November 2015

User Interface Design And Forensic Analysis For DIORAMA, Decision Support System For Mass Casualty Incidents

Jun Yi
University of Massachusetts Amherst

USER INTERFACE DESIGN AND FORENSIC ANALYSIS FOR DIORAMA DECISION SUPPORT SYSTEM FOR MASS CASUALTY INCIDENTS

A Thesis Presented

by

JUN YI

Submitted to the Graduate School of the
University of Massachusetts Amherst in partial fulfillment
of the requirements for the degree of

MASTER OF SCIENCE IN ELECTRICAL AND COMPUTER ENGINEERING

September 2015

Department of Electrical and Computer Engineering

USER INTERFACE DESIGN AND FORENSIC ANALYSIS FOR DIORAMA DECISION SUPPORT SYSTEM FOR MASS CASUALTY INCIDENTS

A Thesis Presented

by

JUN YI

Approved as to style and content by:

_____

Aura Ganz, Chair

_____

C. Mani Krishna, Member

_____

Russell Tessier, Member

_____

Christopher V. Hollot, Department Head
Department of Electrical and Computer Engineering

ABSTRACT

USER INTERFACE DESIGN AND FORENSIC ANALYSIS FOR DIORAMA DECISION SUPPORT SYSTEM FOR MASS CASUALTY INCIDENTS

SEPTEMBER 2015

JUN YI

B.S., NANJING UNIVERSITY OF SCIENCE AND TECHNOLOGY

M.S.E.C.E., UNIVERSITY OF MASSACHUSETTS AMHERST

Directed by: Professor Aura Ganz

In this thesis we introduces the user interface design and forensic analysis tool for DIORAMA system. With an Android device, DIORAMA provides emergency personnel the ability to collect information in real time, track the resources and manage them. It allows the responders and commanders to mange multiple incidents simultaneously. This thesis also describes the implementations of commander app and responder app, as well as two different communication strategies used in DIORAMA. Several trials and simulated mass casualty incidents were conducted to test the functionalities and performance of DIORAMA system. All responders that participated in all trials were very satisfied with it. As a result, DIORAMA system significantly reduced the evacuation time by up to 43% when compared to paper based triage systems.

TABLE OF CONTENTS

Page

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

In March 2011, a huge earthquake hit the coast of Japan and triggered a powerful tsunami. The whole incident caused more than 15,000 deaths and 2,000 people were missing. Mass casualty incidents (MCIs) have caught more attention of the society during the past few years. Disaster is one of the most common reasons causing a MCI. It is defined as "a serious disruption of the functioning of a society, causing widespread human, material or environmental losses which exceed the ability of the affected society to cope using only its own resources" by the United Nations Disaster Management Training Program (UNDMTP)[16]. Though many pre-event preparations can be done to prevent an incident or at least reduce the casualty when one happens, natural disaster is unpredictable. In this thesis, we mainly focus on the incident response and rescue process-the period between the occurrence of the incident and the accomplishment of all evacuations. During this period, Emergency Medical Services (EMS) plays an important role by providing effective, responsible pre-hospital care[16]. Their performance influences the chance of survival among victims.

When responders arrive at the disaster area, victims that need immediate treatment will be rescued and treated at once. In this stage the primary job for them is triage, which is "the process of determining the priority of patients' treatments based on the severity of their condition" [15]. This rations patient treatment efficiently when resources are insufficient for all to be treated immediately. When a paramedic finds a patient, he or she will examine the patient (e.g. by pulse) and determine the seriousness or priority of this person. There are four conventional classifications [16] currently used in emergency response with corresponding colors and numbers:

1. Red / Immediate / Priority 1: The casualty requires immediate medical attention and will not survive if not be treated immediately
2. Yellow / Delayed / Priority 2: The casualty requires medical attention within 6 hours. Injuries are potentially life-threatening, but can be waited until the immediate casualties are stabilized and monitored
3. Green / Minimal / Priority 3: "Walking wounded", the casualty requires medical attention when all higher priority patients have been evacuated, and may not require stabilization or monitoring
4. Black / Expectant / Dead: The casualty is expected not to reach higher medical support alive without compromising the treatment of higher priority patients

To identify and track the triaged victims, paper tags are mostly used in current patient tracking system[16]. After patients are categorized by emergency responders, they are tagged with a visible indicator (triage tags) of their priorities. Several commercially available tag systems are used such as METTAG and Multi-Tag. Figure 1 is an example of triage tag. As the figure shows, each tag is preprinted with a unique number, a section for patient information, and a section with tear-off strips to categorize the patient's current condition.

The evacuation process begins when some victims are triaged with priorities. The casualties with the highest priority are moved to a triage category-specific collection point for further on-site treatment and/or transportation. The walking wounded (green) are readily separated from more seriously injured casualties through good crowd communication and control; most of these casualties could be treated in an urgent care center, clinic, or private physician offices. Dead victims are suggested to be transferred to an isolated location.

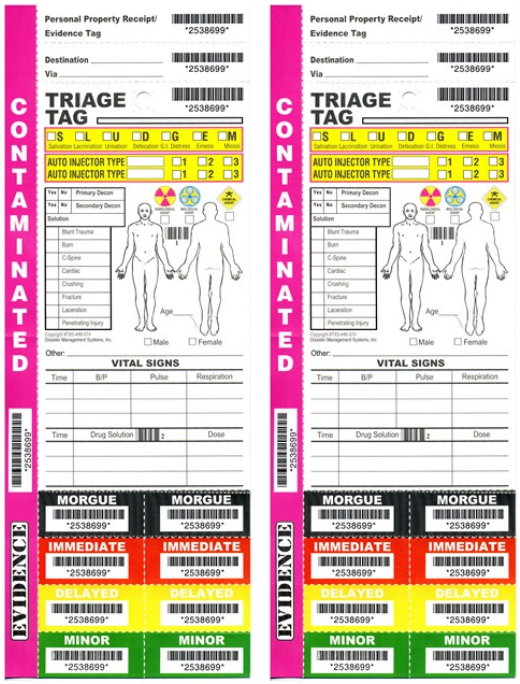


Figure 1: Sample Triage Tag

The paper triage and evacuation technique that is being used currently has several flaws[13][16]:

1. The triage tags can only provide limited information about the condition of the patient. No special information is recorded (such as the source and destination of the patient for a transfer)
2. It's also difficult to modify the card or tag when the condition is changed, for the information written by ink cannot be easily erased and modified; on the other hand, tear-off triage tags may be changed easily by patients or family members in an effort to upgrade the triage classification and expedite medical care or evacuation.
3. Triage tags are often discarded, even though they should be a part of the medical records. They are often in an awkward shape to fit in the medical record and may not be recognized by the medical records clerks.

4. No real time location information of responders and victims. The current triage systems are not able to track victims and responders in real time during the triage and evacuation stages
5. They provide no collaboration tools between the incident commanders and the responders

Our contributions include designing two user-friendly emergency rescue apps with some forensic analysis tools, and an enhanced algorithm for area dividing.

CHAPTER 2

RELATED WORK

There are several research projects related to MCI response. Their proposals substitute or enhance the paper triage tags using digital devices such as tablet PCs or PDAs as well as using different network communication technology (satellite, ZigBee, etc.) to connect each device.

E-Triage system[1] separates the incident area into two parts: on-site segment and the disaster-safe segment. The core network is built in disaster-safe segment and all communication between these two segments is based on satellite link. Also in this system, responder should carry a small cabin-size suitcase which contains many communication devices such as PDA and satellite antenna. Although the satellite link offers the system abilities to work even in the area without any local network, it still takes time and effort to carry and set up during evacuation.

MANET project[2] implemented an autonomous sensor network using ZigBee. In this system, the hanging triage card is replaced by a patient tag equipped with a passive RFID module. A handheld computer is used to identify tags and send the information to the network. Central computer displays the position and the number of the victims in a map after gathering all this information. An analysis and visualization system is also provided in the central computer for the incident personal to analyze the scenario. This system is very similar to our project, except that MANET network is built up using ZigBee communication. This might be a good solution when there is no reliable network, but this kind of short-range communication is requiring lots of router nodes when the incident area is larger. In this case, pre-installation takes lots of time and resources, which might reduce the efficiency of the rescue.

Another e-Triage system described in [3] proposed another interesting aspect of displaying an incident: 3D panoramic view. Using the intra-site view, each patient and staff is displayed as a 3D object on a 3D panoramic photographic background. Commander can monitor the vital signs and location of patients through this view and notice sudden changes of patient conditions. This 3D panoramic view is created using three or more fisheye images of a certain point, converted to a cube map, and displayed in the intra-site view. Therefore, 3D display of patients and staff provides the commander a clearer and more visualized way to understand the situation, with the cost of setting up a camera and taking lots of pictures.

The project proposed by Eva et al. [4] is more similar to our system. The basic components of their system include a set of PDAs for paramedics, a set of tablet PCs for commanders, and a multi-touch table for incident management officials. The functions of PDAs and tablets act as the same role with those in other projects: collecting data, representing them in a map. The multi-touch table is placed in command post. Due to the big screen of the table, officers can work collaboratively and have a better basis for making decisions to allocate their resources. In addition, other authorities and organizations like the police or fire brigade could also work in collaboration on this table.

CrowdHelp is a client-server system proposed in [5]. It's designed for patient assessment that uses mobile electronic triaging accomplished via crowd-sourced information. The system offers its users the ability to determine and submit their physical conditions through a crowd-sourced network as well as provides them a list of emergency treatment places. To access its service, users need to download and install the application; then a login process is required, asking them to log in using e-mail or social network account. After that the app will collect all the censor readings (e.g. GPS) with permission and send them to the server. At the same time, logged in users will receive warning push messages for forthcoming dangerous events. All the data collected is saved in the

CrowdHelp server and can be offered to emergency professionals through a machine learning software.

In [6], the authors proposed a location based disaster preparedness system that uses Disaster Management Server (DMS), GPS supported Android mobile phones, and Android Cloud to Device Messaging (C2DM) service to provide visual and audio disaster warning and evacuation help. In this system, the disaster prone area and the user details are stored in a third party server: Disaster Management Server. Regional weather offices can access the database to update the disaster information and the user of the Android application can get notification of any update. The user's location is sent to the DMS and a Ray-casting algorithm is run to determine whether the user is in the area of disaster. The Android Cloud to Device Messaging server (provided by Google) is then used to push notifications to the registered devices. When the application finds its user in a disaster zone, it will give him a notification and draw the shortest path to the nearest shelter or safe area. Meanwhile, the application will start another service to track the user through GPS and sends the data to DMS so that the responders can rescue him from the disaster area.

Wang Ze-gen et al. proposed an earthquake cooperative emergency rescue auxiliary system[7]. In this system, technologies of GIS, satellite navigation and GSM are used for navigation and real-time sharing of the rescue information and the cooperative rescue. This rescue system consists of a command center and some mobile terminals. The former is in charge of the map management and analysis, controlling all other functions; the latter stores all geographic information for navigation and rescue plotting. In this system, a rescue plotting symbol library is developed and works independent of the base map data; GSM message module is also used to transmit rescue information. This system is flexible and convenient on rescue plotting and data transmission, however it makes no effort on improving the efficiency of data collection and situation analysis.

A telemedicine support system[8] is also developed to provide specialized expertise in local / remote affected regions of a MCI. It includes lots of mobile hardware and software components, which is similar to DIORAMA. In this system, responders use tablets running Windows 8 and NFC technology to triage patients; video camera system and a local communication network is also installed meanwhile. For the critical victims, laptops / tablets running Windows 8 are used as individual equipment to monitor their conditions. Moreover, an on-site container / tent coordination center provides the functions such as management of on-site activities, processing medical data, receive calls and provides location data etc. The use of NFC technology and video camera system in this system are extremely helpful for victim classification and keeping records; but they cannot give responders or commanders any visual situational awareness of current incident.

In article [9], S. L. Toral et al. proposed a wireless in-door system for assisting victims and rescue equipment in a disaster management. Through Bluetooth communication and electronic equipment, this system is able to manage evacuation routes, or obtain information about victims' location and status whenever a building collapses. Bluetooth beacons and PDA terminals are used in such system. When a disaster alarm is activated, auxiliary beacons collect the terminals around them and offer them evacuation routes; principle beacons receive information form auxiliary beacons to identify movements in breakdown buildings. Rescue team can use the information from the principle beacons and even establish an audible link with victims. Although some pre-event setups are necessary in this system, Bluetooth beacons can greatly improve the efficiency of finding and rescuing victims in an indoor environment.

SUMMIT[10] is a modeling & simulation software environment that is used for small and large-scale exercises to accelerate scenario planning, provide scientifically grounded scenario data, and enhance the realism and common operating picture. SUMMIT is based on

Google Earth and equipped with sufficient tools for incident reality simulation and data analysis. With its strong library, users as exercise planners or exercise participants are able to create science-based scenarios, visualize data and get simulation results. Since SUMMIT is specifically designed for exercise and training, it difficult to apply this system into real incident. Modeling and pre-configuration is time-consuming, which makes our DIORAMA more suitable for real emergencies.

[11] and [12] also provide us some other ways of network communication. The former, DistressNet[11] uses different wireless sensors to form different ad-hoc networks in different levels like "TeamNet" or "AreaNet". Communication between sensors and responder devices is established within all these sub-networks. [12] proposed another system architecture that all devices communicate using XMPP protocol. Mobile entities in the system detect their neighbors automatically inside their ad hoc networks and a wireless radio mast carried by a truck will be used when local network breaks down. Both systems can provide reliable backup network when local communication infrastructures are completely destroyed, at the cost of deploying expensive devices and pre-installations though.

Most of those systems require complicated pre-installations such as deploying huge amount of nodes, beacons, or satellite antennas. All these preparations reduce the flexibility of their systems and waste lots of resources. Responders will have to carry some heavy and sometimes expensive devices. In the other hand, none of those proposed projects has developed any user-friendly interface or proper method to visualize the data collected in an incident.

DIORAMA uses only Android smartphones and tablets to run with no pre-installation needed. Responders can use their own phones and a bunch of cheap D-tags (will be described in 3.2.3) to perform every action. DIORAMA also provides both responders and

incident commander visualization techniques used to generate the required situational awareness that can significantly improve the emergency rescue efficiency.

CHAPTER 3

INTRODUCTION OF DIORAMA

DIORAMA is a system that could increase the incident response efficiency by providing an information collection infrastructure for mass casualty incidents. It offers the commander and responder a more visualized way to understand and analyze the scenario in order to make better decisions. Compared to the paper triage and evacuation system mentioned above, DIORAMA has several advantages:

- DIORAMA provides real time tracking of victims and responders using a mobile tracking system carried by each responder
- DIORAMA introduces collaboration tools between the responders and incident commander
- DIORAMA provides novel visualization tools to responders such as Augmented Reality
- DIORAMA also monitors and records all information during the incident, capable of generating detailed archives and forensic reports

Figure 2 shows the architecture of DIORAMA system. It contains three main components: Android smartphone and tablets running DIORAMA applications, RFID smart tags and readers, and the DIORAMA server.

During the incident, each responder that performs triage and evacuation carries an Android smartphone running the DIORAMA responder app, a RFID reader (denoted the DM-Track in the figure) and a bunch of D-tags. D-tag is a pack of objects including a paper triage tag, a passive RFID tag and a NFC tag. When a responder finds an un-triaged victim, he/she takes out a D-tag, scans it with the smartphone, selects a priority and hangs this D-tag on the victim's arm. The information about the tag id, priority and current GPS location will be sent to the server and stored in the database. Meanwhile, the responder's RFID reader

keeps tracking signals of all active RFID tags in surrounding area and the localization algorithm running on DIORAMA server uses this information to determine the location of each patient.

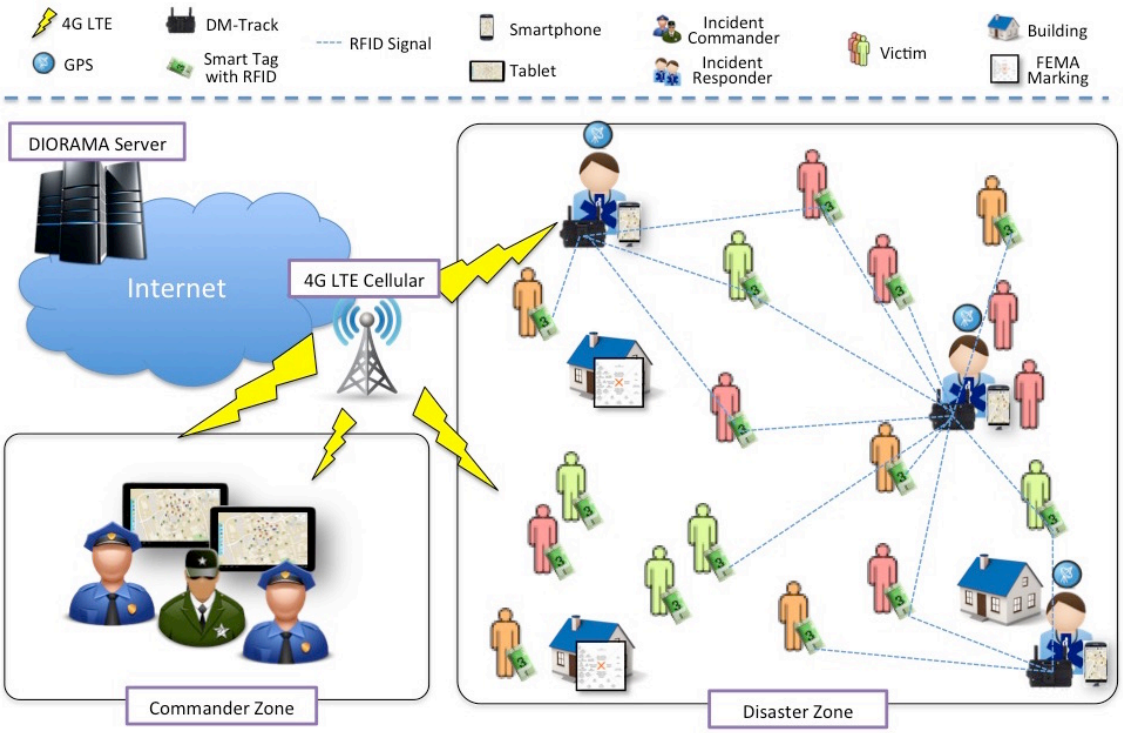


Figure 2: DIORAMA Architecture

While responders are doing triage/evacuation, incident commanders direct the incident using Android tablets that run the DIORAMA IC app. This application receives the information sent from all responders and displays it in multiple modes. Commanders can send several types of commands to responders to conduct the rescue process as well.

Most functions in DIORAMA IC app and responder app will be described in this section. The basic functions of commander app will be illustrated in 3.1 . Section 3.2 covers the functions of responder app. An automatic area-dividing algorithm implemented in commander app will be introduced in Section CHAPTER 6.

3.1 Commander App

Incident Commanders' job is to monitor the scenarios and conducting incident responses. Commander app is equipped with functions like displaying point of interests (POIs), assigning tasks to responders and reviewing the previous events.

3.1.1 Application Flowchart

The flowchart of commander app is shown in Figure 3. When the app starts, users have to log in to the map view① using their responder names. After an incident is selected or created, all POIs including victims, responders and hazards created by other emergency personnel will be shown as icons in the map (section 3.1.2). To assign tasks to responders, the app allows the incident commander to create variety kinds of commands and send to any specific responder② (section 3.1.3). Timeline mode③ and report mode④, or called forensic analysis tools, can be used to replay and analyze historical data of the past incidents (section 3.1.4).

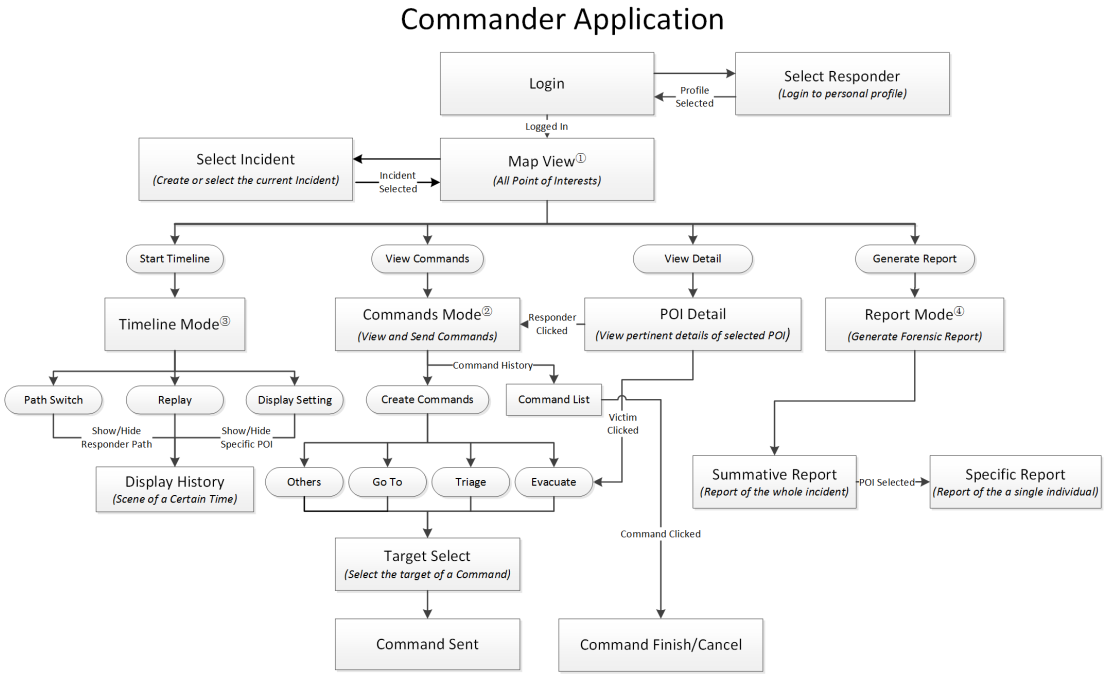


Figure 3: Flowchart of Commander App

3.1.2 Different View Types of Visualization

In DIORAMA Commander app, different view types are used to illustrate the situation of the incident. Icon view is used when the commander needs to focus on the details of a specific area; tile view is used to display a bigger picture of the incident; satellite view can be turned on to show the reality of an area and other auxiliary view types are designed for some more special purposes.

1) Icon View

Figure 4: Icon View of Command App

Figure 5: Legend of Commander App

Figure 4 is the icon view of commander app. In this view, Google Map is used as the bottom layout and icons are overlaid on the map, representing all POIs (victims, responders, etc.) in the incident. When a new POI is created (e.g. new victim triaged) or a current POI is updated (e.g. responder moved), the corresponding icon on the map will be changed

accordingly in real time. In such way commander is able to monitor the distribution and priorities of all patients as well as to track all responders.

Figure 5 shows the meanings of different icons. In the figure, victim icon consists of background color and a number in the center. The color varies from red to black according to the priority of a victim. The number or the letter "D" (indicating "dead") at the center of the icon is a redundant cue to indicate priority in addition to its color. Hence users can easily identify the priority of each patient: either by color or by number.

The map style can also be changed from street map to satellite map (Figure 6) or terrain map, which helps commanders to understand more about the scenario even if they never been to that place.

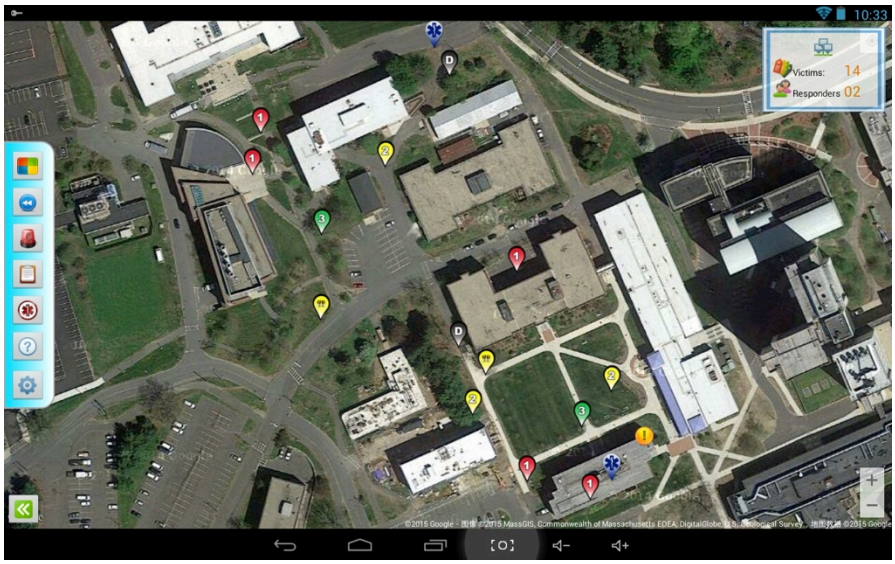
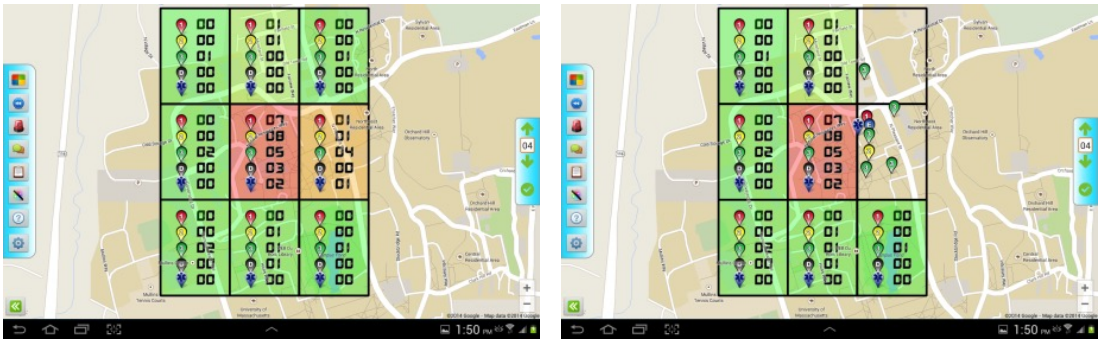


Figure 6: Satellite View of Commander App

2) Tile View

Tile view is another aspect of displaying POIs. In icon view when the number of victims in the map grows larger, separated icons can no longer tell the commander what's happening in the field. Information is overflowed and disconnected segments become meaningless. Therefore, tile view is developed to display a more summative picture of the scenario.



(a) (b)

Figure 7: Tile View of Commander App

Figure 7 shows the interface of the tile view. In Figure 7a, the incident area is divided equally into several rectangles (or tiles). In each tile, icons are hidden from the map and only the quantities of victims in different priorities are displayed. When using the tile mode, commander app sends the parameters (e.g. number of tiles, incident id) to server and requests for the tile data. At the server end, it scans the spatial database and sends back the information of each tile. Receiving all this information, commander app then draws all the edges, icons, and numbers on the map.

The tiles are also filled with certain translucent colors, which are calculated according to the number of victims in each tile. Because victims have different priorities, it's inappropriate to value them equally. In the commander app, a weight value w_{pri} is assigned to a certain priority type and a total score W_{total} is used to indicate the total weight of each tile. W_{total} is calculated through the formula:

$$W_{total} = \sum_{\text{all priorities}} w_{pri} \times num_{pri}$$

The value of w_{pri} is determined according to the seriousness of victims and the total weight indicates the severity of incidents in the tiles. After the calculation for all tiles, an average value of tile weight is kept in the system. Tiles with weights equal to the average value are marked as yellow; tiles with larger weights are marked as red (but different depths) and tiles with smaller weights are marked as green.

Additionally, a tile can be “opened” or “closed” to show or hide the icons (shown in Figure 7b). A single click on a tile will open it and long clicking on the opened tile will close it again. In an opened tile, all POIs including victims and responders are displayed as icons while the ones outside this tile are kept hidden. This is a combination of icon view and tile view, helping commanders keep their focuses on specific details of one or several tiles.

3) Other Views

There are some other views designed in commander app such as certain priority view or evacuated victims view. All of these views mentioned in this section are designed to visualize the huge quantities of information provided by other clients, to help commanders analyze incidents. Since displaying patients and responders is the primary job of the commander app, it’s meaningful to offer users more options to monitor a scenario.

3.1.3 Command Module in Commander’s App

Commanding is another primary function of DIORAMA commander app. Command module allows an incident commander to assign certain tasks to all responders in the field. The most useful commands in an incident response include: triaging, evacuating, re-triaging an area, evacuating a specific victim, going to a position and extracting a trapped victim etc. In current paper triage system, commander assigns all tasks to responders through walky-talky [2]. In this communication process, areas are explained in the words like “north of the building A” and individual patients are hard to be described. Therefore, language description is inaccurate, especially when responders are located in a place where they have never been before. Searching and rescuing in an incorrect area may lead to the waste of human resources, or even falling into a dangerous situation. In DIORAMA, all these issues are solved through the visualized command system.

When use clicks on command button, the command panel will be displayed above the map as shown in Figure 8.

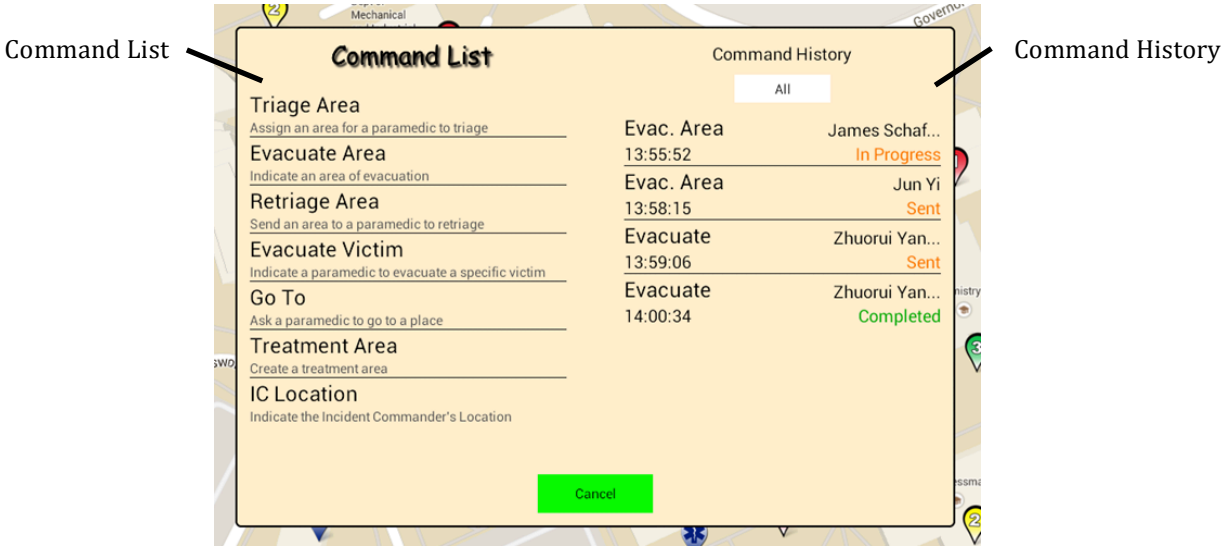


Figure 8: Command Panel

In this panel, a list of available commands that can be sent to the responder is located on the left. Commander can create commands by clicking on the corresponding one in the list. Those commands can be divided into three types: single-targeted command, whose target is only one victim, such as evacuating a victim; area-targeted command, whose target is an area, such as triaging an area; and position-targeted command, whose target is a position, such as going to a place. The list of command history on the right shows the commands already sent out and their current statuses.

In DIORAMA, each command has three stages:

- **Sent:** the command is sent to the server and no response has been received yet
- **In Progress:** responder received the command and clicked "acknowledge" button, indicating the responder's acceptance of it
- **Complete:** the commander is complete (or canceled) and no longer valid

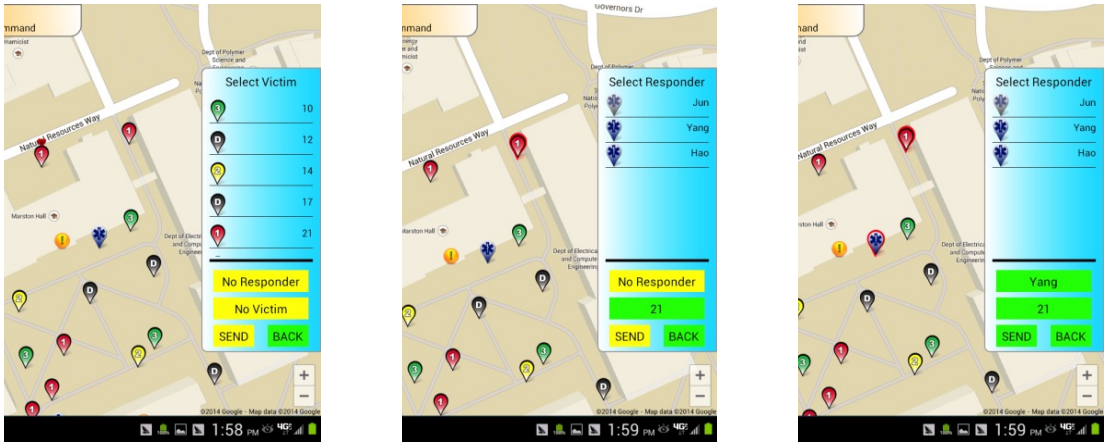
Each command begins with the "sent" status. The process to create a command is shown in Figure 9. It contains the following steps:

1) Select a command in the command list

2) Select a target: for single-targeted commands, choose the victim on the map or in the auxiliary list (Figure 9a); for area-targeted commands, draw an area (Figure 10); for position-targeted commands, tap on a location on the map

3) Select a responder to execute the command (Figure 9b)

4) If the information is correct, click "send" to send the command (Figure 9c)



(a) Select Victim (b) Select Responder (c) Confirm and Send

Figure 9: Process of Creating a Single-targeted Command

Figure 10: Creating an Area-targeted Command

It's worth mentioning that command module allows the commander to create any customized shape for area-targeted commands. In Figure 10, commander app asks the user to draw an area by tapping on the map. Every touching will create a vertex and link it to previous ones forming a new edge. The user can create as many vertices as needed and

customize any complex area. While drawing the shape, the user can always drag the existed vertexes and modify the shapes of them. Compare to the paper triage system, DIORAMA visualizes the areas on the map, making them much more understandable to users.

After the area is created and a responder is selected, this command will be sent to the server and broadcasted to everyone.

Command module also provides the ability to review previous commands and control them. When a previous command in command history list (Figure 8) is clicked, review mode will be turned on. As shown in Figure 11, the detailed information of this selected command is displayed on the right top of the screen. Those details include: the commander who sent this command, the responder who is executing it, command content, current status of execution, and its target.

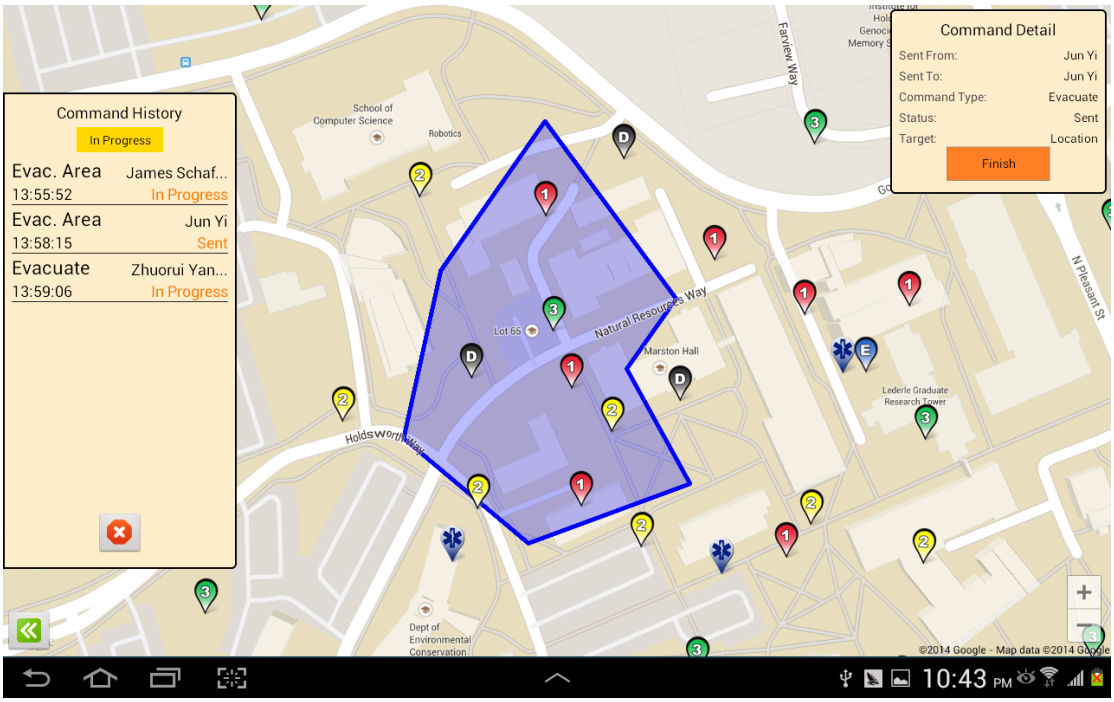


Figure 11: Command Review and Control

The completion of a "victim evacuation" command is determined automatically when DIORAMA system detects the assigned victim is successfully evacuated. This is also applied to the "go to" commands when the responder reaches the designated location. However, for

the commands like “triage an area”, or area-targeted commands, the commander should end them manually. If a commander thinks a command-related area has been successfully searched and rescued, he or she can mark this command “complete” by clicking the “finish” button. This will remove the shaded area that is overlaid above responders’ map and pop up a notification. Then those corresponding responders will be acknowledged and ready to receive the next instruction.

3.1.4 Forensic Analysis Tools

Forensic analysis tools are designed in DIORAMA commander app. They use the data stored in server database to retrieve past events for further analyses. With these tools, users can watch a history playback or generate either a summative (about the whole incident) or specific (about one responder) report. Forensic analysis tools are methods of showing past events, helping users to analyze incidents in a more efficient way.

- **Timeline Playback**

Timeline module is designed to replay the past scenarios and movements within a designated time period. As Figure 12 shows, in timeline mode users can select a specific time by dragging the sliding bar on the bottom. When a time is selected, the app will connect to DIORAMA server and retrieve the incident information of that moment. These POIs will be transferred to icons and replace those currently displayed on the map.

Timeline mode also offers a function that dynamically moves timeline forward and updates those icons according to the real history. When started, the designated time will keep going forward and present the scenarios frame by frame. From users’ aspect, the timeline playback is like watching a “video” of the rescue process so that all movements and actions of responders can be replayed. Dynamic historical playback function helps commander to better understand the changes in the field and analyze the performance of responses to incidents.

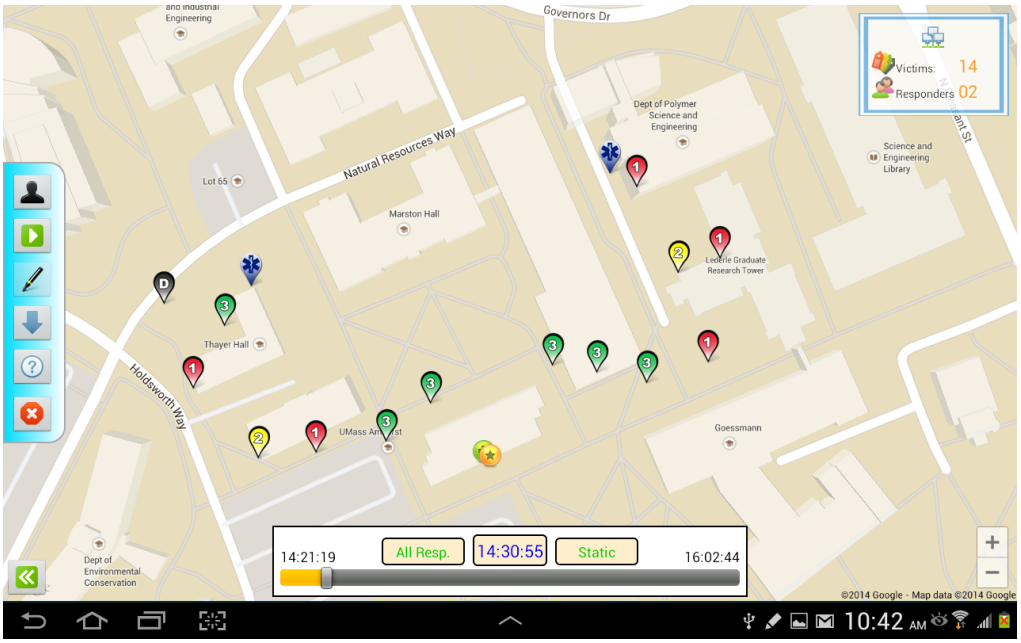


Figure 12: Timeline Mode

In addition, in order to better visualize the moving pattern of responders, commander app is able to draw the paths of them (Figure 13). The app retrieves every position that a responder has been to and links them together using a unique color. Those paths can even be drawn dynamically with the replay going on.

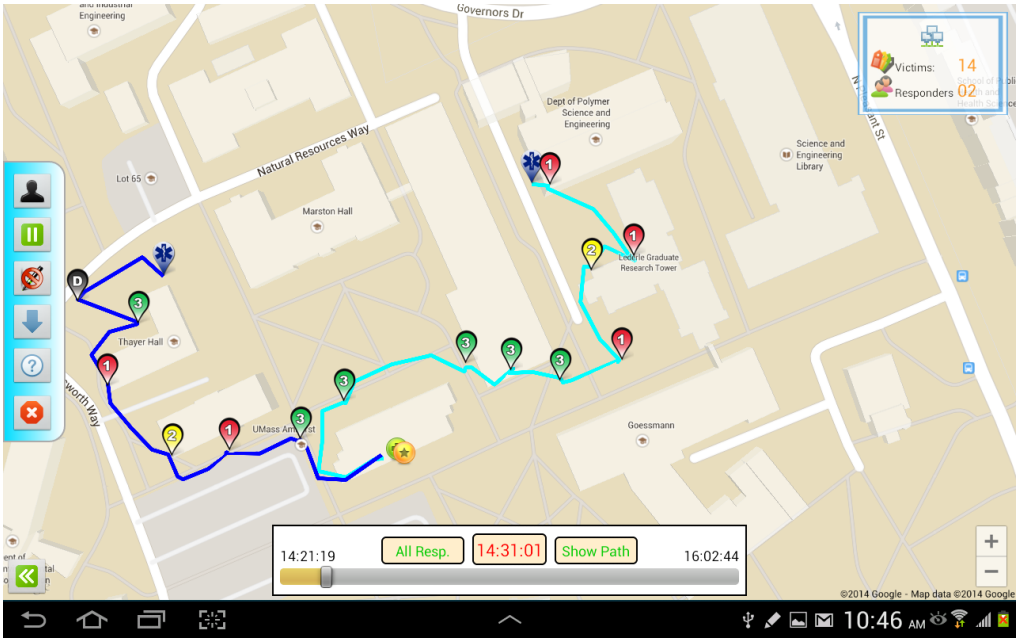


Figure 13: Responder Path in Timeline Mode

- **Statistical Report**

Compared to timeline replay, statistical report generates a more detailed report about the whole incident. It is another part of forensic analysis tools, able to generate summative reports of all POIs as well as specific reports of any individual.

Figure 14 is an example of summative statistical report. The summative report lists all events happened during the incident, showing the commander an overall picture of it. In the report, each record includes the victim ID, the responder who executed the action, the priority of the victim at that moment and the time of the action. Since all records are ordered by time, it is obvious if a victim with a lower priority is evacuated before the higher ones. The number of incidents and the amount of victims are also presented in the report. The triage time is counted from the issuing of the first triage command until the end of triage process (which needs to be designated manually by commanders). Evacuation time is calculated similarly.

In addition to generating a summative report, commanders might also be interested in the performance of a specific responder. To generate such a report for individual, users need to select a responder in the POI list.

Figure 14: Summative Statistical Report

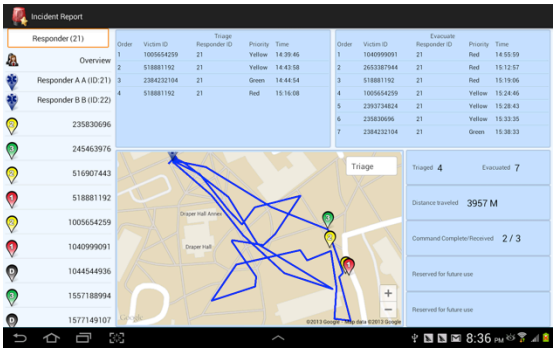
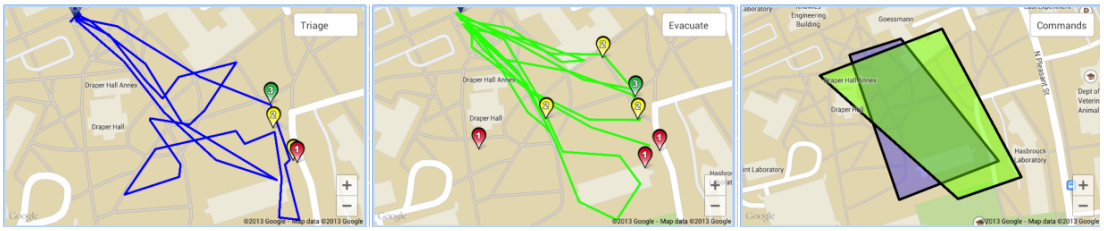


Figure 15: Specific Statistical Report

Figure 15 is an example of the specific report generation. In addition to the triage and evacuation records of this specific responder, spatial details are offered to commanders. In

the figure, travel path of the current responder during the triage stage is drawn on the map. Victims that are triaged by him or her are displayed as well.

More types of information about the specific responder can be displayed on the map as well. Figure 16a is the triage path of responders mentioned above; Figure 16b shows the evacuation path of responders. In this option, only the path during evacuation stage is displayed. Figure 16c shows all assigned commands of responders, including the area commands and individual evacuation commands. Combination of multiple types helps the incident commander easily analysis the performance of responders. For example, a manager can look at the combination of triage path assigned command to see if the responder has fully traversed the whole command area.



(a) Triage Path (b) Evacuate Path (c) Assigned Command

Figure 16: Different Types of Report

3.2 Responder App

In an emergency, responders' job is working on the field to triage and evacuate victims. To help them focus more on their work rather than on screen, the most important requirement of responder app is efficiency. The basic functions of responder app are showing a map and tracking the device's location. Besides these, the app also allows paramedics to triage victims and track them afterwards. When a command is received, responder app will notify the responder and lead him or her to the correct location.

3.2.1 Application Flowchart

The flowchart of responder app is shown in Figure 17. When a responder log into an incident in DIORAMA system①, a map of the incident area will be displayed. All POIs are

displayed on the map as different icons②. Using the map view a responder can see all the victims with their priorities and status (section 3.2.2). During the triage and evacuation process, a background service③ is running to track users' locations through GPS and send them to server in real time. Therefore commanders always know the location of every responder even when they turn off the screen and put the device in their pockets. Triage and evacuation process can be triggered when a user scans a NFC tag or points on a location on the map. Then triage dialog④ will popped up, allowing users to choose the priority of a patient and send to server (section 3.2.3). When a command is received, the app displays it on the map and tries to notify the user⑤ (section 3.2.4).

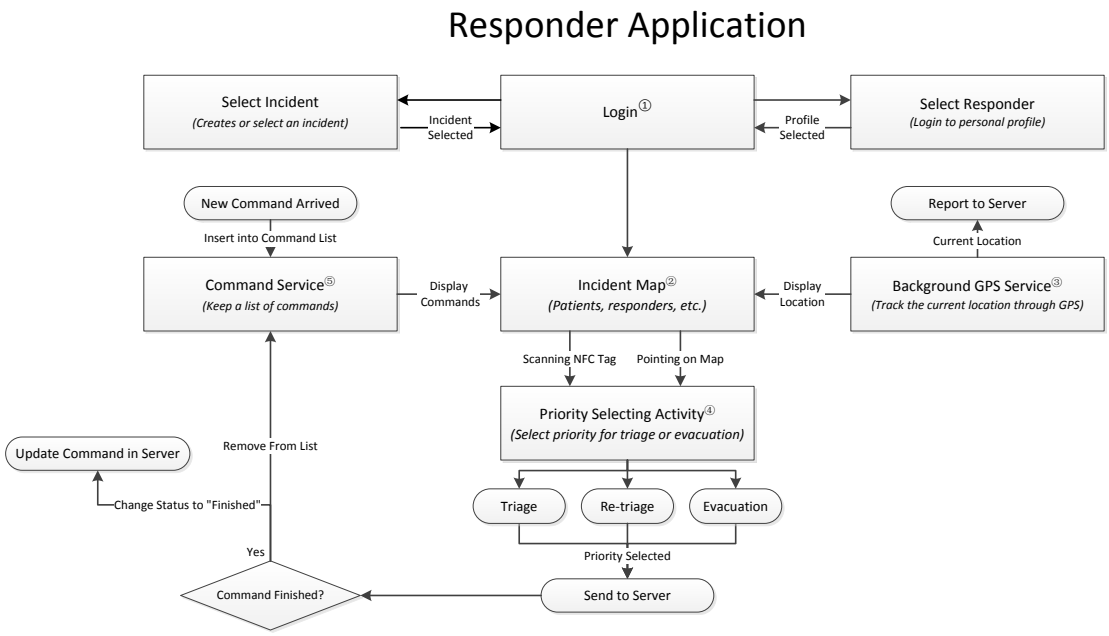


Figure 17: Flowchart of Responder App

3.2.2 Map Interface

Figure 18 is the map view of the responder app. Map view is designed to meet the basic requirements of responders: showing the map and POIs. Similar to the map view of commander app described in section 3.1 , the 3D Google Map takes most of the screen space. Users are able to focus on the map and get all information they need. The status bar is

located at the top of the screen whose job is displaying the current network status of the system. The statuses of the application include: connecting to server, system running, reconnecting, using back-up connection etc. All the function buttons are integrated in a pop-up menu, which can be opened by clicking the menu button in the status bar (Figure 19).

In the menu, responders have the following options:

- **Summary:** an overview of the quantities of all victims in different priorities
- **Compass:** rotate the map according to the direction that the user is facing
- **Indoor Overlay:** overlay current outdoor map with indoor blueprint
- **Connection:** check connection status; reconnect to server
- **Option:** more options about map display and connection

Figure 18: Map Interface of Responder App

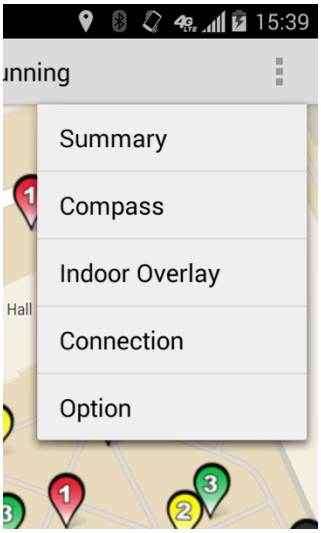


Figure 19: Pop-up Menu

3.2.3 Triage and Evacuation Using NFC Technology

Near field communication (NFC) is a set of standards for smartphones and similar devices to establish radio communication with each other by touching them together or bringing them into proximity, usually no more than a few inches[15]. In order to increase the speed of triage and evacuation, DIORAMA uses this technology to identify and update victims' information. In the first stage of emergency responses, patients are triaged

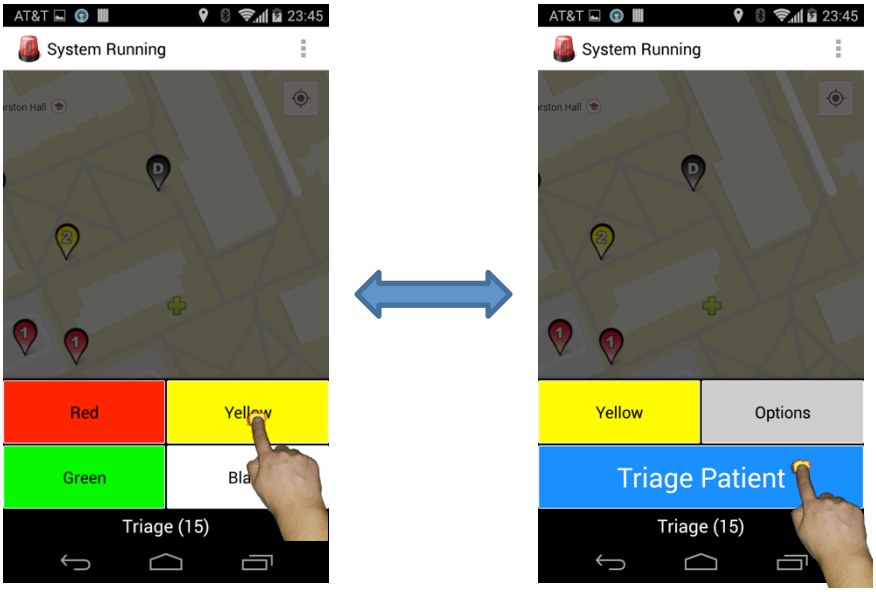
according to their injury levels (red, yellow, green, black) and tagged with D-tags (Figure 20). As described, a D-tag is a pack of objects including a paper triage tag, a passive RFID tag and a NFC tag. When the back of an NFC supported device touches the tag, the ID of the passive RFID tag in this bag can be identified. This unique ID can be used to represent a triaged victim, so further tracking and updating processes could also be accomplished by scanning the same NFC tag.

Figure 20: D-tag

When an NFC tag is scanned, the responder app will automatically determine the possible action the user wants: if this NFC tag ID is not registered in the system yet, do the triage; otherwise do the evacuation or re-triage. In the case of triage, a sliding window will appear from bottom, overlapping a small part of the screen (Figure 21a). Different buttons with corresponding text are shown in the window; the action name "Triage" and victim ID number will be displayed as well.

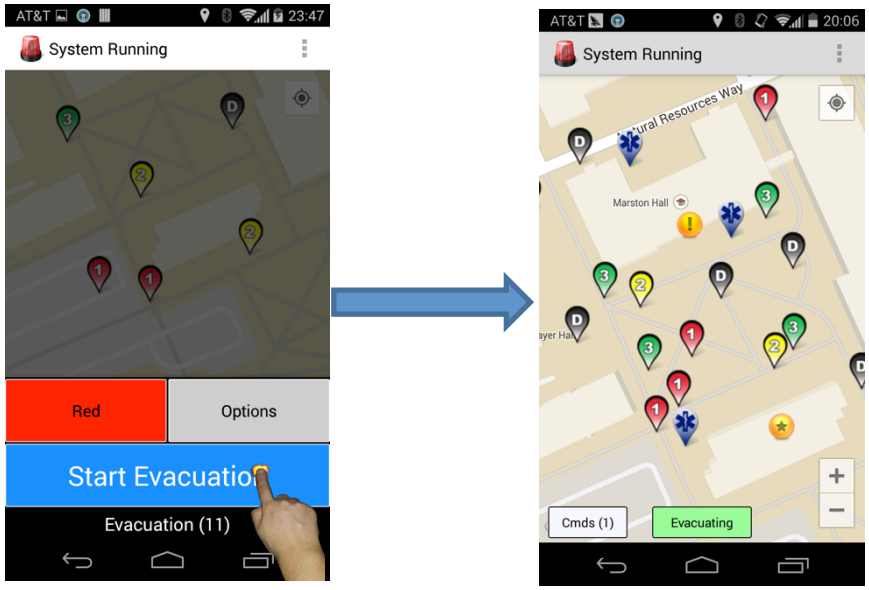
After the user selects a priority, "Triage Patient" button will be visible and enabled. (Figure 21b) More options can be applied through the "Option" button, such as marking the victim "trapped" etc. The user can always change the priority by tapping on it. Finally clicking on "Triage Victim" button will confirm this triage and send the patient's information to server. After that, a patient icon will appear on every user's map according to the selected priority.

A shortcut of triaging patients is to long click one of the priority buttons in Figure 21a. It sends the triage information immediately without any other steps.



(a) (b)

Figure 21: Triage a Victim



(a) (b)

Figure 22: Evacuating a Victim

Performing an evacuation is quite similar. When a victim tag is scanned, system will automatically check the status of this tag: if this tag is already triaged in the system, evacuation option will be popped up directly (Figure 22a). Users only have to click the

"Start Evacuation" button to start the evacuation process. When this process starts, an "Evacuating" button appears on the bottom (Figure 22b). It is a reminder for responders of their current status and will display more details about victims they are evacuating if clicked. Finally when this responder brings the injured victim to any treatment area, he or she can scan the tag and end this process.

More options are offered if user clicks the "Option" button. Figure 23 shows the three advanced options supported currently.

- **Point**: the option that use pointed location instead of GPS. This is useful when responders wish to report any victims or hazards beyond their reach. They can manually select the location on the map.
- **Trapped**: this is a marker for trapped victims. Trapped victims are those who cannot be evacuated by responders manually; they need specific equipment.
- **Retriage**: when the statuses of some victims change, responders can check this box to re-triage them and assign them with different priorities.

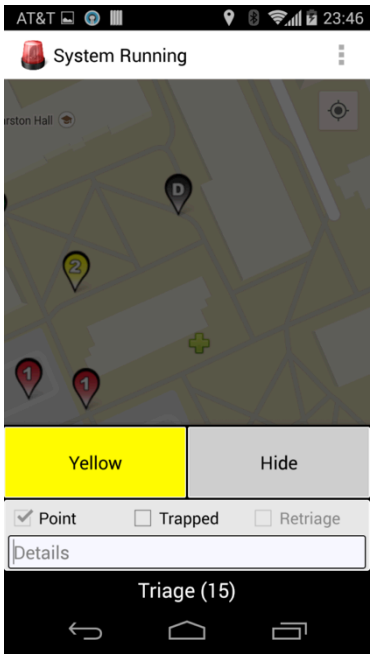
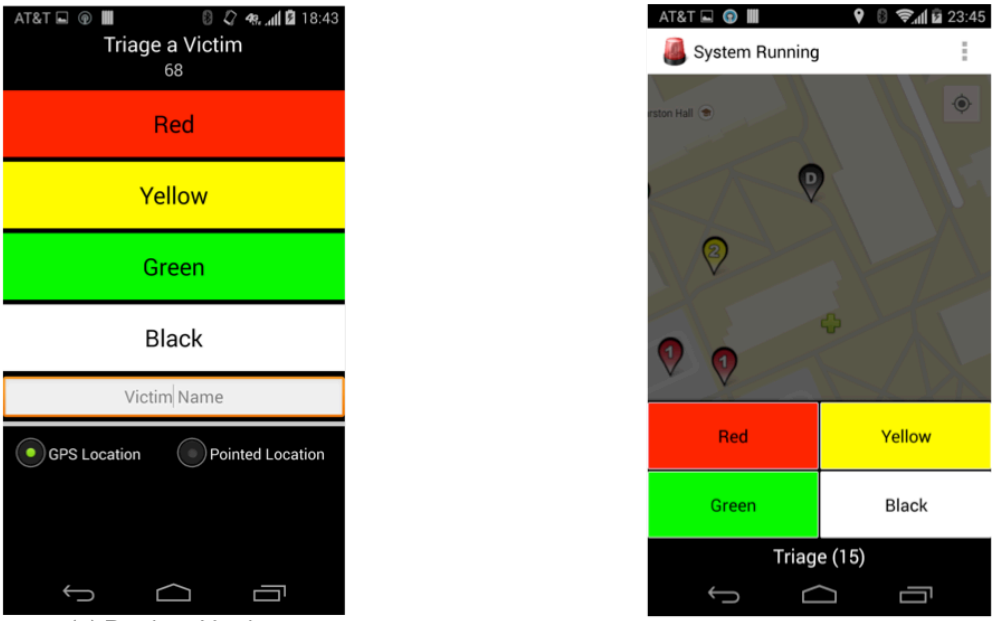


Figure 23: More Options of Triage and Evacuation

One of the advantages of this interface design is the efficiency. During an emergency response users need to quickly navigate to the functions they need and execute them within the least steps. In the previous versions of DIORAMA, when a tag is scanned a new full-screen activity will pop up asking users to choose a priority (Figure 24a). There are two disadvantages in this design. First of all, this full-screen interface overlaps the map that users are currently working on; it completely interrupts their current work. Secondly, starting up a new activity is a slow process. In DIORAMA previous versions, it takes about 1 to 3 seconds to start the triage or evacuation activity, and about 1 second to close it after users' actions. It is a quite reasonable delay for general apps, but in emergency situations where responders might need to triage hundreds of victims, this delay will become annoying and fatal.

To make these processes friendlier and faster, the sliding panel is used (Figure 24b) in current versions. As described in 3.2.3 , instead of popping up a new full-screen activity, the current responder app shows a half-screen sliding panel with priority buttons over the map. The panel takes about a quarter of the screen and the rest of map is shadowed. In this way, the panel does not interrupt users' current focuses while still able to remind them to take new actions.

In the other hand, showing up a sliding panel in an existing activity is much faster than starting a new one – it takes less than a second to display it. This significantly increases the efficiency of triage and evacuation processes as well.

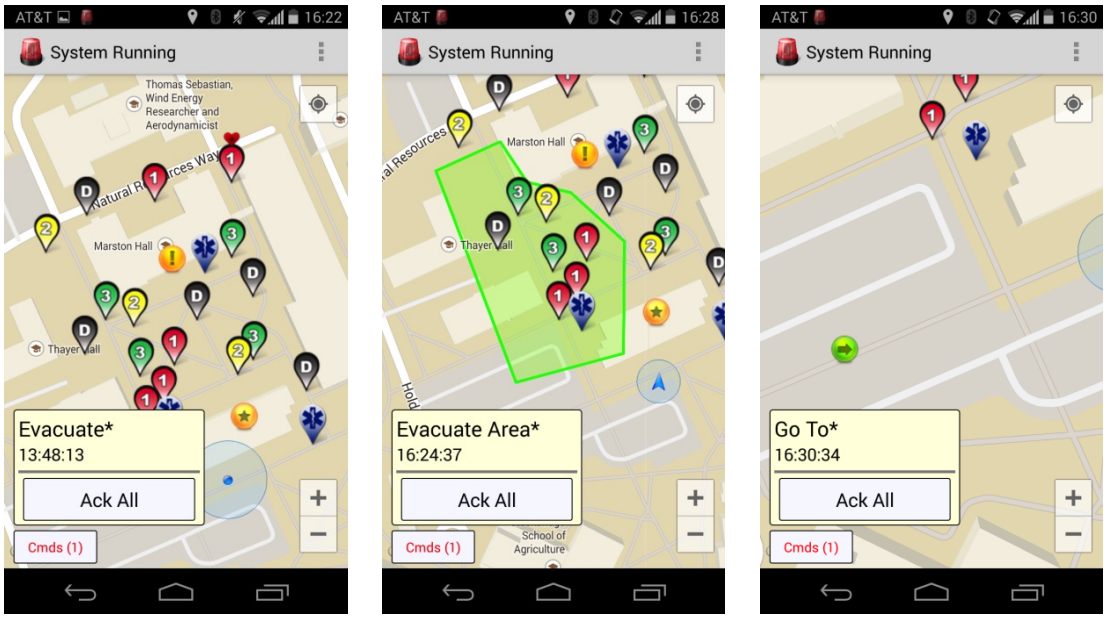


(a) Previous Version (b) Current Version

Figure 24: Full-screen Activity and Sliding Panel

3.2.4 Command Module in Responder's App

When commands are sent to responders, it's important to visualize them properly and comprehensively. In the responder app, received commands are stored in a list and are called out when the responder clicks on the "Cmds" button (Figure 25) on the bottom of the screen. The list shows every incomplete command of the current responder. A command should be acknowledged after the responder sees it (the stages of a command are stated in section 3.1.3). Unacknowledged commands are marked with a star to remind the responder. "Ack All" button is used to acknowledge all commands at once.



(a) Evacuate Victim (b) Evacuate an Area (c) Go To a Position

Figure 25: Commands Display in Responder App

This figure also shows the visualization details of different types of commands. As I mentioned in 3.1.3 , basically three types are used in DIORAMA system:

- **Single-targeted Command**

This command is visualized as Figure 25a. A target victim is marked with a little heart on its icon. When the command in the list is selected, the responder app offers users an option to locate the target victim. When this option is clicked, the camera of the map will be moved towards the victim, helping responders find their target faster.

- **Area-targeted Command**

Areas are directly displayed on the map (Figure 25b). The color of areas depends on the type of commands (evacuate, triage or re-triage) and more details will be shown when the command is selected. Combined with GPS data, a responder is able to locate himself or herself on the map and head to the designated area correctly.

- **Position-targeted Command**

For a "go to" command, the target position is placed as an icon on the map (Figure 25c). This could tell responders an accurate position that they are asked to go. Detailed

information such as the reason of going can also be shown if users select this command in the list.

It's highly possible that when a command reaches a responder's device, the responder might be busy dealing with the triage or evacuation. His or her cellphone may be left in the pocket, or the responder just doesn't have time to check out the app and response the command. So notification is used to help responders view and respond the commands quickly.

Notification[20] is an Android feature that displays a message icon on the top of the screen. Users can slide down the notification panel to check out all notifications without closing the current app. When a new command is received by the responder app, the DIORAMA icon is presented in the notification bar (Figure 26). At the same time, ringtone, vibration, and flashing lights will also be triggered to remind responders the arrival of a new command.

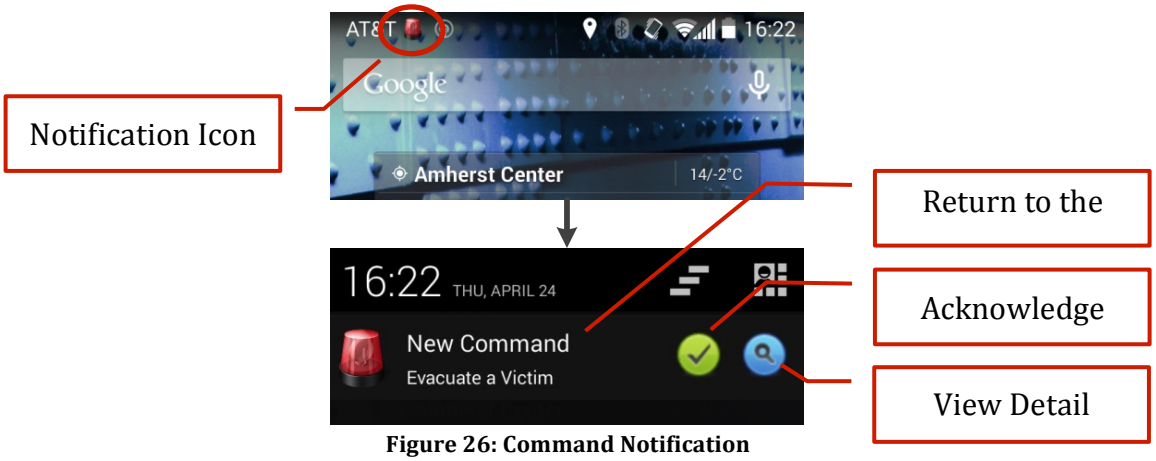


Figure 26: Command Notification

When users slide down the notification panel, the newly received command will be displayed. The content of the command and a brief description are shown to the user. The two buttons in the notification are shortcuts for responding the command. As indicated in the Figure 26, when the notification itself is clicked, the app will be brought to the front. When the "Acknowledge" button is clicked, then the command will be set to "acknowledged"

at once and no further actions need to be taken. Moreover, when the detail button is clicked, the app will be brought to the front as well but the detail of the new command will be displayed.

The new command icon will not disappear in the notification bar until this command is acknowledged. It always reminds responders that actions are still needed for this command. When more than one command has not been acknowledged, the notification only displays the number of commands; the acknowledge button will work as “acknowledge all”.

CHAPTER 4

OPTIMIZED CLIENT-SERVER TECHNOLOGY

Communication is a fundamental part in DIORAMA system. All information of victims, responder locations, hazards, commands etc. are exchanged within this system network. DIORAMA system includes several kinds of client devices and a server, which is the base of all communications between those apps.

The network architecture of DIORAMA is shown in Figure 27. In the figure, server interface is the code running on the server that analyzes the input data, queries DIORAMA database for certain information and gives response. It acts as an agent communicating between other external devices and the server database. During an incident, responder app keeps sending GPS data of the device to server constantly; each GPS record is stored separately in DIORAMA database. Responder app also sends updates of victim information when the he or she performs any action; and commander app sends commands to server when the incident commander wants to assign tasks to responders. All these data are stored in the server database as well. Meanwhile, both responder app and commander app keep track of the latest data of all POIs and commands in the field and refresh their interfaces.

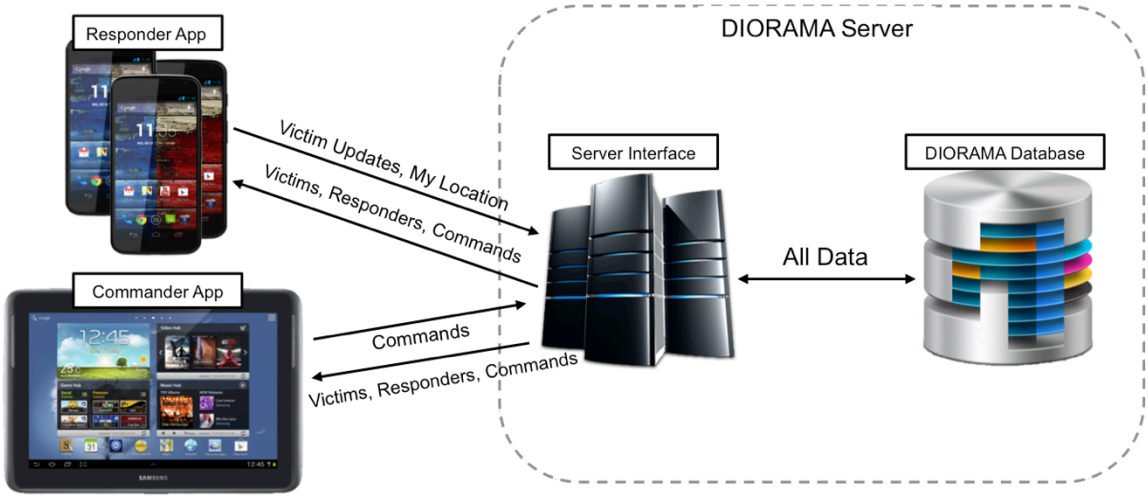


Figure 27: Network Architecture of DIORAMA

In this section we will introduce the communication techniques used between DIORAMA server and other apps – pull and push. However, the details of server implementation and DIORAMA database are not the emphasis of this thesis.

4.1 Pull Technology

Pull coding or client pull is a style of network communication where the initial request for data originates from the client, and then is responded to by the server[22]. In this case, the server responds to the clients only when it receives any request from them.

HTTP pull technology was used in previous versions of DIORAMA. As shown in Figure 28, the data sent from client will be stored in server database and no further actions will be taken (The yellow boxes in the figure are database-related operations; they usually require more time). When the responder or commander app logs into the system, it periodically sends HTTP request to DIORAMA server asking for new updates. These updates include the latest information of victims, responders, commands etc. Server receives the request, analyzes it, queries the database, and finally responds to the client with a bunch of new data. After getting the response, the client app will refresh the information displayed in the app (such as display new victims, change the location of responders or pop up a new command).

This process will repeat after a certain amount of time *x* and keep requesting for new data until the app ends. When this x is small enough (generally three seconds in DIORAMA apps), the information updates will appear to be real-time for users.

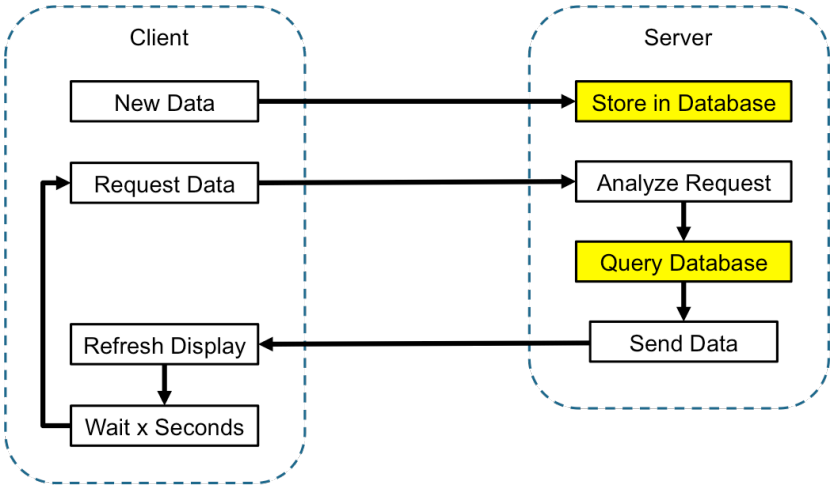


Figure 28: Flowchart of Pull Technology in DIORAMA

In the history of DIORAMA development, two different implementations of pull technologies were used. In the early versions, clients request for ALL information in the incident every time. DIORAMA server responds to each request with the data of all victims, responders, landmarks etc. However, this is an inefficient way because there are huge amount of redundancies within all this data: most of data (like victim locations) keeps unchanged but is still sent to clients. The later versions resolved this problem by requesting only the changes since the last update. In this mechanism, the requests sent from clients include a timestamp in addition. DIORAMA server compares the timestamps of those requests with the records in database and extracts the records or updates after this given time. This method considerably reduced the amount of data transferred between clients and the server, and increased the speed of both responder app and client app.

Pull technology has several advantages.

Firstly, since the server's job are just listening and responding to requests, the complexity of server code is relatively low. In another word, pull technology simplifies all different network communication methods (sending data, getting updates etc.) across different platforms (Android, C# etc.) into simple HTTP requests.

Secondly, there is no connection maintained between clients and the server. Clients will establish connections to the server only when they send requests; those connections will be terminated after they receive the responses. At other times, no connection is necessary, releasing the resources in DIORAMA server and client.

Thirdly, pull has a high fault tolerance because the requests are independent from each other. If the local network goes down for a while, some of the requests may get no response; but as soon as the network recovers, further requests will not be reflected. Since each request contains a timestamp that indicates the last time it receives an update, it's easy for clients to catch up with the newest information.

However, pull technology also has some flaws.

In our early tests and trials, even the time interval of updates was as small as three seconds, the delay was still noticeable. In users' aspects, the responders' movements were not continuous, or more like "frame by frame". However, when we reduced our update interval to less than three seconds, it went even worse – all devices stopped updating after several minutes, the whole system broke down. This issue was caused by the network and server response delay. Each time when server received a request, it made a query to the database for specific data and gave the client a response. This process was quite fast, usually done in several milliseconds if there were only a small amount of requests and a few lines of data in database. But when the number of responder grew larger (such as ten), the server would receive 3 to 4 requests per second, forcing it to query the database more than 10 times per second. What's worse, when more and more data were collected and stored in the database, a single query might take hundreds of milliseconds. In this case, database would be overloaded by the influx of requests, blocking increasingly further requests and finally bringing down the whole system.

This is an unavoidable weakness of pull technology. Therefore, we introduce push technology in the following section, which effectively overcomes this issue.

4.2 Push Technology

Push, or server push, describes a style of Internet-based communication where the request for a given transaction is initiated by the publisher or central server[23]. As opposed to the pull technology, in push, server actively broadcasts data to subscribed clients when necessary - even there is no request. After the broadcast, all subscribers will receive the updates in real time.

In the newer versions of DIORAMA, push is used as primary communication technology. The flowchart of the system is shown in Figure 29. When a client (either a responder app or commander app) logs into DIORAMA system, it connects to the server and subscribes to a certain group with its incident ID. This group is more like a subscriber list stored in server, and multiple lists may exist simultaneously identified by incident ID. After all these steps are finished, a push connection is considered established. Then the client submits a pull request to get the initial data for the incident, which is almost the same procedure described in the previous section except that this is only a one-time operation executed at the beginning of login.

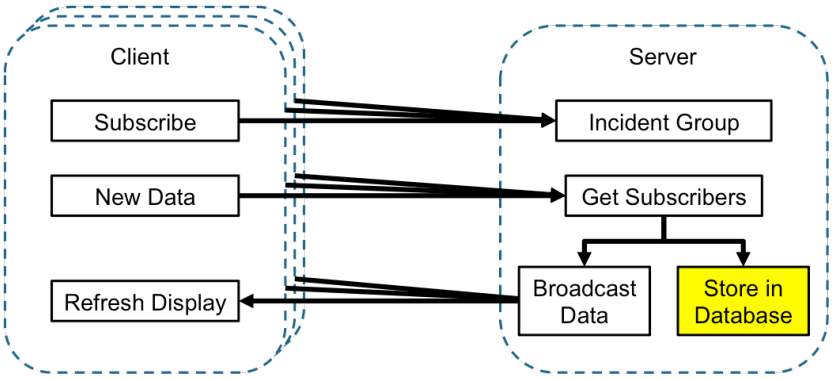


Figure 29: Flowchart of Push Technology in DIORAMA

During the incident, when server receives any new update (victim triage, responder movement etc.), it looks into the subscriber list of that certain incident ID and "broadcasts"

the new update to clients in that list. A more detailed description of this "broadcast" is, the server remotely invokes one of the methods in those clients according to the update type and passes the new data as parameters. For example, to push a victim update, the server will try to invoke the "*onVictimTriage (*Patient *patient)*" method on all clients (which should exist in their codes), and transfer the victim information in the parameter *patient*. Receiving it, those clients then launch corresponding methods to refresh their apps and user interfaces to display the update.

At the same time, DIORAMA server stores this information into the database.

According to the process described above, push technology is "a publish/subscribe model. A client "subscribes" to various information "channels" provided by a server; whenever new content is available on one of those channels, the server pushes that information out to the client[23]. To implement push, DIORAMA clients use a modified version of SignalA[24] – a push library designed for Android. After all clients are subscribed, the server doesn't need to wait for requests to send new updates one by one; instead, it is able to send new updates to the clients right away, and once for all.

It's worth mentioning that since all these procedures are implemented by server code which runs much faster than database-related operations, server can publish the new updates almost immediately after they arrive. Meanwhile, because accessing the database is done in parallel and in the background, it makes no influence on this pushing process or server responding speed. This is quite important because in users' aspects, POI updates are now more influent and responsive, offering them a better experience of real-time DIORAMA system.

Another crucial contribution of push technology is that it eases the burden of DIORAMA server. In push system, clients no longer need to request for redundant updates periodically. This leads to the fact that server no longer needs to query the database several times per

second as well. Most of the frequent database-related operations left for the server to do are just storing incoming data into the database (which is exactly the server's job). This is more acceptable especially when more data is collected and stored in it: in this case, querying will be expensive but inserting will be not. Besides, when more clients are logged in to the system, instead of handling an exponential growth of requests from them, the server only has to send some additional copies of updates. Hence, the benefit that push technology has brought to the system is significant.

Nevertheless, push technology also has its shortage. Since the server is pushing only the changed POIs instead of all of them, all messages sent from server (except for the initial one) are heavily dependent on the previous ones. In another word, missing a broadcast message might leads to a permanent loss of that point of interest. For example, while one of the clients is experiencing a temporary network failure, the server broadcasts a message "victim X is triaged as green". After several seconds, this client recovers and handles all further updates normally. However, if then no responder ever updates this victim X again, the client will miss this message permanently. In the user's aspect, the app is running smoothly all the time, without noticing that he or she would never know the existence of victim X. Therefore further measures need to be taken to monitor the network status and check the continuity of updates.

Although flaws exist in push technology, the improvement it brought to the system is enormous. Section 4.3 tests and analyzes the performance of pull and push in DIORAMA system; the strategy used to solve this problem is illustrated in section 4.4 .

4.3 Performance Analysis

In order to test and analyze the performance of push and pull implementations on DIORAMA system, a set of trials were conducted near the Knowles Engineering Building of University of Massachusetts. In the trials, we randomly distributed 30 cones in the field

indicating victims (Figure 30). Examiners acted as emergency responders, held the DIORAMA responder apps and performed triage to those "cone victims". During the tests, 11 responder apps logged into the system and kept their usual communications (sending locations, retrieving updates etc.) with DIORAMA server. This simulated a realistic situation with sufficient victims and responders to test the performance of push and pull.

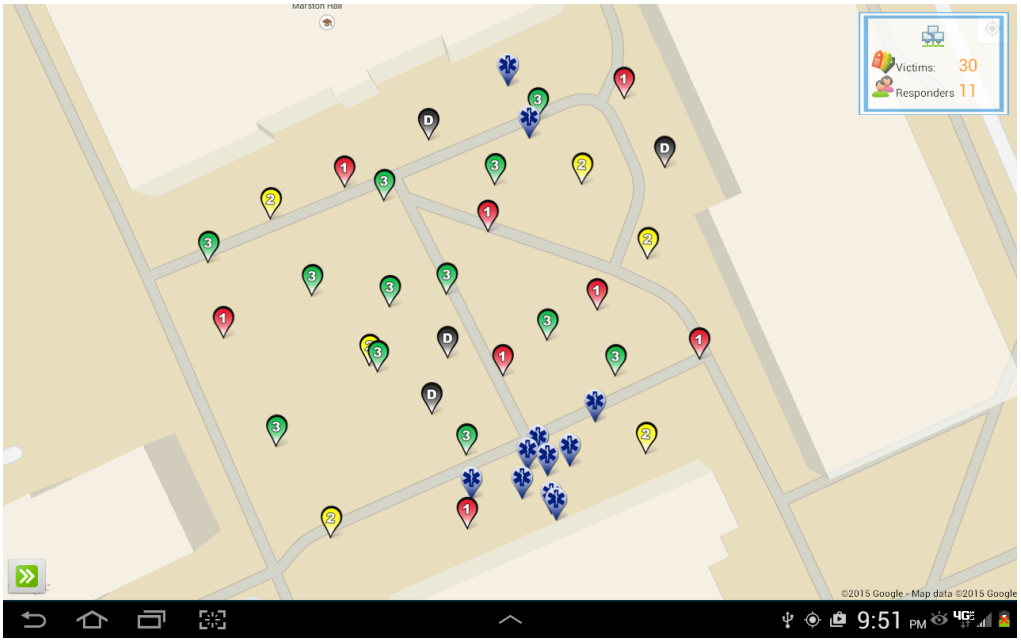


Figure 30: Layout of the Performance Tests

To compare the performance difference between push and pull, we built up two separated tests based on this victim distribution: in the first one all responder apps connected using pull technique only; while in the second test they used push instead. During each test, we kept increasing the number of patients and then recorded the following data: the CPU load, the read operation overhead of DIORAMA server, and the time from the triage of a new victim to the arrivals of this message at all responders' devices.

4.3.1 Response Time Analysis

The purpose of this test is to analyze the response time of push and pull during the triage stage. In the test, the timer started when a responder clicked on "Triage Victim" button to create a triage message. The message was then sent to DIORAMA server and soon

received by all clients through push or pull communication. During this process, we also recorded the time that each client received this piece of information, and found out the latest arrival among all receivers. Finally, we calculated the difference between these two timestamps: the time a message is sent to server and the total time it costs to spread over all other clients.

The response time of triaging nine victims were recorded and 3 cellphones were used to collect sufficient data. Tests of pulling technology with different request frequencies (one query every 2 seconds, every 1 second and every half second) were also conducted in order to observe the affect of request frequency on response time. The test result is shown in Figure 31.

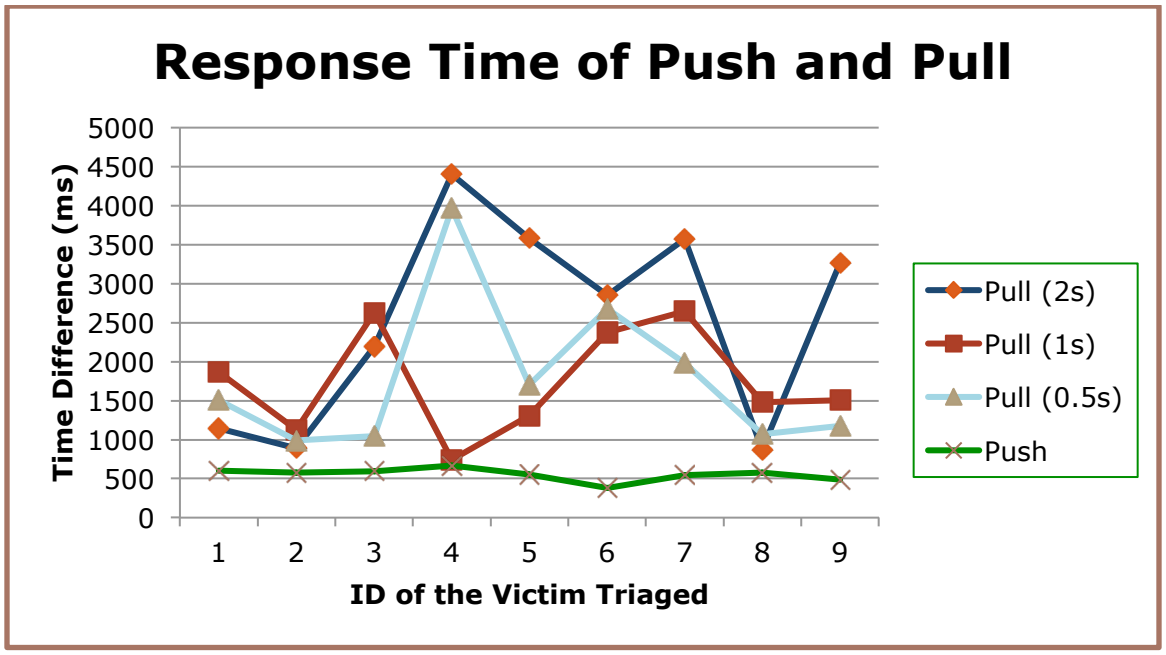


Figure 31: Response Time of Push and Pull

In the figure, the X-axle represents the ID of the victim that is triaged; the Y-axle is the time interval between the triage information is sent to server and received by other devices. As a result, the response time of pulling varies among the nine victims. This is because in pull technology, clients request the server for information by a certain interval, which will not be influenced by other clients. Shortly after an update is posted to server, those clients

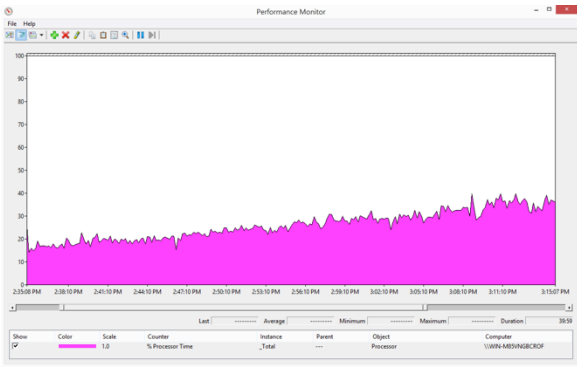
that coincidently send a request to server will receive this new piece of information immediately, while the other clients need to wait until their the next requests to realize this new update. Theoretically, in the worst case, the longest response time will be equal to the request interval; but due to the server and propagation delay, the response delay grew larger during the tests of pulling, with the average delay of 2300 milliseconds.

However, the response time of push is more stable. For DIORAMA server pushes all updates right after they arrive, all clients can receive them without waiting for any intervals. As the green line shows in Figure 31, the response time of push is always around 600ms. This is significantly more responsive than pulling, making all clients in DIORAMA system synchronized in real time.

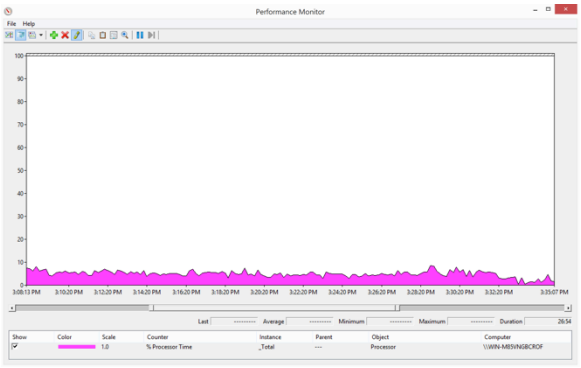
4.3.2 CPU Load and Read Operation Overhead Analysis

Another serials of tests have been conducted to test the load of DIORAMA server using either of the communication methods.

In CPU load test, we monitored the processor time of server during the incident. Processor time is the percentage of time the processor is busy by measuring the percentage of time the thread of the Idle process is running and then subtracting that from 100 percent. It is a way to calculate how many percentage of the CPU calculation capacity does DIORAMA system use during an incident. Similarly, in read operation overhead tests we checked the total number of bytes transferred per second due to read. Since during the test, the only program running on server was DIORAMA, this could represent the expense of database operations. Through these tests, we can monitor how much of the server resources DIORAMA system is using with push or pull technology.

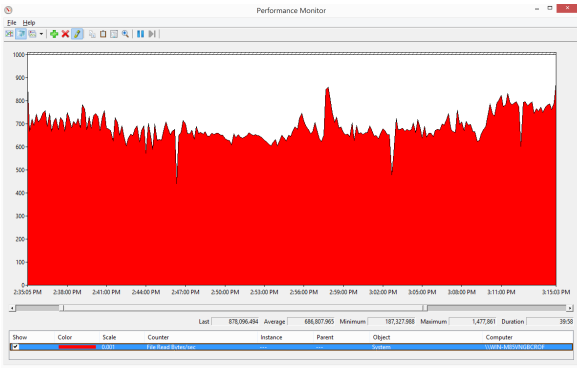


(a) Pull Only

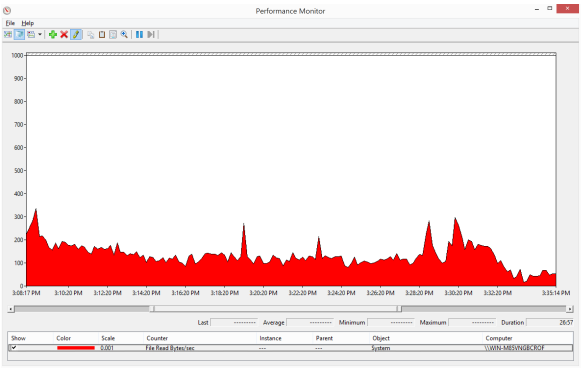


(b) Push Only

Figure 32: CPU Load of Push and Pull



(a) Pull Only



(b) Push Only

Figure 33: Read Operation Overhead of Push and Pull

Figure 32 and Figure 33 show the results of the server load tests. These figures are screenshots of Microsoft Performance Monitor that was running background as we performing the trials. The X-axle of the graphs stands for time, from the logging in of the first responder until the end of the tests. The Y-axle of Figure 32 is the percentage of time the processor is busy; the Y-axle of Figure 33 is the number of Kbytes transferred per second due to read requests.

The performance of CPU load of push technology was stable (Figure 32b), with the percentage of processor time less than 10% during the whole rescue process. However, while using pull technology, the percentage of processor time grew from 15% to 30% as the number of victims increased (Figure 32a). In another word, using pull technology,

DIORAMA system drained out 30% of the CPU capability only with 30 victims and 11 responders. It would probably use up all CPU capability when the incident grows larger.

Similar conclusion can be obtained from the results of read test (Figure 33). The figures show that pulling technology executed more read operations than push technology, wasting more system resources on redundant database queries.

To summarize, compared to pull technology, push is cheaper in server resource cost, more reliable with the growth of victim amount, and more scalable to the larger incidents.

4.4 The Cooperation of Push and Pull

In DIORAMA, push and pull technologies are used together to guarantee a stable and responsive network connection.

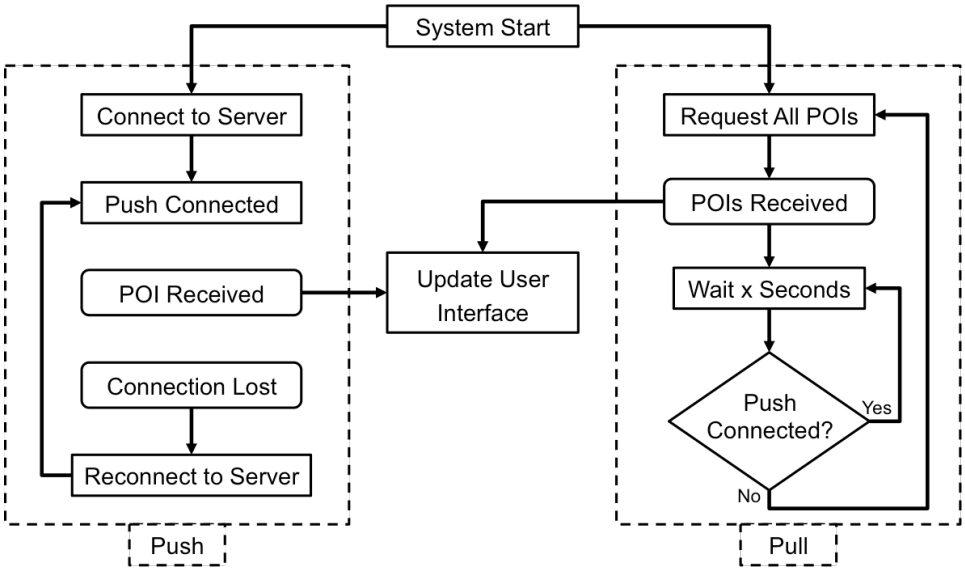


Figure 34: The Cooperation of Push and Pull

As Figure 34 shows, generally when push service is working properly, pull module will stand by and keep checking the status of push connection. Push service updates the app's user interface as soon as it gets an update. In this case, a serial number is attached on every push message before they are published, which is used for clients to compare this number with the previous messages in order to ensure the continuity of this message. If a client

detects a network failure or a missing message (e.g. a missing serial number), it triggers the error handling and immediately tries to reconnect to the server. Meanwhile, when the pull module detects the failure of push connection, it starts to constantly request the latest information from server using HTTP pull. It fetches the information including a full set of POIs of that specific time, and uses it to refresh the app's user interface. After each update, the pull service checks the status of push connection and determines whether to continue pulling or not.

In a word, when push service goes down, pull service will take its place automatically until it recovers; and for each switching, user will always be notified. The push and pull technology works together in the background, creating a responsive and stable network environment for both DIORAMA apps.

However, even with push, the scale of an incident is still limited. Whenever a new responder logs into the system, DIORAMA server has to send one more copy of data through push when each new update arrives. In the case when there are too many clients connecting to pushing service, the data size of each push could exceed the maximal bandwidth of the network, causing network congestion. This issue can be solved by increasing the bandwidth of the whole network, or only sending the related information according to the location of each responder.

CHAPTER 5

INCIDENT MANAGER

5.1 Incident Selection

DIORAMA allows different rescue teams working on different incidents simultaneously. Using the incident manager module users can easily select among several ongoing incidents and find the correct one to enter.

In the previous versions of DIORAMA when users want to enter an incident, they see a list of all incidents with their names (Figure 35). Users scroll down the list and select the incident their want to enter. To create an incident in the previous versions, they need to fill in the name and details of the incident as well as create an area for it. According to the feedbacks of DIORAMA previous users, there are two main issues in these versions: 1) it's easy to get confused with the incidents if only the names of them are displayed; 2) when there are too many incidents in the database, users usually need to scroll many pages to find their incidents.

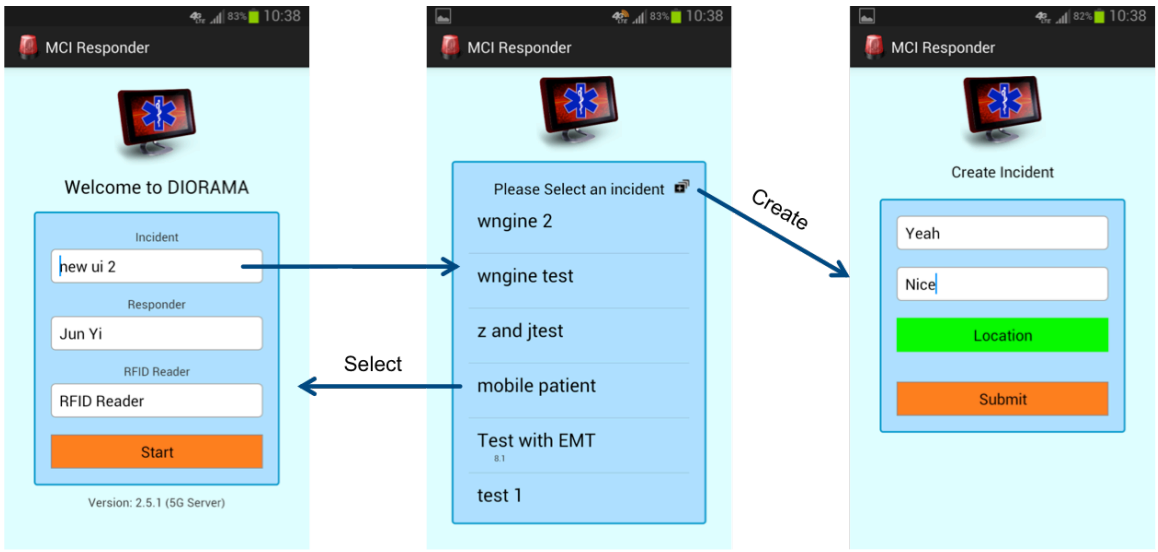


Figure 35: Incident Management of Previous Versions

In order to resolve the issues and bring users an easier way to find and create incidents, incident manager is developed. It visually displays all current active incidents in the map along with summative details of them (Figure 36).

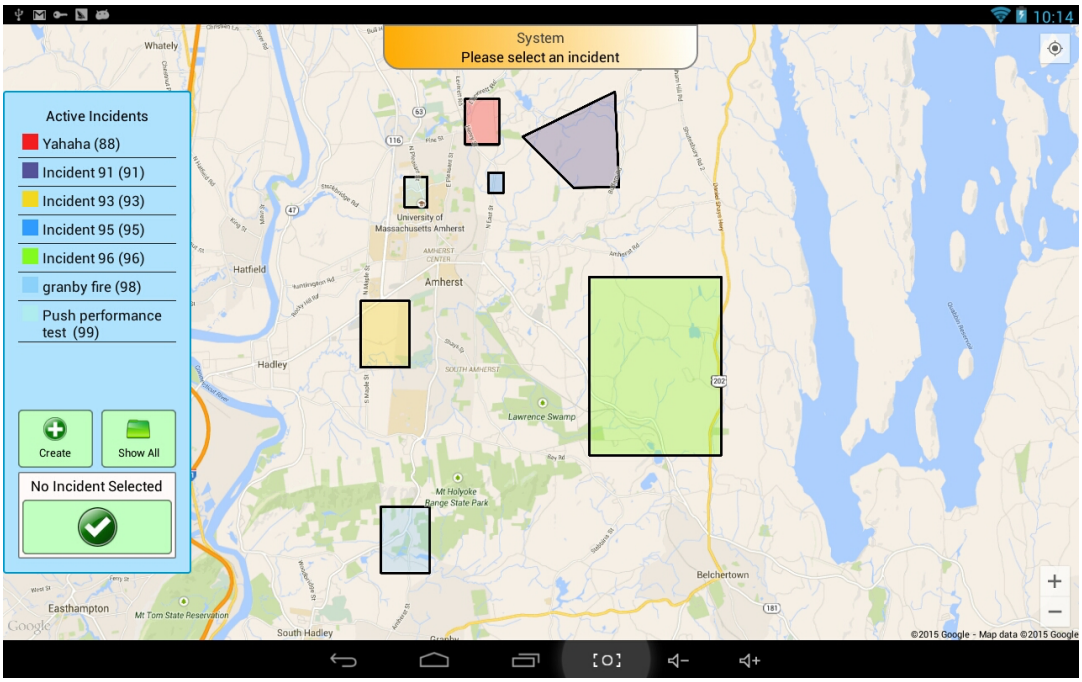


Figure 36: Incident Overview

When users log in to the system with a correct username, they will be brought to the incident overview. In the overview, all currently active incidents are displayed in a list, with their locations marked on the map. Each incident name in the list corresponds to a certain area with the same color on the map, indicating the range of that incident. If the incident that the user was previously working on is still active, it will be selected as default. In DIORAMA, each incident is bounded to a range that needs to be specified by the incident creator before it's created. During the rescue process, it can be extended automatically or manually.

In the panel on the left, several options are offered: the incident list, as described above, is a list of all active incidents; "create" button brings users to incident creation process, which will be introduced in the following paragraphs; "show all" button switches between

showing the active incidents and all incidents. This is a way to review and enter those incidents that are already terminated. Finally if users tap on the enter button, they will enter the selected incident.

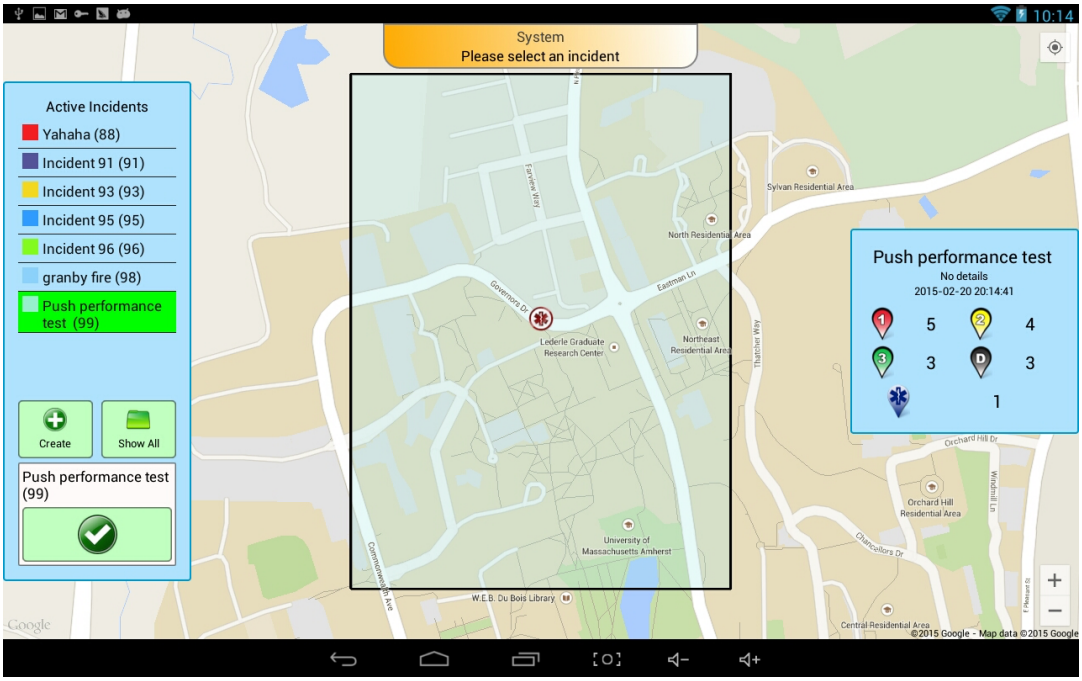
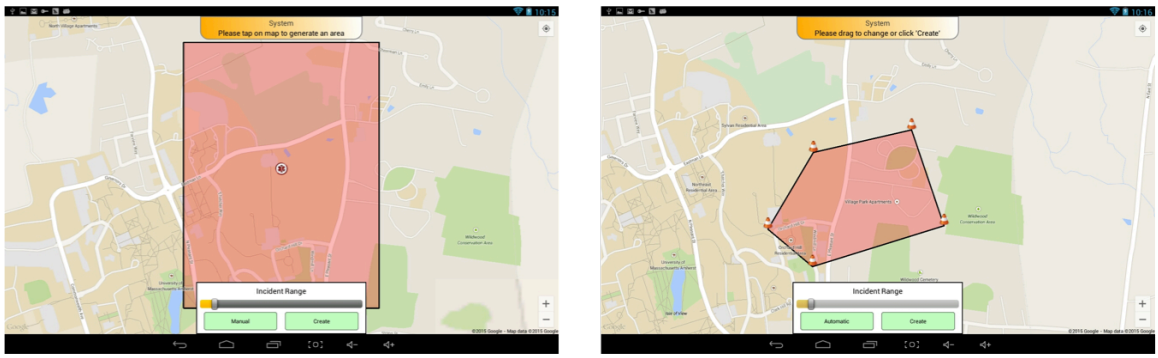


Figure 37: Details of an Incident

When user clicks on an incident in the list or the relevant area, the map will zoom in and load the details of that selected incident (Figure 37). This detail interface shows the basic information of an incident such as: incident name, description (if available), creation date and the number of POIs. This is designed to give users a summary about the incident's scale and scope, a way to assist users with their incident selections.

5.2 Incident Creation

Creating an incident includes specifying its area and filling other parameters. There are two ways of designating the incident area: automatic mode allows users to create a rectangle area in the pointed location; manual mode allows them to draw any irregular areas manually. These two modes of creating an incident are shown in Figure 38.



(a) Automatic Mode (b) Manual Mode

Figure 38: Creating an Incident

In the automatic mode (Figure 38a), users can click on any location on the map to determine the center of the incident; then a rectangle area will be drawn automatically. Users can either change the location of the center by tapping on another position, or adjust the size of it by dragging the sliding bar. The shaded area will change its size and location timely according to the users' actions. Automatic mode offers user a quick way to create an incident, with only few clicks and less than 30 seconds.

When users want to create a more complicated area, they can use manual mode (Figure 38b). This is quite similar to the process of creating an area-targeted command described in 3.1.3 . Each tapping on the map creates a draggable vertex and when the area is closed, it will be filled with color.

After users successfully specify the area for the incident, they can choose to enter more details about the incident or leave it default (Figure 39). This is also designed for a better user experience: users are able to create an incident within as few as three clicks, or spend time filling in more details to describe the incident.

Figure 39: Incident Confirm Dialog

5.3 More Operations in the Incident

While working in an incident, users can always go back to the incident overview. The menu bar of DIORAMA apps offers three options: exit, switch and end incident (Figure 40).

- **Exit incident**: exit the current incident and return to incident overview.
- **Switch incident**: quickly switch to another incident, without returning to the incident overview. A list will pop up for users to choose (Figure 41).
- **End incident**: terminate the current incident and make it no longer active.

Figure 40: More Operations in the Incident

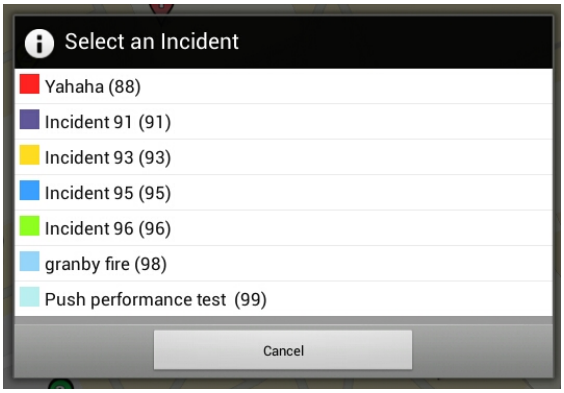


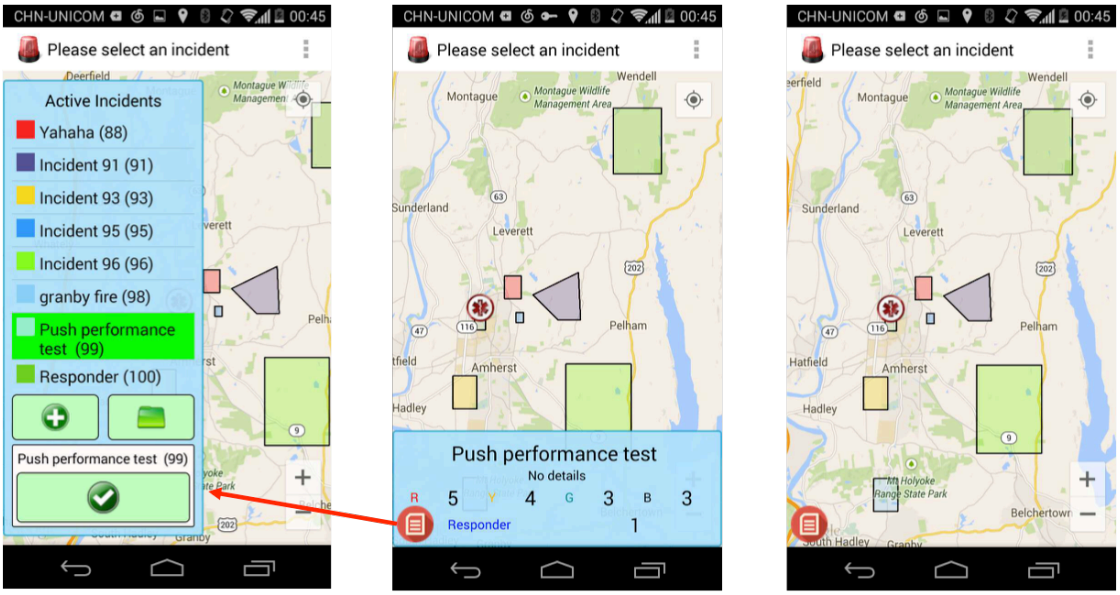
Figure 41: Incident List for Switching

5.4 Incident Manager of Responder App

The incident manager of responder app is quite similar. Due to the limitations of cellphone screen sizes, it would be improper to display both incident list and incident detail panels all at once. Figure 42 shows the compromised solution of responder app design. When users log in, instead of showing them the incident list, the responder app displays the

details of the default incident (Figure 42b). Users can click on the detail panel to enter the incident directly. The incident list will appear from the left when users click on the button on the bottom left (Figure 42a). In this panel users are able to perform the operations such as incident selection and creation as mentioned in 5.1 . When any incident is selected, the app pops up the detail panel and zooms to the incident area.

To help users inspect those incidents, both panels will be hidden automatically when users move the map (Figure 42c). This leaves them only a single map to work on and minimizes their distraction from UI elements.



(a) Incident List (b) Incident Details (c) Incident Map

Figure 42: Incident Manager of Responder App

Just like commander app, users can always go back to the incident overview without log out the system.

5.5 Conclusion

Incident manager provides DIORAMA a convenient and easy-to-use tool to manage multiple incidents simultaneously. Through this module, different incidents are visualized as areas in the map. The feedbacks from DIORAMA users of current version illuminated that

the incidents are now much easier to be identified and creating a new one is also simpler than the previous versions.

CHAPTER 6

AREA DIVIDING ALGORITHM

6.1 Introduction

Well-designed user interfaces of DIORAMA applications help emergency personnel find and rescue victims more efficiently. However, there are more intelligent features can be used to further reduce the time of rescue and evacuation. In this section I'll state a function that could help commanders to assign evacuation tasks faster: area auto-dividing.

Area auto-dividing is a function that can be used at the beginning of evacuation stage. Instead of assigning evacuation command one by one or drawing areas manually, the area-dividing module automatically generates the evacuation areas according to the number of responders and the distance of victims. The module will try to make each area "equally divided", offering each responder a same workload of evacuation (introduced in the next section). Commanders can either send the areas directly to responders or make changes based on them. An example of dividing result is shown in

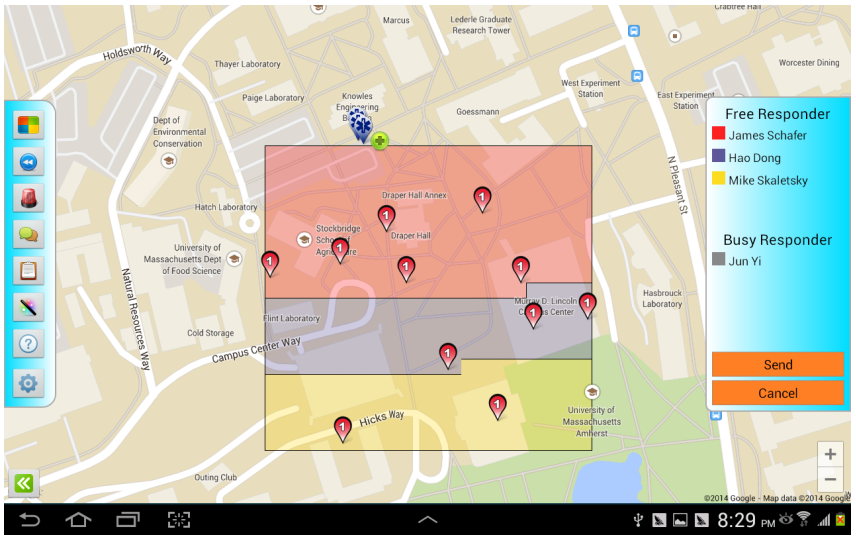


Figure 43: Example of Dividing Result

6.2 The Base of Area Dividing

Before starting introduce the dividing algorithm, a basic assumption used in this module needs to be illustrated. Here, in area dividing module, the app is trying to divide the whole area into "equally valued" parts. "Equally valued" means that all divided parts have the same workload for responders. So in the way of defining "workload of responders", I assume that: the time differences between rescuing different victims are mostly depending on the distance between the victims and the treatment areas. In another word, the workload of evacuating a victim is determined by the distance that the responder has to travel. In this assumption, the time of working on a victim (judging the injury level or putting him on/off a stretcher) is considered same; the time spent on the way of getting a victim and coming back makes a different. Therefore, the purpose of the algorithm is to ensure that in each area the total distance that a responder has to travel is almost equal. The details of the algorithm are described as following:

6.2.1 Select victims with the highest priority

Because victims with lower priorities are not evacuated until the rescue of higher ones is done, the first step of area dividing is filter out all victims with priority lower than designated. For example in Figure 43, only the victims in red are counted while others are filtered out. After the red ones are successfully evacuated, the area dividing module can switch to the yellow victims and re-calculate the areas accordingly.

6.2.2 Divide the area into smaller cells and weight them with distances

For the GPS data of the cellphone device is not accurate enough, the exact positions of victims may vary within several meters. So in the algorithm we use a "cell" to indicate a certain range of area.

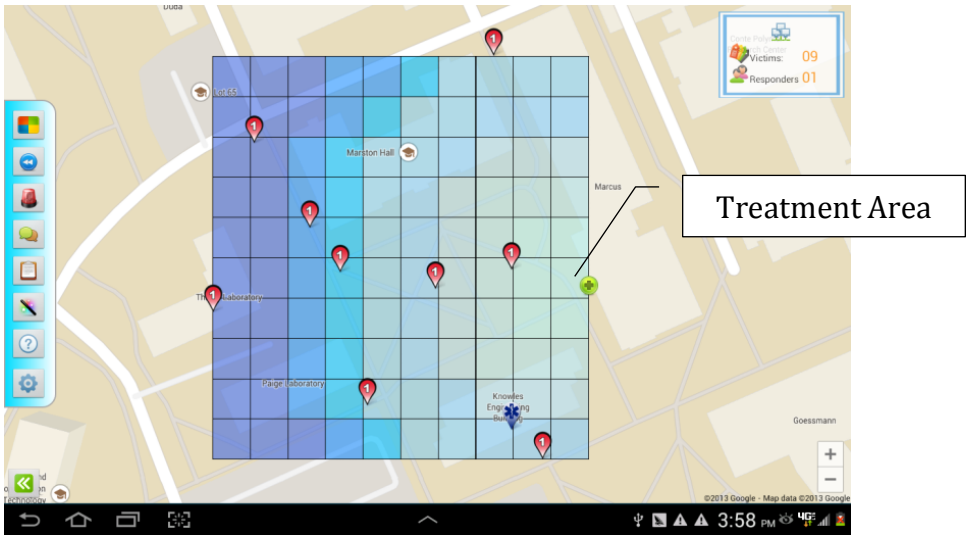


Figure 44: Cells in Area Dividing

As shown in Figure 44, the area boundary is automatically detected, including all red victims and the treatment area. Within this boundary, the area is divided into certain amount of smaller cells. These cells are stored as a matrix in the app and have the same size. They will not be displayed during a real dividing process.

All cells are given a weight value according to its distance from the treatment area. In the figure, the weight of each cell is represented by the depth of its color. This value can be used in the next step for calculation.

6.2.3 Calculate the areas using simple dynamic programming

After all cells are created and weighted, the system will scan all of them horizontally and extract the ones with victims. Those cells are representatives of victims including the effect of GPS inaccuracy. The extracted cells form a list, and for each element in the list another value is calculated: workload value, indicated as V_w. V_w is the value that is used to represent the exact workload of cells. It can be calculated by the formula:

$$V_w = \text{weight of the cell} \times \text{number of victims}$$

When there is only one victim in the cell, the workload value is equal to its weight; otherwise, it's the multiplication of victim number and its weight.

Using the workload value of each cell, an algorithm is used to determine the area division. The algorithm firstly sums up the V_w of all victim cells. Then, an average value V_{avg} is calculated by dividing this sum by the number of responders. Next, the purpose of the algorithm turns to divide the area into certain parts that in each part, the total workload value is as close to V_{avg} as possible. Only in that way, each area will have almost an equal workload value.

5	8	7	6	4	2	9	5	4

Figure 45: Example of Area Dividing Algorithm

Figure 45 is an example of the victim cells and the number in each cell is its workload value. Suppose the area is to be divided into two, it's easy to calculate that $V_{avg} = 25$. To find the two areas with workload value closest to V_{avg}, a greedy algorithm is used as following:

if $\left|(sum_n + v_{n+1}) - V_{avg}\right| \geq \left|sum_n - V_{avg}\right|$

return $cell_n$ as a divider cell

else

$sum_{n+1} = sum_n + v_{n+1}$

The algorithm keeps trying to add up the next element and determine if it is closer to V_{avg} until the closest value is found. The last cell that is counted by the algorithm (the cell with workload value 6 in the example) is the divider cell that will be used as a boundary of different areas. The greedy algorithm finally works out all divider cells and passes them to the next step.

6.2.4 Compare the results of different orientations and select the best one

Through the algorithm mentioned above a horizontal division result can be obtained. Nevertheless, sometimes there are better solutions if the parameters of the calculation are changed. Orientation is one of them. The orientation of scanning the cells can be defined at the beginning of the algorithm; it will affect the order of victim cells. As the order is changed,

a very different dividing result may be generated. Figure 46 is an example of the vertical division. Compared to the horizontal division shown in Figure 43, the cutting line of vertical division is vertical to the area. To generate a better result, different calculations are executed using different parameters. The result is compared at the end and the best value is selected.

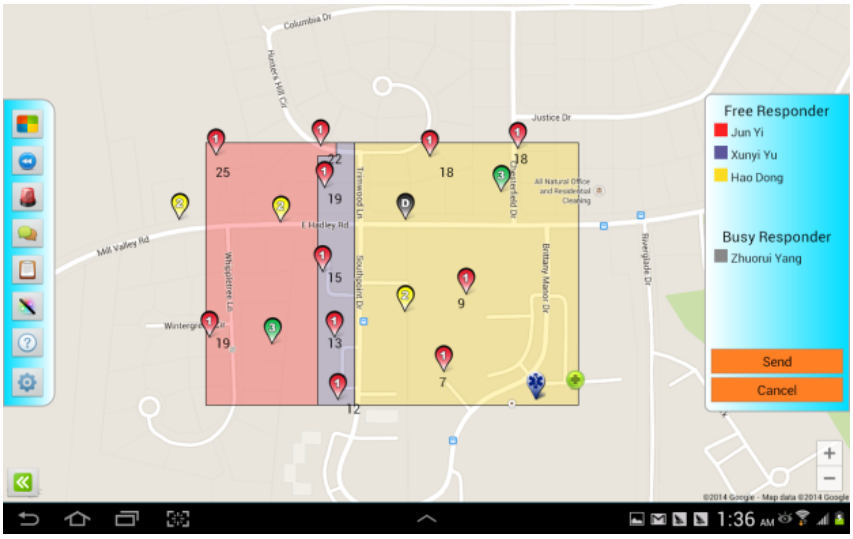


Figure 46: Vertical Division

6.2.5 Re-uniform the cells and draw the areas on the map

When the division is done and divider cells are picked up, the final job of the algorithm is to re-uniform these cells into complete areas and draw them on the map. Then each area and responder is denoted as different colors, as shown in Figure 43.

6.2.6 Adjustment and Modification

In Figure 46, the number of divided areas depends on how many responders available at that moment. Before running the algorithm, system will check all responders and filter out all busy responders (any responder with incomplete commands) by default. These responders are not counted in dividing process. However, system also allows users to add or remove responders within the list. By simply clicking a responder in the available list, commanders can move the responder to the "busy list" and the areas will be re-calculated

based on the new “available list”. Similarly, when a responder in “busy list” is clicked, he will be moved to “available list” and counted in the division. In such way, it’s more flexible and more convenient, giving commanders more options to generate the result they expect.

The result of division is generated through the dividing algorithm automatically, but it also offers commanders the ability to modify the result. When the modification is enabled, all vertexes of the areas will be displayed as icons (as shown in Figure 47). Users can drag those icons to another position and the shape of area will also be changed accordingly. Based on the areas generated by the system, commanders can feel free to modify the shape of them and customize the result.

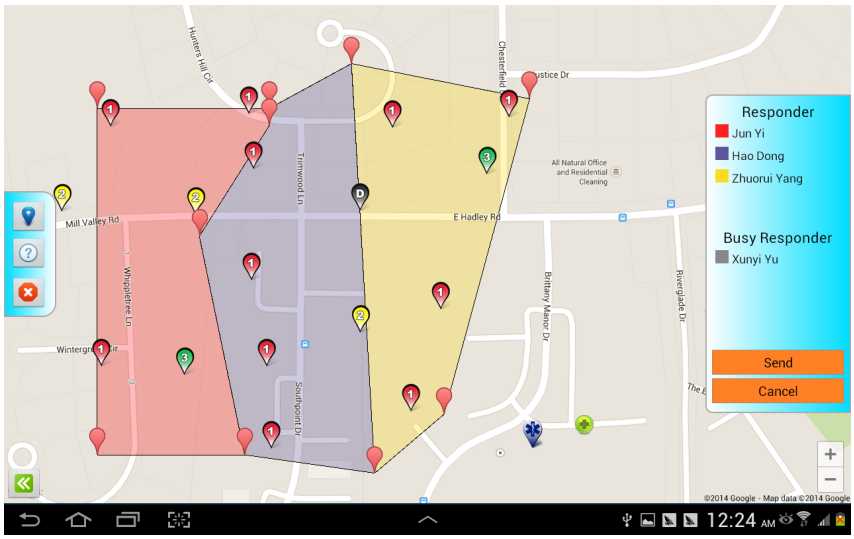


Figure 47: Result Modification

6.2.7 Sending the Result

After division and modification, areas can be sent to responders according to the colors listed in the responders’ list. Each responder will only receive the area assigned to him. The result is shown in Figure 48.

Figure 48: Dividing Result

6.3 The Improved Dividing Algorithm

The parameter that area-dividing module is currently using is the distance between each victim and the treatment area. However, this is inaccurate in reality. The algorithm determines the distance between victims and the treatment area by the straight-line distance on the map and ignores all obstacles, which is too idealized. Actually in real situations like the trials we did on campus, buildings, trees or barriers can stop the responders from passing through, significantly increasing the travel time between two points.

One solution to this issue is using the imbedded navigation system in Google Map. Google Map offers its users and developers the tool to navigate between two locations. It uses the data of buildings and paths that stored in Google server to generate the shortest route. Nevertheless, this is not the best solution for DIORAMA because:

1) The Google Map navigation highly depends on the data collected by Google itself. It is sometimes not detailed enough (especially in some outlying areas) to include all buildings and obstacles, or not dynamic enough to reflect the latest changes in the area.

2) The scene could be changed completely after a disaster happens. Collapsed buildings, broken bridges, floods etc. can all cut off an existing path and force the responders to look

for another. This might increase the time of reaching a victim as well, while Google Map does not consider these factors when generating its route.

In order to improve the accuracy of the dividing algorithm, DIORAMA takes the full advantage of the data collected during the triage process. It is observed that the areas which responders always passed through during the triage process are more likely to be obstacle-free and easy to access.

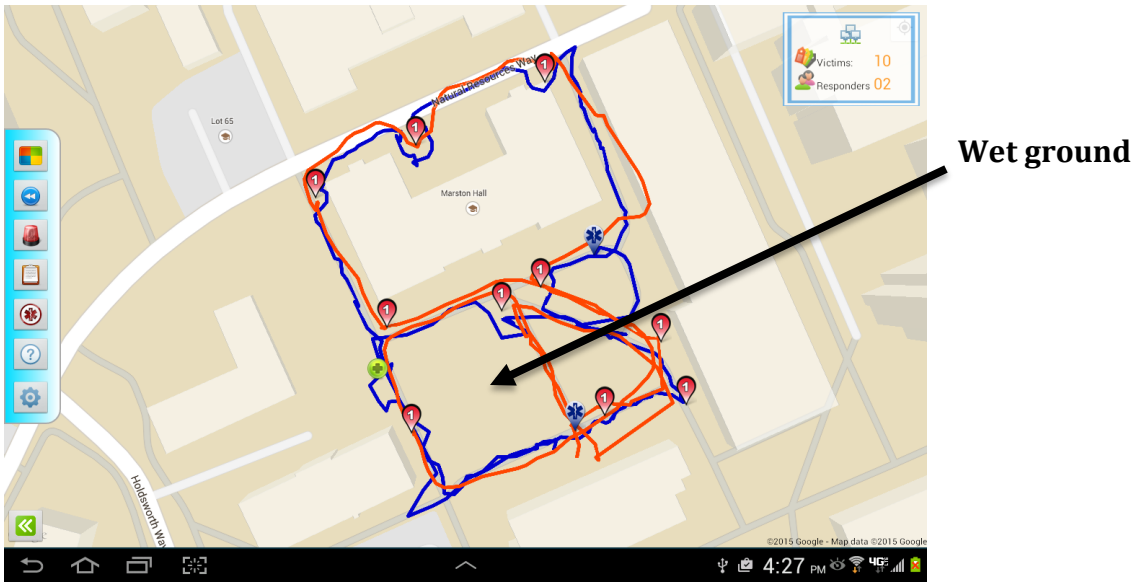


Figure 49: Responder Travel Paths

Figure 49 shows the paths of responders during the triage stage in one of the trials. When responders were wandering around in the field and searching for victims, they tended to avoid the obstacles and look for the easiest way to travel. They walked along the edge of a building to find the victim behind it instead of going through it. The square indicated by the arrow was a piece of grass but soaked with rainwater at that time. Usually responders will pass through this squire when it's dry enough; but in this trial, while the grass was too wet to travel, the responders were not apt to step on it if not necessary - even though sometimes it was closer if they go across it.

Based on the responders' travel patterns analyzed above, the areas or pathways that have been traversed by responders are more likely to be unobstructed and accessible than

those have never been touched. When more data are collected, their travel patterns will be more clustered and characteristic, available for identifying the areas that should be preferred or avoided, generating estimated shortest paths between victims and treatment area, and finally creating a more realistic dividing result without additional cost.

The implementation of this improved algorithm is shown below:

1) Get the historical locations of all responders

Firstly, the commander app queries DIORAMA server database and obtains all past locations of responders in this incident. They constitute a list of geographical points distributed in the incident area.

2) Generate the cells of the area using the points

As mentioned in 6.2.2 , dividing module divides the entire area into smaller cells and assigns each of them a weight value. In the basic algorithm, this weight value is determined by the straight-line distance between the cell and the treatment area, which is inaccurate as described above. The improved algorithm, however, determines the weight of each cell by the previous locations of responders during triage stage.

At first, all cells are assigned to the same weight, such as ten. Then the algorithm traverses the responder location list, retrieves each single geographical point, and fills in the corresponding cell. The weight of each cell will be subtracted by a specific number (such as one) when any point falls into it. As a result, the cells that contain more responder's footprints will have smaller weights, while the ones with fewer will have a larger value.

As Figure 50 shows, the more times a cell was visited by responders, the smaller weight it has (indicated by the depth of its color in the graph). From the figure it is easy to discover that the cells with smaller weights (or lighter colors) are usually sideways and accessible areas; the buildings and wet grass, e.g. the inaccessible areas, are covered with large-weighted cells (or deeper colors). Therefore, the weight value of each cell can be

considered as a representation of the difficulty of traveling through that area. The cells and their weights are then used in the shortest path generation in the next step.

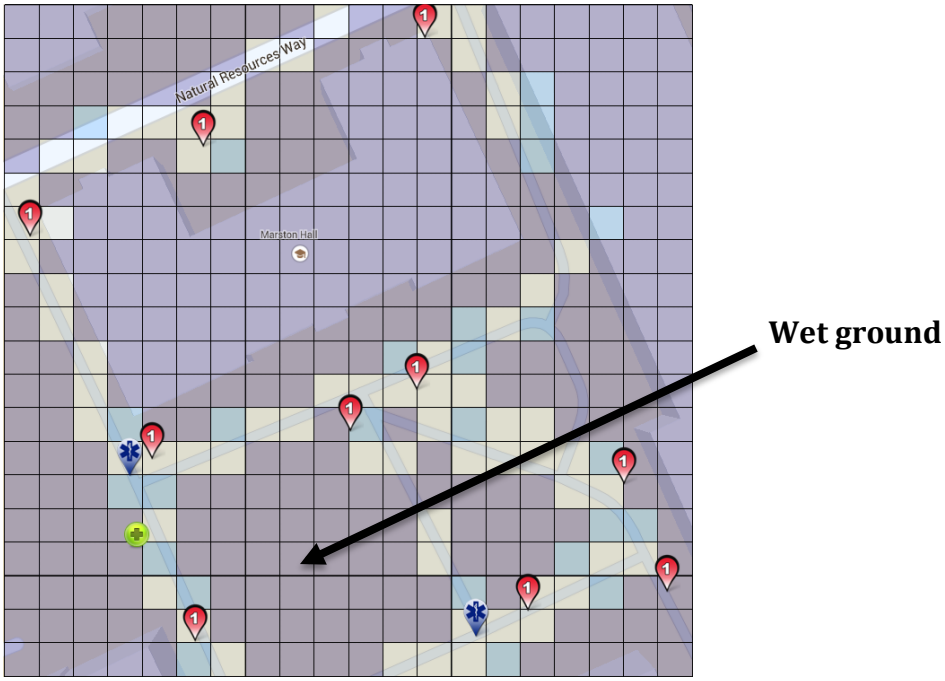


Figure 50: Cells in Improved Algorithm

3) Generate a shortest path for each victim to the treatment area

After dividing the area into several cells with their weight values, the whole problem turns into finding a shortest path between two cells in a weighted graph. To solve this mathematic problem, Dijkstra's algorithm is implemented in the new area-dividing module. Dijkstra's algorithm is an algorithm for finding the shortest paths between nodes in a graph. It fixes a single node as the "source" node and finds shortest paths from the source to all other nodes in the graph, producing a shortest path tree[25].

In the algorithm, cells are considered as vertexes or nodes. A virtual edge is created between each two vertexes with a weight value

$$w_{edge} = w_{Cell_1} + w_{Cell_2}$$

Where w_{Cell_1} and w_{Cell_2} are the weight of each cell respectively. Hence, the weight of an edge between two cells depends on the weights of both cells. The Dijkstra's algorithm then

takes the cell with treatment area as source node, traverses the other nodes and generates a shortest path for each “victim node”.

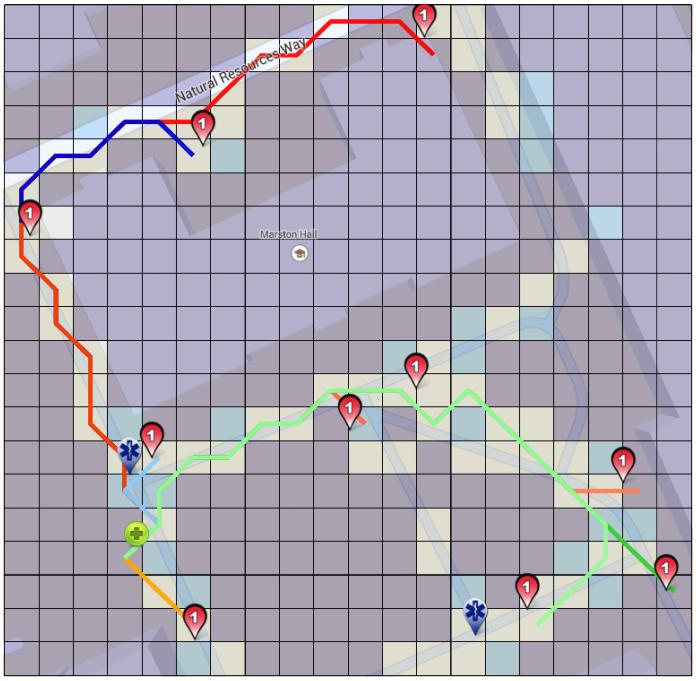


Figure 51: Shortest Paths for Victim Cells

Figure 51 shows the shortest paths generated for each victim. As shown in the graph, Dijkstra’s algorithm takes the weight of every edge into calculation and tries to find a path with the smallest amount of cost. The paths generated by this algorithm naturally overlap with the sideways on the map and automatically avoid most of the inaccessible areas.

4) Calculate the distances by the generated paths

After the shortest paths of all victims are generated, the distance of each path can be calculated as well. The algorithm calculates the total amount of cells that each path goes through and takes it as the distance score of that victim cell. Compared to the simple straight-line distances of victims, the values calculated using this improved algorithm are more realistic because they take the obstacles and the unique situations of different incidents into account.

The final dividing result can be then created using the same processes from 6.2.4 to 6.2.7 . After the algorithm finishes, the shortest paths and those divided areas will be both displayed (Figure 52), allowing the commander to review and send them.

Figure 52: The Result of Enhanced Area Dividing

CHAPTER 7

TRIALS AND RESULTS

Several trials and simulations have been performed to test the applications on the campus of the University of Massachusetts Amherst. Certified responders from the local EMS unit of UMASS Amherst are invited to the trials and 50 student volunteers participated, acting as victims. Responders are asked to use DIORAMA applications to triage and evacuate the victims. Both apps are fully tested and the feedbacks of using DIORAMA apps were recorded. More detail about the trials can be found in [7].

The result of the human trails shows that:

- All responders felt in control of evacuation process
- All responders liked mobility of DIORAMA system
- All of them thought it useful in managing extraction of patients
- All of them found it helpful at managing resources
- They all thought that DIORAMA was beneficial during triage and evacuation trials
- They all mentioned that they would use the system in an actual incident

CHAPTER 8

CONCLUSION AND FUTURE WORK

In previous sections we described the design and implementation of DIORAMA applications. Both of them are designed based on requirements and feedbacks of real trials. A comprehensive communication solution is also developed and tested to handle the data transmission between all the components of DIORAMA. The test result shows that both apps of DIORAMA are easy to learn and use and could make the rescue more efficient as well.

The future work of DIORAMA applications includes the following:

Firstly, the shortest paths generated from area division algorithm (Chapter 6.3) can be used to give responders suggestions of how to get to the victims. Because the routes are generated using the previous experiences of triage, emergency personnel can easily discover the obstacles, inaccessible areas and find a best path.

Secondly, in paper triage system when a responder is scanning an area for patients, it's hard for him to know which area is scanned and which is not. The responder might scan the same area for more than once or on the contrary miss an area and some victims. Although DIORAMA offers the responder a digital map and tracks his location frequently, this could still happen. Therefore, the app should be able to remind responders of the areas already covered.

Area coverage module uses the location data of responders stored in database and displays the possible covered areas in a visualized way. Combined with the area of commands, responders could quickly find the area they have already scanned and head to the uncovered areas. In this way, repeated scanning and missing victims can all be avoided, helping responders work in the field with more confidence.

Thirdly, DIORAMA can also be integrated with other technologies like robots or Bluetooth beacons. They can make the whole system more intelligent.

Finally, the interfaces of DIORAMA apps should still keep improving to offer users the best experience.

BIBLIOGRAPHY

[1] J. Chaves, A. Donner, C. Tang, C. Adler, M. Krusmann, A. Estrem, T. Greiner-Mai, "An Interdisciplinary Approach to Designing a Mass Casualty Incident Management System", *Wireless Personal Multimedia Communications, 14th International Symposium*, Oct. 2011

[2] D. Rodriguez, S. Heuer, C. Kunze, B. Weber, "Management of mass casualty of incidents using an autonomous sensor network", *Wireless Personal Multimedia Communications, 14th International Symposium*, Oct. 2011

[3] T. Mizumoto, S. Imazu, W. Sun, N. Shibata and K. Yasumoto, "Emergency Medical Support System for Visualizing Locations and Vital Signs of Patients in Mass Casualty Incident", *Pervasive Computing and Communications Workshops, IEEE International Conference*, Mar. 2012

[4] E. Artinger, P. Maier, T. Coskun, S. Nestler, M. Mahler, Y. Yildirim-Krannig, F. Wucholt, F. Echtler and G. Klinker, "Creating a common operation picture in realtime with user-centered interfaces for mass casualty incidents", *Pervasive Computing Technologies for Healthcare, 6th International Conference*, May 2012

[5] Liliya I. Besaleva, Alfred C. Weaver, "Applications of Social Networks and Crowdsourcing for Disaster Management Improvement", *Social Computing, 2013 International Conference,* Sept. 2013

[6] K. M. Rahman, T. Alam, M. Chowdhury, "Location Based Early Disaster Warning and Evacuation System on Mobile Phones Using OpenStreetMap", *Open Systems (ICOS), 2012 IEEE Conference*, Oct. 2012

[7] Wang Ze-gen, YangYan-mei, Xiong Jun-nan, "Study on the Earthquake Cooperative Emergency Rescue Auxiliary System", *Wireless Communications, Networking and Mobile Computing*, 2009

[8] Ciprian Lupu, Vasile Olaru, Dorel Bivolan, Andreea Udrea, "Implementation of a telemedicine system for optimal on site medical response in case of disasters and for emergency situations management", *E-Health and Bioengineering Conference (EHB)*, 2013

[9] S. L. Toral et al. "A wireless in-door system for assisting victims and rescue equipments in a disaster management", *Intelligent Networking and Collaborative Systems (INCOS)*, pp. 502 – 506, 2010

[10] SUMMIT, https://dhs-summit.us/index.html

[11] Ronny Klauck, Michael Kirsche, "XMPP to the Rescue: Enhancing Post Disaster Management and Joint Task Force Work", *Pervasive Computing and Communications Workshops (PERCOM Workshops)*, pp. 752-757, 2012

[12] Stephen M. George et al. "DistressNet: A Wireless Ad Hoc and Sensor Network Architecture for Situation Management in Disaster Response", *Communications Magazine*, IEEE, pp. 128 – 136, Vol. 48 Iss. 3, March 2010

[13] Ganz, J. M. Schafer, Z. Yang, J. Yi, G. Lord, G. Ciottone, "Mobile DIORAMA: Infrastructure less Information Collection System for Mass Casualty Incidents", *submitted to 36th Annual International IEEE EMBS Conference of the IEEE Engineering in Medicine and Biology Society*, Mar. 2014

[14] Ganz, J. Schafer, X. Yu, G. Lord, J, Burstein, G. Ciottone, "Real-time Scalable Resource Tracking Framework (DIORAMA) for Mass Casualty Incidents", *International Journal of E-Health and Medical Communications,* 4(2), April-June 2013, pp34-49.

[15] Near field communication, Wikipedia, http://en.wikipedia.org/wiki/Near_field_communication

[16] Gregory R. Ciottone, “Disaster Medicine”, 3rd Edition, Mosby Inc.

[17] Jeff Johnson, “Designing with the Mind in Mind”, Morgan Kaufmann

[18] Jakob Nielsen, “Usability Engineering”, Academic Press

[19] Davic McCandless, “The Visual Miscellaneum: Reviced and Reimagined”, HarperCollins

[20] Notifications, http://developer.android.com/guide/topics/ui/notifiers/notifications.html

[21] Google Maps Android API v2, http://developer.android.com/google/play-services/maps.html

[22] Pull technology, http://en.wikipedia.org/wiki/Pull_technology

[23] Push technology, http://en.wikipedia.org/wiki/Push_technology

[24] SignalA, Github, https://github.com/erizet/SignalA

[25] Dijkstra’s Algorithm, http://en.wikipedia.org/wiki/Dijkstra%27s_algorithm