Hindawi
Complexity
Volume 2018, Article ID 8283919, 9 pages
https://doi.org/10.1155/2018/8283919

*Research Article*

# SOSerbia: Android-Based Software Platform for Sending Emergency Messages

**Mihailo Jovanovic,[1] Ivan Babic,[2] Milan Cabarkapa,[3] Jelena Misic,[4] Sasa Mijalkovic,[1] Vojkan Nikolic,[1] and Dragan Randjelovic[1]**

[1]*Department for Informatics and Computing, Criminalistics and Police University, Belgrade 11000, Serbia*
[2]*Department for Postgraduate Studies, Singidunum University, Belgrade 11000, Serbia*
[3]*School of Electrical Engineering, University of Belgrade, Belgrade 11000, Serbia*
[4]*Faculty of Electronic Engineering, University of Nis, Nis 18000, Serbia*

Correspondence should be addressed to Milan Cabarkapa; cabmilan@etf.rs

Received 18 May 2018; Accepted 3 September 2018; Published 23 October 2018

Guest Editor: Miguel Fuentes



This paper presents Android-based SOS platform named SOSerbia for sending emergency messages by citizens in Serbia. The heart of the platform is SOS client Android application which is an easy and simple solution for sending SOS messages with unique combination of volume buttons. The proposed platform solves a lot of safety, security, and emergency problems for people who can be in dangerous situations. After a person presses a correct combination of buttons, a message with his or her location is sent to the operating center of the Serbian Police. The platform merges several appropriately combined advanced Android technologies into one complete solution. The proposed solution also uses the Google location API for getting user's location and Media Player broadcast receiver for reading pressed buttons for volume. This logic can be also customized for any other mobile operating system. In other words, the proposed architecture can be also implemented in iOS or Windows OS. It should be noted that the proposed architecture is optimized for different mobile devices. It is also implemented with simple widget and background process based on location. The proposed platform is experimentally demonstrated as a part of emergency response center at the Ministry of Interior of the Republic of Serbia. This platform overcomes real-life problems that other state-of-the-art solutions introduce and can be applied and integrated easily in any national police and e-government systems.

## 1. Introduction

Nowadays, IT technologies grow rapidly and constantly. Our daily life cannot be imagined without using these technologies. An accelerated development of the mobile OS, such as Android and iOS, has changed the main point of using mobile phones. A mobile phone is not used only for telephoning and sending messages but also is used for many new and smart features. Some of these features allow location sharing and tracking, which denotes a powerful and efficient tool that should be used carefully because of the private information, such as location [1]. On the other hand, in recent years, people live faster. In general, people could experience a lot of unexpected situations like accidents, hijacking, and criminal rate on a daily basis. Fortunately, people have their mobile phones next to them at any moment, so they can feel more protected. Thus, they can act in emergency situations quickly and save their lives. Due to the fact that Android is the most commonly used OS for mobile [2–5], there are many applications developed and specialized for its easy use (please see [1] and references therein). In other words, nowadays, the problem of emergency, dangerous situations can be potentially solved to the certain extent. According to statistics of the Ministry of Interior of the Republic of Serbia, in the Republic of Serbia, in the last 10 years, there were 139 kidnapping, 791 rapes, 257 rape attempts, 43,482 cases of home violence, 1446 thieveries, 32,584 robberies, 426 trafficking, 4 terrorism actions, 791 cases of school violence, 439,368 car accidents, and 216,041 fire accidents. This paper suggests an approach to the security problem in Serbia

through the implementation of modern mobile architecture, to address the mentioned issues of emergency and dangerous situations for people who are in trouble. The main aim is to enable people to send in an easy and unnoticeable way a SMS message containing their location to the Police Operating Center. To the best knowledge of the authors, this appropriate combination of advanced Android technologies proposed in this paper for the first time constructs unique, complete, and operatively usable software platform for sending emergency messages. The source code of Android application can be pulled from https://bitbucket.org/bicba90/sos_android/. The proposed platform clearly overcomes the problems introduced in previously proposed solutions described in literature.

The rest of the paper is organized as follows. In Related Work, the related works and applications are presented. In The Architecture of the Proposed Platform, the architecture and operation of the proposed solution are presented. A description of implementation of the proposed platform is presented in Implementation. The next section, Experimental Usage and Evaluation, is provided and discussed. Finally, the Conclusion is given in last section.

## 2. Related Work

Mobile applications and mobile services are becoming one of the technology mainstreams in recent years. Android-based applications are becoming a proper tool in order to solve different everyday life problems [1, 6, 7]. Recently, many researchers have tried to find a proper solution to address the security issue in the case of emergency [8–13], but there has been no proper solution.

In [8], an Android application that offers SOS message sending using the GPS location via a WhatsApp messenger to predefine recipient was proposed. To activate an SOS message sending, user has to shake his Android phone while the application is running. The main idea the author had was to enable sending of user's location via some of the modern services such as WhatsApp. However, if the application is not running, and the user is in a situation where he cannot pull out his or her phone from a pocket or a bag, this application is of no use.

Similar application intended for the cases where there is no operation of mobile communication systems was presented in [9]. The proposed application is used in a way that a group of phones, which have this application installed, create an ad hoc, a peer-to-peer-like, wireless network. On the one hand, this application is very good because in many cases, such as earthquake and other natural disaster cases, due to the damages, there is no proper operation of standard communication systems. On the other hand, this is not a good solution because it relies on a fact that each user has an installed proposed application, which could not be the case and which surely decreases the effectiveness of the proposed application. This application is also convenient for the places when there is no mobile communication signal, such as rural places. However, in this modern society, there is almost no place where there is no mobile communication signal. Due to the all abovementioned options, it can be concluded that the proposed application is good to be used in some situations but it cannot address the problem that we try to solve here. If other Android phones near us do not use this application, then this is useless, and the messages go as far as the network of phones goes, and apart from all that, in most cases when there is an emergency, the mobile network is available, so there is no need for this application.

The application that addresses the problem of the location on the roads is presented in [10]. Namely, it was concluded that road accidents are the factors that increase the mortality level. So the basic idea of the application is to warn a driver that he is approaching dangerous corner and help him slow down and prepare for the corner. The application is notifying a driver about the dangerous corner within 700 meters before the sharp corner. The warning is realized by playing "buzz" sound as the alert, to tell the driver that there are dangerous corners ahead. Besides, this system will give suggestions to the closest emergency places by only refreshing the list of the places and pin point the emergency places on map. The localization technique used in this application is very similar to those we use in our application. Both the application presented in [10] and our application use the GPS service to obtain accurate user position. However, the application [10] is customized for road transport and cannot solve several different kinds of dangerous situations.

Furthermore, there is a similar application is *bSafe* Android application [11]. This application provides more than one option to the user; namely, four services are provided: *bSafe Alarm*, *Follow me*, *Timer Alarm*, and *Fake call*. The first one is intended for sending an SMS with the user location, but besides the location, both audio and video data can be sent together with the information about the location. When this option is activated, an automatic recording is triggered and the recorded data is sent to the desired number or numbers. In this application, user can define a friend cycle containing as mush numbers as the user wants, and these numbers can be edited unlimited number of times. The second option allows tracking of the user's location on the map in real time. The third option allows user to set the time he thinks he will reach some location, and if he does not reach the location in defined period, SMS message with his current location will be sent. The last, but not the least, option provides a face call. When user activates this option, the face call is performed, which can be beneficial in the case when user wants to divert attacker's attention by letting him know that there is someone who could hear if attack happens, which could be used as a kind of evidence. However, this app is not integrated with any centralized system and is not useful in several dangerous situations where there is no time for active usage of mobile phone.

The application *GoSuraksheit* [12] is also one of the similar applications. This application is also intended for sending of the SMS with the user's location. The operation principle is the similar to the presented applications [10, 11]. The user's location is obtained by using the GPS and then forwarded to the desired numbers (up to five numbers). The main advantage of this application is that it provides a possibility to share the location on a popular social network, Facebook.

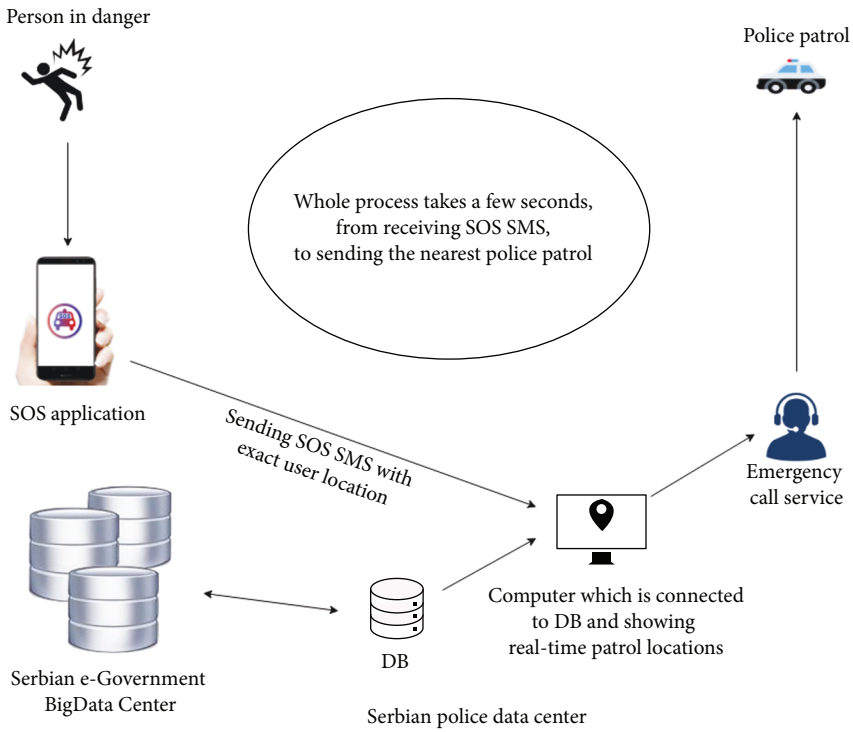



Figure 1: The architecture of the proposed platform.

The main disadvantages of the service presented in paper [12] are as follows:

(i) The service does not include any centralized logic and integration into government system

(ii) The client mobile app is not applicable in many dangerous situations

The application presented in [13] allows mobile phone user in an emergency to send the SMS to the police or rescue center. With each request, user's location coordinates are sent to the police center number. This application can operate in two modes, network mode and GPS mode. All the details of a user, such as description, priority, and current location report, are sent to the server. In other words, this app solves the problem of integration with the centralized system in order to help the citizens, but it is also inapplicable in several dangerous situations when the victim cannot actively use mobile phone.

To summarize, in order to trigger previously listed applications, it is necessary to open the application on a mobile phone which is sometimes impossible to do. The shortcomings of these applications is that there is no emergency button or trigger for fast and silent alert, even if the application has some sort of silent switch, such as phone shaking or detection of user running, which is not good because the application can be triggered accidently. In addition, some of the presented applications do not have the widget. Moreover, there is no any information about experimental usage of the proposed app [8–13] and its reaction time measurements in real-life situation. These services also cannot give any useful analytics after long-time operative usage of client app.

On the other hand, the application we propose here effectively works in the situations when the victim cannot easily and actively use mobile phone. It can even work if the screen is broken and the device is locked, and it sends messages directly to the Police Data Center. This center is integrated with Serbian e-Government BigData Center and it will be possible to apply advanced analytics solutions after the proposed service generates a sufficiently large amount of data.

## 3. The Architecture of the Proposed Platform

The architecture of the proposed application and its operation is presented in Figure 1. As it can be seen, the first step is sending of the SOS message when the user is in danger. The second one is processing of sent data at the police computer center that has a DB connected to Serbian e-Government BigData Center. This DB has a list of users and exact real-time locations of police patrols, so it can be easily calculated the nearest patrol distance. After this step, patrol is informed by the dispatch center about location where they should go. The main purpose of this process is to minimize the time from sending the SMS to getting a help from the police. By using the proposed application, in only few steps, of which the first one is triggering SMS with buttons, the second one is related to the DB with users where we already know who is sending the SMS, and the third is a determination the closest police patrol according to the obtained information on user's location (each patrol is

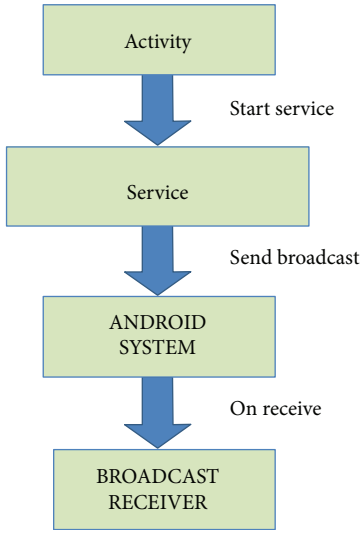



Figure 2: The architecture of the client Android application.

equipped with a GPS transmitter). The patrol should be informed where to go by the operator from Police Data Center, and that is the only part that is not automated.

The proposed architecture of the client Android application is shown in Figure 2. Three components properly combined are necessary to provide a good operation flow of suggested solution: activity, service, and broadcast receiver [14]. An activity interacts with user so it creates a window to place UI (user interface) elements. An Android application can contain several activities, which means that many different screens can interact with each other [15]. In proposed approach, user via view in activity sends a request for starting service that is responsible for getting current location. To explain, *service* is a component which runs in the background without direct interaction with the user and it is used for repetitive and potentially long running operations. As the service has no user interface, it is not bound to the lifecycle of an activity [14]. In the proposed client app, broadcast receiver is responsible for registering system events and allows reading pressed buttons for default combination as an Android component which allows you to register for system or application events. All registered receivers for an event are notified by the Android runtime once this event happens.

To sum up, implemented client solution is consisted of these three components. For release stage, SOS application also uses Google API Client library for locations ("com.google.android.gms:play-services-location:10.2.6") for getting user's location. Google Location Services API is the most popular service for adding location awareness with automated location tracking, geofencing, and activity recognition.

The proposed architecture of the client Android application also allows implementation of multiple options, such as

(i) emergency trigger—for emergency SMS sending to the emergency services

(ii) widget trigger—same as previous, but only for sending when phone is unlocked

(iii) pattern/pin—for opening and activating application

(iv) time locker—for setting application's active hours

(v) SMS remote control—if application receives a message from predefined number with certain code (#123backup), for example, it will trigger data backup (SMS, images, videos, and documents), user will be able to select what he wants to save or upload to cloud

(vi) restore/factory reset—in case the device is stolen or lost

(vii) app icon hidden in whole user system—in case the phone is stolen

(viii) camera remote control—again trigger feature remotely, this time turn camera on, take a picture with front and rear camera, and send it to predefined number

For the first release of the service, we have selected first two options to implement, emergency trigger and widget trigger. In other words, our SOS application is developed for working in two ways. First, service is implemented to have listener for buttons that user needs to press (default combination for sending SOS message). Second, service is implemented when widget has occurred. When user presses widget, it automatically sends message with its own location and any other information if the service is customized for different users.

## 4. Implementation

*4.1. Release Stage.* SOS application for sending emergency messages uses Android platform. This is an offline app, which means that it can work without access to the Internet. What this essentially means is that it gives a better experience to the users, and it can even be a key factor for them in order to retain or uninstall application. Further, the main reasons for using Android as a client platform are as follows:

(i) Open source (can be leveraged without having to worry about the licensing costs)

(ii) Dissociation of the user's interface from the business logic

(iii) Asynchronous calls (easy to code client-side multithreading)

(iv) Customizable user interface (create custom interfaces for different business)

(v) Reusable and responsive components (support for Android material)

(vi) Portability (can easily be ported to other mobile operating systems)

The list of features and advantages embedded in *Android OS* is quite long. The core release-stage functionality is to send message, prompted by the users, based on existing


Figure 3: Main screen implementation.

services implemented for listening and getting location. This client-side module provides integration of *Google Service API Location* and JSON for parsing model.

On the one hand, with the listener services used on the UI side, SOS client communicates with this service that has been started in background and performs location task and starts broadcast receiver for media. On the other hand, there is a widget, which user presses and triggers service during this process.

*4.2. User Interface.* In one word, user interface is everything that the user can see and interact with. User interface of the proposed client Android application is shown in Figure 3. Android provides a variety of prebuilt UI components, such as structured layout objects and UI controls that allow to build the graphical user interface (see Figure 4). UI in SOS application is implemented in XML, using primarily LinearLayout [15]. LinearLayout is a view group that aligns all children in a single direction, vertically or horizontally and also supports assigning a weight to individual child with the android: layout_weight attribute. This attribute assigns an "importance" value to a view in terms of how much space it should occupy on the screen.

*4.3. Widget.* App Widgets are miniature application views that can be embedded in other applications (such as the home screen) and receive periodic updates [15]. These views are referred to as widgets in the user interface, and it can be published one with an App Widget provider. An application component that is able to hold other App Widgets is called an App Widget host. Figure 5 shows implemented SOS App Widget.

As mentioned, SOS application uses App Widget declared in manifest file (see Figure 6) implemented in class *CustomAppWidgetProvider* which extends class AppWidgetProvider. The AppWidgetProvider class extends BroadcastReceiver as a convenience class to handle the App Widget broadcasts. The AppWidgetProvider receives only the event broadcasts that are relevant to the App Widget, such as when the App Widget is updated, deleted, enabled, and disabled. When these broadcast events occur, the AppWidgetProvider receives method calls such as OnUpdate() and OnReceive() method.

The proposed client application calls OnUpdate() method when the user adds the App Widget, so it should perform the essential setup, such as to define event handlers for views and to start a temporary service, if necessary. However, if you have declared a configuration activity, this method is not called when the user adds the App Widget, but is called for the subsequent updates. It is the responsibility of the configuration activity to perform the first update when configuration is done (see Figure 7).

The proposed client application calls OnReceive() method for every broadcast and before each of the rest callback methods. This method checks if location is turned on. If yes, service is starting. If not, it shows toast message for warning to turn the location on in settings (see Figure 8).

*4.4. Service.* A service is an application component representing either an application's desire to perform a longer-running operation while not interacting with the user or to supply functionality for other applications to use [14]. Each service class must have a corresponding <service> declaration in its package's AndroidManifest.xml. Services can be started with Context.startService() and Context.bindService(). Services run in the main thread of their hosting process. In this work, two services are implemented: volume service and widget service.

*4.5. Volume Service.* A volume service is one of two applications services, as we can see in Figure 9. Service can be started and stopped by the user, by pushing one of those two buttons *Pokreni servis* (eng. *Start service*) and *Zaustavi servis* (eng. *Stop service*). It is called volume service mainly because this service is working in background and waiting to capture right the combination of volume buttons sequentially pressed (at this particular implementation case that combination is set to up-down-up-down). This combination of volume buttons pressed triggers the SMS sending (see Figure 10). After service is started, location services must be enabled so that volume service can send user's location in message. Basic principle is that after user triggers the right combination, service starts with gathering information about user's location which usually takes between 1 and 5 seconds (mainly depends on connection quality at the moment of sending SOS SMS message) and immediately after that it sends message with map link. After sending the message, volume service remains active if there is need for a new send.

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@color/colorBlue"
    android:orientation="vertical"
    android:padding="16dp">

    <ImageView
        android:id="@+id/imageView"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_weight="2"
        android:scaleType="fitCenter"
        android:src="@drawable/logo"
        tools:ignore="ContentDescription" />

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical">

        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:orientation="horizontal">

            <Button
                android:id="@+id/start_button"
                style="?android:attr/buttonBarButtonStyle"
                android:layout_width="match_parent"
                android:layout_height="50dp"
                android:layout_marginEnd="8dp"
                android:layout_marginRight="8dp"
```

Figure 4: Main screen implemented in XML.


Figure 5: SOS widget.

*4.6. Widget Service.* A widget service implementation shown in Figure 11 has almost the same logic as volume service; the difference between them is that the widget service is triggered by widget button, after which is not listening for volume button-pressed combination, but instead it starts location services and sends message after finding the right location. Also, widget service is destroyed after sending message, because its only purpose is to send message in the shortest way. So, the implemented principle is that the service sends a message on location change, as we can see in Figure 12. When service finds a location, it sends a message, stops the location updates, and destroys the widget service.

*4.7. Presentation of Solution.* Final result generated by proposed client is shown in Figure 13. We can see a message which contains link to the Google maps with user's coordinates shown as a pin. Message contains text in this form—"SOS my location is *Google maps link*." With this location, user who sends an SOS message can be rescued in the short period of time. The form of SOS message can be easily customized to different kinds of app users.

## 5. Experimental Usage and Evaluation

For evaluation purposes, the platform has been tested multiple times in different real-life situations. Application is working as intended and sends a message every time after a given command.

During the period of experimental usage, the shortest reaction time was less than a minute that includes period from sending SOS SMS until the arrival of police patrol, and the longest period was 3 minutes, and that was the situation where police patrol was on the other side of their patrolling area; in other words, they were at furthest possible location from the accident (approximately 4 km). These specific numbers are roughly measured using few simulation attempts as well as some real-life situations and statistical data from the Ministry of Interior, Republic of Serbia, and they highly depend on internal police organization.

Drawbacks of the app are the locations, which are not correct in closed spaces or not even shown at all, and it is because of device's inability to find GPS satellites. Also, other issue is the high buildings with lots of floors, where the coordinates are the same for the whole building and there are lots of apartments in it, but that can be almost fixed by sending the elevation of device. However, we will still have a problem with multiple apartments, for example. So those are the few things that we have to look back at, during further development of the proposed platform.

```
<receiver android:name="com.sosService.CustomAppWidgetProvider">
    <intent-filter>
        <action android:name="android.appwidget.action.APPWIDGET_UPDATE" />
    </intent-filter>

    <meta-data
        android:name="android.appwidget.provider"
        android:resource="@xml/widget_info" />
</receiver>
```

Figure 6: CustomAppWidgetProvider declared in manifest file.

```
@Override
public void onUpdate(Context context, AppWidgetManager appWidgetManager, int[] appWidgetIds) {
    final int count = appWidgetIds.length;
    this.context = context;
    for (int i = 0; i < count; i++) {
        int widgetId = appWidgetIds[i];

        remoteViews = new RemoteViews(context.getPackageName(),
                R.layout.layout_widget);
        remoteViews.setTextViewText(R.id.textView,  text: "");

        Intent intent = new Intent(context, CustomAppWidgetProvider.class);
        intent.setAction(AppWidgetManager.ACTION_APPWIDGET_UPDATE);
        intent.putExtra(AppWidgetManager.EXTRA_APPWIDGET_IDS, appWidgetIds);
        PendingIntent pendingIntent = PendingIntent.getBroadcast(context,
                requestCode: 0, intent, PendingIntent.FLAG_UPDATE_CURRENT);

        remoteViews.setOnClickPendingIntent(R.id.start_button,
                getPendingSelfIntent(context, BUTTON_SOS));

        appWidgetManager.updateAppWidget(widgetId, remoteViews);
    }
```

Figure 7: OnUpdate method in CustomAppWidgetProvider class.

```
public void onReceive(Context context, Intent intent) {
    super.onReceive(context, intent);

    if (BUTTON_SOS.equals(intent.getAction())) {

        final LocationManager manager = (LocationManager) context.getSystemService(Context.LOCATION_SERVICE);
        if (!manager.isProviderEnabled(LocationManager.GPS_PROVIDER)) {
            context.startActivity(new Intent(Settings.ACTION_LOCATION_SOURCE_SETTINGS));
            Toast.makeText(context, "Uključite GPS/Lokacijske servise i pritisnite dugme ponovo", Toast.LENGTH_LONG).show();
        } else {
            context.stopService(new Intent(context, WidgetService.class));

            context.startService(new Intent(context, WidgetService.class));
        }

    }
}
```

Figure 8: OnReceive() method in CustomAppWidgetProvider class.

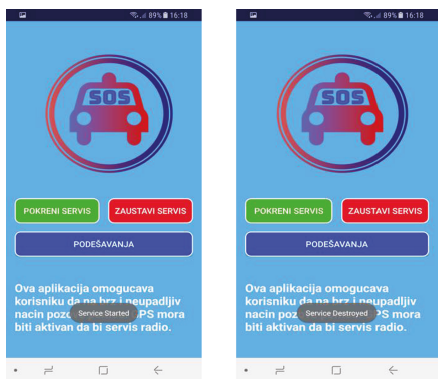



Figure 9: Main screen with start and stop buttons for service and settings button for the app.

## 6. Conclusion

In this paper, an Android-based SOS software platform has been presented. The client SOS application is designed in a way to enable user to send his location in a simple and unnoticeable way. The performance of the proposed application has been verified experimentally. The application is shown to be very useful in a large number of life-threatening situations. Moreover, this kind of applications is very suitable for children monitoring during many of their activities, such as going to the school, for a run, or during travelling, and in the situation where children find themselves in an unknown place or dangerous situation.

One of the main advantages of the proposed application is that it is directly connected to the police system and its

```
private BroadcastReceiver broadcastReceiver = (context, intent) → {
        if ("android.media.VOLUME_CHANGED_ACTION".equals(intent.getAction())) {
            int volume = intent.getIntExtra( name: "android.media.EXTRA_VOLUME_STREAM_VALUE",  defaultValue: 0);
            if (b == 0) {
                if (volumePrev < volume) {
                    b++;
                    vreme = System.currentTimeMillis();
                }
            } else {
                if (volumePrev < volume) {
                    if (tacnaKombinacija[b]) {
                        b++;
                    } else {
                        b = 0;volumePrev = 0;
                    }
                } else {
                    if (!tacnaKombinacija[b]) {
                        b++;
                    } else {
                        b = 0;volumePrev = 0;}}
            if (b == 7) {
                if (System.currentTimeMillis() - vreme < 5000) {
                    final LocationManager manager = (LocationManager) context.getSystemService(Context.LOCATION_SERVICE);
                    if (!manager.isProviderEnabled(LocationManager.GPS_PROVIDER))
                        Toast.makeText(context, "Uključite GPS/Lokacijske servise i pritisnite dugme ponovo", Toast.LENGTH_LONG).show(
                    else {
                        createLocationRequest();
                        mGoogleApiClient = new GoogleApiClient.Builder( context: VolumeService.this)
                                .addApi(LocationServices.API)
                                .addConnectionCallbacks(VolumeService.this)
                                .addOnConnectionFailedListener(VolumeService.this)
                                .build();
                        mGoogleApiClient.connect();}}
                b = 0;
                volumePrev = 0;}
            Log.i(TAG,  msg: "You have pressed " + b + " prosli " + volumePrev + " sadasnji" + volume);
            volumePrev = volume;}
    };
```

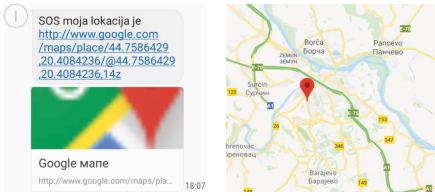
Figure 10: Implemented volume service logic.

```
@Override
public int onStartCommand(Intent intent, int flags, int startId) {
    Toast.makeText( context: this,  text: "Service Started", Toast.LENGTH_SHORT).show();


    final LocationManager manager = (LocationManager) getBaseContext().getSystemService(Context.LOCATION_SERVICE);

    if (!manager.isProviderEnabled(LocationManager.GPS_PROVIDER)) {
        Toast.makeText(getBaseContext(), "Uključite GPS/Lokacijske servise i pritisnite dugme ponovo", Toast.LENGTH_SHORT).show();
        getBaseContext().startActivity(new Intent(Settings.ACTION_LOCATION_SOURCE_SETTINGS));

    } else {
        createLocationRequest();
        mGoogleApiClient = new GoogleApiClient.Builder( context: WidgetService.this)
                .addApi(LocationServices.API)
                .addConnectionCallbacks(WidgetService.this)
                .addOnConnectionFailedListener(WidgetService.this)
                .build();
        mGoogleApiClient.connect();
    }


    return START_STICKY;
}
```

Figure 11: Implemented widget service logic.

```
@Override
public void onLocationChanged(Location location) {
    loc = location;

    //  String phoneNo = "+381652222796";
    String phoneNo =  SharedPreferencesUtil.getSettings(getBaseContext()).getPhone();
    String msg = "SOS moja lokacija je";

    String myLocation = " http://www.google.com/maps/place/lat,lon/@lat,lon,14z";
    myLocation = myLocation.replaceAll( regex: "lat", String.valueOf(loc.getLatitude())).replaceAll( regex: "lon", String.valueOf(loc

    SmsManager smsManager = SmsManager.getDefault();
    smsManager.sendTextMessage(phoneNo,  scAddress: null,  text: msg + myLocation,  sentIntent: null,  deliveryIntent: null);
    Toast.makeText( context: this, "SOS poruka poslata!", Toast.LENGTH_SHORT).show();
    stopLocationUpdates();

}

protected void startLocationUpdates() {
    PendingResult<Status> pendingResult = LocationServices.FusedLocationApi.requestLocationUpdates(
            mGoogleApiClient, mLocationRequest,  locationListener: this);
}

protected void createLocationRequest() {
    mLocationRequest = new LocationRequest();
    mLocationRequest.setInterval(INTERVAL);
    mLocationRequest.setFastestInterval(FASTEST_INTERVAL);
    mLocationRequest.setPriority(LocationRequest.PRIORITY_BALANCED_POWER_ACCURACY);
}
```

Figure 12: Implemented location update logic.




Figures 13: Example of SOS emergency message.

database, which means that a needed and necessary help can be obtained as soon as possible. Besides, the police have the information on all mobile phone users and real-time locations of all police patrols and there are a large number of highly experienced people who can help a victim. The other advantage is that almost the entire process is automated; the only thing that should be done manually by an operator at the Police Operation Data Centre is to determine which of the police patrols is the closest to the victim and to direct them to go the sent location to help a victim. This way, the time needed to react is decreased significantly, which can save many lives.

## Data Availability

The source code of the proposed application used to support the findings of this study is included within the article. Executable installation file with installation instruction can be found (https://drive.google.com/drive/folders/1WI-4KDNrd9cX49Cf3BS6qJC2f9xMr30Y). This file can be easily generated from the source code included in the article. This app can be easily integrated and tested in any police or similar system.

## Conflicts of Interest

The authors declare that they have no conflicts of interest.

## Acknowledgments

The authors would like to thank the people from the Sector for Analytics, Telecommunication and Information Technologies, Ministry of Interior, Republic of Serbia, as well as people from Government Office of Information Technology and e-Government, Republic of Serbia, for their support and constructive comments during the implementation and experimental testing.

## References

[1] G. Athanasopoulou and P. Koutsakis, "eMatch: an android application for finding friends in your location," *Mobile Information Systems*, vol. 2015, Article ID 463791, 11 pages, 2015.

[2] Y. Yan, S. Cosgrove, V. Anand et al., "RTDroid: a design for real-time android," *IEEE Transactions on Mobile Computing*, vol. 15, no. 10, pp. 2564–2584, 2016.

[3] Y. Yan, K. Dantu, S. Y. Ko, J. Vitek, and L. Ziarek, "Making android run on time," in *2017 IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, pp. 25–36, Pittsburgh, PA, USA, 2017.

[4] M. N. Soorki, M. H. Manshaei, W. Saad et al., "Collaborative real-time content download application for wireless device-to-device communications," in *GLOBECOM 2017 - 2017 IEEE Global Communications Conference*, pp. 1–6, Singapore, 2017.

[5] K. Dantu, S. Y. Ko, and L. Ziarek, "RAINA: reliability and adaptability in android for fog computing," *IEEE Communications Magazine*, vol. 55, no. 4, pp. 41–45, 2017.

[6] D. Bucerzan, C. Ratiu, and M. J. Manolescu, "SmartSteg: a new android based steganography application," *International Journal of Computers, Communications & Control*, vol. 8, no. 5, pp. 681–688, 2013.

[7] K. Bahadoor and P. Hosein, "Application for the detection of dangerous driving and an associated gamification framework," in *2016 IEEE 4th International Conference on Future Internet of Things and Cloud Workshops (FiCloudW)*, pp. 276–281, Vienna, Austria, 2016.

[8] H. Akshay Kumar, N. Divyashree, A. Nithu, R. Revathi, and Y. Suresh, "Anuti — an application to aid during emergency," in *2016 International Conference on Circuits, Controls, Communications and Computing (I4C)*, pp. 1–6, Bangalore, India, 2016.

[9] V. Tundjungsari and A. Sabiq, "Android-based application using mobile adhoc network for search and rescue operation during disaster," in *2017 International Conference on Electrical Engineering and Computer Science (ICECOS)*, pp. 16–21, Palembang, Indonesia, 2017.

[10] A. Z. Zulkafi, S. Basri, L. T. Jung, and R. Ahmad, "Android based car alert system," in *2016 3rd International Conference on Computer and Information Sciences (ICCOINS)*, pp. 501–506, Kuala Lumpur, Malaysia, 2016.

[11] "bSafe Android app," October 2018, https://play.google.com/store/apps/details?id=com.bipper.app.bsafe&hl=en.

[12] "GoSuraksheit Android app," October 2018, https://go-suraksheit.soft112.com/.

[13] R. Jadhav, J. Patel, D. Jain, and S. Phadhtare, "Emergency management system using android application," *International Journal of Computer Science and Information Technologies*, vol. 5, no. 3, pp. 2803–2805, 2014.

[14] D. MacLean, S. Komatineni, and G. Allen, *Pro Android 5*, Apress, 2015.

[15] W. Jackson, *Pro Android UI*, Apress, 2014.