An Intelligent cloud ecosystem for disaster response and management leveraging opportunistic IoT mesh networks

Conference Paper · December 2021

DOI: 10.1109/ICT-DM52643.2021.9664137

2 authors:

Jay Lohokare
Stony Brook University
9 PUBLICATIONS **134** CITATIONS

SEE PROFILE

Reshul Dani
University of California, San Diego
8 PUBLICATIONS **88** CITATIONS

SEE PROFILE

An Intelligent cloud ecosystem for disaster response and management leveraging opportunistic IoT mesh networks

Jay Lohokare
Stony Brook University
Department of Computer Science
Stony Brook, New York
jlohokare@cs.stonybrook.edu

Reshul Dani
University of California San Diego
Department of Computer Science
San Diego, California
rdani@eng.ucsd.edu

Abstract*—Access to emergency services like police, fire, rescue, and EMS is life or death during natural disasters. Disaster management faces three critical problems for emergency services - technology constraints (Network infrastructure), demand-supply management (a large number of victims to respond to, but limited on-field agents), information access (for on-field agents). In this paper, we present an end-to-end framework to enable reliable disaster response for emergency services. This framework solves the three problems described by introducing a unique system of collecting SOS messages from disaster victims, presenting and aggregating the messages to control center operators, and making this data alongside various offline tools available to on-field agents. The framework leverages a combination of Ad-hoc mobile networks based on widely used/readily available protocols and hardware to solve the technology constraints. We introduce a novel smartphone-based mesh network that leverages the radio modules already present in smartphones (BLE, Sound, Wi-Fi, Bluetooth) to complement custom hardware-based mesh networks (based on LoRa). SOS messages travel over multiple smartphones until they reach an internet-enabled device. On reaching the internet, we use contextual intelligence for determining the request context and helping emergency service agents prioritize and solve the request. We design an intelligent interface for control center agents to get an aggregated view on disaster victims and on-field agents, helping them make data-driven decisions to help the victims. The framework also provides the on-field agents with an interface to access data and communicate with the disaster victims, even in offline conditions leveraging the mesh network. The critical contribution of this paper is the framework's three-prong approach to support the victims, control center operators, and on-field agents. We present a walk-through for a pilot deployment of our framework alongside its qualitative and quantitative results and show how it can integrate with services like 911.

***Index Terms*—Disaster management, emergency services, mobile ad hoc networks, IoT, mesh networks, opportunistic MANET, Smart phones, LoRa, Artificial Intelligence, Mobile and wireless communication networks, API standards, emergency platform, pervasive networks, ubiquitous networks, cloud**

978-1-6654-3285-6/21/$31.00 ©2021 IEEE

I. Introduction

Emergency services like Police, Fire rescue, EMS form a critical backbone of our society. With the growing population and climate change leading to increased life-threatening incidents, the existing emergency services are getting overwhelmed. Due to their inherent underlying technologies, these services fail to perform in situations involving network failures or constraints [1]. Moreover, the current emergency management process is highly manual and is not data-driven. Control center agents don't have a view of all the user requests (demand) and available on-field agents (supply) who can be deployed, making the process sub-optimal.

We categorize shortcomings of existing emergency helplines into the following categories:

- Dependence on cellular networks - Today, emergency helplines rely heavily on traditional communications networks (cellular towers) for supporting voice calls, which is very vulnerable to natural disasters (wildfires, hurricanes, flooding, earthquakes).
- Lack of automated reporting/data analysis - Given the systems are voice call based, analysis of requests served by the emergency service is not centralized.
- Lack of interoperability between various emergency services - The current tool are siloed (voice calls, management databases, communication tools) with no aggregated view available.
- Lack of streamlined dispatch process - The current dispatch process is inefficient as it doesn't leverage GIS data based optimizations.
- Lack of information access for on-field agents - A digital interface with capabilities to directly communicate with or accurately locate the victims doesn't exist.

We propose a three-component framework to overcome the drawbacks mentioned. The first component is a combination of mesh networks using opportunistic routing to enable seamless communications - A smartphones based mesh (Wi-Fi, Bluetooth, BLE) and a IoT devices based mesh (LoRa, Wi-Fi, BLE). The second component is a cloud-based engine to

capture SOS messages, analyze and route them contextually to appropriate agencies. In addition, the engine generates insights and recommendations to provide data-driven context to control center operators and optimize the on-field agent dispatch process. The third component of the framework is an interface for emergency service operators and agents to act upon incoming requests, access data, communicate and manage the emergency case.

Here are the key technical contributions of this paper:

1) A novel smartphones-based mesh Ad-hoc peer-to-peer network which forms the backbone of access to emergency services even in offline conditions.
2) The LoRa based mesh network which is formed by easy to deploy IoT devices and its interoperability with the smartphones-based mesh network.
3) The proposed way of making the smartphone mesh network the default SOS feature.
4) The cloud engine which enables such SOS systems co-exist with traditional voice based emergency response platforms, aggregates various data sources and enables seamless collaboration between various emergency services like police, EMS, firefighters.
5) The simplified interface for emergency service operators which enables easy dispatching of on-field agents without having to call or connect with them.
6) Application to enable on-field agents to locate disaster victims, provide them with real-time data, and allow offline communication.

To the best of our knowledge, this paper presents a first-of-its-kind end-to-end framework to enable communication and data transfer between disaster victims, control center operators, and on-field responders even when traditional networks are unavailable. Though we leverage existing literature on wireless mesh networks, our work focuses on how such technologies can enable emergency service providers in disaster situations. The emphasis on user experience for all parties involved, the use of hybrid Adhoc networks to relay SOS messages, the end to end system design for three-way emergency response (user, control center service agents, on-field service agents), and use of data for optimal dispatch decisions forms the core of our novel contributions.

II. Related Work

Communications systems are failure prone and become the primary bottleneck in disaster response. Wireless Mesh Networks have an increasing popularity in literature for their use during disasters to tackle the network failures [4], [22], [23]. They provide immediate internet connectivity and can be deployed on-demand to disaster-affected regions. Various solutions have been proposed to establish networks on-the-fly [5], [7]. While most literature focuses on the mesh network architecture [6], routing protocols [8] and traditional schemes for establishing a network, our work provides a holistic view on how such networks can form the backbone of a larger ecosystem for disaster response. We leverage existing works on bringing movable and deployable networks to the disaster-affected area [11], [24]. We create a mesh network based on smartphones similar to [14] and [16] which use relay-by-smartphone to utilize the Wi-Fi functionality of smartphones to relay messages between devices. We draw from a system called TeamPhone [15] for the ad-hoc routing implementation on a network created by using smartphones. Finally, we extend the above research to use Bluetooth, BLE, and sound as additional mechanisms for P2P communication [9]. Furthermore, to increase the range of our mesh network, we extend our standard mesh interface to use LoRa for secure, long-range, and low power data transmission [19]. Existing research evaluates the use of LoRa for incident response as an emergency communication tool [17], [18]. While this is very effective as a low-power solution, there is an additional deployment cost for these devices in the affected area. Our use of a hybrid/heterogenous mesh based on readily available smartphones alongside the LoRa devices offsets the individual mesh networks drawbacks while enabling reliable communications. Our primary focus is to build a modular system of mesh networks, defining a standard interface to bring together various point solutions.

The existing research on multi-technology access points [20], [21] is geared toward network formation and deployment. However, the work fails to address non-network related aspects of disaster response and management. Our work aims to cover the big picture of emergency service platforms including but not limited to network deployment, resources management, optimal on-field agents deployment, data access. There are numerous emergency response platforms serving point use cases. For example, 911-network on wheels (911-NOW) [12] attempted to create a portable cellular system and deploy 911 services (ensuring high availability for the platform). Jun et al. [13] focuses on a service-oriented architectures to support disaster response, enabling modularity. Lohokare et al. [10] presents an optimal agents deployment mechanism by leveraging GIS data. While these ideas and frameworks are novel, these works consider only one part of the whole solution for disaster response. These works fail to present a end-to-end view on the emergency service platform of the future and they don't provide a solution that covers all aspects of the user journey for the 3 persona involved in disaster response.

III. Background

A. Design Considerations

We consider the following aspects as the building blocks for our platform:

Works without cellular network. Be resilient to network failures (power loss/physical damage).

Low Cost and easy deployment. Any hardware components or cloud micro-services designed and deployed need to be low-cost and have a fast deployment time.

Enables on-field communication. Enable on-field communication between disaster victims and on-fields agents alongside assuring delivery or alert and SOS messages.

Aids agents in making quick decisions. Provide control center operators with an easy to use interface to communicate with victims, access aggregated data and support on-field agents.

Modular. Create a standard for data transfer, communications protocols and case management for easy integration with existing systems.

Scalable. Design for world scale, sudden spikes in traffic, concurrency while assuring persistence and data integrity.

B. User Persona

The framework we propose aims to serve 3 key persona during disasters:

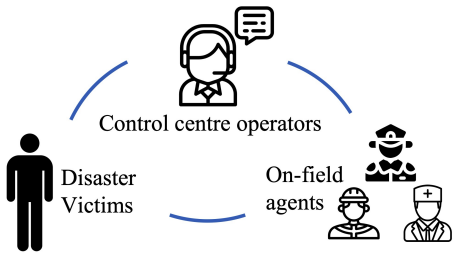

Fig. 1. The three parties served during disasters.

Disaster victims - Citizens directly or indirectly impacted by disasters.

Control center operators - These are the operators who currently handle voice calls landing on emergency services helplines. Our framework entails that these operators will also handle SOS messages from systems extending beyond voice calls.

On-field agents - These are the firefighters, medics, and police personnel on-ground during and after disasters. The framework we introduce makes a key assumption that the on-field agents have access to smartphones.

IV. Framework Components

This section presents a walkthrough of the system's design and details on the technical implementation of its various components.

A. Mesh Network

We use the concept of a 'deployable wireless communication infrastructure' which ensures connectivity even when traditional networks(cellular, internet, telephone) are unavailable due to power loss/physical damage. We do this through a unique combination of two Mesh networks, which we will describe further. This mesh solves step one of disaster management - enabling critical network capacity through offline first communication. Our mesh network design is based on the mesh networks introduced in [15], [17], [20]. We build up on the existing work to further enhance the systems and optimize them to work alongside emergency services.

1) Harwared-based Mesh: We deploy wireless IoT devices to the disaster-affected area to rapidly create Mesh networks on the ground. The basic hardware unit used for creating this mesh was an ESP32 IoT device, and these can be deployed to the affected areas by first responders or even via drones. At any given time, a device is trying to receive messages from various radios and save it to its file storage. It also keeps pushing the entire set of messages to peer devices on the network or the internet. Let us look at how these devices communicate with each other and the outside world using five different threads.

Our devices talk to each other using 915Mhz LoRa low power radio for sending and receiving data. An open-source firmware for mesh network creation based on LoRa was used, which can be used to establish a long-distance network over the affected area and enable message routing described in the next section.

After enabling our devices to communicate over LoRa, we implement a Wi-Fi server to make a Wi-Fi endpoint available to the disaster-affected population. On connecting, users are redirected to a web App (running on localhost) that delivered as a landing page for the Wi-Fi network, removing need for additional app installs. The landing page serves a bare minimum chat interface along with a simple menu to send SOS alerts. Finally, the data captured on this web app is converted to messages to be transferred over the mesh/internet gateway.

For information to reach the outside world, we need one of our devices to be connected to the internet. We call this the 'internet gateway', through which all deployed devices get the ability to upload messages to the internet. For example, the device closest to a network tower can transmit all data from the local network to the internet (when it gains connectivity). The Internet gateway thread actively tries to push all messages/data available to the cloud based message broker via a MQTT client.

Furthermore, the ESP32 devices also connect to smartphones from the smartphone-based mesh described below. They receive data from smartphones via BLE and forward it to other ESP32 devices via LoRa. Eventually, when the data reaches an internet gateway, it is sent to the cloud.

2) Smartphone-based Mesh: Having just the LoRa based mesh faces many problems. These devices need to be deployed into the disaster area, which is hard based on the type of the disaster (e.g. floods, hurricanes) and required resources (e.g. time, effort and equipment like helicopters/drones). Moreover, these devices are limited by their battery and can often get lost/ easily damaged in the affected area. To solve this, we propose using devices that are readily available even in disaster-affected situations - users' smartphones. We create a smartphone-based peer-to-peer Ad-hoc mesh network based on smartphones. It uses Wi-Fi, Bluetooth, Bluetooth Low Energy (BLE), and Sound on smartphones for transferring data. Nodes in this mesh (Smartphones i.e. disaster victims) can move around, carrying data with them. Smartphone Mesh thus enables mobility - first responders can sync messages from the community on the Mesh nodes on their smartphone and then sync these messages with the internet when their

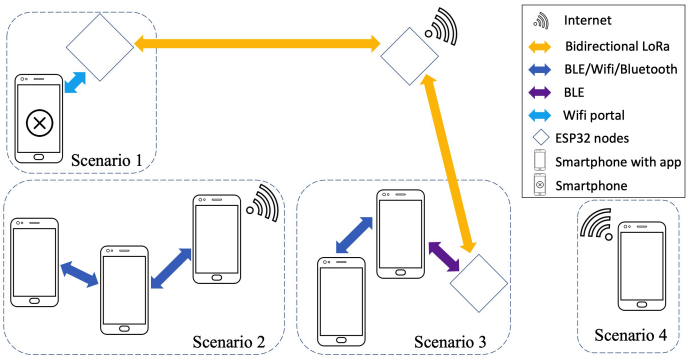

Fig. 2. The four scenarios of communication in the hybrid mesh network.

phones get online. When combined with hardware-based mesh networks running long-range low bandwidth protocols, this system ensures high availability of network access.

Moreover, instead of asking disaster victims to install an app, we propose that this mesh network be a default feature pre-installed in smartphones. As a default 'emergency' feature, the smartphone-based mesh network will ensure that alerts always reach everyone and users can send SOS messages and be assured about their delivery to appropriate authorities. The mesh is Adhoc i.e. it allows free movement of smartphones involved. Users download the app and forget about it, and they will still be helping communities to enable communication. Every single smartphone will act as a mobile tower. The working of this smartphone-based mesh network is very similar to the hardware-based mesh. There is a shared storage layer, an internet gateway, and P2P communication via available physical protocols, all running over background services. We used Google Nearby Connections API and custom BLE and Wi-Fi clients to create the P2P communications layer. The smartphone mesh potentially has an infinite overall range, with the individual node having a range of (Wi-Fi - 400m, BLE/Bluetooth - 150m, Sound - 30m).

In Fig. 2, scenario 1, the disaster victims do not have the SOS application installed on their phones. So they connect over Wi-Fi to the IoT device we deploy, where they can request help over the web interface served. Scenario 2 depicts an SOS message traveling over multiple smartphones before reaching a smartphone with internet, which will push all the messages in its cache to the cloud. Scenario 3 is where the SOS messages first travel over the smartphone mesh and then are sent to the IoT device over BLE. The IoT device then forwards the messages over LoRa to its peers, where an internet-enabled peer uploads the messages to the cloud. Scenario 4 is where users can directly call the emergency services or send SOS messages over the internet (via MQTT) as they have internet access and our application installed.

B. Message Format and Routing

We have designed the message routing to be agnostic of underlying physical protocol, and we handle all the routing logic at the application layer. This enables the framework to be flexible and easy to extend to other physical layer protocols or systems. This routing logic is loosely based on the Ad-Hoc Distance vector routing (AODV) protocol - Communication is JSON based, and every node is stateless as messages contain routing and state data. All devices on the various mesh networks keep searching for peer devices over available physical protocols and establish connections with them in a round-robin fashion to transfer data. To ensure that device discovery is successful, we force all devices to follow a fixed peer discovery cycle. We sync up peer discovery for all devices by making them start the discovery process simultaneously. This is achieved by starting discovery when [systemTimestamp % cycle frequency = 0]. Given that all devices have correct clocks set, this approach assures that the mesh networks always are in a predictable state. All devices use a pub-sub based MQTT protocol for sending messages to the server. The choice of MQTT was due to the low latency and low bandwidth requirement of the MQTT protocol and its dynamic topic-based push messaging nature. We use specific MQTT topics to establish stateful data transfer. A message from a particular device is always published on a particular topic, and the corresponding response is received by subscribing to another topic -

$Publishtopic - receive/\$userID$

$Subscribetopic - response/\$userID$

Once devices are aware of their peer devices, they start sending data in a round-robin fashion. Nodes in the mesh first attempt to send data directly to servers via MQTT. If this fails, the message is added to Cache, and the Mesh transfer services take over to transfer the cached messages to peers see Algorithm. 1. Message send format is as below(in JSON):

Send message format
{
"key": $messageID *($userID_$timeStamp)*,
"userID":$userID *(e.g mesh:3241:johndoe123456789)*,
"timeStamp":$timeStamp *(Timestamp when message was generated e.g 982823474842)*,
"destination":$destination *("cloud" if destination is the cloud, $destinationUserID if message is sent to someone within the mesh)*,
"message":$messageContent *(The actual user message content)*,
"priority": $priority *(1 for alert broadcast and SOS messages, 2 for everything else)*
}

Algorithm 1 For each message in cache

1: If message has expired, delete the message from cache.
2: If destination for the message is cloud, try uploading the message to the server via MQTT. Delete message if successful.
3: Check if the message destination is the same as the current device's unique ID. If so, delete the message from cache and create a notification saying a new message was received.
4: If the message is still in cache, send the message to all nearby devices via any available communication radio.

The smartphones and hardware devices use this algorithm to process and route messages. A message enters the system

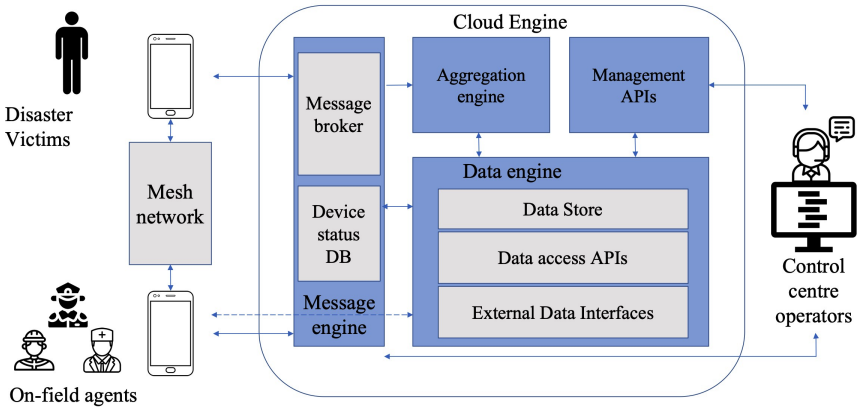

Fig. 3. Overall System Architecture.

when user creates a request (SOS or chat message). A response message/alert from the server will enter the Mesh either through an online phone (via MQTT) or one of the hardware devices connected to the internet. The process for routing messages to a target device is pretty much the same as routing messages to the server. However, the server response message has a slightly different format to differentiate messages originating from users from messages from the server.

Receive message format
{
"key": $messageID *('cloud_$destinationUserID_$timeStamp' e.g cloud_mesh:johndoe123456789_982823474842)*,
"userID": "cloud",
"timeStamp": $timeStamp *(Time stamp when message entered mesh e.g 982823474842)*,
"destination": $destinationUserID, *(As defined in UserIds e.g mesh:3412:johndoe123456789)*,
"message": $messageContent,
"priority": $priority *(1 for alert broadcast, 2 for all other messages)*
}

This standard format is fundamental to our platform's ability to extend to new systems. The message broker can identify the source and target platforms by parsing userID, which is of format [platform]:[uniqueID]. For mesh networks, we use 'mesh' as an identifier for the platform. The application for disaster victims asks them to select a username before starting an interaction (or during setup). It is hard to broadcast data for checking if the userId selected is unique, so we append the device unique ID to the userID such that the final format is [platform]:[deviceId]:[uniqueId].

Priority 1 represents the SOS messages sent by users or alerts sent by control centers in the mesh network. These messages never expire (Infinite TTL). Any direct chat communications are given lower priority and a set TTL (12 hours) as they utilize a significant chunk of the limited bandwidth available on mesh networks. The priority flag is not applicable when the internet is available but can be used to decongest the message brokers in peak traffic.

C. Cloud Engine

The cloud engine is the core backend of the proposed framework, which consists of a set of Python Flask APIs. We have structured the APIs as isolated microservices to enable overall horizontal scalability and achieve scaling up/down individual components. For our pilot deployment, we used IBM cloud to deploy the API containers over Kubernetes. The backend is divided into four key microservices:

Message engine - This microservice presents a publish/subscribe message broker and a database to maintain device status and last known location/timestamps to enable direct communications. The SOS requests coming from users are captured by an MQTT client, which then forwards the request to the data store microservice. We have built the message broker using hiveMQ based on the MQTT protocol. We use POSTGRES for our device status database implementation. We modified the MQTT broker to allow the broker to directly send all incoming messages to the data engine without using an additional client listening to the published messages. The key reason to maintain a separate table with device status is to enable message routing without looking up the data engine. Whenever a node in the mesh network gets access to the internet, it pushes all the data (SOS requests, heartbeat messages) in its cache to the internet using the message engine. Similarly, control center operators can send alerts to victims or data/messages to on-field agents using this broker. The message engine gets heartbeats from all devices (directly connected or connected via mesh networks). This enables the message broker to maintain a map of unique device IDs, last seen timestamps, and last known location. The device ID and connectionID allow sending messages to specific users.

| deviceID | location(lat, lon) | timestamp | connectionID |

Aggregation engine - The aggregation engine is a set of Spark jobs running to aggregate SOS requests in the data store microservice and aggregate the requests based on the type of request, location of requests, urgency, etc. These data pipelines are the ones that generate aggregated data that the control center operators access. The aggregation engine runs

aggregation jobs on the data in the datastore and stores the results back for later use.

Data engine - This microservice is a set of databases the system relies on for storing data and python flask based APIs to manage access control. We made an explicit design decision not to expose any databases directly to other microservices and wrap the access in APIs instead to assure proper access control and form a uniform interface to all datasets (internal or external). We use POSTGRES to ensure consistency in data for SOS requests captured and MongoDB for delivering a highly available metadata store. In addition, we created a data interface i.e. a configurable set of ETL jobs, to sync existing systems like medical records, maps, etc with our data store. When a case is assigned to an on-field agent, the control center operator enables an access flag for the particular disaster victim's data for the agent. The flag expires in a certain amount of time or when the case is resolved, making the data access APIs revoke access for the on-field agent to ensure privacy.

Management APIs - These are the set of python flask APIs connecting the web interfaces used by control operators to the data store. These APIs are responsible for data access, access management, and user management.

D. Applications

To enable fast and easy disaster response, we need easy-to-use interfaces for all the parties involved. The applications we created focus on minimizing user inputs while maximizing features and data available. We propose that these applications will be deployed before disasters and can co-exist alongside systems like 911. This can be done by making appropriate hardware available to host our interfaces and training the personnel involved to use them. We have three different sets of user interfaces created targeting the three user persona served by the framework.

Smartphone application for disaster victims - The application serves two purposes - It deploys backend services that enable the smartphone to act as a node in the mesh network. The application also provides disaster victims an interface to send SOS messages and communicate with emergency service agents. To make it easy to send SOS messages, we made the SOS feature easy to trigger - victims can click the power button five times to trigger the distress menu. We propose that the features presented by our framework should be the default SOS functionality for smartphones. When users open the SOS menu, they are presented with an option to send an SOS message. If a cellular network is available, this will start a voice call with emergency providers selected. We allow users to pre-set a default emergency service (police, ems, fire) or select one when they decide to send a SOS. If a cellular network is not available, we automatically route the user request to the mesh networks.

We have demonstrated the flow in Fig 4. The 1st screen in the figure is the default SOS functionality that all smartphones come with. The 2nd screen shows the optional menu served to users. If a default provider has not been set, disaster victims can choose what service to call/request over the mesh network.

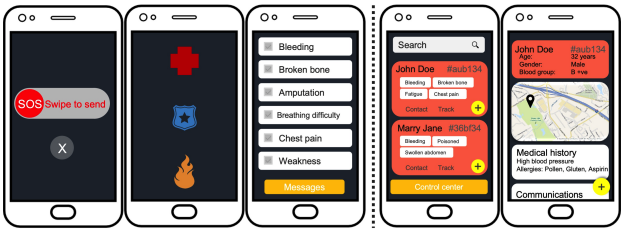

Fig. 4. Application interfaces for disaster victims and on-field agents.

Finally, the 3rd screen shows an optional menu to select options relevant to the help needed. Selecting from the menu is optional and not time bound as we want to minimize the number of inputs we ask disaster victims to make. However, adding these data points makes it easier for control operators to make contextual decisions, so we encourage users to enter as much information as they can.

Smartphone application for on-field emergency service agents - This application is created with the primary purpose of enabling on-field communication for emergency service agents. The application lets the agents view the cases assigned to them.

In Fig 4 we demonstrate the two key interfaces available to on-field agents. On the first screen, the agents can see all cases assigned to them. They can get an overview of the case by looking at the tags assigned to the case and can track or communicate with the victim. The second screen shows the detailed data view available to aid the agents. This view provides them with data from various external data systems

Web interface for control center operators - This interface serves as the control panel for emergency response in the disaster-affected area. The web app gives control agents tools to communicate, track, monitor, and fetch data related to ongoing emergencies in their area. The control panel comes with a simple user management system to enable multiple agents to manage emergencies. In addition, the system ensures equal and intelligent location-based distribution of cases between agents. This is achieved by a simple clustering algorithm run on the incoming requests to allocate the request to a location cluster and thus the agent handling the cluster. The management API microservice handles all the functionality of this interface and its backend.

The interface offers the following features as seen in Fig 5:

- Send alerts to users in a particular area - The agent can select a polygon on a map interface corresponding to the area the alert is targeted for. We then run a 'polygon contains' query on the location data for all users, find device IDs for relevant users, and send alerts using the message broker.
- Responding to emergencies - The interface provides a simple list view for all ongoing/allocated emergencies in the area. The operator gets a view of relevant data like location, last seen, type of emergency, nearby on-field agents, requester's history from various databases, all in

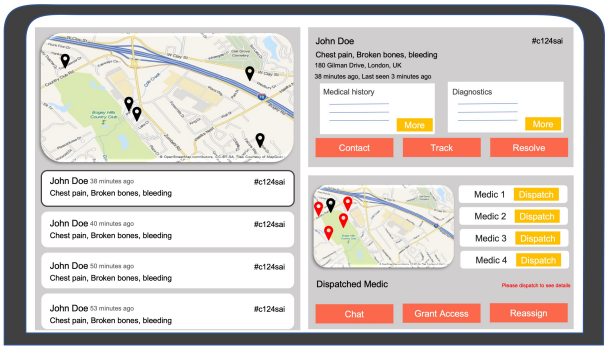

Fig. 5. Web interface for control center operators.

a unified dashboard. The operator can allocate on-field agents, monitor the case status, chat with the on-field agent and disaster victim, approve data access for the on-field agent, all through this unified dashboard.

- Monitoring emergencies in the area - The operator also gets an aggregated view on all emergencies ongoing in the area, with detailed statistics around emergency type, on-field agents availability, resolution time, etc. This is where the data created by our aggregation engine provides new data points to the operators.
- Managing cases - Create new cases based on calls coming over helplines, assign and update the status of cases, take notes corresponding to cases.

V. Pilot and Performance

To study the feasibility of the proposed framework and to evaluate the performance of its underlying technologies, we ran a pilot deployment of our system, and ran various tests to simulate disaster scenarios. We deployed the cloud-based microservices on a Kubernetes cluster on IBM cloud, using B1.1X2 instances [2]. For deploying the LoRa mesh, we deployed a fleet of 25 ESP32 devices running our mesh network codebase, with every node placed 8 miles away from the other, forming a honeycomb mesh formation. We also deployed a smartphone-based mesh network with 12 Samsung galaxy note 10 devices forming a similar honeycomb formation, with every device 10 meters away from the other. These ten devices represent the disaster victims sending SOS requests to our backend. Finally, we ran various experiments on our mesh network to study the performance of the mesh networks:

Experiment 1: Messages count vs. time for delivery of a single message - We studied the behavior of the mesh network comparing the average time for sending a unit message (100 plain string characters) with variation in the number of messages and the number of devices concurrently connected to the sender. The key takeaway for this experiment was that the time per unit message is constant irrespective of the number of devices on the network or the number of messages being transferred.

Experiment 2: Overall time for message processing and delivery with change in the number of messages and number of devices - We ran a script that forced devices to send messages over the network in order to study the network performance with an increasing number of messages. Overall, we found a linear correlation between the number of messages and the average time for message delivery. The change in the number of devices does not lead to a significant change in time for delivery. We attribute the slight difference seen in time taken with an increase in the number of devices to the time a device spends in de-duplicating messages in its cache.

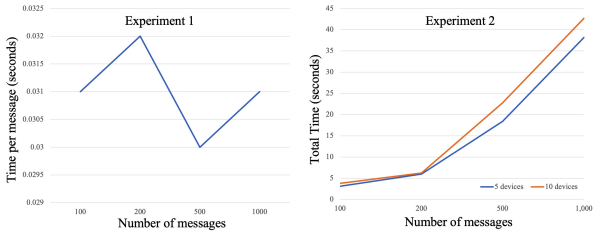

Fig. 6. Performance with increasing number of devices and messages

To evaluate our system's ease of use and adoption, we also conducted a user study on a group of emergency service on-field agents and control center operators. We introduced 20 control center operators (12 from the USA, 8 India) to the web interface we developed and gave them a week to try out its features and evaluate its feasibility and ease of use. After this 'demo' period, we sent out a survey asking them to provide their views on the system. Similarly, we conducted a similar experiment with on-field emergency service agents i.e. police, medics, firefighters from India and the USA. The distribution was police officers - 6 USA and 4 India, medics - 2 USA and 8 India, firefighters - 5 USA and 5 India. We gave them a week to use and evaluate the platform's features and sent a similar survey to capture their qualitative opinion on the framework. Table I presents the responses we received on whether the framework improves upon a criterion compared to existing systems.

TABLE I
Qualitative User Study

	CCO			OFA		
Criteria	D(%)	N(%)	A(%)	D(%)	N(%)	A(%)
User interface	0	45	65	10	30	60
Access to data	5	55	40	5	30	65
Effectiveness in helping disaster victims	0	25	75	0	45	55
Speed of working on a case	0	75	25	0	60	40
Ease of learning	30	30	40	40	20	40
Ease of adoption	35	25	40	45	35	20
Communications	0	15	85	0	25	75
Victim's trust in the system	25	40	35	15	50	35

OFA: On-field agent, CCO: Control center operator.
D: Worse, N: No improvement, A: Better (than existing systems)

We also studied how our cloud microservices scale with a growing number of messages reaching the servers. We stress-

tested the microservices by using scripts to simulate a large number of messages reaching the servers. In our experiments using a single B1.1X2 server per microservice, we observed that the message broker was the primary bottleneck for the cloud backend as it handles the maximum amount of concurrent traffic. With a single server hosting the message broker, our framework can handle a maximum of 11050 concurrent device connections, each sending a message randomly in 5-second cycles. We consider a latency of 40 ms as the upper limit for healthy server health. Practically, the same servers can handle a much higher load but with worse latency. We also studied the horizontal scalability of our cloud backend by comparing average latency for successful responses sent by the message broker with an increasing number of server instances and observed a near-linear scalability factor for the broker.

VI. Discussion

Network Infrastructure scalability - Based on the study of AODV and similar Ad-hoc routing protocols in literature [3], we anticipate our mesh networks to have a theoretical limit to their scale. With the increasing number of devices and messages sent by the devices, we expect degrading performance. However, given the comparatively small scale of our pilot, we did not observe any performance degradation leading to increased time for delivering the SOS messages. We consider large-scale performance evaluation of the mesh networks (LoRa + smartphone) as future work for our framework. To tackle the mesh network limitations arising from the underlying bandwidth limits, we plan to study the use of various 'edge' optimizations in our future work. Our current implementation uses plain text string for data transfer, which can be improved by experimenting with various data compression techniques [25]. The message routing on the network can also be optimized by creating 'sections' within the mesh. Such sections will each contain a fixed number of devices, forming a sub-mesh. We can design the deployment of hardware devices or internet gateways so that every sub-mesh has an internet gateway and can be isolated from other sub-mesh in the overall network.

Cloud scalability - In this paper, we already have presented a view on the horizontal scalability of the microservice architecture presented. We can further enhance the performance by scaling across geographies to keep the microservices geographically closer to the target users. Our backend will typically see spikes in areas that have an ongoing disaster. So we can pre-scale server capacity when disasters are anticipated.

Offline capabilities - Though the framework we presented focuses on emergency response optimization, the underlying technology has many other use-cases and can be scaled up to support many point solutions. The app pre-installed on smartphones can serve as a digital vault for medical records, which users can share with first responders even in offline conditions. Though we currently propose that the control-center operators handle SOS messages, we can use AI to extract context from the messages and automatically deploy on-field agents to help the victims. Control center operators can play the role of auditors in such automated systems.

IoT device design and deployment considerations - Optimal deployment of the IoT devices is a unique problem to solve due to the various factors that affect the mesh performance - Distance from users, line of sight to other devices in the mesh, obstructions, ease of access, protection from natural elements. We can deploy the devices by asking the first responders to place them in pre-selected locations or air-dropping them via drones. Keeping the cost of IoT devices low is crucial (hence the choice of low-cost components like ESP32) as it is likely that the devices will be lost or broken within a couple of weeks of their deployment. We can build sturdy enclosures to keep the ESP32 devices in, but that could have cost implications that we need to optimize. Similarly, the ESP32 devices can not last forever as the battery providing them with power can run out. We used a 500mAh battery for the pilot deployment, which powered our ESP32 devices for an average of 81 days. These batteries contribute a significant amount to the overall costs of the mesh network, so we need to be able to retrieve and recharge them for re-deployment. We propose using solar-powered ESP32 devices as a viable solution to fix the battery constraints.

Application availability - Making the application powering the application a default feature on all smartphones is a challenging task. We either need to partner with all smartphone manufacturers, with operating system providers (E.g., Apple, Google), or partner with governments to pass regulations to enforce the application as a default standard. For an interim solution, we propose that emergency services send application install links to citizens when a disaster is anticipated to install the app and onboard the mesh network.

Security and Privacy - In the system implementation details, we have already addressed concerns around access control for data accessible to on-field emergency providers. However, our mesh network implementation and message routing mechanism need further work to ensure privacy and fine-grain access control. As message routing is highly dependent on message content, we can structure the message to isolate the user-specific data from routing data. For our future work, we aim to explore various ways of making the framework end to end secure (ensuring only operators can access the data sent via SOS messages) and explore how we can encrypt the messages on our mesh networks.

VII. Conclusion

In this paper, we propose a framework for emergency services to enable disaster response and management. To make disaster response resilient from network failures, the framework introduces a novel hybrid ad-hoc network leveraging smartphones and IoT devices as nodes in a mesh. The framework is end-to-end i.e. it focuses on all the three aspects of emergency response - capturing SOS messages, enabling optimal control center operations, and ensuring efficient deployment of the on-field agents. We emphasize the importance of a fluid and seamless user experience for all user persona

interacting with the system through this framework. The framework is meant to co-exist alongside existing helplines like 911 but can be the primary platform for emergency services. The paper describes the framework's modularity and provides a walkthrough of how existing platforms, data sources, interfaces can be onboarded. Finally, we present a qualitative view on the framework by conducting user research on a group of emergency service operators/agents while also providing the performance-related results of experiments we ran as a pilot.

REFERENCES

[1] Manoj, Balakrishan S., and Alexandra Hubenko Baker. "Communication challenges in emergency response." Communications of the ACM 50.3 (2007): 51-53.

[2] IBM Cloud Virtual Servers https://cloud.ibm.com/docs/virtual-servers?topic=virtual-servers-about-virtual-server-profiles

[3] Verma, V.K., Singh, S., Pathak, N.P., 2014. Analysis of scalability for aodv routing protocol in wireless sensor networks. Optik-Int. J. Light Electron Opt. 125 (2), 748–750.

[4] Yarali, Abdulrahman, Babak Ahsant, and Saifur Rahman. "Wireless mesh networking: A key solution for emergency and rural applications." 2009 Second International Conference on Advances in Mesh Networks. IEEE, 2009.

[5] Murugeswari, R., and S. Radhakrishnan. "Reliable data delivery for emergency and disaster recovery in wireless mesh network." International Conference on Information Communication and Embedded Systems (ICICES2014). IEEE, 2014.

[6] Iqbal, Muddesar, et al. "SwanMesh: A multicast enabled dual-radio wireless mesh network for emergency and disaster recovery services." J. Commun. 4.5 (2009): 298-306.

[7] Minh, Quang Tran, et al. "On-the-fly establishment of multihop wireless access networks for disaster recovery." IEEE Communications Magazine 52.10 (2014): 60-66.

[8] Pirzada, Asad Amir, et al. "SafeMesh: A wireless mesh network routing protocol for incident area communications." Pervasive and Mobile Computing 5.2 (2009): 201-221.

[9] Baert, M.; Rossey, J.; Shahid, A.; Hoebeke, J. The Bluetooth Mesh Standard: An Overview and Experimental Evaluation. Sensors 2018

[10] J. Lohokare, R. Dani, S. Sontakke, A. Apte and R. Sahni, "Emergency services platform for smart cities," 2017 IEEE Region 10 Symposium (TENSYMP), 2017, pp. 1-5, doi: 10.1109/TENCONSpring.2017.8070058.

[11] Sakano et al., "Bringing movable and deployable networks to disaster areas: development and field test of MDRU," in IEEE Network, vol. 30, no. 1, pp. 86-91, January-February 2016, doi: 10.1109/MNET.2016.7389836.

[12] D. Abusch-Magder, P. Bosch, T. E. Klein, P. A. Polakos, L. G. Samuel and H. Viswanathan, "911-NOW: A network on wheels for emergency response and disaster recovery operations," in Bell Labs Technical Journal, vol. 11, no. 4, pp. 113-133, Winter 2007, doi: 10.1002/bltj.20199.

[13] W. Jun, "Using Service-Based GIS to Support Earthquake Research and Disaster Response", Comp. Sci. Eng., vol. 14, no. 5, pp. 21-30, Sept.–Oct. 2012.

[14] H. Nishiyama, M. Ito and N. Kato, "Relay-by-Smartphone: Realizing Multihop Device-to-Device Communications", IEEE Commun. Mag., vol. 52, no. 4, pp. 56-65, Apr. 2014.

[15] Z. Lu, G. Cao and T. La Porta, "Networking smartphones for disaster recovery," 2016 IEEE International Conference on Pervasive Computing and Communications (PerCom), 2016, pp. 1-9, doi: 10.1109/PERCOM.2016.7456503.

[16] P. Mitra and C. Poellabauer, "Emergency response in smartphone-based Mobile Ad-Hoc Networks," 2012 IEEE International Conference on Communications (ICC), 2012, pp. 6091-6095, doi: 10.1109/ICC.2012.6364839.

[17] J. Vrindavanam, R. Srinath, A. N. D and S. Y. R, "Incident Response System Prototype Using LoRa," 2020 IEEE International Conference for Innovation in Technology (INOCON), 2020, pp. 1-5, doi: 10.1109/INOCON50539.2020.9298257.

[18] K. C. V. G. Macaraeg, C. A. G. Hilario and C. D. C. Ambatali, "LoRa-based Mesh Network for Off-grid Emergency Communications," 2020 IEEE Global Humanitarian Technology Conference (GHTC), 2020, pp. 1-4, doi: 10.1109/GHTC46280.2020.9342944.

[19] A. Djoudi, R. Zitouni, N. Zangar and L. George, "Do IoT LoRa Networks Support Emergency Evacuation Systems ?," 2019 International Conference on Information and Communication Technologies for Disaster Management (ICT-DM), 2019, pp. 1-2, doi: 10.1109/ICT-DM47966.2019.9032911.

[20] D. M. Molla, H. Badis, A. A. Desta, L. George and M. Berbineau, "SDR-Based Reliable and Resilient Wireless Network for Disaster Rescue Operations," 2019 International Conference on Information and Communication Technologies for Disaster Management (ICT-DM), 2019, pp. 1-7, doi: 10.1109/ICT-DM47966.2019.9032987.

[21] Dilmaghani, Raheleh B., and Ramesh R. Rao. "Hybrid Wireless Mesh Network with Application to Emergency Scenarios." J. Softw. 3.2 (2008): 52-60.

[22] Yarali, A., Ahsant, B., Rahman, S. (2009, June). Wireless mesh networking: A key solution for emergency rural applications. In 2009 Second International Conference on Advances in Mesh Networks (pp. 143-149). IEEE.

[23] Kanchanasut, K., et al. "Building a long-distance multimedia wireless mesh network for collaborative disaster emergency responses." Internet Education and Research Laboratory, Asian Institute of Technology, Thailand (2007).

[24] Deepak, G. C., et al. "An overview of post-disaster emergency communication systems in the future networks." IEEE Wireless Communications 26.6 (2019): 132-139.

[25] Cleary, John, and Ian Witten. "Data compression using adaptive coding and partial string matching." IEEE transactions on Communications 32.4 (1984): 396-402.