INTERNATIONAL TELECOMMUNICATION UNION

TELECOMMUNICATION STANDARDIZATION SECTOR

STUDY PERIOD 2022-2024

FOCUS GROUP ON AI NATIVE NETWORKS

AINN-I-xx

Original: English

Question(s): N/A — Virtual, TBD 2024

INPUT DOCUMENT

Source: *SRM University - Andhra Pradesh*

Title: *HeaxCore Team* – Report on ITU WTSA Hackathon 2024 – *AuRa: AuratRaksha*

Contact:	Mr. Gopi Chand Janjanam	E-mail: gopichand_janjanam@srmap.edu.in
Contact:	Mr. Rohith Kumar Ankam	E-mail: rohithkumar_ankam@srmap.edu.in
Contact:	Mr. Eswar Panchakarla	E-mail: eswar_panchakarla@srmap.edu.in

Abstract: This document contains the submission of a report for *HeaxCore* Team towards ITU WTSA Hackathon 2024 for use case "AuRa: Alert, Unify, Respond, Assist towards AuratRaksha".

Use case introduction: "AuRa: Alert, Unify, Respond, Assist - AuratRaksha"

Ensuring Safety and Immediate Assistance for Women in UnSafe Situation or Constrained Situations

According to the Report, India reported a total of 58,24,946 cognizable crimes in the year 2022. Millions of women worldwide face violence, harassment, or threats in supposedly safe spaces, such as homes, offices, hospitals, public transportation, or while walking alone at night. In these situations, reaching out for help via phone calls or messages can exacerbate the danger. The fear of vengeance or escalation prevents them from seeking immediate assistance. Imagine a lady Sita, or someone close to you is in a situation where the lady is not safe. It could be at home, office, or hospital. Sita might feel it was a safe place. But unfortunately what if she is not safe? In the privacy of her own space, someone she knows or someone she doesn't know tries to harm her?

Harmful scenarios or other dangers can persist anywhere which includes in the streets, public places, walking alone at night, traveling in train or bus, trapped in an office or home and other places having a fear and uncertainty. Sita wants to cry out for help, but the threat is too close. Reaching for her phone, making a call, or even sending a message could put her in greater danger.

How can Sita escape from a particular harmful situation and back to a normal position? Can you help Sita?

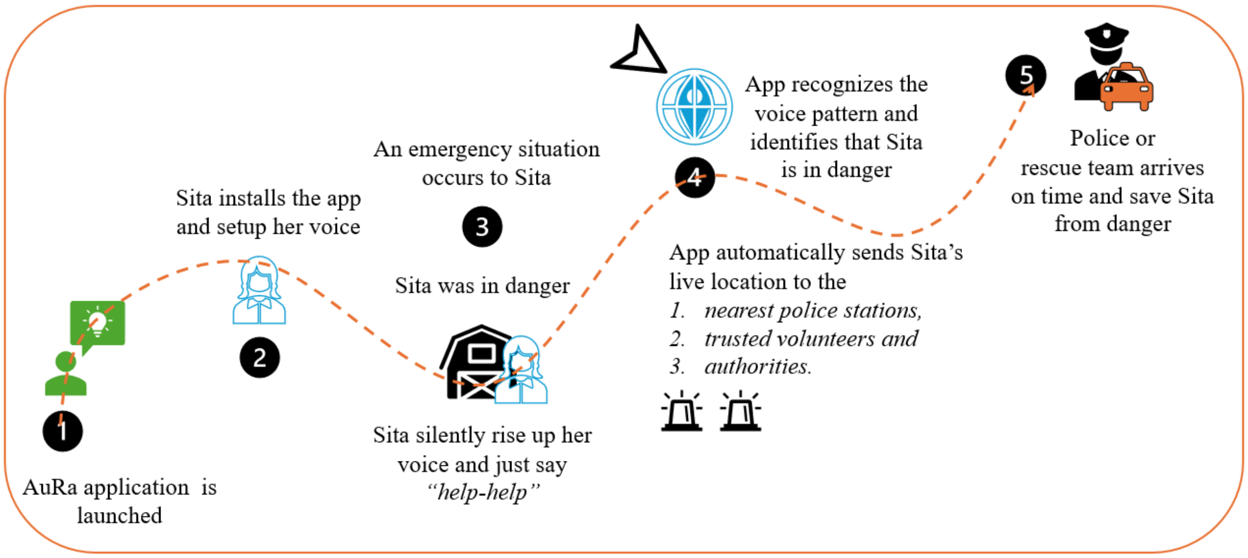
Scenario Story:

A student named Gopi from Andhra Pradesh, while he was browsing on the internet, a news report caught his eye: "India Reports 58,24,946 Cognizable Crimes". He was sad to hear that millions of women worldwide face violence, harassment, or threats in supposedly safe spaces. One line haunted him: "Many victims remain silent, fearing escalation or vengeance." Gopi thought of his close ones. Could they be next?

Gopi thinks, "What if I could create an app that lets women like Sita call for help silently?". His vision took shape: *an AI-driven safety system*, harnessing 5G/6G networks, to protect women wherever they were. With a simple, silent voice command, sita can trigger an alert. Sita's location and live audio feed would be shared with nearest police officers, trusted volunteers and authorities. Real-time tracking and communication will ensure a swift help. Gopi's team partnered with local organizations, training volunteers to respond to alerts. They collaborated with law enforcement, streamlining the process for rapid assistance. The app launched, and the response overwhelmed Gopi. Women from across India downloaded the app, sharing stories of empowerment. Sita felt safer knowing she had a secret power in her pocket. One night, a message arrived: "Gopi, your app saved my life. I was trapped in my own home, but with the app AuRa, I whispered for help. The police arrived in minutes and I am safe. Thank you for hearing my silent cry."

Gopi's work had made a difference. The app AuRa became a symbol of hope, a reminder that technology could be a powerful tool against violence. His story inspired others to join the fight against violence. AuRa app expanded globally, safeguarding women in every corner of the world.

Consider the scene map below:


1
AuRa application is launched
2
Sita installs the app and setup her voice
3
An emergency situation occurs to Sita
Sita was in danger
Sita silently rise up her voice and just say *"help-help"*
4
App recognizes the voice pattern and identifies that Sita is in danger
App automatically sends Sita's live location to the
1. nearest police stations,
2. trusted volunteers and
3. authorities.
5
Police or rescue team arrives on time and save Sita from danger

<u>Outline of the Use-case (Each Phase indicates the Timeline of the Scenario)</u>

Phase 1: Sita is in an unsafe situation where some are attacking her. Sita has "*AuRa, an emergency rescue app*". Sita installs the app and sets up her voice commands successfully.

Phase 2: Unfortunately there is no one to help sita.

Phase 3: Sita " *raises her voice seeking help*". The app which is already having Sita's voice inputs. It automatically recognizes the voice pattern and identifies that sita is in danger. App automatically sends Sita's live location to the nearest police stations, trusted volunteers and authorities.

Phase 4: "*Police or rescue team arrives within minutes*" identifies the situation of Sita, and immediately the police men or rescue team saves Sita from danger.

Phase 5: Sita comes out safely with the help of police officers or the rescue team, who arrives first to secure Sita from an emergency situation.

Use case requirements

Requirement-1: It is critical to have 5G/6G technology to ensure ultra-low latency for voice, audio, and data transmission. This is critical for real-time communication of audio alerts and GPS data to emergency responders, ensuring swift action during emergencies.

Requirement-2: *High Precision AI Detection(AI/ML models)*—The application must integrate high precision AI algorithms capable of accurately detecting the victim's voice and assessing the situation(emotion/emergency detection)

Requirement-3: *Prioritized Resource Allocation Based on Location*—It is critical to prioritize resource allocation based on the user's location during an emergency. This involves deploying and managing RAN slices specifically optimized for emergencies to ensure that resources are available where they are most needed.

Requirement-4: *RAN Slice Management for Uninterrupted Communication*—It is critical to have and must support RAN slice management, specifically selecting URLLC (Ultra-Reliable Low Latency Communication) slices to facilitate uninterrupted communication during emergencies. This ensures that voice commands and data transmissions are processed with minimal delay and maximum reliability.

Requirement-5: *Fake Emergency Detection Mechanism*—To prevent misuse of the system, a mechanism must be implemented to detect potential fake emergencies. This could involve analyzing patterns of voice commands over time, assessing user behavior, and employing machine learning models to distinguish between genuine distress signals and false alarms.

Our use-case has the potential to address the below United Nations Sustainable development Goals (SDG's)

UN Goals

1. SDG-5: Gender Equality
2. SDG-3: Good Health and well being
3. SDG-16: Peace, Justice, and Strong Institutions

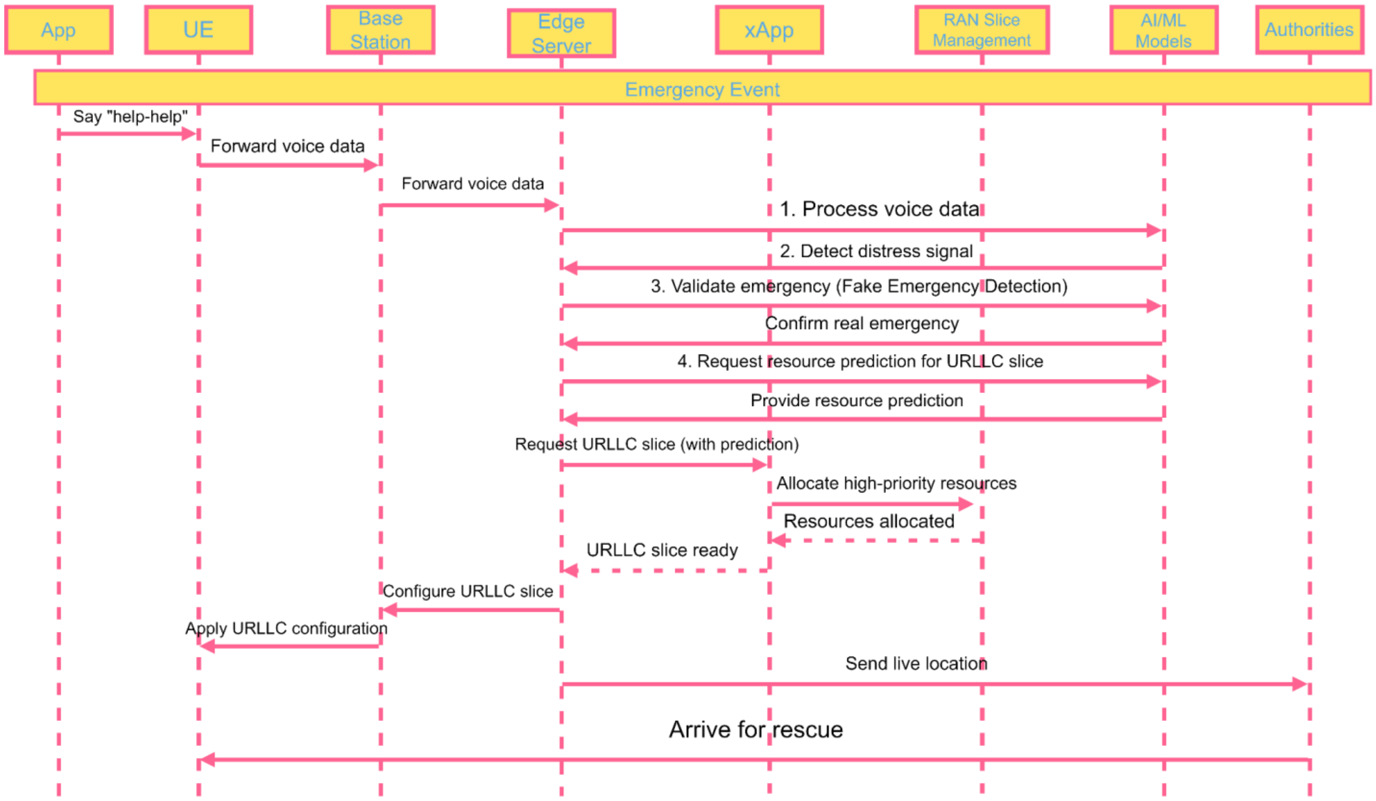
Justify UN Goals Selection:

- SDG - 5: Eliminate all forms of violence against all women and girls in the public and private sphere and other types of exploitation.
- SDG-3: Our use case helps to reduce the stress and anxiety for women through enhanced safety.
- Goal 16: Peace, Justice, and Strong Institutions – Promoting a safe environment is directly linked to reducing violence and ensuring justice. By building systems that can respond to emergencies.

SPS1: pipeline design

- AI /ML Concept used is emotion/emergency detection, real-time data processing, anomaly detection and live racking detection.

- In Relation with ITU Y.3172 – for submission.
  xml link for the above sequence is here: Aura: [sequence xml file]



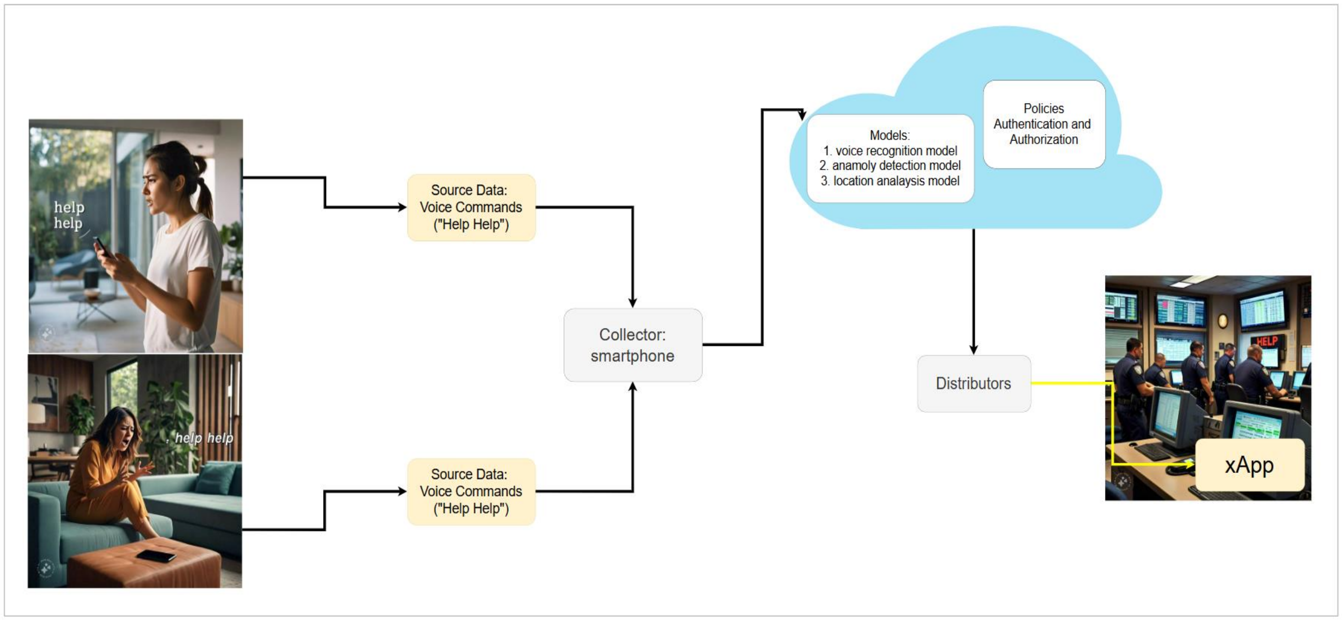
Yaml Code

```
tosca_definitions_version: tosca_simple_yaml_1_3

description: Women's Safety Emergency App Service Template with AI-based threat detection, and enhanced RAN slice management.

node_types:
  App:
    derived_from: tosca.nodes.SoftwareComponent
    properties:
      app_version:
        type: string
      safety_features:
        type: list
        entry_schema:
          type: string

  UserDevice:
    derived_from: tosca.nodes.MobileDevice
    properties:
      device_type:
        type: string
      os_version:
        type: string

  BaseStation:
    derived_from: tosca.nodes.Network.BaseStation
    properties:
      coverage_radius:
        type: float
        unit: km

  EdgeServer:
    derived_from: tosca.nodes.Compute
    properties:
      processing_power:
        type: float
        unit: GFLOPS

  SafetyXApp:
    derived_from: tosca.nodes.SoftwareComponent
    properties:
      api_version:
        type: string
      supported_emergency_types:
        type: list
        entry_schema:
          type: string
      enable_silent_mode:
        type: boolean
      predictive_threat_detection:
        type: boolean

  RANSliceManagement:
    derived_from: tosca.nodes.SoftwareComponent
    properties:
      max_priority_slices:
        type: integer
        dynamic_scaling:
          type: boolean

    SafetyAIModels:
      derived_from: tosca.nodes.SoftwareComponent
      properties:
        model_version:
          type: string
        supported_languages:
          type: list
          entry_schema:
            type: string
        geofencing_enabled:
          type: boolean

    EmergencyAuthorities:
      derived_from: tosca.nodes.SoftwareComponent
      properties:
        jurisdiction:
          type: string
        emergency_services:
          type: list
          entry_schema:
            type: string
        alert_confirmation:
          type: string

  topology_template:
    node_templates:
      app:
        type: App
        properties:

          safety_features:
            - voice_activated
            - location_tracking
            - fake_call



        requirements:
          - host: user_device

      user_device:
        type: UserDevice
        properties:
          device_type: "smartphone"
      os_version: "latest"
    requirements:
      - connection: base_station

  base_station:
    type: BaseStation
    properties:
      coverage_radius: 5.0
    requirements:
      - connection: edge_server

  edge_server:
    type: EdgeServer
    properties:
      processing_power: 1000.0
    requirements:
      - connection: safety_xapp

  safety_xapp:
    type: SafetyXApp
    properties:

      supported_emergency_types:
        - physical_threat
        - medical_emergency
        - harassment
        - predictive_threat
      enable_silent_mode: true
      predictive_threat_detection: true
    requirements:
      - host: edge_server
      - dependency: safety_ai_models

  ran_slice_management:
    type: RANSliceManagement
    properties:
      max_priority_slices: 10
      dynamic_scaling: true
    requirements:
      - host: edge_server

  safety_ai_models:
    type: SafetyAIModels
    properties:
      model_version: "3.3.0"
      supported_languages:
          - English
          -
        geofencing_enabled: true
      requirements:
        - host: edge_server

    emergency_authorities:
      type: EmergencyAuthorities
      properties:
        jurisdiction: "local"
        emergency_services:
          - police
          - ambulance

        alert_confirmation: true

  relationships:
    app_to_device:
      type: tosca.relationships.ConnectsTo
      source: app
      target: user_device
    device_to_base_station:
      type: tosca.relationships.ConnectsTo
      source: user_device
      target: base_station
    base_station_to_edge_server:
      type: tosca.relationships.ConnectsTo
      source: base_station
      target: edge_server
    edge_server_to_safety_xapp:
      type: tosca.relationships.ConnectsTo
      source: edge_server
      target: safety_xapp
    safety_xapp_to_ai_models:
      type: tosca.relationships.DependsOn
      source: safety_xapp
      target: safety_ai_models
    safety_xapp_to_ran_slice:
      type: tosca.relationships.ConnectsTo
      source: safety_xapp
      target: ran_slice_management
    safety_xapp_to_authorities:
      type: tosca.relationships.ConnectsTo
      source: safety_xapp
      target: emergency_authorities
```

```
interfaces:
  Standard:
    operations:
      create:
        description: Set up the women's safety emergency system
      delete:
        description: Tear down the system
      start:
        description: Start the emergency response service
      stop:
        description: Stop the emergency response service
      configure_priority_slice:
        description: Configure high-priority network slice for emergency response

      validate_emergency:
        description: Detect and validate emergency situations using AI models and geofencing
      allocate_emergency_resources:
        description: Allocate high-priority resources dynamically for emergency response
      send_location_to_authorities:
        description: Send real-time location data to relevant emergency authorities
      confirm_alert_receipt:
        description: Confirm that emergency authorities have received the alert and responded
      predictive_threat_alert:
        description: Detect potential threats using predictive AI algorithms
```

Requirements for this type of application?

- SRC of data: UE
- Collector: edge server
- Models: emotion, emergency detection
- Policies: fake emergency detection
- Distributors: edge server
- Model inference Application (SINK): xApp



PS2: xApp design

- The role of xApp?

  In this xApp design main functionalities required for the woman safety are:

  1. This xApp is designed for RAN slice management and slice selection.
  2. Verifying the emergency to prevent false alarms.
  3. Allocating a URLLC (Ultra-Reliable Low-Latency Communication) slice for emergency situations.
  4. Prioritizing network resources for the emergency UE

xApp code here:

```
import json
import time
from datetime import datetime

class WomanSafetyXapp:
    def __init__(self):
        self.emergency_detected = False
        self.current_slice = "default"

    def process_e2_input(self, e2_message):
        message = json.loads(e2_message)
        if message['type'] == 'voice_data':
            self.process_voice_data(message['data'])
        elif message['type'] == 'ue_measurement':
            self.process_ue_measurement(message['data'])

    def process_voice_data(self, voice_data):
        emotion = self.detect_emotion(voice_data)
        if self.detect_emergency(emotion, voice_data):
            self.handle_emergency()

    def detect_emotion(self, voice_data):
        # we will Simulate emotion detection
        # In a real implementation, this use a machine learning model
        print("Detecting emotion from voice data...")
        return "distress"  # Simulated result

    def detect_emergency(self, emotion, voice_data):
        # we will Simulate emergency detection
        # In a real implementation, this would use more sophisticated logic
        emergency_keywords = ["help", "emergency", "danger"]
        return emotion == "distress" and any(keyword in voice_data.lower() for keyword in emergency_keywords)

    def handle_emergency(self):
        self.emergency_detected = True
        self.allocate_urllc_slice()
        self.prioritize_resources()
        self.alert_emergency_services()

    def allocate_urllc_slice(self):
        print(f"Allocating URLLC slice at {datetime.now().strftime('%H:%M:%S')}")
        self.current_slice = "URLLC"
        return self.generate_e2_output('slice_allocation', {
            'slice_type': 'URLLC',
            'action': 'allocate'
        })

    def prioritize_resources(self):
        print(f"Prioritizing resources for emergency UE at {datetime.now().strftime('%H:%M:%S')}")
        return self.generate_e2_output('resource_prioritization', {
            'ue_id': 'emergency_ue',
            'priority': 'high'
        })
    def alert_emergency_services(self):
        print(f"Alerting emergency services with location data at {datetime.now().strftime('%H:%M:%S')}")
        # In a real implementation, this would interface with emergency services systems

    def process_ue_measurement(self, measurement_data):
        print(f"Processing UE measurement: {measurement_data}")

        if self.emergency_detected and measurement_data['rsrp'] < -100:
            self.prioritize_resources()  # Re-prioritize if signal strength is low

    def generate_e2_output(self, message_type, data):
        return json.dumps({
            'type': message_type,
            'data': data,
            'timestamp': datetime.now().isoformat()
        })

# Example usage
xapp = WomanSafetyXapp()

# Simulate E2 input (voice data)
e2_input_voice = json.dumps({
    'type': 'voice_data',
    'data': 'Help! I need emergency assistance!',
    'timestamp': datetime.now().isoformat()
})
xapp.process_e2_input(e2_input_voice)

# Simulate E2 input (UE measurement)
e2_input_measurement = json.dumps({
    'type': 'ue_measurement',
    'data': {
        'ue_id': 'emergency_ue',
        'rsrp': -95,
        'rsrq': -12
    },
    'timestamp': datetime.now().isoformat()
})
xapp.process_e2_input(e2_input_measurement)
```

Relation to Standards

1. ITU-T Y.3173: Integrating real-time machine learning into the network for tasks like emergency detection, resource allocation, and response optimization

Example:

a. Automation Level: Assessing if the network can autonomously allocate resources in response to emergencies.
b. Learning Efficiency: How quickly an AI system learns from emergency patterns and improves its response.

2.ITU-T Y.2091,refers to a structured set of capabilities designed to offer value-added functionality, particularly in this use case emergency response and women's safety. Let's break down how this Application in use case:

- Structured Capabilities:
  - The safety system integrates multiple components like voice command recognition, real-time location tracking, and emergency alert dispatch. Each of these components contributes to the structured functionality of the application.
- Value-Added Functionality:
  - The core value lies in the system's ability to provide timely and effective help during an emergency. By triggering alerts through voice commands or other data sources and seamlessly communicating with authorities and volunteers, the system offers crucial functionality that enhances personal safety.
- Supported by Services:
  - The application would leverage several network services, such as:
    - Location-based services (LBS) for tracking the user.
    - AI-driven voice recognition for detecting distress.
    - Emergency communication protocols for prioritizing critical data transmission over the network (via 5G/6G).

Code submission details

1. I created the GitHub account
2. I created the repository, made it public
3. I commit my file, added the document
4. The repository link is here: Aura: Use-case GitHub Submission Link