Mobile Command Center (MCC)

*Virtual Command Center Mobile Device Interfacer*

Project Researcher: Vidur Prasad

Project Advisors: Shea Tolson, Dr. Rob Williams, Dr. Sivaram Gogineni



Mobile Command Center Final Paper

Vidur Prasad

MCC

Dayton, Ohio, United States

[prasad.12@wright.edu](mailto:prasad.12@wright.edu)

Shea Tolson

YATE2014

Dayton, Ohio, United States

[sheatolson@](mailto:sheatolson@)gmail.com

Dr. Robert Williams

Tec^Edge Innovation and Collaboration Center Director

YATE2014

Dayton, Ohio, United States

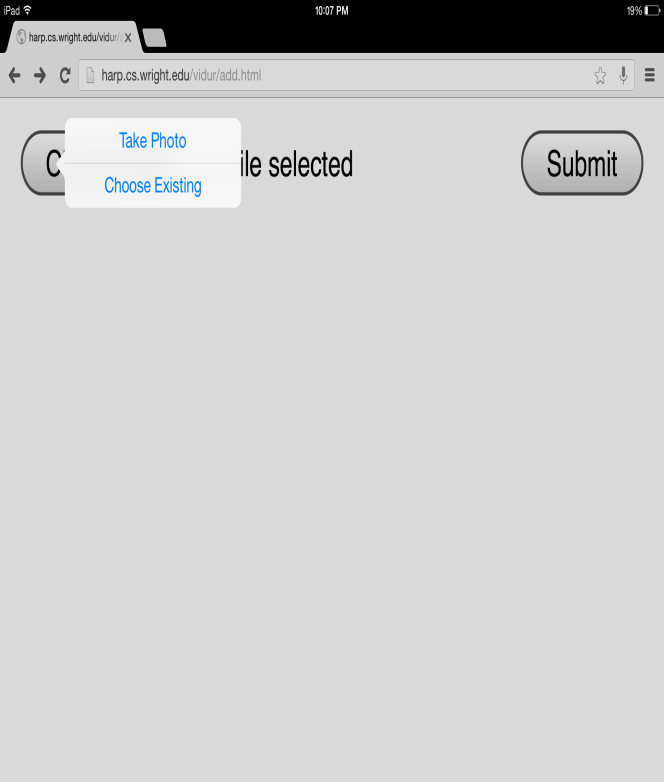
robert.williams@wpafb.af.mil

**ABSTRACT**

The goal of the Mobile Command Center project was to allow a virtual command center located inside the Discovery Lab Outreach Center to send and receive text and image data with a mobile device in an efficient way. This would allow for commanders to receive different pieces of data inside of a command center, process it inside of the virtual world, and then send out data to all the deployed units that had mobile devices. The data transfer is done through multiple steps; with each member receiving the data and then sending it to the next step. All the data going from a mobile device to the command center and vice versa takes the same path, but were just sent in different channels. The data from either end were first sent to a server, which saved the data. The recipient of the data would then ping the same server, which would display the data that was saved earlier. This system not only allows for data to be sent, but also to be stored to allow all parties to go back and view the past communications. The first set of steps explains how images are communicated, and the second set of steps explains how text is communicated. The flowchart, shown at the end of this paper, shows the steps that the data takes in both methods.

**STEP 1: Mobile Device to Server**

The server that was used to act as the middle man was an Ubuntu server with Apache services running, that also had PHP and MySQL capabilities. The address of this server is <http://www.harp.cs.wright.edu>, and is maintained by Wright State University. The first address that needs to be addressed to upload a PNG image file is <http://www.harp.cs.wright.edu/vidur/add.html>. This simple HTML form allows a user on a computer to upload an image, or a user on a mobile device to take or upload a PNG image. Figure 1-1 shows a screenshot of the interface shown to a mobile device user when uploading a media file.

Figure 1-1

**STEP 2: Saving Media on Server**

This HTML file then sends the uploaded image to a PHP script located on the server called insertimagesql.php. This script receives the PNG file, opens the actual file and converts it into text, and then saves this in a MySQL database that is also located on Harp: the name of the server at Wright State University. When the images are saved onto the MySQL server, an ID number is assigned sequentially, allowing for the command center to view specific pictures.

**STEP 3: Viewing Media from Command Center**

The Command Center is powered by the Open Simulation software, which has a feature called Media Prim. A Media Prim can be assigned to a face of an object such as a cube, turning that face into a functioning Internet browser. This is very useful because it allows HTTP POSTs to be made in the form of URL requests. The URL request in this specific situation is sent to Harp at <http://www.harp.cs.wright.edu/vidur/viewimagesql.php>. A specific image is requested by sending the ID number through the URL, in a PHP file, the syntax is ?id=1. The image is specifically parsed by a PHP file that reads in the image data stored on the MySQL database, then packages it back into a PNG file, and then sends it to the Media Prim. The Media Prim then displays the picture that allows the command center to finally view the picture that was taken or sent by the mobile device.

**STEP 1: Sending Text Message to Server**

Since the Open Simulation software allows in-world scripting through LSL, Linden Scripting Language, HTTP POSTs can be carried out to internet servers. This allows for an object to be created in-world that sends a string to Harp based on the text that is inputted into the script. This is actually carried out by sending a message to <http://harp.cs.wright.edu/vidur/themiddleman.php>. The actual message can be sent by sending the variable through the HTTP Post or by adding a ?content = “message” style of sending messages through the URL. This allows for messages to be transmitted from mobile devices by entering the message into the URL bar in the browser.

**STEP 2:Saving Text Message on Server**

The text message is saved on the server by writing it to a database on Harp using MySQL. This is done through the same PHP script that receives the input, so the process is relatively simple.

**STEP 3: Viewing Text Message from Command Center**

The script that reads the text message is located at [http://www.harp.cs.wright.edu/vidur/ androidinterfacer.php](http://www.harp.cs.wright.edu/vidur/%20androidinterfacer.php) . Accessing this website from a browser will display all the communication that has taken place between the Command Center and the mobile device. This allows for someone to see all the past communications to understand what has been done so far. Since the script uses the HTTP echo protocol to return the text, simply sending a HTTP Post to the website from the Command Center will allow the text message to be displayed in the chat window.

**Conclusion**

The MCC project has allowed for the creation of a Command Center that can bring leaders from many different places who can then effectively communicate with the workers with mobile phones. The capability to send media and text will be very useful in a situation where commanders would like to effectively communicate with assets on the ground while still being able to stay in the Open Simulation world, which allows for numerous other logistical benefits. Overall, this project has shown the data transmission can be done through simple server relays from a mobile device to an Open Simulation based command center.

