VMS – Virtual Meeting System

M. Vidanapathirana, IEEE Student Member
Department of Computer Science and Engineering, University of Moratuwa, Sri Lanka.

madhawa.13@cse.mrt.ac.lk

***Abstract*— The Project VMS is a solution developed to allow virtual meetings between two remote business meeting rooms. The solution is designed so as to complement the facilities provided by a typical meeting room environment. The system consists of a single Server subsystem, two Facilitator subsystems and Presenter subsystems. In the production environment, each meeting room would have a central Facilitator subsystem of its own, used by the meeting facilitator who controls the meeting. Each Facilitator subsystem is connected to Presenter subsystems installed in PCs of meeting participants inside the room. The Server subsystem connects two Facilitator subsystems in order to materialise the Virtual Meeting between two meeting rooms. The system supports sharing of presenter screens and voice between meeting rooms. Additionally, the facilitators maintain a shared list of tasks which could be assigned to presenters, whom would be notified upon assignment. In materialising aforementioned, the system uses Client-Server Architecture between connected subsystems. The major output of the project can be identified as the three subsystem applications developed, which together would revolutionize Virtual Meetings. This can be identified as a major step taken forward from the use of internet video calls to materialize virtual meetings.**

Keywords —Virtual Meeting; Business Meeting Room; Client-Server Model; Central Subsystem; Presenter Screen; Presenter Voice; Shared Tasks; Facilitator; Presenter

I. INTRODUCTION

The business community started using Virtual Meetings for internal communication purposes; making emails constrained to Formal Communication. Virtual Meetings introduced many advantages to business over conventional email such as "reducing time taken to resolve an issue due to real-time communication" and "enhancing clarity of communication due to inclusion of tone of communication".

The motivation of developing the VMS system is to address some drawbacks of existing systems that provide Virtual Meeting facilities. Most of the existing solutions are not tailor-made for Business Virtual Meetings, thus not complimenting infrastructure inside a meeting room. Most solutions relay on 3rd party infrastructure, thus raising concerns related to privacy of information. Furthermore, most solutions available for virtual meetings does not represent the role of Facilitator in a meeting. The VMS solution addresses the above mentioned issues through the tailor made solution it provides for Business Virtual Meetings.

VMS – Virtual Meeting System makes it very convenient for formal meetings between offices of a business. VMS makes it possible to cast the screen of presenter through the projectors available at two connected meeting rooms. It also transfers the voice of presenter to the remote meeting room. The aforementioned functionality is jointly controlled by the two facilitators at two meeting rooms. Furthermore, the system provides functionality for the meeting facilitators to keep track of meeting objectives through Shared Task Management System. The Shared Task Management System allows facilitator to assign responsibilities to participants of the meeting.

The VMS System provides aforementioned functionality through a combination of 3 subsystems, Server subsystem, Facilitator subsystem and Presenter subsystem. The person who is in charge of a meeting room is known as the facilitator. The facilitator uses a machine with Facilitator subsystem to control the meeting. This machine is connected to the projector and the speaker system of the meeting room. The participants inside a meeting room connect to the Facilitator subsystem of the meeting room using the Presenter subsystem application installed to their Laptop computer. Two Facilitator Systems are interconnected by Server, thus making a Virtual Meeting between two remote meeting rooms.

This paper describes the design and implementation of above mentioned VMS system. The Section II provides a literature review of Virtual Meeting Systems. The Section III describes the design and functionality of the System. The Section IV describes matters related to implementation of

VMS System. The Section V describes testing procedures and results of VMS System. The Section VI suggests probable future improvements of the system.

II. Literature Review

A. Theoretical Aspects

The developed VMS System is based on streaming of user perceptions between remote locations. The perceptions considered here are screen of users' PC and voice of user. Fig. 1 explains this process in diagram.

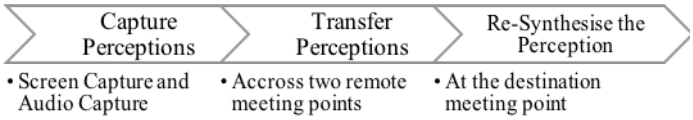

Fig. 1. Process of Streaming Perceptions

The aforementioned functionality is implemented by the livestream component which is developed specially for project VMS. The VMS system uses and controls the livestream component to properly exchange perceptions between sources and destinations.

B. Comparison to Existing Systems

The functionality of VMS at its core is similar to an internet conference video calling service. However, in case of VMS, the features are customised to suite the virtual meeting requirements of the businesses. However, in this short literature review, VMS would be compared against both Virtual Meeting solutions and Internet Conference Video Calling solutions.

Skype [1] provides a conference video calling feature to its premium customers as a paid service. This feature can also be used for Virtual Meetings. However, this is best suitable for meetings with very less number of participants at each location. AnyMeeting [2] is a commercially available web-based virtual meeting solution. It is focussed on joining multiple meeting rooms with very less number of particpants at each meeting room. Join.me [3] is another online service that is focussed upon screen sharing, audio sharing and text chat among a group of participants. Similar to AnyMeeting, this solution is not tailor made to use the physical infrastructure of a meeting room.

All the solutions mentioned above are not tailor made to use the physical infrastructure of a meeting room. They do not recognise Presenters and Facilitator as two different subjects with different responsibilities. Neither of the solutions mentioned above are capable of utilising the projector of meeting room as required for a business meeting. Neither of them provide a mechanism for assignment of responsibilities to meeting participants.

Cisco WebEx [4] is an industry grade solution for Business Virtual Meetings. It offers a free tier which supports upto three participants per meeting. However, WebEx doesn't facilitate the role of facilitator which would be required to maintain proper control structure of a meeting.

The developed VMS solutions surpasses existing products in the market due to its ability to complement the physical infrastructure of a business meeting room. Additionally, the VMS solution provides benefits such as centralised control and Shared Task Management system. Furthermore, VMS solution encourages user to use their own infrastructure to host the server instead of 3rd party service provider. This ensures security of business information.

III. System Models

A. System Requirement

The VMS system consists of 3 subsystems, Server subsystem, Facilitator subsystem and Presenter subsystem. These subsystems jointly deliver the requirements mentioned in this section.

The VMS system is required to manage connection between its subsystems. This includes pairing mechanism between server and facilitator in addition to the Passkey based connection mechanism between Facilitator and Presenter. The system is also required to facilitate connection and disconnection of presenters from a live meeting. The system should also support sharing of screen and voice of presenters connected to the system. The shared screens are displayed at the projectors connected to two Facilitator subsystems at two meeting rooms. The voice of the presenter should also be transferred to the remote meeting room via the speakers connected to the remote facilitator subsystem. Apart from aforementioned communication, VMS also provide proper control facilities to the meeting. The presenters can raise share requests, requesting opportunity to share their screen or voice. The facilitator should accept these requests in order to allow a presenter to be active. Alternatively, the facilitator is able to change the active screen sharer/active speaker according to his/her will among the group of presenters connected to itself. Additionally, the facilitators are provided with the facility to assign responsibilities to presenters, who are notified upon assignment.

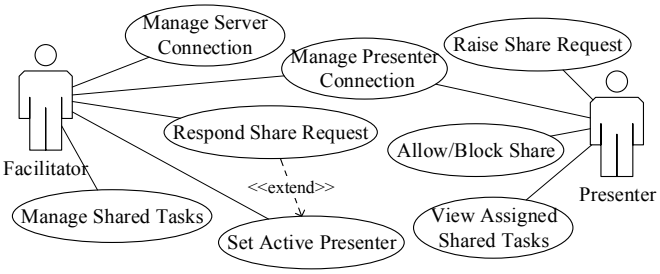

Fig. 2. Main Use Case Diagram of VMS System

Fig. 2, identifies major use cases of Facilitator and Presenter. Additionally, there is a System Administrator who is responsible to manage the Server, not identified in the aforementioned diagram. The use case "Allow/Block Share" associates the consent of presenter to let their screen/voice be shared. This requirement fulfils privacy requirement of presenter. When the facilitator accepts a share request raised by a presenter, it actually sets the active presenter. Thus, the use case "Respond Share Request" extends the Use case "Set Active Presenter".

Considering the non-functional requirements, the system is required to work in a network with Server to Facilitator bandwidth not more than 8 Mbps. The system should maintain time gap between two screen shares less than 3 seconds. The system requires the connection between subsystems to be authenticated. The VMS system should be capable of running across multiple operating systems. Thus, its implementation is undertaken in JAVA.

B. System Design

The VMS system consists of three subsystems, Server subsystem, Facilitator subsystem and Presenter subsystem. The subsystems are interconnected using the Client Server Architecture. The Facilitator subsystem acts as the client of server in Facilitator Server connection. The Presenter subsystem acts as the client of Facilitator subsystem in the Facilitator Presenter connection. These Client-Server connections are based on JAX-WS WebService Technology which uses SOAP messages for communication. In the context of this paper, the term Console is used to identify WebServices that are uniquely published for the use of a particular client. (e.g. – Facilitator Console 1 published for Facilitator 1 by Server to undertake Facilitator 1 – Server communication)

Additionally, the system architecture can be broken down into low level architecture and the high level architecture. The low level architecture is engraved into livestream component which handles transfer of screen captures and audio captures across subsystems. Livestream also handles capture of perceptions (screen and voice) at source and synthesis of perceptions at destination. It follows the Pipe and Filter Design Pattern. The high level architecture manages the functionality provided by livestream component for streaming perceptions.

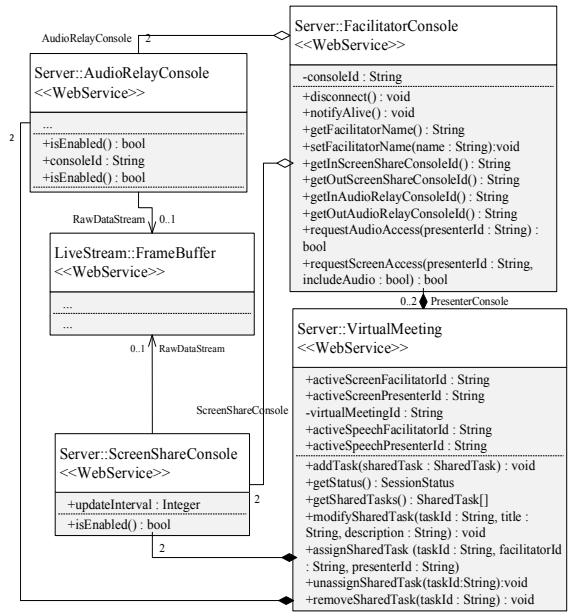

Fig. 3**.** Class Diagram of Services offered by Server Subsystem

Fig. 3 demonstrates main classes of Server subsystem. Each Facilitator subsystem connected to the Server is given a dedicated FacilitatorConsole. The facilitators raise requests from Server using the FacilitatorConsoles. The WebService VirtualMeeting contains state variables of the virtual meeting. It is being shared between both Facilitators. Additionally, FacilitatorConsole contains reference to two AudioRelayConsoles and ScreenShareConsoles that allows transfer of perceptions between Facilitator and Server. The ScreenShareConsoles and AudioRelayConsoles mentioned above are contained in the VirtualMeeting WebService. Both AudioRelayConsoles and ScreenShareConsoles utilise FrameBuffer of livestream component to hold streaming data.

Fig. 4 demonstrates class diagram of major classes of Facilitator subsystem. These classes are also WebServices offered to Presenter Subsystem.

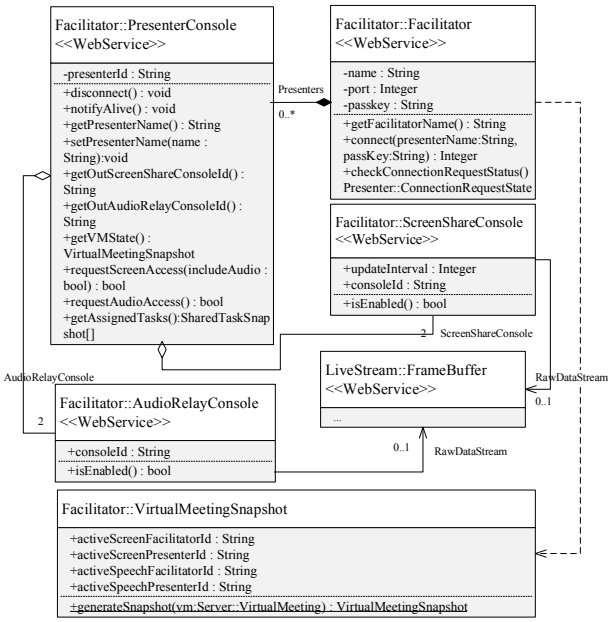

Fig. 4**.** Class Diagram of Facilitator Subsystem

The Facilitator WebService is the WebService used by Presenter subsystems to establish connection with Facilitator. Upon successful connection establishment, the Facilitator subsystem creates and offer a unique PresenterConsole WebService to each Presenter.

VirtualMeetingSnapshot is a static copy of VirtualMeeting WebService provided to presenter subsystem upon request through getVMState method. The method getAssignedTasks of PresenterConsole returns the list of tasks assigned to Presenter. The PresenterConsole contains reference to ScreenShareConsole (discussed previously) and AudioRelayConsole (discussed previously) objects in order to allow presenter to transmit perceptions to facilitator.

The key process of VMS system can be identified as the Control Loop Process. Two subsystems Facilitator and Presenter utilise their own Control Loop processes. The control loop processes control exchange of audio and screen between subsystems.

Fig. 5 demonstrates Control Loop of Facilitator subsystem.

The Control Loop firsts read VM Status variables from the server and then adjust the subcomponents of livestream hosted by Facilitator subsystem to undertake proper data transfer. The ToServer Multiplexer directs active presenter's data to server. The receivers receive screen and audio inputs from server and synthesize them through projector and speaker system. It also notifies presenter subsystems on whether they should transmit to facilitator.

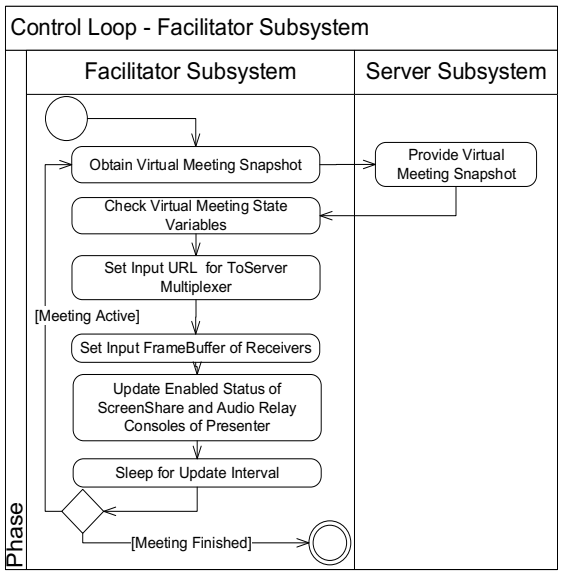

Fig. 5**.** Activity Diagram of Facilitator Control Loop

IV. SYSTEM IMPLEMENTATION

A. Implementation Procedure

The VMS system is implemented using three JAVA 1.7 executable applications based on Swing UI framework. The 3 executables represent the 3 subsystems of the system. The coding is done entirely using IntelliJ Idea IDE and user interfaces are designed using the Swing UI Builder of Idea.

The common functionality used by all 3 subsystems are separated into a separate module called Foundation. This module exposes WebServices such as ScreenShareConsole and AudioRelayConsole which are used by all 3 subsystems of the system.

Each subsystem contains two modules, Core module and UI module. The Core modules (e.g. Server module, Facilitator module, and Presenter module) implement core functionality of subsystems. They expose APIs for the UI modules. The core modules depend on the Foundation module to obtain common functionality. The UI modules (Server-UI module, Facilitator-UI module and Presenter-UI module) are Swing based Executable JAVA applications. The Server-UI module is capable of operating in both Swing UI and Command Line mode.

The subsystems utilise JAX-WS Web Services for the communication between subsystems. The Server and Facilitator subsystems expose JAX-WS WebService Interfaces to their respective clients Facilitator subsystem and Presenter subsystem. The URLs used to host the WebServices are based on the format http://0.0.0.0:[port]/[console_type]/[console_id] and http://0.0.0.0:[port]/[console_type]/[console_id]. The console_id is a random generated string which is unique to each console. The port is provided at the initial configuration of the system.

The persistent storage mechanisms were implemented to the UI layer modules as they are convenient to be separated from core modules. The core modules are focussed on network based functionality. The Two modules Server and Facilitator uses persistent storage for purposes such as storage of Pair Keys. The core modules use methods and events to communicate with UI layer in order to read and update persistent storage. The persistent storage is based on Serializable classes of Java. The passwords exchanged between subsystems or stored in persistent storage are encrypted using SHA-256 [5] algorithm exposed by MessageDigest API of JAVA. Additionally, the persistent storage files are encrypted in order to safeguard the pair keys stored in them. The encryption is based on the Advance Encryption Standard (AES) [6] available in javax.crypto.

The implementation methodology followed a parallel development of 4 modules Foundation, Server (Core module), Facilitator (Core module) and Presenter (Core module). The 3 latter modules mentioned above were initially tested using prototype user interfaces. At the latter stage of development, the 3 UI modules were developed.

The largest storage of materials used by VMS system is the storage of its pair keys. The pair keys are generated by the server upon a pair request from Facilitator subsystem. Then, the two keys are stored in the encrypted persistent storage of both subsystems.

B. The Algorithm

The Control Loop algorithms can be identified as the core algorithms of VMS system. The Control Loop controls flow of screen-share/audio-relay data across subsystems. Fig. 6, demonstrates the main control loop of Facilitator subsystem.

```
Begin
Start Loop
   Call FacilitatorConsole->NotifyAlive()
   Set vm := virtualMeeting.getSnapshot()
   ProcessScreenShare(vm, facilitatorId)
   ProcessAudioRelay(vm, facilitatorId)
   updateSharedTasks(vm, facilitatorId)
   if vm.status == adjourned
        Raise event meeting adjourned
        Break Loop
   If no connectivity error incurred above
        Notify Connectivity Manager that
   _Server Connection is alive
        Raise Control Loop Cycle
   _Completed Event
   Sleep(1000 milliseconds)
Do Loop
End
```

Fig. 6. Facilitator Control Loop Pseudo-code

Fig. 7 demonstrates the ProcessScreenShare support function used by Facilitator Control Loop. The function ProcessAudioRelay of Facilitator subsystem is implemented similar to above mentioned ProcesScreenShare function. The aforementioned pseudo-codes are explained in their relevant comments.

```
Function ProcessScreenShare(vm,facilitatorId)
Begin
//Identify active screen share presenter. Enable its
Screen Share Console. Disable other Screen share
consoles.
For pConsole in facilitator.presenterConsoles
   Set pConsole.enabled = False
End Loop
Set selectedPC := facilitator.presenterConsoles
   _where presenterConsole.consoelId ==
   _vm.activeScreenFacilitatorId
Set selectedPC.screenShareConsole.enabled = True
//Set screenSwitcher to obtain input of selectedPC
If selectedPC is not null
   Set screenSwitcher.InputURL := URL of
   selectedPC FrameBuffer
Else If selectedPC is null
   Set screenSwitcher.InputURL := NULL
End If
//Update screenSwitcher to send frames to server
If outputScreenShareConsole.enabled == true and
   _screenSwitcher.running == false
   Call screenSwitcher.start()
Else If outputScreenShareConsole.enabled == false
_and screenSwitcher.running == true
   Call screenSwitcher.stop()
End If
//Update screenReceiver to receive screens from
Server, if required
If inScreenShareConsole.enabled == true and
   _screenReceiver.running == False
   Call screenReceiver.startReceiving()
Else If inScreenShareConsole.enabled == false and
_screenReceiver.running = true
   Call screenReceiver.stopReceiving()
End If
End
```

Fig. 7. Facilitator Screen Share Control Loop

```
Begin
Start Loop
   Call presenterConsole->notifyAlive()
   Set serverAcceptsAudioShare :=
   _audioRelayConsole.isEnabled()
   Set serverAcceptsScreenShare :=
   _screenShareConsole.isEnabled()
   //Start/Stop Screen Capture
   If serverAcceptsScreenShare == True And
   _allowedScreenShare == True
        Call screenCapture.startCapture()
        Raise event SCapture Started
   Else
```

```
        Call screenCapture.stopCapture()
        Raise event SCapture Stopped
    End If
    //Start/Stop Audio Capture (Similar to above)
    If no connectivity error incurred above
        Notify Connectivity Manager that
    _Facilitator Connection is alive
    Raise Control Loop Cycle Completed Event
    Sleep (1000 milliseconds)
Do Loop
End
```

Fig. 8. Presenter Subsystem Control Loop Algorithm

Fig. 8, demonstrates Control Loop of Presenter subsystem. The Control Loop determines whether the Presenter subsystem should transfer audio captures and/or screen captures to Facilitator subsystems. The variables allowedAudioShare and allowedScreenShare determine whether presenter has given consent to share his/her screen and/or voice.

C. *Main Interfaces*

Fig. 9, demonstrates the Virtual Meeting tab of Facilitator Control Panel. This tab is used to control the virtual meeting. The share requests are displayed in the left panel of user interface. The centre panel is used to manage the Shared Task Management System. The right panel is used to determine which presenter is allowed to share screen or speech. The finish meeting button is used to adjourn the virtual meeting.

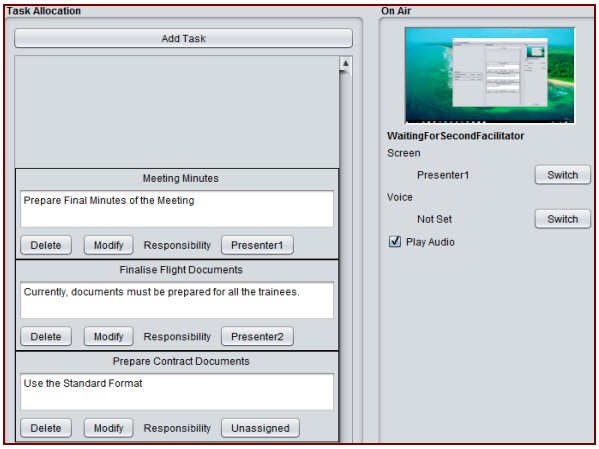

Fig. 9. Virtual Meeting Tab of Facilitator Control Panel – Centre and Right Section

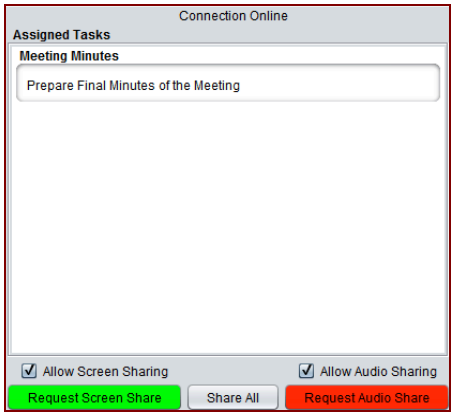

Fig. 10. Presenter Subsystem Main User Interface

The Presenter subsystem user interface demonstrated in Fig. 10. This interface is used by participants of virtual meeting to share screen/speech of themselves. The assigned tasks to the presenter are displayed in the Presenter subsystem user interface. The Allow Screen Sharing and Allow Audio Sharing check boxes are used by the presenter to notify the consent of presenter to share his/her screen. The sharing would not happen if the consent is not provided.

V. System Testing and Analysis

The testing techniques used for VMS system are unit testing, system integration testing, configuration testing, user interface testing and performance profiling. The unit tests and system integration tests were written using TestNG [7] testing framework. User interface tests were undertaken using Marathon ITE UI testing framework [8] on all subsystems of VMS system. Tests were written to cover connection establishment, screen/audio share requests, screen/audio transfer and Shared Task Management System.

Configuration tests were undertaken on all subsystems by varying configuration such as Operating System and Network Configuration. Ubuntu Linux, Mac OS X and Microsoft Windows were used for Operating System testing.

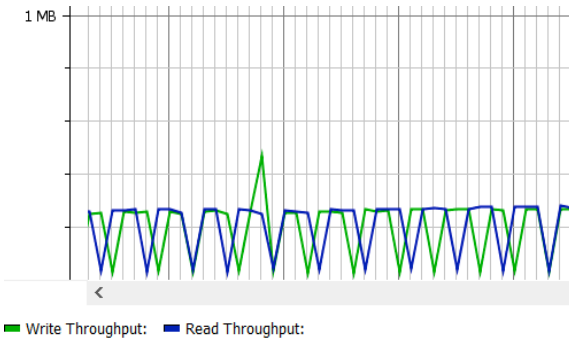

Fig. 11 Network Usage of Server with Two Facilitator Connections

Performance profiling were undertaken using JProfiler [9] Java Profiler. Fig. 11, demonstrates network usage of Server subsystem involving Server Facilitator network communication over Wide Area VPN Network. All 3

Systems performed exceptionally well in the tests undertaken.

VI. Conclusion and Future Work

The VMS system developed can be identified as a competitive candidate among the solutions that provide virtual meeting facility. The system performs well within enforced constraints such as network bandwidth limit. Additionally, the VMS system is capable of complementing the physical infrastructure available at a general business meeting room environment. The Presenter subsystem developed is lightweight and can be easily installed in the participant's machines. The user interface of Presenter subsystem is simple. It contains only 3 sections "Assigned Tasks List", consent checkboxes and share buttons. Furthermore, testing of Presenter subsystem on Laptop machines has indicated that most Laptop machines have sufficiently sensitive inbuilt Microphone systems to capture the voice of presenter. The Facilitator subsystem is also made user friendly by providing step-by-step instructions to user. Additionally, the system provides details of active presenter to the audience through the projector screen. Most importantly, the system has provided total control of the system to facilitators. The Shared Tasks Management System is a shared resource among both facilitators allowing both facilitators to allocate tasks to any presenter. The Server subsystem is developed in two modes, UI mode and console mode. The console mode is capable of operating in a headless server environment such as a cloud hosted virtual machine. Additionally, the server is capable of handling multiple consecutive virtual meetings.

As future improvements of VMS system, it is possible to improve the server subsystem to be able to handle multiple simultaneous virtual meetings. It is also possible to introduce features to record virtual meetings. The bandwidth used for screen share can be reduced by introducing motion image compression technology. This would make sure that screen sharing can be undertaken at a higher frame rate. Currently, VMS system relies on encryption provided by VPN network of user to provide security to Server – Facilitator connection. However, as a key future improvement, encryption can be introduced to this link, making VPN infrastructure not necessary. The pair-keys can be used as encryption keys of Server Facilitator connections. As per the current implementation, the Server subsystem behaves as a single point of failure. This could be eliminated by improving the Facilitator subsystems to automatically connect to a backup server subsystem upon unavailability of primary server. In this process, the two Facilitator subsystems should re-initiate a virtual meeting with backup server. Afterwards, they should update the state of virtual meeting in backup server to suite the last known state of primary server.

Virtual meetings has revolutionised internal correspondence of businesses. VMS introduced by this paper can be identified as a major breakthrough in this evolvement, where business virtual meeting was made closer to physical meetings.

References

[1] "Skype," [Online]. Available: https://www.skype.com/.

[2] "AnyMeeting," [Online]. Available: https://www.anymeeting.com.

[3] "JoinMe," [Online]. Available: https://www.join.me/.

[4] Cisco, "Cisco Webex Meetings," Cisco, [Online]. Available: https://www.webex.com/products/web-conferencing.html.

[5] "How To Generate SHA256 Hash in Java," [Online]. Available: http://www.quickprogrammingtips.com/java/how-to-generate-sha256-hash-in-java.html.

[6] "Java : Encryption and Decryption of Data using AES Algorithm with example code ~ Code 2 Learn," [Online]. Available: http://www.code2learn.com/2011/06/encryption-and-decryption-of-data-using.html.

[7] "TestNG - Welcome," [Online]. Available: http://testng.org/.

[8] "Test Automation for Java/Swing Applications » Marathon," [Online]. Available: https://marathontesting.com/.

[9] "Java Profiler - JProfiler," [Online]. Available: https://www.ej-technologies.com/products/jprofiler/overview.html.