IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 24 March 2024, accepted 23 April 2024, date of publication 1 May 2024, date of current version 8 May 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3395370

RESEARCH ARTICLE

Empowering Collaborative Application Development: A Robust Framework for Ad-Hoc Distributed Systems

IMRAN ABBAS KHAWAJA[1], KAMRAN ABID[2], (Member, IEEE), UZMA FAROOQ[3], ZAKI MALIK[4], AND ADNAN ABID[1,5], (Senior Member, IEEE)

[1]Department of Computer Science, University of Management and Technology, Lahore 54770, Pakistan
[2]Institute of Electrical, Electronics Computer Engineering, University of the Punjab, Lahore 54590, Pakistan
[3]Department of Software Engineering, University of Management and Technology, Lahore 54770, Pakistan
[4]Department of Marketing and Business Analytics, Texas A&M University—Commerce, Commerce, TX 75428, USA
[5]Department of Data Science, University of the Punjab, Lahore 54590, Pakistan

Corresponding author: Adnan Abid (adnanabid7@gmail.com)

ABSTRACT The amalgamation of ubiquity, mobility, computing prowess, storage, and communication of the portable electronic devices including smartphones, wearable devices, tablets have significantly altered the fabric of human existence. The software applications for these mobile devices possess the capability to disseminate and receive information, thereby enable the development of applications that may execute in collaborative manner. However, in situations with ad-hoc connectivity unavailability of infrastructure and communication breakdown are common, which inhibit information sharing across the devices, and hence affect the applications running in the collaborative spaces. Further the applications developers face challenges while developing reliable collaborative applications that can seamlessly execute in ad-hoc communication settings with recurring issues with device connectivity. Therefore, it is necessary to provide a reliable framework to the application developers which provides such an abstraction that they may create robust applications without any concern about random communication disruptions. To address the aforementioned issues, this article proposes a framework that facilitates the development of reliable and efficient collaborative applications that communicate across devices while working in ad-hoc device connectivity settings. The proposed framework provides an abstraction of some major components including service exposition, service registration, storage, and synchronization while ensuring robustness to the challenges posed by the intermittent connectivity failures. The efficacy of the framework has been demonstrated through a detailed experimental evaluation using a custom developed collaborative application run in a variety of operational settings over different portable devices in ad-hoc settings. The results reveal that the framework not only successfully mitigates the challenges posed by the ad-hoc device connectivity, but also the abstractions provided by the framework not only enables the application developers create customized robust collaborative applications with significantly fewer lines of code, but also without worrying about the intermittent communication failures among the involved devices.

INDEX TERMS Ad-hoc distributed systems, ad-hoc collaborative spaces, collaborative applications, dependable systems, mission critical applications.

I. INTRODUCTION

A ubiquitous computing environment highlights the need for reliable collaborative spaces where program and data sharing across heterogeneous devices is possible [1]. Present-day portable electronic devices (smartphones, tablets, laptops, wearables, etc.) make this possible by offering more advanced computing, storage, and connectivity capabilities, helping create innovative life services and additional user values [2]. However, a smooth formation of collaborative

The associate editor coordinating the review of this manuscript and approving it for publication was Nafees Mansoor.

spaces ideally requires fixed infrastructure i.e. cellular/internet services for communicating across multiple devices. In the absence or suspension of these services, and the non-availability of infrastructure, the devices may temporarily disconnect from one another, causing issues in communication across the devices. These requirements lead to the need for a robust collaborative space that can function over an ad-hoc distributed system seamlessly.

Ad-hoc distributed systems do not have a predefined structure. The system must adapt to the changing needs and demands by configuring itself both locally and globally. The design and creation of collaborative applications over ad-hoc distributed systems thus pose numerous challenges, whereby such applications should exhibit availability, reliability, performance, fault tolerance, security, and energy efficiency [3], [4]. Availability refers to the fact that the system remains available to the users with minimal downtime. Reliability demands that the system is consistently able to work as per its specifications. Performance refers to the fact that in most cases, the system performs its functionality in a specified time. Similarly, fault tolerance refers to the ability of the system to recover from failures that may occur due to unexpected conditions. Security in the given context is the ability of the system to keep running services over malicious faults. Lastly, energy efficiency demands that the battery life of the electronic device being used may last for a long time.

Consider the case of *ambient assisted living* which involves the use of wireless sensors and medical sensors to ensure that a senior citizen living alone in a house is safe [5]. The non-availability of the Internet or cellular services here can pose a serious threat to the lives of senior citizens. Similarly, a robust alert and rescue system over ad-hoc connectivity can play a key role in delivering warning messages to the potentially affected area residents and enabling rescue workers to carry out rescue activities effectively [6], [7]. Solutions in this regard focus primarily on the formation of static ad-hoc networks using battery-powered wireless relays or flying vehicles [8], [9]. However, these require external intervention to form a static ad-hoc network and may delay instant rescue and communication activities, causing serious damage. Therefore, there is a need to form a dynamic ad-hoc network on the fly at any time, anywhere, using nearby mobile devices by utilizing the ad-hoc network protocols incorporated within those devices, such as Wi-Fi Direct, Bluetooth Multipeer Connectivity, etc., [10]. Moreover, the research work conducted in the past focused on particular interaction styles such as collaborative visualization, task transfer or distribution, and collaboration, but did not facilitate the application developer in developing customized and robust applications over ad-hoc distributed systems as per their requirements. Furthermore, many of them have not explicitly evaluated their work with regards to robustness. In order to address these issues a framework was proposed to facilitate device users to perform collaboration activities across the devices using an ad-hoc network in [11]. The framework however does not address the network connectivity failure issues and is prone to errors in the disconnection of involved devices. Hence, still there is a dire need to extend and refine the framework by considering the self-adaptation, data management, network management, and client-customized requirements to address the above-mentioned challenges.

This research extends the aforementioned framework [11] so as to make it more robust, reliable, and efficient. Moreover, the features of the proposed framework have been exposed to the developers in an abstract manner using a well-defined Application Programmer Interface (API) to facilitate the development of reliable collaborative applications. The aim is to provide high-level abstraction in components to facilitate the application developers in developing applications without worrying about the underlying complexities of ad-hoc distributed systems, which are taken care of by the proposed framework.

II. RELATED WORK

In 1996, Robertson began the cross-device interaction journey by designing the uni-directional system that enabled the user to operate the television through a personal digital assistant (PDA) [12]. Subsequently, a rising trend of research in this area was observed, which has continued to grow in popularity. Nowadays, the devices available in the commercial market also support cross-device interaction, such as wearable devices used to control the mobile phone or make and receive phone calls without picking up the phone [13]. In recent years, numerous research articles have been presented that address the different cross-device interaction styles [14], [15].

This section presents an overview of existing frameworks for cross-device interaction, which can be broadly classified into two categories: infrastructure-based and ad-hoc network-based. An ad-hoc network is a temporary form of a decentralized network that can be created instantly for a short period and allows nearby devices to share information without the need for fixed infrastructure i.e. internet, cellular, etc. using the short-range communication protocol [16]. In this research, we have particularly focused on ad-hoc networks and proposed a comprehensive framework that is designed to empower application developers to create collaborative applications easily that can work across ad-hoc distributed systems. Importantly, these applications can operate without relying on a fixed infrastructure such as the Internet or cellular services. Instead, they utilize ad-hoc short-range communication protocols equipped with mobile devices. Another category is hybrid networks, where the ad-hoc network and the infrastructure-based network can be utilized, allowing devices on both types of networks to communicate with each other. This allows for greater flexibility and scalability in the network, as devices can connect to either the ad-hoc network or the infrastructure-based network, depending on the availability of resources, connectivity, and application requirements. The Nearby API

by Google to communicate with nearby devices and exposure notifications by Google and Apple for contact tracing during the COVID-19 pandemic are the finest examples [17] and [18].

Today almost all mobile devices that are commercially available in the market are equipped with built-in ad-hoc communication protocols i.e. Bluetooth, and Wi-Fi Direct which enable the devices to exchange information over a short distance. Bluetooth is generally used for sharing data across devices when speed is not an issue. such as printers, tablets, mobile, etc. [19]. In contrast, Wi-Fi Direct covers more distance with a high speed as shown in Table 1 along with more details [20]. Wi-Fi Direct is one of the emerging protocols and several applications have already been developed using this protocol and are available on the Android App Store i.e. SHAREit, Wi-Fi Shoot, HitcherChat, Xbox One, etc. The comparison of Wi-Fi Direct and Bluetooth is presented in Table 1 based on various parameters for better understanding.

TABLE 1. Comparison between Wi-Fi direct and bluetooth.

	Wi-Fi Direct	Bluetooth
Range	200 meters	10 meters
Speed	250 Mbps	1 Mbps
Connected Devices	Unlimited	8
Power Consumption	400 mA+	40 mA

A comparison of the effectiveness and limitations of previous research has been conducted by evaluating their usefulness and is presented in Table 2. The primary factors considered include the framework's interaction style i.e. the way users interact with the devices, compatibility with multiple devices, communication protocol (ad-hoc or fixed), and application development support for building applications. The subsequent sections will cover the functioning details of these frameworks.

A. INFRASTRUCTURE BASED FRAMEWORK

This section explains the details of the infrastructure-based framework presented by the researchers to enable cross-device interaction. By exploring the various approaches to cross-device interaction and user needs, we can develop systems and frameworks that are better able to meet the communication and collaboration needs of a wide range of users in various scenarios. The interesting use cases were extracted from these scenarios and considered during the proposed framework design. One example of an infrastructure-based framework is the Conductor system, designed by Hamilton and Wigdor, which allows users to split tasks into sub-tasks and transfer them to multiple devices based on their usability and convenience [15]. This system relies on a centralized, dedicated web socket-based server machine to synchronize data and session states.

Collaborative web browsing is another form of cross-device interaction, where mobile devices communicate with each other using their web browsers. Several frameworks and systems have been developed to support collaborative web browsing and information gathering, including the Magpie framework for semantic web browsing support the collaborative information gathering by notifying the browsing activities between the users [34], a co-browsing system that allows the participant to point at their contents [35], the Multi browsing system that enables the movement of web information across multiple displays [36], PlayByPlay uses the instant messaging model to support the collaborative browsing task between mobile and desktop users [37]. Polychrome facilitates collaborative web browsing/visualization across multiple devices through the web browsers using the PeerJS [25], and the web-based platform VISTRATES helps to form the collaborative work-space for data analysis and visualization purposes for users having multiple devices [38]. In another research, a single web page is distributed into multiple portions and these portions are served across multiple devices using the designed framework instead of mirroring [27], [39].

Recently, we have seen an increase in the usage of smartwatches but due to the limitation of small size, it is difficult to carry out complex daily tasks on them. By considering this issue, a watch-centric system was presented by the researchers to support collaborative task execution by creating a generic workspace of multiple types of devices [28]. The designed system makes it easier to carry out crowd-writing activities by allowing the users of wearable devices to integrate with the system and effectively take part in performing the job. The News system called Agapie expedites the coordination of local and remote staff to work together to achieve the ultimate goal. The wearwrite team also designed the app for mobile and wearable devices [40]. To facilitate the app development, a web-based framework weave was presented by the researchers to ease the development of cross-device applications for mobile and wearable devices. The system used the jQuery (JavaScript) libraries and exposed APIs for sensing and distributing the occurred events across mobile and wearable devices [14].

In 2014, another researcher proposed a collaborative framework for user interface (UI) distribution across the collaborating devices [41]. The framework is composed of two major components, the client library, and the runtime engine. The client library monitors and tracks the changes that occurred at UI and synchronizes them across the devices to maintain the session state. In contrast, the runtime engine manages all devices and their generated distribution changes. Another research of a similar kind, Panelrama supports Web Pages distribution across multiple devices by managing the allocation of portions across the targeted display as per the screen and input modality [29]. Correspondingly, another web-based platform ADaM facilitates collaboration by distributing the UI across the devices by evaluating the capabilities of devices, roles, rights, and preferences [42].

TABLE 2. Comparative analysis of different frameworks.

Name	Interaction Style	Interoperability	Adhoc Network	Collaboration	Self-Adaptation	General Purpose Development (API)
Samsung Flow [21]	Tasks Transfer	Samsung Hardware	✓	✗	✗	✗
AppleContinuity [22]	Tasks Transfer	Apple Hardware	✓	✗	✗	✗
Conductor [15]	Activities Distribution	Android Based Devices	✗	✗	✗	✗
Duet [23]	Watch/Mobile Joint Interaction	Android Based	✓	✗	✗	✗
Webinos [24]	Service Usage	Runtime Installation Requirements	✗	✗	✗	✗
Polychrome [25]	Collaborative Visualization	Browser Based	✗	✓	✗	✗
Webstrates [26]	Collaborative Editing	Browser Based	✗	✓	✗	✗
Websplitter [27]	Web Interface Portion Distribution	Browser Based	✗	✓	✗	✗
Weave [14]	Mobile/Wearable Interaction	Weave Proxy Requirements on Android.	✗	✗	✗	✗
Wearwrite [28]	Mobile and Wearable Crowd Writing	Android Based Devices	✗	✗	✗	✗
Panelrama [29]	Collaboration	Web Applications	✗	✓	✗	✗
Team Phone [30]	Communication System	Android Based Devices	✓	✓	✗	✗
Disaster Management [31]	Alerts and Messaging	Android Based Devices	✓	✓	✗	✗
Enabling Internet Access P2P [32]	Rescue Activities	Android Based Devices	✓	✓	✗	✗
Local Communication System [33]	Communication System	Android Based Devices	✓	✓	✓	✗
Proposed Framework	Customized	Android Based Devices	✓	✓	✓	✓

B. AD-HOC NETWORK BASED FRAMEWORK

Ad-hoc networks are temporary networks that are formed on the fly and allow the devices to communicate with each other without the need for a preexisting infrastructure or central access point. The Samsung flow allows the users of Samsung devices to seamlessly transfer or switch their activities to other devices while working as per the convenience and features offered by the devices [21]. The framework only works with devices equipped with Wi-Fi or Bluetooth and is within proximity as per the defined ranges of communication protocol. It seamlessly transfers the workflow across the devices efficiently within the user's acceptable time. The Samsung flow only works on Samsung hardware and does not support cross-device collaboration activities. Similarly, the continuity feature of iOS supports workflow transfer like Samsung flow across the Apple devices i.e. Mac, iPhone, and iPad only using AirDrop [22].

In recent years, wearable devices have witnessed steady and consistent growth in the commercial market i.e. smartwatches, medical wearables, etc., but it is helpful to use these devices when combined with other devices due to different types of features and capabilities. To address this constraint, the Duet framework presented by the researchers particularly focused on the joint interaction of mobile and wearable devices for the completion of these specific tasks via Bluetooth [23]. Receiving an email message and phone call without picking up the mobile phone using the smartwatch by establishing a pairing channel with the mobile is an example of this interaction. The researchers have proposed a variety of methods to boost the input and output functionality of wearables by enhancing their functionality and design [43], hand gestures using the sensors [44], skin buttons by integrating the tiny projectors with the watches [45], wristband beyond the watch surface [46], employing multiple application on mobile [47], and by pairing them with other gadgets [48], [49].

The use of mobile devices is widely popular because of the sudden calamities that badly affect different regions of the world. People need to communicate with each other or broadcast any information or alerts in case of emergencies using these devices. Unfortunately, these devices rely on applications that only work for fixed infrastructure services and will be useless in case of the absence of internet and cellular services. In such types of mission-critical situations, there is an essential need to provide reliable communication across these devices without the dependency on fixed infrastructure by ensuring the delivery of correct information within the client's stipulated time.

Team Phone is particularly designed for disaster-like situations for carrying out rescue activities by creating a

temporary network of smartphones on the fly. The system is composed of a two-component messaging and a self-rescue system that automatically activates and sends out emergency messages when rescue workers are nearby. The system is based on Wi-Fi Direct communication protocol and evaluated in terms of energy efficiency [30]. Similarly, a Wi-Fi Direct-based alert dissemination protocol is also proposed by A. Shahin and M. Younis that uses service discovery for alerts generation and nearby devices can capture these alerts without physical connection establishment that improves the performance. Moreover, the protocol also facilitates the forwarding of these alerts to other devices [50]. Hence, it is unfeasible and impractical to completely rely on an infrastructure-based disaster management system to carry out rescue activities. An alternate solution based on Wi-Fi Direct is proposed to support stable and reliable communication between victims and rescue workers by enabling internet access using nearby devices [31], [32]. In another research, Fuliang L proposes the local communication system over Wi-Fi Direct that facilitates information sharing across the devices in an environment where internet and cellular services are not available [33]. The system supports inter-group communication and is also tested using a real test bed.

Smartphones are an important tool of communication in a mission-critical or disaster-like situation. However, most of the applications developed for such situations are dependent on cellular and internet communication, which may not be available at all times. A cloud-based disaster management server that requires the application installed on the Android phone and updates the users about the disaster (tsunami, cyclone, or flood) by enabling the cloud-to-device messaging service [51], fixed server-based mobile application to collect the victim's data and send to server using HTTP post protocol [52], another mobile application MyDisasterDroid is used to carry out rescue activities in disaster-affected areas requires network availability [53]. If the infrastructure is suspended or disrupted due to some untoward situation and the victim wants to broadcast information about their location and status to seek assistance and call for rescue activities. In this critical scenario, mobile users can't communicate with others due to the non-availability of internet/cellular services. However, an application like Fire Chat was also developed by the open garden that utilizes the peer-to-peer connectivity architecture and allows the device users to communicate with each other without the dependency of cellular and internet services [54]. The user can perform chat and place calls but is still unable to customize it based on different situations according to their preferences and requirements.

According to the literature review, it is challenging to create a robust, customized application as per the client's requirements that meets the various quality attributes i.e. availability, reliability, performance, fault tolerance, security, and energy efficiency outlined in the introduction. These attributes are important for ensuring that the application is reliable, efficient, and effective in meeting the needs of users. The research presented in the past focused on particular interaction styles and did not facilitate the application developer in developing customized applications over ad-hoc distributed systems as per the requirements. Thus, there is a need to develop a framework that provides abstraction by hiding the underlying complexities and enabling the developers to build customized, robust collaborative applications easily.

III. PROPOSED FRAMEWORK: ROBUST INFRASTRUCTURE-LESS COLLABORATIVE SPACE

The proposed framework is designed to enable the development of robust applications in an ad-hoc distributed setting by considering self-adaptation, data management, network management, and client-customized requirements. The framework presents its components to developers in an abstract way by hiding the underlying complexities and offering a well-defined Application Programmer Interface (API), which facilitates the creation of robust, customized applications suited for ad-hoc environments. It is responsible for forming a reliable ad-hoc network across nearby mobile devices using short-range communication protocols and facilitates the application developer to particularly focus on application requirements instead of worrying about the underlying complexity of the ad-hoc environment. To the best of our knowledge, the prior research works primarily proposed standalone communication systems based on different interaction styles but do not offer any generic framework that encourages application development over ad-hoc network settings by offering a robust ad-hoc environment. Thus, the proposed framework can potentially be used to easily develop a variety of useful collaborative applications such as document sharing, gaming, messaging applications, alert dissemination, resource sharing, etc.

The framework is based on the Layered Architecture style whereby different components have been grouped in different layered, and their functionalities are exposed with the help of Application Programming Interface (APIs) [55], [56]. It is composed of five major components *Controller*, *Services*, *Network Management*, *Communication Service*, and *Storage*. Figure 1 provides a high-level architectural view showing these major components of the framework and their interaction with one another. The *Controller* is the major component responsible for monitoring, control, and management of all other components. It achieves self-adaptation by continuously monitoring all component's responses and adjusting the behavior of the system by analyzing the monitoring results and recovering it from failure to make the system fault-tolerant. *Services* deal with the registration and discovery of different services available on nearby devices within the ad-hoc network range. Ad-hoc network monitoring and configuration are managed by the network management component that is solely responsible for the establishment and maintenance of the network and updates the information almost in real-time, to provide the exact status of the connected devices. *Communication Service* is a very important component of the framework that acts as a listener and publisher of the messages generated by the

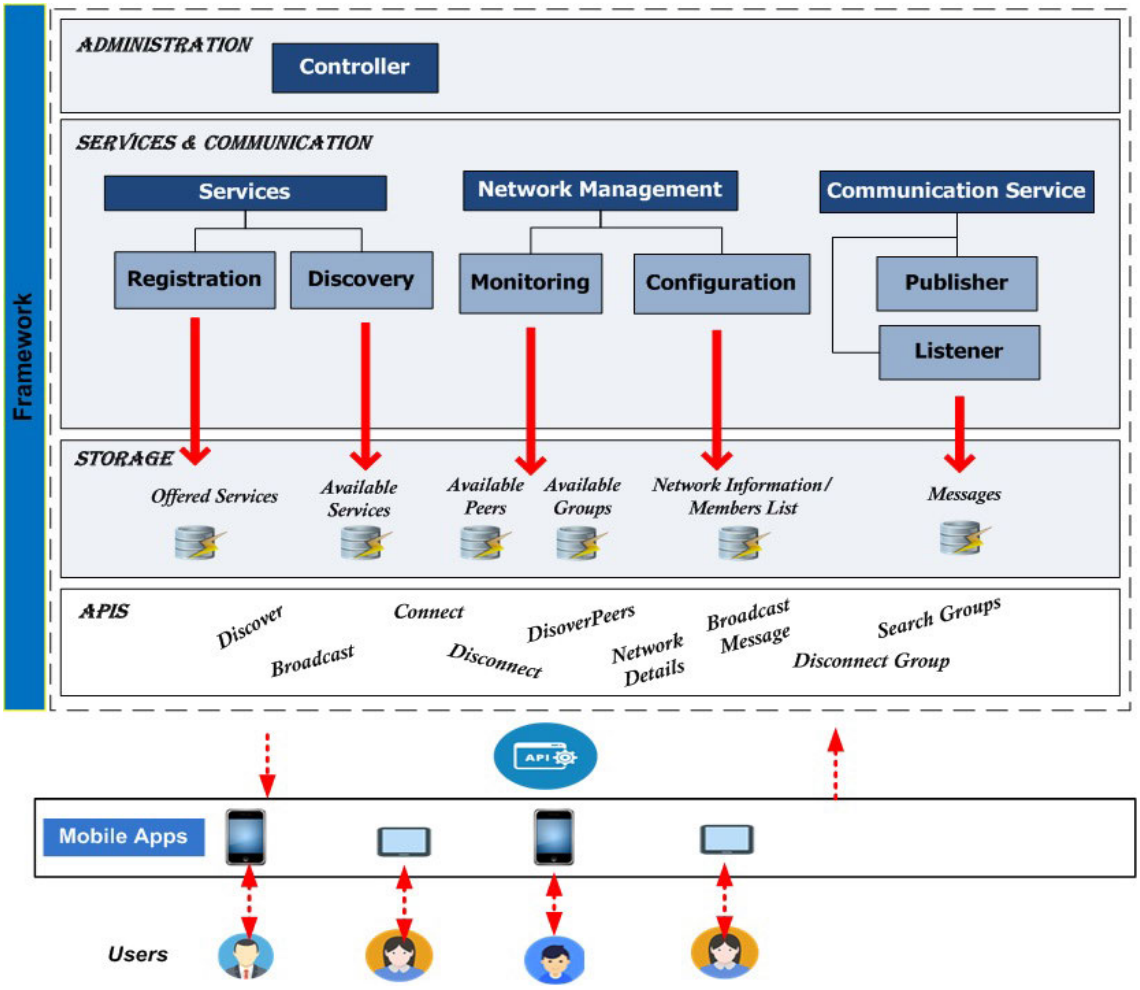


FIGURE 1. High-Level architecture.

devices. *Storage* maintains the data and network repository both at client and server, which is later used by the *Controller* component in case of any intermittent failure or errors to bring back the system to its original state by reforming the network and synchronizing the session state across the all connected devices to ensure data integrity. Furthermore, the architecture and components of the framework are designed with a focus on performance and resource efficiency, so that the applications built using it can consume less battery power.

The functionality of these five components is exposed in the form of APIs, which provide an abstraction layer by hiding the underlying complexity of an ad-hoc distributed environment.

A. CONTROLLER

The *Controller* is the central component of the framework that plays a crucial role in managing and coordinating the activities of the other components. It receives feedback from other components about the state of the network, the functioning of the system, and any events that occur, and uses this information to make decisions and ensure the smooth operation of the collaboration space. In addition to its monitoring and control functions, the *Controller* also initiates an environment and performs any necessary initialization tasks before handing control over to the other components. As the main anchor of the system, it is responsible for setting the stage for the other components to carry out their designated functions.

1) COMMUNICATION PROTOCOL

There are various ad-hoc network protocols that devices can use to communicate with each other, such as Wi-Fi Direct and Bluetooth. The *Controller* is responsible for configuring the device to use the appropriate protocol based on what is available. This involves setting up the device with the necessary communication protocol, which is a set of rules and conventions that define how devices communicate with each other. By setting up the device with the correct communication protocol, the *Controller* can ensure that the device can communicate and collaborate effectively with other devices on the network.

2) BROADCAST INTENTS REGISTRATION

Broadcast intents are used to notify an application when something has happened near the device, for example, a new device found within the range. The event is communicated to the application by sending a message along with event details to perform any necessary actions. The *Controller* registers these intents as listed in Table 3 during the initiating phase and unregisters it after the application closing. It also activates the network component to receive these broadcasts and inform the *Controller* about the occurrence of any unusual activity.

TABLE 3. Broadcast intents.

Intents	Details
Wi-Fi/ Bluetooth State	Wi-Fi P2P/Bluetooth is On/Off.
Peer List	Notify about any change in the peer list identified by the system within Wi-Fi Direct/Bluetoothrange.
Connection/ Disconnection	Notify about connection/disconnection of devices.
Device Configuration	Notify about any change in device's configuration.

3) SCANNING NEARBY DEVICES

Scanning of nearby devices is a crucial aspect of ad-hoc distributed applications, as it enables devices to discover and connect without the need for a central server or other infrastructure. The *Controller* component initiates the scanning activity to continuously search for devices in the surrounding area. This process is known as peer discovery, and it involves sending out messages to find other devices and receiving responses from those devices. The devices can then exchange information about their capabilities and services and can establish a connection if both parties are interested.

4) PEER TO PEER LINK

A peer-to-peer link is a direct communication channel between two devices, without the need for a central server or access point. The devices are equal and can communicate with each other directly. The *Controller* initiates the invitation process and facilitates communication between the devices without replying to a central authority.

5) COMMUNICATION LINK

After successfully establishing of peer to peer-to-peer link between the devices, *Controller* initiates the *Communication Service* for the creation of a socket as the underlying communication mechanism. This virtual tunnel allows the exchange of data between the connected devices. It is identified by a unique combination of an IP address and a port number. The component is designed in such a way as to take minimum time to form a temporary network, as delay or unavailability of services may hinder the effectiveness, availability, and reliability of collaborative applications.

6) SELF ADAPTATION

Self-adaptation is the process by which devices manage their operation without any manual intervention in case of fault occurrence and bring back the system to its normal state. The *Controller* achieves self-adaptation by continuously monitoring all component's responses and adjusting the behavior of the system by analyzing the monitoring results and recovering it from failure to make the system fault-tolerant. In the proposed framework, the Group Owner acts as a leader and plays a key role by managing all devices and their communication. The devices can suffer from disconnection at any time if the group owner disconnects from the group.

To solve this problem, the concept of self-adaptation is employed by the proposed framework. Whereby, it introduces a group reformation feature and allows the devices to seamlessly reconnect and form the network again, and the framework chooses a new group owner. It is important to note that the *Controller* maintains the list of all devices with their Mac addresses and group owner priority values that are calculated based on cost function designed by considering the device capabilities i.e. internal/external storage, battery level. This information is synchronized frequently on all connected devices on every connection/disconnection. In case of group/network disruption, the new group owner device uses this information to connect with the client devices seamlessly without any interruption. The complete step-by-step process of self-adaptation is elaborated in Figure 2.

B. SERVICES

This component is responsible for managing the exposition and discovery of services and enabling mobile devices to provide and consume services. In an ad-hoc distributed setting, mobile services for peer-to-peer applications play the key in the implementation of distributed systems by adopting the Service Oriented Architecture (SOA) architectural pattern [57]. The service is a self-contained function that represents a unit of functionality and can exchange information from other services.

1) REGISTRATION

Devices can expose their services within a short distance to consumer devices by registering their services on the device. For instance, organizers of a conference or event advertise their proximity-based service for the participant's guidance and the participants can register themselves to get the schedule details on their mobile phone for convenience without internet or cellular services dependency. The emergency alerts can be broadcast through the service registration process, and other users can capture these alerts without connecting with these devices. However, they can connect further to share and receive more information based on the scenario. The API takes the two mandatory input parameters *service ID* and *port #* to broadcast the service on a local device and also responds about the execution status, either failure or success, to the *Controller*.

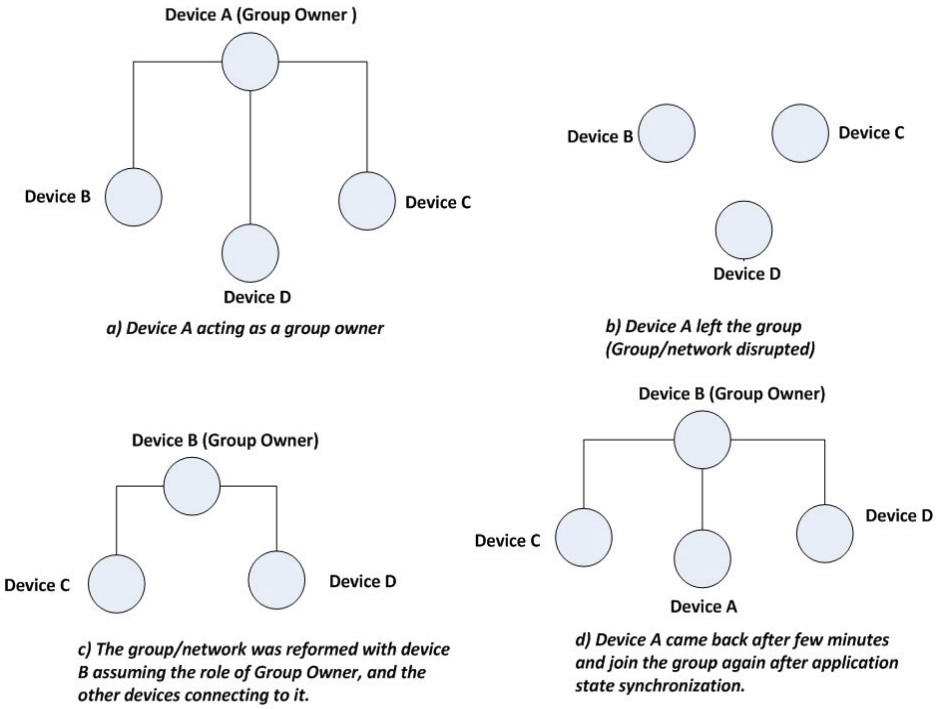


FIGURE 2. Group/Network disruption and reformation.

2) DISCOVERY

Discovery refers to the capturing of nearby available services offered by the devices through the system notifications. It can be the emergency alerts broadcasted by different device users which alerts the users of nearby devices about the happening of some unusual events. Any device that intends to subscribe to a particular service or wants to connect with a device, invokes the discovery process to locate nearby devices and services. The connection details will be utilized for establishing the communication channel and ongoing communication across the devices. Discovery API extracts the network details and stores them for network management.

C. NETWORK MANAGEMENT

The network component is responsible for monitoring and configuration of the ad-hoc distributed system by maintaining the list of all available nodes. Frequent connection and disconnection are normal due to the ad-hoc nature of the network. Therefore, It is the responsibility of the network component to ensure the reliability and correctness of network information.

1) MONITORING

The monitoring component of the framework is responsible for answering the broadcast report by the system. Any device can join or leave the network at any time due to the temporary nature of the network without any prior intimation. In this particular situation, there is a need to continuously monitor the client device's broadcast i.e. connection/disconnection announced by the devices, and act accordingly to manage the resources of the involved devices. The network component is responsible for informing the controller component about the disconnection of any device. Based on this information, the controller synchronizes the device information across the connected devices.

2) CONFIGURATION

There is a need to maintain a list of client devices to record the network topology and update it dynamically. The network component maintains this information using a tree data structure. The network topology reflects the network information and plays a key role in the message forwarding from one device to another, including multiple hops. Direct communication between two different groups is not allowed because they have the same IP address i.e. 192.168.49.1 according to the implementation of Wi-Fi Direct on Android as shown in Figure 3. If a Group Owner (GO) sends a message to another Group Owner (GO) device, it will be routed back to itself. Moreover, the devices cannot connect with multiple groups at the same time and need to leave one group to join the other group. In that case, any client device will act as a relay node for data communication across multiple groups by switching between the two groups. However, devices within the group can communicate directly. However, Bluetooth-enabled devices can directly connect without any limitations.

3) CONNECTIVITY

The *Controller* invokes the request to find the group owner device from the list of available nearby devices and sends the

connectivity invitation to the group owner device to join an existing group. In case of the non-availability of the group, it sends direct connectivity invitations to the available peer to form an ad-hoc network for information sharing across the devices. The group owner will be decided by the framework based on device capabilities. After the formation of the group, new devices can also join the same group. In that case, any device can act as a group owner or client because the main purpose is to establish a communication channel to share and receive information across the devices. In the case of Bluetooth, the device that accepts the invitation can play the role of group owner and serve the all-connected device.

4) GROUP SWITCHING

In an ad-hoc distributed setting where devices are connected without a central authority, they may use various communication protocols such as Wi-Fi Direct, Bluetooth, or Multi-peer connectivity to connect and exchange data. A group in this context refers to a collection of multiple devices that can communicate with each other simultaneously. For example, if there are multiple groups of devices in proximity to each other, the proposed framework can support a tunneling mechanism to allow communication between these groups and enable multi-hop transmission. To facilitate this, any device having the required capabilities can act as a rely-on node and be used to switch between the different groups using a time-sharing strategy, enabling the dissemination of information across all devices that are connected to different groups [58], [59]. This can also help increase the range of the ad-hoc network.

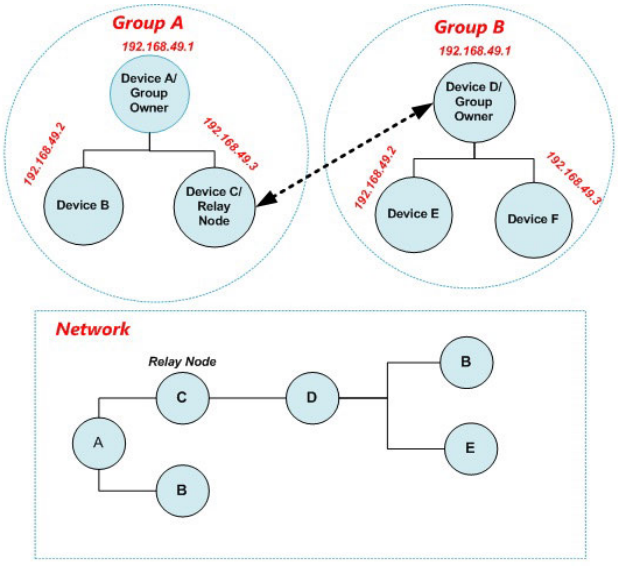


FIGURE 3. Intergroup communication, Device C act as a relay node between Group A and B.

D. COMMUNICATION SERVICE

The *Communication Service* is responsible for the events/data transmission across the connected device and hides the complex communication mechanism by providing high-level abstraction as shown in Figure 4. The *Communication Service* has been developed using the sockets. It is responsible for message communication and also manages the concurrent client's connections efficiently using multi-threading. It is also tasked to ensure the accurate delivery of messages within a user-acceptable time without any errors to ensure reliability. Moreover, if the internet is available, the Communication Service may also use it as a communication medium and send data to a cloud-based server according to the application requirements.

1) SERVER/CLIENT REGISTRATION

Communication service uses sockets that enable two-way event/data communication across the connected devices. It creates the server socket at the group owner device and listens for the client device's request at the specified port. It also stores the client socket information and uses that information for message broadcasting to a particular client. After the establishment of the connectivity link between the devices, the *Controller* executes the *communication service* to set up a communication channel with the host/server device by providing the group owner device IP, port number, and device's mac address. The client devices use this information to establish a socket-based communication link with the host device for data sharing.

2) LISTENER

Listeners wait for the incoming messages from the connected devices and are responsible for sharing the messages with the application for necessary processing. Moreover, it also detects network-related messages and invokes the network configuration management component for updating the network details in the local repository. Moreover, to ensure performance, efficiency, and response issues, individual threads will be created for each client application.

3) PUBLISHER

The publisher is responsible for publishing messages on the network based on the type of the message. To synchronize any update on connected devices, the *Controller* invokes the *Communication Service* event broadcasting API by providing event/message data. When any device connects with the current group, it broadcasts the network information to the newly connected devices. Moreover, messages can also be posted on the server by determining the availability of the Internet.

4) MESSAGE TYPES

Two types of messages will be used by the *Communication Service* for sharing information across the devices. The first one is the network/device information and, the second is the application-specific data that needs to be disseminated across the connected devices. It is pertinent to mention that the network tree will be created/updated based on network syncing messages and will also be used to display the

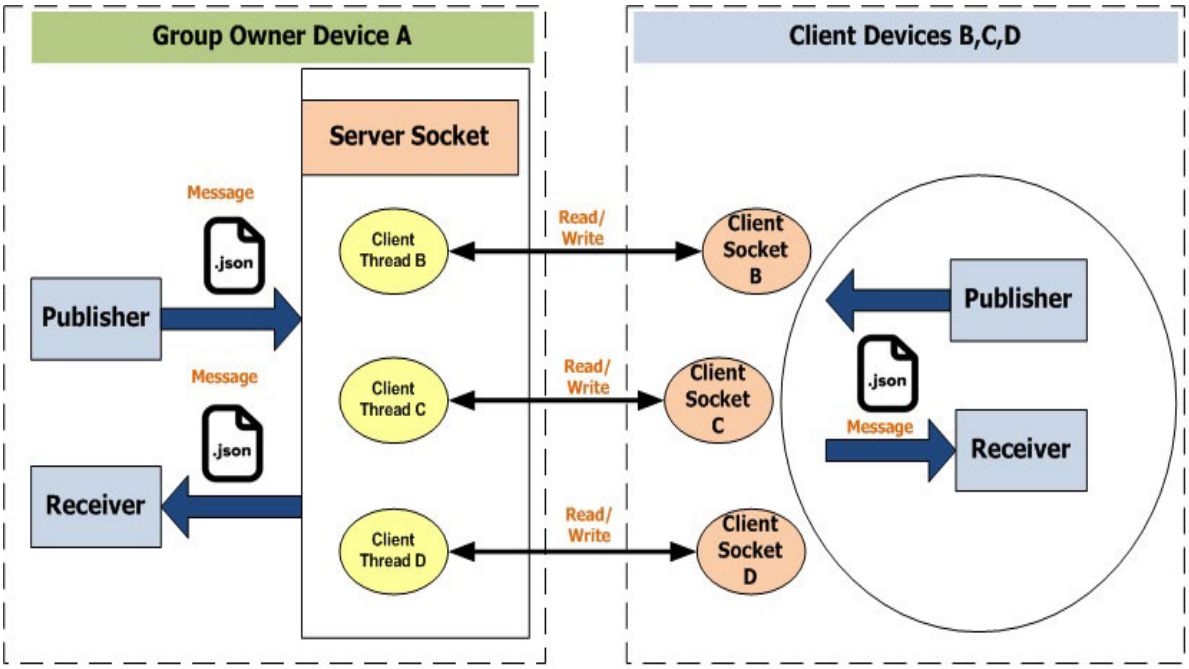


FIGURE 4. Communication service.

information of available devices on UI. When the *Controller* receives the joining or leaving of any device on the network, it will inform all members through the broadcast.

TABLE 4. Message attributes details.

MAC	AC:48:37:153C:32
Device IP	192.168.49.3
Type	Network Synch, Application Data
TimeStamp	YYYY-MM-DD HH:mm:ss.SSS
Relay Node	AC:65:37:482C:39
Category	Promotional, Rescue, Alert, Emergency

The message attributes are also shown in Table 4 for better understanding. The MAC attribute will be used for storing the MAC address, while the device IP attribute stores the IP address of the device over the network. Whereas, the type attribute helps to identify whether the message is meant for network synchronization or carries the data related to the distributed application. The timestamp attribute helps to ensure the synchronization of activities as in the case of collaboration activities, we might need the timestamp to manage the execution order of the messages. A relay node will be used to identify the details of devices that are acting as a bridge between two different groups. The category is an optional parameter that can be used to identify the relevant messages based on the user's interests. It is pertinent to mention that the message communication will be based on the JavaScript Object Notation (JSON) data format due to its simple and flexible design. It enables the users to easily read and understand the data in most cases. Moreover, data can be pushed to the live server using the Web API on the availability of the Internet on that particular node. A binary format for data exchange can also be considered to reduce the size of the message.

The proposed framework facilitates the development of different applications based on user requirements and needs. By considering different application areas and scenarios, we have proposed the design of alert dissemination and SOS call generation applications to highlight the need and importance of these applications during different real-world situations. Listing 1 and Listing 2 provide illuminating samples of application data formatted in JSON. In Listing 1, we delve into the representation of data from an alert dissemination messaging system, unraveling the intricacies of its structure and content. Similarly, Listing 2 offers a detailed exploration of data originating from a promotional messaging application, shedding light on its nuanced aspects. These comprehensive examples not only serve to showcase the diversity of JSON-based data but also facilitate a deeper understanding of the underlying structures and functionalities inherent in alert and promotional messaging systems.

However, the application developers can structure the application data as per the convenience and requirements of the application. It is clearly shown from the listing that the transfer of data across the devices is quite simple and flexible. Any novice developer can develop applications for ad-hoc environments without worrying about the complexities of ad-hoc distributed systems. In the case of a mission-critical application, such types of network disruption can cause

```
{
  "MACAddress": "AC:78:37:173C:36",
  "DeviceIP": "192.168.49.3",
  "Type": "Message",
  "TimeStamp": "2021-03-12 08:12",
  "RelayNode": "",
  "Category": "Alert",
  "Message":
  "A tornado has been sighted or indicated
      by weather radar. There is imminent
      danger to life and property. Move to
      an interior room on the lowest floor
      of a sturdy building. National
      Weather Service, Canada"
}
```

LISTING 1. Alert dissemination messaging.

```
{
  "MACAddress":
      "AC:78:37:173C:36","DeviceIP":
      "192.168.49.3",
  "Type": "Message", "TimeStamp":
      "2021-03-12 08:12",
  "RelayNode": "", "Category": "Emergency",
  "GeographicalLocation": [
    { "longitude": "149.452236",
      "latitude": "-35.275291" }
  ],
  "Message": "I am badly trapped in a
      Trafford Centre lower floor due to
      fire. Please help.",
  "Picture": [
    { "width": "403", "heignt": "393",
      "base64": "data:image /png;base64,
          iVBORw0KGgoAAAANY.." }
  ]
}
```

LISTING 2. Promotional messaging.

serious damage to the lives of people and pose serious challenges to the robustness of the system. By considering these challenges, we have given significant importance to this feature and ensured the lossless and synchronized flow of information across the devices by reforming the network again in case of network failure. The framework will encourage the developers towards the development of such applications and also significantly reduce the application development time.

E. STORAGE

During the collaboration activities, devices generate different types of messages that are required to be saved to maintain the session and network state during the lifetime of the collaboration space. The proposed framework is considering JSON (JavaScript Object Notation) to transmit data across the devices. Because, these are lightweight, easy to generate, and parse [60].

1) MESSAGE CONTAINER

The framework also maintains its data repository both at the client and server ends due to the ad-hoc nature of the network, where any device can leave or join the network at any time without any prior intimation. All the data is stored in a container along with the timestamp. The timestamp information is used by the *Controller* for data synchronization on the devices in case of network disruption and the disconnection of devices. In case a new device joins the network or an existing client rejoins after some interval due to some disruption, the data will be synchronized on the devices. It is pertinent to mention that the timestamp associated with the data will be utilized by the *Controller* to update only generated data instead of all data. It will also significantly improve the efficiency and response of the application and ensure the robustness of the system.

2) GROUP MEMBERS (DEVICES)

In case of connection/disconnection of devices, the Host device sends the updated group member's devices list to all the client devices for synchronization purposes. The list will contain the Device Name, MAC Address, Group Owner Priority, IP, and Network Protocol information that will be used for the identification of nodes and network reformation.

3) AVAILABLE PEERS

Due to the temporary nature of the environment, devices can leave the network at any time during collaboration activities. In that particular scenario, it is essential to detect the left devices and remove them from the network list to clear the unmanaged resources. *Controller* maintains the list of all available peers/devices within the Wi-Fi Direct/Bluetooth range and uses that list for cleaning the unmanaged resources by making a comparison with the group members list provided by the observer of the events. In this regard, it registers the broadcast intent ''Peers Changed'', which is received by the Events Observer in case of any change in the peer list and requests the new list of available peers/devices from the Android system.

IV. EXECUTION CYCLE AND APPLICATION DEVELOPMENT SUPPORT

This section presents the working of different components of the framework with the help of an execution cycle using a case study of the Disaster Management System. It also discusses how the proposed framework facilitates the application developers in developing cross-device collaborative applications.

Several crowdsourcing applications have been developed to deal with emergency and disaster scenarios using mobile devices. However, most of these applications rely on standard communication methods and a web/cloud-based infrastructure. During a disaster scenario, cellular and internet services may not be available, rendering mobile devices useless. This

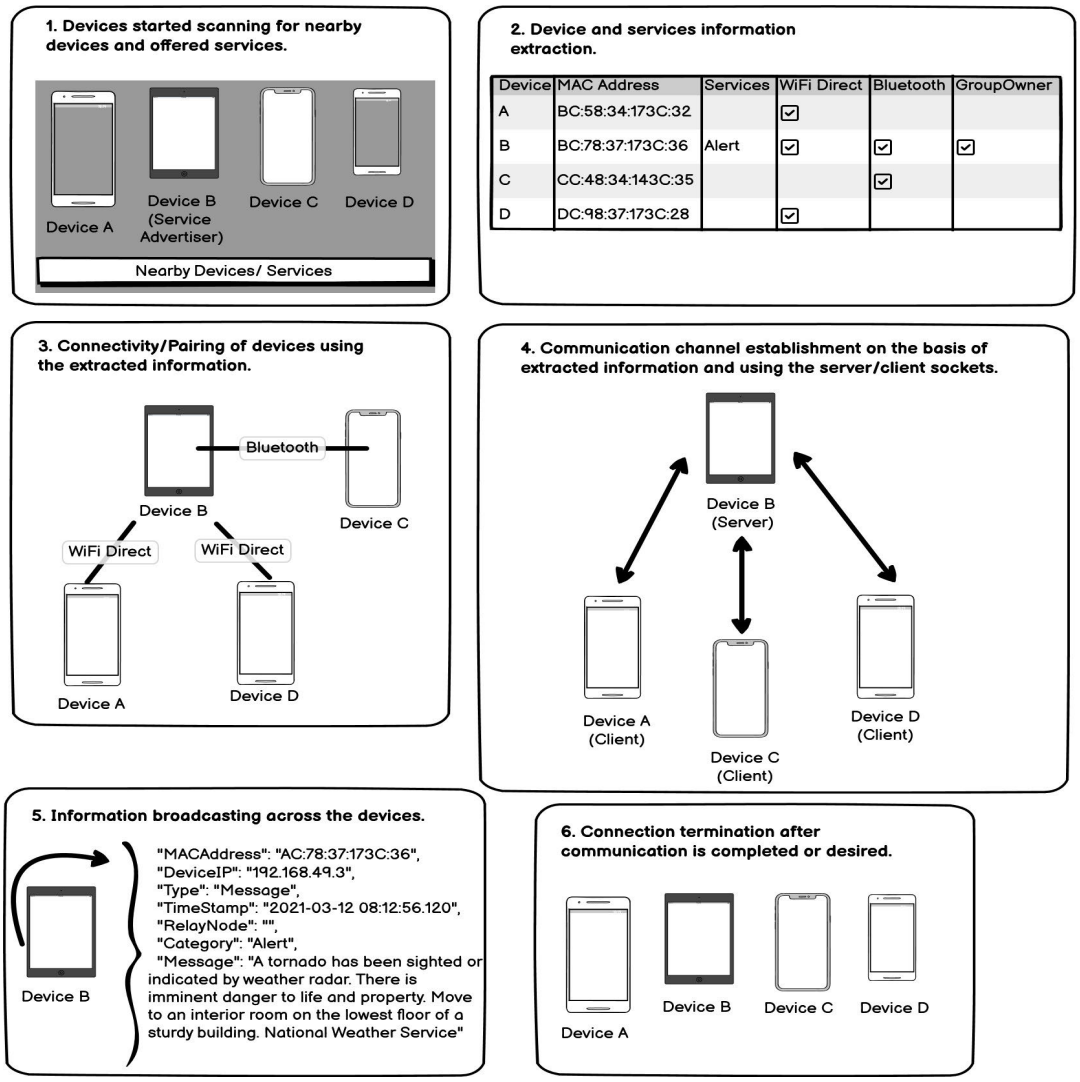


FIGURE 5. Network formation and information sharing across the devices.

application aims to create an ad-hoc distributed system using nearby devices, enabling mobile users to communicate with each other even when standard communication methods are unavailable.

A. EXECUTION CYCLE

To establish a temporary network for exchanging information among devices, these devices perform a series of processing steps to maintain the collaboration space. The execution cycle has been grouped into five major steps, as illustrated in Figure 5.

1) DEVICE DISCOVERY

At the start, devices will scan the network to find the nearby available devices and offered services within the range and request some information from them. The devices will respond to the request and provide the requested information.

2) DEVICE/SERVICE INFORMATION

The given device information will consist of unique MAC addresses and Device Protocols that will be used for establishing connections with the devices.

3) CONNECTIVITY/PAIRING

Devices will form the network by connecting with each other on the basis of extracted information and available protocols.

4) COMMUNICATION CHANNEL ESTABLISHMENT

A communication channel will be established by creating the server and client sockets that will be used later for reading and writing data streams across the devices.

5) INFORMATION SHARING

After the successful establishment by the communication channel, data/information will be shared in JSON format across the connected devices.

TABLE 5. List of APIs.

API	Description	Method	Parameter	Response
Nearby Device Scanning	Initiate the scanning process and creation of nearby available devices list.	DisoverPeers	Call Back Function	List of devices
Ad-hoc Network/ Groups Scanning	The API provide the list of nearby ad-hoc network/groups of devices.	SearchGroups	Call Back Function	Group/ Network Details
Check Wi-Fi	Check the availability of Wi-Fi on the device.	IsWiFiEnabled	Call Back Function	Status (True/False)
Check Bluetooth	Check the availability of Bluetooth on the device.	IsBluetoothEnabled	Call Back Function	Status (True or False)
Broadcast Service	Expose the offered services to nearby devices	BroadcastService	Call Back Function, Service Name, Listening Port	Status (Success or Failure)
Create Group	Forcefully select the device to act as a network hosting/group owner device.	CreateGroup	Call Back Function	Status (Success or Failure)
Services Discovery	Discover advertised services offered by the nearby devices.	DiscoverServices	Call Back Function	List of Services
Send Connectivity Invitation	Send an invitation to connect with the device.	ConnectToDevice	Device Mac, Call Back Function	Status (Accepted or Rejected)
Broadcast Message	Broadcast Message to the connected devices.	BroadcastMessage	JSON Data Object, Devices List, Call Back Function	Status (Success or Failure)
Disconnect	Disconnect with the ad-hoc network by forcibly closing the connection.	Disconnect	Call Back Function	Status (Success or Failure)

6) CONNECTION TERMINATION

When the communication is complete or when desired, terminate the connection between the devices. Connection termination can be achieved using the specific protocol's APIs or by closing the network connections.

B. APPLICATION DEVELOPMENT SUPPORT

The challenges faced by the developers in producing robust software applications for the ad-hoc environment have already been discussed in the introduction of Section I. The proposed framework manages the complexity of the ad-hoc distributed environment by providing a high level of abstraction and encourages the application developer by providing easy-to-use APIs. The proposed framework provides the following features to facilitate the application developer in making collaborative applications in an ad-hoc setting.

1) APIS

To facilitate a better understanding, a list of a few common APIs is provided in Table 5 along with the details of functionality and parameters. These APIs can be used to perform a variety of tasks and provide access to a range of functions. By using these APIs, developers can build robust applications as per their requirements. In addition, the proposed framework manages the mentioned below activities on the behalf of application by hiding the underline complexities efficiently to ensure seamless system operation.

a: COMMUNICATION CHANNEL ACROSS DEVICES

The framework facilitates information sharing across the devices by establishing a socket channel. It is the core responsibility of the framework to take care of the establishment of a communication channel between different devices without compromising performance.

b: STATE MANAGEMENT

The Storage is responsible for maintaining the application state across the devices. In case of connection/disconnection, data will be automatically synchronized across the connected devices.

c: NETWORK REFORMATION

Due to the temporary nature of the network, any device can connect/disconnect from the network at any time with any prior intimation. In case of group owner left, the network will be dissolved and require reformation. In such a scenario, the framework adaptation service will be activated, and reform the network to carry out activities by declaring any of the devices as a group owner.

2) EASE IN DEVELOPMENT AND REDUCE OF LINES OF CODE:

The proposed framework hides the complexity of code by providing a high level of abstraction and exposing the

easy-to-use APIs. Using shorter lines of code is more efficient than spreading the code over several lines. If you have more lines of code, there are more places for bugs to hide, and finding them might be more of a hassle. The sample code listing 1, 2, and 3 illustrates how the framework makes it easier to develop applications for ad-hoc distributed settings. For example, broadcasting a message to nearby devices requires setting up a peer-to-peer ad-hoc network connection and a socket-based communication channel for data transfer. Similarly, maintaining data consistency and performance, which is critical for system robustness and also requires complex code involving resources and multiple thread management.

Code Listing 1 Broadcast Service

```
s : Service Name, p: Listening Port #
    ▷ CommandStatus Status notifies about the execution status
Call BroadcastService(s, p, CommandStatus)
if CommandStatus = true then
     Output: Service Broadcast Successful
else
     Output: Service Broadcast Failed
end if
```

Code Listing 2 Broadcast Message

```
dl : Devices List, appdata: Data
    ▷ CommandStatus Status notifies about the execution status
Call BroadcastMessage(dl, appdata, CommandStatus)
if CommandStatus = true then
     Output: Message Broadcast Successful
else
     Output: Message Broadcast Failed
end if
```

3) EASE IN TESTING

One of the main challenges faced by the application developers is to test the application on multiple devices to observe the behavior of each device. Moreover, the availability of diverse types of devices in the market with different OS Versions makes application development more difficult. However, the proposed framework provides ease by offering easy-to-use APIs to the developers without worrying about the ad-hoc distributed system complexities and focusing on application development.

There are some frameworks and systems proposed by researchers in the past that focus on particular interaction styles i.e. Messaging systems, activities distribution, collaborative visualization, etc. highlighted in the literature review. However, those frameworks do not facilitate the application developer in developing customized applications as per the user requirements.

Code Listing 3 Auto Connectivity of Devices

```
mac:MacAddress, p:Port #
    ▷ CommandStatus notifies about the execution status
Call SearchGroup(CommandStatus)
if CommandStatus = true then
     Output: Group Owner found, group is available.
    m=macaddress                    ▷ Get Device MacAddress
    p=portno                        ▷ Get port #
    Call Join(mac, CommandStatus)   ▷ For P2P Connection
    if CommandStatus = true then
        Output: Connected Samsung-A5
    else
        Output:  Request rejected
    end if
else
    Output: Group is not available, connect with the available peer
    Call ConnectToPeer(CommandStatus)
    if CommandStatus = true then
        Output: Connected to Huwaei-101
    else
        Output:  Request rejected
    end if
end if
```

V. EVALUATION

The proposed framework has been rigorously evaluated by performing a diversified set of experiments to test the functionalities offered by the framework. Whereby, the experiments have been designed by considering different real-world scenarios and tested on various operating parameters to prove the efficacy of the proposed framework. Furthermore, a comparison of the proposed work with existing similar research works has also been presented.

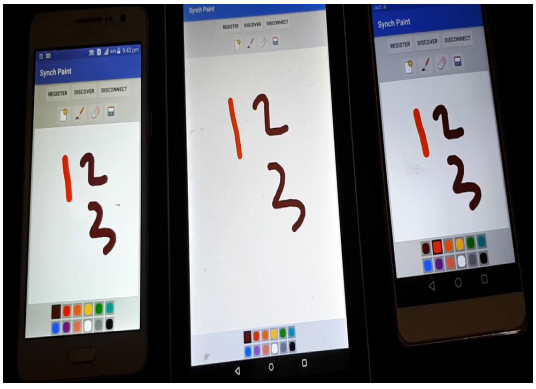


FIGURE 6. Collaborative paint application.

A. EXPERIMENTAL SETUP

This section presents the evaluation of the proposed framework with a variety of experiments designed by involving a

TABLE 6. Experiments and operating parameters (defaults in bold text).

Objective	Operating Parameters	Variations/Values		
P2P Network formation through Service Discovery	Devices	Samsung Tab 4	OS	Kitkat 4.4.2
		Samsung Grand Prime		KitKat 4.4.4
		Q-Tab		Lollipop 5.1
		Samsung Grand Prime		Lollipop 5.0.2
		Huawei Honor 5x		Marshmallow 6.0.1
		Samsung Galaxy A5		Nougat 7.0
		Samsung Galaxy A52		Android 11
Synchronization (Heterogeneous Devices)	No. of Devices	3 - **5** - 7		
	Type of Devices	Group Owner, Client		
	Operations Performed	100, **200**, 500		
Self Adaptation	No. of Devices	3 - **5** - 7		
Device Management	No. of Devices	3 - **5** - 7		
Power Consumption	Operations Performed	100, **200**, 300		

range of operating parameters. These experiments have been conducted on a test bed prepared by developing a specialized collaborative paint application over the proposed framework. This collaborative paint application helps critically reviewing the performance of the proposed framework for many different real-time interaction scenarios/styles which involve seamless collaboration across devices with distributed interfaces as shown in Figure 6. The application has been customized to evaluate the robustness of the system in terms of quality attributes including reliability, fault tolerance, and synchronization to ensure the client's degree of trust. The application size satisfied the storage limitation of mobile devices, i.e. 4.93 MB, and does not have any memory overhead as anticipated from the framework. It is pertinent to mention that Wi-Fi Direct has been used to establish wireless connectivity among the devices for majority of the experiments conducted in this research.

1) EXPERIMENTS AND OPERATING PARAMETERS

The parameters mentioned in Table 6 have been considered for the evaluation of the proposed framework. The experimental approach involves designing experiments and operating parameters specifically tailored to evaluate the robustness and self-adaptive behavior of the framework. The primary goal is to ensure the framework's capability for successful data delivery across devices and its ability to recover from failures. Further, the experiments also compare the proposed work with similar studies carried out by other researchers.

B. RESULTS

This section presents and discusses the design and results of experiments conducted under a variety of operating conditions and settings. The experiments were carefully conducted, and the results were thoroughly analyzed to provide valuable insights. The discussion of the results highlights key findings and their implications for further research and practical applications.

1) P2P NETWORK FORMATION THROUGH SERVICE DISCOVERY

It is crucial to test the application's compatibility across multiple devices of various manufacturers, as the ad-hoc distributed system created using Wi-Fi Direct can include multiple devices. Additionally, the version of Android may vary depending on the device's specifications and support. By testing on multiple devices, we can ensure that the application works seamlessly on all of them. The service discovery and P2P network formation experiments were conducted on various Android-based devices with different OS versions as shown in Table 7. The results show that the P2P network formation took less than a second, which is acceptable and satisfied the robustness in terms of performance.

In the case of the older KitKat versions of Android, we have found service discovery issues with the old version, which may be due to the particular device manufacturer or OS version. However, the service discovery worked well across the same KitKat version. The results also show that the Wi-Fi Direct support has been significantly improved in the latest version of Android.

2) SYNCHRONIZATION

Synchronization refers to the term, where multiple processes running on multiple devices are joined at a certain point by keeping their states identical and accurate across the devices. By considering this fact, state synchronization is ensured by keeping multiple copies of the data across the collaborating devices coherently to maintain data integrity and efficiency.

a: EFFICIENCY

The information-sharing time across the peer devices is the major parameter to evaluate the efficiency of the network. In the case of distributed applications, we cannot afford delay and require the data to be synchronized immediately on all connected devices involved in the collaboration activity.

TABLE 7. Service discovery time across devices with different manufacturers and OS versions.

	Samsung Tab 4 KitKat 4.4.2 (ms)	Samsung Grand Prime KitKat 4.4.4 (ms)	Q-Tab Lollipop 5.1 (ms)	Samsung Grand Prime Lollipop 5.0.2 (ms)	Huawei Honor 5x Marshmallow 6.0.1 (ms)	Samsung Galaxy A5 Nougat7.0 (ms)	Samsung Galaxy A52 Android 11 (ms)
Samsung Tab 4 KitKat 4.4.2	**882**	853	✗	✗	✗	✗	✗
Samsung Grand Prime KitKat 4.4.4	854	**861**	✗	✗	✗	✗	✗
Q-Tab Lollipop 5.1	✗	✗	**838**	993	677	912	889
Samsung Grand Prime Lollipop 5.0.2	✗	✗	993	**918**	764	1088	990
Huawei Honor 5x Marshmallow 6.0.1	✗	✗	677	764	**634**	573	585
Samsung Galaxy A5 Nougat 7.0	✗	✗	912	1088	573	**520**	535
Samsung Galaxy A52 Android 11	✗	✗	812	853	524	495	**492**

An experiment was conducted to record the operation synchronization time across the devices. The objective of this experiment is to assess the impact on operation synchronization time by increasing the number of operations. The results show an average time of 60 ms to 118 ms has been recorded for synchronizing the session state of all devices. It is clearly shown in Figure 7 (a) that an increase in the number of operations has a negligible, small effect on the synchronization time.

b: DISTANCE

In a collaborative ad-hoc environment, the devices may be located at varying distances from each other, which can affect the performance of the application. An experiment was conducted to evaluate the effect of distance on the operation synchronization time, as shown in Figure 7 (b). The operation synchronization time was evaluated by introducing different distance ranges between the devices with a gradual increase in distance. The results indicate the impact of distance on the operation synchronization time. It shows that performance decreases linearly with an increase in the distance between the devices. However, it is still acceptable to develop messaging applications in ad-hoc environments.

c: DATA INTEGRITY

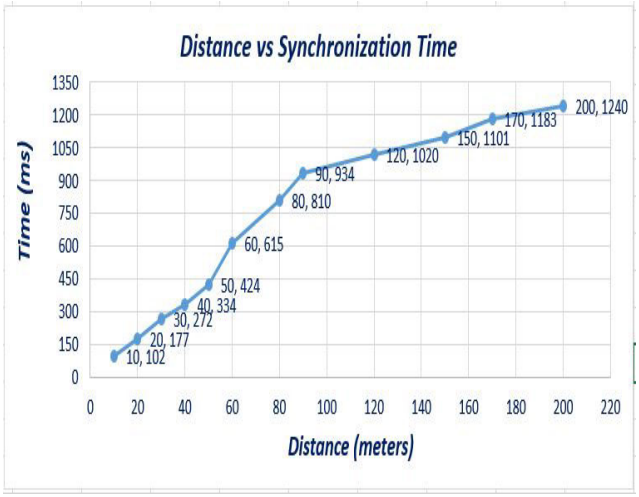
Data Integrity refers to the assurance of data consistency and accuracy across the devices connected with the Wi-Fi Direct group. It is an important aspect and requires to be tested to ensure the harmonization of data across the connected devices.

Another experiment was conducted to ensure that the actions performed by different users on their devices were synchronized in the correct order while maintaining the integrity of the collaborative space. This experiment was conducted by inviting 5 different users along with their devices to participate in the collaboration activity by performing different operations. The users of the devices have performed different operations using the paint application simultaneously. After the completion of the activities, we validated the operation synchronization across all devices by comparing the log files of each device. The analysis shows that all operations are successfully synchronized without any data loss. However, in a few cases, the operations were delivered out of sequence as per the timestamp of the message. This is very much possible in such heterogeneous and ad-hoc settings. The average number of out-of-synch operations for a different number of total operations have been presented in Table 8. The framework was able to address this issue by not only identifying such out-of-synch messages but also managing to perform them by holding a few messages for a specific period to ensure the sequence of activities on each device.

3) SELF ADAPTATION

Self-adaptation is the process by which devices manage their operation without any manual intervention and bring back the system to its normal state in case of fault occurrence. In an ad-hoc distributed system, all devices communicate with each other in a peer-to-peer manner using Wi-Fi Direct without the need for any fixed infrastructure or access point and can join or leave the network at any time without any prior intimation.

(a) Comparison of operation synchronization time.



(b) Comparison of operation synchronization time with distinct distance ranges.

FIGURE 7. Comparison of time and distance.

TABLE 8. Operations synchronization sequence across the devices (Heterogeneous Group of Devices).

Total Operations	Out of Sequence
50	8
100	17
200	18
400	24
600	32

In such kind of situation, there is a need to tackle these scenarios to maintain the stability of the ad-hoc distributed system. There are two possible scenarios: (i) The group owner gets disconnected; (ii) A client device gets disconnected.

In the proposed ad-hoc collaboration space framework, the Group owner acts as a leader and plays a key role by managing all devices and their communication. Group Owner (GO) can leave the network at any time without any prior intimation. In the experiment, as shown in Table 9, a heterogeneous Wi-Fi Direct group of devices was created and asked the users to perform drawing operations. After 10 minutes, we intentionally disconnected the group owner. The test was conducted on 3, 5, and 7 groups of devices to check the framework's effectiveness. The framework successfully detected the group owner left and took almost a few seconds in the group reformation process as shown in the results. The result has clearly shown the framework's strength to ensure the smooth running of the system by forming the group again.

TABLE 9. Group reformation after group owner left without any prior notice to the client devices.

No. of Devices	Reformation Time
3	8 Sec 89 ms
5	10 Sec 48 ms
7	14 Sec 48 ms

In the other scenario, a client device can join the collaboration space at any time during the collaboration activities. There is a need to synchronize the current state of an application on the new device to maintain data integrity across all connected devices. In an experiment as shown in Table 10, the session state synchronization was evaluated on a new client device after joining the in-progress session of the Wi-Fi Direct group of heterogeneous devices. The result has shown that the session state was synchronized in a few seconds ranging from 2 sec to 8 sec with 100-600 operations, which is quite an acceptable time for the users.

TABLE 10. Session state synchronization on a new client device after joining the in-progress session (Heterogeneous Group of Devices).

No. of Operations	Session Update Time
100	2 Sec
300	6 Sec
600	8 Sec

To make the framework more intelligent, we are maintaining multiple copies of data on all client devices along with the timestamp information. In an experiment as shown in Table 11, the client disconnects from the collaboration space and rejoins after some time. The framework improved the performance of the application by synchronizing the updated operations only using the client device data repository and timestamp information. It is evident from the results that the session update time has been significantly improved by avoiding duplicate operation rendering on the client device.

4) DEVICE MANAGEMENT

In the case of ad-hoc collaboration space, device management refers to the efficient management of all devices involved

TABLE 11. The client rejoins after a short period of disconnection detected by the framework and updates the session state efficiently.

Operations	Synchronized Operations (Before Client Left)	Session Update (Time)
100	50	1 Sec
300	150	3 Sec
600	300	4 Sec

in the collaboration activities, which includes devices, session state, and data management. During the collaboration activity, devices can join or leave the network at any time without any prior intimation. In that situation, it is required to continuously monitor the network and respond to the new connection and disconnection of devices.

An experiment was performed by creating a network of multiple devices, as shown in Table 12. The results indicate that the framework quickly detected when a client device disconnected and removed it from the list of connected devices. This was done to free up any unmanaged resources and ensure that the updated list of devices was synchronized across all connected devices, ensuring smooth system operation.

TABLE 12. Time taken by the group owner to detect the left device and synchronize the updated list on all connected devices (Heterogeneous Group of Devices).

No. of Devices	Client Left Detection
3	1385 ms
5	1549 ms
7	1732 ms

5) POWER CONSUMPTION

Mobile devices functioning primarily depends upon the battery and when we run multiple activities on these devices i.e. games, applications, Wi-Fi, etc., the battery consumption rates become higher and drain the battery more quickly due to heavy processing. We have seen a surge in mobile application development in recent times, but energy consumption factors are usually ignored during app development. It is highly recommended that the designed applications should be lightweight and energy-efficient to support more battery hours [61], [62]. By considering this fact, The architecture and components of the framework are designed with a focus on performance and resource efficiency.

An experiment was designed to estimate the battery consumption as shown in Table 13. The experiment was conducted in three different sessions with a time duration of 20 minutes. The results showed an estimated 1% decrease in battery charging for 100 operations during these 20-minute sessions. The result proves the framework's effectiveness in case of a disaster-like situation where rescue activities may take a long time due to disruptive infrastructure.

TABLE 13. Energy consumption of devices during the collaboration activities.

No. Of (Devices)	Operations	Time (Mins)	Battery
3	100	20	1%
5	200	20	2%
7	300	20	3%

TABLE 14. Device connectivity using Bluetooth (Heterogeneous Group of Devices).

No. of Devices	Discovery/Connectivity Time
3	6 Sec
5	11 Sec
7	15 Sec

TABLE 15. Time taken by the message to deliver across the devices on Bluetooth-based network (Heterogeneous Group of Devices).

No. of Operations	Average Time
50	1.2 Sec
100	1.4 Sec
200	1.6 Sec
400	1.9 Sec
600	2.1 Sec

TABLE 16. Time taken by the message to deliver across the devices on the hybrid network based on Bluetooth and Wi-Fi Direct (Heterogeneous Group of Devices).

Operations	Wi-Fi Direct Devices	Bluetooth Devices	Average Time
50	2	1	1.4 Sec
100	3	2	1.7 Sec
200	4	3	1.8 Sec

6) HYBRID NETWORK OF COMMUNICATION PROTOCOLS

A few experiments have also been conducted on the Bluetooth and hybrid network of Bluetooth and Wi-Fi Direct to evaluate the device discovery and connectivity time. According to the results, the device discovery and connectivity took 14 to 15 seconds on average time to connect different client devices with the master device as shown in Table 14 on a Bluetooth-based network.

However, it will be significantly improved in the case of already paired devices. Data transmission across the devices is a little slow but suitable for information dissemination across the Bluetooth-enabled devices as shown in Table 15. By considering these results, the framework proved its effectiveness by covering more application areas, as discussed in the introduction section. Moreover, if the device does not support Wi-Fi Direct, Bluetooth can be used for connectivity

TABLE 17. Comparative analysis with existing frameworks across various evaluation parameters.

Frameworks/Systems	Performance (Subject to hardware specs)			Throughput (Factors Influencing)			Self-Adaptation	
	Data Rate/Delivery Time	% of Loss	Discovery	Distance	Data Size	No of Operations	Node Failure	Session State Synchronization
Team Phone	20 Mb/s	-	-	✓	-	-	-	-
Enabling Internet Access P2P Framework	5.54 seconds to reach the web server (Internet)	-	0.5 to 2 seconds	✓	-	-	-	-
Disaster Management System Wi-Fi Direct Based	-	1.11	-	-	✓	-	-	-
Local Communication System	36 Mb/s	1	-	✓	-	-	✓	
Proposed Framework	50 ms to 600 ms	0	492 ms to 1088 ms	✓	-	✓	✓	✓

and information sharing with the other devices. In the last experiment as shown in Table 16, operation synchronization time in a hybrid network consisting of Bluetooth and Wi-Fi Direct has shown some overhead to the low speed of Bluetooth-based devices. But it can be utilized for message dissemination.

C. COMPARATIVE ANALYSIS WITH EXISTING STUDIES

This section presents the comparative analysis of the proposed research work with the existing similar efforts, as presented in Table 17. Many of them conducted experiments using different methods such as simulations, and real testbeds of smartphones by considering different operating parameters i.e. performance and throughput. While, in some cases, the user experience was also collected through questionnaires. Whereas, in a few cases, developers' feedback was analyzed by allowing them to create applications by providing an authoring environment. However, there is no significant trace for the validation and evaluation of robustness in the existing research work.

The results show the robust performance of the proposed framework in terms of data delivery and device discovery without any data loss. However, the distance and number of operations naturally affected the throughput just like other frameworks. Self-adaptation was not verified in most of the studies, which plays the key role of managing the operations without any manual intervention and bringing back the system to its normal state in case of any intermittent failures. Thus, the proposed research work in this article improves upon two different perspectives. This work rigorously evaluates the robustness of the framework by not only conducting experiments involving the reformation of the network due to node failure as discussed in the evaluation section, but also ensured the synchronization of the session state across the devices to continue operation without disrupting the system functionality.

Lastly, another significant additional contribution provided by this research is the provision of abstraction for the application developers with well-defined Application Programming Interfaces (API). These APIs enable the developers with ease of application development without worrying about the complexities of the underlying distributed system, and easily managing many different possible intermittent failures, which are taken care of by the proposed framework.

VI. CONCLUSION

This article presents a framework for developing robust, reliable, and efficient collaborative applications in ad-hoc distributed settings. The components of the proposed framework have been discussed in detail to illustrate how they work together to create a dependable system. The functionality of these components is exposed to developers using well-defined APIs, providing an abstraction for application developers. This enables them to develop robust applications without worrying about the ad-hoc connectivity of the underlying infrastructure. Thereby, it significantly reduces the lines of code for developing such applications.

An extensive experimental evaluation was conducted to demonstrate the usefulness and efficiency of different components of the proposed framework, employing heterogeneous settings. The results showcase the framework's effectiveness in handling challenges such as network failures, data sharing, and self-adaptability, thereby exhibiting high availability, reliability, and fault tolerance. The proposed framework holds great potential and can be extended in the future to facilitate cross-platform ad-hoc collaborative spaces, including security considerations to avoid malicious intervention and enhance energy efficiency. Furthermore, specialized APIs can be developed for various applications, encouraging application development in an ad-hoc environment.

REFERENCES

[1] M. Weiser, "The computer for the 21 st century," *Sci. Amer.*, vol. 265, no. 3, pp. 94–105, 1991.

[2] C. Mao and D. Chang, "Review of cross-device interaction for facilitating digital transformation in smart home context: A user-centric perspective," *Adv. Eng. Informat.*, vol. 57, Aug. 2023, Art. no. 102087.

[3] R. de Lemos et al., "Software engineering for self-adaptive systems: Research challenges in the provision of assurances," in *Software Engineering for Self-Adaptive Systems III. Assurances*. Cham, Switzerland: Springer, 2017, pp. 3–30.

[4] Z. Bakhshi, G. Rodriguez-Navas, and H. Hansson, "Dependable fog computing: A systematic literature review," in *Proc. 45th Euromicro Conf. Softw. Eng. Adv. Appl. (SEAA)*, Aug. 2019, pp. 395–403.

[5] F. Palumbo, J. Ullberg, A. Štimec, F. Furfari, L. Karlsson, and S. Coradeschi, "Sensor network infrastructure for a home care monitoring system," *Sensors*, vol. 14, no. 3, pp. 3833–3860, Feb. 2014.

[6] A. Khan, A. Munir, Z. Kaleem, F. Ullah, M. Bilal, L. Nkenyereye, S. Shah, L. D. Nguyen, S. M. R. Islam, and K.-S. Kwak, "RDSP: Rapidly deployable wireless ad hoc system for post-disaster management," *Sensors*, vol. 20, no. 2, p. 548, Jan. 2020.

[7] A. Elshrkasi, K. Dimyati, K. A. B. Ahmad, and M. F. B. M. Said, "Enhancement of cellular networks via an improved clustering technique with D2D communication for mission-critical applications," *J. Netw. Comput. Appl.*, vol. 206, Oct. 2022, Art. no. 103482.

[8] U. Demir, C. Tapparello, and W. Heinzelman, "Maintaining connectivity in ad hoc networks through WiFi direct," in *Proc. IEEE 14th Int. Conf. Mobile Ad Hoc Sensor Syst. (MASS)*, Oct. 2017, pp. 308–312.

[9] K. Singh and A. K. Verma, "Flying adhoc networks concept and challenges," in *Advanced Methodologies and Technologies in Network Architecture, Mobile Computing, and Data Analytics*. Hershey, PA, USA: IGI Global, 2019, pp. 903–911.

[10] M. A. Hossain, S. K. Ray, and J. Lota, "SmartDR: A device-to-device communication for post-disaster recovery," *J. Netw. Comput. Appl.*, vol. 171, Dec. 2020, Art. no. 102813.

[11] I. A. Khawaja, A. Abid, M. S. Farooq, A. Shahzada, U. Farooq, and K. Abid, "Ad-hoc collaboration space for distributed cross device mobile application development," *IEEE Access*, vol. 8, pp. 62800–62814, 2020.

[12] S. Robertson, C. Wharton, C. Ashworth, and M. Franzke, "Dual device user interface design: PDAs and interactive television," in *Proc. SIGCHI Conf. Hum. Factors Comput. Syst. Common Ground*, 1996, pp. 79–86.

[13] *Wear Os Smartwatches*. Accessed: Mar. 9, 2021. [Online]. Available: https://wearos.google.com/wearcheck/

[14] P.-Y. Chi and Y. Li, "Weave: Scripting cross-device wearable interaction," in *Proc. 33rd Annu. ACM Conf. Hum. Factors Comput. Syst.*, Apr. 2015, pp. 3923–3932.

[15] P. Hamilton and D. J. Wigdor, "Conductor: Enabling and understanding cross-device interaction," in *Proc. SIGCHI Conf. Hum. Factors Comput. Syst.*, Apr. 2014, pp. 2773–2782.

[16] S. Zhu, S. Xu, S. Setia, and S. Jajodia, "LHAP: A lightweight hop-by-hop authentication protocol for ad-hoc networks," in *Proc. 23rd Int. Conf. Distrib. Comput. Syst. Workshops*, May 2003, pp. 749–755.

[17] *Google Nearby*. Accessed: May, 14, 2022. [Online]. Available: https://developers.google.com/nearby/

[18] *Google Exposure Notifications*. Accessed: May 14, 2022. [Online]. Available: https://www.google.com/covid19/exposurenotifications/

[19] *Bluetooth Technology website*. Accessed: Apr. 12, 2021. [Online]. Available: https://www.bluetooth.com/

[20] *Wi-fi Direct | Wi-fi Alliance*. Accessed: Feb, 14, 2021. [Online]. Available: https://www.wi-fi.org/discover-wi-fi/wi-fi-direct

[21] *Samsung Flow*. Accessed: Jan. 14, 2021. [Online]. Available: https://www.samsung.com/us/support/owners/app/samsung-flow

[22] *Macos—Continuity—Apple*. Accessed: May, 10, 2021. [Online]. Available: https://www.apple.com/macos/continuity/

[23] X. '. Chen, T. Grossman, D. J. Wigdor, and G. Fitzmaurice, "Duet: Exploring joint interactions on a smart phone and a smart watch," in *Proc. SIGCHI Conf. Hum. Factors Comput. Syst.*, Apr. 2014, pp. 159–168.

[24] H. Desruelle, S. Isenberg, A. Botsikas, P. Vergori, and F. Gielen, "Accessible user interface support for multi-device ubiquitous applications: Architectural modifiability considerations," *Universal Access Inf. Soc.*, vol. 15, no. 1, pp. 5–19, Mar. 2016.

[25] S. K. Badam and N. Elmqvist, "PolyChrome: A cross-device framework for collaborative web visualization," in *Proc. 9th ACM Int. Conf. Interact. Tabletops Surf.*, Nov. 2014, pp. 109–118.

[26] C. N. Klokmose, J. R. Eagan, S. Baader, W. Mackay, and M. Beaudouin-Lafon, "Webstrates: Shareable dynamic media," in *Proc. 28th Annu. ACM Symp. User Interface Softw. Technol.*, Nov. 2015, pp. 280–290.

[27] R. Han, V. Perret, and M. Naghshineh, "webSplitter: A unified XML framework for multi-device collaborative web browsing," in *Proc. ACM Conf. Comput. Supported Cooperat. Work*, Dec. 2000, pp. 221–230.

[28] M. Nebeling, A. To, A. Guo, A. A. de Freitas, J. Teevan, S. P. Dow, and J. P. Bigham, "WearWrite: Crowd-assisted writing from smartwatches," in *Proc. CHI Conf. Hum. Factors Comput. Syst.*, May 2016, pp. 3834–3846.

[29] J. Yang and D. Wigdor, "Panelrama: Enabling easy specification of cross-device web applications," in *Proc. SIGCHI Conf. Hum. Factors Comput. Syst.*, Apr. 2014, pp. 2783–2792.

[30] Z. Lu, G. Cao, and T. La Porta, "TeamPhone: Networking SmartPhones for disaster recovery," *IEEE Trans. Mobile Comput.*, vol. 16, no. 12, pp. 3554–3567, Dec. 2017.

[31] R. Alnashwan and H. Mokhtar, "Disaster management system over WiFi direct," in *Proc. 2nd Int. Conf. Comput. Appl. Inf. Secur.*, May 2019, pp. 1–6.

[32] M. El Alami, N. Benamar, M. Younis, and A. A. Shahin, "A framework for enabling Internet access using Wi-Fi peer-to-peer links," *Wireless Pers. Commun.*, vol. 109, no. 1, pp. 521–538, Nov. 2019.

[33] F. Li, X. Wang, Z. Wang, J. Cao, X. Liu, Y. Bi, W. Li, and Y. Wang, "A local communication system over Wi-Fi direct: Implementation and performance evaluation," *IEEE Internet Things J.*, vol. 7, no. 6, pp. 5140–5158, Jun. 2020.

[34] J. Domingue, M. Dzbor, and E. Motta, "Collaborative semantic web browsing with magpie," in *Lecture Notes in Computer Science*. Cham, Switzerland: Springer, 2004, pp. 388–401.

[35] A. W. Esenther, "Instant co-browsing: Lightweight real-time collaborative web browsing," in *Proc. of WWW*, 2002, pp. 1–17.

[36] B. Johanson, S. Ponnekanti, C. Sengupta, and A. Fox, "Multibrowsing: Moving web content across multiple displays," in *Proc. Int. Conf. Ubiquitous Comput.*, 2001, pp. 346–353.

[37] H. Wiltse and J. Nichols, "PlayByPlay: Collaborative web browsing for desktop and mobile devices," in *Proc. SIGCHI Conf. Hum. Factors Comput. Syst.*, Apr. 2009, pp. 1781–1790.

[38] S. K. Badam, A. Mathisen, R. Rädle, C. N. Klokmose, and N. Elmqvist, "Vistrates: A component model for ubiquitous analytics," *IEEE Trans. Vis. Comput. Graphics*, vol. 25, no. 1, pp. 586–596, Jan. 2019.

[39] T. Maekawa, T. Hara, and S. Nishio, "A collaborative web browsing system for multiple mobile users," in *Proc. 4th Annu. IEEE Int. Conf. Pervasive Comput. Commun.*, Jul. 2006, p. 12.

[40] E. Agapie, J. Teevan, and A. Monroy-Hernández, "Crowdsourcing in the field: A case study using local crowds for event reporting," in *Proc. 3rd AAAI Conf. Hum. Comput. Crowdsourcing*, 2015, pp. 1–15.

[41] L. Frosini and F. Patern, "User interface distribution in multi-device and multi-user environments with dynamically migrating engines," in *Proc. ACM SIGCHI Symp. Eng. Interact. Comput. Syst.*, Jun. 2014, pp. 55–64.

[42] S. Park, C. Gebhardt, R. Rädle, A. M. Feit, H. Vrzakova, N. R. Dayama, H.-S. Yeo, C. N. Klokmose, A. Quigley, A. Oulasvirta, and O. Hilliges, "AdaM: Adapting multi-user interfaces for collaborative environments in real-time," in *Proc. CHI Conf. Hum. Factors Comput. Syst.*, Apr. 2018, pp. 1–14.

[43] D. Ashbrook, K. Lyons, and T. Starner, "An investigation into round touchscreen wristwatch interaction," in *Proc. 10th Int. Conf. Hum. Comput. Interact. Mobile Devices Services*, Sep. 2008, pp. 311–314.

[44] J. Kim, J. He, K. Lyons, and T. Starner, "The gesture watch: A wireless contact-free gesture based wrist interface," in *Proc. 11th IEEE Int. Symp. Wearable Comput.*, Oct. 2007, pp. 15–22.

[45] G. Laput, R. Xiao, X. Chen, S. E. Hudson, and C. Harrison, "Skin buttons: Cheap, small, low-powered and clickable fixed-icon laser projectors," in *Proc. 27th Annu. ACM Symp. User Interface Softw. Technol.*, Oct. 2014, pp. 389–394.

[46] S. T. Perrault, E. Lecolinet, J. Eagan, and Y. Guiard, "Watchit: Simple gestures and eyes-free interaction for wristwatches and bracelets," in *Proc. SIGCHI Conf. Hum. Factors Comput. Syst.*, Apr. 2013, pp. 1451–1460.

[47] M. T. Raghunath and C. Narayanaswami, "User interfaces for applications on a wrist watch," *Pers. Ubiquitous Comput.*, vol. 6, no. 1, pp. 17–30, Feb. 2002.

[48] J. M. Zacks, B. Tversky, and G. Iyer, "Perceiving, remembering, and communicating structure in events," *J. Experim. Psychol. Gen.*, vol. 130, no. 1, pp. 29–58, 2001.

[49] Y. Xu, L. Wang, Y. Xu, S. Qiu, M. Xu, and X. Meng, "Cross-device task interaction framework between the smart watch and the smart phone," *Pers. Ubiquitous Comput.*, vol. 25, no. 6, pp. 1039–1049, Dec. 2021.

[50] A. A. Shahin and M. Younis, "Alert dissemination protocol using service discovery in Wi-Fi direct," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Jun. 2015, pp. 7018–7023.

[51] K. M. Rahman, T. Alam, and M. Chowdhury, "Location based early disaster warning and evacuation system on mobile phones using open-streetmap," in *Proc. IEEE Conf. Open Syst.*, Oct. 2012, pp. 1–6.

[52] T. Juhana, R. N. Widyani, and E. Mulyana, "Mobile application for rapid disaster victim assessment," in *Proc. 7th Int. Conf. Telecommun. Syst. Services, Appl. (TSSA)*, Oct. 2012, pp. 324–329.

[53] J. T. B. Fajardo and C. M. Oppus, "A mobile disaster management system using the Android technology," *WSEAS Trans. Commun.*, vol. 9, no. 6, pp. 343–353, 2010.

[54] *Firechat—Wikipedia*. Accessed: Feb. 18, 2021. [Online]. Available: https://en.wikipedia.org/wiki/FireChat

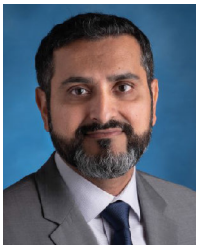
[55] M. Richards, *Software Architecture Patterns*. Sebastopol, CA, USA: O'Reilly Media, 1005.

[56] Z. Á. Mann, "Notions of architecture in fog computing," *Computing*, vol. 103, no. 1, pp. 51–73, Jan. 2021.
[57] G. Gehlen and L. Pham, "Mobile web services for peer-to-peer applications," in *Proc. 2nd IEEE Consum. Commun. Netw. Conf.*, Jan. 2005, pp. 427–433.
[58] C. Casetti, C. F. Chiasserini, L. C. Pelle, C. D. Valle, Y. Duan, and P. Giaccone, "Content-centric routing in Wi-Fi direct multi-group networks," in *Proc. IEEE 16th Int. Symp. A World Wireless, Mobile Multimedia Netw. (WoWMoM)*, Jun. 2015, pp. 1–9.
[59] C. Funai, C. Tapparello, and W. Heinzelman, "Enabling multi-hop ad hoc networks through WiFi direct multi-group networking," in *Proc. Int. Conf. Comput., Netw. Commun. (ICNC)*, Jan. 2017, pp. 491–497.
[60] *Json*. Accessed: Jun. 15, 2021. [Online]. Available: https://www.json.org/json-en.html
[61] S. Roy, S. Chatterjee, A. K. Das, S. Chattopadhyay, N. Kumar, and A. V. Vasilakos, "On the design of provably secure lightweight remote user authentication scheme for mobile cloud computing services," *IEEE Access*, vol. 5, pp. 25808–25825, 2017.
[62] L. Chom Thungon, N. Ahmed, S. Chandra Sahana, and M. I. Hussain, "A lightweight authentication and key exchange mechanism for IPV6 over low-power wireless personal area networks-based Internet of Things," *Trans. Emerg. Telecommun. Technol.*, vol. 32, no. 5, p. e4033, May 2021.

IMRAN ABBAS KHAWAJA was born in Lahore, Pakistan. He received the master's degree in computer science and the M.S. degree in software engineering from the University of Management and Technology, Pakistan. He possesses vast experience in software development. Currently, he is a Project Manager of the Virtual University of Pakistan. He is also enrolled as a Ph.D. candidate with the University of Management and Technology.

KAMRAN ABID (Member, IEEE) received the M.Sc. degree in computer science from Hamdard University, Pakistan, in 2000, the M.S. degree in total quality management from the University of the Punjab, Pakistan, and the Ph.D. degree in electrical engineering from the University of Glasgow, in 2011. Currently, he is a Professor with the Department of Electrical Engineering, University of the Punjab. He has published more than 50 papers in international journals and conferences.

UZMA FAROOQ received the B.S. degree in computer science from the University of the Punjab, in 2005, the M.S. degree in computer science from the National University Computer and Emerging Sciences, Pakistan, in 2007, and the Ph.D. degree from Universiti Teknologi Malaysia, Malaysia, in 2023. Currently, she is an Assistant Professor with the Department of Software Engineering, University of Management and Technology, Pakistan.



ZAKI MALIK received the M.S. and Ph.D. degrees in computer science from Virginia Tech, in 2008. Currently, he is the Department Head/a Professor of business analytics with Texas A&M University—Commerce. His research interests include service computing, business analytics, reliable distributed systems, social networks, and semantic integration systems. He has served as a reviewer for many international conferences and journals.

ADNAN ABID (Senior Member, IEEE) received the B.S. degree from the National University of Computer and Emerging Science, Pakistan, in 2001, the M.S. degree in information technology from the National University of Science and Technology, Pakistan, in 2007, and the Ph.D. degree in computer science from Politecnico di Milano, Italy, in 2012. He spent one year at EPFL, Switzerland, as a Research Assistant. Currently, he is a Professor with the Department of Data Science, University of the Punjab, Pakistan. He has more than 100 publications in different international journals and conferences. His research interests include computer science education, information systems, and ICT4D. He is serving as an editor and an associate editor for different well-reputed journals.