Providing Application Access to Voice Streams: Enhancing PTT Services for Emergency Response

Jiachen Chen
WINLAB, Rutgers University
Email: jiachen@winlab.rutgers.edu

K. K. Ramakrishnan
University of California, Riverside
Email: kk@cs.ucr.edu

***Abstract*—Mission-Critical Push-To-Talk (MCPTT) is an important tool for first responders in the aftermath of a disaster. Most existing MCPTT solutions try to reduce the call-setup delay or mouth-to-ear latency yet fail to optimize the user experience of the whole system. End-system applications primarily manipulate the voice streams through access to the control channel but have limited access to the voice streams. By allowing applications to have access to the voice streams and manage them, we can enable a multitude of features that are critical to first responders. In this paper, we propose Next-Gen MCPTT, a system that aims to improve the efficiency of delivering mission-critical communications. We support concurrent conversations and smooth switching among them, store & replay on demand, a "push-*and*-talk" capability facilitated by serialization of voice streams at the receiver, and device-to-device (D2D) communication. While MCPTT platforms require the speaker to wait for the floor to become clear before sending a message, Next-Gen MCPTT aims to automatically deliver messages when the channel is available and automatically stores them so the user can switch to other conversations as needed. By creating a system that stores, pauses, and replays messages both online and offline, Next-Gen MCPTT saves precious time for first responders by eliminating the need to constantly wait to acquire the "floor" to speak, repeatedly ask for information that the receiver did not hear clearly. It allows first responders to pay attention to urgent tasks in between and come back and catch up with the conversation later. The cost is a slight increase in the mouth-to-ear latency from allowing applications to get access to the voice streams.**

***Index Terms*—Mission-Critical Push-To-Talk, Disaster Management, D2D Communication**

I. Introduction

First responders critically depend on the ability to have interactive communication with other team members while responding to an emergency, for reacting to urgent needs, coordination and better overall situational understanding. Traditional mission-critical push-to-talk (MCPTT) technologies have been a key communication tool in emergency response. Historically, MCPTT relies on Land Mobile Radio (LMR) with both analog [1] and digital standards [2]. The 3GPP's SA Working Group 6 (SA6) Release 13 [3] also added standardization for MCPTT over LTE with priority and preemption. As a major implementation of mission-critical services in the US, the First Responder Network Authority (FirstNet) [4] hosts multiple MCPTT vendors on its network.

First responders need flexible group-based real-time interactive communication, to be able to respond to immediate needs for coordinated action. However, traditional MCPTT has all of the communicating parties (in essence broadcast) to be in a synchronous call, but using a messaging paradigm. One person speaks, and then another speaks, taking turns, waiting for the previous speaker/message to complete. While we seek to retain this "messaging" paradigm, there is a clear need for additional flexibility and enhancements to facilitate the needs of first responders: grouping, removing the restriction to have to wait to acquire the "floor" *etc.* Alternative approaches, such as voice "download-and-play" in instant messaging platforms have also evolved to provide interactive conversations (*e.g.*, WhatsApp, Skype, *etc.*). Yet neither the MCPTT or instant messaging approach provides the seamless combination of these capabilities (*i.e.*, interactive communication and a messaging paradigm for voice communications) to support the many capabilities that first responders dealing with emergencies need.

Existing systems and research has focused on the "mouth-to-ear" latency of systems to deliver voice streams [5]–[9]. For this, dealing with the voice in the OS and leaving the application only the control of calls and "floor" may be the right thing to do, to minimize delay. Yet it is important to provide applications the capability to manipulate the voice to provide features that are critical for first responders. Our conversations with first reponders indicate that the time wasted on the MCPTT communication is caused both with inconvenient functionality available for users, the interfaces, and the protocol-level interactions (*e.g.*, allow concurrent flows, similar to VoIP *vs.* traditional connection-oriented calls, and ability to have D2D communication).

Our work addresses these needs. The service-layer only only packetizes the voice streams, it also gives the applications full control of the voice data so that they can manipulate the voice messages and present them in different ways to users. We still seek to provide real-time interactive communications, but have the application on the receiver serialize the voice streams, enabling speakers to just speak (*push-and-talk*) instead of speakers having to serialize their speaking by waiting to acquire the floor (push-to-talk). There are a number of additional features we believe are important for first responders, including store and play the voice stream, allow users to switch back and forth when they have to pay attention to other tasks, and pause the conversation. Emergencies have a lot of different activities ongoing at the same time, and several individuals (especially incident commander, dispatcher and others) may need to participate in multiple concurrent conversations, seamlessly switching from one to the other, and still having access to all the content of each of the conversations as needed. When a user is participating in a conversation, first responders crucially depend on being able to override for emergencies. Disaster situations often have substantial ambient noise, where a user may not be able to hear clearly or their voice may be drowned by ambient noise.

978-1-6654-7036-0/22/$31.00 ©2022 IEEE
DOI 10.1109/CPS-ER56134.2022.00011

To facilitate communication, we see it desirable to provide for textual communication. However, users may also not be able to see or do not have a free hand to text. Therefore both text-to-speech and speech-to-text conversions are equally important. A final component is the real likelihood of communications infrastructure being damaged. Nonetheless, communication needs among first responders continues to be important. For this, we need to be able to support device-to-device (D2D) communication, and the MCPTT services need to be supported over D2D links.

This paper focuses on unifying existing MCPTT systems, (*e.g.*, MCOP) with many of the features needed for emergency communications that can be supported by treating the voice stream as yet another packet data stream accessible to the application layer. With this fundamental change, we are able to seamlessly support a variety of the features we described above in our Next-Gen MCPTT system. After outlining the needs of first responders, this paper describes the architecture and features of Next-Gen MCPTT.

II. Related Work

MCPTT has long been used in Land Mobile Radios (LMR) [1], [2]. In cellular networks, it is standardized by 3GPP in 3G, LTE, and 5G [3]. Many existing research tries to analyze [5], [6] and improve KPIs of MCPTT (*e.g.*, access latency, mouth-to-ear latency) via better routing strategy [7], signal compression [8], or closer edge-service location [5]. While these improvements reduce mouth-to-ear latency in the scale of milliseconds, we find that the first responders might end up wasting seconds to minutes dealing with the inconvenient functionalities (*e.g.*, waiting for the "floor"). Most of the work in the literature also assume a connected environment, either connected directly to the Internet, or the mobile devices are connected among each other [9]. This could difficult to consistently achieve in disasters and emergency situations due to the disruption to the infrastructure and the mobility of the first responders. It is desired that the delay-tolerant features be enabled alongside the real-time interactive MCPTT features and can smoothly switch between these two based on the connectivity of the user.

Our work focuses on improving the experience of the first responders when using the MCPTT applications. Thus, it is complementary to much of the existing research.

III. Use Cases

With packetized voice and the availability of capable handheld devices that are essentially networked computers, the capability of the MCPTT facilities can be dramatically expanded. We first describe at a very high level the need for features that our Next-Gen MCPTT can provide with typical off-the-shelf smartphones. We then discuss the features in detail and how we provide them in Next-Gen MCPTT in §IV and §V.

Case 1 [Acquiring Floor] – A simple and obvious case with traditional MCPTT systems is the difficulty in users having to serialize speech. This can be especially disruptive and inefficient on a connection with long delay, *e.g.*, a satellite link. First responders end up wasting a lot of time waiting for the floor, or just talk over each other. It is desirable to leverage the flexibility that comes from packetizing voice and storage to release MCPTT systems from this constraint of having toe acquire the "floor" to speak.

Case 2 [Repeat] – Voice communication, especially in emergency situations can be highly unreliable, because of ambient noise, poor connectivity, *etc.* Therefore, a receiver would often ask the sender to repeat what was said. This not only wastes communication channel resources but also the precious time of the first responders because messages are not stored and re-played.

Case 3 [Focus] – When a first responder has to focus on a task and then come back to the interaction with team members later, it is highly desirable to pause the ongoing conversation on the receiver end, and play them when he is ready to listen. This is especially true for group communication with multiple people interacting together.

Case 4 [Multi-tasking] – First responders, and especially incident managers, have to take care of multiple tasks while responding to emergencies. It would be desirable to store incoming messages and play on demand when the user wants to. A user may also want to switch between multiple communication sessions and switch back and forth between them without losing critical conversation that they can go back to and listen and pick up on the conversation as needed.

Case 5 [D2D] – Emergency response is inherently prone to operating in environments that may have disruptions in infrastructure-based communications. We see the need to seamlessly integrate the capability of device-to-device (D2D) communication and delayed forwarding and delivery to the recipient. This requires delay-tolerant network (DTN) support.

Case 6 [Mapping and Geo-location] – Another aspect that can be invaluable to integrate with voice and text communications is the ability for first responders to synchronize geo-located tasks (*e.g.*, inspect buildings after an earthquake) among the related teams. This could help everyone keep track of the conditions (better situation awareness) and avoid redundant work on the same task because of the lack of timely, relevant communication.

IV. Architectural Design

Our enhancements to the basic MCPTT include a number of innovations, many of which come from the basic conversion of communications to packetized voice and allow the application access the voice stream instead of just controlling calls. With this capability, we allow the applications serialize the conversation on the receiver end and play the voice on demand. At the same time, the devices can store and forward messages among themselves to provide D2D communication in the infrastructure-less environments. We also take advantage of speech-to-text and text-to-speech capabilities to make it more convenient, potentially accelerating communication among first responders through asynchronous communication in difficult emergency situations.

A. Providing Application Access to Chunked Voice

We built our implementation, starting from the base of an MCOP SDK [10]. We made modifications to the lower-

level components of MCOP to get access to the audio data. MCOP only allow application control of the audio playback (*e.g.*, initiate/hangup a call, take/release the "floor"). That is reasonable for handling audio in the lower layer (using native C/C++) more efficiently, especially when thinking about reducing mouth-to-ear latency. However, we expose the real audio data to the application so that it can control when to play or send audio as needed. While this is slightly more inefficient in terms of mouth-to-ear latency, we believe it eventually saves a lot of time for the user, especially first responders. This capability occurs after packetizing voice. A bypass (using the inter-module communication in Android) is created to communicate between the App-level and low-level recorders and players (so that the low-level functionality becomes a "shadow" component that only forwards data). We believe that our new platform capability that allows Apps to get access to the audio data can enable a variety of capabilities to meet different application requirements.

On the sender side, we chunk up the audio, packetize and send them. The receiver can play the short audio clips of each chunk, to give a real-time interactive communications experience, while fully taking advantage of packetized communications. We use Android's support for packetized voice, and chose the Pulse Code Modulation (PCM) 16-bit option with 6000 Hz sampling rate, which results in a bit rate of 96 Kbps. We chunk up the audio voice into 40 ms pieces thus resulting in 480 bytes per chunk. This offers a slightly higher quality than the standard PCM bit rate at 64 Kbps, but does offer clarity that is helpful for emergency communications.

B. *Store, Play on Demand, Forward, and Manipulate*

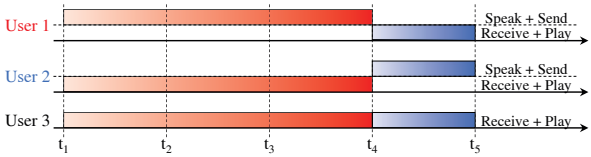
Once an application gets the audio data, it can store and schedule the playout of the voice messages based on the user's need. For example, to repeat a message when the user did not hear it clearly, pause a conversation when the user needs to focus on a task and come back to the interaction later, and let the emergency messages interrupt an ongoing (normal) conversation, *etc.* It can perform forwarding via both D2D and infrastructure-based communication whenever a channel is available, thus providing better communication quality in the field. It can further manipulate the voice, *e.g.*, performing speech-to-text to further assist first responders in noisy areas where listening to MCPTT could be challenging. By encoding and decoding text and binary data (*e.g.*, pictures) as part of the "voice stream", we can enable features like SMS, picture sharing, task progress synchronization, *etc.*

V. Features Enabled by Next-Gen MCPTT

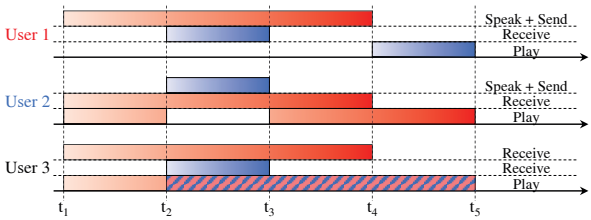
Build on top of the basic functionalities mentioned in §IV, we enable several features that can help first responders save time when dealing with disaster management.

A. *Push-**To**-Talk* vs. *Push-**And**-Talk*

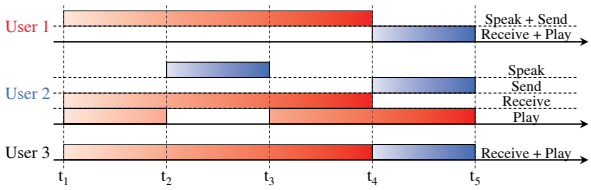
In Next-Gen MCPTT, since the application gets the access to both the incoming and outgoing voice streams, it can allow the first responders speak whenever they want to (push *and* talk, instead of push *to* talk). We consider a scenario as shown in Fig. 1a. User 1's stream is red, while User 2's stream is blue.



(a) Traditional: User 2 has to wait till t_3 to speak



(b) Next-Gen MCPTT: User 2 can speak and send at t_2; User 3 can choose which clip to listen to between t_2 and t_5



(c) Next-Gen MCPTT: User 2 can speak at t_2; Application can figure out when the channel is available then send

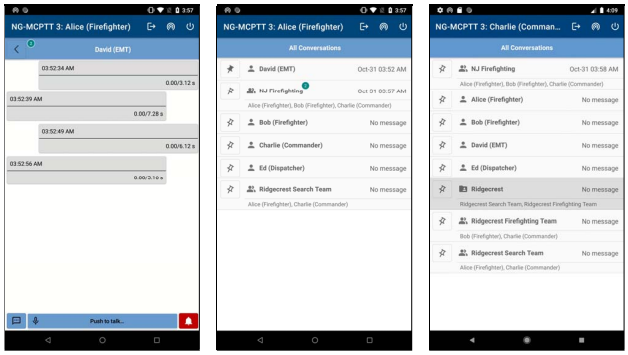
Fig. 1: MCPTT with concurrent speakers: User 2 wants to speak at t_2.

We show in two rows for each user the phases of speaking and sending *vs.* receiving and playing. Time steps are along the x-axis. User 3 only listens to the other two. Say user 1 is sending a long report from t_1 to t_4, and at t_2, User 2 wants to interrupt and speak. In traditional MCPTT, since the "floor" is occupied by User 1, User 2 has to wait till User 1 releases the token (t_4). In an extreme case, if User 1 does not release the PTT button on time by accident, the whole channel is blocked, and therefore no one can send anything, nor can anyone notify User 1 to release the token. This could result in important messages being delayed. At the same time, frequently waiting for the "floor" also wastes a lot of precious time of the first responder, which can be used instead to save lives.

In Fig. 1b, User 2 (blue) can just press the button to speak at t_2 without the need to wait till t_4 (after User 1 (red) finishes). The application can hold the playback of User 1's voice until User 2 finishes (t_3). At the same time, the application can send out the voice chunks so that all the receivers in the group can receive the message. From t_2 onward, User 3 (or any other receiver in the group) can choose to listen to whichever message they need, thus not missing urgent and/or important messages.

In cases with constrained channel availability, the application may also hold the outgoing messages and deliver them whenever the channel becomes available. In Fig. 1c, if the channel can only support 1 voice stream at a time, User 2 can still speak whenever he wants to (at t_2). The application stores the outgoing message, monitors the channel, and sends the voice only after User 1 finishes speaking (at t_4). We still save the time of User 2 since he does not have to wait to speak.

This also applies to the scenarios where User 2 is offline, or has intermittent connectivity (*e.g.*, satellite links). The



(a) Stay in a session (b) Switch among sessions (c) Consolidated view

Fig. 2: Providing conversational views for first responders.

application can store the outgoing voice and the user does not have to worry about the channel availability. We notice that in these scenarios, the messages could be delayed. To provide the first responders better context awareness, we carry the send time at the beginning of each voice message and show them on the user interface (see Fig. 2a). Our application also chooses to sort the voice messages based on their sending time to help first responders to better understand the time sequence.

B. Providing a Conversational View

In Next-Gen MCPTT, the application has the full access to all the incoming messages and can control when each one should be played. It can present the user a conversational view to provide first responders a view of the interactions within a context. The user can select a (group or private) conversation to focus on and the application only shows and plays the messages within that conversation (see Fig. 2a). On receiving messages from other conversations, the application stores the messages without playing them. Instead, it shows a marker on top left to indicate the # of messages pending (with a short notification sound). Thus the user does not get distracted by the messages that are not urgent. She can go back to those conversations after the current work is finished and re-listen to the missed messages.

The application also allows smooth transition between sessions. Fig. 2b provides a conversation overview for a user. Each item in the list represents a group conversation. The number on the right of the conversation name shows the # of missed messages in that conversation. The application can order these conversations based on different criteria like time last message is sent, private *vs.* group conversations, user's favorite (pinned group), *etc.* Since we store messages, when a receiver switches to a session, the application can play the messages missed by the user belonging to that session. Of course, if a user cannot hear a message clearly, she can just click the message to listen to it again. This saves time for the sender from constantly repeating the message. We considered providing a voice-activated navigation for the conversational view. However, we were concerned about it being error-prone, especially in a noisy environment. Therefore, we need to explore changes to hardware or other user interfaces to reduce the complexity of this navigation for first responders.

While the conversational view helps the user to focus on a context at a time, we do realize the need to prioritize some emergency messages (*e.g.*, a commander ordering everyone in the team to retreat). Since the application has full control of the video chunks, it is easy to allow emergency messages to override session boundaries to be played immediately (even preempt an ongoing message). The application can also allow the user to prioritize some conversations/senders (*e.g.*, messages from a direct command). The messages from these conversations or senders can be treated like emergency messages (with preemption), or just have a higher priority to be played (without preemption of the ongoing message). The application has the control to schedule the messages based on the scenario and the requirement of the user.

C. Multi-tasking and Command Hierarchy

For most first responders, a single conversation view like Fig. 2a may be adequate. However, for an incident commander managing multiple teams, we provide the ability to handling multiple groups (conversations) of first responders working on different tasks. Switching back and forth between them could be time consuming and distracting. It would also be undesirable to put the users into the same group, since that would result in excessive amount of information with the potential of confusing and distracting some of the first responders. In Next-Gen MCPTT, we allow first responders to create combined conversations with an integrated view, allowing the ability to communicate information relevant to multiple groups. In Fig. 2c, the highlighted conversation is a consolidated session at a commander that combines both the "Ridgecrest search team" and the "Ridgecrest firefighting team" in a forest fire mission in Ridgecrest. The commander can receive messages sent to both groups, without the need to switch between them. At the same time, the commander can also choose to send a message to one group, or even both groups. The application can replicate the message and send it onto multiple "channels", or take advantage of the network to perform multicast to all the first responders in these groups.

In many cases, first responder teams form a predefined structure to deal with a particular type of incident [11]. The structure is usually a command hierarchy where the nodes in the hierarchy represent groups at different granularity. For example, a search and rescue mission can be formed by a search team, a rescue team, a planning team, a law-enforcement team, *etc.* The search team can be further divided into 2 canine teams, 2 technical search teams (*e.g.*, drones), a coordinator team, *etc.* Next-Gen MCPTT can take advantage of the predefined structures (*a.k.a*, preplans) and build consolidated conversations automatically. In the example above, we can create a consolidated session that includes the canine, technical, and coordinator, teams (all of which are individual MCPTT groups). The commander of the search team can see all the messages going back and forth in these teams, without the need to switch out of the context. Similarly, we can also create a conversation for the commander for the whole search and rescue mission by combining the search team, rescue team, planning team, *etc.* The first responders can send messages to

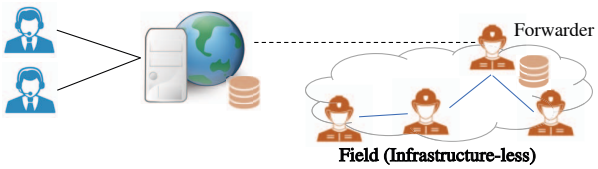


Fig. 3: Enabling seamless D2D communication.

any consolidated group at any granularity without the need to repeat the same message to multiple groups, thus saving time for the commander.

Please note that since the preplans are defined before the disaster, our application can digitize these preplans and create templates that instructs how the consolidated sessions should be created. When a disaster strikes, the incident commander can easily pull up a template and the application can create the templates automatically without any intervention or supervising from the commander (or coordinator). First responders can join the corresponding teams when they are dispatched to a particular role, and this can be done automatically by the application [12], [13] and the time spent on coordinating channels can be used to save lives.

D. Unifying Connected and D2D Communications

First responders often have to work in areas with limited or intermittent connectivity, or even without communication infrastructure support (*e.g.*, for urban search and rescue in the aftermath of an earthquake). In these cases, D2D communications (*e.g.*, Bluetooth, WiFi-Direct) can be used to satisfy the need for communication among first responders. While many existing DTN solutions support a store-and-play mechanism to deliver messages under such conditions, they lack a smooth transition between the connected and infrastructure-less environments – their solution lacks real-time interactive communication, even when the devices are connected to the Internet.

In Next-Gen MCPTT, since the applications have access to the audio chunks received, they can also use it for data forwarding between devices to either extend the coverage range of infrastructure (e.g., cellular) communication service (similar to call forwarding), carry the messages and deliver them to the users in the field (similar to DTN), or even provide local MCPTT delivery in the infrastructure-less environments via D2D communication. As shown in Fig. 3, when users have an Internet connection (the blue first responders in the figure), they can send MCPTT messages to groups via a (logically-centralized) server. The chunked voice provides real-time interactive communication, just like existing MCPTT applications (with our enhancements).

When a group of first responders are dealing with an incident in a communications infrastructure-less environment ("Field" in Fig. 3), and one of them happens to carry a device with Internet capability (*e.g.*, satellite phone, like the Forwarder in the figure), that forwarder can disseminate messages received to the other first responders via D2D communication *in real time*. This provides almost equivalent functionality for all the first responders, that once one device is connected to the

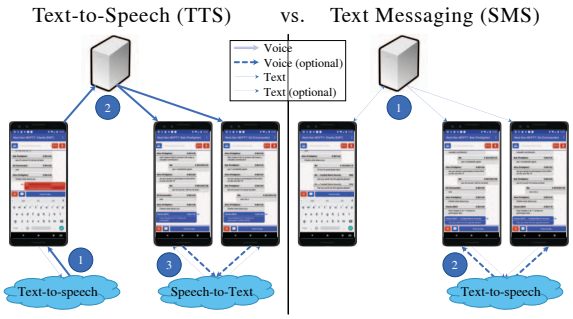


Fig. 4: Text-to-Speech (TTS) *vs.* Text Message (SMS).

Internet, all the other devices are also connected automatically. The forwarder can even help deliver the messages that are not destined to him (or the group he is in). The application can simply forward the messages without playing or showing it to the forwarding device's user.

When the Internet connection of the forwarder becomes intermittent, the server and the forwarder can store the messages and send them whenever the communication channel becomes available. This would be very helpful in situations where first responders have to deal with underground incidents. Similar to the feature in push-and-talk (§V-A), we mark the send time of each message to maintain a time sequence even when some messages are delayed.

When the forwarder is fully disconnected from the Internet, he can also elect himself as a temporary server to enable communication among several users (say within a shelter). In our system, we allow a device to forward messages for multiple users, but only connect to 1 upstream device or server. Therefore, the users will eventually form a tree rooted at the server. This structure has proved to be efficient, especially when communications have to go through the server, compared to a mesh network where routing can be more complex.

E. Enabling Text-to-Speech and Speech-to-Text

Since our application has full access to the voice streams, they take advantage of the speech-to-text service to translate every received message to text. This can help first responders in noisy places (where they cannot hear the messages clearly), or in places where they cannot play the audio. This can also help first responders who might receive a large number of messages, so that they can glance through the texts and play only the messages whose transcription are confusing. For each message received, our application performs speech-to-text in the cloud. To make it easier for the users, we also put the sender name and the length of the message at the top.

We also incorporate the text-to-speech capability. First responders can type a message in text. The application would generate a synthesized voice and send the voice to the group. This would be helpful in the environment where the first responders are not able to speak (see the left side of Fig. 4). SMS is also natively supported, and combined with voice synthesis (see the right side of Fig. 4). It allows first responders to be "eyes-free" even when they receive text messages. In comparison SMS could save the bandwidth in the common

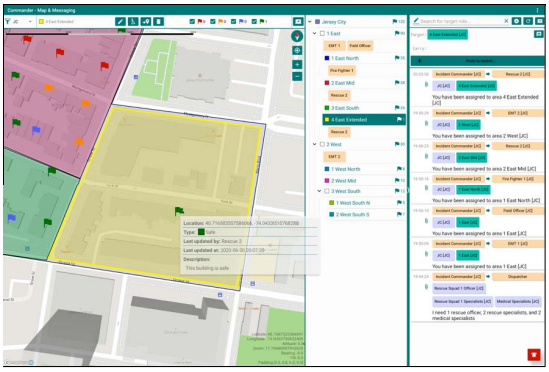


Fig. 5: Integrating mapping into Next-Gen MCPTT.

channel (to/from the server) since text message is much smaller compared to the synthesized voice. Our application can use cloud-based text-to-speech and speech-to-text services whenever the devices are connected to the Internet. To facilitate the first responders without Internet connectivity, we can also deploy services directly in the field. These services can cache the latest results to avoid redundant translation between text and speech for the messages broadcasted in the same group (step 3 of TTS and step 2 of SMS in Fig. 4).

F. Integrating Mapping Capability

With the support for command hierarchies and the applications being able to access all the communication data (including voice, text and any binary blobs), we go further and integrate a mapping capability in Next-Gen MCPTT. The incident commander can divide a disaster region into hierarchically structured areas and dispatch first responder teams to deal with tasks in corresponding areas. Fig. 5 shows the view on an incident commander. Beyond the MCPTT panel shown on the right, we have added a map task view. In the middle, we see that the incident area ("Jersey City") divided into "East" and "West" sections. Each section is further divided into areas and first responders are dispatched to an area (we can support arbitrary number of levels to deal with incidents at any scale, but we use 3 levels here for the ease of explanation). The left panel shows the map and the corresponding areas (the selected region is highlighted with a yellow shade). First responders can add, update and remove tasks on the map (*e.g.*, inspection of the structural stability of each building), and each task is represented by flag and we use different colors to represent different task types. When the incident commander hovers onto a task, he can see the current status and the progress towards completion.

The communication for synchronizing tasks on the map are based on our SMS feature (with D2D communication support). Whenever a task is created, updated or deleted, an SMS (using JSON format) is sent to the group of the corresponding area/section/region. The application reads the SMS and updates the local map accordingly. We also support transmission of binary data (*e.g.*, pictures) through the channel enabling first responders to attach pictures to the tasks. These are all enabled by the application having access to the data transmitted in the channel to present them in different ways.

VI. Conclusion

In this paper, we presented Next-Gen MCPTT, a novel system that improves the mission-critical push-to-talk service by enabling features like push-and-talk, providing a conversational view, multitasking, device-to-device communication, text-to-speech and speech-to-text support, and map functionalities that are very well suited for first responders to use during emergency management. The fundamental changes we made was to not only packetize voice clips, but also give applications full control of the voice streams. Thus, applications can easily manipulate the voice streams in the way they want, and present the messages to first responders appropriately.

We demonstrated our application and had several first responders try our application in their training exercises at Disaster City at Texas A&M University. Their feedback was that the new features saved them a lot of effort in communicating, and helped them focus on their work during the exercise.

There are 3 directions for our future work. 1) We plan to integrate our application into ATAK (Android Team Awareness Kit) so that more first responders can take advantage of the application in their real work. 2) We plan to explore other wireless technologies like LoRa to provide better D2D coverage in the infrastructure-less environments. 3) We also need to quantify increase in the mouth-to-ear latency because of our design. We also foresee that more applications can take advantage of the exposed audio data and provide even better services to first responders.

Acknowledgment

This work was supported by the US NIST Tech-to-Protect Challenge and US NIST PSIAP grant 70NANB17H188.

References

[1] Telecommunications Industry Association (TIA), "Land Mobile FM or PM Communications Equipment Measurement and Performance Standards," 2016. Standard 102.E.

[2] Technical Committee Terrestrial Ttrunked Radio and Critical Communication Evolution, "TErrestrial Trunked RAdio (TETRA)."

[3] 3GPP TSG SA WG6 (SA6), "Release 13 – Mission Critical Push To Talk over LTE," 2015.

[4] "FirstNet." https://www.firstnet.com/. accessed on 02/20/2022.

[5] A. Sanchoyerto, R. Solozabal, *et al.*, "Analysis of the Impact of the Evolution Toward 5G Architectures on Mission Critical Push-to-Talk Services," *IEEE Access*, pp. 115052–115061, 2019.

[6] C. Brady and S. Roy, "Analysis of Mission Critical Push-to-Talk (MCPTT) Services Over Public Safety Networks," *IEEE Wireless Communications Letters*, pp. 1462–1466, 2020.

[7] J. Tang, G. Chen, and J. P. Coon, "Route Selection Based on Connectivity-Delay-Trust in Public Safety Networks," *IEEE Systems Journal*, pp. 1558–1567, 2019.

[8] Z. Niu, H. Li, and C. Xu, "A Signaling Compression Method for MCPTT Trunking Communication System Based on Carrier Aggregated Broadband System," in *ICEIEC*, 2017.

[9] Y. Sun, W. Garey, *et al.*, "Access Time Analysis of MCPTT Off-network Mode over LTE," *Wireless Communications and Mobile Computing*, 2019.

[10] "Mission Critical Open Platform." https://www.mcopenplatform.org/. accessed on 02/20/2022.

[11] "National Incident Management System." https://www.fema.gov/emergency-managers/nims, 2022. accessed on 02/20/2022.

[12] M. Jahanian, J. Chen, and K. K. Ramakrishnan, "Graph-based Namespaces and Load Sharing for Efficient Information Dissemination in Disasters," in *ICNP*, 2019.

[13] K. K. Ramakrishnan, M. Yuksel, *et al.*, "Resilient Communication for First Responders in Disaster Management," in *ISCRAM*, 2021.