TSINGHUA SCIENCE AND TECHNOLOGY
ISSN 1007-0214 11/13 pp333–346
DOI: 10.26599 / TST.2018.9010082
Volume 23, Number 3, June 2018

# BLE Mesh: A Practical Mesh Networking Development Framework for Public Safety Communications

Bo Zhang, Yufeng Wang*, Li Wei, Qun Jin, and Athanasios V. Vasilakos

**Abstract:** Owing to advanced storage and communication capabilities today, smart devices have become the basic interface between individuals and their surrounding environment. In particular, massive devices connect to one other directly in a proximity area, thereby enabling abundant Proximity Services (ProSe), which can be classified into two categories: public safety communication and social discovery. However, two challenges impede the quick development and deployment of ProSe applications. From the viewpoint of networking, no multi-hop connectivity functionality component can be directly operated on commercially off-the-shelf devices, and from the programming viewpoint, an easily reusable development framework is lacking for developers with minimal knowledge of the underlying communication technologies and connectivity. Considering these two issues, this paper makes a two-fold contribution. First, a multi-hop mesh networking based on Bluetooth Low Energy (BLE) is implemented, in which a proactive routing mechanism with link-quality (i.e., received signal strength indication) assistance is designed. Second, a ProSe development framework called BLE Mesh is designed and implemented, which can provide significant benefits for application developers, framework maintenance professionals, and end users. Rich application programming interfaces can help developers to build ProSe apps easily and quickly. Dependency inversion principle and template method pattern allow modules in BLE Mesh to be loosely coupled and easy to maintain and update. Callback mechanism enables modules to work smoothly together and automation processes such as registration, node discovery, and messaging are employed to offer nearly zero-configuration for end users. Finally, based on the designed ProSe development kit, a public safety communications app called QuoteSendApp is built to distribute emergency information in close area without Internet access. The process illustrates the easy usability of BLE Mesh to develop ProSe apps.

**Key words:** public safety communications; device to device; bluetooth low energy; mesh networking; development framework

## 1 Introduction

Recently, Proximity Service (ProSe) has become a promising mobile industry, which aims to discover valuable instances of the applications running on devices within proximity of each other, and ultimately exchange application-related contents. People's interactions in ProSe become inherently location-centric and tied to their current physical neighborhood. The purely local, ephemeral, and decentralized characteristics of ProSe can facilitate sharing

• Bo Zhang, Yufeng Wang, and Wei Li are with Nanjing University of Posts and Telecommunications, Nanjing 210023, China. E-mail: wfwang@njupt.edu.cn.
• Qun Jin is with Department of Human Informatics and Cognitive Sciences, Waseda University, Saitama 359-1192, Japan. E-mail: jin@waseda.jp.
• Athanasios V. Vasilakos is with Department of Computer Science, Electrical and Space Engineering, Luleå University of Technology, Sweden. E-mail: th.vasilakos@gmail.com.
* To whom correspondence should be addressed.
Manuscript received: 2017-12-11; accepted: 2018-01-23

of experiences between users in real time and in a specific place. ProSe provides users additional spatial and temporal semantics, i.e., a sense of "here and now", on which abundant proximity applications can be built easily.

Existing technologies used to serve the ProSe can be broadly divided into Over-The-Top (OTT) and Device-to-Device (D2D) solutions[1]. In the OTT model, a server (usually located in the cloud) receives periodic location updates from user mobile devices (typically using GPS), and the server then determines proximity based on location updates and interests. The constant location updates not only result in a significant battery impact because of GPS power consumption and the periodic establishment of cellular connections, but also causes serious privacy problems. Moreover, many scenarios are characterized by a lack of adequate access to mobile telecommunications because of extra-ordinary events that can disable disrupt or overwhelm mobile telecommunications infrastructure. In these scenarios, D2D communication is the only approach to realize proximity communication.

Different from OTT, D2D schemes forgo centralized processing in identifying relevancy matches, and instead autonomously determine relevance at the device level by transmitting and monitoring relevant attributes. According to the spectrum bands used D2D communications, D2D can be classified into licensed D2D (exploiting the cellular spectrum for both D2D and cellular links, e.g., LTE Direct) and unlicensed D2D communications (exploiting industrial, scientific, and medical bands, or the so-called ISM band)[2]. Considering that abundant commercially available mobile instruments and D2D applications have been deployed in an unlicensed spectrum partially because of the tedious processes to obtain usage permission from licensed spectrum owners, this study focuses on the design and implementation of the development framework in an unlicensed spectrum, such that this framework could be deployed to widely commercial mobile devices. Typical D2D-enabled communication technologies in the unlicensed spectrum include Bluetooth (including Bluetooth Classic and Bluetooth Low Energy or BLE) and Wi-Fi Direct.

Generally, most proximity service applications can be classified as two categories of scenarios: public safety communication and social discovery. In particular, public safety communication implies that D2D links can operate unimpeded in a disaster-hit area where all BSs are paralyzed. In fact, 3GPP (Third Generation Partnership) Release 12 of the LTE-Advanced standard specifies a general concept of Proximity-based Services (ProSe) that allows physically close devices to discover themselves and communicate via direct links[3]. ProSe is meant for public safety communication and commercial applications although Release 12 solely emphasizes on the public safety. Public safety communication covers a variety of situations including: (a) war or terror attacks, where infrastructure may be purposefully targeted; (b) adverse weather that may disrupt logistical supply chains to mobile communications infrastructure, e.g., energy supply, or affect the infrastructure itself, or both; (c) disasters such as earthquakes, floods, bush fires or fire storms which may damage, inundate or isolate mobile communication assets through various means, including disabling the back-haul that connects the assets to the rest of the network, thereby reducing the effective range of the cellular signal, e.g., due to heat and smoke from fire-storms; and (d) civil or other emergency where a surge in demand overwhelms the infrastructure capacity to provide service[4].

### 1.1 Research motivation

With the growing popularity of proximity service, several proximity solutions or applications have been presented. However, the absence of practical proximity solutions in the market is a concern. Among other issues, two challenges impede the quick development and deployment of ProSe services: from the viewpoint of networking, no multi-hop mesh connectivity functionality component can be directly operated on commercially off-the-shelf devices (i.e., without customizing/rooting those devices), and from the programming viewpoint, there is a lack of easy-to-use and reusable development frameworks that allow application developers to exploit the potential advantages of proximity service with minimal knowledge of the underlying communication technologies and connectivity.

Traditionally, D2D networking mainly consists of local, opportunistic, and single-hop communication, whereas multi-hop message forwarding schemes are typically needed to extend the coverage of ProSe applications. Typically, two categories of multi-hop enabling schemes exist:

- Store-carry-forward multi-hop schemes (or asynchronous schemes), in which, single-hop direct networking technologies are intrinsically utilized. That is, users in single-hop contact area exchange and store data, and through users' mobility, the data could be propagated. These schemes are based on single-hop direct networking (not mesh networking).
- Wireless mesh schemes (or synchronous schemes), in which, each mobile terminal simultaneously connects to

more than one other terminal, and relays data to widen the communications range in a synchronous manner.

Although exploiting user mobility and utilizing the store-carry-forward pattern, single-hop networking based ProSe apps can disseminate content in larger local areas. However, static scenarios (e.g., public transports systems, campuses, bars, concerts, and others) are also popular in ProSe applications. Therefore, D2D-based mesh networking is also necessary. However, in practice, creating D2D-based mesh networking is challenging because typical mobile devices do not implement the configuration, routing, and name resolution functions required to operate in an ad-hoc scenario. Software restrictions on modern mobile operation systems, such as Android and iOS, even prevent mobile devices from actively participating in ad-hoc networks without circumventing vendor barriers. Of course, “rooting” enables to access to advanced capabilities, but we do not take consider this possibility because the rooting process requires skills that are beyond what the average user possesses, and the process renders the warranty null and void. Thus, we only act upon application-layer functionalities, i.e., no changes can be performed at the transport or network layer.

Besides the multi-hop networking issue, from the programming viewpoint, to create ProSe applications, developers would have to begin with deciding how to handle ProSe’s requirements in different layers, including communications technologies, peer discovery, connectivity maintenance, routing policy, profile matching, and various specific application requirements. Occasionally, the process may be extremely complex if developers decide to implement applications from scratch. An appropriate ProSe development framework can help app designers address these demands and highlight meeting the application-specific requirements by realizing the required functionalities in middle framework which can be efficiently and stably reused by ProSe developers.

### 1.2 Main contributions

The contributions of our study are two-fold. First, a BLE-based multi-hop mesh networking is proposed. Specifically, besides the traditional routing metric of the number of hops, D2D link quality represented as RSSI is incorporated into the routing selection. Then, a ProSe development framework called BLE Mesh is designed and implemented, which can provide significant benefits for various ProSe stakeholders, such as app developers, framework maintainers, and end users. Rich APIs help developers to build ProSe apps easily and quickly. Dependency inversion principle and template method pattern make modules loosely coupled and easy to maintain and update the BLE Mesh framework. Callback mechanism enables modules to efficiently work together and automation processes such as registration, node discovery, and messaging are utilized to offer end users nearly zero-configuration of applications. Finally, based on the designed ProSe development kit, a public safety communications app called QuoteSendApp is built (to distribute emergency information in a close area with Internet), which illustrates the easy usability of BLE Mesh to develop ProSe apps.

The rest of this paper is structured as follows. In Section 2, we briefly discuss several typical development ProSe frameworks, the existing ProSe mesh networking schemes, and their weakpoints. Section 3 describes the designed BLE based ProSe development framework, BLE Mesh, including the fundamental components and their implementation, and the proposed link-quality assistant routing path selection method is also offered. Section 4 provides a public safety communications service built on the BLE Mesh framework called QuoteSendApp, which illustrates the easy usability of BLE Mesh to develop ProSe applications. Finally, this paper is concluded.

## 2 Related Work

### 2.1 Existing multi-hop mesh networking schemes

As described in the preceding section, by exploiting users’ mobility and utilizing store-carry-forward patterns, single-hop networking based ProSe services can disseminate contents in a large local area. However, static scenarios are also popular in ProSe applications. Besides, most traces have highlighted that nodes are stationary for a large portion of time interrupted by mobility events. Therefore, mesh networking schemes are also necessary.

WiFi is one of the major occupants in the unlicensed band, which is actually an indispensable component for all commercial mobile devices. WiFi-based solutions for direct connectivity include WiFi ad-hoc and WiFi Direct. However, none of them can be directly used to construct mesh networking without rooting/customizing the device and operating systems.

Specifically, ad-hoc mode is one of the two standard modes for WiFi, but almost no commercial mobile terminals and mobile operating systems support this mode, partly due to lack of security, inability to be used in parallel with infrastructure mode, and other issues. Based on WiFi virtualization technologies, Ref. [5] presented an approach for WiFi infrastructure mode ad-hoc networks, in which mobile devices simultaneously function as AP and as a station to mesh with other mobile devices that assume both roles, thereby establishing multi-hop communication networks in WiFi infrastructure mode.

WiFi Direct builds upon the successful IEEE 802.11 infrastructure mode, and enables devices to negotiate who will take over the roles of AP-like functionalities (i.e., P2P Group Owner (P2P GO)) and P2P Clients. Although Wi-Fi Direct specification states that these two logical roles could be executed simultaneously by the same device through multiple physical/virtual interfaces, for example, by using different frequencies (if the device has multiple physical radios) or time-sharing the channel through virtualization, which can facilitate building the mesh networking structure, it cannot actually support the simultaneous connection until now. In detail, after the GO is elected, the role of each peer remains unchanged during the entire group session. Only when the GO leaves the group do the peers become disconnected and a new group must be created; in practice, direct communication between clients in a formed P2P group (without relaying through P2P GO) is not supported yet.

Although the available WiFi Direct cannot conceive two virtual interfaces simultaneously, Refs. [6, 7] designed a multi-hop communication technology on Android mobile devices, based on the idea that a WiFi device can enable two network interfaces: the conventional standard WiFi interface and the P2P GO interface for WiFi Direct connection. Therefore, integrating these two interfaces can create a multi-group physical topology (i.e., bridge nodes), enable these two interfaces on a mobile device, and allow a GO (in WiFi Direct mode) to be a legacy client in another group. Indeed, each group represents a different WiFi Basic Service Set.

Instead of the continuous multi-hop networking, a method introduced by Ref. [8] achieves intermittent multi-hop communication among open-source, non-rooted Android devices using Wi-Fi Direct Technology. Specifically, this method makes all devices become GO when no data transmission occurs. In this way, all devices are ready to be discovered and connected. If a device is trying to initiate data transmission, it must first remove its GO status and connect to the target device as a WiFi P2P client. Once the connection is completed, the device is allowed to send the data; after the target device has received the data, it disconnects from the group and removes the client status. Then the device becomes a group owner again and prepares for the next transmission cycle.

To support simultaneous and multi-hop connection on commercially available smartphones, a multi-hop connectivity framework called BWMesh is proposed by combining Bluetooth and WiFi Direct technologies, in which devices with those two wireless interfaces are enabled to act as bridges for mesh networking[9].

All these works based on virtualization and/or multiple physical interfaces are complicated and unstable. Furthermore, all WiFi-based D2D technologies in mobile devices are limited by powersaving features. Basically, the device publishing a service has to keep broadcasting information about the service, even if there are no clients nearby, while the device searching for a service has to keep listening, even though nobody around is providing the necessary service. In the worst case scenario, the battery drain would never get rewarded, thereby making the technology unsatisfactory to end users.

Recently BLE has gained significant momentum. However, the original design of BLE focused on star topology networking, which limits network coverage range and precludes end-to-end path diversity. Scatternets[10], where a Bluetooth master device is also a slave in a different network, are standardized, but a single device still encounters difficulty in scanning and establishing the connection at once. Bluetooth 4.1 and later versions have been modified to be able to connect to multiple centrals to standardize BLE mesh networks[11]. That is, a BLE device can act as both Central (i.e., master in traditional BLE) in a piconet and Peripheral (i.e., slave in traditional BLE) in another piconet, which naturally enables mesh networking, i.e., scatternet topology[12].

BLE Mesh Network (BMN) is a Directed Acyclic Graph (DAG) based routing solution designed over Bluetooth 4.1, and its operation consists of three phases: construction, maintenance, and optimization[13]. The goal of the construction phase is to establish link layer connections between neighboring devices, determining nodes' parents, and creating routing tables. The maintenance phase aims to improve BMN parameter settings and forward packets to their intended destinations. The optimization phase is intended for node weight balancing so that all nodes in the network have nearly equal distance to the DAG root.

Reference [14] proposed an algorithm to form scatternets and on-demand routing for Bluetooth 4.1; the algorithm consists of two phases: scatternet formation and route discovery. In the scatternet formation phase, masters create a list of their connected slaves and vice versa. Nodes that perform both roles elaborate both slave and master lists. To allow connection establishment between a new node and its neighbors, the node alternates scanning and advertising states. The node assumes the master or slave role depending on its role as a scanner or an advertiser, at connection establishment time. In route discovery, the source node first sends a route request to its master. If the target destination is not in the slave list of the master, the latter initiates a breadth-first search by forwarding the route request to any slave in its piconet that participates in another piconet. Such slaves resend the route request to their masters in other piconets. This process continues until the destination is found. However, most of these routing algorithms are based on hop count because the D2D wireless link quality varies significantly and at times, the chosen path may not be stable enough.

### 2.2 Existing ProSe development kit

To facilitate and improve the efficient of the development process of ProSe applications, it is imperative to develop a reusable framework that allows application developers to exploit the proximity, mobility, and more communication technologies, quickly leverages existing solutions to solve the issues in different layers, and emphasizes the provision of specific functions or services in various scenarios. Aside from helping developers with their work, the development kit should also consider the easy maintenance of the suite itself and the usage convenience to the end user. In the literature, several ProSe development kits are described.

Serval Mesh allows smartphones to build self-organized mesh networks using WiFi in AP and client modes and a store-and-forward protocol without the need for a mobile phone operator. Normally complex processes, such as network address allocation, have been solved in a completely transparent manner, relieving end users of the demands of such tasks. This feature is important to increase the solution's utility among people with low technological literacy. Based on mesh network, Serval Mesh provides rich functions such as mesh-based telephone calls, text messaging, and file distribution[15].

Another kit, Proxima, is a framework that employs ad-hoc D2D connections and proactive mesh routing for a decentralized topology with proximity-based rich content dissemination. The Proxima framework provides a fully asynchronous, thread-safe API and can be used by multiple client applications on a single device. Thus, API users need not to worry about how to discover neighbors and what services are available; the framework itself takes care of these low level details. Communication with the framework is conducted using asynchronous method calls with user-supplied callback functions. For mobile application developer, using the Proxima framework to build ProSe apps only requires registering the application with the framework as a client at the beginning. Then, the framework will send the devices found in the neighborhood to the developer, and any changes in the network will be updated via self-defined callback function. Furthermore, Proxima provides a nearly zero-configuration interface. The only aspect that has to be configured is the device/user name (similar to that of the Bluetooth device name) which can be achieved programmatically with a simple API call. However, this framework needs root access to the device to enable Wi-Fi ad-hoc, which is difficult to promote among off-the-shelf devices[16].

Another framework, USABle, offers developers multi-hop and carry-and-forward strategies and a set of technology-independent interfaces to handle communication requirements such as neighbor discovery, connection and disconnection, and message delivery. This framework is based on a modular, extensible, and layer-based architecture that permits developers to plug in new communication technologies and routing strategies, and reuse most of the existing functionalities. USABle follows the layered architectural style and is divided into three layers (connection-aware layer, network layer, and application layer) to improve modularity and adaptability; any layer can be adapted or modified according to application requirements[17].

In brief, the aforementioned development kits mainly exploit the ad-hoc networking technologies or store-carry-forward methodology to form multi-hop network and forward messages. The former is difficult to adopt and deploy directly on off-the-shelf mobile terminals, and the latter cannot meet the real-time feature of ProSe apps in relatively static scenarios. Therefore, investigating the D2D-based mesh networking schemes, especially those applicable for off-the-shelf terminals, is necessary.

## 3 Design of BLE Mesh Framework

### 3.1 System architecture of BLE Mesh

As shown in Fig. 1, the system structure of BLE Mesh mainly consists of three modules in middle layers: one-hop direct connection, message management, and multi-hop networking. The direct connection module utilizes the underlying D2D communication interface to enable peer discovery, connection, and bitdata transmission between

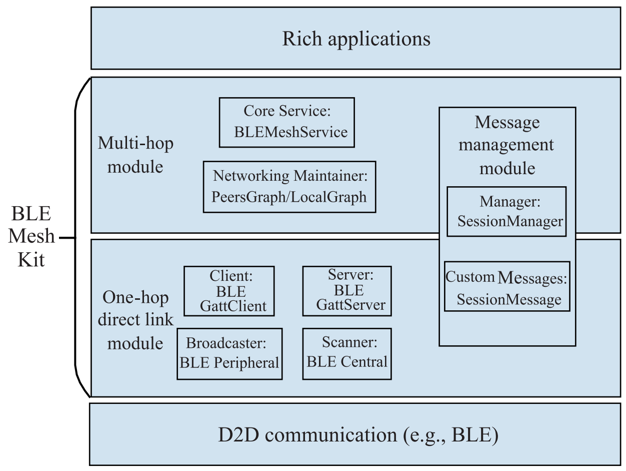

Rich applications
Multi-hop module
Core Service: BLEMeshService
Networking Maintainer: PeersGraph/LocalGraph
Message management module
Manager: SessionManager
Custom Messages: SessionMessage
BLE Mesh Kit
One-hop direct link module
Client: BLE GattClient
Server: BLE GattServer
Broadcaster: BLE Peripheral
Scanner: BLE Central
D2D communication (e.g., BLE)

**Fig. 1 System architecture of BLE Mesh kit.**

directly connected devices in close range. Then, based on the core functionalities provided by one-hop direct connection module, the message management module designs a variety of message types, manages the message transmission/receipt, and maintains connectivity among devices. Subsequently, the multi-hop module realizes multi-hop networking functionality including routing selection and topology maintenance. Finally, based on the layering principle, those three modules open their core interfaces for the ProSe developer to quickly build abundant featured apps.

### 3.1.1 Direct link module

BLE defines a complete protocol architecture to enable low power communication between devices, consisting of two main parts: the Controller, which performs radio interface tasks, and the Host, which offers higher layer functionality and supports applications. The communication between Host and Controller is conducted through the Host Controller Interface. In particular, GATT in the Host part defines client and server roles, whose procedures can be classified into three basic types: discovery, client-initiated, and server-initiated procedures. Specifically, BLE devices can have two different roles, either as central (i.e., scanner) or peripheral (i.e., advertiser). In the discovery phase, central is responsible for scanning other surrounding devices, and peripheral for constantly advertising to be seen by other devices. When a scanner finds the beacon of another advertiser, and if the advertiser opens the GATT server, then the GATT client of the scanner can start the connection request. After accepting the connection from the scanner, the advertiser's GATT client launches the connection to its peer's GATT server. Once successfully completed, the peer connection can be built between those two devices. To comply with the procedure, the BLE-based direct link module in our architecture is divided into the following four sub-components.

• BLE Central, which provides the ability to start/stop scanning and inform the GATT Client of the scanning results;

• BLE Peripheral, which provides the ability to start and stop advertising/broadcasting;

• BLE GattServer component configures, opens, and stops the GATT server. When the server is open, it can receive the outside connection establishment request and process the data if the connection is accepted;

• BLE GattClient component is responsible for initiating a connection request to the server, and sends data to the server after the connection is established.

Table 1 illustrates the main APIs provided by one-hop direct link module.

### 3.1.2 Message management module

The message management module is based on the direct connection module and is responsible for managing service discovery and sending and receiving specific messages with designated formats. Messages in different formats are used for various specific services, including identity message, routing message, and application-dependent message. All specific types of message are inherited from the base class SessionMessage. Its header contains the following basic fields: "TYPE" (type of message), "BODY_LENGTH" (length of message body), "ID" (unique ID of message), and "MAC_ADDRESS" (MAC address of the message sender). Customized SessionMessage header and body can be extended from the base class, and their semantic meanings are interpreted by the upper layers. In the current BLE Mesh framework, three types of messages are defined, and all messages are stored and transmitted in JSON (JavaScript Object Notation) format.

• **IdentityMessage.** This message is used by the local node to send its identity to other nodes. In addition to

**Table 1 Main APIs in direct connection module.**

```
    public abstract void startDiscovery();
//simultaneously launch advertising and scanning
    public abstract void advertise(); // Only start advertising
    public abstract void scanForPeers(); //Only start scanning
    public abstract void stop(); //Stop advertising and scanning
    public abstract boolean sendData(byte[] data, Set<String> identifier);
//Broadcast data to the peers in identifier set
    public abstract boolean sendData(byte[] data, String identifier);
//Send data to the designated identifier
```

all fields in base class SessionMessage, "ALIAS" field is added to the header area, to indicate the user-defined human-readable name. The body of this message area is empty.

• **GraphMessage.** This message is designed to describe the change of network topology, when nodes join or leave the network, which stores the information of the changes of nodes and links in the mesh network. In GraphMessage, two extra header fields are appended. "REMOTE_ACTION" (the value can be set as "Join" or "Leave") represents the type of network change: "Join" indicates a new node joining the network and "Leave" indicates that some node leaves the network. "CAST_FORM" represents the mode of message dissemination: "Unicast" is used to exchange mesh information when two close nodes within one-hop range are connected to each other; "Broadcast" is used to notify other nodes about the changes in network topology. The body of GraphMessage consists of two parts: "VERTEXES" composed of all nodes' information and "EDGES" composed of all links' information, including the link quality characterized by RSSI.

• **DataTransferMessage**. This message is used to carry multi-hop application-related data. The following fields are appended to the header area: "DESC" (address of destination device), "SEND_DATE" (send time), "SOURCE" (address of source device), and "TTL" (time-to-live, lifetime of message). The message body carries the application-related data.

In brief, all message types are extended from the base class SessionMessage, and composed of the customized message header and body. Furthermore, SessionManager class manipulates all types of messages.

The main APIs provided by the message management module is reported in Table 2.

#### 3.1.3 Multi-hop networking module

By exploiting the customized messages in GraphMessage, the multi-hop network module can obtain the topology and quality of links between nodes in the mesh network. Then the link-quality assistant routing mechanism can be used to select a data path, and then DataTransferMessage is utilized to send a message to the designated destination along the selected path. In particular, the path selection and routing table maintenance are integrated into the PeersGraph/LocalGraph Networking Maintainer Sub-Module. PeersGraph is a generic class used to describe mesh information. LocalGraph, inheriting from PeersGraph, adds a set of methods to maintain and update local network information, and calculates the routing path and next hop to the specified node.

Several main APIs enabled by multi-hop networking maintainer class LocalGraph are listed in Table 3.

With the help of LocalGraph, the core service, BLEMeshService, which interacts with the upper application, will provide the core APIs shown in Table 4. These APIs will help developers quickly implement user registration, device discovery, and messaging features.

### 3.2 Low coupling of modules

We have to consider how to maintain and update the ProSe development kit to smoothly incorporate the latest and future emerging D2D communications technologies. A mainstream approach is to reduce coupling of modules. In the software design, the dependency inversion principle is

**Table 2 Main APIs provided by message management module.**

| |
|---|
| public void broadcastMessage(SessionMessage message); //Broadcast message |
| public synchronized void sendMessage(SessionMessage message, Peer recipient); //Send the customized message to the directly connected peer "recipient" |
| public Set<Peer> getAvailablePeers(); //List all directly connected peers |

**Table 3 Main APIs in multi-hop maintainer, LocalGraph.**

| |
|---|
| synchronized public void newDirectRemote(Peer remoteNode); //Discovery new direct node and connect |
| synchronized public void mergeGarph(Peer remoteNode, PeersGraph otherGraph); //Receive the updating information of network topology, and merge it into the current node's local graph |
| synchronized public void lostDirectRemote(Peer remoteNode); //The direct connection to the peer "remoteNode" broken |
| synchronized public void trimGraph(Peer remoteNode, PeersGraph otherGraph); //Find the unreachable nodes and delete it from local graph (i.e., trim graph) |
| public void calCluateShortestPath(); //Calculate the paths from the local node to other nodes |
| public Peer getNextReply(String desc); //Return the next hop to the designated node "desc" |

**Table 4 Main APIs in multi-hop networking module.**

| |
|---|
| Public void registerLocalUserWithService(String userAlias, String serviceName);//Register the local node with human-readable name "userAlias" |
| public void startDiscovery();//Start the device discovery |
| public LocalPeer getLocalPeer()//Return the instance of local node |
| public void send(byte[] data, Peer recipient);//Send data to the designated node "recipient" in the multi-hop network |

an important principle to reduce the module coupling. The core idea is that the dependency between modules should rely on abstraction instead of detailed implementation. From a programming viewpoint, this idea implies that the implementation class depends on the abstract class or interface, not vice versa. BLE Mesh uses the template method pattern to enforce the principle of inverse dependency, making the development kit easily plug in new emerging D2D communications technologies or new multi-hop networking scheme.

For example, Fig. 2 depicts the design of the direct connection module. Transport is an abstract class that defines the main method, while the specific implementation functions are deferred to the corresponding subclasses. Currently we use BLE as underlying D2D communication technology, so the inherited class BLETransport is defined and functions in Transport are performed in BLETransport. If other alternative communication technologies would be adopted, a new derived class can be realized to replace BLETransport, without affecting the upper layer.

BLE Mesh uses the template method pattern to enforce the principle of inverse dependency, thereby enabling the development kit to easily plug into emerging D2D communications technologies or new multi-hop networking schemes.

## 3.3 Lightweight configuration of BLE Mesh

Generally, end users want to use modern applications without extremely complex operations. A suitable ProSe development kit should help developers to design easy-to-use applications for end users. We argue that, for easy configuration, the development framework should smoothly realize the core functionalities required by ProSe applications, including user registration, node discovery, and multi-hop messaging. These tasks can be performed by using the asynchronous method calls along with developer-supplied callback functions.

For easy configuration, the upper modules have to invoke the APIs provided by lower modules to use the functionality in lower modules. Equally importantly, the lower modules should notify the upper modules about interesting events in an asynchronous manner such that the upper modules can correspondingly deal with them in real time through the automatic event response. For example, to find the nodes within multiple hops, a one-hop direct link module should notify the event of finding a new node to the message management module, and the message management module updates the network information table and sends the table to the multi-hop networking module for topology maintenance and path calculation.

As shown in Fig. 3, the BLE Mesh framework utilizes the callback mechanism, that is, some callback notification interfaces are defined and inserted between two neighboring upper and lower modules The upper module is responsible for implementing callback interfaces, and the references of the implemented interfaces are passed to the lower module. In brief, the functionalities in the lower modules can be explored through invoking the APIs in the lower module, and the events of interest to the upper modules can be asynchronously notified through the callback functions, such that all modules can collaborate efficiently. Then, the BLE Mesh can smoothly provide the

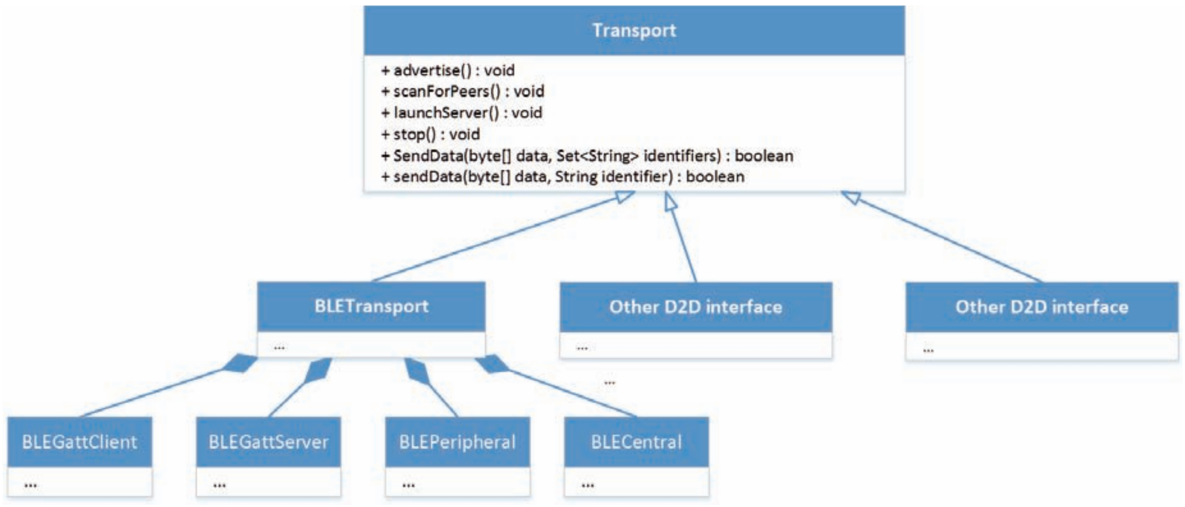


**Fig. 2 Class diagram for transport using template method pattern.**

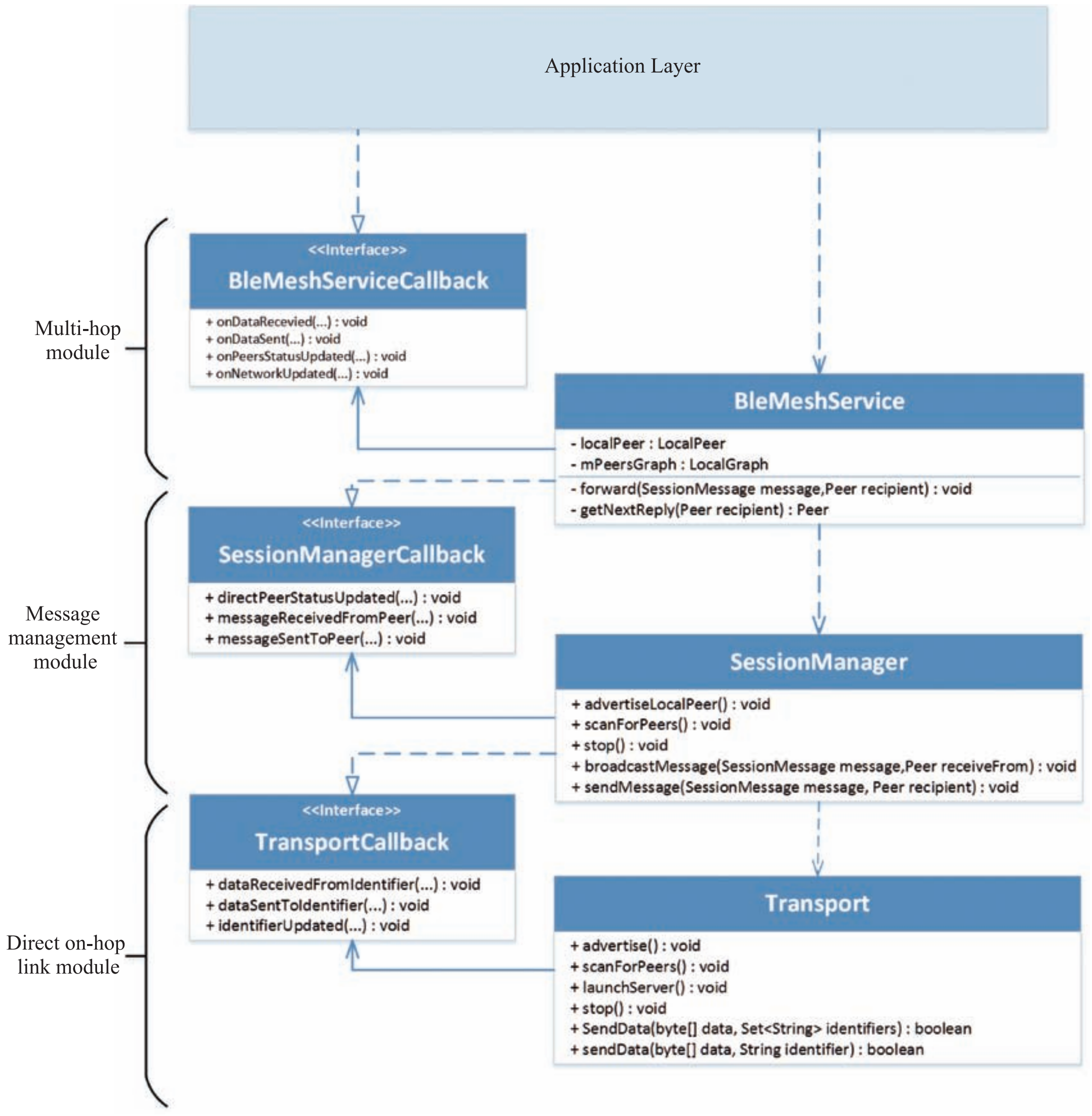

Application Layer
Multi-hop module
<<Interface>>
BleMeshServiceCallback
+ onDataRecevied(...) : void
+ onDataSent(...) : void
+ onPeersStatusUpdated(...) : void
+ onNetworkUpdated(...) : void
BleMeshService
- localPeer : LocalPeer
- mPeersGraph : LocalGraph
- forward(SessionMessage message,Peer recipient) : void
- getNextReply(Peer recipient) : Peer
Message management module
<<Interface>>
SessionManagerCallback
+ directPeerStatusUpdated(...) : void
+ messageReceivedFromPeer(...) : void
+ messageSentToPeer(...) : void
SessionManager
+ advertiseLocalPeer() : void
+ scanForPeers() : void
+ stop() : void
+ broadcastMessage(SessionMessage message,Peer receiveFrom) : void
+ sendMessage(SessionMessage message, Peer recipient) : void
Direct on-hop link module
<<Interface>>
TransportCallback
+ dataReceivedFromIdentifier(...) : void
+ dataSentToIdentifier(...) : void
+ identifierUpdated(...) : void
Transport
+ advertise() : void
+ scanForPeers() : void
+ launchServer() : void
+ stop() : void
+ SendData(byte[] data, Set<String> identifiers) : boolean
+ sendData(byte[] data, String identifier) : boolean

**Fig. 3 Illustration of asynchronous methods and developer-supplied callback functions.**

necessary procedures required by all ProSe apps, including user registration, node discovery, and message sending and receiving, which in turn, allow users to operate the system as simply as possible. Figure 4 shows the automation procedures, including the interactions among end users (i.e., ProSe apps) and three core modules in BLE Mesh development kit.

• User registration. End users only need to input the nickname, and then BLE Mesh performs other tasks such as using a nickname to build a local node instance, constructing instances of underlying modules, and completing local node registration.

• Node discovery. Once started by the user, BLE Mesh automatically calls the direct connection module to discover directly connected nodes and interacts with them to transmit network information and notify the multi-hop networking module. Then the multi-hop module updates network information and informs network topology changes to the application layer.

• Message sending. After the user sends application-dependent messages to a specific target node, the multi-hop networking module calculates the routing path to the next hop node based on the routing policy discussed in Section 3.4, and then forwards the message to the next hop node.

• Message receiving. When the direct connection module receives the message, it reports the event to the multi-hop network module, which then judges whether the destination node of the message is the local node itself. If yes, report the message to the application layer for further processing; if not, the message is forwarded to the next hop in the route path.

With the automatic procedures implemented, BLE

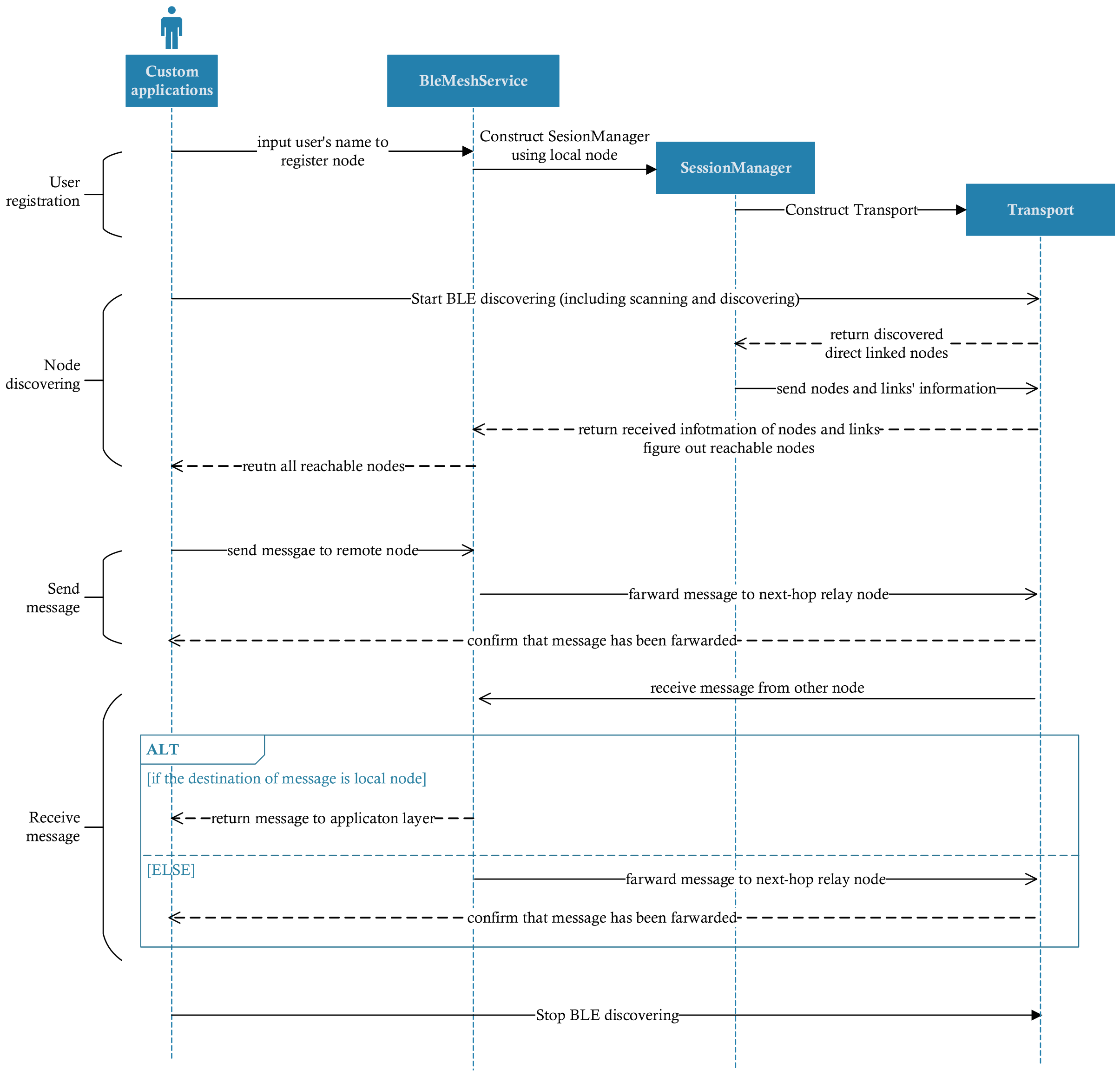

Custom applications
BleMeshService
SessionManager
Transport
User registration
input user's name to register node
Construct SesionManager using local node
Construct Transport
Node discovering
Start BLE discovering (including scanning and discovering)
return discovered direct linked nodes
send nodes and links' information
return received infotmation of nodes and links
figure out reachable nodes
reutn all reachable nodes
Send message
send messgae to remote node
farward message to next-hop relay node
confirm that message has been farwarded
Receive message
receive message from other node
ALT
[if the destination of message is local node]
return message to applicaton layer
[ELSE]
farward message to next-hop relay node
confirm that message has been farwarded
Stop BLE discovering

**Fig. 4 Workflow of main services provided by BLE Mesh: user registration, node discovery, and message sending and receiving.**

Mesh makes the main functions of multi-hop mesh networking easy to configure and use. Applications only need to set a nickname, start discovery, and send application related data to a selected destination, and then other configurations are entirely done by BLE Mesh.

## 3.4 Path selection with link-quality assistance

High node mobility may cause the routes to break frequently and force the source to switch between routes (if available) or rediscover new routes, thereby increasing both packet losses and latency. Basically, hop count is used as a routing metric in most routing schemes. However, the shortest-hop routes may contain weak links that are likely to break due to the change of link quality caused by complex physical environment and user mobility.

In BLE Mesh, we designed and implemented a link-quality assistant path selection mechanism, in which the Received Signal Strength Indicator (RSSI) along with hop count is combined to select a more stable and efficient route. Intuitively, the larger a link's RSSI value is, the better the link quality level is. We set up an RSSI threshold value, for example, $-90$ dB, and a path with link quality worse than $-90$ dB will be processed separately. We assume that $R$ is the original multi-hop path to a destination node, and $R'$ is another optional new path. $Q$ ($Q'$) is the link quality sequence of the path $R$ ($R'$), which is sorted in ascending order, that is, $Q[i]$ ($Q'[i]$) represents the link quality level of the $i$-th worst link in $R$ ($R'$), and then $Q[1]$ ($Q'[1]$) is the worst quality level in path $R$ ($R'$).

The following simple rules are used to compare and select between the two paths $R$ and $R'$.

• First, we determine whether the worst link quality is less than the RSSI threshold, that is, if $Q[1]$ is less than $-90$ dB, and $Q'[1]$ is larger than the threshold, $R$ can be considered unstable and $R'$ is selected to replace $R$. The rationality lies in that although the hop count of $R'$ might be larger, each hop is reliable.

• When $Q[1]$ and $Q'[1]$ are both greater than the threshold, we simply select the path based on hop count, since none of these two path are reliable enough.

• When $Q[1]$ and $Q'[1]$ are both less than the threshold, the path with the smaller hop count is selected to improve efficiency. If the hop count is the same, then the link quality level sequences of two paths are recursively compared: starting with the worst link ($i$=1), if $Q'[i]$ is less than $Q[i]$, the new path $R'$ is to be chosen; if $Q[i]$ and $Q'[i]$ are equal, the $Q[i+1]$ and $Q'$ $[i+1]$ will be compared.

## 4 Usage of BLE Mesh Framework

The engineering structure is shown in Fig. 5. The BLE Mesh kit exists in the form of a library module, with the package name "com.blemesh.sdk". To develop applications using BLE Mesh, the first step is to import BLE Mesh library module and compile it. Once the dependencies are established, App developer can use all services provided by BLE Mesh for development.

To illustrate the convenience of the designed development kit and the efficiency of routing policy in the multi-hop mesh network, we have developed a public safety communications application on Android using the BLE Mesh framework, QuoteSendApp, which helps users spread emergency information when wide Internet access is not functional.

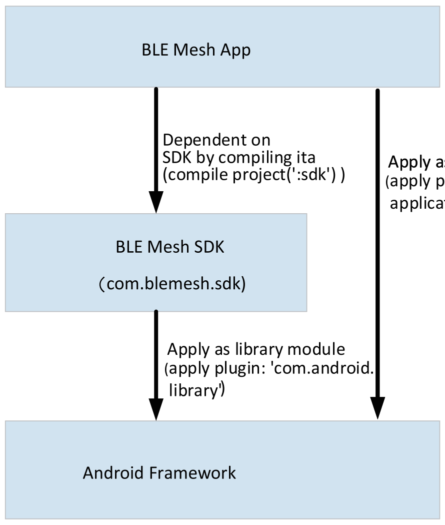

BLE Mesh App
Dependent on SDK by compiling ita (compile project(':sdk') )
Apply as application module (apply plugin: 'com.android. application')
BLE Mesh SDK (com.blemesh.sdk)
Apply as library module (apply plugin: 'com.android. library')
Android Framework

**Fig. 5 Engineering structure of BLE Mesh (applications compiling BLE Mesh SDK and then using its services).**

Based on the main functions provided by BLE Mesh, the use case of QuoteSendApp is shown in Fig. 6. We use three Android phones to test the quickly built app, including two Xiaomi 4C phones and one Huawei 4A phone. All devices maintain their native system and do not have root access.

Figure 7 shows typical snapshots of the QuoteSendApp in test devices Xiaomi.1, Xiaomi.2, and Huawei.1. After the process of peer discovering, the left device "Xiaomi.1"

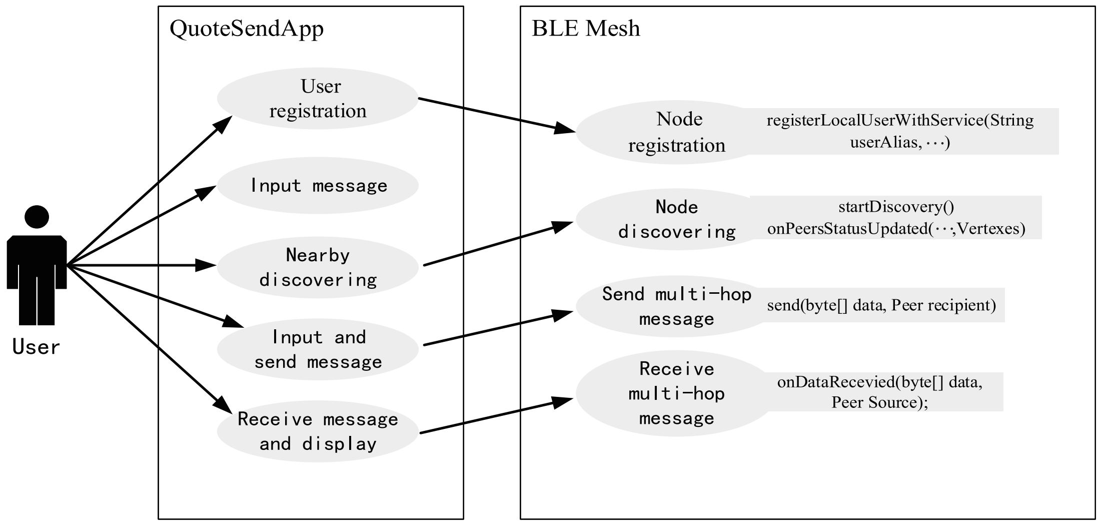
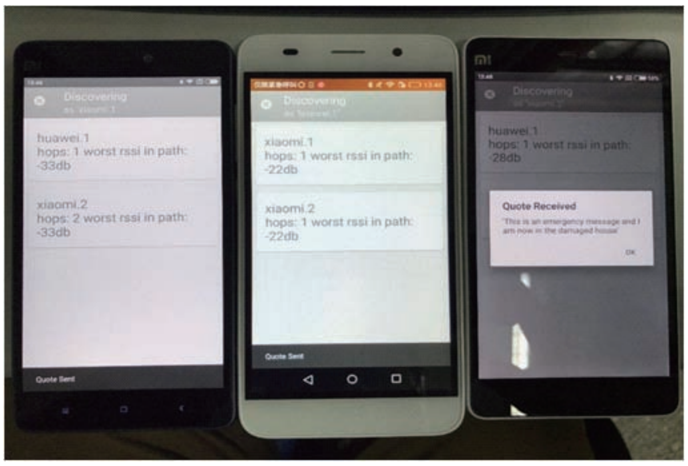

QuoteSendApp
BLE Mesh
User
User registration
Input message
Nearby discovering
Input and send message
Receive message and display
Node registration
registerLocalUserWithService(String userAlias, ···)
Node discovering
startDiscovery()
onPeersStatusUpdated(···,Vertexes)
Send multi-hop message
send(byte[] data, Peer recipient)
Receive multi-hop message
onDataRecevied(byte[] data, Peer Source);

**Fig. 6 Use cases of QuoteSendApp.**



huawei.1
hops: 1 worst rssi in path:
-33db

xiaomi.2
hops: 2 worst rssi in path:
-33db

xiaomi.1
hops: 1 worst rssi in path:
-22db

xiaomi.2
hops: 1 worst rssi in path:
-22db

huawei.1
hops: 1 worst rssi in path:
-28db

Quote Received
'This is an emergency message and I am now in the damaged house'

**Fig. 7 Snapshots of QuoteSendApp: Xiaomi.1 (left) sending information to Xioami.2 (right) via the relay device Huawei.1.**

spreads information to the right device Xiaomi.2 via Huawei.1 as the relay node. The snapshot show that the quote message from Xiaomi.1, as relayed by Huawei.1, has been successfully forwarded to the next hop, and the destination Xiaomi.2 has received the message and displays it on the screen.

## 5 Conclusion

In this paper, we have created a multi-hop mesh development framework called BLE Mesh, which has several distinctive features. First, the designed mesh networking scheme does not require breaching the software restrictions on modern mobile operation systems, and the scheme can be directly developed and deployed on commercially available off-the-shelf mobile terminals without rooting a device to access advanced capabilities, thereby making this scheme convenient for developers and end users. Second, the BLE Mesh framework is easily reusable and beneficial for various stakeholders such as app developers, framework maintainers, and end users in ProSe field. Furthermore, the BLE Mesh development kit can be used to quickly build a public safety communications app called QuoteSendApp, which distributes emergency information in close areas without Internet access. The development process illustrates the easy usability of the BLE Mesh framework.

Note that BLE Mesh is an open framework that utilizes the unlicensed D2D technology BLE as the underlying communications infrastructure. While open architectures with dynamically recruitable resources can bring up significant security and privacy risks, they can also improve system efficiency (through resource sharing), resilient (through dynamic reconfiguration leveraging redundant resources), and capability, thereby enabling abundant applications, especially for public safety communication scenarios in which wide cellular infrastructure may not be functional.

## Acknowledgment

The study was supported by the National Natural Science Foundation of China (No. 61171092), Jiangsu Educational Bureau Project (No. 14KJA510004), and NUPTSFs (Nos. NY215177 and NY217089)

## References

[1] Y. F. Wang, L. Wei, A. V. Vasilakos, and Q. Jin, Device-to-Device based mobile social networking in proximity (MSNP) on smartphones: Framework, challenges and prototype, *Future Gen. Comput. Syst.*, vol. 74, pp. 241–253, 2017.

[2] A. Asadi, Q. Wang, and V. Mancuso, A survey on device-to-device communication in cellular networks, *IEEE Commun. Surv. Tut.*, vol. 16, no. 4, pp. 1801–1819, 2014.

[3] S. Y. Lien, C. C. Chien, F. M. Tseng, and T. C. Ho, 3GPP device-to-device communications for beyond 4G cellular networks, *IEEE Commun. Mag.*, vol. 54, no. 3, pp. 29–35, 2016.

[4] P. Gardner-Stephen, The serval project: Practical wireless ad-hoc mobile telecommunications, http://developer.servalproject.org/files/CWN_Chapter_Serval.pdf, 2011.

[5] H. Wirtz, T. Heer, R. Backhaus, and K. Wehrle, Establishing mobile ad-hoc networks in 802.11 infrastructure mode, in *Proc. 6th ACM Workshop on Challenged Networks*, Las Vegas, NV, USA, 2011, pp. 49–52.

[6] Y. F. Duan, C. Borgiattino, C. Casetti, C. F. Chiasserini, P. Giaccone, M. Ricca, F. Malabocchia, and M. Turolla, Wi-Fi direct multi-group data dissemination for public safety, in *Proc. World Telecommunications Congress 2014*, Berlin, Germany, 2014, pp. 1–6.

[7] C. Casetti, C. F. Chiasserini, L. C. Pelle, C. Del Valle, Y. F. Duan, and P. Giaccone, Content-centric routing in Wi-Fi direct multi-group networks, in *Proc. IEEE 16th Int. Symp. A World of Wireless, Mobile and Multimedia Networks*, Boston, MA, USA, 2015, pp. 1–9.

[8] K. C. Liu, W. L. Shen, B. Yin, X. H. Cao, L. X. Cai, and Y. Cheng, Development of mobile Ad-hoc networks over Wi-Fi direct with off-the-shelf Android phones, in *Proc. 2016 IEEE Int. Conf. Communications*, Kuala Lumpur, Malaysia, 2016, pp. 1–6.

[9] Y. F. Wang, J. Tang, Q. Jin, and J. H. Ma, BWMesh: A multi-hop connectivity framework on Android for proximity service, in *Proc. 12th Int. Conf. Ubiquitous Intelligence and Computing and 12th Int. Conf. Autonomic and Trusted Computing and 15th Int. Conf. Scalable Computing and Communications and Its Associated Workshops*, Beijing, China, 2015, pp. 278–283.

[10] K. E. Persson, E. Manivannan, and M. Singhal, Bluetooth scatternets: Criteria, models and classification, *Ad Hoc Netw*., vol. 3, no. 6, pp. 777–794, 2005.

[11] K. H. Chang, Bluetooth: A viable solution for IoT? *IEEE Wirel. Commun*., vol. 21, no. 6, pp. 6–7, 2014.

[12] S. M. Darroudi and C. Gomez, Bluetooth low Energy mesh networks: A survey, *Sensors*, vol. 17, no. 7, p. E1467, 2017.

[13] S. Sirur, P. Juturu, and H. P. Gupta, P. R. Serikar, Y. K. Reddy, S. Barak, and B. Kim, A mesh network for mobile devices using Bluetooth low energy, in *Proc. 2015 IEEE Sensors*, Busan, South Korea, 2015, pp. 1–4.

[14] Z. L. Guo, I. G. Harris, L. F. Tsaur, and X. B. Chen, An on-demand scatternet formation and multi-hop routing protocol for BLE-based wireless sensor networks, in *Proc. 2015 IEEE Wireless Communications and Networking Conf.*, New Orleans, LA, USA, 2015, pp. 1590–1595.

[15] P. Gardner-Stephen, R. Challans, J. Lakeman, A. Bettison, D. Gardner-Stephen, and M. Lloyd, The serval mesh: A platform for resilient communications in disaster & crisis, in *Proc. 2013 IEEE Global Humanitarian Technology Conf.*, San Jose, CA, USA, 2013, pp. 162–166.

[16] J. L. Salmon and R. Yang, A proximity-based framework for mobile services, in *Proc. 2014 IEEE Int. Conf. Mobile Services*, Anchorage, AK, USA, 2014, pp. 124–131.

[17] M. E. F. Maia, R. M. C. Andrade, C. A. B. De Queiroz Filho, R. B. Braga, S. Aguiar, B. G. Mateus, R. Nogueira, and F. Toorn, USABle—A communication framework for ubiquitous systems, in *Proc. 28th Int. Conf. Advanced Information Networking and Appl.*, Victoria, Canada, 2014, pp. 81–88.

**Bo Zhang** is a senior experimentalist in School of Natural Sciences, Nanjing University of Posts and Telecommunications (NUPT), China. She received the master degree from NUPT, China, in 2013. Her research interests are complex networks and theoretical physics.

**Qun Jin** is a tenured professor at the Networked Information Systems Laboratory, Department of Human Informatics and Cognitive Sciences, Faculty of Human Sciences, Waseda University, Japan. He has been engaged extensively in research work in computer science, information systems, and social and human informatics. He seeks to exploit the rich interdependence between theory and practice in his work with interdisciplinary and integrated approaches. Dr. Jin has published more than 150 refereed papers in the world-renowned academic journals and international conference proceedings. His recent research interests cover ubiquitous computing, human-centric computing, human-computer interaction, behavior and cognitive informatics, life logs and big data mining, user modeling, information search and recommendation, e-learning, e-health, and computing for well-being.

**Li Wei** received the BS and master degrees from Nanjing University of Post and Telecommunications (NUPT), in 2014 and 2017, respectively. His main research interests include mobile social networks and proximity service.

**A. V. Vasilakos** is currently a professor with Department of Computer Science, Electrical and Space Engineering, Luleå University of Technology, Sweden. He has authored or co-authored over 200 technical papers in major international journals and conferences. He is the author/coauthor of five books, 20 book chapters in the areas of communications. He served as general chair, TPC chair, and symposium chair for many international conferences. He served or is serving as an editor or/and guest editor for many technical journals, such as IEEE TSMC-Part B, IEEE TITB, IEEE TWC, IEEE Communications Magazine, ACM TAAS. He is the founding editor-in-chief of the journals: *International Journal of Adaptive and Autonomous Communications Systems*, *International Journal of Arts and Technology.* His research interests are intelligent systems, etc.

**Yufeng Wang** received the PhD degree from Beijing University of Posts and Telecommunications (BUPT), China, in 2004. Currently, he acts as a full professor in Nanjing University of Posts and Telecommunications, China. From 2008 to 2011, he was an expert researcher in National Institute of Information and Communications Technology (NICT), Japan. He is a guest researcher at Media Lab, Waseda University, Japan. His research interests focus on cyber-physical-social systems, mobile social networks, etc.