DEFCON: 개발자 친화적인 Wi-Fi Direct 통신을 위한 안드로이드 라이브러리 설계 및 구현

DEFCON: Design and Implementation of Android Library for Developer-Friendly Wi-Fi-Direct Communication

박기웅, 백승훈[1)]
Ki-Woong Park, Sung Hoon Baek

(300-716) 대전시 동구 대학로 62 대전대학교 해킹보안학과
woongbak@dju.kr
(367-805) 충북 괴산군 괴산읍 문무로 85 중원대학교 컴퓨터시스템공학과
shbaek@jwu.ac.kr

요 약

스마트폰 사용자 간 데이터 공유에 대한 필요성이 증가되고 있다. 스마트폰 사용자들은 HD 비디오, 이미지 데이터를 인근 사용자와 공유를 하는 경우가 많다. 하지만, 두 스마트폰 사용자 간 Wi-Fi Direct를 통한 소켓 기반의 1:1 통신 채널을 형성하기 위해서는, 개발자 측면에서는 네트워크 프로그래밍에 관한 이해가 요구되며, 사용자 측면에서도 장치 간 네트워크를 형성하기 위한 구동 절차가 요구된다. 본 논문에서는 DEFCON이라 명명된 개발자 친화적인 Wi-Fi Direct 통신을 위한 안드로이드 라이브러리를 설계 및 구현하였다. DEFCON은 기존 장치 간 소켓 통신 채널을 형성하기 위한 대부분의 절차를 자동화 시켜 개발자와 사용자로 하여금 장치 간 네트워크 구현을 위한 개발 코드 및 사용자 개입을 간소화 시켜주는 효과가 있다.

Abstract

There is an increasing trend among smartphone users to share contents. Smartphone users usually share large content such as high definition videos and images with other nearby phone users. However, establishing a socket connection between two smartphones does not only require the developer of an application to have a deep understanding of network programming, but also the user of mobile networking application to have the knowledge of how to connect to an existing infrastructure or setting up an peer-to-peer network. In this paper, we present a developer friendly system library for sharing rich content among nearby phone users, termed DEFCON. It automates most of the steps necessary to establish a peer-to-peer connection between two smartphones, thus it simplifies the development of peer-to-peer networking applications for smartphones.

※ 본 과제는 미래창조과학부 IT/SW 창의연구과정-기술개발형(과제번호:20130170, 과제명: 하둡 기반 통합보안시스템 개발)의 지원으로 수행되었음.

1) 교신저자

논문투고일자 : 2013. 11. 28 심사일자 : 2013. 12. 02 게재확정일자 : 2013. 12. 23

키워드: 와이파이 다이렉트, P2P 통신, 안드로이드, 모바일 디바이스
Keyword: Wi-Fi Direct, Peer-to-Peer Communication, Android, Mobile Devices

1. Introduction

The proliferation of smartphones has led to an increase in the use of content sharing high definition video and images with other nearby phone users [1]. This has led to a new class of applications that allow a user to push content directly from the convenience of the smartphones [2]. This trend is to make it easier for users to share interesting content with other nearby phone users.

However, establishing a socket connection between two mobile devices does not only require the developer of an application to have a deep understanding of network programming, but also the user of mobile networking application to have the knowledge of how to connect to an existing infrastructure or setting up an peer-to-peer network such as Wi-Fi direct [3]. For example, we consider the example of two smartphone users willing to establish a peer-to-peer connection between their mobile devices using wireless networking as shown in Fig. 1 The first user has to enable his wireless networking hardware, set it in ad-hoc mode and configure a service set identifier (SSID). At that point this user can tell the other user to connect to his network using the SSID. The second user then has to turn on his wireless networking hardware and also set it in ad-hoc mode and configure the SSID. When the connection between the two devices is established, the users have to agree on IP addresses to use for their devices. These steps require the user to have an understanding of configuring a network.

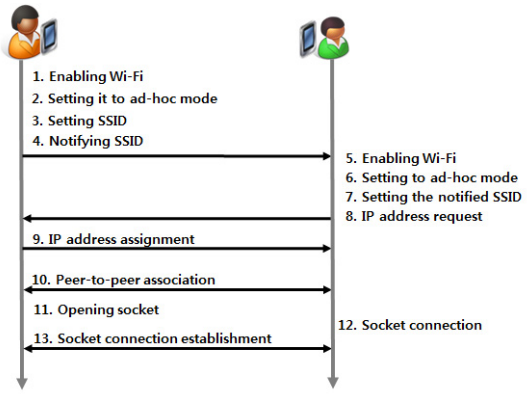

Fig. 1 One example of two smartphone users willing to establish a peer-to-peer connection

When all these steps are performed, the IP address of the two devices can be delivered to an application and the application can start to establish a connection on the application level. Still one of the devices has to be chosen as the server and one as the client. The server device has to open a socket and listening for an incoming connection. When the client is connecting to the server, the connection has to be accepted. At this point a connection is established on the application level and the devices can exchange data over the connection.

In this paper, we present DEFCON, a developer friendly system library for sharing rich content among nearby phone users. DEFCON automates most of the steps necessary to establish a peer-to-peer connection between two mobile devices, thus it simplifies the development of mobile networking applications.

To achieve it, a critical challenge need to be addressed to enable the developer friendly system library for sharing rich content among

nearby phone users. The communication between two devices should be intuitive and the communications view should match the users' view. For example, the user should not have to connect to a device using its network address or name. Rather it should be possible to specify the parameters of a connection by the distance range of one device relative to another device.

This paper makes the following two contributions: First, we presents a mechanism for mobile devices to share contents with other nearby phone users. Second, we have implemented a functional version of DEFCON on Android platform-based embedded devices.

The remainder of the paper is organized as follows: In Section 2, we present the overall system design and components of the proposed system. Section 3 describes our implementation issues. In Section 4, we evaluate the performance of the proposed system. Finally, in Section 5, we present our conclusions.

2. DEFCON Design

In this section, we present an overview of the developer-friendly Wi-Fi Direct communication library, termed DEFCON. We first introduce the role compositions between two mobile devices that are to connect directly. We then describe how each device can establish the peer-to-peer connections by describing the overall procedure.

2.1 Role Compositions for Establishing peer-to-peer Connection

A node on the basis of DEFCON is an Android platform implementing the DEFCON library and is thus able to establish a connection to other DEFCON nodes. Two roles have to be defined for each node. A node can either act as a master node or a slave node.

- **Master Node :** The master node is responsible for initializing the wireless ad-hoc network. This is done by broadcasting a specific DEFCON message over SSID, which indicates that this node is acting as a master so that the node is willing to accept connections from slave nodes. To achieve it, a predefined IP address has to be set for this node and a DHCP server has to be started. Then, connecting slave nodes can request an IP address. After the IP address is assigned and the DHCP server is started, the master node is opening a socket and is listening for a slave node to connect.

Table 1. Message specification on the basis of DEFCON

Field	Value		Size
Identifier	'D' 'E' 'F'		3 Bytes
Connection Mode	0: Sit-Together-Mode, 1: Public Key Encryption Mode		1 Byte
Message Type	0: DEFCON_DISCOVERY, 1: DEFCON_CONNECTION		1 Byte
Message Data	Message Type (0)	Flags: Capabilities of this node	1 Byte
		MAC address: MAC address of this node	6 Bytes
	Message Type (1)	Flags: Capabilities of this node	1 Byte
		Master MAC address: MAC address of the master node	6 Bytes
		Slave MAC address: MAC address of the slave node	6 Bytes
		Time stamp: Time stamp of the creation of this connection	8 Bytes

• **Slave Node** : The slave node can connect to a master node by using the same DEFCON message as the master node. Acting so, the node will connect to the same wireless ad-hoc network. Once the connection is established, it can request an IP address using the DHCP protocol [4]. Once the slave node receives the DHCP offer from the master node, it can configure its own IP address and connect to the open socket on the master node. At this point, the IP address of the master node is known by the slave node, since it was sent to the slave node in the DHCP offer.

2.2 Operation Mode Design for Secure Communication

One important building block of the above role composition for establishing peer-to-peer connection is the algorithm to predict whether two phones are within Wi-Fi range.

Since the position information is important for authorizing a connection, additional concepts have to be considered in order to prevent a accidental connection to a wrong node and to ensure the security of the connection. With the sit together mode and the public key encryption mode, two concepts are designed to solve this issue as shown in Fig. 2

• **Sit Together Mode** : In sit together mode, the two users willing to establish a connection are in close proximity to each other. As soon as the connection is established and the IP address configuration is completed, each node can derive one picture using shared hardware address from a image pool of DEFCON library. The two users then compare the picture each other. If the pictures match, the users can be certain that they have established a connection between their devices. However, this approach is vulnerable to man in the middle attacks [5].

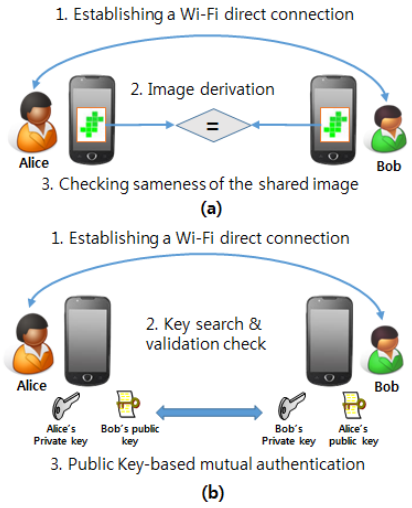

Fig. 2 Two modes of DEFCON: (a) Sit together mode, (b) Public key encryption mode

• **Public Key Encryption Mode** : The sit together mode does not prevent other users from listening to data being sent. To ensure data security, an asymmetric encryption system can be used. The public keys of the master and slave node are exchanged during the IP address configuration using the DHCP protocol. The DHCP protocol allows the embed user specific data in a DHCP request and in the DHCP offer using the so called vendor extension. However this does not authenticate a user completely since it is not verifiable if the public key corresponds to a specific user. If user authentication is required the public key should be retrieved using external infrastructure, e.g. downloading the public key from a key server from the internet. In this scenario only the fingerprints of the public key have to be shared using the DHCP protocol.

2.3 DEFCON Protocol for Peer-to-Peer Wi-Fi Connection to Nearby Phones

Detecting other mobile devices capable of handling a DEFCON connection is closely related to exchanging information between two or more devices prior to having established a connection. For establishing Wi-Fi connection in peer-to-peer manner, all networking devices in ad-hoc mode will broadcast information about their network [6].

On the basis of DEFCON, each mobile device exchange information by broadcasting and discovering predefined service set identifier (SSID). SSID is a case sensitive, 32 alphanumeric character unique identifier attached to the header of packets sent over a Wi-Fi. The SSID differentiates one Wi-Fi network from another, so all devices attempting to connect to a specific Wi-Fi network must use the same SSID to enable effective connection [7]. As part of the association process, a Wi-Fi client must have the same SSID as the one put in the access point or it will not be permitted to join the Wi-Fi network.

As described in Table 1, DEFCON nodes use the 32 bytes of the SSID to exchange information among each other. To ensure a reliable communication, a protocol is defined to exchange information. Each information is capsuled into a so called DEFCON message. A message starts with the three characters 'D' 'E' 'F', followed by the connection mode. The next byte is used for identifying the type of the DEFCON message, i.e. there can be 256 DEFCON messages. The remaining 27 bytes can be used to transmit data related to the type of the message.

- **DEFCON Discover Message :** The first message being defined is the *DEFCON_ DISCOVER* message. It is used to indicate to other DEFCON nodes. The nodes are implementing the DEFCON library and is currently available for establishing a connection. The first byte of the message data is used as a flag field to indicate the capabilities of this node. Capabilities like encryption support and the *sit_together_mode* are defined here. The next six bytes are used to transmit the hardware address of the wireless networking hardware. This is simply to prevent DEFCON nodes from establishing a connection at that point by using the same DEFCON message.
- **DEFCON Connection Message :** The section message is the *DEFCON_CONNECTION* message. It is used by the DEFCON master node to indicate to other DEFCON nodes which is acting as a master node and is waiting for nodes to connect. The first byte of the message data is again a flag field. The flags indicate the requirements for a slave node to connect to this master node. As for the *DEFCON_DISCOVER* message, no flags are defined. The next six bytes are the hardware address of the wireless network hardware used by the master node. With this information the master node in an ad-hoc network with more DEFCON node can easily be identified. The next six bytes are the hardware address of the wireless network hardware used by the client node. If this field is set to a non zero number, the DEFCON node matching this hardware address only should be able to connect to this master node. The last 8 bytes of the message are used for a time stamp. This is helpful if two nodes trying to establish a connection to each other and both nodes want to take the role of the master node. In this case the master node with the larger time stamp value, so created later in time, has to accept the other node as a master node.

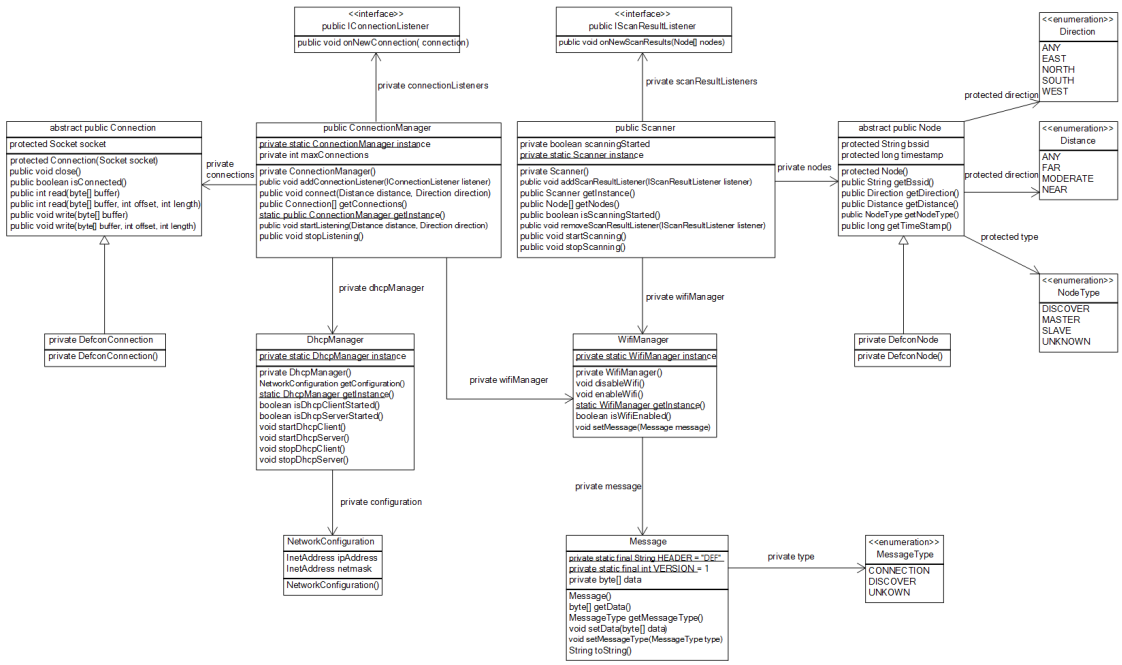

Fig. 3 UML diagram for the Java classes of the DEFCON library

3. Implementation

The implementation of the DEFCON library is done on the Android platform. The Android platform already provides the basic programs for implementing most features of the DEFCON library. With *dnsmasq* and *dhcpcd*, it provides a DHCP server and client respectively. The program *wpa_supplicant* can be used to set the wireless networking hardware into ad-hoc mode and send the DEFCON messages. However, not all functionality required by the DEFCON library is exported through the Android Java API. For this reason some parts of the DEFCON library have to be implemented in a native C/C++ library. This library can be accessed using the Java Native Interface (JNI).

Fig. 3 shows a UML diagram for the Java classes of the DEFCON library. The classes sketched out by the UML diagram are implemented in Java. The developer using this library should mainly work with the four classes *Connection*, *ConnectionManager*, *Node* and *Scanner*.

The class *Connection* represents an established connection between two DEFCON nodes. It can be used to send and receive data and close the connection to the remote node. The class *ConnectionManager* is to be used to establish a connection to another node. It is used by both the master node to listen for new nodes to connect and the slave node to connect to a master node. The interface *IConnectionListener* should be implemented by the application being developed to get notified from the library when a new DEFCON node connected to the master node.

The class *Node* represents a discovered DEFCON node. It contains the required information about a similarity of Wi-Fi signals and whether this node is already connected to another node or still available for a connection.

The class *Scanner* should be used by the application using the library to start and stop the scanning for DEFCON nodes in the proximity. The interface *IScanResultListener* should be implemented by the application in order to get notified by the library when new nodes are found during a scan.

The classes *DhcpManager* and *Wi-FiManager* should only be used by the library itself. They are responsible for starting and stopping the DHCP client and server and for enabling and disabling the wireless networking hardware respectively.

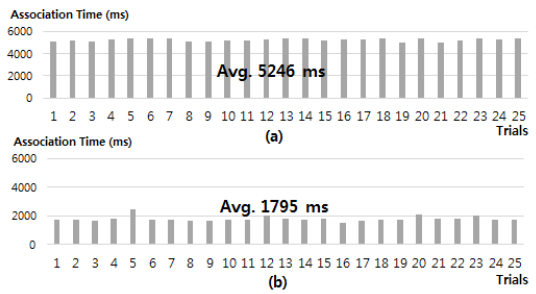

Fig. 4 The average association time per peer: (a) Wi-Fi direct connection by humans' configuration (b) DEFCON

4. Test-bed and Experiment

In this section, we present the performance results obtained with our prototype implementation of DEFCON. First, we demonstrate the overall experimental environment. We then describe the association latency in comparison with the conventional manner.

4.1 Test-bed Description

The wireless ad hoc network deployment that we used to test and experiment our library for content sharing consists of two mobile devices including a embedded board and one smartphone. Each device is equipped with an IEEE 802.11n wireless card. The ad hoc connectivity is maintained thanks to DEFCON demons run by the each device. All devices are supposed to implement DEFCON and to participate to the sharing of the file.

4.2 Experimentation Result

The metric tracked during our experiments was the association time between the two mobile devices, which is one of the most important metrics from the user's viewpoint. Two versions of peer-to-peer connection were tested over the settings described earlier in this section. The first version is the classical peer-to-peer connection, which require the users' interventions like turning on Wi-Fi card, mode setting, IP address setting etc. The second one is DEFCON-enhanced peer-to-peer connection.

Fig. 4 plots the average association time per peer. Each point is computed by averaging the association time over 25 trials. The figure shows that the classical peer-to-peer connection time engenders two to three times longer than DEFCON. This is because that DEFCON automates most of the steps necessary to establish a peer-to-peer connection between two mobile devices.

5. Conclusion

In this paper, we present a developer friendly system library for sharing rich content among nearby phone users, termed DEFCON. It automates most of the steps necessary to establish a peer-to-peer connection between two mobile devices, thus it simplifies the development of mobile networking applications.

The benefits of DEFCON library are not limited to smartphones. The developed system library also can be integrated into several mobile computing platforms like a car navigation or multimedia devices for easy data sharing or update their firmware.

참고문헌

[1] M.K.Sbai, C.Barakat, J.Choi, A.Al Hamra, T.Turletti, "Adapting BitTorrent to wireless ad hoc networks" in proceedings of Ad-Hoc Now 2008

[2] S. Gaonkar, J. Li, R. R. Choudhury, L. Cox, and A. Schmidt. "Micro-Blog: Sharing and Querying Content through Mobile Phones and Social Participation", In 6th MobiSys, 2008.

[3] Official Homepage of the IEEE 802.11 Working Group, http://grouper.ieee.org/groups/802/11/.

[4] I. Papapanagiotou, E. M. Nahum, and V. Pappas, "Configuring DHCP leases in the smartphone era", In Proceedings of IMC '12, ACM, NY, USA, 365-370.

[5] Hiroaki Anada and Seiko Arita. 2010. Identification schemes of proofs of ability secure against concurrent man-in-the-middle attacks. In Proceedings of ProvSec'10, Springer-Verlag, Berlin, Heidelberg, 18-34.

[6] M. Bisignano. An infrastructure-less peer-to-peer framework for mobile hand-held devices. European Transactions on Telecommunications, 2004.

[7] M. Abusubaih, S. Eddin, and A. Khamayseh, "IEEE 802.11n dual band access points for boosting the performance of heterogeneous WiFi networks", In Proceedings of PM2HW2N '13. ACM, NY, USA, 1-4.

저자소개

◆ 박 기 웅

- 2005년 연세대학교 Computer Science 학사
- 2007년 KAIST Electrical Engineering 석사
- 2012년 KAIST Electrical Engineering 박사
- 2008년 Microsoft Research Asia, Wireless and Networking Gourp, Research Intern
- 2009년 Microsoft Research Redmond, Network Research Group, Research Intern
- 2012년 국가보안기술연구소 연구원
- 2012년~현재 대전대학교 해킹보안학과 조교수
- 관심분야 : 시스템 보안, 모바일-클라우드 컴퓨팅, 보안 프로토콜, 디지털 포렌식 등

◆ 백 승 훈

- 1997년 경북대학교 Electrical Engineering 학사
- 1999년 KAIST Electrical Engineering 석사
- 1999년~2005년 한국전자통신연구원 컴퓨터시스템연구부 연구원
- 2008년 KAIST Electrical Engineering 박사
- 2008년~2011년 삼성전자 메모리사업부 책임연구원
- 2011년~현재 중원대학교 컴퓨터시스템공학과 조교수
- 관심분야 : 컴퓨터 저장장치, 운영체제, HCI, 가상 데스크톱, 시스템 보안, 클라우드 컴퓨팅