Professor, remove a Professor, get the current Student list for a Professor, get the student info for a Professor, get the next gotop request for a Professor.

ConnectionManager is a module in the controller unit that is responsible for setting up the connection for any VNC data traffic. When a user/professor decides to connect to a student, a request is sent from the GUI unit in the web browser to the controller unit. The ConnectionManager acquires the connection-specific information of the student from the Server component, and uses the information to create the Context XML file under Apache Tomcat. When a successful response is returned to the GUI unit in the user/professor's web browser, the VNC data traffic is ready to start.

6.4.7 Implementation of the GUI unit

The GUI unit of the Professor component consists of client-side HTML/JavaScripts that are loaded onto the user/professor's web browser. The entry point of the GUI unit is a trivial HTML page, and all the functional JavaScript codes are grouped into files and are linked by the main HTML page. Most updates in the GUI are dynamic and are implemented using AJAX (asynchronous JavaScript and XML) and the jQuery JavaScript Library.

The GUI unit mainly provides users/professors with the functionality to communicate with the controller unit, manage student connections, and manage the display component that renders the student's shared screen. Similar to VNCMeeting 2.0, the GUI starts with prompting the user/professor for a name, and then presents a list of available students to the Professor. A polling mechanism is used. A 'list' type request is sent to the controller unit in a fixed interval for an updated list of available students, which also triggers a 'touch' pulse for keeping the Professor alive at the Server component.

As discussed earlier in the paper, Guacamole allows a user to connect to a VNC server and renders in a web browser. This needs to be integrated into the GUI unit. One option is to open a new window for each shared screen, but managing a certain group of windows in a web browse could be difficult because of the lack of standardization across browsers. To achieve better results, HTML iframe is used for the integration. An HTML iframe tag defines an inline frame that contains another HTML document, and it can co-exist with other HTML elements.

In the GUI unit, each shared screen is embedded within an iframe tag that is displayed next to the menu and occupies the majority of the screen space. Only one active shared screen is visible at a time. Through the GUI unit, users/professors can easily switch between active shared screens, as well as adding/removing screens by connecting/disconnecting a student.

With this design, the more advanced features in VNCMeeting 2.0 such as the raising window feature can be easily implemented.

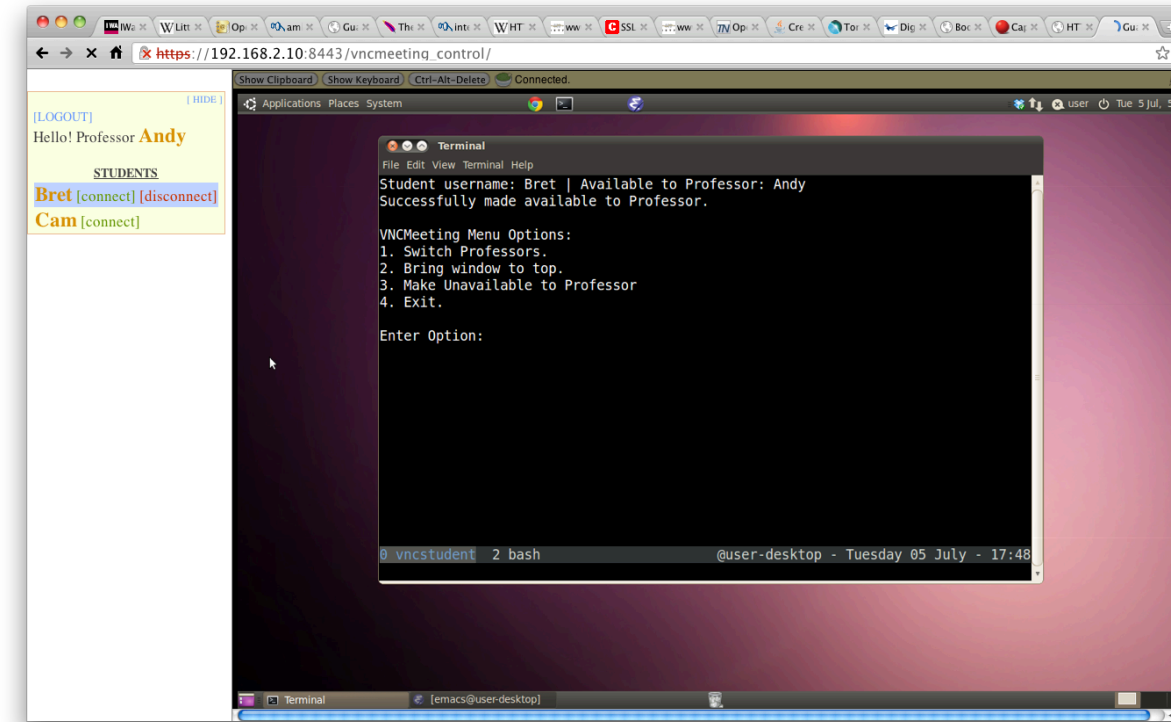


Figure 12. Screenshot of using VNCMeeting to display a student's shared screen in a web browser.

6.4.8 An imperfect solution to the rendering glitch

While a student's shared screen can be rendered correctly within an iframe tag, the flow of the operation to connect to the shared screen is not always smooth.

A user connects to a student's shared screen by pointing the web browser to the Context XML file (see 6.4.4), which is freshly created and deployed under Tomcat. When the Context XML file is first accessed from a web browser, Tomcat checks and copies the actual application files specified in the Context XML to a certain directory. This is called Hot Deployment, which is to deploy a new web application without restarting Tomcat. Unfortunately, accessing a hot deployed application from a web browser is not always successful on the first attempt, which could be affected by the fact that in this system, the time between the Context XML is deployed and the time it is first accessed is usually less than one second. The result of an unsuccessful access to the hot deployed Context XML is the '400 Bad Request' HTTP response from Tomcat.

No solution can be found to eradicate this problem. A workaround solution is implemented. When a student's shared screen is to be loaded into the iframe element, a polling operation is started. It keeps checking the loading status in certain intervals, and reloads the iframe element until the loading succeeds.

6.4.9 GUI unit termination and clean-ups

The VNCMeeting system requires the Professor component to notify the Server component when the Professor leaves the system. In the context of this new system, this needs to take place when the user/professor closes the webpage in the web browser or navigates away from the webpage.

An attempted solution is to send out the 'remove' request to the controller unit silently when the webpage is closed, but the request usually cannot be finished due to the nature of the event ('onbeforeunload'). Another solution is used. When the webpage is to be closed, a warning message pops up to prevent the webpage from being closed and asks the user to 'LOGOUT' first if he/she has not. The user should follow the advice to click the 'LOGOUT' button and then tries to close the page again. Nonetheless, this does not guarantee the user will follow the advice, and this type of ungraceful exit will leave out-dated user files in the system.

In the server component, the 'cleanup' program checks and deletes out-dated files. In the Professor component, a similar cleanup program is created to delete the out-dated Context XML files under Apache Tomcat. It is a separate program from the controller unit (the CGI scripts).

6.4.10 Security

This version of VNCMeeting is able to retain most of the features from VNCMeeting 2.0, but it is also important to maintain the level of security for the system.

Similar to VNCMeeting 2.0, the level of security is ensured by using SSH tunnels: the communication between the Student machine and the Server machine is through SSH forced commands, and the VNC data traffic between the Student machine and the Server machine (the Professor controller unit) is within an SSH tunnel.

Different from VNCMeeting 2.0, the communication between the Professor machine and the Server machine is no longer via an SSH tunnel because the communication is now between a web browser and the Server component. Instead, HTTPS is used. Hypertext Transfer Protocol Secure (HTTPS) is the Hypertext Transfer Protocol (HTTP) running over the SSL/TLS protocol [10]. It provides encrypted communication and secure identification of a network web server.

The Apache Tomcat in this system is configured to use HTTPS over port 8443. The entry point of the GUI unit (the web application) requires a login username/password. This is to authenticate the Tomcat user role that is created for using the web application. All users of

the web application use the same username/password to authenticate. The authentication alone is a form-based authentication and is not secure, but it is secure over HTTPS. After the initial authentication, the rest of the communications between the Professor's web browser and Apache Tomcat during the session is secured by HTTPS.

6.4.11 Insecurity

Although the system is secured by SSH and HTTPS, there are potential security risks.

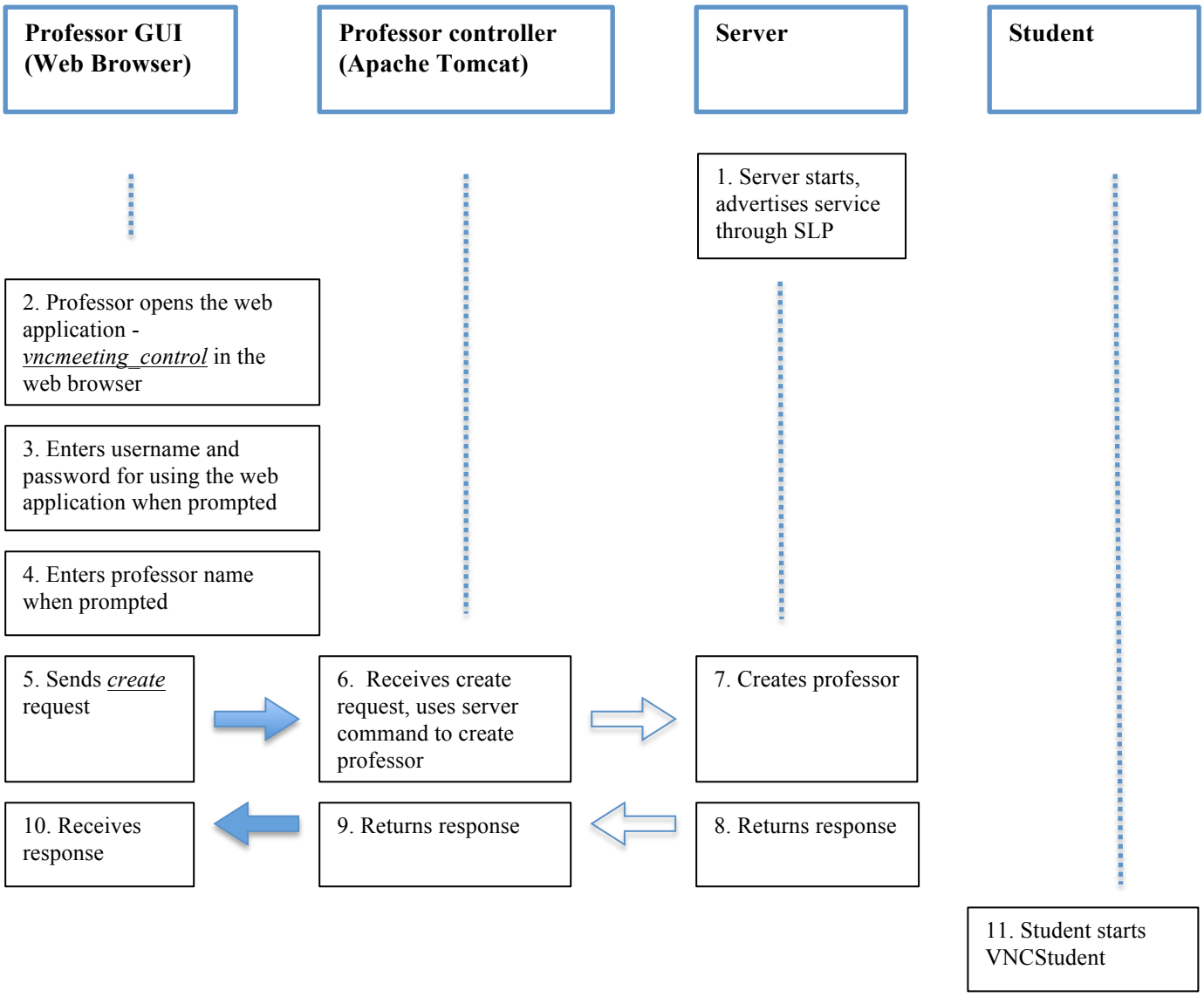
In VNCMeeting 2.0, the Professor component connects to the VNC server by first copying the VNC password file from the Student via SCP, and then uses the password file to connect. All communications involving the VNC password file is between the Student and the Professor. Guacamole allows users to connect to a password protected VNC server, but the current version requires users to specify the password in clear text in the Context XML file. In this new system, the controller unit of the Professor component exchanges information with the Student component indirectly through the Server component, and it includes the VNC password. Therefore, a VNC password is stored in clear text with the rest of the Student profile information in the Server component, as well as appears in the Context XML under Apache Tomcat. This increases the chance of leaking the password.

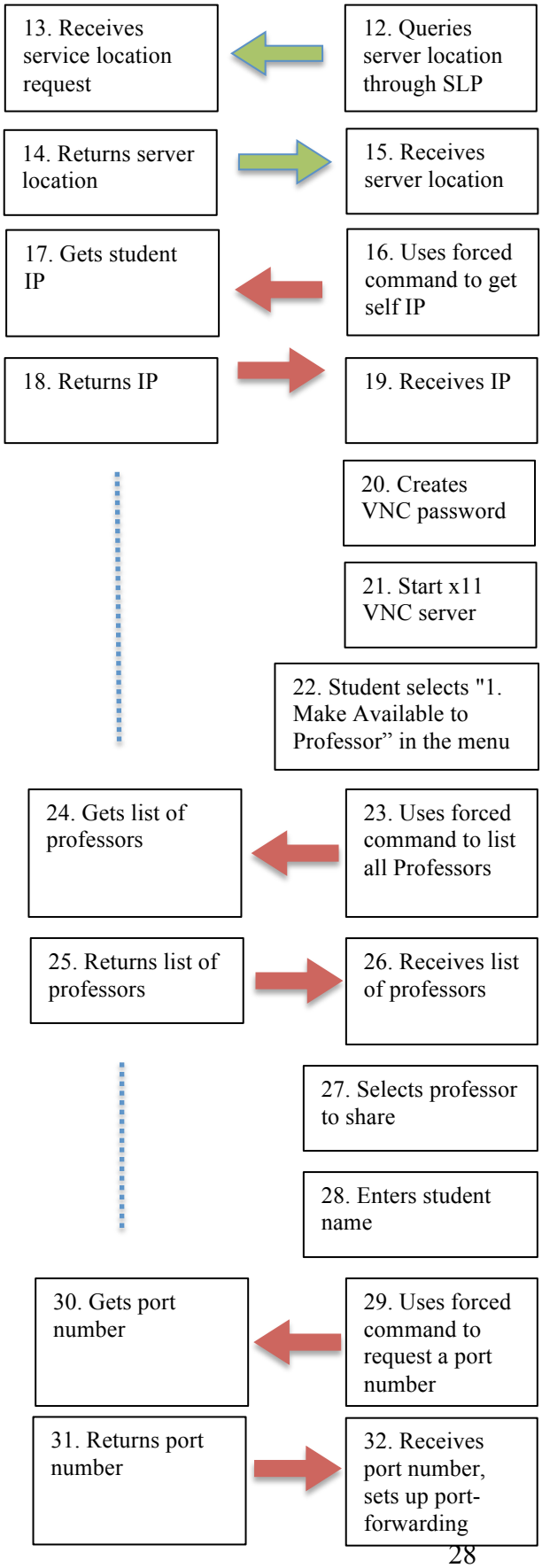
The entry point of the Professor web application is protected with a general password applicable to all Professor users. If a malicious user acquires the password, the user can compromise the system by exploiting the insecurity of the communication protocol. Specifically, the user is able to send custom HTTP request to remove an existing Professor from the system. This issue also exists in VNCMeeting 2.0, in which when a user has the right to communicate with the Server via the *professorcommands* program, the user can send false information to remove any existing Professors. Although the protocol does not allow the Professor user to know the list of other existing users, this is still a potential risk. A more secure Professor authentication mechanism may be used to fix the problem.

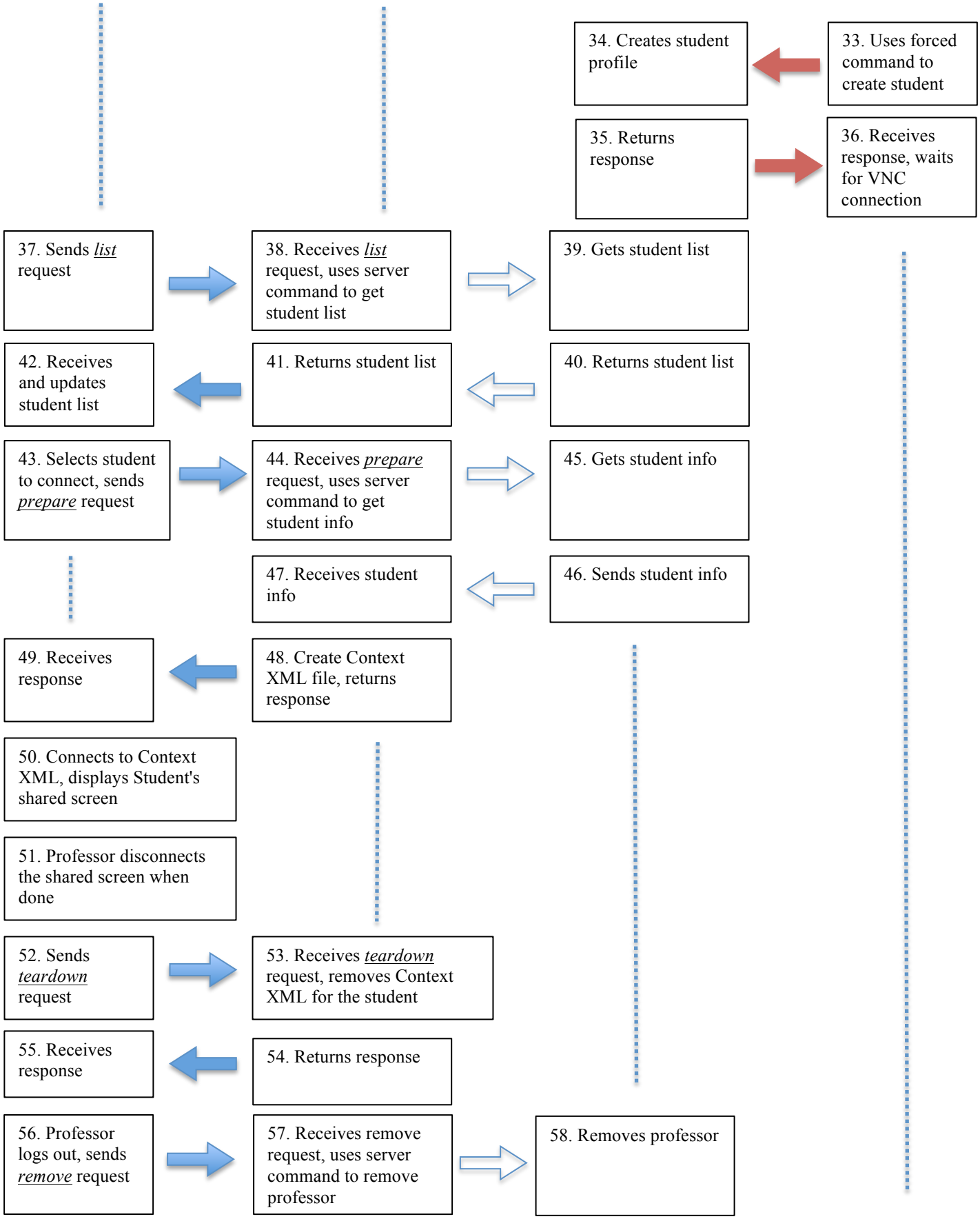
7. Action Sequence

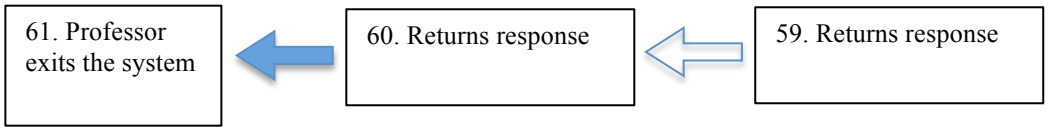
The following action sequence diagram shows the interactions between the components in the system. It is based on the use case described in 3.1, where one Professor uses the system to connect to one Student's shared screen.

Indicator	Meaning
	HTTPS communication
	SSH communication
	Communication within the same machine
	SLP communication




13. Receives service location request
12. Queries server location through SLP
14. Returns server location
15. Receives server location
17. Gets student IP
16. Uses forced command to get self IP
18. Returns IP
19. Receives IP
20. Creates VNC password
21. Start x11 VNC server
22. Student selects "1. Make Available to Professor" in the menu
24. Gets list of professors
23. Uses forced command to list all Professors
25. Returns list of professors
26. Receives list of professors
27. Selects professor to share
28. Enters student name
30. Gets port number
29. Uses forced command to request a port number
31. Returns port number
32. Receives port number, sets up port-forwarding


34. Creates student profile
33. Uses forced command to create student
35. Returns response
36. Receives response, waits for VNC connection
37. Sends *list* request
38. Receives *list* request, uses server command to get student list
39. Gets student list
42. Receives and updates student list
41. Returns student list
40. Returns student list
43. Selects student to connect, sends *prepare* request
44. Receives *prepare* request, uses server command to get student info
45. Gets student info
47. Receives student info
46. Sends student info
49. Receives response
48. Create Context XML file, returns response
50. Connects to Context XML, displays Student's shared screen
51. Professor disconnects the shared screen when done
52. Sends *teardown* request
53. Receives *teardown* request, removes Context XML for the student
55. Receives response
54. Returns response
56. Professor logs out, sends *remove* request
57. Receives remove request, uses server command to remove professor
58. Removes professor


61. Professor exits the system
60. Returns response
59. Returns response

8. Performance evaluation

This section presents some performance data results and some analysis. The performance measurement focuses on the Professor component, and the results are compared to VNCMeeting 2.0 to illustrate how the change in design in the Professor component affects the system in performance.

8.1 Testing Environment

All measurements in this section are performed on one Macbook Pro with an Intel Core 2 Duo 2.4GHz processor and 4GB of memory. The GUI unit of the Professor component is run on Google Chrome browser (v12.0) on the Mac OS system. The Student component, the Server component and the controller unit of the Professor component are run on virtual machines from VMware Fusion 3.1. Each virtual machine has 1 Processor, 512MB of memory, and runs the Ubuntu 10.04 operating system. All virtual machines are interconnected via virtual Gigabit Ethernet interfaces.

8.2 Methodology

8.2.1 Professor connection time

The time elapsed between the Professor clicks the connect button and the student's shared screen is successfully displayed is recorded. This elapsed time is recorded 5 times to generate 5 data points for each of the VNCMeeting 2.0 and the new VNCMeeting. It is assumed that before the measurement starts, a professor has logged into the system and a student is available for the professor to connect to.

8.2.2 Screen Latency

To ensure that the new system still meets the usability requirement, the response time of the interactions with the shared screen is observed, and is compared to VNCMeeting 2.0. The method is to send keyboard/mouse inputs and observe the latency of the actions.

8.3 Results and analysis

	Time for Student's screen to be displayed (sec)				
Data point	1	2	3	4	5
VNCMeeting 2.0	8.5	7.1	9.2	6.6	7.5
New VNCMeeting	6.5	10.2	4.5	13.3	13.6

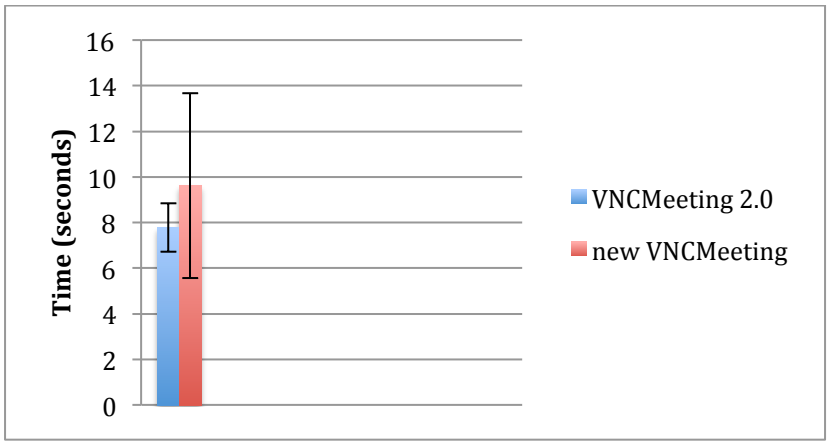


Figure 13. Time for Student's screen to be displayed

The data result table shows the results of the time it takes to successfully display the student's shared screen on the Professor machine. For VNCMeeting 2.0, the average time is 7.78 sec, with a standard deviation of 1.05, and for the new VNCMeeting, the average time is 9.62, with a standard deviation of 4.05.

From the data table, it is shown that the shortest time to successfully display the shared screen for the new VNCMeeting is 4.5sec. This is actually the only case, among the five, that the shared screen is displayed successfully on the first load attempt, and in the other cases multiple reloads are attempted because of the hot deployment issue (see 6.4.8). This suggests that in the new system, it actually takes a shorter time to setup and connect to the VNC server, but the hot deployment issue is unpredictable, and its requirement to reload multiple times affects greatly the overall connection time.

The new VNCMeeting has a better performance in setting up the connection to a student if the reload time is ignored. The reason is that in the new VNCMeeting, the controller unit of the Professor component is deployed on the same server machine with the Server component, and the SSH communication is replaced with direct process communication. More importantly, the new VNCMeeting also avoids the time-costly 'sleep' commands that are used in VNCMeeting 2.0 to ensure the correctness of the SSH communications.

There is no measured data for the screen latency because the latency is usually less than 1 second and it is difficult to obtain precise measurements, instead, only observation is used for the evaluation. Under the testing environment, it can be easily seen that the display of the shared screen in the new VNCMeeting (through Guacamole) has a longer latency than using a regular VNC viewer in VNCMeeting 2.0, but the latency is still likely less than 1~1.5 second. The latency is expected to be even more obvious in the real working environment. This is because in the new VNCMeeting, the VNC data needs to travel from the student machine to the server machine and finally to the web browser on the Professor machine, while in VNCMeeting 2.0, the VNC data traffic is only between the student and the Professor.

8.4 Performance conclusion

Compared to VNCMeeting 2.0, the new VNCMeeting has a slightly longer shared screen connection time and a higher latency, but its performance can be considered close to VNCMeeting 2.0, and is acceptable in most classroom environment.

9. Related Work

9.1 Guacamole 0.3.0

At the time of writing this paper, a new testing version of Guacamole, version 0.3.0 [11], is released. Compared to 0.3.0rc1, which is the version used in this project, Guacamole 0.3.0 features efficient native components and an extendable framework.

Guacamole 0.3.0rc1 is a Java-only version that the server-side VNC viewer proxy is implemented completely in Java. Guacamole 0.3.0, instead, uses a server-side proxy written in C, and a tunnel written in Java, and the performance is almost as responsive as a native VNC viewer.

Guacamole 0.3.0 could replace Guacamole 0.3.0rc1 in the VNCMeeting system to possibly provide a better performance. The disadvantage would the extra dependencies for compiling and installing the C-based VNC viewer proxy, which unlike Java, is not platform independent.

9.2 noVNC

noVNC is another web-based VNC client/viewer that uses HTML5 [12]. It is a more pure web-based technology than Guacamole, in the sense that it uses WebSocket to connect to the server so that the full VNC client implementation is in JavaScript/HTML5, as opposed to Guacamole which relies on a server-side VNC viewer proxy implemented in Java or C. Nonetheless, noVNC is still limited by the nature of WebSocket that WebSocket is not TCP, and it is not able to communicate directly with the VNC server. Therefore, noVNC still requires a WebSocket proxy server that translates between the WebSocket and the VNC server.

noVNC could be a better candidate than Guacamole for the VNCMeeting system for being a more pure client-side web-based VNC viewer. Nonetheless, noVNC was not released before the topic of this project is decided, and the performance and the integration compatibility with VNCMeeting are not investigated.

9.3 ThinVNC

ThinVNC is a commercial pure-web Remote Desktop solution [13]. It uses a pure HTML5 based client to connect to a server via HTTP/s. ThinVNC is not a traditional VNC as it does not implement the RFB protocol. Instead, it uses web standards: AJAX, JSON and HTML5

to display the remote desktop. On the server side, it requires the ThinVNC server be installed on the screen-sharing machine.

ThinVNC server is only for Windows operating systems. It also includes a component Library called ThinVNC SDK that allows users to add Sharing capabilities to certain Windows Applications, making the application windows accessible for custom remote support procedures, peer-to-peer application collaboration, authorization procedures, etc.

10. Conclusion

VNCMeeting 2.0 provides a solution for students to easily and securely share computer screens with Professors in a classroom environment. The new VNCMeeting introduced by this paper changes the architecture of the Professor component, and it allows Professors to gain access to the system more easily while most of the usability features and the level of security are maintained from VNCMeeting 2.0. It also enhances the system by adding service discovery protocol support to allow the Student machine to find the Server machine automatically. Although the performance of the new system is slightly worse than VNCMeeting 2.0, it is acceptable for a classroom environment, and making the Professor component accessible from a web browser should provide an overall positive usability value.

11. References

[1] http://en.wikipedia.org/wiki/Virtual_Network_Computing

[2] Richardson, Tristan, and Stafford-Fraser, Quentin, and Wood, Kenneth, and Hopper, Andy. "Virtual Network Computing" IEEE Internet Computing, Volume 2, Number 1, (January/February 1998).

[3] Wong, Roxanna. "VNCMeeting End-to-End Security & Serverless Operation", University of Alberta, July 2010

[4] Ye, Mengtao. "VNC Meeting - Raising Window and Switching Session", University of Alberta, July 2010

[5] Guacamole, http://guacamole.sourceforge.net/

[6] Apache Tomcat, http://tomcat.apache.org/

[7] RFC 2608, Service Location Protocol, Version 2, June 1999

[8] http://www.openslp.org/doc/html/IntroductionToSLP/index.html

[9] http://tomcat.apache.org/tomcat-5.5-doc/deployer-howto.html#A_word_on_Contexts

[10] RFC 2818, HTTP Over TLS, May 2000

[11] http://guac-dev.org/

[12] http://kanaka.github.com/noVNC/

[13] http://www.thinvnc.com/thinvnc/html5-vnc.html

12. Appendix

12.1 VNCServer Files

File		Description
cleanup.c		Cleans any old files from the /users/ folder. Always running when the server is considered available.
createprofile.c	*	Creates Professor and Student profiles.
forcedcommands.c	*	Functions used throughout all the forced commands.
getip.c		Gets the IP of the other end of the SSH connection.
getport.c	*	Gets the port of the other end of the SSH connection.
getprofessorkey.c		Prints the key of a professor.
getstudentinfo.c	*	Retreives and prints out the information for the given Student.
gotop.c		Functions to handle the gotop request from students.
listusers.c		Lists either all Students of a given Professor or all available Professors in the VNCMeeting system.
professorcommands.c		Handles forced command calls from Professor
removeuser.c		Removes the profile of a Professor or Student
studentcommands.c	*	Handles forced command calls from Student
touchuser.c		Update given profile's modification time. Used to keep profile alive.
vncmeetingServerStart.sh	+	Script to start the server: starts slpd and the cleanup process.
vncmeetingServerStop.sh	+	Script to stop the server: closes slpd and the cleanup process.

Files without denotation are files from VNCMeeting 2.0.
Files denoted with * are files from VNCMeeting 2.0 and are modified for this version.
Files denoted with + are new files.
All files should be placed under the same directory before using ‘make’. Example deployment directory: ~/bin/VNCMeeting/.

12.2 VNCStudent Files

File		Description
chooseprof.c		To get the name of the chosen/preferred Professor.
openandcapture.c	*	Opens a pipe for the given command and then captures the output into a predetermined file.
randompassword.c	*	Generate random password
readconfig.c		Reads and interprets the configuration file.
readoutput.c	*	Reads and interprets the output files.
slpGetServiceLocation.c	+	Use OpenSLP to find the network address of the Server
vncstudent.c	*	Main program for user interactions.

Files without denotation are files from VNCMeeting 2.0.
Files denoted with * are files from VNCMeeting 2.0 and are modified for this version.
Files denoted with + are new files.
All files should be placed under the same directory before using ‘make’. Example deployment directory: ~/bin/VNCStudent/.

12.3 Professor Component Files

12.3.1 Web application (aka vncmeeting_control, GUI unit) Files

File	Description
index.html	Main application entry
control.css	CSS style sheet
WEB-INF/cgi/commands.cgi	CGI script to handle AJAX request from the JavaScript codes
WEB-INF/cgi/formatter.rb	Help functions to format CGI response
javascript/control.js	Functions to communicate with the CGI script
javascript/frame-control.js	Functions to manage the display component that renders the student's shared screen
Javascript/hideMenu.js	Functions to manage the side menu in the GUI
Javascript/student-manager.js	Functions to manage student connections in the GUI

These files are deployed as parts of the Tomcat web application called vncmeeting_control. vncmeeting_control should be deployed using the included ant build file. Configuring the build attributes before deployment is necessary.

12.3.1 Controller Unit Files

File	Description
connection_manager.rb	Functions to setup the connection for VNC data traffic
constants.rb	Constants definition
guacamole_control_cleanup.rb	The cleanup program that should be run separately.
professor_manager.rb	Functions to cleanup inactive professor
server_command.rb	Functions to communicate with the Server component.
util.rb	Help utility functions.

These files are deployed using the same ant build file that deploys the web application. By configuring the build attributes, these files can be deployed close to the CGI scripts, or on a separately directory. Example deployment directory: ~/bin/vncmeeting_professor/.