Next Generation Emergency Call System with Enhanced Indoor Positioning

Wonsang Song

Submitted in partial fulfillment of the
requirements for the degree
of Doctor of Philosophy
in the Graduate School of Arts and Sciences

COLUMBIA UNIVERSITY

2014

ABSTRACT

Next Generation Emergency Call System with Enhanced Indoor Positioning

Wonsang Song

The emergency call systems in the United States and elsewhere are undergoing a transition from the PSTN-based legacy system to a new IP-based system. The new system is referred to as the Next Generation 9-1-1 (NG9-1-1) or NG112 system. We have built a prototype NG9-1-1 system which features media convergence and data integration that are unavailable in the current emergency calling system.

The most important piece of information in the NG9-1-1 system is the caller's location. The caller's location is used for routing the call to the appropriate call center. The emergency responders use the caller's location to find the caller. Therefore, it is essential to determine the caller's location as precisely as possible to minimize delays in emergency response. Delays in response may result in loss of lives.

When a person makes an emergency call outdoors using a mobile phone, the Global Positioning System (GPS) can provide the caller's location accurately. Indoor positioning, however, presents a challenge. GPS does not generally work indoors because satellite signals do not penetrate most buildings. Moreover, there is an important difference between determining location outdoors and indoors. Unlike outdoors, vertical accuracy is very important in indoor positioning because an error of few meters will send emergency responders to a different floor in a building, which may cause a significant delay in reaching the caller.

This thesis presents a way to augment our NG9-1-1 prototype system with a new indoor positioning system. The indoor positioning system focuses on improving the accuracy of vertical location. Our goal is to provide floor-level accuracy with minimum infrastructure support. Our approach is to use a user's smartphone to trace her vertical movement inside buildings. We utilize multiple sensors available in today's smartphones to enhance

positioning accuracy.

This thesis makes three contributions. First, we present a hybrid architecture for floor localization with emergency calls in mind. The architecture combines beacon-based infrastructure and sensor-based dead reckoning, striking a balance between accurately determining a user's location and minimizing the required infrastructure. Second, we present the elevator module for tracking a user's movement in an elevator. The elevator module addresses three core challenges that make it difficult to accurately derive displacement from acceleration. Third, we present the stairway module which determines the number of floors a user has traveled on foot. Unlike previous systems that track users' foot steps, our stairway module uses a novel landing counting technique.

Additionally, this thesis presents our work on designing and implementing an NG9-1-1 prototype system. We first demonstrate how emergency calls from various call origination devices are identified, routed to the proper Public Safety Answering Point (PSAP) based on the caller's location, and terminated by the call taker software at the PSAP. We then show how text communications such as Instant Messaging and Short Message Service can be integrated into the NG9-1-1 architecture. We also present GeoPS-PD, a polygon simplification algorithm designed to improve the performance of location-based routing. GeoPS-PD reduces the size of a polygon, which represents the service boundary of a PSAP in the NG9-1-1 system.

Table of Contents

List of Figures

List of Tables

Acknowledgments

First of all, I would like to thank my advisor, Prof. Henning Schulzrinne. I am very fortunate to work with him as his PhD student. He taught me to be practical, see the big picture, but also take care of details. I am sure that these lessons will be helpful not only in my professional career, but also in my personal life. I am also grateful for his shepherding during my PhD study. Whenever I got lost in my research, his guidance always gave me the right direction. He was always patient and helped me find my way by myself. Without his guidance and patience, I would have never finished my PhD study.

I would like to thank the rest of my thesis committee: Prof. Gail Kaiser, Prof. Angelos Keromytis, Prof. Archan Misra, and Dr. Thierry Klein. Their criticism and feedback helped me shape my research and improved my dissertation. I truly appreciate their time and effort to serve as my thesis committee.

I would like to thank the entire NG9-1-1 project team. Jong Yul Kim has worked with me all the time since we joined the team together from the beginning of our PhD study. His excellent system design and programming skill solved all technical challenges we had during the project. I am also grateful to other NG9-1-1 team members. Matthew Mintz-Habib, Anshuman Rawat, and Amrita Rajagopal developed the early prototype system that became the foundation of our prototype system. Kundan Singh and Jonathan Lennox helped us customize their SIP stack for our prototype system. Xiaotao Wu provided the SIP client code that became the core of our caller and call taker software. I also thank the NG9-1-1 team in Texas A&M University and Booz Allen Hamilton. Dr. Walt Magnussen gave us insight into telecommunications networks and helped us demonstrate our prototype system in many NENA annual conferences. Anna Zacchi, Anupam Jain, Harshavardhan Chenji, Chris Magnussen, Chris Norton, Ian Schworer, and Ken Trinh helped us design and implement the NG9-1-1 POC system during the US DOT project.

I would like to thank all my colleagues at the IRT lab. I cannot thank Jae Woo Lee enough for all his help during my stay in Columbia University. He helped me design the architecture of GeoPS-PD and indoor positioning system, and improve algorithms in the systems.

The work described in this thesis has been funded in part by NTIA grant 48-60-04012, U.S. Department of Transportation, Verizon Communications, and NSF grants CNS-0751094.

I must give thanks to my parents, Jang Sung Song and Myung Sook Kim for all their support and patience. They always pray for me and now is the time for me to pray for them. And many thanks to my parents-in-law, Hong Kyu Choi and Myung Hee Shin.

Lastly but most importantly, I thank my wife, Jung Hyun Choi for EVERYTHING. Thank you and love you, Honey!

To my parents.

Chapter 1

Introduction

The Next Generation 9-1-1 (NG9-1-1) project in the United States aims to replace the current emergency communication systems based on PSTN technology with a new system based on IP technology. The U.S. Department of Transportation successfully completed the NG9-1-1 Initiative [110] to study the feasibility of transition and to estimate the deployment cost. Similar efforts are underway in other parts of the world. In Europe, the REACH112 project [23] has integrated video and real-time text into the emergency communication systems across five European countries.

The NG9-1-1 system provides a number of functionalities that are unavailable in the current PSTN-based systems. First, the NG9-1-1 system supports multimedia communications. People can make video calls, send photos, or exchange text messages to report emergency incidents. Multimedia support can be especially beneficial to the disabled. For example, the deaf and hard-of-hearing can benefit from the ability to send text messages in emergency situations. If video calls are available, they can also communicate using sign languages with emergency dispatchers.

Second, emergency communications in the NG9-1-1 system carry more detailed information which can be helpful for emergency responders. When an emergency call is placed, the call taker's screen can display various information related to the caller and the caller's location, such as the caller's health record or the floor plan of her residence. In case of a car accident, the vehicle's telematics device will automatically send the crash data. Additional information about the caller and the incident can reduce delays in emergency response and help the responders make better decisions at the scene.

Lastly, the NG9-1-1 system interoperates more efficiently with the networks of telecommunication carriers. Many landline and wireless carriers are transitioning their networks to IP-based infrastructure in order to provide new features, enhance quality, and reduce cost. The call routing and data transfer between the carriers and the NG9-1-1 networks become much simpler and more efficient if they are all IP-based, since the interconnections do not require cumbersome conversions between the networks.

1.1 Problems and Challenges

This thesis identifies the following problems and challenges in designing and building the NG9-1-1 system.

Integrating legacy and next generation devices and networks The NG9-1-1 system must support various kinds of IP-based devices such as Voice over IP (VoIP) phones, real-time text devices, and video call software. This requires that the system should be based on a standardized communication protocol. In addition to the IP-based communication devices and networks, the NG9-1-1 system must support legacy methods currently in use for emergency communications. In particular, the system must provide a bridge for the old PSTN-based telephone lines. Integrating different devices and networks, both old and new, presents a significant challenge.

Routing emergency calls An emergency call should be routed to the appropriate Public Safety Answering Point (PSAP) based on the caller's location. The appropriate PSAP is the one whose service boundary includes the caller's location. The NG9-1-1 system uses the Location-to-Service Translation (LoST) architecture [63] for this location-based routing, but there can be performance issues on LoST servers, considering the large volume of queries the servers must handle. Moreover, the boundary information of PSAPs can be very large in size, so storing the boundary information and finding an enclosing boundary for a particular location can be burdensome for LoST servers. Improving the speed and efficiency of emergency call routing is an important problem that must be addressed before the NG9-1-1 system can be successfully deployed.

Determining callers' location The caller's location is the most important piece of information in emergency call systems. The location is first used to route the call to the appropriate PSAP. The emergency responders also use the caller's location to find the caller. GPS can provide a user's location accurately when the user makes an emergency call outdoors using a mobile phone. Indoor positioning for a mobile caller, however, presents a challenge because GPS does not generally work indoors. Moreover, unlike outdoors, vertical accuracy is very important in indoor positioning because an error of few meters will send emergency responders to a different floor in a building, which may cause a significant delay in reaching the caller. Determining the caller's vertical location inside a tall building is a critical problem for the NG9-1-1 system.

1.2 Contributions of the Thesis

In this thesis, we make the following contributions.

First, we design and implement **the NG9-1-1 Proof-of-Concept (POC) system**. Our POC system implements various functional and performance requirements set forth by the NG9-1-1 community. It deals with technical challenges such as determining a networked device's physical location, routing calls based primarily on that location, integrating different networks into a SIP-based backbone network, and sharing data between these networks. We successfully demonstrate the feasibility of the NG9-1-1 by deploying our system at five live PSAPs across the United States. The call takers at each PSAP tested the POC system's core functions including emergency call termination at the PSAP, location information and additional data delivery, and call transfer to other call takers or PSAPs.

Second, we integrate **Instant Messaging (IM) and Short Message Service (SMS) into the NG9-1-1 architecture**. For IM integration, we propose a SIP-based model of integration to handle proprietary IM protocols. To deliver multiple messages within a session to the same call taker, we implement state-keeping mechanisms in the routing components in the NG9-1-1 architecture. For SMS integration, we solve three core challenges: we add location information to SMS messages, we introduce a gateway architecture to bridge SMS networks to the SIP-based NG9-1-1 network, and we implement messaging session in order to delivery multiple messages to the same call taker.

Third, we present GeoPS-PD, **a polygon simplification algorithm designed to improve the performance of location-based routing** in the NG9-1-1 architecture. GeoPS-PD reduces the size of the polygons representing PSAP service boundaries while it minimizes the loss of area coverage. Unlike existing algorithms, GeoPS-PD never produces a false positive, is tunable at runtime for the desired balance between the target polygon size and area coverage, and optionally takes into account the population density to result in better population coverage. The simplified service boundaries can be cached in mobile clients with limited processing power and storage, which in turn can reduce service latency and server load.

Lastly but most importantly, we present **a new indoor positioning system to determine callers' vertical location inside tall buildings**. Our system is a hybrid architecture that combines beacon-based infrastructure and sensor-based dead reckoning, striking a balance between accurately determining a user's location and minimizing the required infrastructure. We present three analysis modules to trace a user's vertical movements inside buildings using smartphones' inertial sensors. The elevator and the escalator module track a user's movement in an elevator and on an escalator. The modules address three core challenges that make it difficult to accurately derive the user's traveled distance from the acceleration. The stairway module determines the number of floors a user has traveled on foot. Unlike previous systems that track users' foot steps, our stairway module uses a novel landing counting technique.

1.3 Overview of the Thesis

This thesis is divided into two parts. Part I describes our NG9-1-1 prototype system, which has been a part of the NG9-1-1 Proof-of-Concept Initiative led by the U.S. Department of Transportation. The NG9-1-1 prototype system motivated us to investigate the floor localization problem inside tall buildings. Part II describes our resulting floor localization system which is the main contribution of this thesis. A brief overview of the chapters is as follows.

Chapter 2 presents the architecture of the NG9-1-1 prototype system and software components we designed and implemented during the NG9-1-1 Proof-of-Concept Initiative. We then describe main features of our prototype system such as conferencing, call transfer, and call queue. This chapter also includes our deployment scenario during the POC trials.

Chapter 3 shows how we integrate text communications into the NG9-1-1 system. We identify the technical challenges in the integration of IM and SMS networks with the NG9-1-1 system, and propose a solution for each challenge. We also describe a working prototype system using our approach.

Chapter 4 presents GeoPS-PD, a polygon simplification algorithm designed for location-based service (LBS) applications. We discuss the requirements of the polygon simplification algorithm for LBS applications and describe our algorithm, especially our adaptive approach–taking the population density into consideration for better user coverage. We evaluate the efficacy of GeoPS-PD using the US state boundary data.

Chapter 5 starts the discussion of our indoor positioning system by presenting an architecture overview. It then delves deeper into the design and algorithms of our three analysis modules and the activity manager. The chapter also provides the implementation details and evaluation results we conducted with the prototype system running on an iOS device.

Part I

Next Generation 9-1-1 Prototype System

Chapter 2

Designing and Implementing an NG9-1-1 Proof-of-Concept System

2.1 Introduction

The Next Generation 9-1-1 Proof-of-Concept system (NG9-1-1 POC system) is an IP-based system developed within the NG9-1-1 initiative driven by the U.S. Department of Transportation and the National Emergency Number Association (NENA). The goal is to implement and deploy an all IP-based emergency communication system and also integrate it with legacy emergency communication systems used in public switched telephone networks (PSTN) and cellular networks. The design of the system is based on specifications from the Emergency Context Resolution with Internet Technologies (ECRIT) Working Group[1] in the IETF and NENA i3 solution [37].

NG9-1-1 POC system provides media convergence and data integration that is not possible with the current emergency communication system. For example, it is not possible to send a picture or video to the call taker using the current system, but NG9-1-1 is designed to allow such multimedia communications. Also, additional data such as floor plans, health records, or telematics data can be delivered to the call taker when they are needed.

Two main problems arise when trying to implement IP-based emergency communication

[1] http://tools.ietf.org/wg/ecrit/

systems. One problem is how to acquire the caller's location. Location is critical for routing the call and quickly dispatching help to the right place. Another problem is how to determine where to route the call. Given caller's location, the system has to route the call to the Public Safety Answering Point (PSAP) which serves the caller's location.

The POC system addresses the location acquisition problem using standardized location discovery techniques. These techniques either allow the end user's device to discover its own location (e.g., GPS, Link Layer Discovery Protocol for Media Endpoint Devices (LLDP-MED) [35], Dynamic Host Configuration Protocl (DHCP) [87,95]), or allow components in the network to determine location on behalf of the end user's device (e.g., DHCP).

Once the caller's location has been identified, the call origination network routes the call based on caller location to the NG9-1-1 network, specifically to an Emergency Services Routing Proxy (ESRP). The ESRP then routes the call towards the most appropriate PSAP using a combination of information such as caller's location, call origination type, and/or policy enforced by the authorities. In both cases of call routing, a service called Location-to-Service Translation (LoST) [63]. LoST is a mapping service that translates location and requested service type to a URL where the call can be routed to.

This chapter is organized as follows. Section 2.2 presents the architecture of NG9-1-1 POC system. Section 2.3 enumerates the software components and their implementation details. Section 2.4 describes main features of our system and Section 2.5 provides our test deployment scenario. In Section 2.6, we conclude and discuss future work.

2.2 System Architecture

The NG9-1-1 POC system is divided into two parts, namely the call origination network and the NG9-1-1 network, as shown in Figure 2.1. The call origination network is the network in which emergency calls are initiated such as Public Switched Telephony Network (PSTN), Voice over IP (VoIP) network, cellular Network, or telematics network. The NG9-1-1 network is an IP-based network shared by a group of Public Safety Answering Points (PSAPs). Each PSAP serves users within its service boundary. Therefore, each NG9-1-1 network covers a region that is the union of the boundaries of the PSAPs participating in

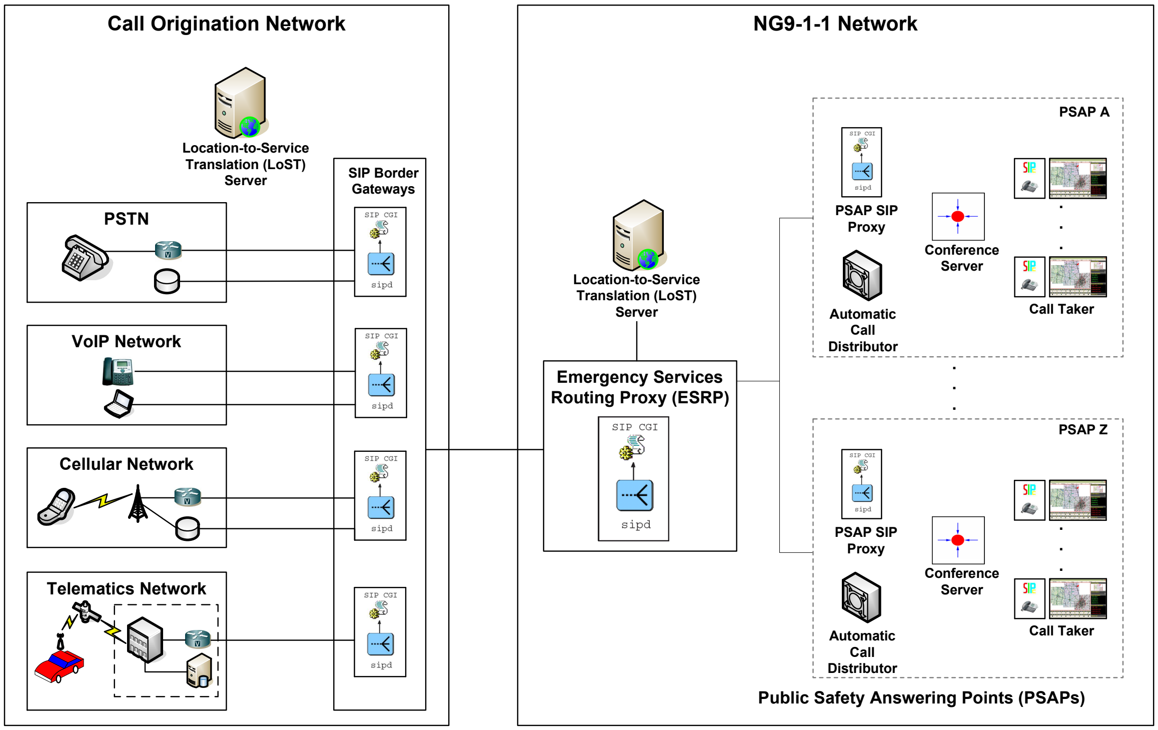


Figure 2.1: NG9-1-1 POC system architecture.

that NG9-1-1 network.

The NG9-1-1 network contains Emergency Services Routing Proxies (ESRPs), Location-to-Service Translation (LoST) servers [63], and IP-based PSAPs. The ESRP is a SIP routing entity that forwards the call to the most appropriate PSAP. There can be multiple NG9-1-1 networks that cover different regions and each NG9-1-1 network is identified by its ESRP URL. The LoST server is a location resolution entity that maintains the PSAP boundaries and PSAP URLs. An ESRP needs a LoST server in order to determine the PSAP URL based on the caller's location. Together, these two components provide the call routing function. Routing is primarily based on the caller's location, but it can also be affected by business rules or other policies in the NG9-1-1 network.

An IP-based PSAP contains a SIP proxy, an automatic call distributor, a conference server, and call taker workstations. The automatic call distributor is a signaling component that routes incoming calls to a call taker based on the PSAP's local policy. Voice and video streams are handled by the conference server. The IP-based PSAP provides the call

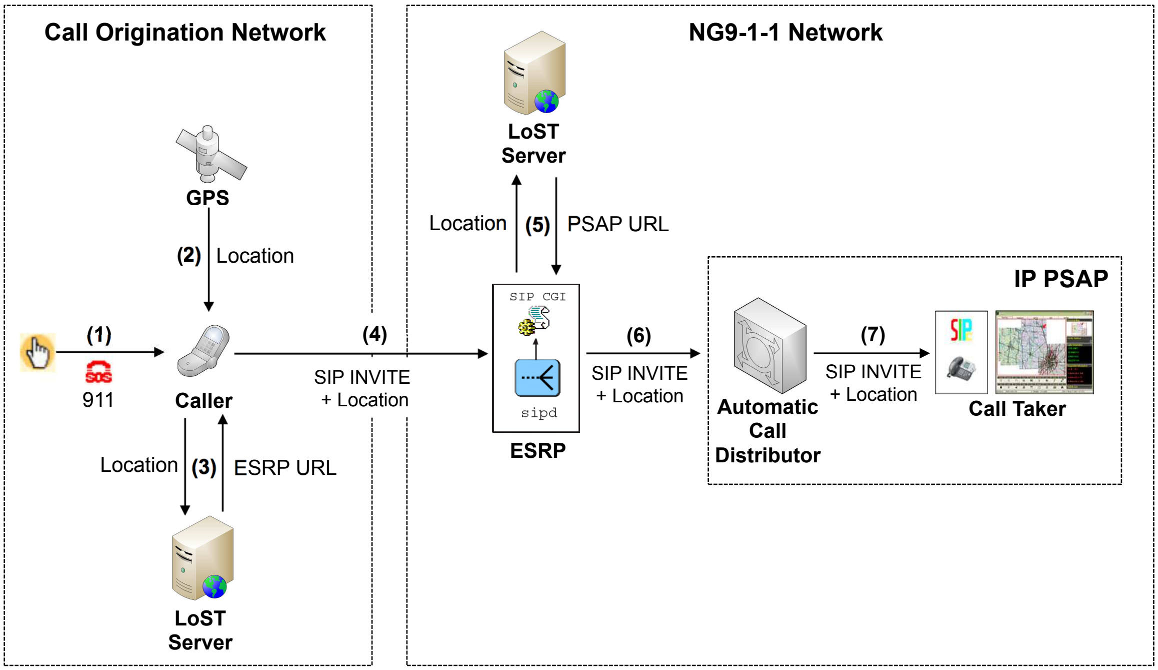


Figure 2.2: Emergency call flow in NG9-1-1.

termination function such as answering the emergency call, dispatching the responder to the incident location, and recording the call and incident information.

There are many types of call origination networks, but in the NG9-1-1 architecture, they share common responsibilities: identifying an emergency call, obtaining the location of the caller and making it available to the PSAP, and routing the call to the appropriate ESRP. For this purpose, the POC system introduced SIP border gateways [64], which are SIP outbound proxies that enable aforementioned functions for emergency calls that the end device cannot provide. Also, since the NG9-1-1 network is SIP-based, it is the call origination networks' responsibility to convert their signaling protocols to SIP. For example, a PSTN call is converted to a SIP call through a telephony gateway. A LoST server in the call origination network is a public LoST server for location-to-ESRP resolution.

The emergency call flow within the NG9-1-1 architecture can be divided into four steps: identifying emergency calls, determining location, routing to correct PSAP, and presenting call to call taker. Figure 2.2 demonstrates an example of an emergency call flow in the NG9-1-1 architecture. First, the caller's device identifies an outgoing call as an emergency call when the user dials 9-1-1 (1). The destination URI of the emergency call is set to the emergency service URN [96] so that subsequent call handling entities will be able to recognize it as an emergency call. Second, the caller's device determines its civic or geospatial location (2). The location information is then used to route the call to the correct NG9-1-1 network, and is delivered with the emergency call. Third, the emergency call is routed to the correct PSAP based on the caller's location. To this end, two routing decisions are made in the call origination network and the NG9-1-1 network. In the call origination network, the caller's device sends the location information to the public LoST server and receives the URL of ESRP that covers the caller's location (3). Once the ESRP URL is resolved, the caller's device generates an emergency SIP INVITE message and sends it to the ESRP (4). In the NG9-1-1 network, the ESRP queries the LoST server for the PSAP URL using the location information within the SIP INVITE message (5), and then forwards the message to the PSAP (6). The last step is presenting the call to the call taker in the PSAP. The automatic call distributor selects an available call taker and forwards the call to the call taker (7). The call taker's status changes to busy and remains so until the end of the call. Meanwhile, the call taker communicates with the caller to verify the location, determine the type of emergency, and dispatch help to the caller.

Each step described above can be executed in a different order to improve system performance. For example, the caller's device can determine its location and performs the LoST query before the user initiates an emergency call. This can reduce the emergency call setup time, which is defined as the duration from dialing of the last digit to ringing at the PSAP.

2.3 Software Components

2.3.1 Caller SIP Client

SIPc [121] is a software SIP user agent developed by Columbia University. With SIPc, users can make a call with multimedia support such as audio, video, and text.

In the POC system, SIPc is used as an IP user agent (IP UA) call origination software. We extended SIPc to support emergency calls. SIPc determines its location, routes emergency calls to the proper ESRP with the help of LoST server, and sends emergency SIP

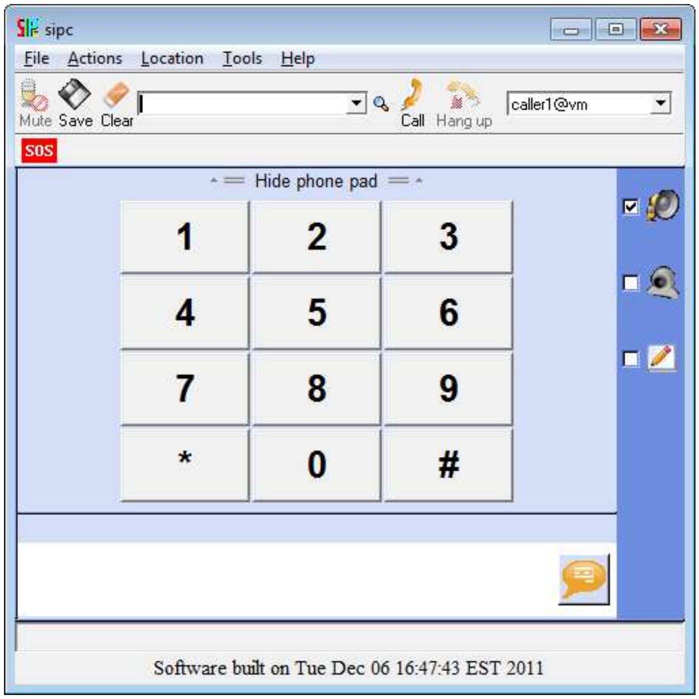


Figure 2.3: Screenshot of caller SIPc.

INVITE requests.

The location of the caller can be determined by GPS, Wi-Fi, DHCP, LLDP-MED, or manual entry. In the POC system, we used LLDP-MED as the primary location source, but also tested GPS, Wi-Fi, DHCP, and manual entry. To support routing of emergency calls, SIPc sends a LoST query to the LoST server in order to resolve the destination ESRP URL after determining its location. This location determination and LoST query are performed before the user places an emergency call.

Figure 2.3 shows the screenshot of SIPc. Users can place emergency calls by either dialing 9-1-1 or pressing the “SOS” button on SIPc. After identifying the emergency call, SIPc sends the emergency SIP INVITE request. The INVITE request contains the location information in its body as well as the SDP for media negotiation. The request URI and the SIP To header field are set with the emergency service URN, which is “urn:service:sos” [96]. This URN acts as an emergency call identifier for subsequent entities in the routing path. The SIP Route header field is set with the ESRP URL that is derived from the LoST query in the previous step.

The SIPc user interface and SIP stack were written in Tcl/Tk [27], a scripting language for quick prototyping and easy GUI development. The media processing was implemented by external programs. For example, the audio processing features such as capturing audio, encoding and decoding, transmitting RTP [97] packets, and playing audio were implemented using audio library from Columbia University. Vic [32], an open source video conferencing application, was used for video conferencing. Real-time text (T.140) [65] was implemented using the t140handler, developed at Columbia University. Those external modules were integrated with SIPc using Tcl/Tk.

2.3.2 Location-to-Service Translation Server

In the NG9-1-1 architecture, the Location-to-Service Translation (LoST) server is used for the location-based routing. The LoST server maintains PSAP service area information in its database. Given the caller's location in civic or geospatial format, it returns the contact information of the PSAP which serves that area.

The LoST request contains the location of the caller and the emergency service identifier, which is "urn:service:sos" for emergency calls. The location information in the request could be either in civic or in geodetic format. The LoST response contains the mapping information. The mapping information includes the PSAP URL, the PSAP display name, and the PSAP boundary.

In the POC system, we deployed two levels of LoST servers, namely the public and private LoST servers. The public LoST server maintained the state-level mapping information. The call origination devices used the public LoST server in order to get the ESRP URL. The private LoST server kept the county-level mapping information. It was accessed by ESRPs in the NG9-1-1 network to get the PSAP URL.

The LoST server was written in Java as a web application. It runs on the Apache Tomcat [1] web application server. The PostgreSQL server [22] with the PostGIS extension [21] was used to store the mapping database in both civic and geodetic format.

2.3.3 SIP Border Gateway

The SIP border gateway is a SIP proxy in the call origination network. It acts as an interface between the call origination device and the NG9-1-1 network. Each of the call origination networks in the POC system, such as the PSTN, IP network, and cellular and SMS networks, had their own SIP border gateway. The SIP border gateway fills the gap between what the call origination networks need to provide for emergency calls and what they can provide using the limited capability of their devices.

For example, the PSTN telephone does not have the capability to determine its own location. In this case, the PSTN SIP border gateway queries the Automatic Location Identification (ALI) database to retrieve the location of the PSTN telephone. Likewise, the cellular or SMS SIP border gateway interacts with the simulated Mobile Positioning Center (MPC) to get the location of the cellular phone. The IP UA SIP border gateway uses DHCP to determine the location of the hardware SIP phone. For all of these cases, the location information is added to the SIP INVITE request by the SIP border gateway.

The SIP border gateway was implemented using sipd and a SIP-CGI script [74]. Sipd is a SIP proxy and registrar within CINEMA [105], a SIP software suite developed at Columbia University. Emergency call-related functions were implemented in the SIP-CGI script. SIP-CGI is “Common Gateway Interface (CGI) for service creation in a SIP environment” [74]. The SIP-CGI script was written in Perl. Sipd invokes the SIP-CGI script whenever it receives a SIP request message for emergency calls.

2.3.4 SMS Server

The SMS server, shown in Figure 2.4, interacts with the VoIP GSM gateway to convert SMS messages to SIP MESSAGE requests. The VoIP GSM gateway and the SMS server maintain a TCP connection to communicate with each other. The SMS server is located between the VoIP GSM gateway and the SMS SIP border gateway. The SMS server was implemented using Java and the JAIN SIP stack [16].

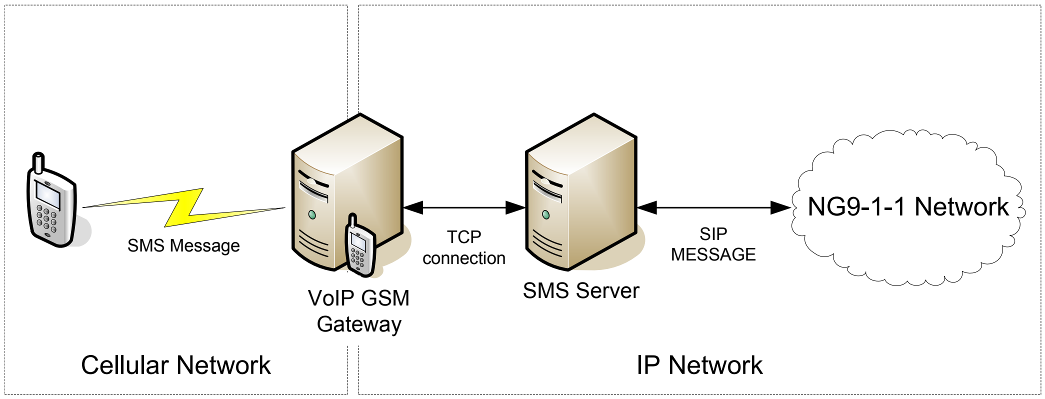


Figure 2.4: SMS server in NG9-1-1 POC system.

2.3.5 Emergency Services Routing Proxy

The Emergency Services Routing Proxy (ESRP) is a SIP proxy in the NG9-1-1 network. Its primary role is to route emergency SIP messages to the proper PSAP. The routing is based on the caller's location and/or the business rule of the NG9-1-1 network. In the POC system, the NG9-1-1 network business rule can override any routing decision that is based on the caller's location. For example, authorities may use business rules to redirect calls from a non-operational PSAP to its neighbor PSAP.

We deployed two ESRPs: ESRP West for the Western United States and ESRP East for the Eastern United States. In the real deployment, there may be more ESRPs, reflecting the business relationship between groups of PSAPs.

We used sipd and a SIP-CGI script to implement the ESRP. Like the SIP border gateway, the SIP-CGI script implemented the emergency call related functions and was written in Perl. The business rule was stored in the MySQL database [17].

2.3.6 Psapd

Psapd is the automatic call distributor within PSAPs. It distributes incoming emergency calls to the call taker depending on the call taker's status as well as other PSAP policies such as language preference of call takers. It also manages the call queue, logging, and conferencing.

We developed psapd using the CINEMA SIP library from Columbia University. The

MySQL database was used to store the call information. The call distribution logic was written in a Tcl script for easy customization.

2.3.7 Video Recorder

In emergency communications, all media streams need to be recorded so that they can be replayed later. In the POC system, the audio was recorded by the conference server. We used the Dialogic IP Media Server [6] for the conference server. However, it did not support video recording.

To support video recording, we developed and deployed a video recorder in the POC system. The video recorder can relay the video stream in RTP, transcode it to MPEG-4, and store it into local files. The recording files can later be accessed using the HTTP GET method for replay.

The video recorder is transparent: it records the video while forwarding it to the original destination. Here is how it works in the POC system. When initiating the emergency call session, psapd sends a "start recording" signal to the video recorder. Then it redirects the video stream from the caller to the video recorder by modifying the SDP sent to the caller. Originally, the conference server is supposed to receive the video stream for video conferencing. During the call, the video recorder records the incoming video from the caller while forwarding it to the conference server seamlessly. This way, the caller's video is recorded without interrupting video conference.

The video recorder ran on a Linux platform and was implemented using the GNU ccRTP RTP library [13] and FFmpeg [11], a video conversion tool.

2.3.8 PSAP Call Taker SIP Client

Call takers in the POC system used SIPc to answer calls. The call taker SIPc, shown in Figure 2.5, handles multimedia calls, displays emergency information such as the caller's location and the callback number, and allows the call taker to enter incident details. It also uses Google Maps [14] to display the caller's location on a map. Other functions include dynamic display of call scripts and standard operating procedures based on incident types, customizable one-click speed dial buttons, and links to call supporting information.

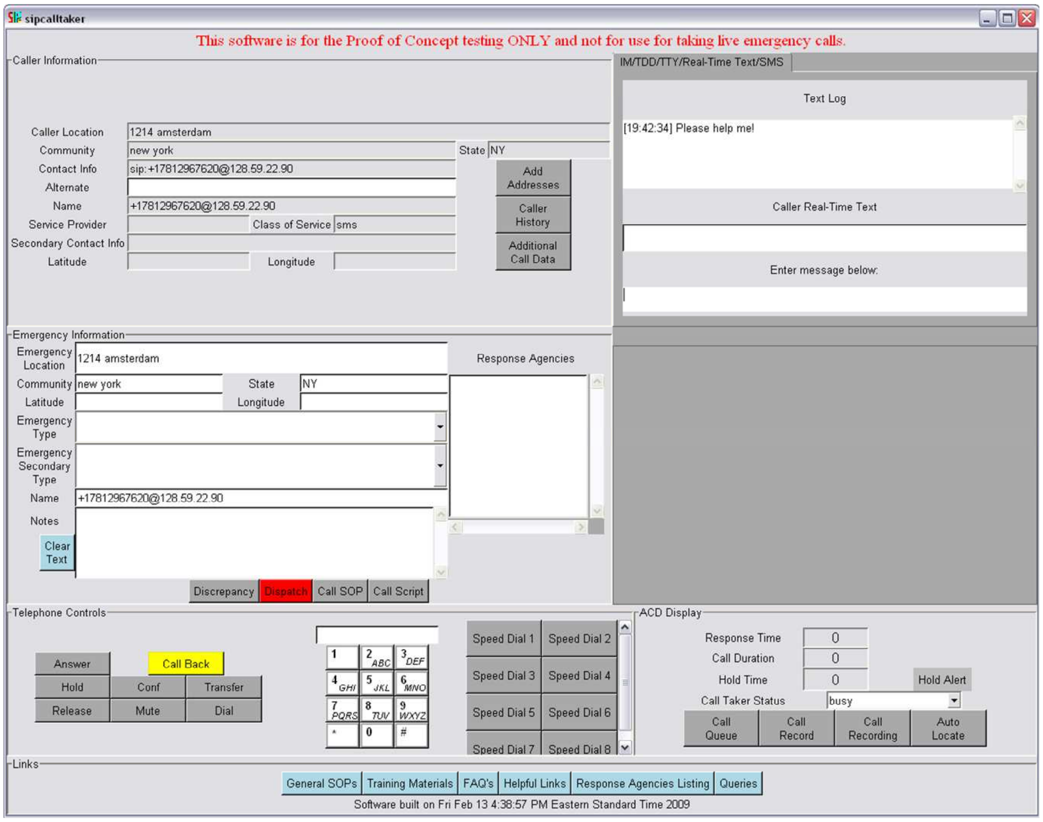


Figure 2.5: Screenshot of PSAP call taker SIPc.

The call taker SIPc shares the code base with the caller SIPc, except that the user interface implements the extra functions needed by call takers.

2.4 Features and Use Cases

2.4.1 Call Origination from Various Devices

The POC explored call origination use cases from five different communication service classes: IP user agents such as laptops and IP phones, telephones, cellular phones, SMS, and telematics devices. Even though these are five different services, emergency calls that originate from these services go through a similar call setup procedure. The procedure involves protocol conversion, location determination, and call routing. All calls are converted to SIP (for call signaling) and RTP (for media streams) if needed, and then forwarded to

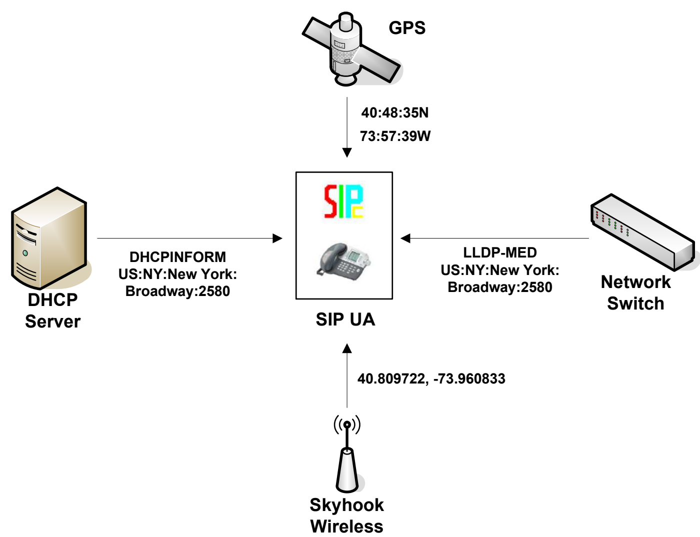


Figure 2.6: Location determination techniques used by SIPc.

a SIP border gateway. The SIP border gateway is a SIP outbound proxy with emergency features such as determining location of the calling device and querying a LoST server to route the call to an appropriate ESRP. Location determination is necessary to forward the call to the correct ESRP and also for the call takers to dispatch help to the correct location. The POC system places responsibility of determining location to the call origination network. Once the location is determined, it is formatted into a PIDF-LO object [119], an XML document that carries the user's location, and added to the SIP INVITE request as a MIME attachment.

IP User Agents

IP user agents (UAs) are SIP-based communication software or devices for end users. The POC system tested two types of IP UAs: a laptop with SIPc and a Cisco IP phone. SIPc represents fully implemented NG9-1-1 capable IP UAs while the Cisco IP phone represents SIP phones without NG9-1-1 capability. The POC system supported emergency calls from both IP UAs. Both IP UAs were SIP-based, so there was no need to convert the original protocol to SIP. Also, since both were SIP-based, the SIP border gateway doubled as a registrar and IP UAs registered to the SIP border gateway. The following describes the call flow originating from SIPc and the Cisco IP phone.

SIPc tries to determine its location every time it launches. Location is determined automatically by four different technologies as shown in Figure 2.6. First and the most accurate method is GPS. We use a GPS dongle that is attached to the caller's laptop using the USB interface. SIPc retrieves the geodetic coordinates of the current position from the GPS dongle via the NMEA 0183 protocol [73].

Second, SIPc determines its location using Wi-Fi positioning. We use the Skyhook Location SDK [25] to implement Wi-Fi positioning in SIPc. The Skyhook SDK returns the latitude and longitude of the laptop's location by comparing the Wi-Fi signals received by the laptop with known Wi-Fi "fingerprints".

Third, SIPc queries for its location to the DHCP server. The DHCP server maintains the mapping between the MAC address of a client and its static location in either civic or geodetic coordinates format. When the DHCP server receives the DHCPINFORM message including the MAC address of the client and the option codes, 123 (for geo-coordinates) and 99 (for civic address) [87,95], it returns the client's location using the DHCPACK message.

Lastly, we use an LLDP-MED [35] enabled Ethernet switch. Like the DHCP server, the switch maintains the location of the devices attached to its Ethernet ports. SIPc receives the LLDP-MED frames that are periodically advertised by the switch and extracts the location from the received frame.

Additionally, the location can be determined manually by the caller who enters the street address or geodetic coordinates in the location configuration window.

With the location, SIPc queries a LoST server, which returns an appropriate ESRP URL for that location. In case of an emergency, the ESRP URL is included in the SIP Route header field of the emergency SIP INVITE request. When the location changes, SIPc queries the LoST server again to find the appropriate ESRP URL for the updated location. This step is done before the caller places an emergency call.

SIPc recognizes an emergency call when the caller clicks on the emergency call button on its main window or when the caller dials an emergency number (e.g., 9-1-1 in the US, 1-1-2 in Europe). If the call is recognized as an emergency call, SIPc performs NG9-1-1 functions such as adding a SIP Geolocation header field to the SIP INVITE request, appending location information in PIDF-LO format to SIP INVITE request as a MIME attachment, filling the

SIP To header field and the request URI with "urn:service:sos", and sending the message to the SIP border gateway. The SIP border gateway does not have much to do except to forward it to the ESRP URL already included in the SIP Route header field.

Unlike SIPc, the Cisco IP phone does not distinguish an emergency call from a normal phone call. It simply forwards a SIP INVITE to the SIP border gateway. The SIP border gateway recognizes an emergency call by inspecting the SIP To header field. If 9-1-1 or "sos" is present in the SIP To header field, it is considered an emergency call. Once recognized as an emergency call, the SIP border gateway determines location information and queries the LoST server to find out where to forward this message.

The SIP border gateway determines the location of the Cisco IP phone by querying a simulated location database. A MySQL database keeps the mapping between the phone's SIP address and its physical location. Like landline phones, this mapping is static and maintained by the system administrator. The SIP border gateway adds a SIP Geolocation header field to the SIP INVITE request and appends location information in PIDF-LO format to SIP INVITE request as a MIME attachment.

The SIP border gateway uses the location information to query a LoST server. The LoST server returns the appropriate ESRP URL. Then the SIP border gateway forwards the INVITE request to the ESRP.

PSTN Phones

An analog telephone was connected to a Cisco telephony gateway to demonstrate how the POC system can answer PSTN calls. The Cisco telephony gateway generated SIP messages and packetized voice into RTP streams, but was not NG9-1-1 capable.

As with the Cisco IP phone, the telephone and the gateway does not distinguish between an emergency call from a non-emergency call, so the task of recognizing an emergency call is assigned to the PSTN SIP border gateway. The behavior of the SIP border gateway is identical to the one in the Cisco IP phone use case, namely, it determines location information and queries the LoST server to find out where to forward this message. The difference is in how the SIP border gateway determines the telephones location.

The PSTN SIP border gateway determines the location of the telephone by querying a simulated ALI database. The ALI database, which also runs on a MySQL database, maintains (telephone number, location information) tuples as the key, value pair. The SIP border gateway queries this database using the caller's telephone number and receives the caller's location. Then it appends the location information in PIDF-LO format to SIP INVITE request as a MIME attachment and adds a SIP Geolocation header field to the SIP INVITE request.

Cellular Phones

To support cellular phone calls, a PSTN VoIP gateway was introduced into the POC system. A telephone number was assigned to the PSTN VoIP gateway so that cellular phone callers can dial this number as the emergency number instead of dialing 9-1-1 on their cellular phones. When the call arrived at the PSTN VoIP gateway, it generated SIP messages and packetized voice into RTP streams. However, just like the Cisco telephony gateway, it was not NG9-1-1 capable.

The PSTN VoIP gateway simply forwards a normal SIP INVITE request to the cellular SIP border gateway. It is the cellular SIP border gateway that recognizes an emergency call, determines and includes location, and routes the call to the correct ESRP.

The cellular SIP border gateway determines the location of the cellular phone by querying a simulated Mobile Positioning Center (MPC) database. The database is identical to the ALI database: it uses the (cellular phone number, location information) tuple as the key, value pair. After the cellular SIP border gateway gets location information from this database, it appends location information in PIDF-LO format to SIP INVITE request as a MIME attachment and adds a SIP Geolocation header field to the SIP INVITE request. Then, as with other SIP border gateways, the cellular SIP border gateway queries the LoST server to resolve location to an ESRP URL.

Short Message Service

The POC system explored the idea of using SMS as an emergency communication tool. To support this, a VoIP GSM gateway was paired with the SMS server. The VoIP GSM gateway was similar to the PSTN VoIP gateway. It had a 10-digit telephone number assigned to it so that an SMS message can be sent to the gateway. The telephone number was used in lieu of 9-1-1. Once the SMS message reached the VoIP GSM gateway, the gateway converted it into a format that is readable by the SMS server. The SMS server's role was to generate a SIP MESSAGE request from the SMS content. The VoIP GSM gateway and the SMS server sent messages to each other over a persistent TCP connection. Both the VoIP GSM gateway and the SMS server were not NG9-1-1 capable.

The SMS server simply forwards a SIP MESSAGE request to the SMS SIP border gateway. The behavior of the SMS SIP border gateway to support NG9-1-1 is identical to the cellular SIP border gateway, except that the forwarded message is a SIP MESSAGE request instead of a SIP INVITE request.

Telematics

A vehicle telematics system embedded in the car detect accidents, automatically dial an operator, and transmit relevant data [8]. The important feature in telematics is the delivery of vehicle crash data along with the call.

The POC system emulated this feature with help from OnStar [19]. OnStar provides commercial telematics services in the United States and Canada. Drivers can communicate with OnStar representatives for emergency services, navigation assistance, or vehicle diagnostics. A cellular phone was assigned as a telematics call device. All calls from this cellular phone were converted to SIP by OnStar's gateway, and then forwarded to the telematics SIP border gateway.

OnStar used a simulated MPC database to get the caller's location, similar to the cellular phone use case. The location was then appended to the SIP INVITE request. Crash data was also included in the SIP INVITE request as an HTTP URL in the SIP Call-Info header field. Later, the call taker was able to pull the vehicle crash data using the HTTP GET method, as shown in Figure 2.7.


```
<?xml version="1.0" encoding="UTF-8"?>
<vei:VehicularEmergencyIncident xmlns:vei="http://www.comcare.org/schemas/vei">
  <DataSource>
    <Type>Telematics Service Provider</Type>
```

```
    <IncidentID>1</IncidentID>
    <IncidentOriginator>true</IncidentOriginator>
    <CallBackNum>1112223333</CallBackNum>
    <ProviderName>ONSTAR</ProviderName>
  </DataSource>
  <IncidentData>
    <IncidentTime>15:51:43.000-04:00</IncidentTime>
    <IncidentDate>2008-05-02-04:00</IncidentDate>
    <LocationTime>15:51:43.000-04:00</LocationTime>
    <ReceivedTime>15:51:43.000-04:00</ReceivedTime>
    <Location>
      <Latitude>38.93949889999999669498720322735607624053955078125</Latitude>
      <Longitude>-78.3533936000000039712176658213138580322265625</Longitude>
      <LocationDescription>Airport</LocationDescription>
    </Location>
  </IncidentData>
  <AutomatedIncidentData>
    <CrashData>
      <IgnitionState>off</IgnitionState>
      <Heading>0</Heading>
      <Orientation>Normal</Orientation>
      <Impact/>
    </CrashData>
    <SeatData/>
    <VehicleData>
      <BodyType>Passenger car</BodyType>
      <Color>yellow</Color>
      <LicensePlateNum>XX911XX</LicensePlateNum>
      <Make>Chevrolet</Make>
      <Manufacturer>GM</Manufacturer>
      <Model>Camaro</Model>
      <Owner/>
      <PowerSource>main battery</PowerSource>
      <VIN>00000000000000000</VIN>
```

```
      <Weight>3288</Weight>
      <Year>2009</Year>
    </VehicleData>
  </AutomatedIncidentData>
</vei:VehicularEmergencyIncident>
```


Figure 2.7: An example of vehicle crash data from OnStar.

Since OnStar provided the caller's location, the telematics SIP border gateway simply routed the call to an ESRP after a LoST query.

2.4.2 Conferencing and Call Transfer

During an emergency call, call takers may need to add more participants or transfer the call. For example, the call taker may need assistance from the supervisor or the call may have to be transferred to the local police station. Both functions are implemented in psapd and the call taker software. The call taker software provides the user interface to initiate these functions while psapd provides the logic that makes these functions work.

In the POC system, all calls are conference calls. This means that all incoming calls from callers are automatically put into conference rooms where call takers are ready to receive calls. This way, it becomes easier to add new call takers, administrators, or other emergency personnel to the call as needed.

Figure 2.8 shows how an emergency call is handled by psapd. When a new call comes in (1), psapd creates a conference session first (2), selects one of the available call takers (3), then invites the call taker to the conference (4). As a third party call controller, psapd facilitates the media negotiations between the conference server and the call taker software (5–9). After the negotiations are successful (10), the caller is added to the conference call (11). Once again, psapd facilitates media negotiations between the conference server and the caller software (11–15). If successful, the caller and the call taker can talk to each other in the same conference room (17).

It is simple to add another party into this conference room. The call taker enters the third party's SIP URL and clicks on the conference button in the call taker software.

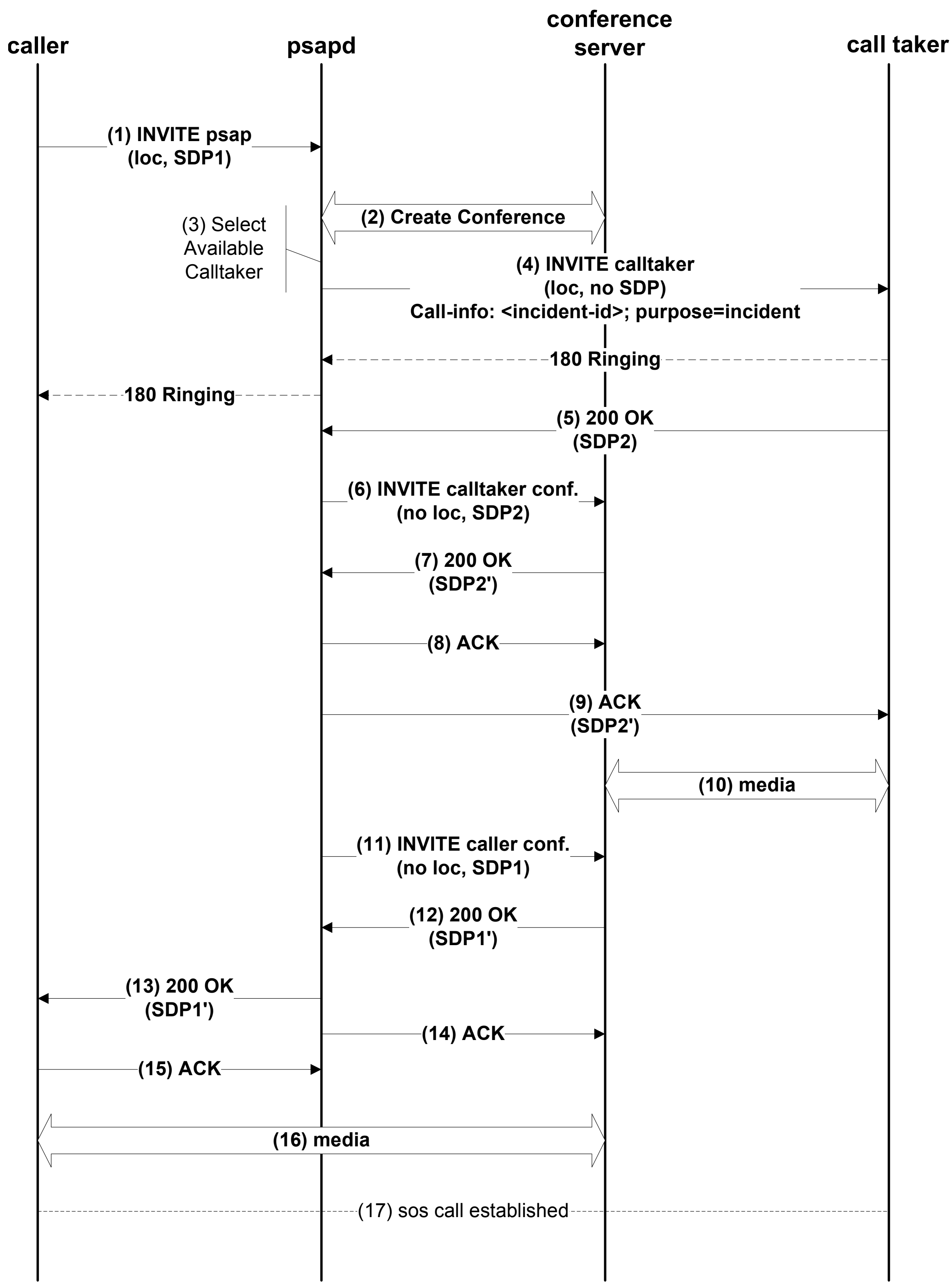


Figure 2.8: Basic emergency call flow.

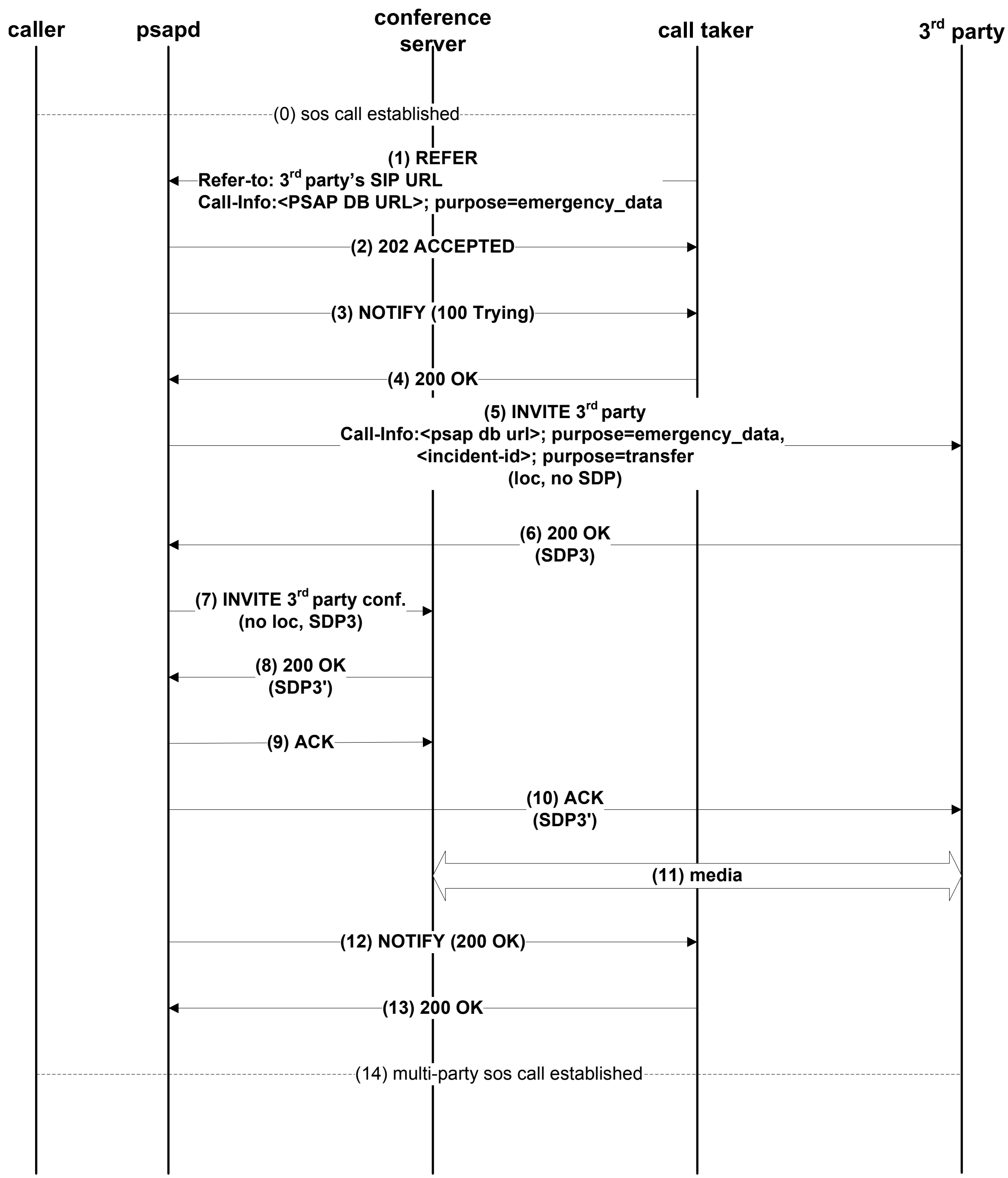


Figure 2.9: Call transfer flow.

Figure 2.9 shows how the third party is added into the existing 9-1-1 call. The call taker software constructs a SIP REFER request [107] with the third party's SIP URL in the SIP Refer-To header field and sends it to psapd (1). Using information in the REFER request, psapd constructs and sends a SIP INVITE request to the third party (5). This begins media negotiations between the conference server and the third party's software (6–11). The third party is in the conference as soon as the negotiations are successful (11).

The call transfer is implemented by adding a third party into the conference room, as described above, and then the transfer-requesting call taker voluntarily withdrawing from the conference room. This is due to the requirement that the transfer-requesting party has to make sure that the transfer is successful before dropping out of the call.

In both conferencing and call transfer, invited participants receive the caller's location. The psapd includes caller's location in the SIP INVITE request sent to invited participants. The SIP INVITE request also contains the incident ID in the SIP Call-Info header field. The incident ID is a unique string assigned to each emergency call by psapd and acts as the primary key in the PSAP database. Additionally, in call transfer, psapd includes the PSAP database URL so that the invited party can pull relevant information such as incident information and comments from the original call taker. The URL is conveyed in the SIP Call-Info header field.

2.4.3 Call Queue and Interactive Voice Response

PSAP can be flooded by incoming emergency calls when a mass disaster occurs or many people try to report the same incident at the same time. To handle overflow calls, the IP PSAP in the POC system maintains a call queue and puts incoming calls into the call queue when there is no available call taker in the PSAP. The calls in the call queue will be served in the First In First Out (FIFO) manner whenever a call taker becomes available in the PSAP later.

While the caller is waiting in the call queue, the Interactive Voice Response (IVR) component can play prerecorded announcements, collect preliminary information about the incident, or instruct the caller to stay on the line or hang up the call, depending on the situation. For example, if there is a nearby incident from the caller's location that is already reported to the PSAP, the IVR application can describe the incident location and details, and then ask whether the caller is reporting the same incident. This feature can filter out the duplicated reports in the call queue so that the load of the call takers can be alleviated during PSAP overload.

Figure 2.10 describes the call flow between the POC components when the caller is placed in the call queue, and later answered by the call taker. When psapd receives an incoming call from the caller (1), psapd creates a conference (2), and tries to select an available call taker in the PSAP as in the normal case. If there is no available call taker, psapd adds the call to the call queue (3) and proceeds the media negotiations between the caller and the conference server (4–8). Then, psapd creates a SIP INVITE request and sends it to the media server (10). This request includes the VoiceXML URL for the IVR application. The media server acts as a VoiceXML browser which runs the IVR application after fetching the VoiceXML document using this URL. After the media negotiations between the media server and the conference server are completed (11–16), the caller and the IVR application are in the same conference room so the caller can hear the announcement played by the IVR application (17).

The following steps describe how the caller is dequeued and served by a call taker. When a call taker becomes available (18), the call taker software sends a SIP PUBLISH request to psapd with the status parameter as available (19). This message triggers psapd to dequeue the caller (21) and to invite the call taker to the conference room where the caller and the IVR application are currently in (22-26). The IVR session in the conference is then terminated when psapd sends SIP BYE messages to the conference server and the media server (27–30). The media negotiations between the call taker and the conference server are completed after the SIP ACK message (32), and the caller and the call taker starts the conversation in the same conference room (34).

Figure 2.11 shows the overall architecture of the IVR system. We use a Dialogic IP Media Server as the media server to implement the IVR application. Combined with the media server, an IBM WebSphere Voice Server provides Text to Speech (TTS) and Automated Speech Recognition (ASR) functions using the Media Resource Control Protocol (MRCP) [102] connection. MRCP is a standard protocol for controlling TTS and ASR

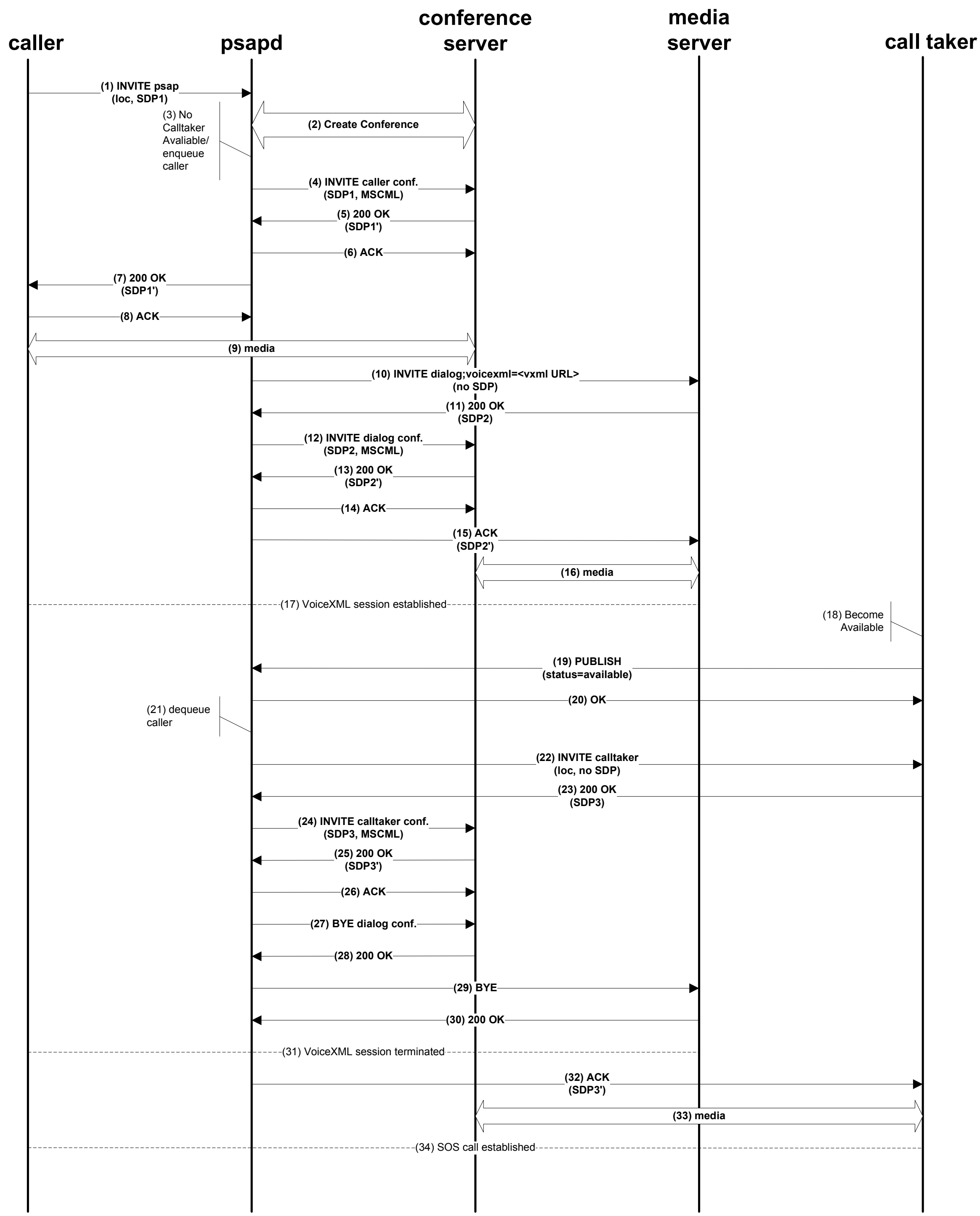


Figure 2.10: Call flow when overflown calls are enqueued and later dequeued.

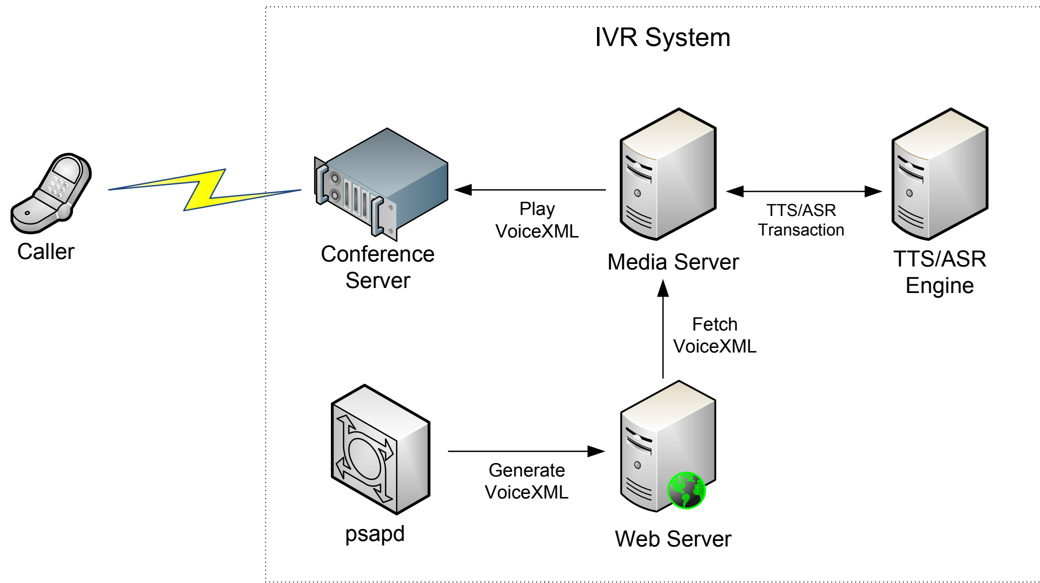


Figure 2.11: IVR system architecture.

transactions in IVR systems. For TTS function, for example, the media server sends an MRCP request carrying the text to be synthesized, and the TTS engine returns a speech stream as a response. MRCP control messages are encapsulated in Real Time Streaming Protocol (RTSP) [99]. MRCP uses Real Time Protocol (RTP) [97] for media delivery.

The VoiceXML document for the IVR application is automatically generated by psapd based on the caller's location and timestamp. If there is no incident report near the caller's location within a preconfigured time period, the VoiceXML document shown in Figure 2.12 is generated. The VoiceXML document plays the announcement that all lines are busy in the PSAP while the caller is waiting in the call queue. If the caller's location and timestamp are similar to an existing incident in the PSAP database, a more complex VoiceXML document shown in Figure 2.13 is generated and played as the IVR application. The VoiceXML document asks questions using the TTS function and instructs the caller differently based on the caller's verbal answer processed by ASR or the Dial-Tone Multi-Frequency (DTMF) input.

```
<?xml version="2.0"?>
<vxml version="2.0" xmlns="http://www.w3.org/2001/vxml">
  <form id="wait">
```

```
    <field name='bogus'>
      <grammar src="builtin:dtmf/boolean"/>
      <prompt bargein="false" timeout="1s">
        <audio src="prompts/generic/en_US/beep.wav"/>
        <audio src="prompts/generic/en_US/circuit_busy.ulaw"/>
        <audio src="prompts/wait.mu"/>
      </prompt>
      <goto next="#wait"/>
    </field>
  </form>
</vxml>
```

Figure 2.12: VoiceXML script for playing simple announcement.

```
<?xml version="2.0"?>
<vxml version="2.0" xmlns="http://www.w3.org/2001/vxml">
  <form id="welcome">
    <block bargein="false">
      <prompt> Are you calling about 'FIRE' on 'Amsterdam Ave'? </prompt>
    </block>
    <field name="answer">
      <option dtmf="1"> yes </option>
      <option dtmf="2"> no </option>
      <filled>
        <if cond="answer=='yes'">
          <goto next="#yesman"/>
        <elseif cond="answer=='no'"/>
          <goto next="#lines_are_busy"/>
        </if>
      </filled>
    </field>
  </form>
  <form id="yesman">
    <block bargein="false">
```

```
      Thank you. We are currently addressing the situation.
      Would you like to stay on the line?
    </block>
    <field name="answer2">
      <option dtmf="1"> yes </option>
      <option dtmf="2"> no </option>
      <filled>
        <if cond="answer2=='yes'">
          <goto next="#wait_in_queue"/>
        <elseif cond="answer2=='no'"/>
          <goto next="#goodbye"/>
        </if>
      </filled>
    </field>
  </form>
  <form id="lines_are_busy">
    <block bargein="false">
      All lines are currently busy. Please wait.
      <goto next="#wait_in_queue"/>
    </block>
  </form>
  <form id="wait_in_queue">
    <field name='bogus'>
      <grammar src="builtin:dtmf/boolean"/>
      <prompt bargein="false" timeout="1s">
        <audio src="prompts/generic/en_US/circuit_busy.ulaw"/>
        <audio src="prompts/wait.mu"/>
      </prompt>
      <goto next="#wait_in_queue"/>
    </field>
  </form>
  <form id="goodbye">
    <block bargein="false">
      Goodbye!
```

```
      <exit/>
    </block>
  </form>
</vxml>
```

Figure 2.13: VoiceXML script for handling multiple reports for same incident.

2.4.4 Call Termination

Figure 2.14 shows three different cases of how psapd handles incoming SIP BYE requests.

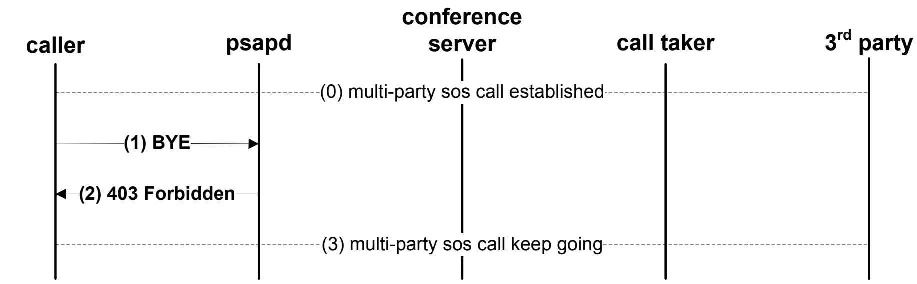
First, the POC system systematically enforces the requirement that callers should not hang up until call takers instruct them to do so in emergency calls. If the caller sends the SIP BYE request first as shown in Figure 2.14(a), psapd replies with the "403 Forbidden" response. We modify the caller SIPc not to terminate the ongoing session until it receives the "200 OK" response after sending the SIP BYE request.

Second, a third party participant can hang up anytime while not terminating the ongoing emergency call. Psapd removes only the call leg of the participant who is leaving the conference, as shown in Figure 2.14(b).

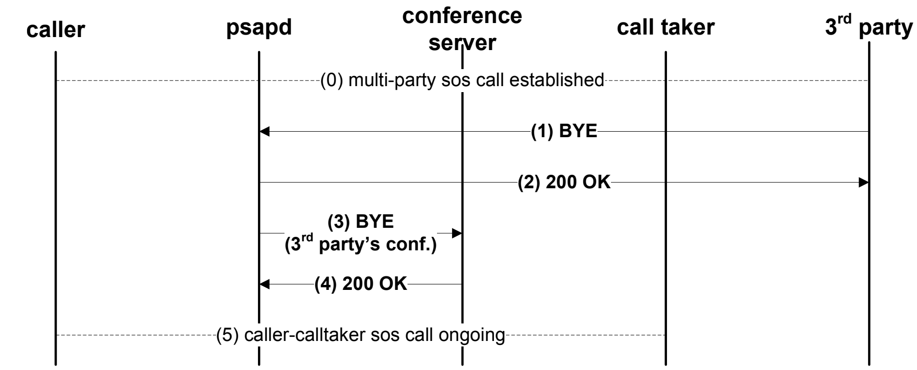
Lastly, the *primary call taker* can terminate the emergency call as shown in Figure 2.14(c). The primary call taker is the last and the only agent in the call. Initially, the call taker who picked up the call is the primary call taker, and if the call is transferred, the transferee becomes the new primary call taker. Psapd sends the SIP BYE request to the caller and removes all the call legs in the conference.

2.4.5 Callback

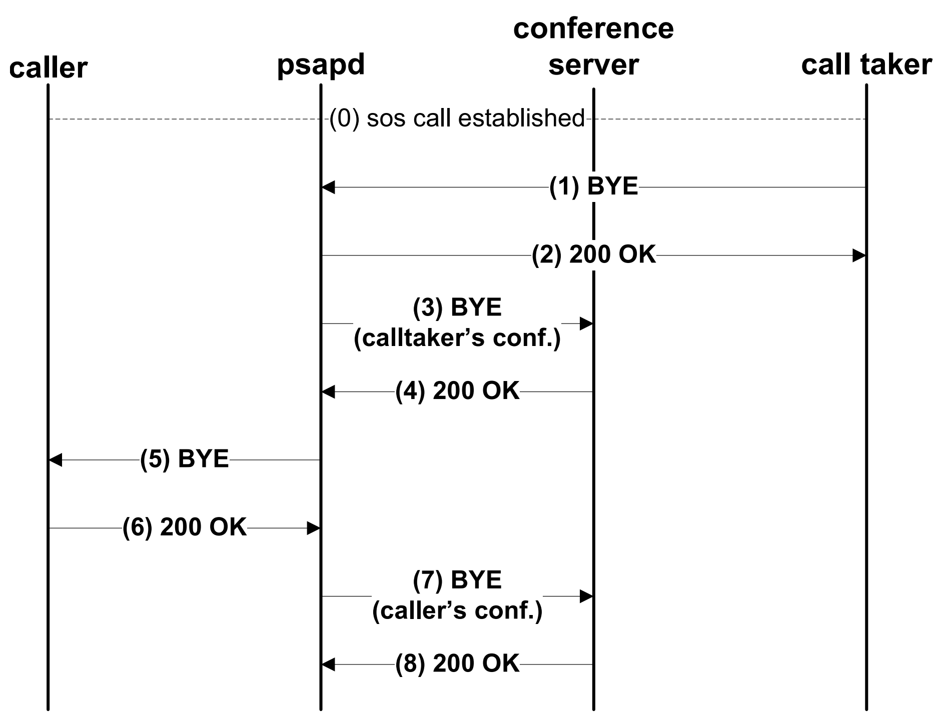
Call takers may need to call back the emergency caller after calls are disconnected or to follow up on continuing incidents. The POC system implements a simple callback feature. The callback number is extracted from the SIP From header field in the caller's SIP INVITE request. The call taker is able to call back with one click on the callback button. To the NG9-1-1 network, which is an all-SIP network from ESRP to individual PSAPs, the callback is simply a SIP INVITE request from the call taker to the caller.



(a) Caller tries to hang up first.



(b) Third party hangs up.



(c) Call taker terminates the call.

Figure 2.14: Call termination flows.

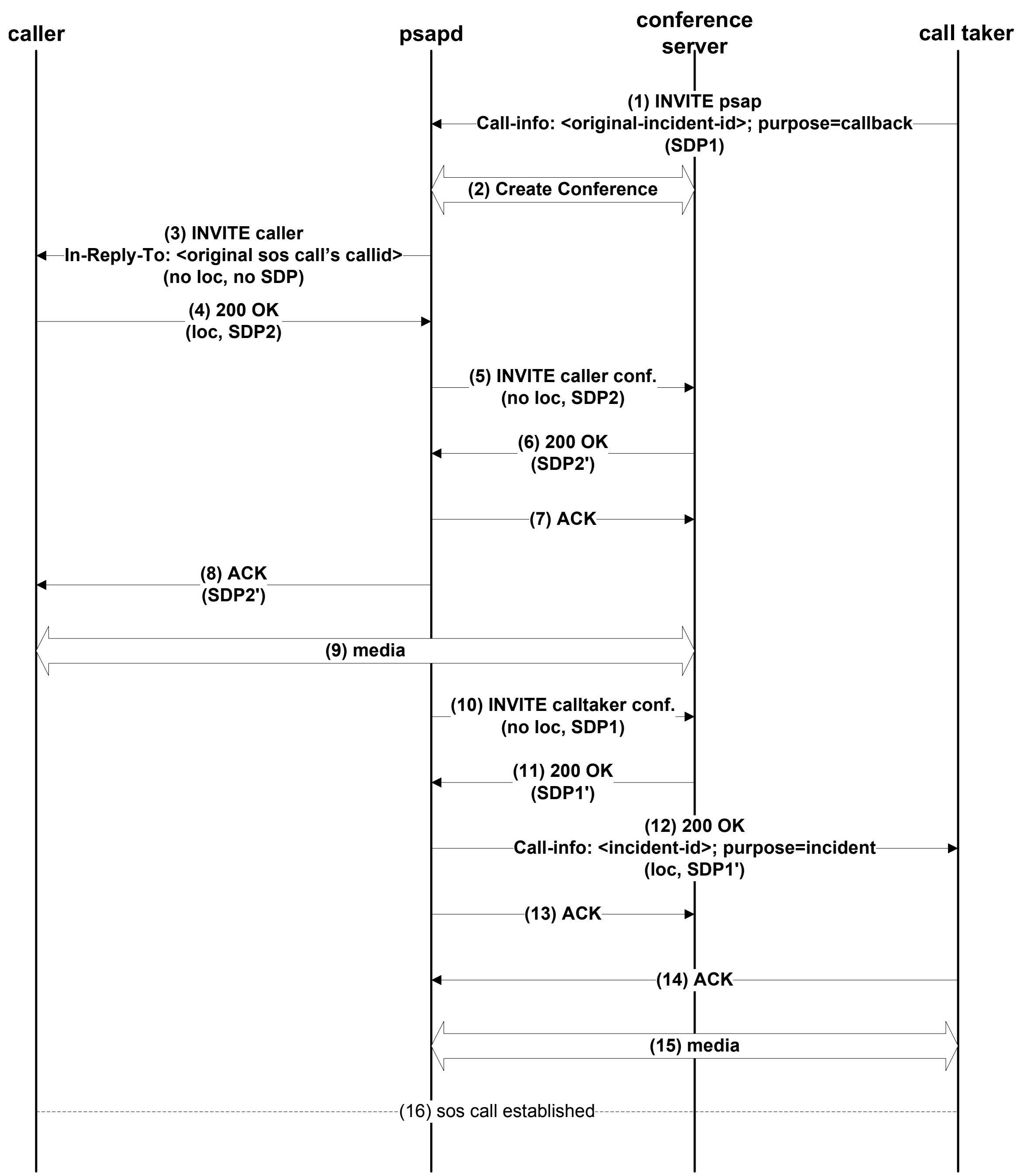


Figure 2.15: Callback flow.

Figure 2.15 shows how the callback is handled by psapd. When the call taker pushes the callback button, the call taker software sends a SIP INVITE request to psapd (1). The request contains the incident ID of the original emergency call in the SIP Call-Info header field. Using the incident ID, psapd retrieves the caller's SIP URL from its database and also creates a relationship between the original emergency call and the callback call for the logging purpose. Then, psapd creates a conference room in the conference server (2) and invites the caller to the conference room (3). After all media negotiations with the conference server are completed (5–14), both the caller and the call taker can talk to each other in the same conference room (16).

When this SIP INVITE request comes to the call origination network, each network handles the request a little bit differently. In an IP UA network, the SIP INVITE request is forwarded to the caller without any modification. In a PSTN or cellular network, the SIP INVITE request is converted by gateways. The same happens with telematics networks as there is no data flowing from the call taker to the caller. Only the voice is transmitted. In the SMS network, the SMS server converts the SIP MESSAGE request to a format readable by the VoIP GSM gateway, which in turn converts it into an SMS message. The SMS message is then sent to the caller's SMS device.

2.4.6 Logging and Recording

The ESRP logs incoming SIP messages and outgoing SIP messages so that administrators can see routing decisions for each emergency call. The logs of psapd include call logs, call distributions, and status of call queues. Call logs contain information such as timestamp, caller ID, caller location, call taker who received this call, call duration, join and leave time of various parties, and call status. The call taker software also logs information such as the call taker's status, call hold time, incident details, and the call taker's comments on incidents. Psapd and call takers share a database to log such details. The ESRP has its own log repository.

Call recording is divided into three parts based on the type of media used. Voice is recorded and stored in the conference server. This is a natural choice because every call participant's voice goes through the conference server. Text is recorded by the call taker software and stored in a database after call termination. Video is recorded by the video recorder which intercepts video frames going to the conference server and records the frames while relaying it to the conference server.

The logs and the recordings are available to the PSAP administrator through the PSAP web page shown in Figure 2.16.

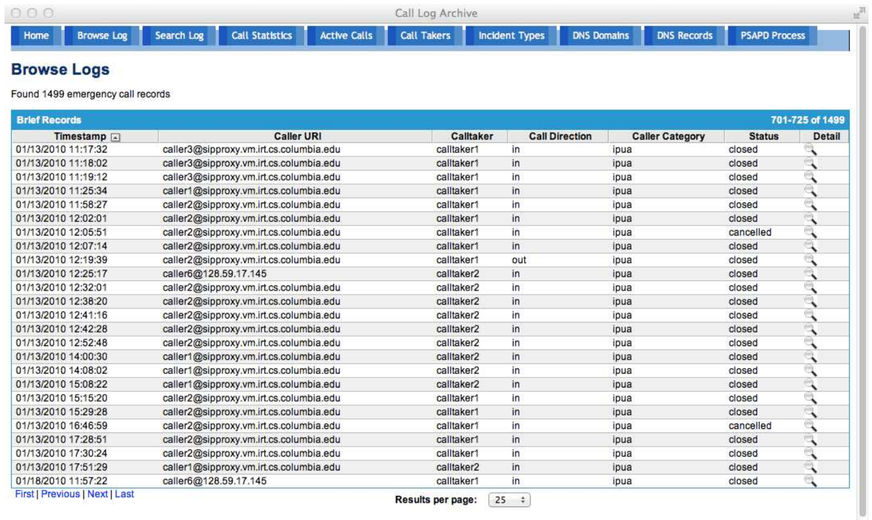
2.5 System Deployment

The components of the NG9-1-1 POC system were deployed at three laboratories and five live PSAPs in the United States, and tested by professional PSAP call takers. Figure 2.17 shows the locations of the deployment points.

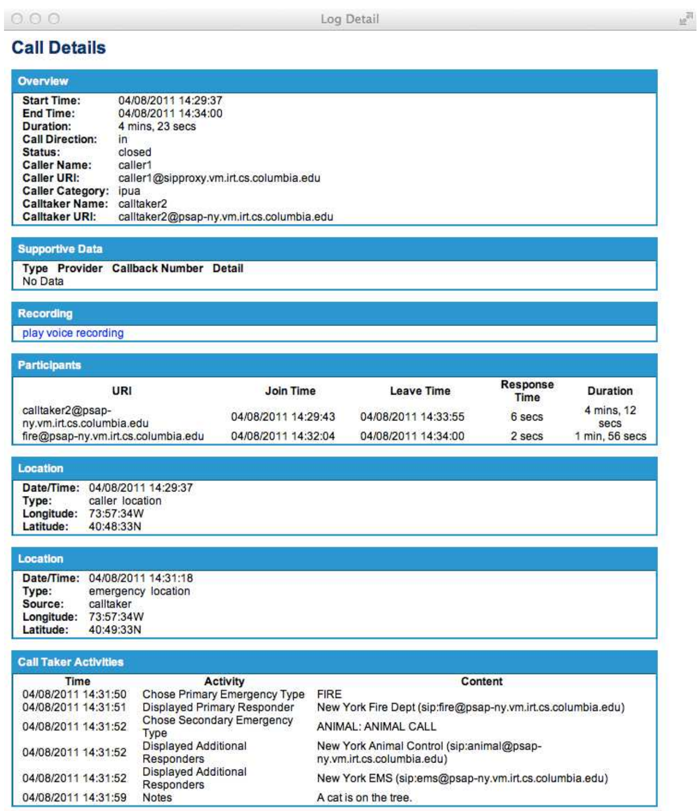
The three laboratories were Booz Allen Hamilton Center for Network & Systems Innovation (CNSI) in Herndon, Virginia; Texas A&M Internet2 Laboratory in College Station, Texas; and Columbia University Internet Real Time (IRT) Laboratory in New York City, New York. The Booz Allen Hamilton lab housed equipments that make up the call origination networks, except for telematics equipments, which were in the OnStar lab. Most backbone servers within the NG9-1-1 Network, such as the ESRPs and LoST servers, were located in the IRT Lab at Columbia University. The Texas A&M University lab was the main network connectivity point for the POC system providing routers and telephony gateways.

The IP-based PSAP systems for the call termination function were deployed at the five PSAPs participating in the POC. The five PSAPs were located in Rochester, New York; King County, Washington; St. Paul, Minnesota; Helena, Montana; and Fort Wayne, Indiana.

Booz Allen Hamilton developed the test procedures and use cases based on the NENA NG9-1-1 requirements document [36], and proceeded the testing at each PSAP with trained call takers. The detailed test procedures and results are available in NG9-1-1 Proof of Concept Testing Report [111] published by the U.S. Department of Transportation.



(a) List of call logs.



(b) Detail of a call log.

Figure 2.16: PSAP web administration pages showing call logs and recordings.

Figure 2.17: NG9-1-1 POC test deployment.

2.6 Conclusion

During the POC trials, we implemented the NG9-1-1 system and deployed it in three labs and five real PSAPs. We were able to make emergency calls from devices such as telephone, software SIP client, Cisco IP phone, and cellular phone. SMS was also tested as a potential emergency communication tool. Emergency calls were routed to the proper PSAP based on caller's location and policy in the NG9-1-1 network. It was also shown that data can be delivered along with the call. Examples of such data included the caller's location and vehicle crash data.

The POC trials also opened new areas that need more work. First, data transfer mechanisms between PSAPs should be standardized. The ECRIT working group in the IETF is standardizing the architecture, protocols, and mechanisms for routing emergency calls. However, data transfer standard is currently not being addressed. Second, the general NG9-1-1 architecture should be scrutinized for possible security issues. Lastly, in order to replace the current emergency infrastructure, the NG9-1-1 system must be shown to have comparable reliability. Further research is needed to evaluate and improve the reliability of the IP-based NG9-1-1 architecture.

Chapter 3

Integrating Text Communications into the NG9-1-1 Prototype System

3.1 Introduction

Text communication is increasingly popular, with an estimated 35 million Instant Messaging (IM) users[1] in the United States alone in 2012 [43]. In 2012, US Short Message Service (SMS) users sent and received 2.19 trillion SMS messages [52]. As such, text communication has become a major communication method.

As a consequence, text communication is currently being used for a variety of limited cases in emergency situations. In Boston and New York, SMS messages are used to report anonymous crime tips [3, 18]. It is also being used for emergency alerts on university campuses.

However, text communication should become an alternative method available to the general public to request for help in emergency situations, just like a 9-1-1 voice call. If text communication were to be used for emergency communications, it would have the following

[1] This only includes PC-based IM users. Facebook messenger and other mobile messengers are not included.

advantages. It can be used in situations where voice calls are not possible. For example, during the campus shooting incident in Virginia Tech, students would have been able to send their location using text messages while not making noise [10]. Another example of such situations was during Hurricane Katrina, when voice calls were not possible because of overload, people used SMS messages to successfully communicate with their friends [104]. The deaf and hard-of-hearing can also benefit from emergency text communications.

In the United States, wireless carriers already provide text-to-911 service [34] in limited areas. For example, in Durham, North Carolina, Verizon Wireless customers can send emergency SMS messages to the Durham Emergency Communication Center [7]. However, current text-to-911 service is an interim solution prior to the nation-wide deployment of NG9-1-1, and is recommended to be used only in a situation where making a voice call is impossible. In many areas, the text-to-9-1-1 service is not available. Moreover, the location of the sender is not automatically delivered to the call taker.

Our goal is to integrate text communications in the NG9-1-1 system. Integrating multimedia capable networks such as IM and SMS networks is one of the core concepts of NG9-1-1. NG9-1-1 is an IP/SIP-based system that will replace the current PSTN-based emergency infrastructure. We explained the NG9-1-1 system in more detail in Chapter 2.

Our key contribution is the integration of IM and SMS networks into the NG9-1-1 architecture. We introduce a model of integration for IM and SMS networks, and focus on a few technical challenges in the integration of the text communication system into the NG9-1-1 architecture. In the case of IM network, one problem is that most popular IM protocols are proprietary and therefore incompatible with the NG9-1-1 system. Thus, we propose a SIP-based model of integration. Another problem is consistent delivery of multiple messages within a session to the same call taker. We solve this problem by implementing state-keeping mechanisms in three different components within the NG9-1-1 architecture. In the SMS network, there are three problems: location conveyance, SIP conversion, and consistent delivery of multiple messages within a session to the same call taker. We solve the first problem by sending the location information and the user's message in one unit. We introduce an SMS gateway to solve the second problem. To solve the consistent message delivery problem, we implement state-keeping mechanisms similar to the one used in the

IM network.

This chapter is organized as follows. Section 3.2 describes how we can integrate IM networks into our NG9-1-1 prototype system. Section 3.3 illustrates how SMS messages can be delivered to IP-based PSAPs. Section 3.4 describes implementation details. Lastly, we conclude and discuss future work in Section 3.5.

3.2 Integration of Instant Messaging (IM)

As mentioned in the previous section, it is the responsibility of the IM origination network to convert its protocols to SIP when sending an emergency message. Although at least 85% of the IM market is dominated by AOL, Yahoo!, and Microsoft [12], which use proprietary protocols, our work does not focus on converting these protocols to SIP. Instead, we focus on how these service providers can use SIP MESSAGE requests [45] to integrate their systems with the NG9-1-1 system.

There is a reason we do not focus on the conversion problem. Commercial IM network providers already use standard IM protocols such as SIP or XMPP [94] in their systems. Yahoo! and Microsoft use SIP to allow their subscribers to talk across networks while AOL and Google use XMPP for the same purpose.[2] Therefore, if we base our approach on using SIP MESSAGE method for integration, it may be easily adapted by existing commercial IM providers.

Within SIP, there are two different ways to send IM messages. One method is to use the Media Session Relay Protocol (MSRP) [44]. MSRP is an example of a session-mode IM communication protocol that can be used with SIP call establishment, session negotiation, and teardown mechanisms. In MSRP, a TCP connection is established between the participants once a session is initiated. Since sessions are already designed into the protocol, MSRP can be used by IM systems without having to keep track of the session by other means. However, MSRP is not widely used today.

[2] The interoperability between Yahoo! and Microsoft was discontinued from December 2012 as Microsoft moved their IM users to Skype. At the time of this writing, Google Talk still interoperates with AIM, but Google Hangouts, Google's new messaging service, is not open to other IM services.

The other method, which we focus on, is a SIP MESSAGE method, which is an example of page-mode IM communication. In page-mode, each message is independent from each other, and there is no concept of session or flow between the messages [45]. However, the fact that each SIP MESSAGE request stands alone poses a problem in the NG9-1-1 system. The problem and our solutions are described in the following section.

3.2.1 Dealing with Multiple Messages within a Single Session

Once an IM conversation starts between a user and a call taker, the user expects to communicate with the same call taker until the conversation ends. However, this is not guaranteed because of the way SIP MESSAGE method works in page-mode.

The reason is that each SIP MESSAGE request stands alone, without initiating a SIP dialog. Without a SIP dialog, every SIP MESSAGE request traverses SIP routing entities such as proxies and back-to-back user agents instead of being sent directly to the particular call taker. Also, all routing entities that process the SIP MESSAGE request must make a new routing decision for the request every time, independent of previous decisions.

Before the SIP MESSAGE request reaches a PSAP, these routing decisions are based primarily on location. Therefore, if the user's location changes within a conversation, the routing decision may change, and the SIP MESSAGE request can be delivered to another PSAP. After the SIP MESSAGE request reaches the PSAP, the routing is based on local policy such as the availability of call takers. Therefore, the SIP MESSAGE request that belongs to an ongoing conversation may be delivered to a different call taker.

To solve this problem, we introduce three state-keeping mechanisms. The first mechanism resides within the user's SIP client. When the user requests a new emergency IM conversation, the SIP client opens a window in which the user can type messages. Once that window is open, the SIP client sends all messages to the same ESRP regardless of changes in location. This can be thought of as an "UI-based" session. That is, there is no session in the underlying network, but the user still thinks there is a session with the same call taker while the window is open during the emergency communication.

The second mechanism is implemented in the ESRP. The ESRP has a soft-state mechanism of keeping track of the session. In a soft-state mechanism, there is no explicit start or end of a session. When the first SIP MESSAGE request from a user reaches an ESRP, the ESRP records this as an implicit start of a session. Specifically, the user's address and the destination PSAP's address are recorded. The ESRP then starts a timer to keep track of this session. During the session, any SIP MESSAGE request from this user would be routed to the same PSAP regardless of changes in the location of the user. Also, the SIP MESSAGE request resets the timer. When the timer expires, the ESRP deletes the session. Any SIP MESSAGE request coming from the same user after the timer expires may be routed to a different PSAP since the session information is no longer available.

The third mechanism is found within the automatic call distributor and the call taker software. The automatic call distributor uses a hard-state mechanism which means there is an explicit start and end of session. The first SIP MESSAGE request received from a particular user is an explicit indication of the start of a new IM session. The automatic call distributor assigns an available call taker to handle this session and forwards all messages from the user to the same call taker during the session.

When the call taker considers the IM conversation to be over, he or she releases the session by clicking a button on the call taker software. At this point, the call taker software automatically sends a "conversation is over" message to the user as a notice. Simultaneously, the call taker software sends a SIP PUBLISH message [80] to the automatic call distributor as an explicit end-of-session signal. The SIP PUBLISH message contains the call taker's status, in accordance with the presence event package [92]. This makes the automatic call distributor close the session and update the status of the call taker to available.

3.3 Integration of Short Message Service (SMS)

We focus on three problems in our approach of integrating the SMS origination network to the NG9-1-1 architecture. The problems are conveying location information using SMS messages, converting SMS messages to SIP messages, and maintaining a persistent session with a call taker during a conversation.

3.3.1 Location Conveyance

As stated in the background on NG9-1-1 in Section 2.2, origination networks are responsible for determining and sending the location information of the subscriber in case of an emergency. This is also true with SMS networks. Here, we adopt an endpoint-centric approach, which assumes that the cellular phone determines its own location information.

Our approach of sending the location information is to append it to the user's SMS message. Cellular phones obtain location information as geographic coordinates from the assisted GPS (A-GPS). Latitude and longitude in four digit precision can be written in 18 characters without any compression or omission. For example, "+40.8101 -073.9612" takes 18 characters including the decimal points and the space between latitude and longitude. Even after appending the location information, the user still has space to enter 142 GSM 7-bit characters.

If the space consumed by the location information is found to be a problem, we can reduce the number of bytes needed to approximately 12, by base-64 encoding two 32-bit floating point numbers. Treating the coordinates as 3-byte fixed-point numbers can reduce the overhead to about 9 characters.

The main advantage of this approach is that the existing SMS network infrastructure can be used without any modification. However, it comes at the cost of deploying a new SMS application that recognizes an emergency SMS message and appends location information to it.

3.3.2 Converting SMS to SIP

The SMS gateway is an entity within the network that converts an SMS message to a SIP MESSAGE request and vice versa. When the SMS gateway receives an SMS message with location information, it extracts the user's cellular phone number, the user's message, and the location information. With the location information, it queries a LoST server to obtain the most appropriate ESRP. Once it gets a reply back from the LoST server, it has everything it needs to compose a SIP MESSAGE request.

The SIP From header field is set to the SIP URL formed by combining the user's phone number and the IP address of the SMS gateway. An example is "sip:+19171234567@10.0.1.2".

The IP address is appended to make sure that call taker's replies reach the SMS gateway. The SIP To header field and the request-URI are filled with the emergency service URN, urn:service:sos, as the emergency message identifier [96]. The SIP Route header field is set with the ESRP address discovered from the LoST server so that the SIP MESSAGE request is sent to the correct ESRP [91]. The user's message and the location information are packed into the SIP MESSAGE request, and the request is forwarded to the ESRP.

When the call taker's reply arrives at the SMS gateway, it extracts the user's phone number from the SIP To header field and the call taker's message from the SIP MESSAGE request body. The SMS gateway then sends the call taker's message to the user's cellular phone in an SMS message.

3.3.3 Dealing with Multiple Messages within a Single Session

Within the NG9-1-1 network, the same state-keeping mechanisms we proposed for the IM origination network are reused in the SMS origination network without any change in the software or its configuration. That is, the soft-state mechanism used in the ESRP, the hard-state mechanism used in the automatic call distributor, and the explicit end-of-communication notification from the call taker software to the automatic call distributor, are reused to deal with multiple SMS messages within the same session. After all, both IM messages and SMS messages are converted to SIP MESSAGE request, so the same challenges and solutions apply.

In the origination network, the state-keeping mechanisms for IM and SMS messages differ. In contrast to the IM scenario, it is not possible to use an UI-based session since SMS applications do not provide windows that are persistent throughout the session. Therefore, a soft-state mechanism was implemented in the SMS gateway. This is the same mechanism used in the ESRP: the originating address and the destination address are stored for a limited amount of time. During this time, all messages from the same originating address are routed to the same destination address regardless of changes in location. In the SMS gateway, the originating address is the user's phone number and the destination address is the SIP URL of the ESRP.

3.4 Implementation

3.4.1 IM Prototype System

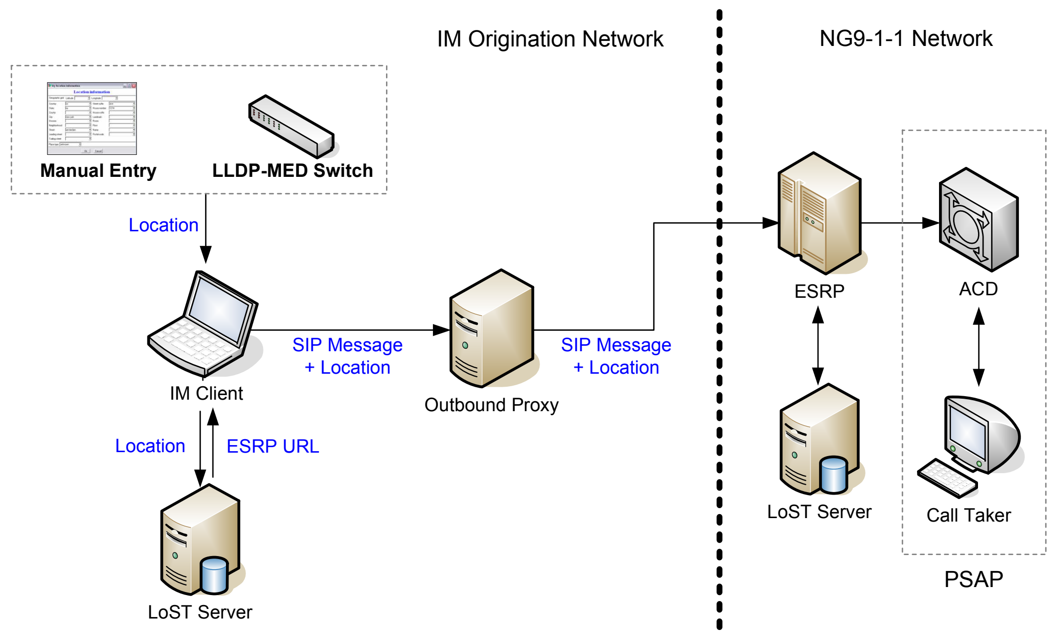


Figure 3.1: IM prototype system.

Our IM prototype system is divided into two parts, as shown in Figure 3.1. On the left hand side is the representation of the IM origination network. It consists of an IM client, a SIP proxy, and a public LoST server for location-to-ESRP resolution. On the right hand side is the NG9-1-1 network, which is a network of PSAPs. It consists of an ESRP, a private LoST server, and servers and clients in PSAPs such as the automatic call distributor and call takers' SIP clients.

The IM network is responsible for identifying an emergency conversation request, obtaining and sending the user's location information, and querying for the most appropriate ESRP to send the user's message to. In the prototype system, all of these tasks are performed by the SIP client. We modified SIP Communicator [24], an open source Java-based IM client, to implement these functions as shown in Figure 3.2.

An emergency conversation is initiated when the IM user presses the emergency button on SIP Communicator. The request-URI and the SIP To header field are set as `urn:service:`

Figure 3.2: Modified SIP Communicator with an emergency button.

sos in order to distinguish emergency SIP MESSAGE requests from others [96].

We modified SIP Communicator so that it can obtain the user's location information either manually from the user or automatically from LLDP-MED capable switches. LLDP-MED is a link layer protocol that allows a network device to discover its physical location. When requested, an LLDP-MED enabled network switch periodically sends the location information of an end device to the port which the device is attached to [35].

SIP Communicator then uses the location information to query a public LoST server to determine the most appropriate ESRP. The modified SIP Communicator goes through these steps of location configuration and ESRP resolution when it first loads and runs on a computer and also whenever the user requests an emergency IM session.

When the user enters a message, the message and the location information is packed into one SIP MESSAGE request and sent towards an ESRP. Location information is embedded in the request body as PIDF-LO, which is a standard XML format for conveying location information [119]. Also, the SIP Geolocation header is added to indicate that the request contains location information [86].

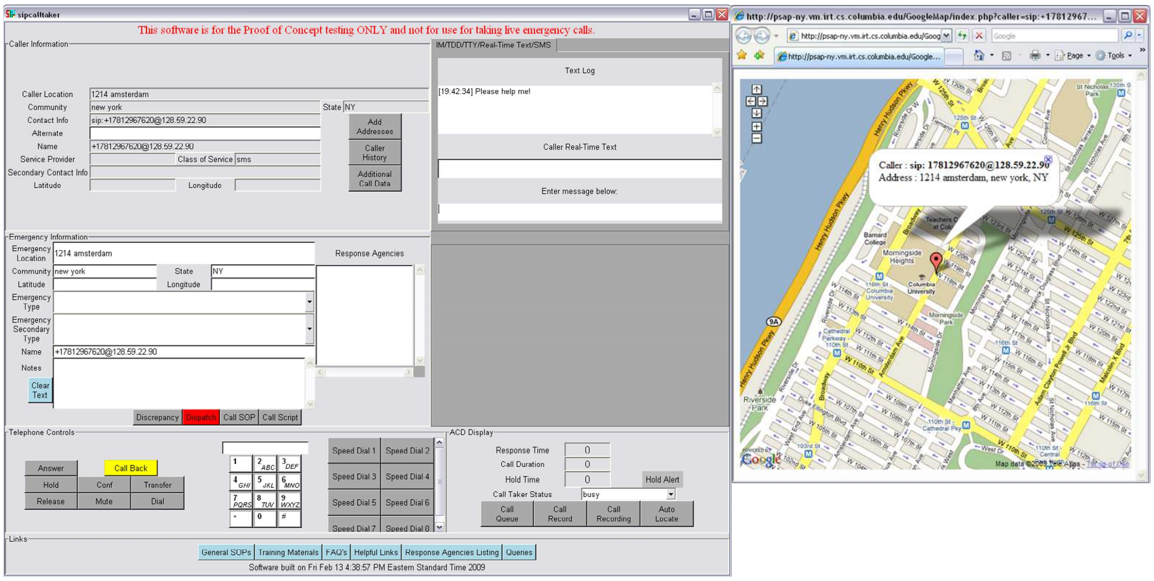


Figure 3.3: Call taker software with Google Maps for IM.

The message sent by the user travels from the originating IM network to the NG9-1-1 network. Within the NG9-1-1 network, the message first hits the ESRP. As described in the previous section, the ESRP forwards the message to a PSAP based on the soft-state mechanism, and then the automatic call distributor in the PSAP forwards the message to a call taker based on the hard-state mechanism. When the call taker software receives the message, it alerts the call taker of a new message using a popup window and populates the call taker's screen with the message, the location information, and the user's other information as shown in Figure 3.3.

When the call taker replies back to the user, the message travels straight from the PSAP to the originating network, bypassing the ESRP. In this case, the call taker software and the automatic call distributor already know the user's originating address, so going through the ESRP is unnecessary.

3.4.2 SMS Prototype System

The SMS prototype system, shown in Figure 3.4, is divided into the SMS origination network and the NG9-1-1 network. The NG9-1-1 network used in the SMS prototype is identical to the NG9-1-1 network used in the IM prototype.

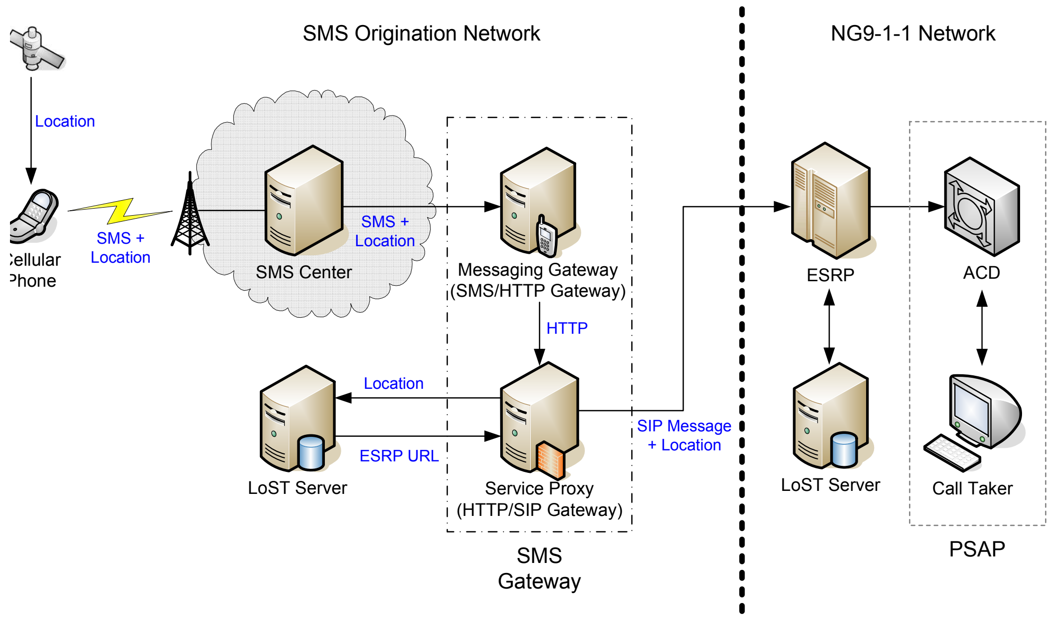


Figure 3.4: SMS prototype system.

The origination network in our SMS prototype consists of the Verizon Wireless cellular network, an SMS gateway, and a LoST server.

As part of the prototype, we used a Verizon Wireless cellular feature phone with VZ Navigator [33] installed[3] as shown in Figure 3.5. VZ Navigator is a location-aware application for Verizon Wireless subscribers. It allows users to determine the cellular phone's location and send it to remote parties using SMS messages, although SMS messages can only be sent to other Verizon Wireless subscribers. The VZ Navigator application uses A-GPS to determine the geographic coordinates of the cellular phone. Once it acquires the coordinates, the server-side software performs reverse geocoding on the coordinates and converts it to a civic address. Therefore, the location information that is sent by VZ Navigator contains both a civic address and geographic coordinates.

Every SMS message sent by the user must contain the user's phone number, the text

[3] Readers are cautioned that the VZ Navigator application is used only as a convenient tool to test some aspects of the emergency SMS system. It is not a solution or part of any solution that is proposed as a fully developed emergency SMS system.

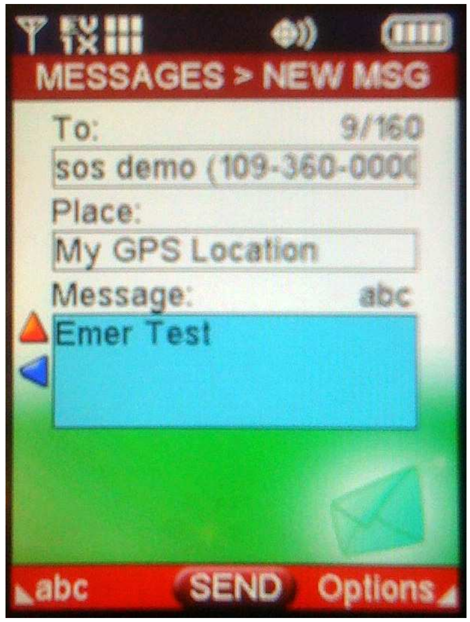


Figure 3.5: Sending SMS message with location using VZ Navigator.

message itself, and the location information. In the prototype, all of these items are sent by the VZ Navigator application.

The prototype provides a five digit short code (10936) for SMS messages. Five digit short codes are closer to emergency numbers since they are easier to remember and enter. However, the VZ Navigator application requires the destination address to be seven digit or ten digit phone number so, for testing purposes, the ten digit short code (1093600000) is used instead. In the United States, the destination of emergency SMS messages should eventually be 9-1-1 [58].

The SMS gateway is built with an SMS-HTTP gateway and an HTTP-SIP convergence server. The SMS-HTTP gateway used in the prototype is Sybase 365's Messaging Gateway [26]. The HTTP-SIP convergence server used in the prototype is called service proxy, which is a web application running on Apache Tomcat server [1]. The service proxy also runs a SIP client based on JAIN SIP [16], an open-source Java SIP stack.

There are two reasons for building the SMS gateway this way. First, we wanted to integrate production network servers into the prototype to make it more realistic. The SMS-HTTP gateway in the prototype is a production server that handles live SMS messages.

Second, we used commercial off-the-shelf (COTS) products as much as possible in the prototype. Instead of building a one-unit SMS-SIP gateway from scratch, we configured two COTS products to work together. By using readily available and market-proved products, we reduced development and testing time. Also, COTS products are replaceable. Service proxy, the HTTP-SIP convergence server in the prototype, can be replaced with similar products like SailFin [29].

When the SMS-HTTP gateway receives an emergency SMS message, it generates an HTTP POST message which contains the user's phone number and the text. The text is a concatenation of the user's message and the location information. The gateway is shared with other services, so the SMS messages used for this project are distinguished by four characters we enter in front of the actual message: "emer". Without the "emer" in the beginning of the message, the SMS-HTTP gateway will not properly process the message.

There are a couple of steps that the service proxy goes through to convert this HTTP POST message into a SIP MESSAGE request. First, it extracts the user's phone number and the text from the HTTP POST message body. Second, the service proxy separates the user's message from the location information. Then the "emer" tag in the user's message is trimmed. The location information, which is an unstructured one-line data format such as "1214 Amsterdam Ave. New York, NY 10027", is transformed into a structured PIDF-LO [119] format. Third, the service proxy generates a query to the LoST server using PIDF-LO to determine the most appropriate ESRP to forward this message. Once it gets a reply back from the LoST server, it composes and sends a SIP MESSAGE request.

When the service proxy receives a "200 OK" SIP response, indicating that the PSAP has received the message, then it also sends a "200 OK" HTTP response to the SMS-HTTP gateway. This ends the HTTP POST transaction from the SMS-HTTP gateway to the service proxy.

The reply from the call taker takes the reverse path within the SMS gateway.

We observed that most SMS messages[4] are delivered to the call taker in 10 seconds on

[4] There is a configuration problem in the cellular phone which we couldn't solve by the time of publication that does not allow sending a complete emergency SMS message with the location information. We measured the average delay using plain SMS messages without the location information. The only difference in average

average after they are sent from the cellular phone. The latency measured between the service proxy and the call taker software is under 1 second. However, depending on various factors such as network congestion or signal strength, these results may be different [78].

3.5 Conclusion

In this chapter, we introduced our approach to integrating IM and SMS networks into the SIP-based NG9-1-1 architecture. In the case of IM networks, one problem is that most popular IM protocols are proprietary. Fortunately, the IM providers use SIP to interconnect with each other. So we proposed a SIP-based model of integration. Another problem is consistent delivery of multiple messages within a session to the same call taker. We solved this problem by implementing state-keeping mechanisms in three different components within the NG9-1-1 architecture.

In the SMS network, there are three problems: location conveyance, SIP conversion, and consistent delivery of multiple messages within a session to the same call taker. The first problem was solved by appending the location information within the SMS message. We developed an SMS gateway to solve the second problem. The third problem had a solution identical to the solution for the IM network except for one component. Instead of keeping state at the endpoint, we kept it at the SMS gateway.

To build a deployable text communication system, there are problems that need further investigation. One of them is the problem of SMS message segmentation when the size of the user's message and the location information exceeds 160 GSM 7-bit characters. This is a problem since some segments may not contain location information, or the location information may be split into two segments and thus unrecognizable until the segments are reassembled.

Instead of having the cellular phone insert the location, we can also consider the case where the cellular network is responsible for obtaining the cellular phone's location. This also means, there is no need for a specialized SMS application on the cellular phone.

---

delay will come from the location information processing time, but we believe that is negligible. Further note that the configuration problem is not a fundamental problem in the design or implementation of the prototype.

For both IM and SMS origination networks, security of the network is of the utmost importance and needs a thorough investigation.

Chapter 4

Simplifying Service Boundaries for Emergency Call Systems Using Population Density

4.1 Introduction

The explosive growth of mobile devices with GPS capability has brought location-based services (LBS) into the mainstream. For example, Pedestrians with smartphones looking for nearby restaurants are commonplace in urban areas. Typical LBSs provide the location of desired services based on the distance from a user's current location [69]. There are, however, an important group of LBSs that use the user's location in a slightly different way. Rather than calculating the direct distance to the user's location, they first map the user's location into a predefined area, and then return a service provider responsible for the area. For example, a 9-1-1 call must be routed to the PSAP station whose service boundary includes the caller's location.

Such a mapping between the user location and the service boundary is common in many LBS systems, and thus can be implemented using a generic component, such as a Location-to-Service Translation (LoST) [100] system. The mapping server maintains a set of polygons which describe geographic areas, which in turn represent service boundaries.

Given a point on earth, the server will find the polygon enclosing the point. When a user uses a mobile device to contact the server, the mobile device can cache polygon information so that multiple queries originating from nearby locations can be processed locally by the mobile device without contacting the server. This reduces the service latency as well as the load on the mapping server. Caching also increases the battery lifespan of the mobile device because it does not need to communicate with the mapping server whenever it moves. The network module is known to be a major source of energy consumption in mobile devices [46]. Caching service boundaries works well in many LBS situations, such as 9-1-1 calls, where the user's location is relatively static.

However, storing service boundaries and processing them can be a significant burden on typical mobile devices which have limited processing power and storage. The number of vertices of a polygon required to accurately represent a geographic area can be very large. In this chapter, we will refer to the number of vertices of a polygon as *the size of a polygon*. For example, the polygon size of the jurisdiction of the Austin police department in Texas is 41,151 [28]. In addition, the large polygon size might be problematic with the limited bandwidth of current mobile networks. The polygon for the Austin police department encoded in Geography Markup Language (GML) [82] is approximately 2 MB.

One way to address this problem is for the mobile client to store a simplified version of a polygon rather than the large original one. A number of polygon simplification algorithms have been developed in digital cartography for map generalization [77], such as the Douglas-Peucker algorithm [54], the Visvalingam-Whyatt algorithm [114], and the bend-simplify algorithm [117]. None of these algorithms, however, satisfies one critical requirement for mappings used in LBS applications: the simplified polygon should be fully enclosed in the original polygon so that the simplified polygon should never produce false positives. For example, if there is an area included in the simplified polygon, but not in the original polygon, a 9-1-1 call from that area will be mistakenly routed to the authority responsible for the original polygon, when in fact, the call has originated from outside of its jurisdiction. This can lead to a significant delay in emergency response.

We present GeoPS-PD, a polygon simplification algorithm designed with LBS in mind. Our GeoPS-PD is based on an algorithm in computer graphics by Cohen-Or *et al.* [49], which generates an inner cover of a shape. We modified Cohen-Or's algorithm for LBS applications, and the resulting GeoPS-PD has the following features:

- It never produces false positives.
- It can be tuned at runtime by providing target polygon size and required area coverage.
- It takes population density into account during the simplification process.

We implemented GeoPS-PD in the LoST server of our NG9-1-1 prototype system [106]. Our evaluation shows a significant performance improvement of LBS queries when a mobile client caches simplified polygons instead of original polygons.

The rest of the chapter is organized as follows. We discuss the requirements of the polygon simplification algorithm for LBS in Section 4.2. We describe our algorithm in Section 4.3 and the implementation details in Section 5.4. Section 5.5 provides performance evaluation. Lastly, we conclude in Section 4.6.

4.2 Requirements of Polygon Simplification Algorithm for Location-Based Services

A polygon simplification algorithm for LBS applications must fulfill the following requirements:

- *Reduction*: the size of the simplified polygon should be significantly smaller than the original polygon.
- *Inclusion*: the simplified polygon must be fully enclosed in the original polygon.
- *Coverage*: the simplified polygon should contain most of the area of the original polygon.

4.2.1 Reduction

Reduction is the primary purpose of polygon simplification. A simplification algorithm should produce a polygon which consists of a much smaller number of points. The perfor-

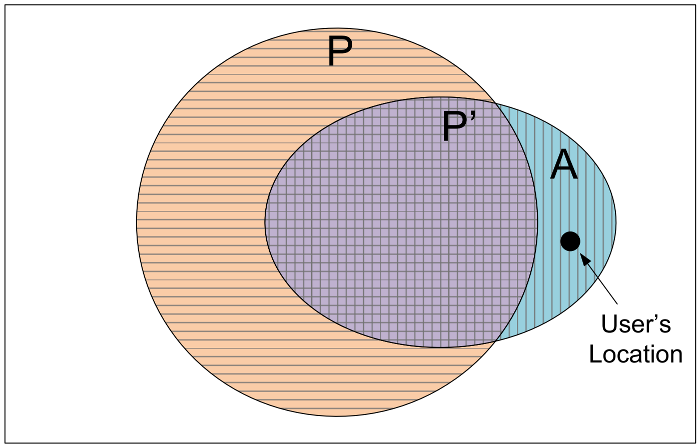


Figure 4.1: False positive case in LBS query.

mance of an algorithm can be measured by the ratio of the simplified size to the original size.

4.2.2 Inclusion

The inclusion requirement prevents the cached polygon from producing false-positives. A false-positive directs a user to an incorrect service provider, as shown in Figure 4.1. Assume that a polygon P represents the boundary of a service provider S, and a polygon P' is the simplified version of P. If P' is not entirely enclosed in P, there is an area A which is inside P', but outside P. If a user in A makes an LBS query, she will be directed to S, the wrong service provider, even though her location is outside S's service boundary.

The simplified polygon must also retain the topology of the original polygon. If there is a hole in the original polygon, it should remain or be replaced by a bigger hole. Otherwise, the simplified polygon will violate the inclusion requirement.

4.2.3 Coverage

One disadvantage of using a simplified polygon for the client cache is that the users in some areas near the border will experience constant cache miss due to the fact that the simplified polygon does not cover those particular areas. To minimize the number of users not covered by the simplified polygon, it should cover as much of the original area as possible. The area coverage, which is the ratio of the simplified area to the original area, is another performance measure of a polygon simplification algorithm.

4.3 GeoPS-PD: Geodesic Polygon Simplification With Population Density

GeoPS-PD is based on Cohen-Or's polygon simplification algorithm [49], which was proposed as the first step to find an inner cover of a non-convex shape in computer graphics. Cohen-Or's algorithm produces a reduced polygon that is fully enclosed within the original polygon, satisfying our inclusion requirement in Section 4.2. We take the core method of Cohen-Or's algorithm, make it tunable, and enhance it to provide better user coverage.

GeoPS-PD takes an adaptive approach in order to produce simplified polygons with better user coverage. The goal is to prevent the simplification process from losing areas where many users make LBS queries. To this end, GeoPS-PD uses query density to guide the polygon simplification process. Query density can be calculated from query history data. Since we do not have such data at this point, we approximate query density using population density.

4.3.1 Population Density

In general, urban areas have higher population density than rural areas. In New York State, for example, New York City has a population density of 26,402 per mile^2, while Hamilton County has 3.1 per mile^2 [108]. A polygon simplification algorithm for LBS applications should take into account the population density so that it does not lose a highly populated area during the simplification process. A simplified polygon that lost a small fraction of New York City would result in a lot more cache misses than the one that lost the same amount of area of Hamilton County.

In order to take the population density into account, GeoPS-PD uses the census tracts boundary information published by the U.S. Census Bureau. A census tract is a small geographic region used by the Census Bureau. Census tracts cover the entire United States and one census tract generally contains between 1,500 and 8,000 people [109]. Since a

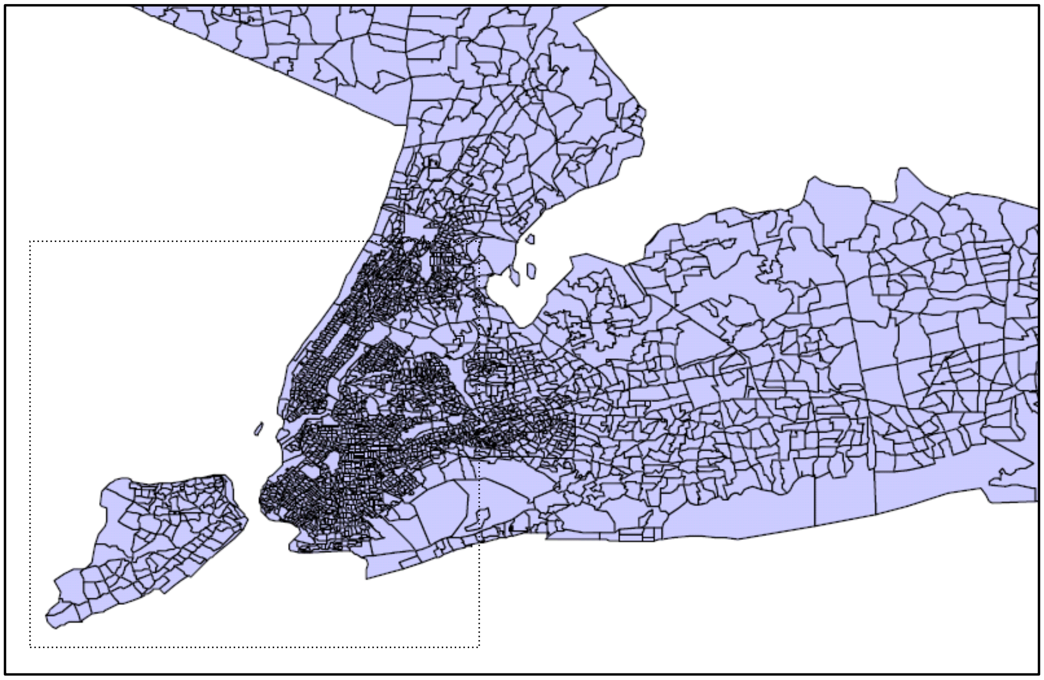


Figure 4.2: Census tracts around New York City.

single census tract contains a comparable number of people, the spatial sizes of census tracts depend on their population density. Figure 4.2 shows that the census tracts in New York City are much smaller in size than those in the nearby counties.

4.3.2 Polygon Simplification

Algorithm 1 describes GeoPS-PD in pseudo-code.

Algorithm 1 GeoPS-PD($P_{org}, N_{target}, R_{req}$)

$P \leftarrow P_{org}$
$P' \leftarrow P$
while $(\frac{area(P')}{area(P_{org})} > R_{req})$ **do**
    if $size(P') \leq N_{target}$ **then**
        return $(success, P')$
    end if
    $P' \leftarrow \{p_1\}$
    $p_{prev} \leftarrow p_1$
    for $i = 2$ **to** $size(P)$ **do**
        triangle $T = \{p_{prev}, p_i, p_{i+1}\}$
        if P does not contain T
        or $\frac{area(T)}{area(P)} \geq \epsilon$
        or $num\_tracts\_in(T) \geq \delta$ **then**
            // p_i is kept in simplified polygon
            add p_i to P'
            $p_{prev} \leftarrow p_i$
        else
            // p_i gets removed
        end if
    end for
    $P \leftarrow P'$
end while
// failed to reach N_{target} while keeping R_{req}
return $(failure, P')$

There are three input parameters: P_{org}, N_{target} and R_{req}. P_{org} is the original polygon which consists of a set of vertices, $p_1, p_2, \ldots, p_n$. N_{target} is the desired polygon size. A successful execution of the algorithm will return a simplified polygon whose size is less than or equal to N_{target}. R_{req} is the required area coverage. For example, if R_{req} is 0.9, GeoPS-

PD must produce a simplified polygon that covers at least 90% of the original area. If it cannot reach N_{target} while keeping R_{req}, GeoPS-PD will return its best effort.

For each point p_i in the polygon P, GeoPS-PD considers if p_i can be eliminated in the simplified polygon P' by checking three conditions (line 11). First, we consider the triangle T formed by p_i and its two neighboring points p_{prev}, p_{i+1}. The point p_{prev} is the last point kept. If T is fully enclosed in P, T indicates a convex region in P, and thus p_i can be safely eliminated without violating the inclusion requirement. Second, we make sure that eliminating p_i does not lose too big an area. We define an acceptable loss threshold, ϵ, which is typically a small percentage (we used 0.2% in our evaluation), and compare it with the ratio of the area of T to the area of P. Lastly, we take the population density into consideration. We count the number of census tracts within T and compare it with a predefined threshold δ (we used 10 in our evaluation). We eliminate p_i only if T contains fewer census tracts than δ. This prevents the simplified polygon from losing highly populated areas. The algorithm runs repeatedly until the simplified polygon's size reaches N_{target} (line 4) or the area coverage drops below R_{req} (line 3).

Note that, for clarity, the pseudo-code omits a few details of our implementation. Checking for population density can be turned on and off. Determining when to terminate the algorithm is slightly more complicated than shown in Algorithm 1 because we make sure that the area does not dip below R_{req} during the last iteration, and that the loop terminates when the algorithm fails to converge. We also take into account various holes in a polygon caused by lakes or autonomous jurisdictions.

4.4 Implementation

We applied GeoPS-PD to the LoST server. We use the LoST server to route emergency calls in our NG9-1-1 system, but it is a generic component which can be used for any LBS application that maps the user's location to the service boundary. The client-side application first determines its location and sends a request to the LoST server. The server then replies with a response containing the appropriate service provider's URL and a polygon representing the service boundary. After receiving the response, the LoST client caches the polygon for later queries.

4.4.1 Client

We implemented our LoST client on an Android Dev Phone 1, equipped with Qualcomm 528 MHz processor and 192 MB RAM [47]. It communicated with the LoST server using 802.11g wireless LAN.

The XML description of a polygon in a LoST response message was converted to two arrays–one for latitude and the other for longitude of double precision floating point numbers, and then stored as BLOBs in a SQLite database along with other information. Before making a LoST request, the client first checks the user's location against all polygons stored in the database. For point-in-polygon test, we used the `PNPOLY` function [60].

4.4.2 Server

The LoST server was hosted on a Dell PowerEdge 1950 Server, equipped with dual Intel Xeon 1.6 GHz CPUs and 2 GB RAM running Linux 2.6.18. We implemented the LoST server as a Java Servlet running on Apache Tomcat [1] web application server. We used the PostgreSQL database [22] to store LoST mappings. A service boundary was stored in the database as a geometric object. PostGIS extension [21] adds geometric objects and functions to the PostgreSQL database. PostGIS provides the `ST_Contains` function for point-in-polygon test.

4.4.3 GeoPS-PD Function

We implemented the GeoPS-PD algorithm as a function written in PL/pgSQL [20], the procedural language for the PostgreSQL database system. We compute the simplified polygon for each service boundary using the GeoPS-PD function, and store it in the database along with the original polygon. The computation is done offline.

4.4.4 Geographical Data Set

We used the actual state boundaries of the United States as the polygons representing service boundaries. In our NG9-1-1 system, 9-1-1 calls are first routed to a statewide emergency service routing proxy (ESRP), and the ESRP further routes the calls to local authorities [98]. From the user's perspective, therefore, the state boundaries make up the 9-1-1 service boundaries.

We used the state boundary data available from the U.S. Census Bureau [5]. The original data file is in the ESRI shapefile [56] format. We loaded it into the PostgreSQL database using the `shp2sql` data loader included in PostGIS. The census tracts data file was loaded in the same way.

4.5 Evaluation

4.5.1 Polygon Simplification

We evaluated the performance of GeoPS-PD using the size and area coverage of the simplified polygons. We did not measure the running time of GeoPS-PD. The running time of the algorithm is not critical because the simplified polygons need to be recalculated only when the service boundaries change and the calculation is done offline.

For our evaluation, we picked five US states (or state equivalent areas), and simplified their boundaries using GeoPS-PD. The five states are District of Columbia, Utah, Massachusetts, New York, and Texas. They represent various points in the polygon size spectrum, ranging from 229 to 6,534.

Table 4.1: GeoPS-PD results for 5 states. ($N_{target} = 100, R_{req} = 0.9, PD = \text{off}$)

State	Original size	Simplified size	Simplification ratio (%)	Area coverage (%)
DC	229	61	26.64	97.01
UT	533	97	18.20	98.12
MA	1,421	100	7.04	92.55
NY	3,093	94	3.04	95.05
TX	6,534	91	1.39	94.77

Table 4.1 shows the performance of GeoPS-PD. The population density option (PD) was turned off. The simplification ratio is the ratio of the simplified polygon's size to the

original size in percentage, and the area coverage is the ratio of the simplified polygon's area to the original area. In the case of New York, for example, the original polygon size is 3,093 and the simplified polygon size is 94, which is only about 3% of the original size but still covers more than 95% of the original area.

Table 4.2: States that were affected by PD.

State	Cities with high population density
California	San Francisco, Long Beach, and San Diego
Florida	St. Petersburg
Illinois	Chicago
Kentucky	Covington
Michigan	Detroit
Missouri	St. Louis
New York	New York City
Pennsylvania	Philadelphia
Texas	McAllen
Virginia	Arlington
Washington	Seattle

When we ran GeoPS-PD with the population density option turned on, 11 out of the 50 states produced results that differed from the population-agnostic simplified polygon. Table 4.2 shows the 11 states along with the densely populated cities in each state that caused the differences.

Table 4.3: Simplification of New York with and without PD. ($N_{target} = 100, R_{req} = 0.9$)

	Polygon size	Area coverage (%)	Census tracts included
Original polygon	3,093	100.00	4,711
Simplified w/o PD	94	95.05	2,611
Simplified with PD	100	95.43	4,186

The results for New York clearly demonstrate the effectiveness of the population density option. Table 4.3 compares the results for New York with and without the population density option. While the polygon size and the area coverage remain similar between the two cases, the number of census tracts included in the simplified polygon is increased by 60%, jumping from 2,611 to 4,186. Put another way, the simplification using population density covers 6.3 million more people, according to the U.S. Census Bureau's statistics which states that the average population of all census tracts is about 4,000 [109]. This is about 1/3 of the total population of New York State.

Figure 4.3 illustrates how such a huge difference in population coverage is possible when the area coverage is almost same. Figure 4.3(a) shows that, without the population density option, most of New York City gets cut off during simplification. Figure 4.3(b) shows that the population density option prevents the loss of New York City.

4.5.2 LoST Query Performance

In order to measure the performance benefit of caching simplified polygons, we analyzed the expected LoST query time for two cases. In one case, the LoST server sends the original polygon which the client then caches. In the other, the server sends the client the simplified polygon. The simplified polygon was computed in advance.

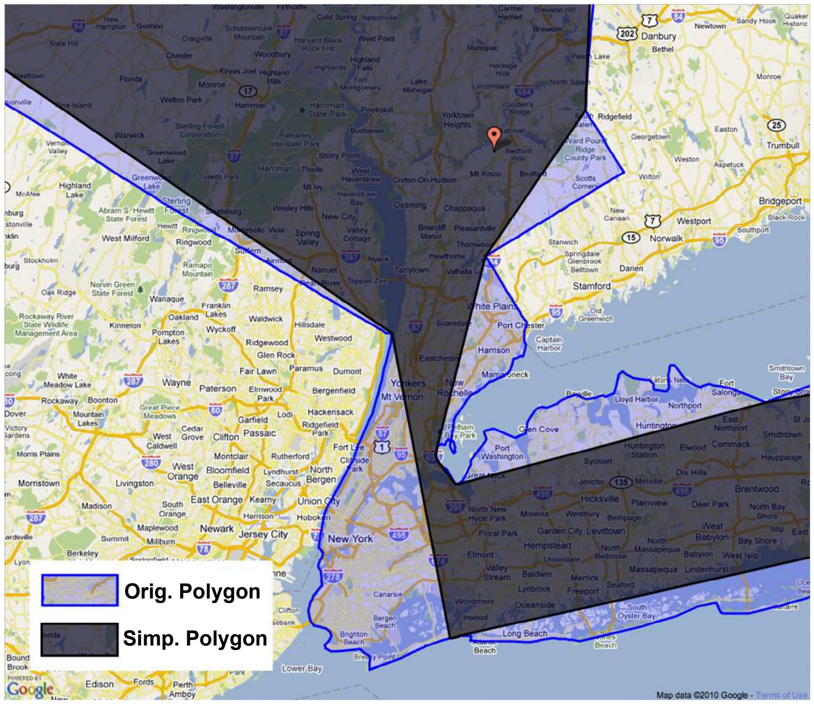
The expected query time is given by the following formula:

$$E(Query\ time) = Hit\ time + Miss\ penalty \times Miss\ rate$$

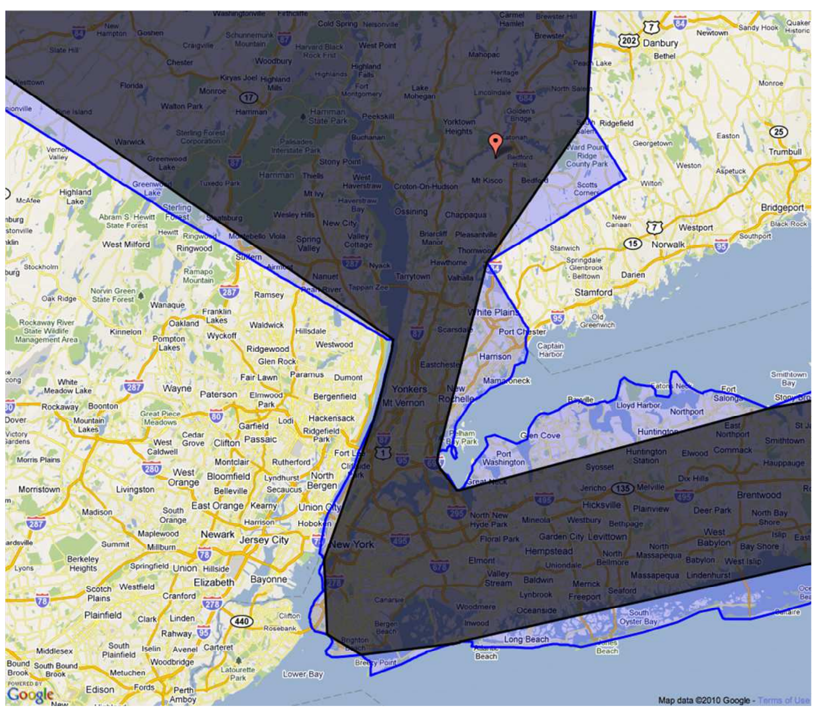
We measured $Hit\ time_{org}$, $Miss\ penalty_{org}$, $Hit\ time_{simp}$ and $Miss\ penalty_{simp}$ as follows. (org and $simp$ denote the original and simplified polygon being cached, respectively.) For the miss penalty, we let the client issue a query with an empty cache and measured the response time. At that point, the client cached the polygon that it received from the server. We let the client issue the same query again, which caused a cache hit, to measure the hit time. We selected the query location to ensure the cache hit.

For the case of original polygons, the only cache miss will occur for the first query in each service boundary. All subsequent queries within the same boundary will result in cache hits. Thus, as the number of query increases, the expected query time converges to the hit time:

$$E(Query\ time_{org}) \rightarrow Hit\ time_{org}$$



(a) without population density option



(b) with population density option

Figure 4.3: Polygon simplification of New York State near New York City.

For the case of simplified polygons, a cache miss will occur every time the query location is in a region that is included in the original polygon but not in the simplified polygon. The total percentage of such regions in a service boundary is $1 - \mathit{Area\ coverage}$, which is the $\mathit{Miss\ rate}_{simp}$. The expected query time for the simplified polygon case becomes:

$$
\begin{aligned}
E(\mathit{Query\ time}_{simp}) = &\ \mathit{Hit\ time}_{simp} \\
&+ \mathit{Miss\ penalty}_{simp} \times (1 - \mathit{Area\ coverage})
\end{aligned}
$$

Table 4.4: LoST query time for 5 states. (in milliseconds)

State	Original Polygon		Simplified Polygon			
	Size	Hit time	Size	Hit time	Miss penalty	Miss rate
DC	229	67	61	43	466	0.029
UT	533	84	97	58	511	0.018
MA	1,421	148	100	61	516	0.074
NY	3,093	263	100	59	527	0.045
TX	6,534	566	91	46	526	0.051

Table 4.4 shows our measurements for five states: hit time, miss penalty, miss rate, and the polygon sizes. Figure 4.4 compares the resulting query times between the two cases. The expected LoST query times using simplified polygons are faster in all five states, and the bigger the original polygon size, the larger the performance improvement.

To see where the performance improvement comes from, let us look more closely at the numbers for New York. The expected query time using the simplified polygon is 3.17 times faster–82 ms versus 263 ms. This comes from the large difference in hit time–59 ms versus 263 ms. It takes a long time to access and process the cache of the original polygon due to the large size of 3,093 points. On the other hand, the simplified polygon case incurs a heavy penalty of 527 ms on every cache miss, but the low cache miss rate of 0.045 mitigates the effect of the penalty on the expected query time.

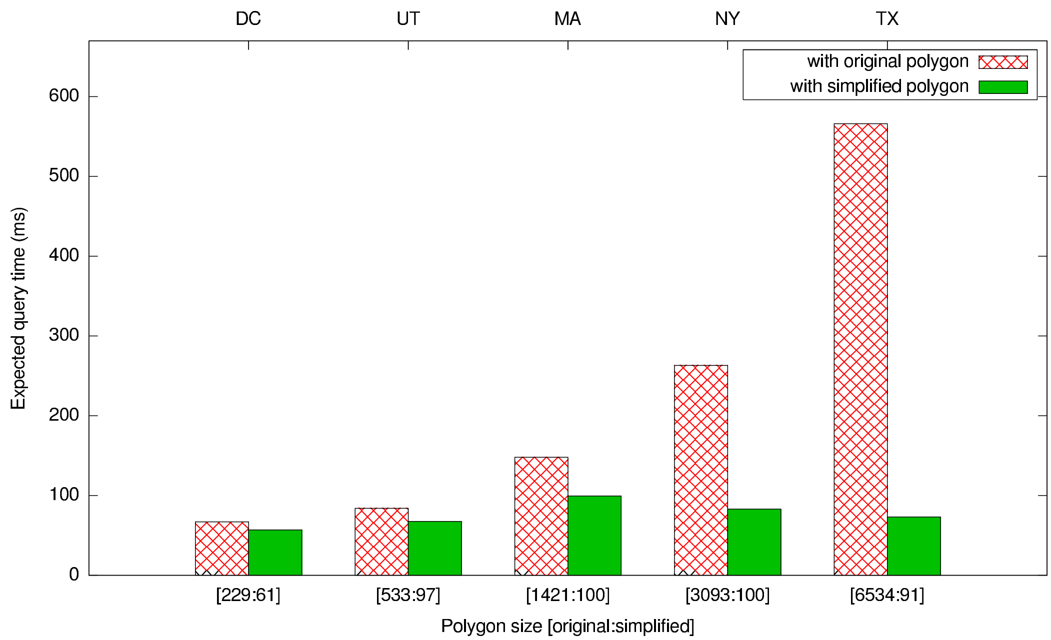


Figure 4.4: Comparison of the expected LoST query time with original and simplified polygons.

4.6 Conclusion

We presented GeoPS-PD, a polygon simplification algorithm for LBS. GeoPS-PD produces a simplified polygon that is fully enclosed in the original one so it never produces false positives. It can be tuned at runtime by providing target polygon size and required area coverage. It also provides an option to take the population density into consideration, which can result in better population coverage in certain cases.

Our measurements with real US state boundary showed that GeoPS-PD produces simplified polygons that are much smaller in size but still cover most of the original areas. Enabling the population density option prevented GeoPS-PD from losing highly populated areas near the state borders. We have shown that the performance of LBS queries can be improved if mobile clients cache simplified polygons produced by GeoPS-PD, especially when the size of the original service boundary is large.

Part II

Indoor Positioning System

Chapter 5

Improving Vertical Accuracy of Indoor Positioning for Emergency Calls

5.1 Introduction

The emergency call systems in the United States and elsewhere are undergoing a transition from the PSTN-based legacy system to a new IP-based system. The new system is referred to as the Next Generation 9-1-1 (NG9-1-1) system [110] in the US and NG112 system [38] in Europe. We have previously built a prototype NG9-1-1 system [68] based on the Session Initiation Protocol (SIP) [93], described in Chapter 2.

The most important piece of information in the NG9-1-1 system is the caller's location. The location is first used for routing the call to the appropriate call center. The emergency responders then use the caller's location to find the caller. Therefore, it is essential to determine the caller's location as precisely as possible to minimize delays in emergency response. Delays in response may result in loss of lives or property.

In order to determine a caller's location, different techniques can be used depending on how the call has been made. If the call came from a landline or a VoIP software on a desktop computer, the caller's location can be determined by looking up a location database. The database can be constructed from the caller's billing address or from MAC-address-to-location mapping [87]. For mobile calls, the Global Positioning System (GPS), cellular-based positioning, and Wi-Fi positioning system are currently in use. GPS is the dominant outdoor mechanism due to wide deployment and high accuracy.

Indoor positioning, however, presents a challenge because GPS does not generally work indoors. Moreover, unlike outdoors, vertical accuracy is very important in indoor positioning because an error of few meters will send emergency responders to a different floor in a building, which may cause a significant delay in reaching the caller. The importance of vertical positioning makes GPS not a good solution even if GPS signals can somehow reach indoors, since the altitudes reported by GPS are usually inaccurate. Lammel *et al.* [72] reported that the altitude from a GPS receiver varied more than 40 m when they measured it at the same point in a building.

Cellular-based positioning is not suitable for indoor positioning because of its low accuracy and lack of vertical location. In FCC's indoor location trials [51], Qualcomm's location solution reports an average error of 136.4 m in a dense urban area. The Qualcomm system is a hybrid of A-GPS and cellular-based positioning techniques such as Advanced Forward Link Trilateration (AFLT) [116], but the system runs in cellular-only mode indoors because GPS signals are unavailable at the test points inside tall buildings. The system does not provide vertical locations.

Ladetto and Merminod [70] proposed a barometer-based solution for vertical positioning. Using a barometric pressure sensor and reference barometers, the NextNav system reported indoor vertical location with an average error of 2 m in FCC's trials [51]. However, today's smartphones are not typically equipped with a barometric sensor. Moreover, barometers have a critical limitation when they are used in a vertical positioning system intended for emergency situations. Firefighters use a technique called positive pressure ventilation (PPV) [67], which means blowing air into a burning building in order to clear out smoke. PPV will result in pressure changes in the building, which will in turn cause large fluctuations in barometer readings. In addition, parts of some buildings are intentionally pressurized for various reasons [57], which will also affect barometer readings.

This chapter presents a proposal to augment our previous NG9-1-1 prototype system with floor localization. We aim to provide floor-level accuracy with minimum infrastructure support. Our approach uses multiple sensors, all available in today's smartphones, to trace a user's vertical movements inside buildings.

When a user enters a building, the user's smartphone receives the information about the building and the current floor from a beacon deployed at the entrance. The smartphone starts tracking the user's vertical movements when she rides elevators or walks on stairs. Additional beacons deployed sparsely throughout the building provide periodic corrections to the user's location.

Our design is largely driven by the requirements for emergency calls. First of all, a positioning system intended for emergency calls must be immune to transient conditions or on-going changes inside the building. For example, interfering electromagnetic signals, re-arranged equipment and furniture, or the number of current occupants should not affect the system's operation. Because of this requirement, we had to rule out wireless fingerprinting, an effective technique used in many other indoor location systems [48, 103, 112]. Secondly, the infrastructure should be reduced as much as possible because an extensive infrastructure requirement hinders wide adoption. We chose a hybrid design, combining beacon-based infrastructure and sensor-based dead reckoning, in order to fill the gap between sparsely deployed beacons. Lastly, in an emergency call system, a partial failure must not result in a complete system failure. In our system, partial failures caused by power outage or structural damage in the building result in gradual degradation of performance.

In this chapter, we make three contributions. First, we present a hybrid architecture for floor localization with emergency calls in mind. We believe that the architecture strikes the balance between accurately determining a user's location and minimizing the required infrastructure.

Second, we present the elevator module for tracking a user's movement in an elevator. The elevator module calculates the elevator's displacement by double-integrating vertical acceleration. Double integration is considered too noisy for tracking human movements in general. However, we show that the constrained movement of an elevator enables a number of error correction techniques, making double integration a viable method.

Third, we present the stairway module which determines the number of floors a user has traveled on foot. Previous proposals counted a user's steps on stairs [84, 124]. That approach has a critical limitation that it cannot account for a user walking up multiple stairs in each step. Instead, our stairway module counts landings, the level areas either at the top of a staircase or in between flights of stairs.

This chapter is organized as follows. Section 5.2 presents our overall architecture. Section 5.3 describes the design and algorithms of our three analysis modules and the activity manager. Section 5.4 describes implementation details. Section 5.5 provides our evaluation results. Section 5.6 discusses related work. Lastly, we conclude and discuss future work in Section 5.7.

5.2 Architecture Overview

Figure 5.1 shows the overall architecture of our vertical positioning system. We describe each component in detail in the following subsections.

5.2.1 Sensor Array

The sensor array includes different kinds of sensors available in most of today's smartphones. The Inertial Measurement Unit (IMU) integrates a three-axis accelerometer, a three-axis gyroscope, and a three-axis magnetometer. Thus, the IMU provides motion sensing with a total of nine degrees of freedom. The accelerometer measures linear accelerations along the three spatial axes. The measured accelerations can be used to detect whether a user is moving, and if so, the user's velocity or traveled distance can be derived from them. The gyroscope measures the angular velocities of rotations around the three spatial axes. The orientation of the device can be derived from the gyroscope measurement. The magnetometer is a digital compass that measures the strength of the Earth's magnetic field. The compass provides the heading of the device. Heading refers to the angle which the device forms with the magnetic north on a level plane.

GPS provides the device's location in geodetic coordinates using satellite signals. GPS cannot be used indoors but it can help detect when a user moves from outdoors to indoors.

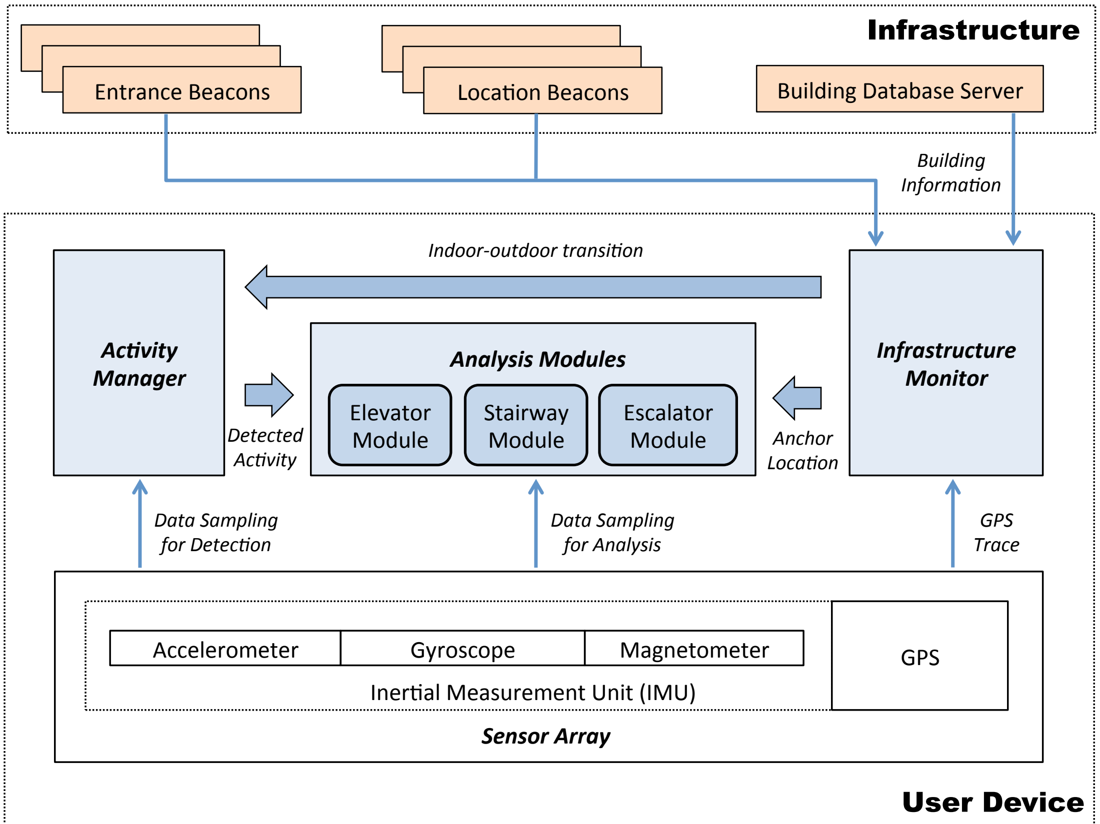


Figure 5.1: Architecture overview of our indoor positioning system.

5.2.2 Analysis Modules

The analysis modules collect data from the sensor array and compute a user's location. There are three analysis modules in our architecture: the elevator module, the stairway module, and the escalator module.

The elevator module calculates the vertical displacement of an elevator by measuring its linear acceleration. The linear acceleration is measured using the device's accelerometer. Integrating the linear acceleration twice with respect to time yields the distance that the elevator has traveled.

The stairway module determines the number of floors a user has traveled by counting the number of landings in stairways. Our landing detection algorithm is based on an intuitive fact that there is less vertical movement on landings than on steps. The stairway module utilizes the accelerometer, the gyroscope, and the magnetometer. We describe the details in Section 5.3.2.

The escalator module also calculates the vertical distance that a user has traveled by double-integrating the vertical component of the acceleration measurements, as we did in the elevator module. In both escalator and elevator modules, the vertical distance is converted to the number of floors by looking up the floor-to-floor heights. The user's smartphone receives the floor height information from the infrastructure components. We describe the infrastructure in Section 5.2.4.

5.2.3 Activity Manager

The activity manager coordinates the interactions between the sensor array and the analysis modules. The activity manager monitors the sensors to detect changes in a user's activity, such as indoor-outdoor transitions, riding an elevator, or walking on a stairway. Once the user's activity is identified, the activity manager selects the proper analysis module to process the data from the sensor array.

5.2.4 Infrastructure

As we will show in Section 5.5, the elevator, stairway, and escalator modules perform well within limited ranges, but the modules cannot reliably capture the user's movement over longer vertical distances. Moreover, the sensor-based components can only report *relative location*, i.e., the number of floors that the user has traveled. Therefore, the initial anchor location must be provided in order to obtain the *absolute location*.

These problems can be solved by deploying an infrastructure for indoor positioning. Densely deployed infrastructure, such as beacons installed every floor and every entrance, can provide accurate location, but the high cost of such installation is a hindrance to ubiquitous deployment, which is an important consideration for an emergency call system. On the contrary, sparsely deployed infrastructure will not be able to provide the required level of accuracy.

Our architecture combines sensor-based dead reckoning with minimum and practical beacon-based infrastructure. First, the infrastructure includes location beacons deployed at each entrance of a building. The beacons provide the location of a user's entry to the building. The floor of entry becomes the anchor for all subsequent calculations of the user's vertical location. In addition to the floor of entry, the beacons also provide other building information which is needed by the analysis modules. The additional building information includes the floor-to-floor height and the number of landings between each pair of floors. User devices include the infrastructure monitor, which interacts with the location beacons.

Second, for the buildings that are not equipped with these beacons, we propose that central authorities such as local governments maintain well-known building database servers. When a user enters a building not equipped with the beacons, the infrastructure monitor sends the last known GPS location to the building database server to retrieve the same building information that the location beacons would have provided. This GPS-based entrance detection is not as reliable as the beacon-based approach, especially in urban canyons. Thus, we only use it as a fallback.

Lastly, the limited range of the sensor-based components can be overcome by sparsely deploying location beacons at the edge of the range. For example, if the location tracked by the elevator module is reliable up to 20 floors, beacons can be placed at elevator entrances every 20 floors.

One advantage of our hybrid architecture is that partial failures caused by power outage or structural damage result in gradual degradation of performance rather than a complete system failure. If an entrance beacon fails, the smartphone will not have the initial anchor location and other building information, but it can still keep track of the user's relative location. If some location beacons are unavailable to provide periodic corrections, the system simply produces less accurate locations. This is an important characteristic of an emergency call system because even incomplete information can be helpful to first responders.

5.3 System Design and Algorithms

5.3.1 Elevator Module

Figure 5.2 shows how the elevator module works. In this section, we explain the three core challenges in the elevator module: how to extract the vertical component in the accelerometer measurement, how to subtract Earth's gravitational acceleration, and how to eliminate

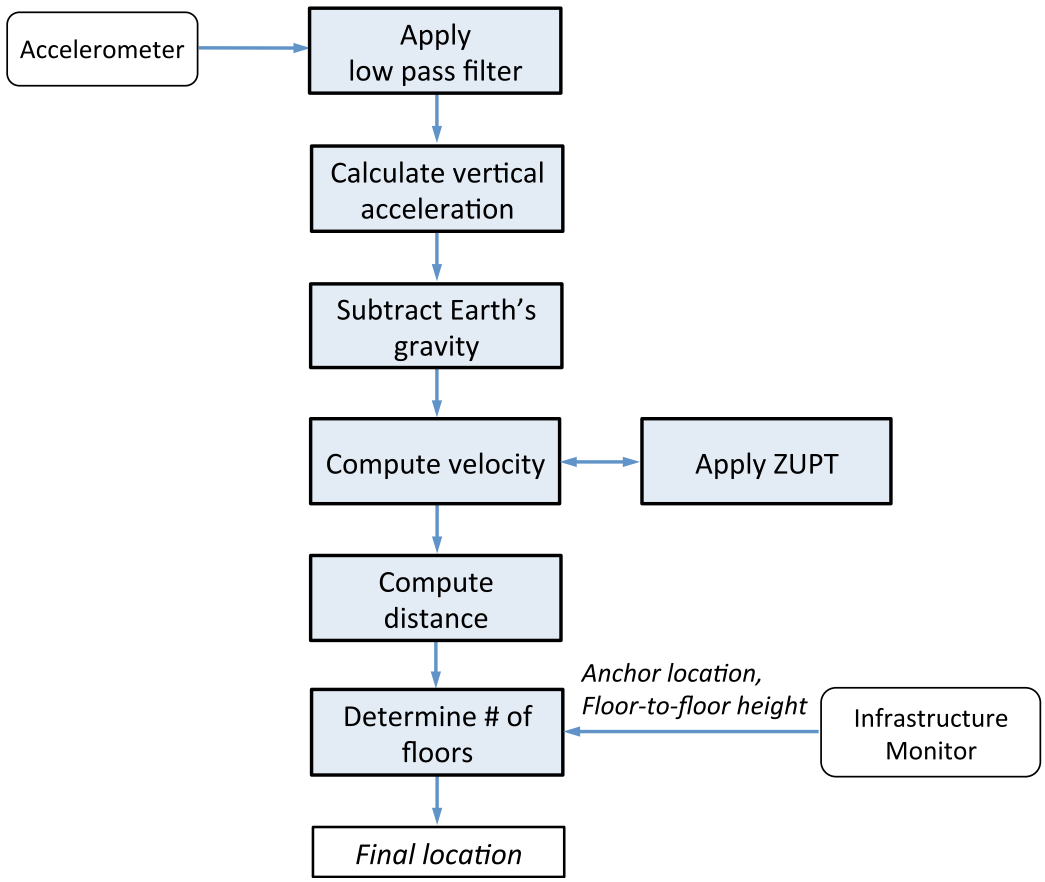


Figure 5.2: Overview of the elevator module.

noise and errors.

The accelerometer returns linear accelerations along the three axes. Those three axes are not aligned with the world coordinate system. Instead, they are aligned with the frame of the device as shown in Figure 5.3. Thus, the axes in the device coordinate system keep changing as the orientation of the device changes. One way to extract vertical acceleration is to combine the accelerometer measurement with the gyroscope measurement. In fact, we do this in the stairway and escalator modules. In the elevator module, however, we take advantage of the fact that, in the elevator, the dominant movement of the device is in the vertical direction. We simply assume that the measured acceleration is close to vertical, and approximate the vertical projection with the vector itself. Thus, the vertical acceleration is

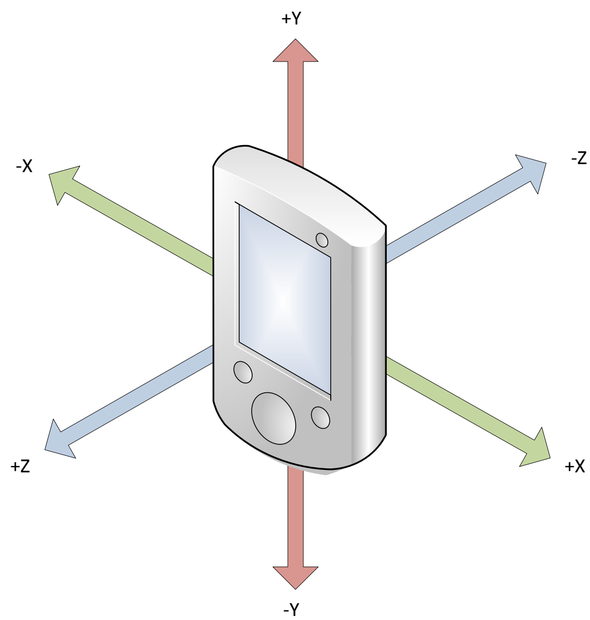


Figure 5.3: Three axis accelerometer in smartphone.

calculated as follows:

$$a_{vertical} \approx |\overrightarrow{a}| = \sqrt{x^2 + y^2 + z^2} \tag{5.1}$$

where x, y, and z are three-axis accelerometer measurements. We do not need a gyroscope in this calculation. We justify our approach by making the following two observations. First, a user's sudden movements in the elevator will be filtered out by the low-pass filter, which we will describe shortly. Second, users typically stand still in the elevator, and when they move, the accelerations of the movements are small compared to the vertical acceleration of the elevator. The consequence of this approximation is that whenever there is non-vertical acceleration, we overestimate the vertical acceleration by $\frac{1}{\cos\theta}$, where θ is the angle that the measured acceleration vector makes with the vertical axis. This overestimation is small, and we compensate it by applying zero velocity update (ZUPT), which we describe later. Our measurement shows that the approximation does not affect the resulting distance calculation.

The vertical acceleration calculated above includes the gravitational acceleration (g), which we need to subtract before computing the traveled distance. In theory, g should be constant at $9.8\,\mathrm{m/s^2}$, but we found slight variations in our experiments. Table 5.1 shows

Device	Measured g at loc #1	Measured g at loc #2
iPhone 4S	9.854 m/s	9.853 m/s
iPhone 4	9.870 m/s	9.930 m/s

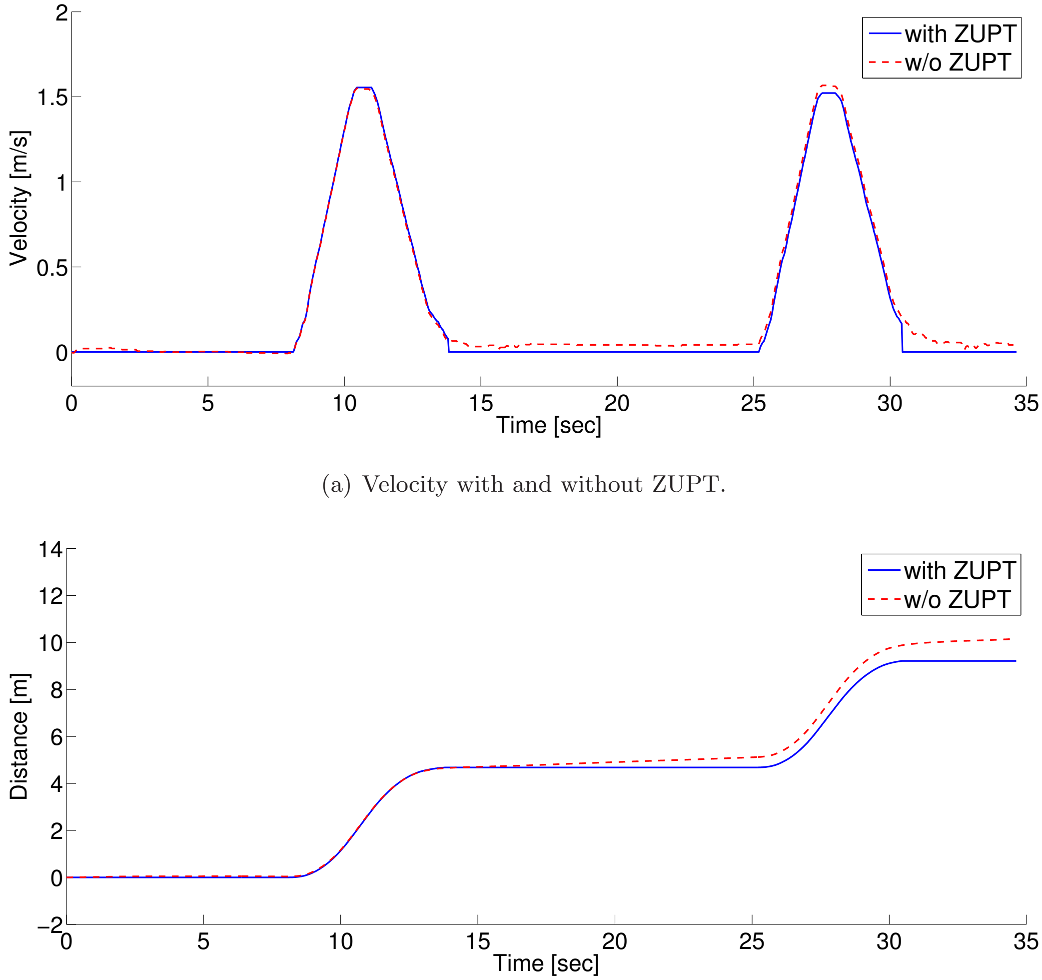
Table 5.1: Measured g using different devices at different locations.

the average vertical accelerations measured by two different smartphones sitting still on a desk. The sampling frequency and duration are 30 Hz and 60 seconds, respectively. We repeated the measurement at two different locations where the altitudes were similar to each other. The measured values deviated slightly from g, and moreover, the variations were different on different devices and at different locations. Smartphone SDKs provide APIs returning g-free acceleration, but they exhibited the same deviation. Small deviation in the acceleration will cause a large drift in the user's location since we double integrate the acceleration to calculate the distance. We eliminate the effect of the deviation in g as follows. We take advantage of the fact that, if we take g out of the acceleration, the integral of the acceleration taken over the duration of the trip must be zero because the elevator is not moving at the end of the trip. Thus we can deduce that the value of g measured by the device is the mean of the acceleration samples taken over the trip.

The output from the accelerometer contains a significant amount of noise. We apply two existing techniques to tackle this problem. First, we apply the low-pass filter to the accelerometer output. The low-pass filter can be represented by the following formula:

$$\begin{aligned} y_i &= \alpha * x_i + (1 - \alpha) * y_{i-1} \\ \alpha &= \frac{1}{f} / (\frac{1}{f} + \frac{1}{fc}) \end{aligned} \tag{5.2}$$

where y_i is the out sequence, x_i is the input sequence, f is the sampling frequency, and fc is the cutoff frequency. The sampling frequency and the cutoff frequency we used are 30 Hz and 15 Hz, respectively. This filters out the user's sudden movements and the accelerometer's inherent noise which we refer to as drift. Second, we apply a technique called zero velocity update (ZUPT) [62] to eliminate accumulated errors. Integrating the acceleration yields the velocity of the elevator. We reset the velocity to zero during the period when the acceleration is zero and the velocity is within a predefined threshold. The threshold value we choose is small compared to the speed of the elevator, so that we do not mistakenly zero out the



(a) Velocity with and without ZUPT.

(b) Distance with and without ZUPT.

Figure 5.4: Comparison of the distance calculations with and without ZUPT.

velocity of an elevator moving at a constant speed. The accuracy of the distance calculation is improved in that, at each stop, ZUPT has an effect of wiping out the accumulated errors due to the drift and the user's non-vertical movements.

Figure 5.4 demonstrates the effectiveness of ZUPT. We compare the computed velocities and distances when an elevator traveled from the first, to the second, and then to the third floor. Without ZUPT, the accumulated acceleration errors result in non-zero velocities when the elevator is at the second and the third floor. This in turn results in an error of approximately one meter in the distance calculation at the end.

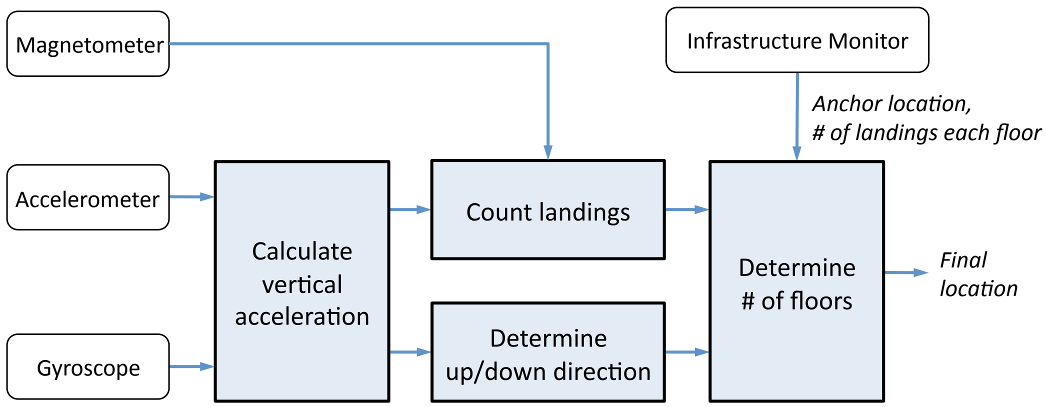


Figure 5.5: Overview of the stairway module.

In general, double integration is considered too noisy for tracking human movements. In our case, however, an elevator moves only in the vertical axis, making it easy to extract the vertical component of the acceleration. An elevator also comes to a zero velocity when it stops at a floor, making it possible to apply ZUPT to eliminate the accumulated errors.

5.3.2 Stairway Module

The stairway module determines the number of floors a user has traveled using our landing counting algorithm. To the best of our knowledge, landing detection has not been used for vertical positioning systems.

Figure 5.5 illustrates how the stairway module works. First, the stairway module calculates vertical acceleration from the accelerometer and gyroscope measurements. Unlike an elevator's movement, a user's movement on a stairway is more complex. A gyroscope is needed to transform the acceleration in the device coordinate system to the world coordinate system. We convert the accelerometer measurements in the device coordinate system to the world coordinate system using a rotation matrix as shown below:

$$\overrightarrow{a}' = R\overrightarrow{a} \tag{5.3}$$

where $\overrightarrow{a}'$ is the acceleration in the world coordinate system, $\overrightarrow{a}$ is the acceleration in the device coordinate system, and R is the rotation matrix. Most smartphone platforms provide an API to obtain R. We then take the resulting z-axis acceleration in the world coordinate system and subtract g from it. We calculate g in the same way as in the elevator module.

The landing counting algorithm compares the amplitude of vertical acceleration between steps and landings. The algorithm is based on the intuitive fact that the amplitude of the vertical acceleration is much smaller on landings than on steps because there are less vertical movements on landings.

Figure 5.6(a) shows a measurement of a user's vertical acceleration when she walks down four floors passing eight landings. The amplitude difference between steps and landings is clearly observed. Figure 5.6(b) is the magnitude spectrogram $|X(t,f)|$ in dB scale, transformed from Figure 5.6(a)'s acceleration data. The regions of small amplitude in Figure 5.6(a) manifest as reduced magnitude in the frequency range between 0.5 to 2 Hz, which corresponds to human walking.

We define $p_{walk}(t)$ to extract human walking activity from the magnitude spectrogram:

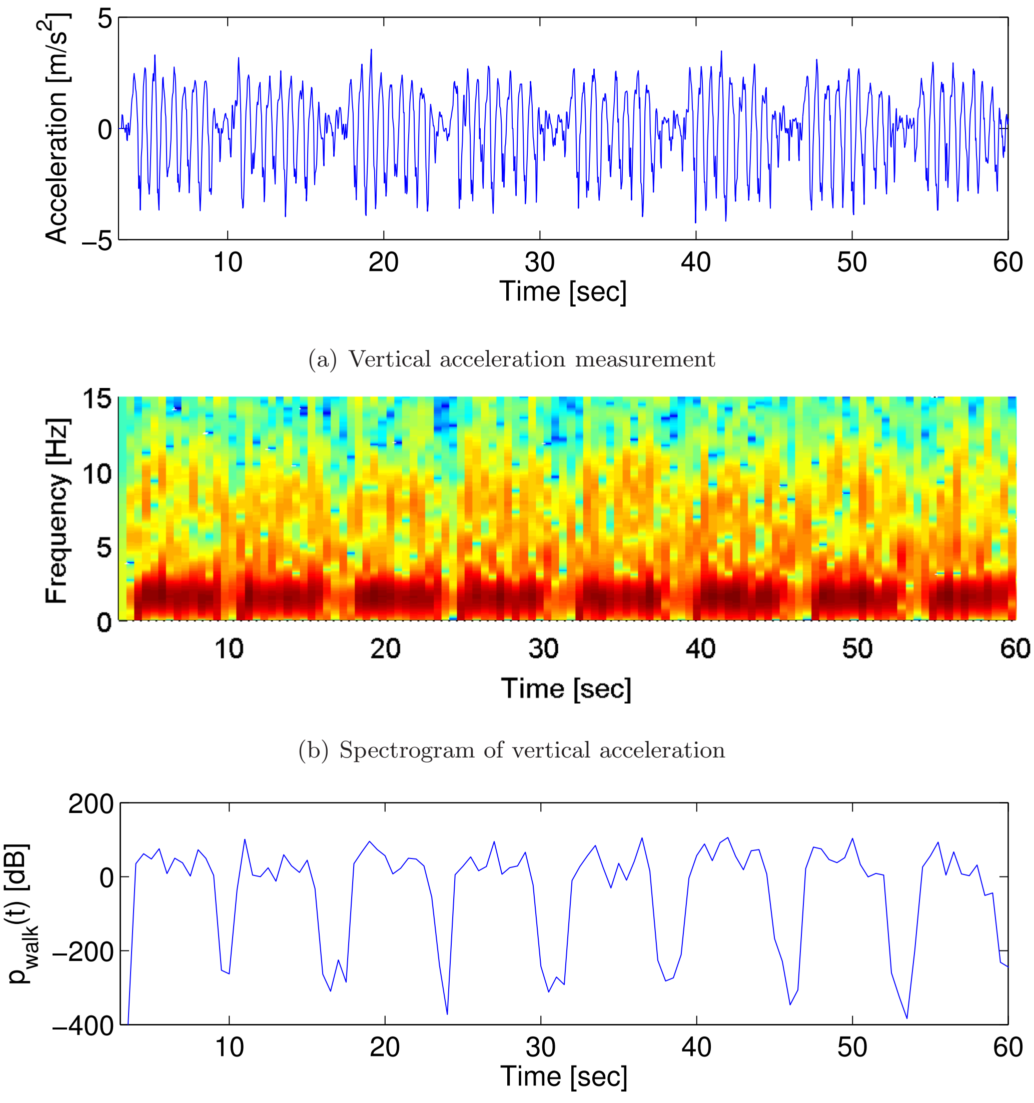
$$p_{walk}(t) = \sum_{0.5\,\mathrm{Hz}<f<2\,\mathrm{Hz}} 10\log_{10}|X(t,f)| \tag{5.4}$$

where t is time and f is frequency. Figure 5.6(c) shows $p_{walk}(t)$, where we can clearly observe the dips at landings.

Our landing counting algorithm traces the p_{walk} level shown in Figure 5.6(c) to count the number of landings. Figure 5.7(a) illustrates this process. Each landing is characterized by a dip below its mean value. The fall and rise of the level crossing the mean value indicate the beginning and end of a landing, respectively. The beginning and end of a landing are shown as the bumps of the "Landing detection" line in Figure 5.7(a).

In addition to vertical acceleration, the stairway module uses heading information from the magnetometer to improve the accuracy of landing detection. Most of the time, users turn around 180 degrees on landings. We use such heading changes to correct errors in landing detection, specifically to remove incorrectly identified landings. Since we are only interested in 180 degree turns, our magnetometer reading does not require calibration.

Figure 5.7(b) shows a case where our algorithm removes two incorrectly identified landings using the heading information from the magnetometer. The dotted line labeled "Heading" shows the heading changes reported by the magnetometer. The heading largely stays the same from 15 sec to 25 sec, and changes from 220° to 40° in the next two seconds. This 180° turn, combined with the bumps on the landing detection line, confirms a landing. Note

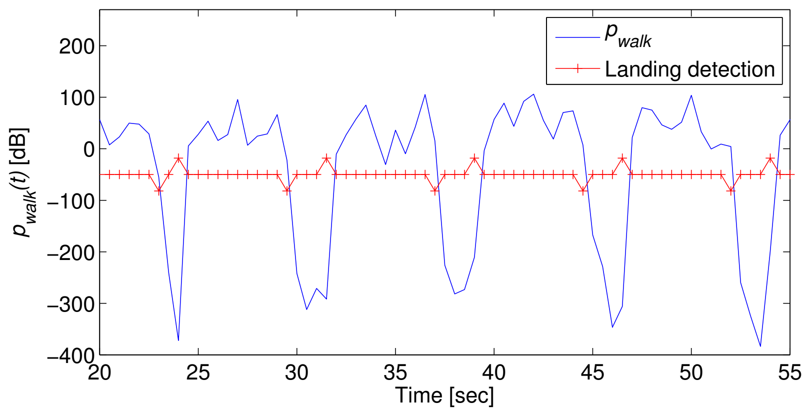


(a) Vertical acceleration measurement

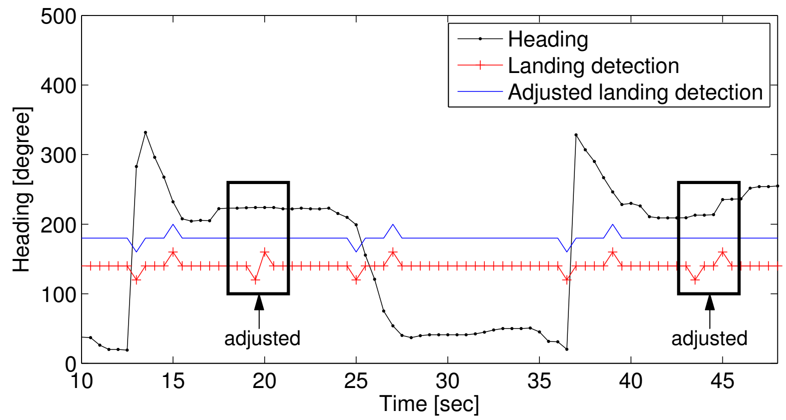
(b) Spectrogram of vertical acceleration

(c) $p_{walk}(t)$ extracted from (b)

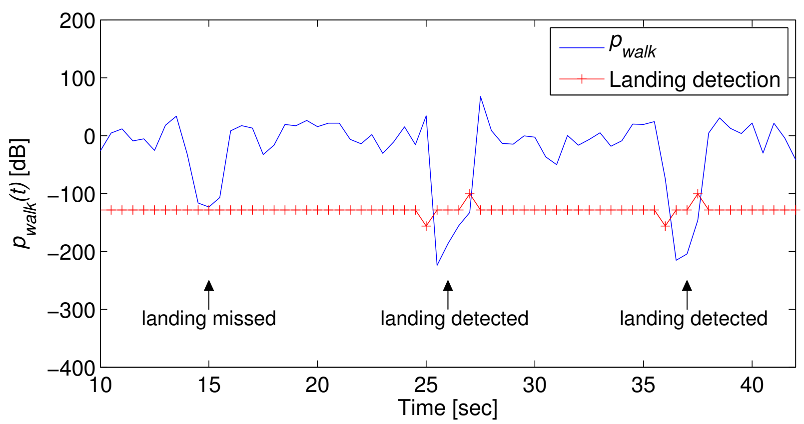
Figure 5.6: Landing detection using vertical acceleration (4 floors, 8 landings).



(a) Case 1: no error.



(b) Case 2: false positives get fixed by magnetometer.

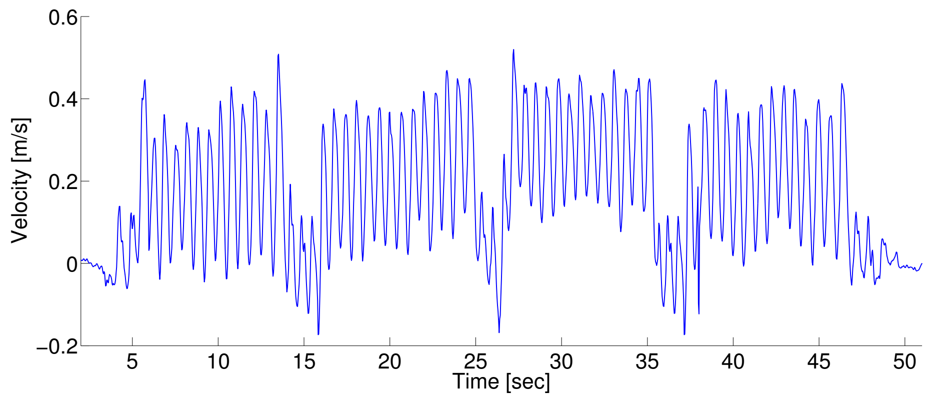


(c) Case 3: missed one landing.

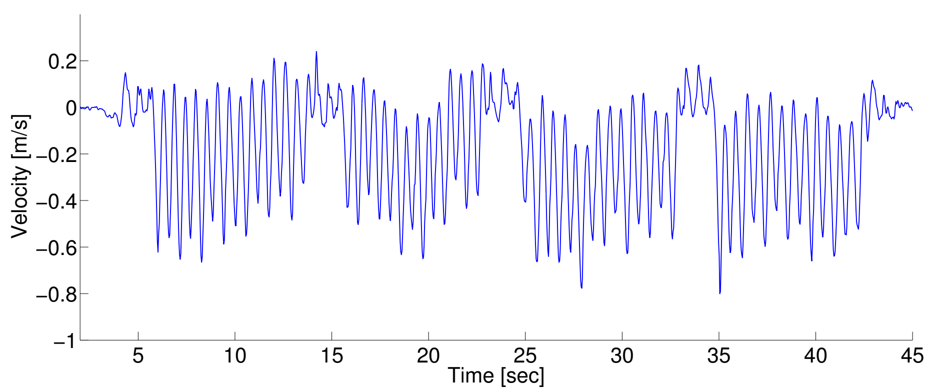
Figure 5.7: Three landing detection cases.

that the seeming discontinuity in the heading from 20° to 330° at 37 sec is in fact a steady change from 20° to −30°, wrapping around. The two rectangles in the figure highlight two incorrectly identified landings being removed because the heading did not change during the period.

This heading-based verification of landings makes it unlikely that our algorithm produces false positives. If the acceleration-based landing detection misses a landing to begin with, however, the heading information does not help recover it. Figure 5.7(c) shows this case. Therefore, our algorithm produces a conservative estimate of the number of landings.



(a) Walking up



(b) Walking down

Figure 5.8: Velocity measurements of a user walking on a stairway.

We determine whether a user is moving up or down by comparing the average vertical velocity on steps and landings. Figure 5.8 shows the vertical velocity measurements when a user walks up and down two floors passing four landings. The figure clearly illustrates the difference in the velocity patterns between the up and down cases. We determine that the user is ascending if the velocity on steps is higher than the velocity on landings, and vice versa. In theory, the average vertical velocity should be zero on landings, positive when the user walking up steps, and negative when walking down. But in practice, the velocity values can shift due to the noise and errors that have been introduced while extracting vertical acceleration and subtracting g.

The stairway module returns a relative location which is the number of floors the user has traveled from the initial floor. Like the elevator module, the stairway module relies on the information from the infrastructure monitor to get the initial anchor location. The infrastructure monitor also provides the number of landings between each pair of floors. There are typically two landings per floor but the number can vary depending on the design of a building. In some buildings, for example, there are more landings between the lobby and the second floor.

5.3.3 Escalator Module

The escalator module combines the elements of both elevator and stairway modules as shown in Figure 5.9. The escalator module uses double integration like the elevator module. However, the user's movement on an escalator is not vertical, so we use the gyroscope measurements to extract the vertical component from the measured acceleration, as we did in the stairway module.

5.3.4 Activity Manager

The activity manager classifies a user's movements as one of the following activities: elevator riding, walking, and standing. The classification is based on the user's vertical acceleration. The current version of the activity manager does not identify escalator riding. Vertical acceleration does not work well for escalators because of the complexity of a user's movement. We are investigating other ways to detect escalators, such as the technique of magnetic field variance proposed by Wang *et al.* [115].

Figure 5.10(a) depicts the pattern of a user's vertical acceleration when she is riding an elevator. The elevator starts with zero acceleration (1), accelerates to a steady velocity (2),

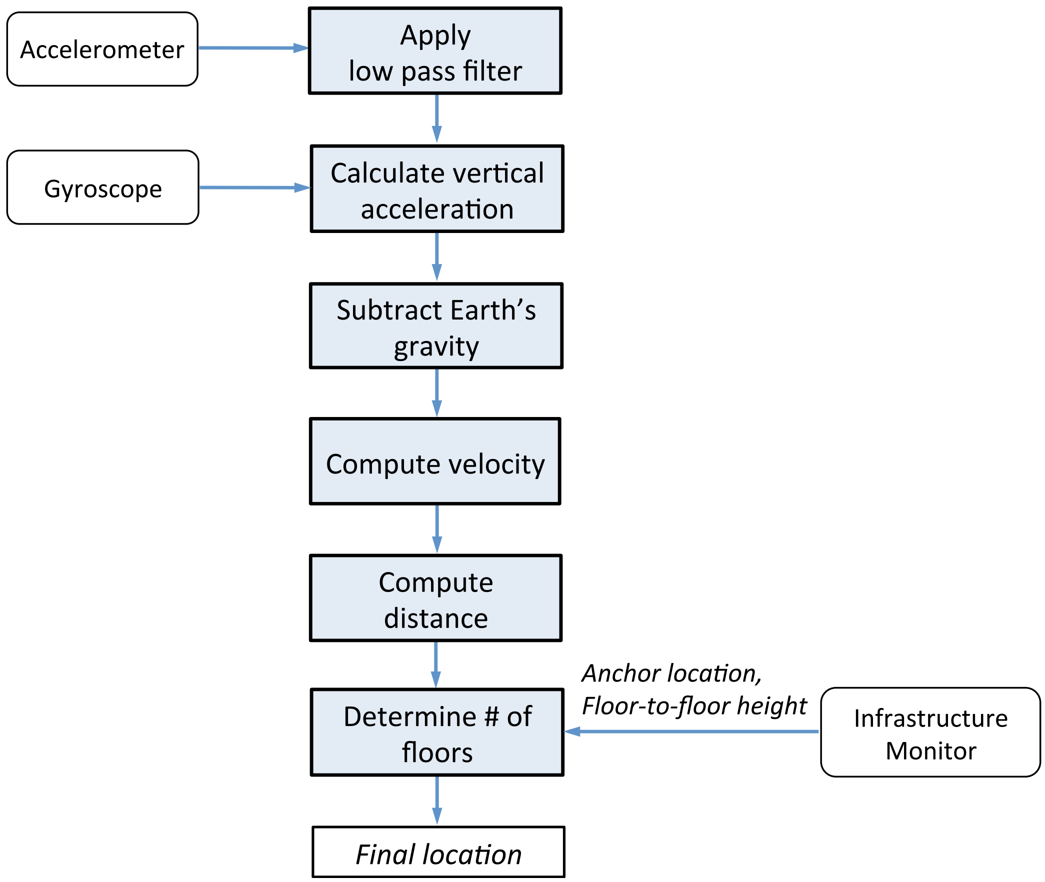
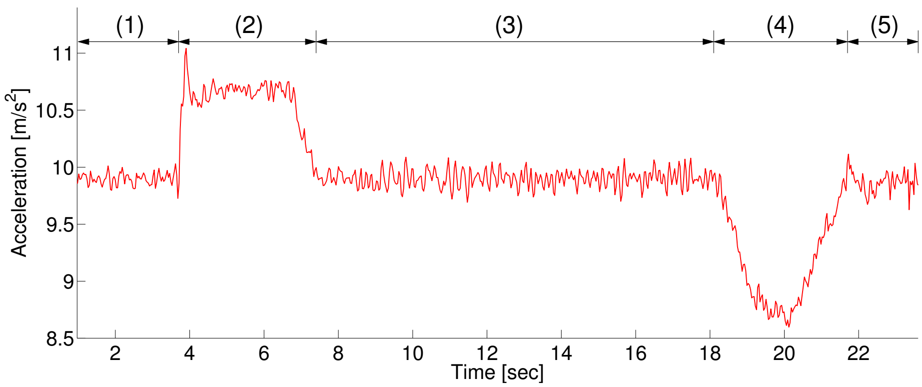


Figure 5.9: Overview of the escalator module.

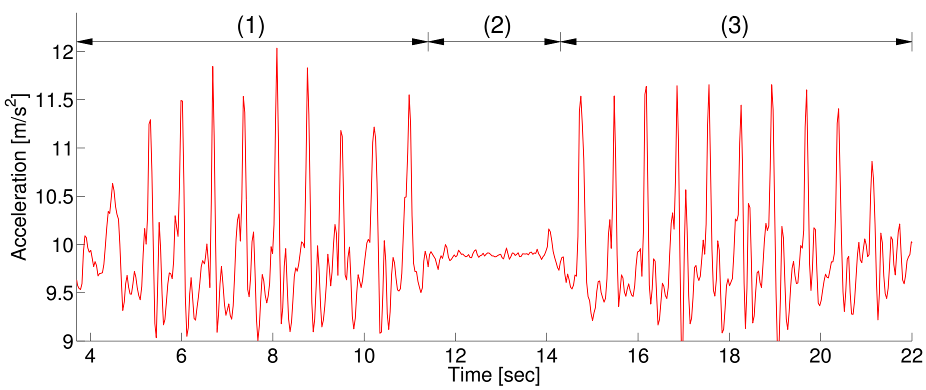
moves at a constant speed (3), decelerates (4), and stops (5). If there are multiple stops, these five intervals will be repeated for each stop. When the activity manager detects this pattern, the sensor measurements during that period are passed to the elevator module.

Figure 5.10(b) shows a user's vertical acceleration when the user walks a few steps, stops for a bit, and then resumes walking. The activity manager detects human steps by identifying local extrema of amplitude in vertical acceleration. One step contains exactly one maximum and one minimum in a short time interval. The period (1) and (3) in Figure 5.10(b) contain human steps so they are walking periods.

The period (2) in Figure 5.10(b), where the vertical acceleration is under a threshold, is classified as a standing period. The activity manager uses the standing period to partition the sensor measurements.



(a) Elevator



(b) Walking

Figure 5.10: Acceleration of elevator and walking.

Each walking period, separated by the standing periods, is passed to the stairway module. A single walking period can be either walking on the stairway (stairway walking) or walking on the same floor (same-floor walking). Ideally, the activity manager should detect all periods of same-floor walking, and filter them out, so that they do not get passed to the stairway module. The current version of our activity manager does not implement this filtering. Thus, the stairway module needs to handle not only stairway walking (as described in detail in Section 5.3.2), but also same-floor walking.

To detect same-floor walking, the stairway module calculates the total distance that a user has traveled vertically during the walking period using double integration, and see if the vertical distance turns out to be close to zero. If so, the walking period is considered to be an instance of same-floor walking.

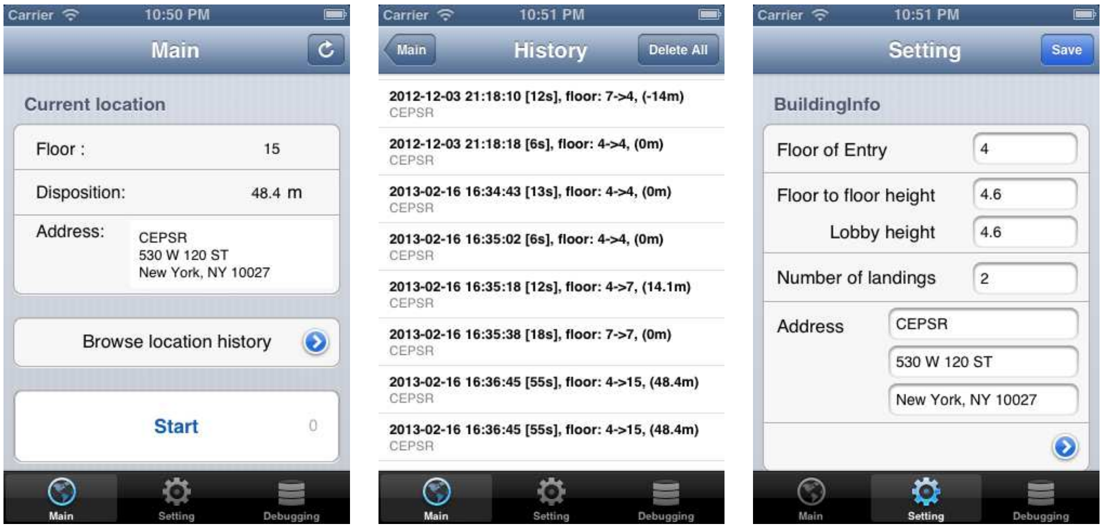
Normally, a single walking period does not contain both stairway and same-floor walking. A user would typically stop to open a door to the stairway, producing a standing period which separates them into two walking periods. It is possible, however, that both stairway and same-floor walking are included in a single walking period if the user avoids stoppage in the middle. In this case, the same-floor walking portion will be detected as a landing by the stairway module, assuming that the user has made a significant change in the heading during the same-floor walking. The only case that the stairway module will not be able to handle is the one where the user walks a long straight corridor between two flights of stairs without stoppage. Our stairway module will not identify this as a landing due to the lack of any heading change.

5.4 Implementation

5.4.1 Hardware Platform

For the implementation and evaluation, we used the Apple iPhone 4 and 4S, running iOS version 6. The iPhone 4 contains an accelerometer, a gyroscope, and a magnetometer. The accelerometer in iPhone 4 can measure acceleration from $-2\,g$ to $+2\,g$, where $1\,g$ is $9.8\,\mathrm{m/s^2}$ [101]. The sampling rate can be adjusted from 0.5 Hz to 1 kHz. We used 30 Hz for our measurements. The gyroscope measures angular velocity from -250 degree/sec to +250 degree/sec [101]. We also read the gyroscope at 30 Hz. The magnetometer is a three-axis electronic compass manufactured by Asahi Kasei [53]. According to the specification from the manufacturer, the measurement range is $\pm$1,200 μT.

Most smartphones based on Google's Android platform are also equipped with the same set of sensors with comparable specifications. Both iOS and Android offer APIs to access the sensors.



(a) (b) (c)

Figure 5.11: Screenshot of the prototype on iPhone.

5.4.2 Data Collection from Sensor Array

In our current prototype[1], an application running on the iPhone collects data from the sensor array. Figure 5.11 shows the screenshot. Table 5.2 provides the list of data types we collect on iPhone. The measurements from the accelerometer and gyroscope in iPhone can be accessed using the Core Motion framework in iOS. The Core Motion framework provides APIs to retrieve the raw data such as the timestamp and three-axis accelerations shown in Table 5.2. The framework also provides processed motion data, such as *attitude*, which is derived from both the accelerometer and gyroscope. Attitude refers to the spatial orientation of the device with respect to the world coordinates, and can be obtained either as a rotation matrix or as a quaternion. We use the rotation matrix in our implementation of the stairway and escalator modules.

The heading information from the magnetometer can be accessed using the Core Location framework in iOS. The framework provides two headings: magnetic heading and true heading. Magnetic heading points to the magnetic north pole, and true heading points to the geographic north pole. We use the magnetic heading in our implementation. Both types of heading will satisfy our need to detect a user turning around on landings, but

[1] The source code of our prototype is available at https://github.com/ws2131/indoor-location.

Data	Unit	Source
Timestamp	ms	System clock
X-axis acceleration	g	Accelerometer
Y-axis acceleration	g	Accelerometer
Z-axis acceleration	g	Accelerometer
Rotation matrix	N/A	Accel. & Gyro.
Heading	degree	Magnetometer
Latitude & longitude	degree	GPS

Table 5.2: Data types collected on iPhone.

using magnetic heading avoids additional processing to calculate the true heading from the current location, which may consume more energy.

We collect GPS traces outdoors. The Core Location framework provides an API to obtain the device's location. Normally the framework determines the locations using various sources including GPS, Wi-Fi, and cellular network, but a flag can be passed to indicate that we are only interested in GPS locations.

In order to improve the accuracy of GPS in urban canyons, we also collect the *course*, which refers to the direction of the device extrapolated from its GPS trajectory. We compare the course with the heading from the magnetometer, and reject the GPS data if the difference is too large.

5.4.3 Data Analysis

Our iPhone application also includes the implementation of the activity manager and the analysis modules.

At the early stage of the project, we prototyped the activity manager and the analysis modules in MATLAB and tested our algorithms using collected sensor data. After verifying our algorithms, we have ported the MATLAB code to Objective-C so collecting sensor data and analyzing them can be done on the user's smartphone. It is desirable to run all analysis locally on the user's device whenever possible, so that the user's privacy is preserved as much as possible.

5.4.4 Data Collection from Infrastructure

In this section, we describe the infrastructure monitor running on the user's smartphone, the location beacons deployed in the building, and the central building database server.

We chose Bluetooth technology for location beacons because Bluetooth is available on most smartphones. The infrastructure monitor and the beacon communicate using the service discovery protocol (SDP) [41]. SDP allows Bluetooth devices to discover available services and their characteristics without initiating a pairing process.

Currently, iOS does not provide APIs for Bluetooth communication. We implemented the infrastructure monitor using BTstack [4], an open source Bluetooth stack for iOS. Installing BTstack requires jailbreaking the iPhone.

Figure 5.12: Foil-wrapped Mac mini as Bluetooth beacon.

We used a Mac mini computer wrapped in aluminum foil to prototype a Bluetooth beacon as shown in Figure 5.12. The foil wrapper decreases the Bluetooth signal strength so that it does not reach the adjacent floors. The Bluetooth beacon is written as a Java application using BlueCove [2], an open source Java library for Bluetooth.

The beacon interacts with the smartphone's infrastructure monitor in the following

Data item	Default
Floor of entry	1
Number of landings between floors	2
Floor-to-floor height	3.5 m

Table 5.3: Predefined default values for building information.

sequence. First, the infrastructure monitor scans for nearby Bluetooth devices by sending periodic inquiry messages. Second, the infrastructure monitor sends an SDP ServiceSearchAttribute request to all the discovered Bluetooth devices. The request includes a unique identifier defined for location beacon service (UUID 0x6666), so the request is ignored by all devices that are not location beacons. Lastly, the infrastructure monitor receives an SDP response from a location beacon. An SDP response contains the building's address, the floor where the beacon is located, and for each pair of floors, the height and the number of landings.

The infrastructure monitor falls back on a central building database server when a building is not equipped with location beacons. While a user stays outdoors, the infrastructure monitor tracks the user's location using GPS. When GPS signal is lost, the infrastructure monitor assumes that the user has entered a building, and sends the last known GPS coordinates to the building database server. The building database server finds the nearest entrance from the user's last GPS location, and returns the same information that the location beacon returns. Figure 5.13 is the web admin interface to the building database server that we have built for the evaluation, displaying the information describing a building in Columbia University campus. We implemented the building database server as a web application running on a Apache Tomcat. The infrastructure monitor and the building database server communicate using the HTTP GET method.

Table 5.3 shows the default values for building information when the infrastructure monitor fails to obtain the information specific to a building. The default value for the floor-to-floor height is the average value for commercial and residential buildings reported by the Council on Tall Buildings and Urban Habitat [50].

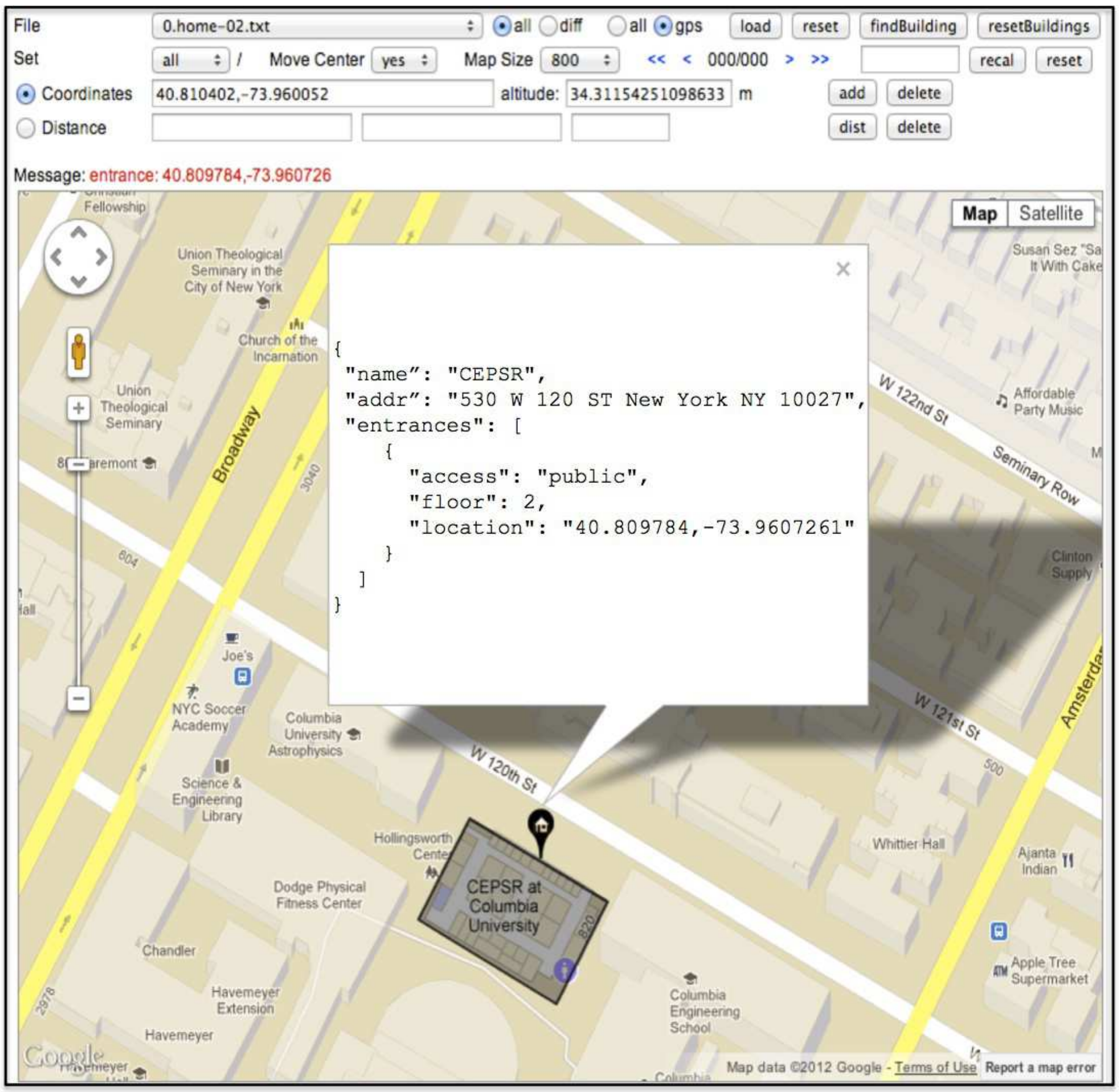


Figure 5.13: Web interface to the building database server.

Building name	Floor number	Reference floor height	Average error (and stdev.)	Error-to-height ratio
CEPSR	6th	4.65 m	0.08 m (0.05 m)	1.6%
Mudd	1st	3.67 m	0.06 m (0.04 m)	1.7%
Pupin	8th	3.48 m	0.09 m (0.08 m)	2.7%

Table 5.4: Errors in one floor distance calculated by elevator module.

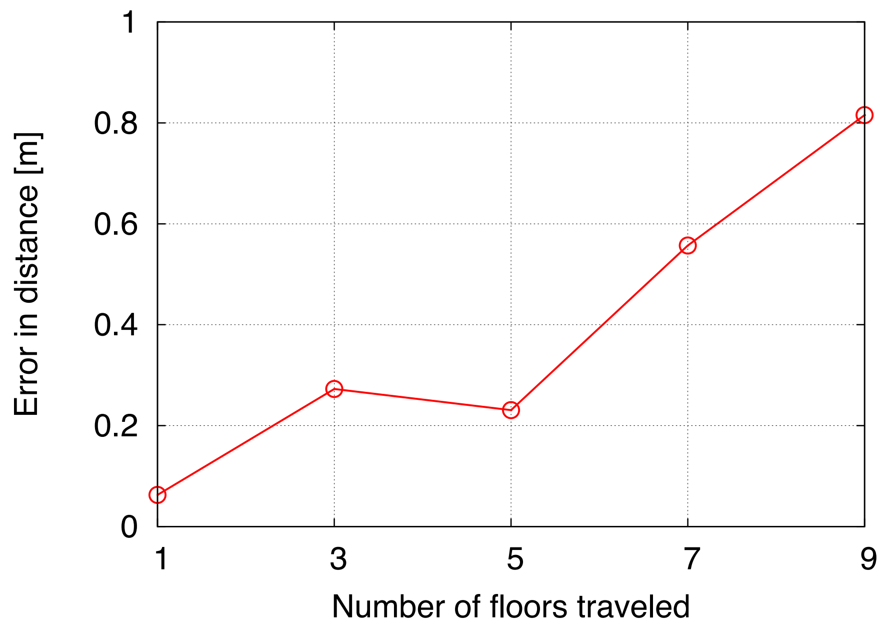
5.5 Evaluation

First, we evaluate the algorithms of our elevator, stairway, and escalator modules individually. The individual evaluation scenarios assume that the activity manager correctly identifies the test subject's activity and selects the proper analysis module. Then, we present a combined case where the test subject's travel involves multiple types of movements including riding an elevator, walking on a stairway, and walking around on the same floor, which are all detected by the activity manager.

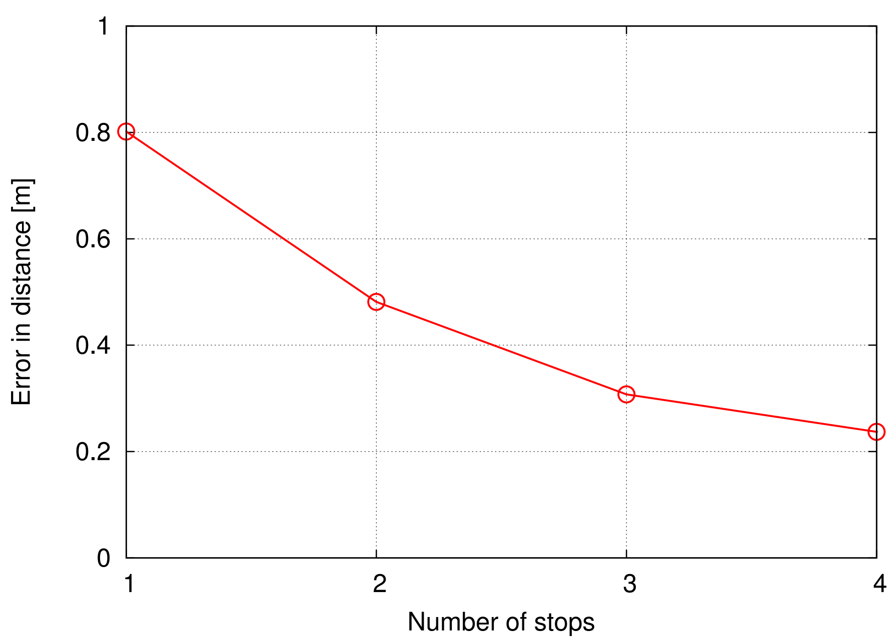
5.5.1 Elevator Module

We evaluated the elevator module in three different research and classroom buildings at Columbia University: CEPSR, Mudd, and Pupin. They have 10, 15, and 13 floors, respectively. Table 5.4 shows the reference floor-to-floor height of each building, which we measured using a tape measure, followed by the error of the result from the elevator module. The error is the difference between the reference height and the distance calculated by the elevator module when the test subject moves one specific floor in an elevator in each building. The error is an average of ten trials, five moving up and five moving down.

Errors are small in all three buildings, indicating that the elevator module can provide accurate vertical location up to a reasonable number of floors. We can extend the range by strategically deploying location beacons. For example, in the Pupin case in Table 5.4, the error is under 3%, so the elevator module will be accurate up to about 15 floors. Thus, location beacons can be deployed conservatively in every 10 floors to cover the entire building.



(a) Traveling 1, 3, 5, 7, 9 floors without stopping.



(b) Traveling 9 floors with increasing number of stops.

Figure 5.14: Distance errors of elevator module measured in Mudd building.

Figure 5.14(a) shows distance errors from the elevator module as we increase the number of floors traveled in an elevator without stopping. The graph shows that the errors accumulate as the elevator travels farther. The error of 0.82 m when the test subject traveled nine floors is about 22% of the floor-to-floor height, which is well within the margin of error for accurately determining the destination floor.

Figure 5.14(b) plots the distance errors of traveling nine floors in an elevator as we vary the number of stops that the test subject has made during the travel. The graph shows that the error decreases as the subject makes more stops. This shows the effectiveness of applying ZUPT in the distance calculation. At each stop, ZUPT eliminates accumulated errors by removing residual velocity. Therefore, if the elevator makes stops during the trip, the elevator module's distance estimation becomes much more accurate, extending the upper bound of the elevator module's distance limitation.

5.5.2 Stairway Module

5.5.2.1 Evaluation with Single Test Subject

We evaluated the stairway module in two buildings. One was an office building and the other was a residential building. Both buildings have two landings between each pair of floors. Figure 5.15 shows the two stairways.

Figure 5.16 shows our stairway module measurements. In each building, we performed 50 trials of walking four floors. The graph compares the landing counting results with and without our heading-based correction algorithm described in Section 5.3.2. Our heading-based correction was able to eliminate all miscounted landings in both buildings, producing the correct landing count in all 50 trials. Without the heading-based correction, only 44 and 32 trials produced the correct landing count in the residential and the office building, respectively. The graph shows the number of trials that produced one or more miscounted landings in each building. For instance, two trials in the office building miscounted four landings, which would have resulted in an error of two floors, if the heading-based correction had not been applied.

Figure 5.16 also shows that, without the heading-based correction, the stairway module performs better in the residential building than in the office building. We attribute this

(a) Office building. (b) Residential building.

Figure 5.15: Stairways used for evaluation.

Building type	Number of steps between 2 floors	Height of a step
Office	27	16 cm
Residence	16	19.5 cm

Table 5.5: Stair specification in the office and residential buildings.

difference to the steeper stairs in the residential building as shown in Table 5.5. The difference in the vertical acceleration between steps and landings is more pronounced on the steeper stairs. In general, our landing detection works better when the amplitude difference in the acceleration is pronounced. This is also in line with our observation that the waveforms are generally cleaner in the walking-down cases than in the walking-up cases. Human steps are typically a bit bouncier when walking down.

We note that, in all trials in Figure 5.16, the test subject moved at a normal walking speed. If a person walks very fast or very slowly, the amplitude difference of the accelerometer reading between steps and landings is much less pronounced. We can address this issue by giving more weight to the heading information from the magnetometer. In the extreme

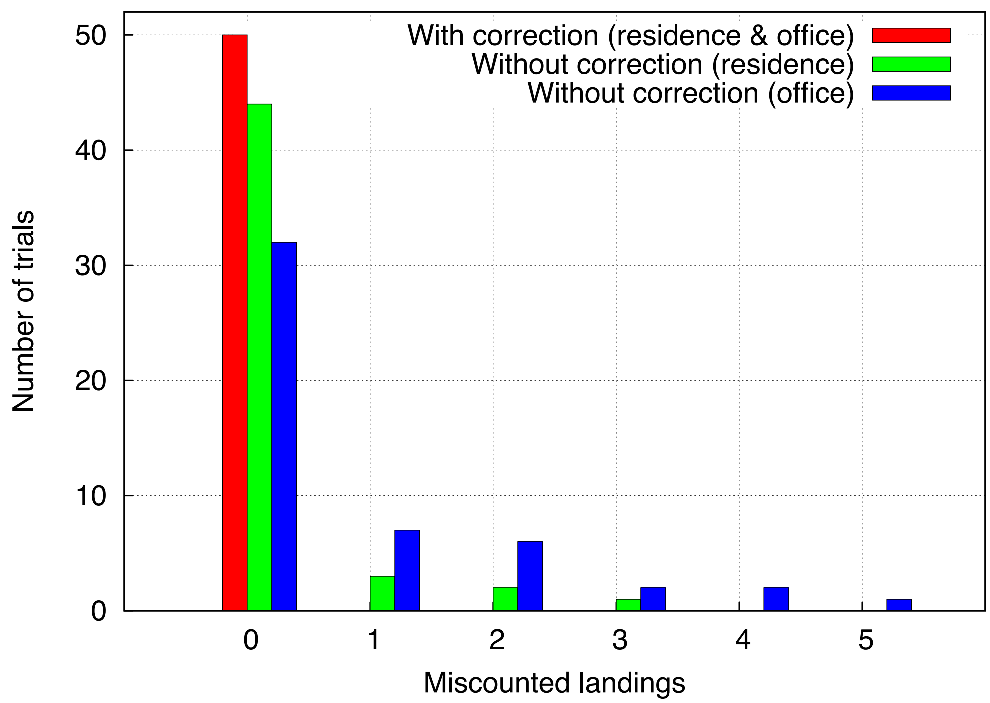


Figure 5.16: Stairway module measurements of 50 trials of walking four floors. The heading-based correction eliminated all miscounted landings in both buildings.

case, we can reverse the roles of the accelerometer and the magnetometer, i.e., instead of using the magnetometer to make adjustments to the landings identified by the accelerometer, we can use the magnetometer first to identify landings. The relative weights of the two sensors can be dynamically determined depending on how pronounced the amplitude difference is.

The iPhone's magnetometer readings, however, often showed large fluctuations in our experiments even when the test subject did not change direction. For this reason, we chose to use the magnetometer conservatively, i.e., only for correcting false positives. In order to see the effectiveness of the magnetometer-first approach, we conducted the same experiment with the test subject walking very fast and very slowly, and selected the measurements that did not contain incorrect magnetometer readings. We confirmed that the magnetometer-first approach, when the magnetometer readings are reliable, can cover a wider range of human walking speed.

Test subject	Number of correct results	Number of incorrect results
Subject 1	4	0
Subject 2	4	0
Subject 3	4	0
Subject 4	4	0
Subject 5	3	1
Total	19	1

Table 5.6: Stairway module measurements by 5 different test subjects.

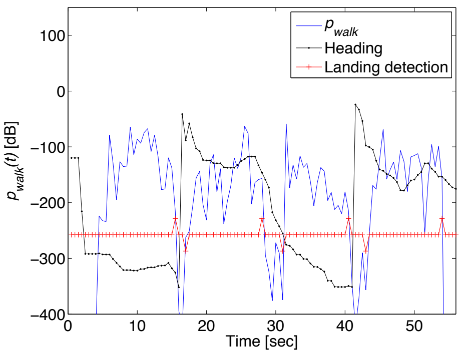
5.5.2.2 Evaluation with Multiple Test Subjects

In this evaluation, five different test subjects walked on the same stairway in the office building used in the previous evaluation. Each test subject performed two trials of walking up two floors and two trials of walking down two floors, resulting in 20 trials in total. All test subjects were asked to walk with normal gait.

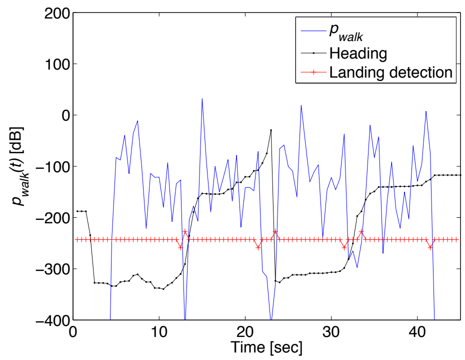
Table 5.6 and Figure 5.17 show our evaluation result. In 19 out of 20 trials, the stairway module resulted in the correct number of landings.

In one trial, however, the stairway module returned 3 landings instead of 4 landings. Figure 5.17(f) shows this case. This is the case where the acceleration-based landing detection misses a landing, but the heading-based correction does not correct it as we described in Section 5.3.2. If we give more weight to the magnetometer measurement, the heading-based correction can recover this missed landing.

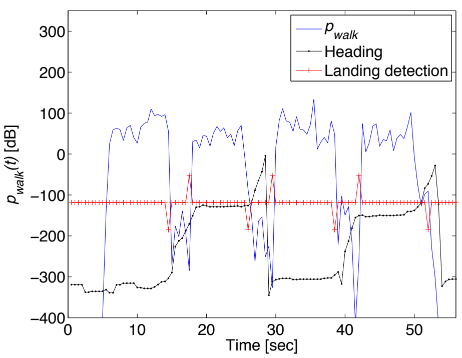
The evaluation result confirms that our stairway module can accurately determine a person's movement on the stairway without any personalized information. Our landing counting algorithm exploits the commonality of people's walking patterns on stairways. It detects the difference between steps and landings by observing vertical acceleration and heading information.



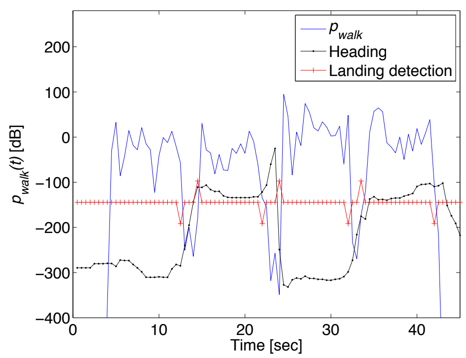
(a) Test subject 1: accurate result.



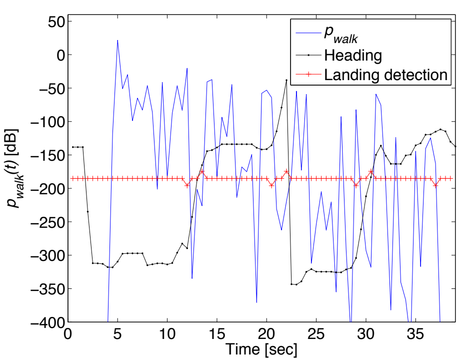
(b) Test subject 2: accurate result.



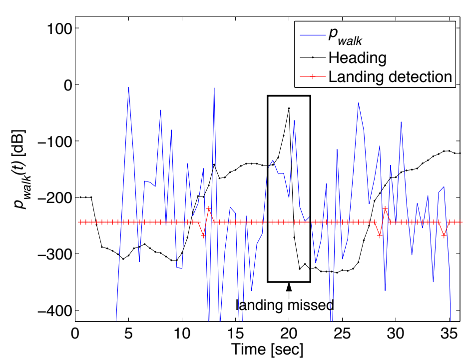
(c) Test subject 3: accurate result.



(d) Test subject 4: accurate result.



(e) Test subject 5: accurate result.



(f) Test subject 5: error case.

Figure 5.17: Landing detection results from multiple test subjects.

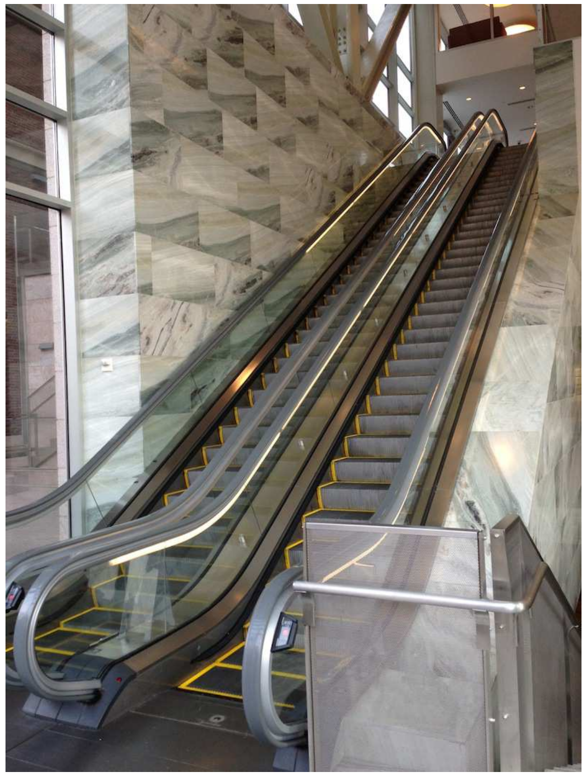


Figure 5.18: Escalator in Northwest Corner Building used for evaluation.

5.5.3 Escalator Module

We evaluated the escalator module in Northwest Corner Building at Columbia University where the escalator connects the second and the fourth floor as shown in Figure 5.18. We used a tape measure to obtain the reference height between the two floors: 7.3 m.

We measure two different cases. In one case, the test subject stood still while riding the escalator. In the other case, the subject was also walking during the ride. For each case, the subject took the escalator 50 times. Out of the 50 walking trials, the subject walked during the entire ride 20 times, only the first half 15 times, and only the second half 15 times. We do not show these cases separately because there was no significant difference between them.

Table 5.7 shows the reference floor-to-floor height between the two floors and the error of the result from the escalator module. In both cases, the escalator module accurately determines the number of floors the test subject has traveled. Compared to the elevator

Case	Reference height (2 floors)	Average error	Error-to-height ratio
Without walking	7.3 m	0.99 m	13.6%
With walking	7.3 m	1.34 m	18.4%

Table 5.7: Errors in two floor distance calculated by escalator module.

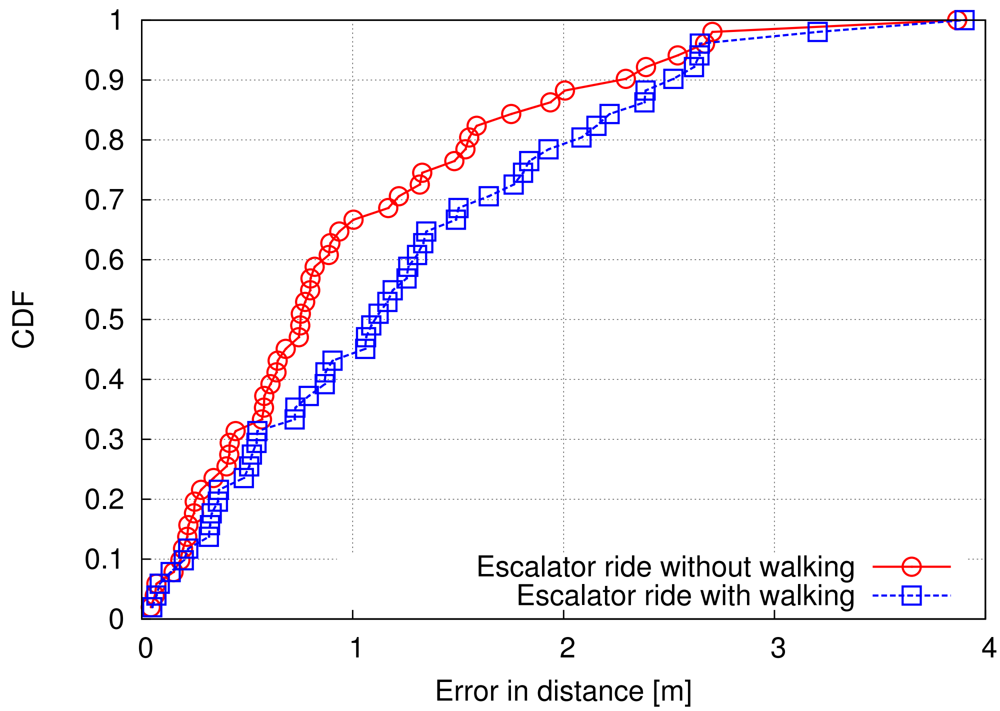


Figure 5.19: CDF of error in the distance measurement by the escalator module.

module, the average errors from the escalator module are large. For example, the error-to-height ratio in the walking case is 6.8 times bigger than the elevator case in the Pupin building shown in Table 5.4–18.4% versus 2.7%. Considering the test subject travels two floors in the escalator case, the escalator module will be accurate up to 5–7 floors.

Figure 5.19 shows the CDFs of the error in the distance reported by the escalator module. In the case without walking, the 50% and 80% of the results are within 0.75 m and 1.5 m, respectively. In the case with walking, 50% and 80% are within 1.1 m and 2.1 m, respectively. As we expected, walking on the escalator generates more noise in the vertical acceleration, causing larger errors in the distance calculation.

The 2.1 m error–the 80th percentile in the case with walking–is about 29% of the reference height between the 2nd and 4th floors. Again, this error-to-height ratio is large compared to the elevator cases. This is because, unlike an elevator where the movement starts and ends at a standstill, a person steps on and off an escalator that is constantly moving, making it harder to separate the acceleration purely due to the escalator.

However, an escalator typically covers 1-2 floors, so the error in the distance measurement will still not cause an error in determining the number of floors. If a person rides a series of escalators one after another, the error from one ride will not carry over to the next one because we can apply ZUPT at landings.

5.5.4 Combined Case

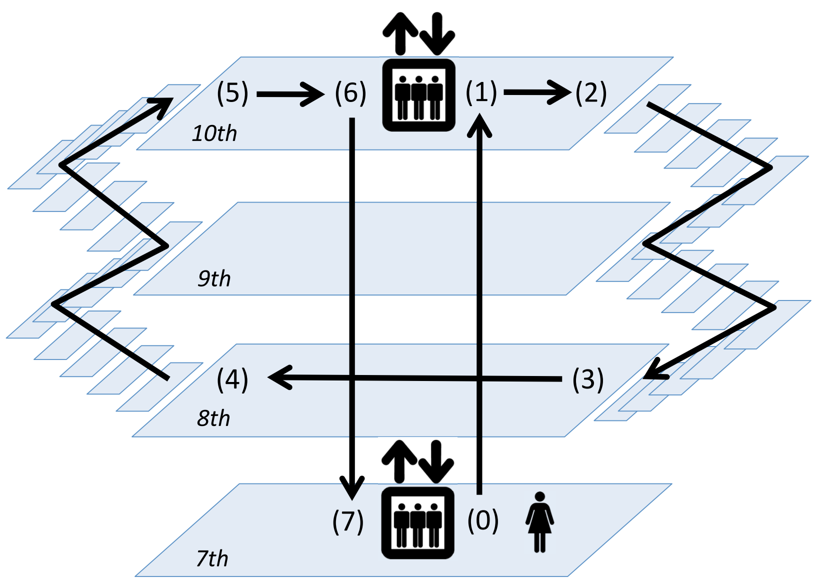


Figure 5.20: A person's travel between 7th and 10th floor including elevator riding, stairway walking, and same-floor walking.

Figure 5.20 shows our evaluation scenario of a case involving multiple types of movement: elevator up/down, stairway up/down, and same-floor walking. The activity manager detects each activity and sends the sensor measurements to the corresponding modules.

The travel scenario consists of seven intervals. First, the test subject takes an elevator

# of trials	(0)	(1)	(2)	(3)	(4)	(5)	(6)	(7)
8x	7	10	10	8	8	10	10	7
2x	7	10	10	8	**8.5**	**10.5**	**10.5**	**7.5**

Table 5.8: Floor levels reported at each stage.

on the 7th floor (0), and gets off on the 10th floor (1). The subject then walks on the 10th floor to the door to the stairway (2). She walks down the stairway from the 10th to the 8th floor (3). On the 8th floor, she comes out of the stairway, walks to the other side of the floor to enter another stairway (4). She walks up the stairway from the 8th back to the 10th floor (5). On the 10th floor, she comes out of the stairway, and walks to the elevator entrance in the middle of the floor (6). Finally, she takes the elevator on the 10th floor and goes down to the 7th floor, back to where she started (7).

Table 5.8 shows the floor levels reported at each stage in Figure 5.20 when we repeated the travel ten times. In eight out of the ten trials, the correct floor was reported at every stage. In the two remaining trials, the stairway module failed to recognize the walking period between (3) and (4) as same-floor walking. The module incorrectly reported one landing instead of zero. This caused an error of $\frac{1}{2}$ floor in the subsequent stages.

The evaluation results show that our system can successfully track a person's complex movements in general. At the same time, the errors in the two trials reveal the weakness in our system: it may fail to distinguish between stairway walking and same-floor walking. This is indeed a difficult problem that remains as an active research area. At the time of this writing, other activity recognition systems have similar success rates [123].

5.5.5 Infrastructure Monitor

When the infrastructure monitor communicates with Bluetooth beacons, the range of the signal should be far enough so that a person's device can receive the information about the entrance. On the other hand, the signal should not reach devices at a different floor. If it did, the devices would receive wrong information. For example, if a device starts with a wrong anchor location, all subsequent calculations would be invalid.

Distance	Without foil wrapping	With foil wrapping
0 m	100%	100%
3 m	100%	100%
−1 floor	60%	0%
+1 floor	100%	0%

Table 5.9: Success rates of Bluetooth communications.

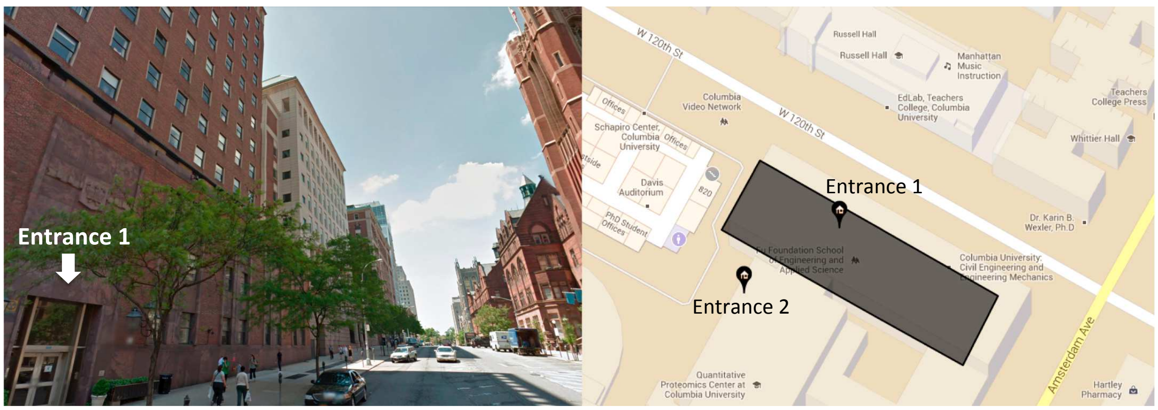


Figure 5.21: Experiment setting for GPS-based entrance detection.

Table 5.9 compares the success rates of Bluetooth SDP transactions between the infrastructure monitor and the Bluetooth beacon with and without foil wrapping. We tested whether the infrastructure monitor was able to communicate with the beacon at two locations on the same floor and at two adjacent floors. The locations on the different floors were exactly above and below the beacon's location. The results show that the foil wrapping can decrease the strength of the Bluetooth signal effectively.

In addition to the entrance beacon, we evaluated our GPS-based entrance detection mechanism at a building in Columbia University. Figure 5.21 shows the picture and the map around the test building. A test subject walks along the street toward the building and enters through the street level entrance marked as Entrance 1 in Figure 5.21. The building has another entrance at campus level (Entrance 2) which is 40 m away from Entrance 1. Entrance 1 and Entrance 2 are on the first and fourth floor, respectively. As described in Section 5.2.4, the infrastructure monitor sends the last known GPS coordinates to the building database server when GPS signal is lost, and then the building database server

Success/Total	Average error	Max error	Min error	Stdev.
19/20	13.19 m	27.03 m	2.42 m	7.68 m

Table 5.10: Result of GPS-based entrance detection.

returns the information of the building that has the nearest entrance from the point.

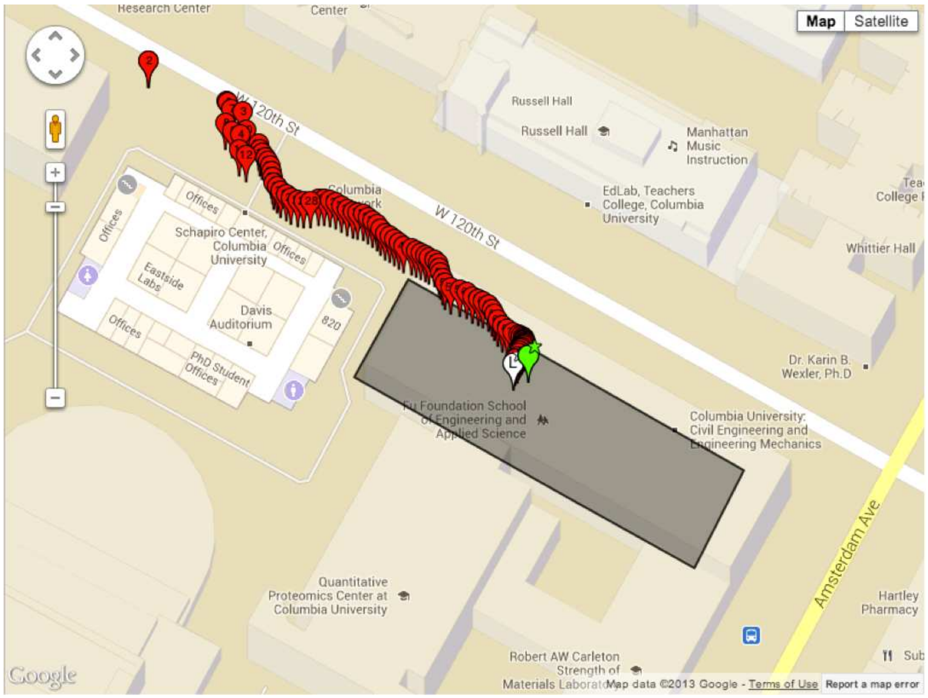
Table 5.10 shows GPS-based entrance detection works well in our test case. In 19 out of 20 trials, the infrastructure monitor can detect the correct entrance in cooperation with the building database server. However, the average error of 13.19 m shows the limitation of our GPS-based approach. In a dense urban area, a distance between two buildings' entrances can be less than 10 m, and GPS location would not be accurate enough to distinguish them. For this reason, our infrastructure monitor uses the GPS-based approach as a fallback of the entrance beacon.

Figure 5.22 describes one success and one failure case in detail. In the success case, the last GPS point is 2.42 m away from the actual entrance so the infrastructure monitor detects successfully the correct entrance. In the failure case, however, the GPS error of 21.2 m leads the infrastructure monitor to detect Entrance 2 as the initial anchor point instead of Entrance 1.

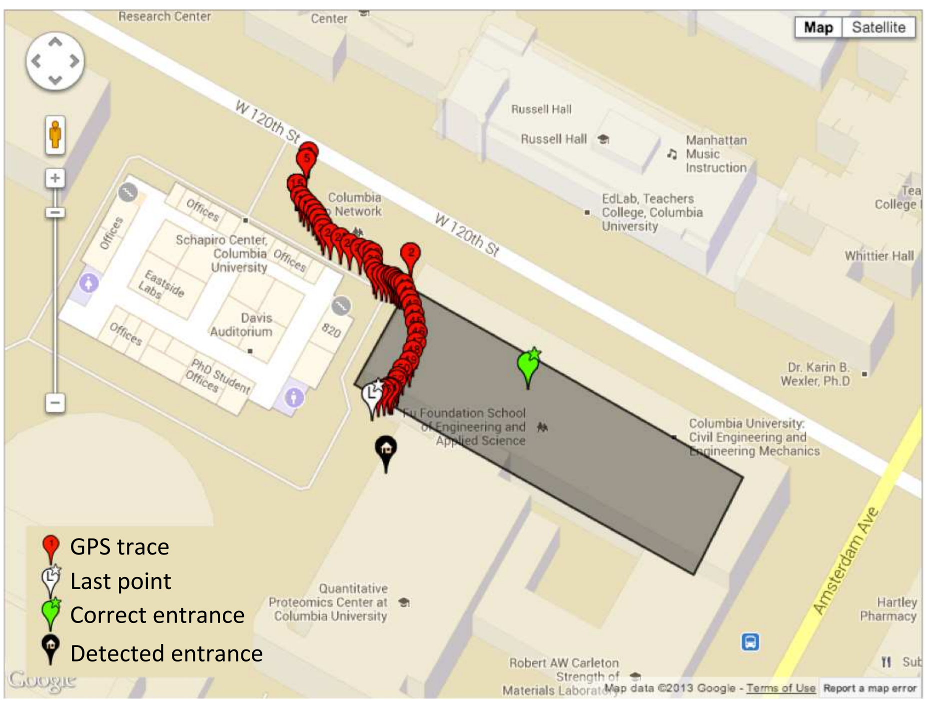
5.5.6 Energy Consumption

Since our indoor positioning system runs on a mobile phone, it should not consume too much energy for long battery lifespan. Although inertial sensors are known as low-power sensors, our system keeps capturing the sensor measurements at the 30 Hz sampling rate while a person is in the building. This intensive use of inertial sensors may consume a large amount of the energy from the smartphone's battery and make our system infeasible for the real deployment.

Figure 5.23 compares the energy consumption of our prototype system with music and video playback for one hour. We first measure the energy consumption of the idle state when the phone is awake but not running any applications. Then, we subtract it from the energy measurements of our prototype system and music playback to get the energy consumption purely caused by the main tasks–processing data from inertial sensors and



(a) Success case.



(b) Failure case.

Figure 5.22: Results of GPS-based entrance detection.

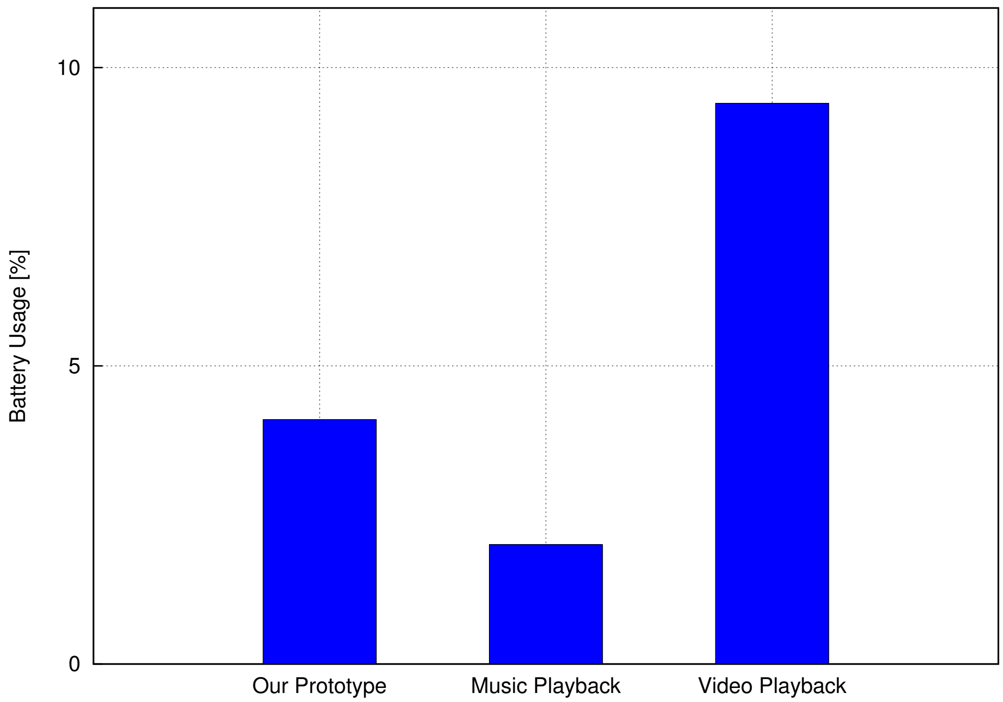


Figure 5.23: Energy consumption of our prototype, music playback, and video playback for one hour.

playing music, excluding the display-related energy consumption.[2] According to [46], the display-related subsystems consume the largest portion of the energy in the idle state. We measure the energy consumption by comparing the battery percentage indicator shown in the iPhone's status bar before and after running each application for one hour. We turned on the airplane mode to disable any radio transmission during the measurement and also turned off the auto-lock function to prevent the application from being suspended in the background state. All results are the average of 10 iterations.

Our prototype, music playback, and video playback take 4.1%, 2%, and 9.4% of the iPhone's battery capacity, respectively. Since our system consumes 4.1% of the battery

[2] We envision that the indoor positioning system will run in the background state as one of the operating system's services in the real deployment. However, due to the restriction of the current version of iOS, we implemented our prototype system as an application running in the foreground state. Applications on iOS devices cannot access the sensor measurements in the background state.

Sampling rate	Battery usage	Distance error	Landing counting error
15 Hz	3.3%	0.34 m	1
30 Hz	4.1%	0.06 m	0
60 Hz	5.3%	0.08 m	0

Table 5.11: Trade-off between energy consumption and location accuracy.

capacity while running in an hour, it can run up to about 24 hours until it drains the battery completely. The graph shows that the energy consumption of our indoor positioning system is not large, but may be unacceptable for the real deployment.

Table 5.11 shows changes of energy consumption and location error when we capture the sensor measurements at different sampling rates. We measured the error from the elevator and stairway module when a test subject traveled one floor. The table shows that we can reduce the energy consumption of our system if we use a lower sampling rate. For example, if we decrease the sampling rate from 30 Hz to 15 Hz, our prototype can run about six more hours. However, there is a trade-off between the energy consumption and location accuracy. If we decrease the sampling rate too much, the reported location becomes less accurate. In addition, increasing the sampling rate does not necessarily improve location accuracy after a certain point.

One way to reduce the energy consumption of our system is turning on the inertial sensors for a short period of time or dynamically adjusting the sampling rate depending on the person's activity [83]. If a person stands still, we can decrease the sampling rate while not degrading location accuracy. Moreover, there is a proposal for energy efficient location sensing on mobile phones using a dedicated low-power sensing processor [88]. The Moto X smartphone from Google has one low-power processing core that only monitors sensors in order to capture the phone's movement all the time [75]. Energy consumption for continuous sensing right now is quite high and we realize that this may be unacceptable to real users. However, with technologies coming up, e.g., low-power coprocessors, or if these techniques get deployed on custom devices, then this issue may become less of a concern in the future.

5.6 Related work

Indoor positioning systems can be put into two categories: infrastructure-based and dead reckoning systems. Infrastructure-based systems rely on infrastructure support such as sensors or beacons deployed in buildings. Sensors detect signals that are emitted by user devices, and beacons transmit signals that are received by user devices. Dead reckoning systems do not rely on any external entity. Instead, inertial sensors in user devices are used to keep track of users' movements indoors. There are hybrid systems that combine elements from both categories. Our solution is an example of a hybrid system.

5.6.1 Infrastructure-based Systems

5.6.1.1 Proximity Detection

Proximity detection based systems locate users by detecting signals which emitted by user devices. The signals usually carry unique IDs for the devices.

Active Badge [118] uses infrared (IR) signals. Since IR signals cannot penetrate walls, there is no interference among sensors in different rooms or floors. Active Badge can determine the room that a user is located with high precision, but requires the user to wear the badge. The location accuracy is around 6 m, which is the range of IR signals used in the system.

Many systems use Bluetooth technology for location detection [30,39] because Bluetooth is inexpensive and ubiquitous. Bluetooth's longer range (about 10 m with Class 3) and its ability to penetrate walls, however, result in lower accuracy and precision.

Systems relying solely on proximity detection would require a large number of sensors if they were to provide room-level indoor location. Floor-level indoor location might require fewer sensors since only access points such as stairway entrances and elevators need to be covered, but it still presents a significant infrastructure challenge.

5.6.1.2 Triangulation

Triangulation measures the distances from multiple known reference points to determine a user's location.

Cricket [89] and WALRUS [42] transmit RF and ultrasonic signal simultaneously. Because the two signals travel at different speeds, a receiver can derive the distance to the transmitter from the difference in the arrival times. This eliminates the need to synchronize the clocks of all transmitters and receivers, as is the case for other systems based on time-of-arrival.

Ubisense [31] provides a commercial solution for indoor positioning using ultra-wideband (UWB). UWB has an advantage for indoor positioning in that it does not suffer from multipath effect. UWB signal has a short pulse timing, thus the path signal can easily be distinguished from the reflected ones. The Ubisense system provides very accurate indoor locations, with errors of less than 15 cm.

Compared to proximity detection, triangulation requires fewer sensors, but the number is still high. For example, a Ubisense system installed for a 1,800 m assembly line consists of 470 sensors [85]. In addition, multipath and shading effect make it hard to use triangulation for vertical positioning. Floors and ceilings of a building degrade accuracy. Installing sensors on every floor will solve the problem, but such an installation will lose the triangulation's advantage over proximity detection.

5.6.1.3 Fingerprinting

Fingerprinting identifies signals that have long-term stability at each location. During the *offline* phase, the signal strengths at different location coordinates are recorded to build a fingerprinting database. During the *online* phase, the real-time signal measurement is looked up in the fingerprinting database to find a matching location.

Many kinds of signals have been used for fingerprinting systems. RADAR [40], Place Lab [71], Horus [126], FindingMiMo [103], and Ekahau [9] use ubiquitous Wi-Fi signals. WALRUS [42] and Centaur [79] combine Wi-Fi and acoustic localization. SkyLoc [112] uses GSM signals. MoteTrack [76] is based on RF signals generated by low-power radio transceivers. There are also a number of systems that use the distortions of Earth's magnetic field caused by the steel structure of a building [15, 48, 61].

Some of the fingerprinting-based systems consider vertical location. SkyLoc [112] was in fact the first floor localization paper. SkyLoc was also motivated by the importance of floor localization in emergency situations, and it tackled the problem using GSM fingerprinting. Shin *et al.* [103] and Chung *et al.* [48] also considered floor-level vertical localization as part of their system using Wi-Fi signals and geo-magnetism, respectively.

One disadvantage of fingerprinting is the effort required to conduct offline surveys. To achieve an acceptable accuracy, signals should be sampled about every meter, and on top of that, at least toward four different directions at each location [40], which generates an enormous amount of data.

Moreover, fingerprinting-based approaches are vulnerable to transient conditions like interfering electromagnetic signals, or on-going changes like rearranged equipment and furniture. New cell towers and changes in RF patterns due to refarming also affect fingerprinting-based approaches. This characteristic makes fingerprinting an unsuitable approach for emergency call systems.

5.6.2 Dead Reckoning

Double Integration

Systems based on dead reckoning typically measure a user's acceleration, and calculate the distance by double integration. The main disadvantage of double integration is that the accumulation of errors degrades the accuracy of the distance estimation over time.

NavShoe system [59] uses foot-mounted inertial sensors and applies ZUPT to achieve a significant reduction of errors. While a person is walking, one foot is in "stationary stance phase", while the other is in "moving stride phase". At every stationary stance phase, the velocity of the foot is zero. We have also used ZUPT in our elevator module. As the foot's velocity becomes zero at the stance phase, the elevator's velocity becomes zero when it stops on a floor.

Ojeda and Borenstein [81] have successfully traced a user's movement on a stairway using a foot-mounted IMU. Their system provides the elevation changes in meters, while our stairway module returns the number of floors a user has traveled. The direct measurement of a vertical displacement is possible because their IMU is mounted on the user's foot. It is hard to measure the accurate vertical acceleration of human steps with a smartphone at an arbitrary position.

Xuan *et al.* [122] and Shanklin *et al.* [101] use smartphones to develop indoor positioning systems. Both systems do not reach the accuracy of the foot-mounted systems because of the lack of adequate mechanisms to handle accelerometer drifts. Moreover, inertial sensors in smartphones are more prone to errors. Our elevator module only considers movements in one direction, thus we can easily filter out noises in other directions. Our use of ZUPT in the elevator module also increases the overall accuracy.

Step Counting

Step counting detects steps in human movements and measures the displacement vector of each step. The displacement vector is composed of the stride length and direction. The user's location is then calculated by adding all displacement vectors to the initial location.

Yeh *et al.* [125] and Vildjiounaite *et al.* [113] use sensors which are mounted on a user's shoes or ankles for measurement. On the one hand, foot-mounted sensors can directly measure human steps, so errors in step detection and stride estimation can be reduced. On the other hand, those systems need customized hardware, which can be an obstacle to wide deployment.

Emilsson [55] built a step counting system on iPhone 3GS, a device that has no gyroscope. The lack of gyroscope makes it hard to handle an arbitrary orientation of the device. Therefore, Emilsson had to attach the phone on the user's body in a known direction. Such a deployment is impractical in real life.

Jin *et al.* [66] estimate the length of a stride using a model for human steps. The model contains a parameter that is dependent on a person's gait. Thus, the model requires training by the user.

These systems detect human steps by identifying the local maximum and minimum of vertical acceleration. A pair of local maximum and minimum within a short time period identifies one human step. Our activity manager uses this approach to detect the act of walking, but it does not need to count the steps.

Our stairway module similarly monitors the amplitude of vertical acceleration. The difference is that, instead of trying to identify each and every step by scrutinizing vertical acceleration, we detect landings by focusing on large amplitude changes in acceleration, which are easier to identify.

Time-based Approach

Yet another way to implement dead reckoning is to measure the travel time. The traveled distance can then be calculated from the predetermined velocity, which is commonly obtained through training.

Two systems [84,124] implement floor localization using the time-based approach. Both systems track a user's movement in elevators and on stairways. In [84], a user's current activity is classified into one of four classes, elevator up/down and stairs up/down, using the smartphone's real-time accelerometer data. The system then estimates the number of floors that the user has traveled simply by dividing the total travel time by the time it takes to travel one floor. This system requires a training period to build a classifier for each activity and to calculate the average times needed to travel one floor. FTrack [124] takes a similar approach, but uses a novel crowdsourcing technique to construct a mapping from the starting floor and travel time to the destination floor. Crowdsourcing, however, requires the willing participation of a large number of users, which may not be feasible. In addition, there is still a privacy concern during FTrack's offline map construction phase.

The main disadvantage of the two time-based approaches is that it cannot take account of speed variations. Different elevators can have different speeds. Users may walk at different speeds on stairs, or may even climb up multiple stairs in each step. Our system do not have such limitations. Our elevator and escalator module is distance-based, rather than time-based. Our stairway module works by counting landings, rather than counting individual steps or measuring the travel time.

5.6.3 Hybrid Systems

Hybrid systems combine infrastructure-based systems and dead reckoning in order to overcome the shortcomings associated with taking a single approach. The estimated locations from dead reckoning are periodically adjusted by the information from the infrastructure, such as RFID beacons [125] or Wi-Fi fingerprinting [90, 120]. Beacons in this case can be deployed in much coarser granularity compared to the systems purely based on infrastructure.

Woodman and Harle [120] proposed a hybrid indoor localization system that uses dead reckoning and Wi-Fi fingerprinting to track a pedestrian through multiple floors. Their system tracks the user's movement using a foot-mounted IMU, and aligns the user's path with the floor plan of the building. Wi-Fi fingerprinting constrains the possible initial locations into a particular region of the building, which in turn reduces the complexity of the alignment algorithm, and resolves the ambiguity arising from the symmetries in the floor plan. Zee [90] also combines dead reckoning and Wi-Fi fingerprinting, but for a different purpose. Zee aims to eliminate the calibration effort required to build the Wi-Fi fingerprinting database.

Our system can be viewed as a hybrid system because we primarily rely on dead reckoning, but we anchor the user's location using the information from the entrance beacon. The user's location is also checked and adjusted by the location beacons sparsely deployed throughout the building.

5.7 Conclusion

This chapter makes three contributions toward improving vertical accuracy of indoor positioning. First, we present a hybrid architecture for floor localization with emergency calls in mind. The architecture combines beacon-based infrastructure and sensor-based dead reckoning, striking a balance between accurately determining a user's location and minimizing the required infrastructure. Second, we present the elevator module for tracking a user's movement in an elevator. The elevator module addresses three core challenges that make it difficult to accurately derive displacement from acceleration. Third, we present the stairway module which determines the number of floors a user has traveled on foot. Unlike previous systems that track users' foot steps, our stairway module uses a novel landing counting technique.

We recognize that there are many hurdles to overcome before our system can be deployed in the real world. For instance, our elevator module assumes that the acceleration inside an elevator is mostly vertical. This will not be the case if a user happens to pace back and forth during the ride. Similar shortcomings also exist in the stairway module. The stairway module can produce false positives in some unusual cases. For example, a user can stop in the middle of a stairway, slowly turn around 180 degrees, and walk the rest of the stairway backward. This is highly unlikely, but it illustrates the general limitation of our approach that relies on behavioral norms. As future work, we plan to study the effects of various unusual behaviors, and explore possible solutions to address them.

We also plan to improve activity detection using ambient signals. For example, an entry to a building can be detected using the RFID signals from anti-theft gates, which are typically installed at the entrances of libraries and retail stores. Identifiable magnetic signatures can be detected around elevators and escalators. Even though we have argued against relying on a barometer for vertical location, a barometer can be useful as an additional input to distinguish between stairway and same-floor walking.

Part III

Conclusions

Chapter 6

Conclusions

This thesis presents a floor localization system intended for emergency calls. We aim to provide floor-level accuracy with minimum infrastructure support. Our approach uses multiple sensors, all available in today's smartphones, to trace a user's vertical movements inside buildings. Our system is immune to transient conditions or on-going changes inside the building. Our hybrid design reduces infrastructure requirements by filling the gap between sparsely deployed beacons with dead reckoning. In our system, partial failure does not result in a complete system failure, but results in gradual degradation of performance.

This thesis makes three contributions toward improving vertical accuracy of indoor positioning. First, we present a hybrid architecture for floor localization with emergency calls in mind. The architecture combines beacon-based infrastructure and sensor-based dead reckoning, striking a right balance between accurately determining a user's location and minimizing the required infrastructure. Second, we present the elevator module for tracking a user's movement in an elevator. The elevator module addresses three core challenges that make it difficult to accurately derive displacement from acceleration. Third, we present the stairway module which determines the number of floors a user has traveled on foot. Unlike previous systems that track users' foot steps, our stairway module uses a novel landing counting technique.

We implemented our indoor positioning system on the iOS platform and evaluated it in the four different buildings. The evaluation results show that all three analysis modules can provide accurate vertical location when the user travels using elevators, stairways, and escalators. The elevator module can provide accurate vertical location up to about 15 floors in the worst case, and the range can be extended if the elevator makes stops during the travel due to ZUPT. The stairway module can report the exact number of floors the user has moved up and down on stairways. The escalator module can determine the traveled number of floors accurately up to about five floors even when the user walks on the escalator while riding it.

We compare our solution with infrastructure-based and dead reckoning systems. Our hybrid approach reduces the number of infrastructure sensors as much as possible by combining the beacon-based infrastructure and IMU-based dead reckoning. Extensive infrastructure requirements in proximity detection and triangulation hinder wide deployment of these systems. Unlike fingerprinting systems, our system does not require any offline surveys. Moreover, our system is resilient to transient conditions or on-going changes inside the building, which is a big advantage over the fingerprinting-based systems in emergency situations. Existing systems using dead reckoning focus on the movements on foot, thus, they cannot determine a user's vertical location if the user rides an elevator or an escalator, while our elevator and escalator modules can accurately provide the user's vertical location in such cases. Since our stairway module detects landings by monitoring large amplitude changes in the vertical acceleration, our system needs not mount sensors on a user's foot, nor requires training period, which are commonly seen in many indoor positioning systems using step counting.

Additionally, this thesis presents our three prior efforts on the NG9-1-1 system. We first demonstrate how emergency calls from various call origination devices and networks are identified, routed based on the caller's location, and terminated by the call taker software at the PSAP. We also present the architectural overview and the details of the software components in our NG9-1-1 prototype system. Second, we show how text communications such as IM and SMS can be integrated into the NG9-1-1 architecture. We identify the technical challenges in the integration of IM and SMS networks with the NG9-1-1 system, and propose a solution for each challenge. Lastly we present GeoPS-PD, a polygon simplification algorithm designed to improve the performance of location-based routing. GeoPS-PD never produces a false positive, is tunable at runtime for the desired balance between target polygon size and area coverage, and optionally takes into account the population density.

Part IV

Bibliography

Bibliography

[1] Apache Tomcat. http://tomcat.apache.org/.

[2] BlueCove. http://bluecove.org/.

[3] Boston Police Department Crime Stoppers. http://www.cityofboston.gov/Police/crimstop_mobile_terms.asp.

[4] BTstack: A Portable User-Space Bluetooth Stack. http://code.google.com/p/btstack/.

[5] Census 2000 State and State Equivalent Areas Cartographic Boundary Files. http://www.census.gov/geo/www/cob/st2000.html.

[6] Dialogic IP Media Server. http://www.dialogic.com/.

[7] Durham 911 center extends texting trial for emergency help. http://durhamnc.gov/Pages/NNDetails.aspx?detailId=60.

[8] eCall: Time saved = lives saved. http://ec.europa.eu/digital-agenda/en/ecall-time-saved-lives-saved.

[9] Ekahau - Wi-Fi Tracking Systems, RTLS and WLAN Site Survey. http://www.ekahau.com/.

[10] FCC Updating 911 for the Texting Generation. http://www.cnn.com/2010/TECH/mobile/11/23/fcc.911.texting.wired/index.html.

[11] FFmpeg. http://ffmpeg.org/.

[12] Global Instant Messaging Market Share - Open Data. http://billionsconnected.com/blog/2008/08/global-im-market-share-im-usage/.

[13] GNU ccRTP. http://www.gnu.org/software/ccrtp/.

[14] Google Maps. https://maps.google.com/.

[15] IndoorAtlas. http://www.indooratlas.com/.

[16] JAIN SIP - Java API for SIP Signaling. http://jain-sip.dev.java.net/.

[17] MySQL. http://www.mysql.com/.

[18] New York Police Department Crime Stoppers. http://a056-crimestoppers.nyc.gov/crimestoppers/public/index.cfm.

[19] OnStar. http://www.onstar.com/.

[20] PL/pgSQL - SQL Procedural Language. http://developer.postgresql.org/pgdocs/postgres/plpgsql.html.

[21] PostGIS. http://postgis.refractions.net/.

[22] PostgreSQL. http://www.postgresql.org/.

[23] REACH 112. http://www.reach112.eu/view/en/112.html.

[24] SIP Communicator - the Java VoIP and Instant Messaging Client. http://sip-communicator.org.

[25] Skyhook Location SDK Overview. http://www.skyhookwireless.com/location-technology/.

[26] Sybase 365 Mobile Messaging Services. http://www.sybase.com/mobileservices/.

[27] Tcl/TK. http://www.tcl.tk/.

[28] Texas 9-1-1 GIS Collaboration Portal - Download GIS Data. https://tx911map.911.state.tx.us/download/.

[29] The SailFin Project. https://sailfin.dev.java.net/.

[30] TOPAZ. http://www.tadlys.co.il/.

[31] Ubisense. http://www.ubisense.net/.

[32] vic - Video Conferencing Tool. http://ee.lbl.gov/vic/.

[33] VZ Navigator - Overview. https://vznavigator.vzw.com/.

[34] What You Need to Know About Text-to-911. http://www.fcc.gov/text-to-911.

[35] Link Layer Discovery Protocol for Media Endpoint Devices. Technical Report ANSI/TIA-1057-2006, Telecommunications Industry Association, April 2006.

[36] NENA i3 Technical Requirements Document. Technical Report NENA 08-751, National Emergency Number Association (NENA), September 2006.

[37] NENA Functional and Interface Standards for Next Generation 9-1-1 Version 1.0 (i3). Technical Report NENA 08-002, National Emergency Number Association (NENA), December 2007.

[38] Next Generation 112 (NG112): Introduction to Next Generation Emergency Services in Europe. Technical Report NG112-001, The European Emergency Number Association, 2011.

[39] R. Agrawal and A. Vasalya. Bluetooth Navigation System using Wi-Fi Access Points. Technical Report arXiv:1204.1748, Cornell University Library, April 2012.

[40] P. Bahl and V. Padmanabhan. RADAR: An In-Building RF-based User Location and Tracking System. In *Proc. of the Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, March 2000.

[41] P. Bhagwat. Bluetooth: Technology for Short-Range Wireless-Apps. *IEEE Internet Computing*, 5(3):96–103, 2001.

[42] G. Borriello, A. Liu, T. Offer, C. Palistrant, and R. Sharp. WALRUS: Wireless Acoustic Location with Room-Level Resolution using Ultrasound. In *Proc. of the 3rd*

International Conference on Mobile Systems, Applications, and Services (MobiSys), June 2005.

[43] Business Insider. In The Biggest Blown Opportunity Ever, AOL Instant Messenger Has Utterly Collapsed. http://read.bi/HIulvq.

[44] B. Campbell, R. Mahy, and C. Jennings. The Message Session Relay Protocol (MSRP). RFC 4975, September 2007.

[45] B. Campbell, J. Rosenberg, H. Schulzrinne, C. Huitema, and D. Gurle. Session Initiation Protocol (SIP) Extension for Instant Messaging. RFC 3428, December 2002.

[46] A. Carroll and G. Heiser. An Analysis of Power Consumption in a Smartphone. In *Proc. of the 2010 USENIX Annual Technical Conference (USENIX ATC)*, June 2010.

[47] B. Chun and P. Maniatis. Augmented Smartphone Applications Through Clone Cloud Execution. In *Proc. of the 12th Workshop on Hot Topics in Operating Systems (HotOS)*, May 2009.

[48] J. Chung, M. Donahoe, C. Schmandt, I. Kim, P. Razavai, and M. Wiseman. Indoor Location Sensing Using Geo-Magnetism. In *Proc. of the 9th International Conference on Mobile Systems, Applications, and Services (MobiSys)*, June 2011.

[49] D. Cohen-Or, S. Lev-Yehudi, A. Karol, and A. Tal. Inner-cover of Non-convex Shapes. *International Journal of Shape Modeling*, 9(2):223–238, 2003.

[50] Council on Tall Buildings and Urban Habitat. Height Calculator. http://http://www.ctbuh.org/TallBuildings/HeightStatistics/HeightCalculator/tabid/1007/language/en-GB/Default.aspx.

[51] CSRIC III, Working Group 3. Indoor Location Test Bed Report. http://transition.fcc.gov/bureaus/pshs/advisory/csric3/CSRIC_III_WG3_Report_March_%202013_ILTestBedReport.pdf, March 2013.

[52] CTIA - The Wireless Association. Semi-annual Wireless Industry Survey. http://www.ctia.org/advocacy/research/index.cfm/aid/10316/, May 2013.

[53] S. Dixon-Warren. Motion Sensing in the iPhone 4: Electronic Compass. http://www.memsjournal.com/2011/02/motion-sensing-in-the-iphone-4-electronic-compass.html, February 2011.

[54] D. H. Douglas and T. K. Peucker. Algorithms for the Reduction of the Number of Points Required to Represent a Digitized Line or its Caricature. *Cartographica: The International Journal for Geographic Information and Geovisualization*, 10(2):112–122, 1973.

[55] A. Emilsson. Indoor Navigation using an iPhone. Master's thesis, Linköpings universitet, June 2010.

[56] Environmental Systems Research Institute. ESRI Shapefile Technical Description, An ESRI White Paper. http://www.esri.com/library/whitepapers/pdfs/shapefile.pdf, July 1998.

[57] R. Falke. Building Pressure Diagnostics. http://contractingbusiness.com/enewsletters/cb_imp_9596/, May 2005.

[58] Federal Communications Commission. Further Notice of Proposed Rulemaking. *Facilitating the Deployment of Text-to-9-1-1 and Other Next Generation 911 Applications, Framework for Next Generation 911 Deployment*, FCC 12-149, December 2012.

[59] E. Foxlin. Pedestrian Tracking with Shoe-Mounted Inertial Sensors. *IEEE Computer Graphics and Applications*, 25(6):38–46, 2005.

[60] W. R. Franklin. PNPOLY - Point Inclusion in Polygon Test. http://www.ecse.rpi.edu/Homepages/wrf/Research/Short_Notes/pnpoly.html.

[61] B. Gozick, K. Subbu, R. Dantu, and T. Maeshiro. Magnetic Maps for Indoor Navigation. *IEEE Transaction on Instrumentation and Measurement*, 60(12):3883–3891, 2011.

[62] D. Grejner-Brzezinska, Y. Yi, and C. Toth. Bridging GPS Gaps in Urban Canyons: The Benefits of ZUPTs. *Navigation*, 48(4):217–225, 2001.

[63] T. Hardie, A. Newton, H. Schulzrinne, and H. Tschofenig. LoST: A Location-to-Service Translation Protocol. RFC 5222, August 2008.

[64] J. Hautakorpi, G. Camarillo, R. Penfield, A. Hawrylyshen, and M. Bhatia. Requirements from Session Initiation Protocol (SIP) Session Border Control (SBC) Deployments. RFC 5853, April 2010.

[65] G. Hellstrom and P. Jones. RTP Payload for Text Conversation. RFC 4103, June 2005.

[66] Y. Jin, H. Toh, W. Soh, and W. Wong. A Robust Dead-Reckoning Pedestrian Tracking System with Low Cost Sensors. In *Proc. of IEEE International Conference on Pervasive Computing and Communication (PerCom)*, March 2011.

[67] S. Kerber and W. Walton. Effect of Positive Pressure Ventilation on a Room Fire. Technical Report NISTIR-7213, National Institute of Standards and Technology, March 2005.

[68] J. Kim, W. Song, H. Schulzrinne, A. Zacchi, A. Jain, H. Chenji, C. Magnussen, C. Norton, W. Magnussen, I. Schworer, and K. Trinh. The Next Generation 9-1-1 Proof-Of-Concept System. In *Proc. of ACM SIGCOMM*, August 2008.

[69] A. Küpper. *Location-Based Services: Fundamentals and Operation*. Wiley, 2005.

[70] Q. Ladetto and B. Merminod. In Step with INS: Navigation for the Blind, Tracking Emergency Crews. *GPS World*, 13(10):30–38, 2002.

[71] A. LaMarca, Y. Chawathe, S. Consolvo, J. Hightower, I. Smith, J. Scott, T. Sohn, J. Howard, J. Hughes, F. Potter, J. Tabert, P. Powledge, G. Borriello, and B. Schilit. Place Lab: Device Positioning Using Radio Beacons in the Wild. In *Proc. of the 3rd International Conference on Pervasive Computing (Pervasive)*, May 2005.

[72] G. Lammel, J. Gutmann, L. Marti, and M. Dobler. Indoor Navigation with MEMS sensors. *Procedia Chemistry*, 1(1):532–535, 2009.

[73] R. B. Langley. NMEA 0183: A GPS Receiver Interface Standard. *GPS world*, 6(7):54–57, 1995.

[74] J. Lennox, H. Schulzrinne, and J. Rosenberg. Common Gateway Interface for SIP. RFC 3050, January 2001.

[75] Levy, S. The Inside Story of the Moto X. http://www.wired.com/gadgetlab/2013/08/inside-story-of-moto-x/, August 2013.

[76] K. Lorincz and M. Welsh. MoteTrack: a robust, decentralized approach to RF-based location tracking. *Personal and Ubiquitous Computing*, 11(6):489–503, 2006.

[77] R. B. McMaster and K. S. Shea. *Generalization in Digital Cartography*. Association of American Geographers, 1992.

[78] N. Mulkijanyan. Evaluation Procedure for QoS of Short Message Service. Master's thesis, KTH Royal Institute of Technology, November 2011.

[79] R. Nandakumar, K. K. Chintalapudi, and V. N. Padmanabhan. Centaur: Locating Devices in an Office Environment. In *Proc. of the 18th Annual International Conference on Mobile Computing and Networking (MobiCom)*, August 2012.

[80] A. Niemi. Session Initiation Protocol (SIP) Extension for Event State Publication. RFC 3903, October 2004.

[81] L. Ojeda and J. Borenstein. Personal Dead-reckoning System for GPS-denied Environments. In *Proc. of the IEEE International Workshop on Safety, Security, and Rescue Robotics (SSRR)*, September 2007.

[82] Open Geospatial Consortium, Inc. OpenGIS Geography Markup Language (GML) Encoding Standard. http://www.opengeospatial.org/standards/gml, August 2007.

[83] J. Paek, J. Kim, and R. Govindan. Energy-Efficient Rate-Adaptive GPS-based Positioning for Smartphones. In *Proc. of the 8th International Conference on Mobile Systems, Applications, and Services (MobiSys)*, June 2010.

[84] A. Parnandi, K. Le, P. Vaghela, A. Kolli, K. Dantu, S. Poduri, and G. Sukhatme. Coarse In-building Localization with Smartphones. In *Proc. of the First Annual Inter-*

national Conference on Mobile Computing, Applications, and Services (MobiCASE), October 2009.

[85] T. Phebey. The Ubisense Assembly Control Solution for BMW. http://www.rfidjournal.net/masterPresentations/rfid_europe2010/np/chandler_phebey_nov2_500_rtlsManu.pdf, November 2010.

[86] J. Polk, B. Rosen, and J. Peterson. Location Conveyance for the Session Initiation Protocol. RFC 6442, December 2011.

[87] J. Polk, J. Schnizlein, and M. Linsner. Dynamic Host Configuration Protocol Option for Coordinate-based Location Configuration Information. RFC 3825, July 2004.

[88] B. Priyantha, D. Lymberopoulos, and J. Liu. LittleRock: Enabling Energy-Efficient Continuous Sensing on Mobile Phones. *IEEE Pervasive Computing*, 10(2):12–15, 2011.

[89] N. Priyantha, A. Chakraborty, and H. Balakrishnan. The Cricket Location-Support System. In *Proc. of the 6th Annual International Conference on Mobile Computing and Networking (MobiCom)*, August 2000.

[90] A. Rai, K. K. Chintalapudi, V. N. Padmanabhan, and R. Sen. Zee: Zero-Effort Crowdsourcing for Indoor Localization. In *Proc. of the 18th Annual International Conference on Mobile Computing and Networking (MobiCom)*, August 2012.

[91] B. Rosen and J. Polk. Best Current Practice for Communications Services in support of Emergency Calling. RFC 6881, March 2013.

[92] J. Rosenberg. A Presence Event Package for the Session Initiation Protocol (SIP). RFC 3856, August 2004.

[93] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler. SIP: Session Initiation Protocol. RFC 3261, June 2002.

[94] P. Saint-Andre. Extensible Messaging and Presence Protocol (XMPP): Core. RFC 6120, March 2011.

[95] H. Schulzrinne. Dynamic Host Configuration Protocol (DHCPv4 and DHCPv6) Option for Civic Addresses Configuration Information. RFC 4776, November 2006.

[96] H. Schulzrinne. A Uniform Resource Name (URN) for Emergency and Other Well-Known Services. RFC 5031, January 2008.

[97] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson. RTP: A Transport Protocol for Real-Time Applications. RFC 3550, July 2003.

[98] H. Schulzrinne and R. Marshall. Requirements for Emergency Context Resolution with Internet Technologies. RFC 5012, January 2008.

[99] H. Schulzrinne, A. Rao, and R. Lanphier. Real Time Streaming Protocol (RTSP). RFC 2326, April 1998.

[100] H. Schulzrinne, H. Tschofenig, A. Newton, and T. Hardie. LoST: A Protocol for Mapping Geographic Locations to Public Safety Answering Points. In *Proc. of IEEE International Performance Computing and Communications Conference (IPCCC)*, April 2007.

[101] T. Shanklin, B. Loulier, and E. Matson. Embedded Sensors for Indoor Positioning. In *Proc. of IEEE Sensors Applications Symposium (SAS)*, February 2011.

[102] S. Shanmugham, P. Monaco, and B. Eberman. A Media Resource Control Protocol (MRCP) Developed by Cisco, Nuance, and Speechworks. RFC 4463, April 2006.

[103] H. Shin, Y. Chon, K. Park, and H. Cha. FindingMiMo: Tracing a Missing Mobile Phone using Daily Observations. In *Proc. of the 9th International Conference on Mobile Systems, Applications, and Services (MobiSys)*, June 2011.

[104] Shklovski, I. and Burke, M. and Kiesler, S. and Kraut, R. Technology Adoption and Use in the Aftermath of Hurricane Katrina in New Orleans. *American Behavioral Scientist*, 53(8):1228–1246, 2010.

[105] K. Singh, W. Jiang, J. Lennox, S. Narayanan, and H. Schulzrinne. CINEMA: Columbia InterNet Extensible Multimedia Architecture. Technical Report CUCS-011-02, Columbia University, 2002.

[106] W. Song, J. Y. Kim, H. Schulzrinne, P. Boni, and M. Armstrong. Using IM and SMS for Emergency Text Communications. In *Proc. of the 3rd International Conference on Principles, Systems and Applications of IP Telecommunications (IPTComm)*, July 2009.

[107] R. Sparks. The Session Initiation Protocol (SIP) Refer Method. RFC 3515, April 2003.

[108] U.S. Census Bureau. Census 2000 Summary File 1. http://www.census.gov/prod/cen2000/doc/sf1.pdf, July 2007.

[109] U.S. Census Bureau. Geographic Areas Reference Manual. http://www.census.gov/geo/reference/pdfs/GARM/GARMcont.pdf, November 2007.

[110] U.S. Department of Transportation. Next Generation 9-1-1. http://www.its.dot.gov/ng911/.

[111] U.S. Department of Transportation. Next Generation 9-1-1 (NG9-1-1) System Initiative: Proof of Concept Testing Report. http://www.its.dot.gov/ng911/pdf/NG911_POCTesTReport091708.pdf, September 2008.

[112] A. Varshavsky, A. LaMarca, J. Hightower, and E. de Lara. The SkyLoc Floor Localization System. In *Proc. of the 5th Anual IEEE International Conference on Pervasive Computing and Communications (PerCom)*, March 2007.

[113] E. Vildjiounaite, E. Malm, J. Kaartinen, and P. Alahuhta. Location Estimation Indoors by Means of Small Computing Power Devices, Accelerometers, Magnetic Sensors, and Map Knowledge. *Pervasive Computing, LNCS*, 2414:5–12, 2002.

[114] M. Visvalingam and J. D. Whyatt. Line generalisation by repeated elimination of points. *Cartographic Journal*, 30(1):46–51, 1993.

[115] H. Wang, S. Sen, A. Elgohary, M. Farid, M. Youssef, and R. R. Choudhury. No Need to War-Drive: Unsupervised Indoor Localization. In *Proc. of the 10th International Conference on Mobile Systems, Applications, and Services (MobiSys)*, June 2012.

[116] S. Wang, M. Green, and M. Malkawa. E-911 Location Standards and Location Commercial Services. In *Proc. of IEEE Emerging Technologies Symposium: Broadband, Wireless Internet Access*, April 2000.

[117] Z. Wang and J. Muller. Line Generalization Based on Analysis of Shape Characteristics. *Cartography and Geographic Information Science*, 25(1):3–15, 1998.

[118] R. Want, A. Hopper, V. Falcao, and J. Gibbons. The Active Badge Location System. *ACM Transactions on Information Systems (TOIS)*, 10(1):91–102, 1992.

[119] J. Winterbottom, M. Thomson, and H. Tschofenig. GEOPRIV Presence Information Data Format Location Object (PIDF-LO) Usage Clarification, Considerations, and Recommendations. RFC 5491, March 2009.

[120] O. Woodman and R. Harle. Pedestrian Localisation for Indoor Environments. In *Proc. of the 10th International Conference on Ubiquitous Computing (Ubicomp)*, September 2008.

[121] X. Wu and H. Schulzrinne. SIPC, a Multi-function SIP User Agent. *Management of Multimedia Networks and Services, LNCS*, 3271:269–281, 2004.

[122] Y. Xuan, R. Sengupta, and Y. Fallah. Making Indoor Maps with Portable Accelerometer and Magnetometer. In *Proc. of the International Conference on Ubiquitous Positioning, Indoor Navigation and Location-Based Service (UPINLBS)*, October 2010.

[123] Z. Yan, V. Subbaraju, D. Chakraborty, A. Misra, and K. Aberer. Energy-Efficient Continuous Activity Recognition on Mobile Phones: An Activity-Adaptive Approach. In *Proc. of the Sixteenth Annual International Symposium on Wearable Computers (ISWC)*, June 2012.

[124] H. Ye, T. Gu, X. Zhu, J. Xu, X. Tao, J. Lu, and N. Jin. FTrack: Infrastructure-free Floor Localization via Mobile Phone Sensing. In *Proc. of the 10th Anual IEEE International Conference on Pervasive Computing and Communications (PerCom)*, March 2012.

[125] S. Yeh, K. Chang, C. Wu, H. Chu, and J. Hsu. GETA sandals: a footstep location tracking system. *Personal and Ubiquitous Computing*, 11(6):451–463, 2006.

[126] M. Youssef and A. Agrawala. The Horus WLAN Location Determination System. In *Proc. of the 3rd International Conference on Mobile Systems, Applications, and Services (MobiSys)*, June 2005.

Part V

Appendices

Appendix A

Measuring Altitude Using a Barometer

We measured the altitude of indoor location using a barometric pressure sensor. This is a preliminary experiment to show how viable the barometer-based approach is for the indoor vertical location.

A.1 Setup

Figure A.1 shows the hardware we used for the experiment. The barometric pressure sensor we used is BMP085, a digital pressure sensor from Bosch. The sensor runs in ultra high resolution mode where the root mean square noise is 0.25 m, according to the data sheet from the manufacturer. The data sampling rate is 60 Hz. The pressure sensor is connected to the Arduino Uno board with the I^2C interface. Arduino Uno is a programmable microcontroller that reads pressure measurement from the sensor, calculates altitude, and passes the result to the iPhone 4S using the C2-DB9 connector. The iPhone runs an app that collects data from the barometer and transfers it to a central repository for future analysis.

We used the following equation to derive altitude from measured pressure:

$$altitude = 44300 \times \left(1 - \left(\frac{p}{p_0}\right)^{\frac{1}{5.255}}\right) \tag{A.1}$$

where p_0 is mean sea level pressure (MSLP) and p is measured pressure. We used hourly

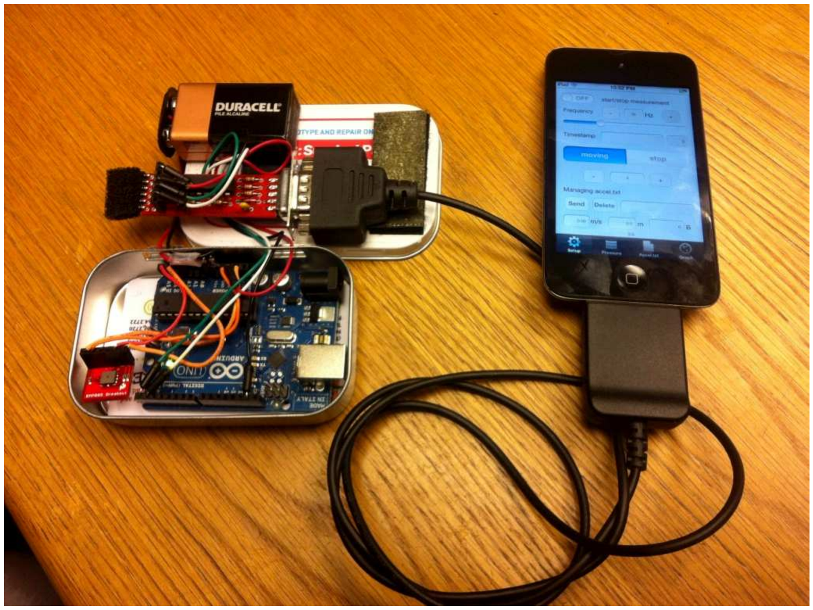


Figure A.1: Barometer system for altitude measurement.

MSLP report available at the National Oceanic and Atmospheric Administration (NOAA) web site[1] as a reference pressure to calibrate the barometer. The MSLP is measured at Central Park, New York, and the distance between the observatory and our test building is about 2 miles.

A.2 Experiment Results

A.2.1 Absolute Altitude

Figure A.2 shows the altitude measured at the same location in the CEPSR building. The measured altitude decreased 5.65 m for 30 minutes, due to increased outside air pressure, even though the barometer was stationary in the building. The error of 5.65 m is larger than the floor-to-floor height of the building so the barometer alone cannot be used to determine the correct floor number. The graph shows that MSLP increased 130 Pa during the measurement.

Figure A.3 shows the altitude from the barometer at different measurement points on

[1] http://w1.weather.gov/obhistory/KNYC.html

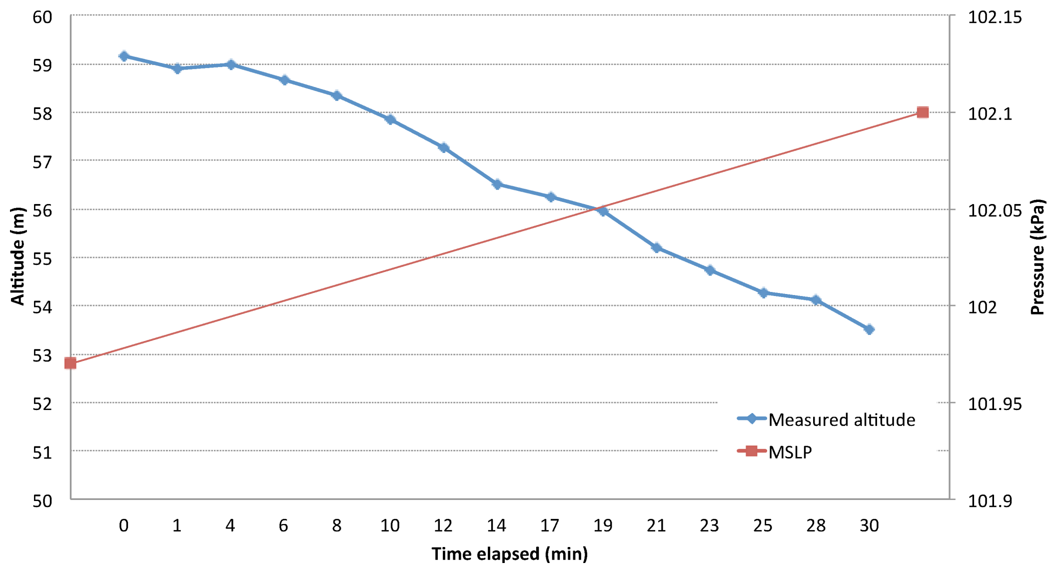


Figure A.2: Altitude from barometer at one indoor location.

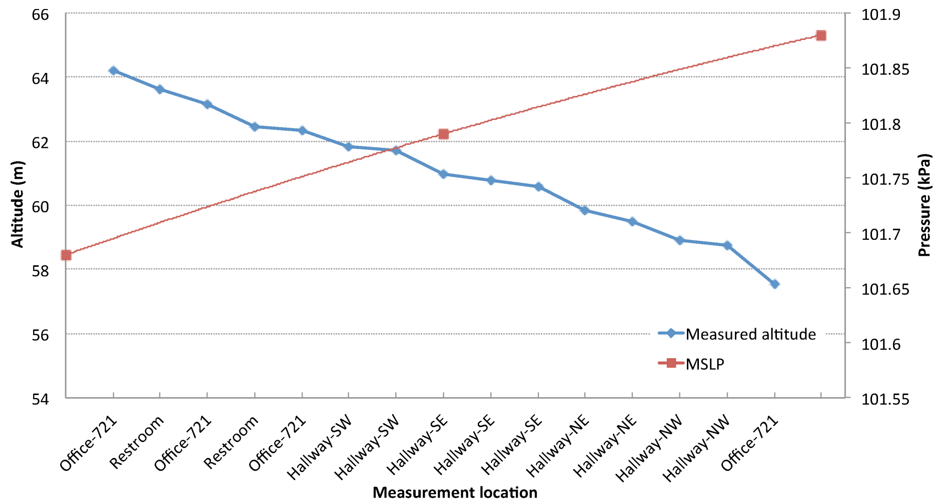
the same floor. We observed that the measured altitude did not change much when we moved around on the same floor. The altitude decreased (or increased) as the outdoor air pressure increased (or decreased).

During the experiment, we did not find any case where barometer readings were affected by the building's heating, ventilation, and air conditioning. Barometer readings at the hallway and in the restroom are not much different. In our experiment, indoor altitude measurements were primarily affected by the changes of outdoor air pressure. Therefore, it is possible to accurately measure the indoor vertical location using barometers if we can get real-time weather information and calibrate barometers with it.

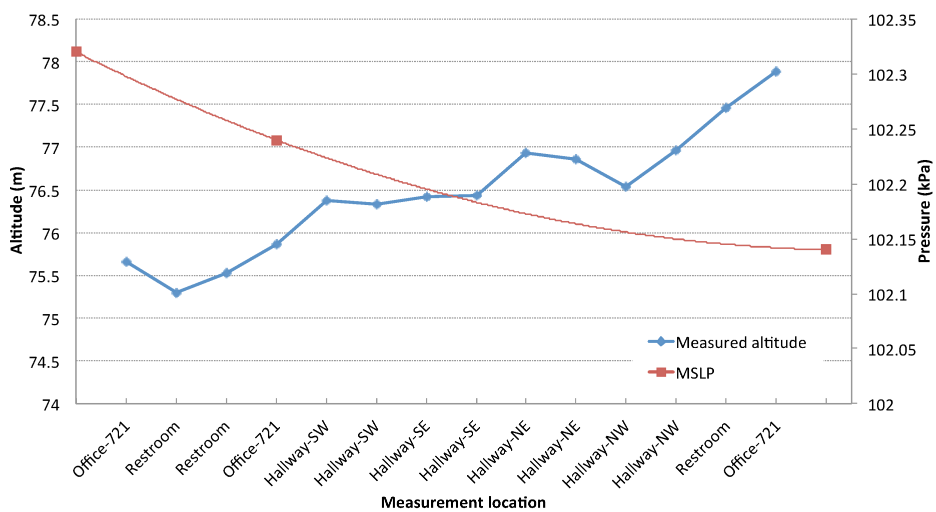
A.2.2 Relative Altitude

Figure A.4 shows daily altitude measurement on the second, fourth, sixth, and seventh floor in the CEPSR building for 10 days. The absolute value of the altitude for each floor fluctuates every day. However, the difference between floors are almost constant.

Table A.1 compares the floor-to-floor height derived from barometer measurements with the actual height. In all three buildings, the average error in the relative altitude is under 0.52 m.



(a)



(b)

Figure A.3: Altitude from barometer at different locations on the same floor.

Building name	Actual floor height	Average error (and stdev.)
CEPSR	4.65 m	0.43 m (0.34 m)
Mudd	3.67 m	0.52 m (0.39 m)
Pupin	3.48 m	0.32 m (0.22 m)

Table A.1: Errors in one floor height measured by barometer.

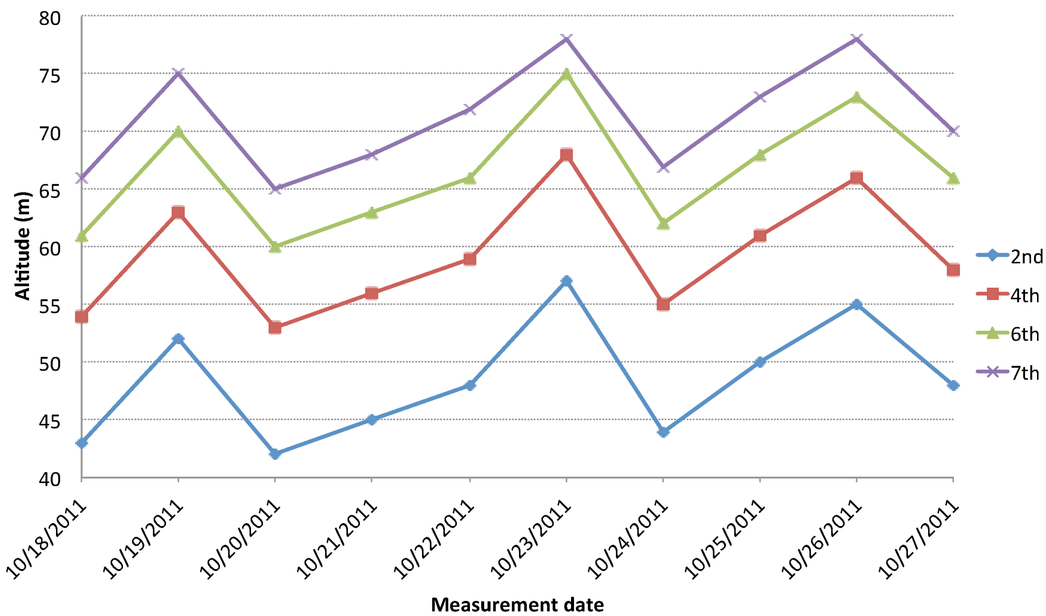


Figure A.4: Fluctuations in daily altitude measurements using barometer.

The result shows that we can accurately determine the absolute vertical location if we have a reference barometer at a known height. Using the location of the reference barometer as an anchor point, barometer readings on any other floors can be used to accurately determine the indoor vertical location.