IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received February 10, 2020, accepted March 5, 2020, date of publication March 12, 2020, date of current version April 15, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2980319

Ad-Hoc Collaboration Space for Distributed Cross Device Mobile Application Development

IMRAN ABBAS KHAWAJA[1], ADNAN ABID[1], (Member, IEEE), MUHAMMAD SHOAIB FAROOQ[1], (Member, IEEE), ADNAN SHAHZADA[1], UZMA FAROOQ[2], AND KAMRAN ABID[3], (Member, IEEE)

[1]Department of Computer Science, University of Management and Technology, Lahore 54770, Pakistan
[2]Department of Software Engineering, University of Management and Technology, Lahore 54770, Pakistan
[3]Department of Electrical Engineering, University of the Punjab, Lahore 140413, Pakistan

Corresponding author: Uzma Farooq (uzma.farooq@umt.edu.pk)

ABSTRACT In last few years, a tremendous increase has been observed in the usage of portable electronic devices including smart phones, tablets, laptops, and wearables. These devices are produced by different manufacturers and work on different platforms. People surrounded by these devices need to interact with them during the meeting, presentation, class room and lots of other collaborative activities to share and receive information across the devices. Recent research trends lead towards better utilization of these mobile devices by connecting them together, whereas the interaction among these devices is still device centric and is dependent on expensive fixed software and hardware infrastructure. However, ad-hoc settings, where fixed infrastructure services do not exist, or may suspend the interaction across these devices, require specialized collaborative space. This research presents an architectural framework, named Ad-hoc Collaborative Space (ACS) that provides an abstraction layer by hiding the complexities of ad-hoc environment thus resulting into reduced application development time by providing the easy to use API's. The experimental evaluation based on different operating parameters shows that the proposed framework efficiently manages service registration, service discovery, synchronization, and connectivity between different devices.

INDEX TERMS Cross-device, collaborative applications, distributed interface, WiFi Direct, ad-hoc network, ad-hoc collaboration space.

I. INTRODUCTION

We have seen a tremendous increase in the demand of electronic devices during the last few years. Many different types of devices such as tablets, cell phones, and smart watches are easily available in the market with affordable prices. These devices are manufactured by different vendors and work on a variety of platforms [35]. There are numerous situations where many users get connected with one or more devices to share data and use services. The study conducted by the Facebook on multi device usage has revealed people intense interest to use electronic devices in their daily activities [1]. The result has shown that more than 60% of the adults in the US use at least two devices every day, and one quarter use three devices, while more than 40% start activity on one device and complete it on another. In all above cases work done is bound with the device, and the users are unable to transfer to or share the activities with the other devices. In this regard, researchers and industry have presented a number of solutions to overcome these issues with the help of cloud based services, but most of them are limited to file synchronization across the user's devices like drop box, Microsoft Office 365, Apple iCloud, and few allow collaboration activities like google collaborative editing. In all mentioned cases, the solutions presented are tightly bound with the fixed infrastructure.

Weiser's vision of ubiquitous computing that envisages seamlessly integration of multiple devices of different sizes and capabilities to share program and data, requires ease in cross-device communication [2]. One such manifestation of cross device collaborative spaces, whereby the functionality and interactions are distributed across multiple devices, and the users of these devices actively participate in the

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

collaborative activities. Similarly, Sørensen has categorized cross-device interaction into four categories on the basis of device usage [4] namely: communality, collaboration, continuity, and complementary. The first theme communality refers to situation where many users are interacting with the device sequentially for example Koday's facebook photo-printing machine. The second theme collaboration covers the scenario that involves simultaneous interaction by many users, where interface components are distributed among the users; for instance, division in a multi-player game. The third theme continuity enables people to use several devices and re-access content on different devices; such continuous interactions can be facilitated by keeping data consistent across devices. The last one is complementary, where interactions with one artifact add to the interaction with another artifact, and these jointly make up a larger whole, for instance, Adobe Nav app allows iPad as a two-way communication device with Photoshop. Whereby, one can browse, select, and open Photoshop documents and activate Photoshop tools through a wireless network.

The ubiquitous availability of electronic gadgets i.e. smart phones and tablets leads towards the collaborative ad-hoc portable setups spanning across multiple devices [37]. Researchers have been investigating this problem of interaction between multiple devices while offering collaboration among the users [21], [22]. This research aims to propose a framework which is particularly focused on providing a layer of abstraction to develop multi device interactions for collaborative activities. To develop a cross-device collaborative application that spans across multiple devices requires monitoring the distribution changes and synchronize it on all devices in order to ensure the collaboration space state. Moreover, programming cross device with these tasks at low level is technically challenging as it involves input and output on multiple devices and demands serious effort from the programmer for developing a cross device applications. Moreover, testing the cross-device interaction across multiple devices is the most challenging task due to the variety of electronic devices available in the commercial market.

The proposed framework facilitates the development of distributed cross-device mobile applications for ad-hoc network. Mainly, it hides the complexities involved in service registration, service discovery, connection management, synchronization, and fault tolerance, by providing abstraction layer. This abstraction enables the application developer to simply focus on the development of collaborative cross-device applications while leaving all the aforementioned complexities of distribution to the framework. Thus, the novelty of proposed approach is that it simplifies the development of distributed cross device collaborative applications, as the programmer does not need to implement low level details anymore; instead he invokes few standard functions exposed in the form of API by the proposed framework.

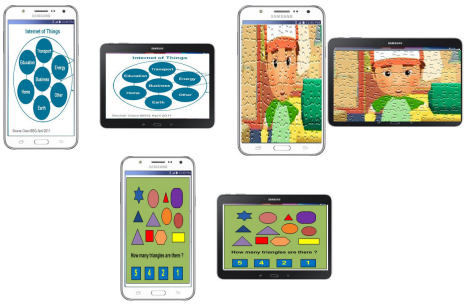

FIGURE 1. Collaborative e-learning applications, contents synchronization across all connected devices.

A. MOTIVATION AND CHALLENGES

The motivation for this research can be highlighted with the help of following example scenarios.

1) LECTURE PRESENTATION

Suppose a teacher intends to conduct a class during a field visit outside the class room, but at same time wants to discuss the points with the help of presentation. In that scenario, it is painful to arrange computer/laptop connected with the large screen to show the presentation. But what if the teacher is able to present without any fixed infrastructure and all the students are viewing the presentation on their personal portable devices. It will enable the teacher to present any time anywhere without worrying about the infrastructure limitation.

2) JIGSAW PUZZLE

In a relatively more complicated scenario, consider some kids are solving a Jigsaw puzzle but want to do it in a collaborative manner. Let us suppose an application allows them to load the puzzle on their tablets and enable them to solve it in a collaboratively. Anyone move by one kid will immediately be synchronized on all other kids' tablets. This type of application helps the students to learn from each other, and promote the collaboration culture.

The development of such collaborative application poses challenges that must be overcome for the reliability of these collaborative applications. The proposed framework ad-hoc collaboration space aims to address the following objectives and challenges to easily implement corss-device collaborative services in ad-hoc settings.

3) SERVICE MANAGEMENT

In ad-hoc network setting, service management refers to the management of service registration and discovery for the devices. It not only includes the detection of nearby devices i.e. smart phone and tablets etc. but also includes service discovery, and management of device connectivity requests.

4) COMMUNICATION CHANNEL

The communication channel is the layer that is solely responsible for data transmission across the host/client devices using

the sockets. The layer must be capable enough to manage the connection threads with multiple devices by hiding the complex communication details.

5) EFFICIENCY

Efficiency of distributed application is measured on the basis of time taken by the device to exchange the information with the peer devices. It is desirable to keep the data updated on all the involved devices almost in real time.

6) MULTI-DEVICE APPLICATION TESTING

Multi-device testing involves execution of test scenarios simultaneously on a number of devices, i.e. client/server devices at the same time to observe the behavior of distributed application. This challenge carries the application developers away from the development of complicated distributed applications.

The rest of the article has been structured in the following manner. Section II provides the theoretical background of cross device interactions. This section also discusses the cross device frameworks presented by the researchers for various types of interaction styles. The details of the proposed framework have been discussed in Section III, where different components of the framework and their interactions have been discussed in detail. Section IV signifies the value of the proposed framework by highlighting the difference between developing an application for ad-hoc connectivity with and without the proposed framework. The experimental evaluation of the proposed framework has been provided in Section V. The conclusion and future directions have been discussed in Section Vi.

II. RELATED WORK

The journey of cross-device interaction started in 1996, when Robertson developed a system to control television through a personal digital assistant (PDA), but the scope was limited to one directional communication i.e. from PDA to the television [3]. After that we have seen a potential growth in multi-devices interactions. Existing commercial devices have also started to support cross-device interactions such as controlling the content on the phone through a smart watch as remote control [15]. A number of research articles have been presented by researchers in the recent years that discuss different interaction styles [7], [11].

In this section, we have categorized the frameworks in two categories ad-hoc network based frameworks and fixed infrastructure based frameworks. An ad-hoc network is a form of network technology that allows communication on temporary/ad-hoc basis [36]. A mobile device is one, which is equipped with the ad-hoc networking technology (Bluetooth, WiFi). Bluetooth is a wireless technology that uses for exchanging data over a short distance when speed is not an issue, such as printer, headsets, telephone etc [34]. On the other hand, Wi-Fi Direct supports device to device transfer speed of up to 250 Mbps with ranges up to 200 meters [23]. To evaluate the strength and weakness of different frameworks, we have compared them on the basis of interaction style, network, and collaboration support as shown in Table 1.

TABLE 1. Classification of frameworks on the basis of interaction style, network and collaboration support.

Framework	Interaction Style	Adhoc Network	Collaboration
Samsung Flow	Task Continuity	✓	✗
Apple Continuity	Task Continuity	✓	✗
Conductor	Task Distribution	✗	✗
Duet	Watch/Mobile Joint Interaction	✓	✗
Polychrome	Collaborative Visualization	✗	✓
Websplitter	Web Page Portion Distribution	✗	✓
Weave	Mobile/Wearable Interaction	✓	✗
Wearwrite	Mobile/Wearable Crowd Writing	✗	✗
Panelrama	Collaboration	✗	✓
ACS	Collaboration	✓	✓

Samsung flow is a framework that was launched in 2014 and allows the Samsung device users to seamlessly switch between devices like smartphones, tablets, and smart watches while interacting with the apps or contents [6]. It works with Bluetooth or WI-FI and required that devices should be physically close to each other or on the same WIFI network so as to tolerate the switching of devices. The major limitation of Samsung flow is that it only works with Samsung hardware and does not allow collaboration activities. Similarly, the iOS cross platform (Mac, iPhone, and iPad) continuity feature allows the user to move work flow between these devices [14].

Wearable gadgets have flourished in the market and gained a lot of popularity in the recent years, but due to limited input and output capabilities, it is useful to utilize them by pairing with the other devices. To overcome this limitation, Duet focused on the joint interaction of watch and mobile phone owned by a single user for the completion of a particular task by recognizing the touch and motion gestures across the devices, and by transmitting this information via Bluetooth [33]. Example of this interaction is to receive an email notification on the smart watch by pairing it with the mobile phone, and opening the email on mobile phone by tapping on the smart watch. A number of approaches have been presented by the researchers to increase the input and output capabilities of the smart watches by pairing them with other devices [27]–[32], [38]. A contribution in the field of cross-device interaction to reduce the developers' effort to implement a cross-device interaction for mobile and wearables devices by Pei-Yu (Peggy) Chi and Yang Li presented

the weave a web-based framework, and is inspired by popular JQuery (JavaScript) libraries. This framework aims to provide a set of high level abstraction based APIs for distributing sensing events and User Interface (UI) output across mobile and wearable devices [7].

Hamilton, P. and Wigdor designed the conductor, a cross-device interaction system that supports single user multiple device interaction styles by allowing the user of the system to split their activities across multiple devices for flexibility and usability [11]. The system's backbone is a dedicated server that manages the connected devices, their session information, and states through web sockets.

Another form of cross-device collaboration is collaborative web browsing, where browsers on multiple devices are connected and enable cross-device interaction between the devices [16]–[18], [26]. Polychrome is a web based cross-device interaction framework that supports collaborative web browsing/visualization by allowing the browsers of different devices to connect with each other using the PeerJS for P2P communication [9]. Similarly VISTRATES, a web-based platform provides a collaborative environment for data analysis and visualization [40]. The Websplitter Maekawa targets multi-device and multi-user Web browsing by providing a framework for dividing a single web page into several portions, and by serving each portion of the page to a different device instead of mirroring [24], [25].

By considering the input and output limitation of smart watches in performing the complex daily tasks, Wearwrite a watch centric collaborative cross-device interaction system was presented by the researchers to interact with the documents [10]. The system allows the watch user to integrate with the other devices and manages the crowd writing process i.e. research paper, blogs etc. from the watch. An example of crowd writing is provided by Agapie in the context of writing by using a combination of local and remote workers to produce a news event [20]. Wearwrite team has developed one android mobile app and one android wear app.

Luca Frosini in 2014 presented a framework that supports cross-device user interface distribution and collaboration among multiple devices [8]. The framework consists of two blocks client library and runtime engine, whereby the client library is responsible for sending distribution changes and receiving updates of changes to apply on the client device UI, while the runtime engine is responsible for managing all the devices and their request of distribution changes. Panelrama also supports similar kind of UI distribution but for web UI distribution across multiple devices [19]. It dynamically allocates each portion of the UI to a target display based on its screen real estate and input modality. Similarly, AdaM is a web based platform for automatic UI distribution across multiple devices, and users in a collaborative setting interact with one another on the basis of device capabilities, user roles, preferences, and access rights [39].

Despite number of solutions presented in the past, there is still a need of comprehensive framework that provides seamless collaboration across the devices in a situation, where fixed infrastructure services are not available.

III. PROPOSED FRAMEWORK: AD-HOC COLLABORATION SPACE(ACS)

The proposed framework aims to provide a flexible mechanism of collaboration across multiple devices by hiding complexities of distributed systems. The framework is built on Android Wi-Fi (peer-to-peer or P2P) APIs and Java sockets that allows application to connect to any nearby devices without using any dedicated access point or infrastructure in order to share data [12]. It also provides an abstraction layer to hide the complexity of Wi-Fi (peer-to-peer or P2P) APIs and Java sockets.

In this article, term device is used to refer to the electronic devices that include smart phones, tablets, or smart watches that are equipped with wireless technology, and have an operating system that can run different application software. The proposed framework considered Wi-Fi Direct as a communication protocol for the exchange of data over short distances due to high speed and range. Devices are categorized into two different types;the provider devices invoke the service registration process to advertise their services to nearby devices, while the consumer devices discover the available services by invoking the connectivity request to join the collaboration space. Finally, a communication channel provides two-way communication link between two programs running on different devices to send and receive the activities performed by the device users so as to synchronize the session state.

The workflow of collaborative space is initiated by a service broadcast message by the provider device which wants to collaborate with the other devices for program and data sharing. The broadcast message includes information about the service and the connectivity parameters. After broadcasting, the service makes itself available to all nearby WiFi direct enabled devices, which can discover the services and can request for the connectivity with the service hosting device. The provider device accepts the connectivity request, and devices start collaboration with each other without any fixed infrastructure.

It is pertinent to mention that WiFi (peer-to-peer or P2P) APIs allow the application to connect to nearby android based devices. While for apple iOS based devices, multi-peer connectivity framework supports the discovery of services provided by nearby devices and support communication through the message-based data. Unfortunately, peer to peer connectivity frameworks offered by android and iOS for WiFi Direct do not allow connectivity across the Android and iOS based devices. Therefore, the scope of this work is limited to the connectivity of android based devices in an ad-hoc environment.

The ad-hoc collaboration space consists of three major components *ACS Manager*, *Connection Manager* and *Events Observer*. Figure 2 provides a high level view of the ad-hoc collaboration space and its major components.

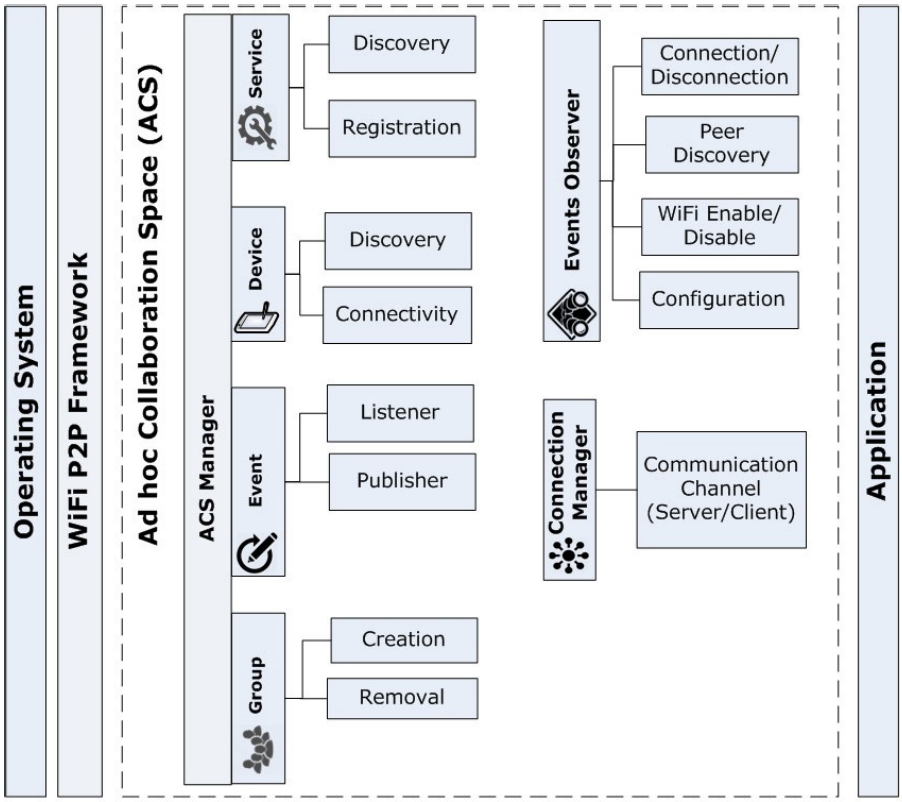

FIGURE 2. High level architecture.

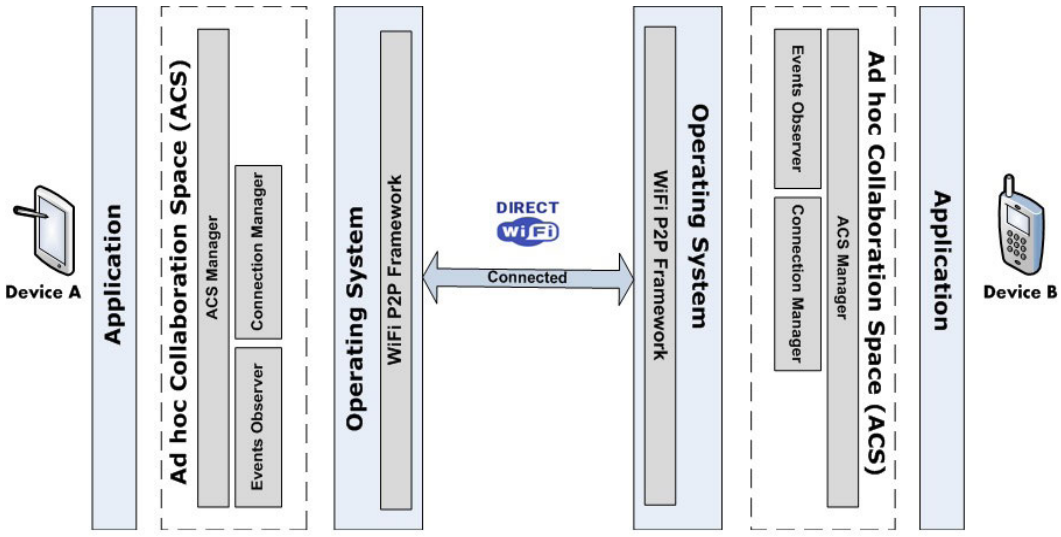

FIGURE 3. Cross-Device interaction using the proposed ACS framework.

To develop a better understanding of the framework usability, the cross-device interaction across devices is visualized using the proposed ACS framework in Figure 3. It clearly shows that the ad-hoc collaboration layer provides abstraction by hiding the complexities of distributed cross device environment required during the service registration, service discovery, connectivity, message exchange, and synchronization activities.

A. ACS MANAGER

ACS Manager is the main controller component of the framework; all other components are controlled and monitored by the *ACS Manager*. It manages the service registration and discovery processes on the Android WiFi P2P framework using the service details provided by the application. All other components execute as per the instructions of the *ACS Manager*, and continuously report to it about the state of

Algorithm 1 Service Registration

```
s: Service Identity, p: Listening Port
    ▷ Feedback Status notifies about the execution status
Call RegisterService(s, p, FeedBackStatus)
if FeedBackStatus = true then
    Output: Service Successfully Registered
    ▷ Create group to allocate group owner role to service hosting device
    Call createGroup(FeedBackStatus)
    if FeedBackStatus = true then
        Output: Group Created Successfully
    else
        Output: Group Creation Failed
    end if
else
    Output: Service Failed To Registered
end if
```

Algorithm 2 Service Discovery

```
s:Service Name, mac:MacAddress, p:Port No
    ▷ FeedBackStatus notifies about the execution status
Call DiscoverService(s, FeedBackStatus)
if FeedBackStatus = true then
    Output: Service Found
    m = macaddress        ▷ Retrieve Device MacAddress
    p = portno            ▷ Retrieve port no
    Call connectDevice(mac, FeedBackStatus)    ▷ For P2P Connection
    if FeedBackStatus = true then
        Output: Connected to Device
    else
        Output: Connectivity request rejected
    end if
else
    Output: Service Discovery Failed
end if
```

collaboration space. All collaboration activities performed by the device are notified to all devices through the connection manager on the instruction of *ACS Manager*.

1) REGISTRATION

The service registration is a process which allows the provider devices to advertise their services for the nearby consumer devices. For Instance, if any device wants to provide a collaboration service to the nearby devices directly without being connected to local network or hotspot, it must register the service on WiFi P2P framework for service discovery. For instance, in case of collaborative learning application, the teacher can register the presentation service on the WiFi P2P framework for service discovery so as to make it available for the nearby student devices. Once the service is registered, the framework automatically responds to the peer (student) devices for service discovery requests. The service registration API requires the unique identity of the service and listening port number to register the service on the local device, and provides the feedback status i.e. success or failure to the *ACS Manager*. The service registration cycle is explained in Algorithm 1.

2) DISCOVERY

Service Discovery is the detection of nearby devices and the services offered by these devices. Any device that wants to subscribe with a particular service, invokes the service discovery process in order to locate the service and connection details. The discovery API requires unique identity and connection details of a service to initiate the connection request. As an example, in case of collaborative learning application, the student devices can invoke the discovery API to locate the presentation service broadcasted by the teacher's device by providing the name of the service. The discovery process looks for all available services broadcasted by the nearby device and provides the feedback status of the discovery API i.e. success or failure to the *ACS Manager* along with the service connection details. The service discovery cycle is explained in Algorithm 2. Furthermore, the detail of data communication among the devices is explained in the connection manager section.

B. EVENTS OBSERVER

An *Event Observer* is a component that responds to the broadcasts announced by the system. Due to the ad-hoc nature of the collaboration space, any device can join or leave the network at any time. There is a need to detect this connection/disconnection and act accordingly so as to manage the space in a robust manner. Moreover, on successful device connectivity, it requests the WiFi P2P for the group information, which automatically sends the group information to the ACS manager. *ACS Manager* uses this information to determine the group owner and client device to create the data communication channel across the devices. In case of collaborative learning application, the teacher remains up to date about the number of students connected with the presentation, and their participation activities which encourages them to remain active in their participation.

In order to listen to these broadcast events, the *ACS Manager* must register the broadcast intents as shown in the Table 2.

C. CONNECTION MANAGER

This is the most complex component of the system that is solely responsible for data transmission across the host and client devices as shown in Figure 4. The *Connection Manager* uses the standard Java sockets along with multiple threads to manage a number of clients simultaneously while maintaining high performance.

1) SERVER REGISTRATION

Connection Manager uses Java sockets that provide two way data communication across the devices. To start the client

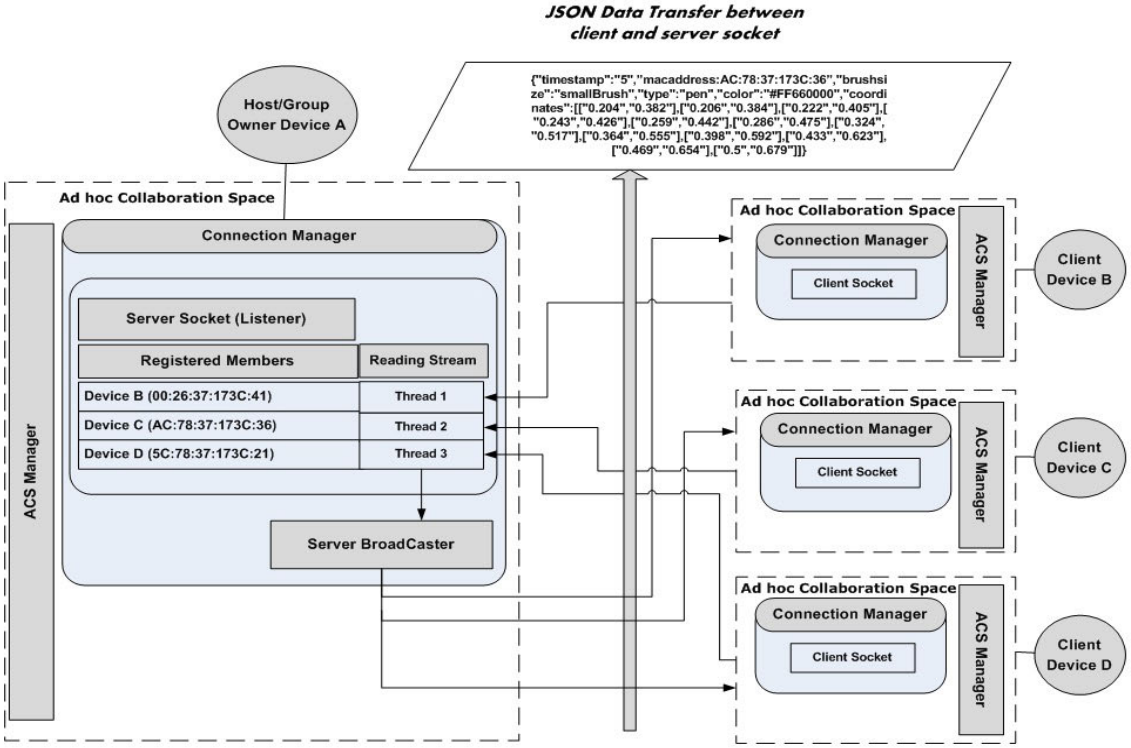

FIGURE 4. ACS connection manager.

TABLE 2. Broadcast intents.

Intents	Details
WiFi State	Indicates whether Wi-Fi P2P is enabled/disabled.
Peer List	Indicates that the available peer list has changed new device is found by the android system within the WiFi Direct range.
Connection/ Disconnection	Indicates the state of Wi-Fi P2P connectivity has changed (Connection/Disconnection of device).
Device Configuration	Indicates the device's configuration details have changed.

devices registration process on the host device, it creates the Java server socket with the port number provided by the *ACS Manager*. After the acceptance of client device request, it stores the client information in the registered clients list along with the mac address as identification in order to maintain the list of registered client devices. The registered clients list is subsequently used for message broadcasting to all registered devices.

2) CLIENT REGISTRATION

After the establishment of WiFi P2P link with the group owner device, ACS manager invokes the connection manager by providing the host device IP Address, Listening port number and device's mac address for establishing the connection with the host/group owner device. In case of collaborative learning application student devices continuously read the updates coming from the teacher device and synchronize their state with the teacher's device.

3) MESSAGE BROADCASTING (CLIENT/SERVER)

In case of any update at the client device or host device, *ACS Manager* invokes the connection manager event broadcasting API with event/message data for broad-casting the event across all the clients for session state synchronization. In case of collaborative learning application the activities performed by the devices users can be synchronized across all devices without any delay, so as to keep the session state consistent on all devices.

4) AVAILABLE PEERS

During the collaboration activity, devices can leave the network any time. In that scenario, it is necessary to detect any device that leaves the group and remove it from the collaborative space so as to clean the unmanaged resources.

IV. DISCUSSION

It would be interesting to understand how the proposed ACS framework facilitates the application developers in building such cross device collaborative applications. To this end, this section presents the comparison of development efforts

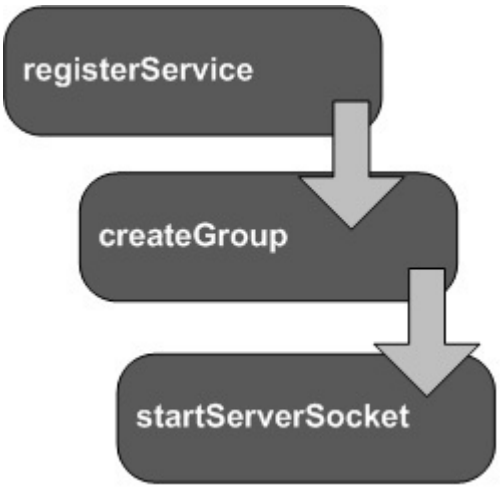

FIGURE 5. Server device.

required with and without ACS API's. It also discusses how ACS makes application development faster by providing an abstraction layer on the complex integration details. As a matter of fact, from the application developers' prospective, the overall implementation requires very small and easy to implement code using the API's provided by the ACS framework.

A. APPLICATION DEVELOPMENT WITH ACS API's

API stands for application programming interface that allows two different applications to talk to each other in a seamless manner by hiding the complex integration details. This section explains the usage of ACS API's for the development of cross-device mobile applications. We will also discuss the technical challenges that the proposed API addresses and facilitates the application developer to develop a distributed cross-device mobile application.

1) REGISTRATION

The process starts with registration, as shown in Listing 1, we invoke the registerService to advertise our service to the nearby devices to connect with it using the ad-hoc WiFi Direct based network. After the successful registration of service, we call the createGroup API to forcibly allocate the role of Group Owner to the service hosting device. The *ACS Manager* also invokes the *Connection Manager* startServerSocket API to start the server socket on the device for accepting the connection request from the client/group member devices for data communication across the devices. Figure 5 demonstrates the execution flow of service registration process. As shown in the Listing 1, we have supplied the service name, port number, and call back method which inform the user about the execution status of the API.

2) DISCOVERY

The device that wants to connect to the desired service to perform collaboration activity will invoke the discoverService API to fetch the list of available services on the WiFi

```
ACSManager.registerService("SynchPaint",4545,
    new ACSCallback() {
@Override
public void response(boolean cmdStatus, int
    failureReason) {
if (cmdStatus) {
Toast.makeText(MainActivity.this, "Service
    Registered Successfully",
    Toast.LENGTH_SHORT).show();
ACSManager.createGroup(new ACSCallback() {
@Override
public void response(boolean cmdStatus, int
    failureReason) {
if (cmdStatus){
Toast.makeText(MainActivity.this, "Group
    Created Successfully",
    Toast.LENGTH_SHORT).show();
}
}
});
} else {
Toast.makeText(MainActivity.this, "Service
    Registration Failed",
    Toast.LENGTH_SHORT).show();
}
}
});
```

Listing 1, Registration

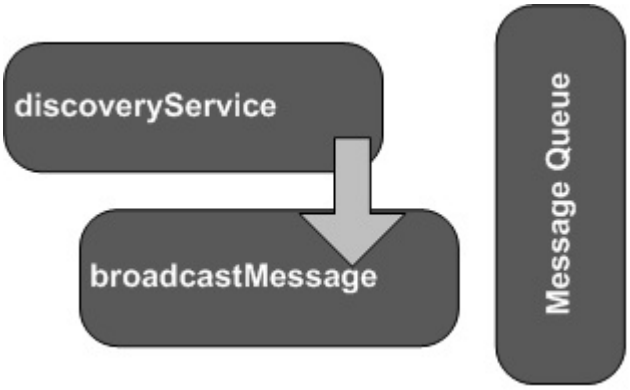

FIGURE 6. Client device.

network and connectToService API to connect with particular service as shown in the Listing 2.1 and 2.2. After the successful discovery of service, the *ACS Manager* invokes the *Connection Manager* connectToServer API by providing the service connectivity parameters for socket connection establishment with the group owner device for data communication across the devices. It is pertinent to highlight that these socket connection methods are hidden from the application developer and are managed by the *ACS Manager* to facilitate the application developer by writing just few lines of method invocation while ignoring the complexities of the underlying connections. Figure 6 demonstrates the execution flow of service discovery process.

3) MESSAGE BROADCASTING

In case of any update at the client device or host device, application developer will invoke the broadcastMessage API

```
ACSManager.discoverAvailableServices( new
    ACSCallback() {
@Override
public void response(List<String>
    availableServices, boolean cmdStatus,
    int failureReason) {
if (cmdStatus) {
Toast.makeText(MainActivity.this, "List of
    avaialble services.",
    Toast.LENGTH_SHORT).show();
}
else {
Toast.makeText(MainActivity.this, "Service
    discovery failed",
    Toast.LENGTH_SHORT).show();

}
}
});
```

Listing 2.1, Discovery

```
ACSManager.connectToService("SynchPaint",
    new ACSCallback() {
@Override
public void response(boolean cmdStatus, int
    failureReason) {
if (cmdStatus) {
Toast.makeText(MainActivity.this,
    "Connected with the service",
    Toast.LENGTH_SHORT).show();
}
else {
Toast.makeText(MainActivity.this, "Service
    connectivity failed",
    Toast.LENGTH_SHORT).show();

}
}
});
```

Listing 2.2, Discovery

to broadcast an event across the connected devices as shown in Listing 3. The application captures the touch event performed on the device, converts it into JSON based string as shown in Figure 7 and calls the broadcasting function to synchronize the data across the all connected devices. ACS Manager also incorporates the unique timestamp and device mac address along with the JSON based data packet to maintain the identity of event. Though the ad-hoc device connectivity makes it complex to manage the events' state across the devices, yet *ACS Manager* manages the state in an efficient manner, and allows the developer to remain focused on the application development, instead of managing device connections and message delivery in ad-hoc settings.

Many teachers like to engage their students in a group activity during the class by forming small groups. They believe that such type of activities provides a great opportunity to the students to learn from each other and promote the collaboration culture in the society to enhance their team work skills.

```
ACSManager.broadcastMessage(acsData,new
    ACSCallback() {
@Override
public void response(boolean cmdStatus, int
    failureReason) {
if (cmdStatus) {
Toast.makeText(MainActivity.this, "Message
    sent ", Toast.LENGTH_SHORT).show();
}
else {
Toast.makeText(MainActivity.this, "Message
    sent failed.",
    Toast.LENGTH_SHORT).show();

}
}
});
```

Listing 3, Message Broadcasting

{"timestamp":"5","macaddress:AC:78:37:173C:36",
"brushsize":"smallBrush","type":"pen","color":"#FF
660000","coordinates":[["0.204","0.382"],["0.206",
"0.384"],["0.222","0.405"],["0.243","0.426"],["0.259
","0.442"],["0.286","0.475"],["0.324","0.517"],["0.3
64","0.555"],["0.398","0.592"],["0.433","0.623"],["0
.469","0.654"],["0.5","0.679"]]}

FIGURE 7. Event data (paint application).

{"timestamp":"12","macaddress:AC:78:37:173C:36",
"TaskNo":"14","GroupName":"51214","StudentID":"
15006114001","text":"Reflects the historical rate of
learning of operation can be used for resource
leveling on the basis of efficient working."}

FIGURE 8. Event data (group case study).

The sample shown in the Figure 8 presents a small group activity in the class room where teacher creates the students' groups to answer the case study questions by allocating the questions to different groups. The students are instructed to answer the question through their mobile device by posting the answer into the relevant/allocated section to participate in the learning activity. Teachers are able to monitor the overall activity performed by all the groups via mobile application interface. Moreover, students are also able to view the answers submitted by other groups. The use of modern technology in the education sectors will encourage the collaborative learning environment.

4) RECEIVE MESSAGE

In order to receive the event data, client devices create a handler that allows retrieving data from the threaded

```
handler=new Handler(){
@Override
public void handleMessage(Message msg) {
drawView.Execute(msg.getData().getString("msg"));
//Application process the data
}
};
```

Listing 4, Receive Message

```
ACSManager.disconnectWithService(new
    ACSCallback() {
@Override
public void response(boolean cmdStatus, int
    failureReason) {
if (cmdStatus) {
Toast.makeText(MainActivity.this,
    "Disconnected with service",
    Toast.LENGTH_SHORT).show();
} else {
Toast.makeText(MainActivity.this,
    "Disconnection Failed",
    Toast.LENGTH_SHORT).show();
}
}
});
```

Listing 5, Disconnect Service

Message Queue. ACS Manger posts the event data directly into the Message Queue of the client/host device. Application developer accessed the Message Queue and utilizes that data for display/execution or similar purposes according to the need of the application. Listing 4 clearly highlights that ACS facilitates the application developer by providing a high level of abstraction on complex communication details.

5) DISCONNECT SERVICE

Client device can leave the collaboration session any time by calling the disconnectWithService API as shown in the Listing 5. In this particular scenario, *Event Observer* plays the key role by informing the *ACS Manager* about the occurrence of events to manage the collaboration space.

B. APPLICATION DEVELOPMENT WITHOUT ACS API's

As discussed in the previous section, the proposed framework enables the application developer to focus on application development without worrying about the underlying ad-hoc network formation and communication details. But without ACS creating an application that spans across multiple devices requires the following steps.

1) AD-HOC NETWORK

Select ad-hoc network technology that allows communication on temporary basis. In the absence of ACS framework, the robustness and synchronization required for managing the communication between different devices in ad-hoc settings will be responsibility of the application developer.

2) SERVICE REGISTRATION/DISCOVERY

Implementation of service registration and discovery routines that allows an application to advertise their services and find the available services hosted by other devices will also be responsibility of the application developer.

3) TWO WAY COMMUNICATION

There is a stringent requirement of two way communication by the device that acts as an access point, also referred to as group owner (GO). Again, in the absence of ACS framework the application developer has to write code to establish a socket connection between server and client devices to allow two way data communication.

4) DISTRIBUTION CHANGES

In the absence of ACS framework the application developer has to write code for maintaining the distribution changes on server device, and ensure that these changes are updated on all the connected client devices.

5) CONNECTION/DISCONNECTION

Due to the ad-hoc nature of the application, any device can join or left the network at any time. Thus, there would be a need implement a routine to detect this connection/disconnection and act accordingly in order to manage the application. This requires the implementation of a challenging task of synchronization of actions performed during the absence of a device.

6) MULTI-DEVICE APPLICATION TESTING

Application developer needs multiple devices in order to test the behaviour of application on different devices. This poses a challenge to the developers of such distributed application.

In short, writing code for cross device application with the aforementioned low level tasks is technically challenging as it involves input and output on multiple devices in an ad-hoc settings. These tough requirements distract the application developers and they shy away from developing the cross device applications due to high complexity level that is difficult to develop and maintain. However, the provision of ACS framework makes the life of a programmer very easy, whereby one can work out all the complex details just by calling appropriate functions exposed by the rich and robust API provided by the ACS framework.

V. EVALUATION

It is pertinent to evaluate the ability and effectiveness of the proposed framework. The main purpose of this evaluation is to evaluate the effectiveness of the abstraction provided by the framework. Different assessment methods and scenarios have been used for the validation of the proposed ACS framework.

A. EXPERIMENTAL SETUP

In order to evaluate the effectiveness of the proposed framework different types of interaction scenarios/styles i.e collaborative visualization, portion distribution, joint interaction

TABLE 3. Evaluation parameters (defaults in bold text).

Service Discovery & Group Formation	Devices	**Samsung Grand Prime**	OS	**Lollipop 5.0.2**
		Samsung Galaxy A5		Nougat 7.0
Synchronization	No of Devices	3 - 5 - 7		
Impact of distance (Synchronization)	Distance	10, **20**, 30, 40, 50, 60, 70, 80, 90, 100 (meters)		
State synchronization on late joined devices	Existing Operations	**200**, 300, 400		
Client device state synchronization after rejoining	Operation performed in inactive period	**100**, 200, 300		
Memory Overhead	Operation performed	**200**, 400, 600		
Energy Consumption	Operation performed	**100**, 200, 300		

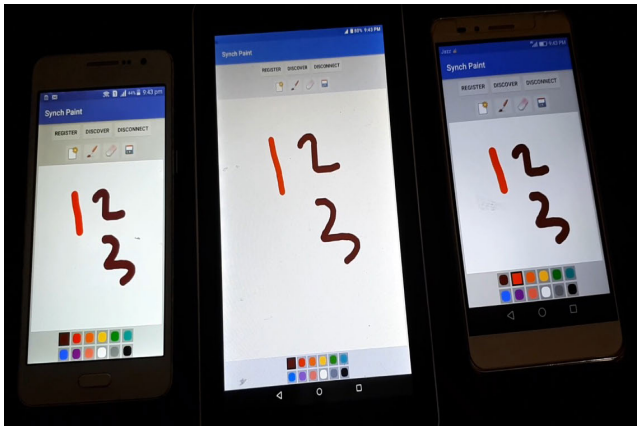

FIGURE 9. Paint application developed using the ACS framework.

and collaboration have been critically reviewed to develop a suitable application that covers these interaction scenarios.

By considering the above mentioned interaction styles, we have designed and develop the paint mobile application as shown in Figure 9 using the proposed framework that allows the users to draw in collaborative manner using their devices with the help of different drawing tools, colour palettes and save the image file on the mobile phone. The size of paint application developed using the ACS Framework is 3.93 MB, which is quite suitable for distributed mobile application development due to the storage limitation of mobile devices and proves that framework does not have any memory overhead.

1) EVALUATION PARAMETERS

Experiments have been designed based on following evaluation parameters as shown in Table 3.

1. Service Discovery & Group Formation: Time taken by the device to discover the broadcasted service and form the WiFi Direct based ad-hoc network.

2. Synchronization: Events synchronization across the multiple devices connected with collaboration space.

3. Impact of Distance: To measure the impact of distance on the data synchronization time across the devices.

4. State Synchronization on Late Joined Devices: Time to update the session state on the device that joins the collaboration space late.

5. State Synchronization after Device Rejoin: Time to synchronize the session state on the client device that rejoins after disconnection.

6. Memory Overhead: Estimate the device memory used during the collaboration activities.

B. EXPERIMENTS

Different experiments were conducted by considering the evaluation parameters mentioned above to validate the framework effectiveness. The experiments details along with their results are briefly discussed in this section. In order to avoid any bias, we have performed number of experiments and report the average results.

1) SERVICE DISCOVERY AND GROUP FORMATION

WiFi Direct group can comprise of devices with different android version. To evaluate the ability of framework to operate on diverse range of android based devices for data exchange, we have conducted the experiment on android based devices namely Samsung Grand Prime and Samsung Galaxy A5. In this regard we have evaluated the service discovery across the devices and conducted 10 experiments and report the average discovery time in Table 4. According to the results, the service discovery worked well and found the requested service within few milli second time that is quite acceptable and satisfies the speedy ad-hoc network formation requirement. The service discovery and connectivity invitation processes are shown in the Figure 10 for better understanding of the device connectivity process.

2) EFFICIENCY

Efficiency of distributed application is measured on the basis of time taken by the device to exchange the information with

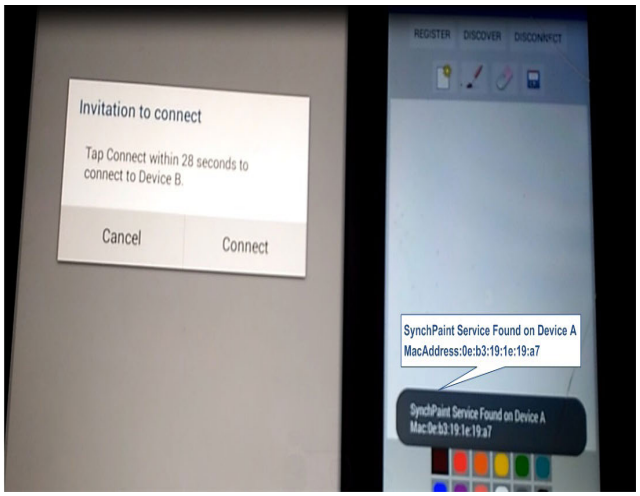

FIGURE 10. Device requests to connect with the service hosting by other device.

TABLE 4. Service discovery time across the group of devices.

	Samsung Grand Prime Lollipop 5.0.2	Samsung Galaxy A5 Nougat 7.0
Samsung Grand Prime Lollipop 5.0.2	**919 ms**	1088 ms
Samsung Galaxy A5 Nougat 7.0	1088 ms	**520 ms**

the peer devices. It is an important factor in case of distributed applications and requires that the data should be synchronized immediately on all the devices without any delay.

In an experiment as shown in Table 5, we have distributed 3 Samsung devices among the users and asked them to perform 50 drawing operations simultaneously on their devices and captured the operation broadcast and receive time. Furthermore, we have increased the number of devices in order to test the efficiency of collaboration space. The result shows an average time of 55 ms with 3 Devices, 57 ms with 5 Devices and 61 ms with 7 Devices taken by an operation to synchronize on all the devices.

3) IMPACT OF DISTANCE

We have also evaluated the operation synchronization time with different distance ranges and plotted in the form of graph as shown in Figure 11. The results clearly show the impact of distance on the operation synchronization time across the devices. As with the increase in distance the time significantly increased but still it is acceptable in case of collaborative learning applications discussed in the previous section.

4) STATE SYNCHRONIZATION

In an experiment as shown in Table 6, we have measured the data synchronization time and accuracy by joining the new device to an existing WiFi Direct group of devices doing the collaboration activities. The results shows that the state is synchronized in 4 seconds with 200 operations, 5 seconds with 300 operations and 7 seconds with 400 operations, respectively.

TABLE 5. Time taken by an operation to synchronize across the devices during simultaneous operations.

Manufacturer	Total Devices	Total Operations	Average Time
Samsung Grand Prime (Lollipop 5.0.2)	3	150	55 ms
Samsung Grand Prime (Lollipop 5.0.2)	5	250	57 ms
Samsung Grand Prime (Lollipop 5.0.2)	7	350	61 ms

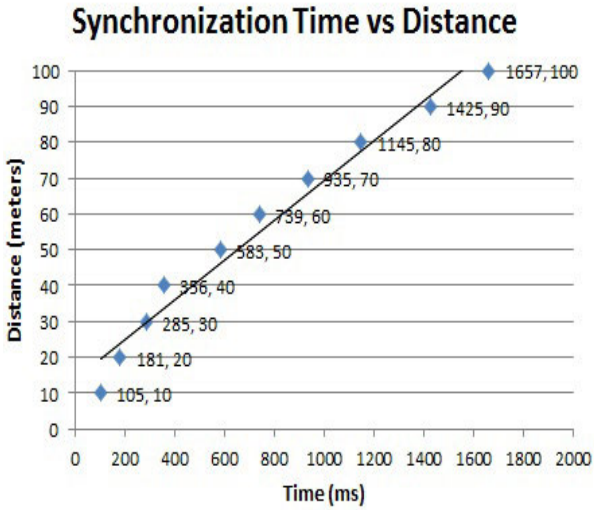

FIGURE 11. Comparison of operation synchronization time across the devices with different distance ranges.

TABLE 6. Time to update the session state on newly joined client, when connected clients has already performed activities.

WiFi Direct Group	Total Operations	Synch	Session Update Time
Samsung Grand Prime (Lollipop 5.0.2) 3 Devices	200	✓	4 Seconds
Samsung Grand Prime (Lollipop 5.0.2) 3 Devices	300	✓	5 Seconds
Samsung Grand Prime (Lollipop 5.0.2) 3 Devices	400	✓	7 Seconds

TABLE 7. Time to synchronize the session state on the client device after rejoin as other client devices have performed some activities during the disconnection period.

WiFi Direct Group	Inactive Period Operations	Synch	Session Update Time
Samsung Grand Prime (Lollipop 5.0.2) 3 Devices	100	✓	2 Seconds
Samsung Grand Prime (Lollipop 5.0.2) 3 Devices	200	✓	4 Seconds
Samsung Grand Prime (Lollipop 5.0.2) 3 Devices	300	✓	5 Seconds

TABLE 8. Memory used by the devices during the collaboration activities.

WiFi Direct Group	Operations	Memory (Each Device)
Samsung Grand Prime (Lollipop 5.0.2) 3 Devices	200	48 KB
Samsung Grand Prime (Lollipop 5.0.2) 3 Devices	400	96 KB
Samsung Grand Prime (Lollipop 5.0.2) 3 Devices	600	144 KB

5) ROBUSTNESS/STATE SYNCHRONIZATION AFTER DEVICE REJOIN

The system has to be resilient to small disconnections in ad-hoc settings. Thus, a client device may leave the group any time without any intimation and can rejoin later after some time. There is a need to synchronize the activities performed by the other client devices during the inactive period. In an experiment as shown in Table 7, we have measured the time taken by the device to synchronize the in active period activities on the client device. It is pertinent to mention that the ACS Manager only synchronizes the new operations on the client device by validating the timestamp.

6) MEMORY OVERHEAD

We have already mentioned in the discussion section that the application will capture the touch event performed on the device, convert it into JSON based string for broadcasting and synchronization across the collaboration devices. Mobile application will interpret the operation and execute on the mobile for synchronization purposes. In this experiment, we have estimated the memory consumed by the devices by calculating the operation size as shown in Table 8.

7) ENERGY CONSUMPTION

Mobile phone primarily runs on a battery power and you may notice that battery life become short when many activities simultaneously run on mobile phone i.e. games, WiFi etc and drain the battery quickly. In the last few years, we have seen a significant increase in the popularity of mobile application for businesses, education etc. But unfortunately energy consumption factors are usually ignored during the application development. We have conducted an experiment by estimation the battery drainage during the collaboration session as shown in Table 9. The activity was conducted in three different session and each session continued till 15 minutes. The results show that there is roughly 1% decrease for 100 operations during these 15 minutes sessions.

TABLE 9. Energy consumption of devices during the collaboration activities.

WiFi Direct Group	Operations	Duration (Minutes)	Battery Consumption (Out of 100 %)
Samsung Grand Prime (Lollipop 5.0.2) 3 Devices	100	15	1%
Samsung Grand Prime (Lollipop 5.0.2) 3 Devices	200	15	2%
Samsung Grand Prime (Lollipop 5.0.2) 3 Devices	300	15	3%

C. DISCUSSION

Performance is the key factor in case of mobile application, if your application does not perform well, the end user will uninstall the your app and look for other application that performs much better than your application. In case of distributed application it is more critical as multiple users are performing activities simultaneously. There is a need to synchronize the state across the all devices without any delay to ensure the application reliability. The experimental results empirically prove that the efficacy of ACS. The results show that the group /ad-hoc network formation takes few seconds and can be created on the fly at anytime, anywhere without the dependency of fixed infrastructure. Moreover, it can be seen that synchronization of operations also takes negligible time of 60ms. However distance between the collaborating devices increases the operations synchronization time, but it is still acceptable in case of distributed applications. In some cases, framework often jeopardizes some systems real-time performance due to inclusion of extra layer with the application but in this case as shown in the experiments, efficiency of system did not affect and application perform well as per the expectation of distributed environment.

Limited storage of mobile devices and power consumption is also considered while working on the framework and the experiment result shown in Table 8 reflects that the framework does not have any memory overhead and worked well on mobile devices. In case of paint application, we are storing the operation semantic and broadcasting to the

collaborating devices. However memory consumption will be variable depends upon the data semantic defined by the application developer for their applications. Moreover, the power consumption experiment as shown in Table 9 also reflects that the application energy consumption in satisfactory and does not have any negative impact on the mobile battery.

VI. CONCLUSION AND FUTURE DIRECTION

This article presents an architectural framework that is a step towards the materialization of the vision of ubiquitous computing by presenting a framework named ad-hoc Collaboration Space (ACS). The proposed framework aims to address a challenging task of providing seamless integration of different types of android based devices to share data and program in ad-hoc network settings, thus resulting into an ad-hoc collaboration space. The ACS framework facilitates the development of cross-device distributed mobile applications by hiding the complexities of mobile environment and empowers the application developer to focus on an application development using the easy to use common set of APIs provided by the framework. The novelty of our approach is that, to the best of our knowledge, it is the first framework that provides abstraction layer for the development of a distributed cross-device application in ad-hoc network settings. The results show that the proposed system is efficient in terms of device discovery, connectivity, and data and event synchronization. It has also been exposed in the form of APIs to reduce the application development time.

In future we intend to work on the stability of framework to impart fault-tolerance into the system, which would be required when the system involves group reformation in case of group owner disconnects. We also intend to evaluate the framework on different types of android based devices produced by different manufacturers, and for different android versions. Furthermore, we also intend to make the proposed framework inter-operable and heterogeneous by connecting android and iOS based devices in an efficient manner. We would also like to test the proposed framework on different types of applications including video based applications, or game based applications.

REFERENCES

[1] Facebook for Business. (Mar. 10, 2014). *Finding Simplicity in a Multi-Device Study commission by Facebook*. [Online]. Available: https://www.facebook.com/business/news/Finding-simplicity-in-a-multi-device-world

[2] M. Weiser, "The computer for the 21st century," Scientific American, 1991.

[3] 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 (CHI)*, Vancouver, BC, Canada, 1996, pp. 79–86.

[4] H. Sørensen, D. Raptis, J. Kjeldskov, and M. B. Skov, "The 4C Framework: Principles of interaction in digital ecosystem," in *Proc. ACM Int. Joint Conf. Pervasive Ubiquitous Comput.*, 2014, pp. 87–97.

[5] A. Sanctorum and B. Signer, "Beat Signer: Towards user-defined cross-device interaction," in *Proc. ICWE -Int. Workshops*, Cham, Switzerland, Oct. 2016, pp. 179–187.

[6] Samsung. *Samsung Flow*. Accessed: Feb. 5, 2017. [Online]. Available: http://www.samsung.com/uk/support/samsungflow

[7] P.-Y. Chi and Y. Li, "Weave: Scripting cross-device wearable interaction," in *Proc. 33rd Annu. ACM Conf. Hum. Factors Comput. Syst. (CHI)*, Seoul, South Korea, 2015, pp. 3923–3932.

[8] L. Frosini and F. Paternò, "User interface distribution in multi-device and multi-user environments with dynamically migrating engines," in *Proc. ACM SIGCHI Symp. Eng. Interact. Comput. Syst. (EICS)*, Rome, Italy, 2014, pp. 55–64.

[9] S. K. Badam and N. Elmqvist "Polychrome: A cross-device framework for collaborative Web visualization," in *Proc. ACM ITS*, 2014, pp. 109–118.

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

[11] P. Hamilton and D. J. Wigdor, "Conductor: Enabling and understanding cross-device interaction," in *Proc. CHI*, New York, NY, USA: ACM, 2014, pp. 2773–2782.

[12] *Create P2P Connections With Wi-Fi Direct | Android Developers*. Accessed: Mar. 10, 2020. [Online]. Available: https://developer.android.com/training/connect-devices-wirelessly/wifi-direct

[13] *P2P Technical Group, Wi-Fi Peer-to-Peer (P2P) Technical Specification*, Wi-Fi Alliance, Austin, TX, USA, Apr. 14, 2017.

[14] Apple Inc. *Apple-IOS 8-Continuity*. Accessed: Feb. 6, 2017. [Online]. Available: https://www.apple.com/ios/whatsnew/continuity/

[15] Google Inc. *Android Wear: The Developer'S Perspective*. Accessed: Sep. 23, 2017. [Online]. Available: https://www.google.com/events/io/io14videos/

[16] J. Domingue, M. Dzbor, and E. Motta, "Collaborative semantic Web browsing with Magpie," in *Proc. Eur. Semantic Web Symp., Semantic Web, Res. Appl.*, C. J. Bussler, J. Davies, D. Fensel, and R. Studer, Eds. Berlin, Germany: Springer, 2004, 388–401.

[17] A. W. Esenther, "Instant co-browsing: Lightweight real-time collaborative Web browsing," in *Proc. World Wide Web Conf.*, 2002, pp. 107–114.

[18] B. Johanson, S. R. Ponnekanti, C. Sengupta, and A. Fox, "Multibrowsing: Moving Web content across multiple displays," in *Proc. 3rd Int. Conf. Ubiquitous Comput. (UbiComp)*, London, U.K., 2001, pp. 346–353.

[19] J. Yang and D. Wigdor, "Panelrama: Enabling easy specification of cross-device Web applications," in *Proc. SIGCHI Conf. Hum. Factors Comput. Syst.*, Toronto, ON, Canada, Apr./May 2014, pp. 2783–2792.

[20] E. Agapie, J. Teevan, and A. Monroy-Hernandez, "Crowdsourcing in the field: A case study using local crowds for event reporting," in *Proc. HCOMP*, 2015, pp. 1–10.

[21] S. Kreitmayer, Y. Rogers, R. Laney, and S. Peake, "UniPad: Orchestrating collaborative activities through shared tablets and an integrated wall display," in *Proc. UbiComp*. New York, NY, USA: ACM, 2013, pp. 801–810. [Online]. Available: https://www.overleaf.com/project/5c44a99fa21e4668b5549120

[22] T. Pering, K. Lyons, R. Want, M. Murphy-Hoye, M. Baloga, P. Noll, J. Branc, and N. De Benoist, "What do you bring to the table?: Investigations of a collaborative workspace," in *Proc. 12th ACM Int. Conf. Ubiquitous Comput. (Ubicomp)*, 2010, pp. 183–192.

[23] *Wi-Fi Direct | Wi-Fi Alliance*. Accessed: Mar. 10, 2020. [Online]. Available: https://www.wi-fi.org/discover-wi-fi/wi-fi-direct

[24] 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 (CSCW)*, 2000, pp. 221–230.

[25] 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. (PERCOM)*, Mar. 2006, pp. 22–35.

[26] H. Wiltse and J. Nichols, "PlayByPlay: Collaborative Web browsing for desktop and mobile devices," in *Proc. 27th Int. Conf. Hum. Factors Comput. Syst. (CHI)*, 2009, pp. 1781–1790.

[27] 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 (MobileHCI)*, 2008, pp. 311–314.

[28] 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.

[29] G. Laput, R. Xiao, X. A. 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. (UIST)*. New York, NY, USA: ACM, 2014, pp. 389–394.

[30] 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. (CHI)*. New York, NY, USA: ACM, 2013, pp. 1451–1460.
[31] 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.
[32] J. M. Zacks, B. Tversky, and G. Iyer, "Perceiving, remembering, and communicating structure in events," *J. Exp. Psychol., Gen.*, vol. 130, no. 1, p. 29, 2001.
[33] X. A. Chen, T. Grossman, D. Wigdor, and G. Fitzmaurice, "Duet: Exploring joint interactions on a smart phone and a smart watch," in *Proc. ACM CHI*, 2014, pp. 159–168.
[34] *Bluetooth Technology Website*. Accessed: Mar. 10, 2020. [Online]. Available: https://www.bluetooth.com
[35] L. D. Geronimo, M. Husmann, and C. M. Norrie, "Surveying personal device ecosystems with cross-device applications in mind," in *Proc. 5th ACM Int. Symp. Pervasive Displays (PerDis)*. New York, NY, USA: ACM, 2016, pp. 220–227.
[36] S. Zhu, S. Xu, S. Setia, and S. Jajodia, "LHAP: A lightweight hop-by-hop authentication protocol for ad-hoc networks," in *Proc. IEEE Int. Workshop Mobile Wireless Netw. (MWN)*, May 2003, pp. 749–755.
[37] F. Brudy, C. Holz, R. Rädle, C. J. Wu, S. Houben, C. Klokmose, and N. Marquardt, "Cross-device taxonomy: Survey, opportunities and challenges of interactions spanning across multiple devices," in *Proc. CHI Conf. Hum. Factors Comput. Syst.*, Glasgow, U.K., May 2019, pp. 4–9.
[38] 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," in *Personal and Ubiquitous Computing*. Springer, 2019, pp. 1–11, doi: 10.1007/s00779-019-01280-7.
[39] 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. (CHI)*. New York, NY, USA: ACM, 2018, Art. no. 184. [Online]. Available: https://doi.org/10.1145/3173574.3173758
[40] S. K. Badam, A. Mathisen, R. Radle, 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, doi: 10.1109/TVCG.2018.2865144.

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. He is currently working as the Project Manager with the Virtual University of Pakistan.

ADNAN ABID (Member, IEEE) was born in Gujranwala, Pakistan, in 1979. He 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 the Politecnico di Milano, Italy, in 2012. He has spent one year in EPFL, Switzerland, to complete his M.S. thesis. His research interests include computer science education, information retrieval, and data management. He is currently working as an Associate Professor with the Department of Computer Science, University of Management and Technology, Pakistan. He has almost 40 publications in different international journals and conferences. He has served as a reviewer in many international conferences and journals. He is an Associate Editor of IEEE Access.

MUHAMMAD SHOAIB FAROOQ (Member, IEEE) was born in Lahore, Pakistan. He received the M.Sc. degree from Quaid-e-Azam University, Pakistan, in 1995, the M.S. degree in computer science from Government College University, in 2007, and the Ph.D. degree from Abdul Wali Khan University, Pakistan, in 2015. He possesses more than 20 years of teaching experience in the field of computer science. He is currently working as an Associate Professor with the Department of Computer Science, University of Management and Technology, Pakistan. His research interests include theory of programming languages, and computer science education.

ADNAN SHAHZADA was born in Pakistan. He received the B.S. degree in computer science from the National University of Computer and Emerging Sciences, Pakistan, the M.S. degree in artificial intelligence from the University of Dalarna, Sweden, and the Ph.D. degree from the Politecnico di Milano, Italy. He possesses a diversified experience from the industry and academia.

UZMA FAROOQ was born in Lahore, Pakistan, in 1983. She received the B.S. degree in computer science from the University of the Punjab, in 2005, and the M.S. degree in computer science from the National University Computer and Emerging Sciences, Pakistan, in 2007. She is currently working as an Assistant Professor with the Department of Computer Science, University of Management and Technology, Pakistan. She is teaching basic and advanced computer programming courses in the undergraduate programs in computer science and software engineering.

KAMRAN ABID (Member, IEEE) was born in Gujranwala, Pakistan, in 1977. He 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, and the Ph.D. degree in electrical engineering from the University of Glasgow, in 2011. He is currently working as an Assistant Professor with the Department of Electrical Engineering, University of the Punjab, Pakistan. He has published nearly 40 articles in international journals and conferences.