NFC Connection Handover Protocol: An Application Prototype

Saravanan Dhanabal

University of Tampere
School of Information Sciences
Computer Science
M.Sc. Thesis
Supervisor: Zheying Zhang
February 2015

University of Tampere
School of Information Sciences
Computer Science
Saravanan Dhanabal: NFC Connection Handover Protocol: An application prototype
M.Sc. Thesis, 55 pages
February 2015

Near field communication (NFC) is an radio frequency based wireless communication technology and it is widely being used in mobile devices. The NFC technology enables users to share data between devices and also supports other types of communication such as banking, ticketing, etc. This thesis work explores the interoperability between two NFC supported devices which are from different device vendors. NFC Forum is an non-profit industry association which promotes the standardization of the NFC technology. Connection handover is a method used in the NFC technology to connect the mobile and the multi-media devices quickly and seamlessly to exchange data. The NFC Forum has released a reference application protocol specification to ensure the interoperability between two NFC devices which are supporting the connection handover. As part of this thesis work, a prototype application has been implemented to test the interoperability by performing the connection handover, based on the connection handover protocol from the NFC Forum. The results demonstrate the ability to pair devices quickly and easily using NFC and also discuss the current situation of interoperability.

Keywords: NFC, Connection Handover protocol, Interoperability

Contents

1 Introduction

Near Field Communication (NFC) technology [1] is a proximity based secure wireless technology which supports the use of hand held devices to share data by by bringing them to close proximity. This technology operates in 13.56 MHz and requires around 10cm or less distance, and supports data transfer speeds 106/212/424kbit/sec. NFC technology is an evolution of Radio-frequency identification (RFID) [2] and it provides the possibility of two way communication between two NFC enabled devices. RFID supports only reading un-powered contact-less cards, which are also called tags.

Various international standards organizations including International Organization for Standardization - International Electro technical Commission (ISO/IEC), European Telecommunications Standards Institute (ETSI), NFC Forum [3] involve in defining the standards which specify the protocols to be used in this technology and physical characteristics of devices, etc.

To perform NFC based communication, the setup needs a pair of two NFC devices such as an NFC Reader/Writer device and any NFC Tag or another NFC Reader/Writer in close proximity. An NFC Reader/Writer or NFC tag technology shall be integrated in hand-held devices, and nowadays many of the smart phones provide NFC support. NFC tags contain data with read or write support and they can be encoded and decoded. NFC Reader/Writer could be an embedded device which supports reading and writing of NFC tags or send or receive data with another NFC device which emulates itself as tag. Two NFC devices can also communicate and share data using peer to peer mode.

NFC standards are defined in NFC Forum Technical Specifications [4] which also provides compliance certifications for devices and tags. NFC Forum is a non-profit association which was founded by Nokia, Philips and Sony in 2004.

NFC is an open platform technology and it was approved by ISO/IEC standard and also by European Computer Manufacturers Association (ECMA). The standards from ISO/IEC and ECMA define the physical characteristics such as RF (Radio Frequency) modulation schemes, data transfer speed and frame formats. Along with these, NFC Forum has defined the data exchange format protocol called NFC Data Exchange Format (NDEF) which helps to store various data types like URLs.

NFC technology has its advantages and disadvantages over other wireless technologies like Wireless LAN and Bluetooth. In some scenarios, NFC technology might need to be used with other technologies to create a productive user interface and solve communication issues like data exchange speed without compromising usability and speed.

NFC forum defines Connection Handover Protocol [5] which intends improve user experience while using NFC with other technologies. The connection handover specification by NFC Forum defines the means for two NFC capable devices to communicate quickly with each other and negotiate about using an Alternative Carrier which suites better to perform data exchange. The Alternative Carrier could be Bluetooth or wireless technology. NFC technology enables the devices to pair with each other when they are brought to close proximity. After knowing each other, they can start using the other technology to perform data transfer. Thus, NFC close proximity feature helps quick association and due to its limited data transfer speed, it hands over the data exchange operation to the next carrier.

When the NFC capable devices belong to different vendors, it is necessary to have a common standard to communicate with each other. To address this issue and apply a standard to provide interoperability between different vendor specific devices, NFC Forum has released a reference application protocol as Connection Handover Protocol.

This thesis studies the NFC based connection handover which is being used in smart phones and other NFC supported devices like Bluetooth headsets or wireless printers. The aim is to analyze the connection handover protocol by developing an application prototype which implements the connection handover behavior based on the protocol to enable easy and quick pairing of Bluetooth based audio devices that support NFC. The study also intends to analyze the functionality to be more generic by adding support for other Wireless LAN based devices such as printers or wireless routers which support NFC, but it was limited due the hardware availability.

Implementing an NFC based application requires understanding the different layers of NFC at the protocol level. These protocols spread across NFC architecture including NFC Controller hardware (includes NFC Firmware which has physical level protocol implementations), NFC Middleware (mostly data exchange protocols), NFC Application (reference application protocols and end user applications). As part of this thesis work, the protocol level study has been done before implementing the application.

The study and prototype implementation aims to discuss the interoperability requirements in the NFC applications [6]. Interoperability is one of the subcharacteristics of functionality in software quality model which concerns the ability of a software component to interact with the other components [7]. The connection handover application implemented on a mobile device is tested with other target device, which is from a different vendor than the mobile device. The end user can expect the NFC on his mobile to work with different NFC target devices in a similar, consistent manner. This interoperable feature is analyzed with respect to the current state of the connection handover protocol from NFC Forum.

This thesis starts with the introduction of the NFC. The second chapter provides information about NFC basics and compares it with other similar technologies and also the NFC tag types are explained. NFC specific standards are explained. Chapter 3 explains the purpose of the thesis work by comparing the NFC with other similar technologies. Application scenario for prototype development and analysis is explained in this chapter. Overview of Bluetooth, WLAN and Android based application development [8] are also discussed in this chapter. Chapter 4 contains the prototype design and implementation details. The steps involved in the development of the application prototype are explained here. Chapter 5 contains the testing steps including specific test cases and results analysis. Chapter 6 further discusses the interoperability issues and concludes the thesis. Appendices section contains the source code examples from the application prototype.

2 NFC Fundamentals

2.1 Introduction to NFC

Near Field Communication (NFC) is a proximity based wireless connectivity technology, which is evolved from the existing Radio Frequency (RF) based contact-less identification technology. NFC based products help the customer devices to interact with each other in the aspect of helping to make the connectivity quickly, sharing information and enabling secured data exchange. They enhance the user experience specifically with the mobile devices.

NFC communication happens when two devices are brought together, within a 10 cm distance between them. A simple touch or waving can establish the connection between the devices. NFC devices are capable of reading and writing. Depending upon the use case, a NFC device could be restricted with either one of them or both capabilities. Apart from RF capable, micro controller based NFC hardware, NFC also involves control and data exchange protocols, that are regulated by standards organizations like ISO, ETSI and ECMA. Since the transmission happens in close proximity, NFC based communication is considered as secure by default.

NFC can be used in a variety of devices, from smartphones that enable payment and information transfer to multimedia devices such as digital cameras and printers where users can send a photo to print or to display in TV with just a touch. Its applications have been available in several areas [8] including smart posters, ticketing, personal computing, payments and banking [9].

In order to standardize and regulate the NFC, a group of companies including Nokia, Philips and Sony formed an alliance in early 2004 and created a non profit organization called NFC Forum. The NFC forum specifies the protocols in NFC based devices which helps to advance the use of RFID technology in consumer applications.

The NFC forum was later joined by many companies like Visa, MasterCard, Google, Samsung, Microsoft, etc. Currently the forum has more than 150 members and most of them are contributing and promoting the NFC technology.

2.2 NFC Operating modes

NFC is based on integrating the existing RFID technology to mobile based consumer devices. As a communication technology, NFC has its own advantages and disadvantages.

NFC has compatibility with the existing RFID technology. RFID was mostly used in access control and public transport ticketing, where RFID reader devices can discover, read and write the target devices. For example, an RFID reader device mounted on a bus can read and write into the Smart Cards which are being used as tickets by passengers.

NFC can replace the RFID, since it is capable of acting either as a card or a reader writer device. An NFC device is capable of emulating itself as an RFID tag which can have any data like ticket or payment information, or as RFID reader or writer which can communicate with the RFID tags and modify their memory to do the access control or ticketing.

Apart form card emulation and reader writer mode, NFC can also include a secure element which is basically a secure memory location that can be inside a mobile phone or inside the NFC controller itself. This provides improved security for critical applications like credit card information. The secure memory area can only be accessible by the NFC controller after authentication.

NFC forum also defines another mode for data exchange, called peer to peer mode (P2P) or NFCIP-1, defined in the ECMA-340 Standard [10]. This mode is used to transfer data between NFC enabled devices such as mobile phones. This is comparable to other technologies like Bluetooth and Wireless LAN, although the transfer mechanism is different and there are other limitations as discussed earlier. Though NFC provides similar functionality as Bluetooth or Wireless LAN, it can also complement those technologies. NFC can quickly establish connection between two devices and then handover data exchange operation to other technology, like Bluetooth.

NFCIP-2 is another peer to peer mode specification from the NFC Forum which defines how to automatically choose the correct operating mode in peer to peer connection. This standard is called ECMA-352 [11].

In P2P mode, the device which starts the communication is called an initiator and the other device is called a target. P2P can operate actively or passively. In an active mode, both the initiator and target devices generate their own RF field which is needed as a carrier to exchange data. In a passive mode, the initiator generates the carrier field and the target does not generate the field and relies on the field generated by the Initiator. The target uses load modulation, to communicate back to the target.

Passive P2P communication is used when the target counter part device does not have any capability to generate any field, such as tags which only has the memory. This feature is also being used as a way of power saving.

NFC enabled mobile phones can also save power when they just have to respond to another reader which wants to do data exchange or transaction such as a banking transaction. The NFC controller which is embedded on the mobile device can emulate itself as a passive tag and respond to the external NFC reader using the external RF field. It is also possible to consume minimal power and still act as a passive device without using the external RF field.

In short, NFC can operate in three different modes, as described in Table 1.

Reader / Writer mode	Allows the device to read and write data with another active or passive device
Card emulation mode	Allows the device to act as a smart card which is supposed to have the confidential or other user defined data.
Peer-to-peer mode	Allows two NFC devices to exchange data in link level mode. This mode is comparable to other technologies such as Bluetooth and Wireless LAN.

Table 1: NFC Operating modes.

2.3 NFC tag types

As mentioned above, initiation of NFC service needs two devices, i.e. a host device and a counterpart device to work with.

If the counterpart is not an NFC controller which is actually a reader and writer device and capable of card emulation, it could be a tag, also known as a smart card. Smart cards are already available in consumer electronics, such as credit cards which are mostly contact based. NFC technology is adding value to the banking industry by supporting contactless payments.

NFC tags can be defined as contactless smart cards supporting a certain data exchange and storage format, defined by the NFC Forum. The NFC forum defines four different tag types as type 1, type 2, type 3 and type 4, as shown in Table 2. Theses tag types differ in their memory size and other features [12].

Physical size of the NFC tag affects the performance, mostly due to the power that needs to be generated from the reader writer device for operation. Choosing the right tag type depends on the use case of an application. Also, the required memory storage is another factor.

NFC tags can be embedded in a large range of consumer devices, inside plastic covers, printers and bill boards. Tags do not need any line of sight for normal operation. RF field generated by active device antenna could be affected by various objects such as metal objects. So, the placement of tags in electronic devices need to be carefully considered during product design phase.

NFC tags can support different read and write modes, such as read only, read/write and one time programmable mode. Vendor data sheets define the physical and software characteristics of the NFC tags.

Tag type 1	This tag type is based on the NFC forum standard ISO14443 A [13]. These tags are manufactured by Innovision and are called Topaz. Memory capacity is around 96 bytes and these tags are preferred for low cost product solutions. These tags support 106kbit/s data rate.
Tag type 2	This type is also based on the NFC forum standard ISO14443 A which are from NXP semiconductors [30], previously Philips. Many variations are available such as Mifare classic and Mifare ultralight. These tags usually have less memory capacity than the Type 1. These tags support data rates such as 106/212/424kbit/s.
Tag type 3	This type supports higher memory and higher data rates, e.g. Sony Felica tag has 1 or 2 Kbytes of memory and operates in 212kbit/s data rate. The NFC Forum defines Felica tag type for this standard.
Tag type 4	Type 4 supports ISO14443 type A and type B standards. This standard also supports higher data rates ranging from 106kbit/s to 848kbit/s. Type 4 tags available from NXP semiconductors and some other manufacturers.

Table 2: NFC Tag types and characteristics.

2.4 NFC standards

NFC is an ecosystem which includes various parts of technology industry such as telecommunications, banking and security domain. Application areas include data sharing, secured key management, ticketing and payment. This widespread application area requires different vendors to agree and to regulate the way NFC technology being applied in the consumer electronic devices. The ecosystem should also provide means to privacy and security and requires the NFC technology to fulfill all the requirements from different players in the ecosystem. This is done by various standards organizations whose role is to provide these standards to the NFC ecosystem and maintain them by improving and updating the standards.

The standard organizations include NFC Forum, ETSI, ECMA and ISO/IEC.

2.4.1 Overview of NFC Forum standards

The NFC Forum is a non-profit organization sponsored by many companies in the IT industry such as Nokia, Sony, Philips, etc., who also develop products using NFC.

The NFC Forum aims at advancing the applications of NFC in mobile devices, consumer electronics and personal computing devices. It develops specifications which ensure security and interoperability for the NFC supported devices.

The NFC Forum is responsible for the following activities and it defines the upper layer protocols which are part of different NFC modes.

- Developing NFC specifications for devices that are device independent.
- Defining protocols for data exchange and service discovery.
- Providing certifications to the devices which require NFC compliance.
- Promoting NFC technology.

Figure 1 illustrates the NFC forum standards architecture. It shows the three operating modes of an NFC device, mentioned as the Peer-to-peer mode, the Reader/Writer mode and the NFC card emulation mode. It also shows the three layers, referred as the low level RF layer, the middle level NFC protocol layer and the application layer. The RF layer includes the RF level protocols such as ISO 18902, ISO 14443. The RF protocols specify the protocols to exchange the data in the RF level. The RF level data will be processed in the NFC controller, in the NFC Firmware. The middle level NFC protocol layer specifies protocols such as Logical link control protocol (LLCP), NDEF protocol, etc. These middle level protocols specify the data exchange format which can be used by the applications. The applications layer shall include the NFC applications or reference application protocols. The listed protocols are used by the the application layer of the NFC based implementation.

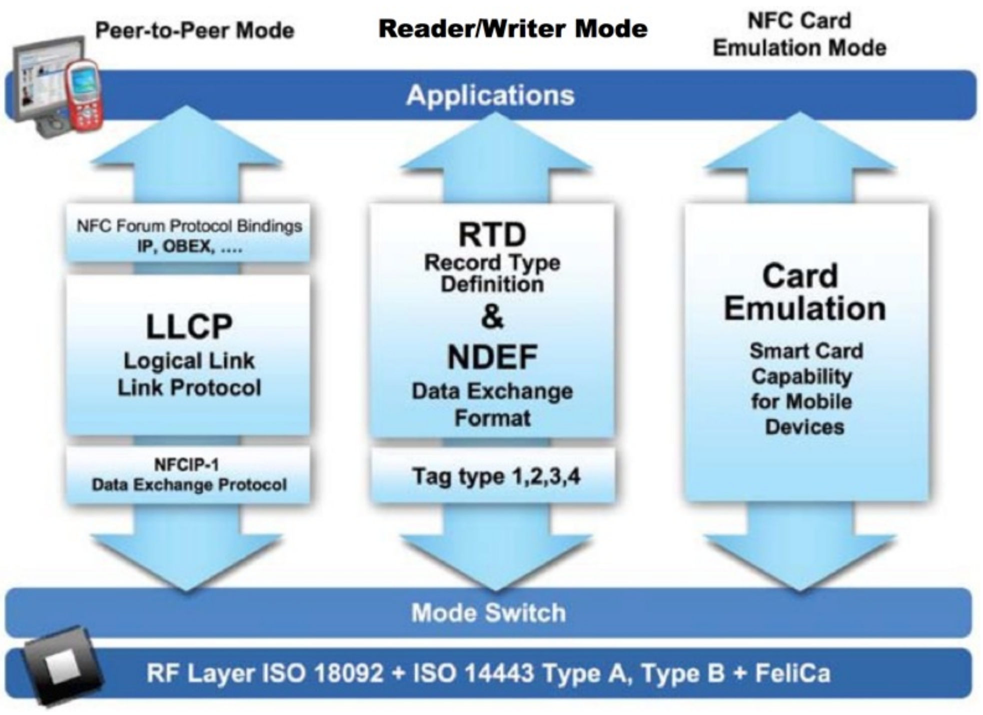
Peer-to-Peer Mode
Reader/Writer Mode
NFC Card Emulation Mode
Applications
NFC Forum Protocol Bindings IP, OBEX, ...
LLCP Logical Link Link Protocol
NFCIP-1 Data Exchange Protocol
RTD Record Type Definition & NDEF Data Exchange Format
Tag type 1,2,3,4
Card Emulation Smart Card Capability for Mobile Devices
Mode Switch
RF Layer ISO 18092 + ISO 14443 Type A, Type B + FeliCa

Figure 1: NFC standards architecture [14].

Figure 2 illustrates the NFC forum standards overview with respect to interoperability. Interoperability in NFC can be defined as a feature which enables seamless and spontaneous data sharing between different types of consumer electronics devices which are manufactured by different vendors [15]. Any NFC enabled device which has alternative carrier technologies could implement the connection handover in an interoperable way, so that it works with another NFC enabled multimedia device which could be from the same or from a different vendor.

Figure 2 shows the complete set of protocols which are used in implementing NFC. RF Analog protocols are at the lowest layer and these are usually implemented in NFC controller firmware. The layers above are usually implemented partially or completely in the host controller and it could include protocols such as Simple NDEF Exchange Protocol (SNEP), LLCP and Record type definition (RTD). Application documents are reference specifications which promote interoperability. NFC Connection handover protocol is a reference specification which belongs to the application documents group. Bluetooth secure simple pairing (SSP) using NFC is one such method used in connection handover.

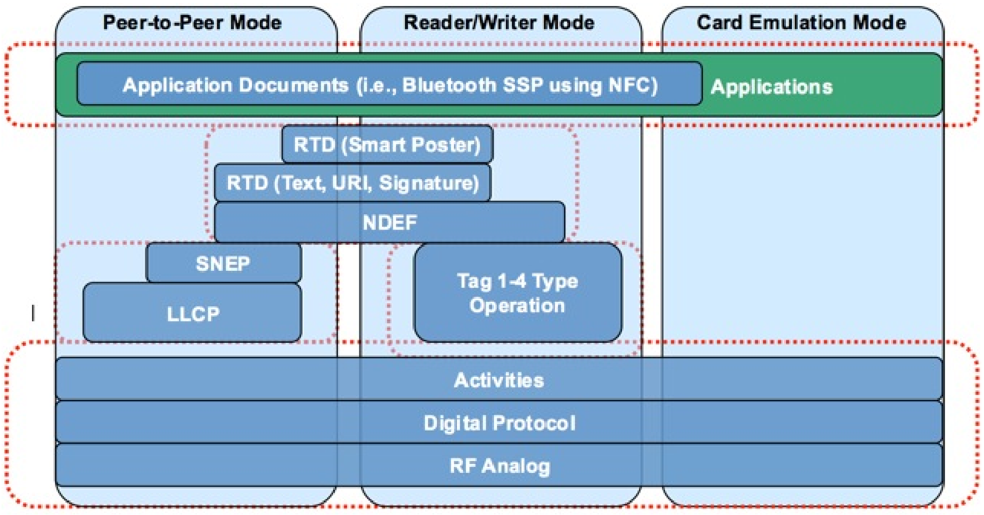

Figure 2: NFC standards architecture and interoperability [14].

NFC Forum maintains several NFC specific protocol specifications. The specifications which are related to connection handover are listed in the following sections.

- **NFC digital protocol technical specification**

  The NFC digital protocol technical specification [16] defines the digital protocol to be used for NFC enabled devices. It defines the digital interface and the transmission protocol for the NFC forum device in its four different roles as Initiator, Target, Reader/Writer and card emulator. The digital interface is about RF modulation schemes, bit level coding during transmission, data bit rates to be used and the protocol part defines about the frame formats and command sets that needed to be used for communication.

  The specification builds on top of ISO/IEC 18092 and ISO/IEC 14443 standards.

- **NFC activity technical specification**

  The NFC activity technical specification [17] specifies how the NFC digital specification can be used to setup communication with another NFC device or NFC tag in terms of building blocks called activities. Activities could be any sequence of operations to setup communication. Activities are combined in to a profile. Each profile can have a specific parameter set to achieve a specific use case. Example profile would be polling for another NFC device and establish peer to peer communication or polling for an NFC tag and read data. Both of these profiles require different configuration data.

- **NFC controller interface technical specification**
  The NFC controller interface technical specification (NCI) [18] defines the interface between the NFC controller and the host in which the NFC controller could be embedded, such as smartphone which has NFC controller attached. The smartphone host uses the NCI protocol format to interact with the NFC controller (NFCC). This enables the device manufacturers who want to integrate NFC in their products can have common interface to communicate with the NFCC. All the host devices such as smart phones, printers, consumer electronics, and other appliances can use the generic interface standard. NCI protocol provides a logical communication interface that can be used any physical transport. Currently I2C, SPI and UART are being supported as physical transport layer. Host processor will communicate with the NFCC using NCI through SPI/I2C/UART.

- **NFC data exchange format technical specifications**
  NFC data exchange format technical specifications (NDEF) [19] define a common data exchange format for NFC forum complaint devices and tags

- **NFC Forum tag type specifications**
  NFC Forum supports 4 tag types [12] to provide interoperability between NFC tag providers and NFC controller based solution providers. NFC Forum provides separate specification for each tag type and defines the physical characteristics of the tag and data formats.

  Table 3 lists the base standards from which the NFC Forum derived the tag type specifications.

Type 1 tag operation specification	Based on ISO/IEC 1443A, read/write capable, available memory 96 bytes, and expandable up to 2kbyte.
Type 2 tag operation specification	Based on ISO/IEC 1443A, read/write capable, available memory 48 bytes, and expandable up to 2 kilo bytes.
Type 3 tag operation specification	Based on JIS X6319-4 (Japanese industrial standard), read/write capable, memory limit is 1Mbyte, also called as Felica (from Sony).

Table 3: NFC Tag type base standards.

- **NFC connection handover specification**
  The NFC Connection Handover specification [5] defines the sequences of interaction and data structure between two NFC devices to setup communication using other wireless technologies. Connection handover defines to achieve

simple, one touch setup of NFC with other wireless technologies such as Bluetooth or wireless which support higher data transfer rates. The developers can choose the communication carrier for data exchange after negotiation with the counterpart device. Configuration data required for the connection setup can be stored and parsed using NFC data exchange format (NDEF). Static connection handover is also supported in which the available carrier configuration data will be stored in an NFC tag which will be embedded into the host device.

2.4.2 Overview of ECMA standards

- Near Field Communication Interface and Protocol (NFCIP-1) / ECMA-340 Near Field Communication Interface and Protocol specification [10] defines the interface for NFC using inductive coupled devices operating at 13,56 MHZ for interconnection of computer peripherals. It also defines the peer to peer protocol, specifically active and passive modes to realize a communication network using NFC. The standard also defines a transport protocol including protocol activation and data exchange methods. This standard is also approved as ISO/IEC 18092.

The above list of standards is not inclusive of all the available standards meant for NFC. Only the major protocol standards, that are necessary for this thesis work are highlighted. Detailed description of the protocols used in this thesis work is explained in Section 2.5.

2.5 Description of NFC protocols

2.5.1 NDEF protocol

The NDEF protocol specification [19] defines the message encapsulation format to exchange information between two NFC devices. An NDEF message will be in a binary format, that can be used to encapsulate one or more application defined payload data of arbitrary type and size in to a single message construct. Each payload contains the actual data, size and a message identifier. The identifier could be of any format including URIs, MIME media types, or NFC specific data types. It can also be a name space defined by vendor for their own NFC specific purposes.

Payload size is an unsigned integer, indicating the number of octets in the payload. The NDEF payload can contain single or chained data messages. Chained messages could be indicated using message header and combined later after transmission.

NDEF is only a message format and does not have any characteristics with respect to actual message transmission. The NDEF specification defines rules to construct valid messages as an ordered collection of NDEF records. The record types are specified in an inter-operable way and detailed record types are defined in separate specifications.

An example use case would be an NFC reader which reads the website URI data, stored in NDEF format from another NFC device or NFC tag attached in a notice board. Both the NFC tag and the NFC device will have the URI stored in the same NDEF format.

A typical NDEF record format is shown in Figure 3. An NDEF message can be a group of NDEF records and each NDEF record has a header and payload. Header is further split into 3 bytes as identifier, length and type. Identifier represents the NDEF message type, length represents the payload size.

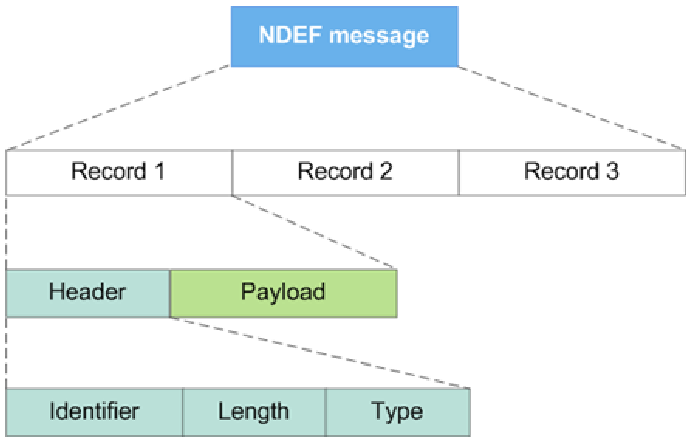

Figure 3: NDEF record overview [19].

Type byte in the header is explained in Figure 4. Each bit is used to define the NDEF message status such as first or last message, chaining status, etc.

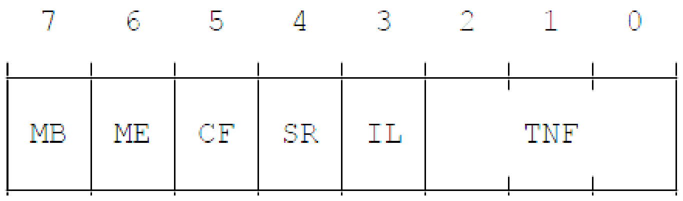

Figure 4: NDEF header format [19]

Each bit in the Type byte is explained in Table 4.

MB	Message Beginning, indicates whether this record is the first one or not in the NDEF message.
ME	Message End, indicates whether this record is end of the NDEF message.
CF	Indicates chained status of the NDEF message.
SR	Short Record. If set, indicates that the payload length field is single octet.
IL	If set, indicates that the ID_LENGTH field is presented as a single octet. Else, indicates that no ID_LENGTH from header and ID field from record will not be present.
TNF	Indicates the structure of the value of the type field

Table 4: NDEF Header bytes.

The NDEF full record format including the NDEF header is shown in Figure 5. Along with the NDEF header, the full record has the payload data and its length.

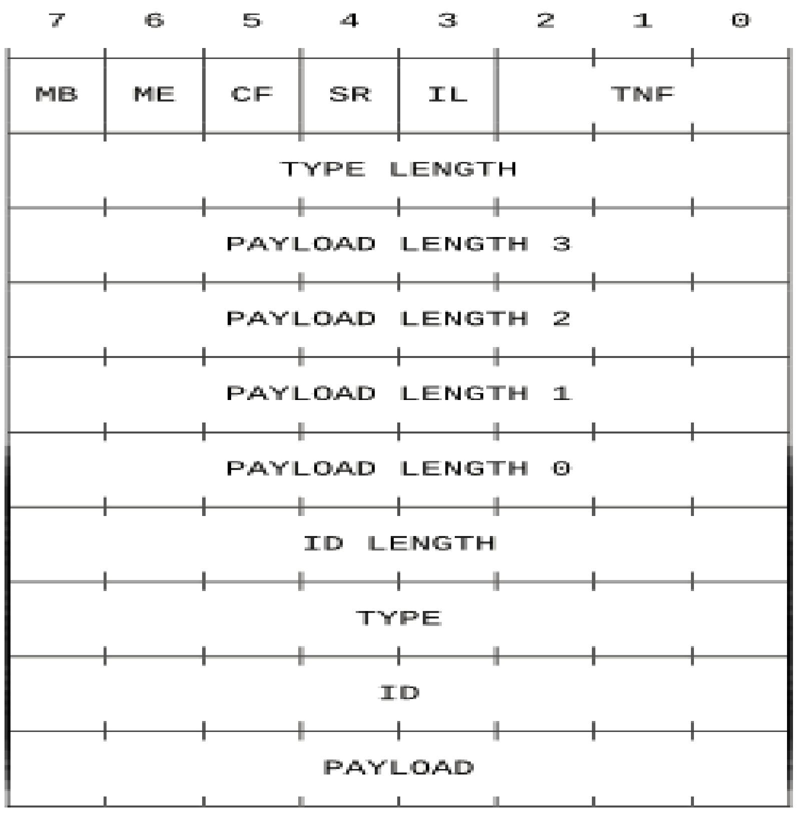

Figure 5: NDEF record format [19].

2.5.2 ISO14443A protocol standard

NFC Forum Type 1, Type 2 and Type 4 tags belong to the ISO14443 standard [13]. While the NFC Forum tag specifications mention how to exchange data with tags using NDEF in a tag specific way, the ISO 14443 standard addresses various low level characteristics such as physical, RF interface operations. The ISO1443 standard has four parts and these parts define the physical characteristics of NFC tags or proximity integrated circuit cards (PICCs), radio frequency power and signal balance, initialization and anticollision, transmission protocol. The transmission protocol part is being used in NFC based application development.

2.5.3 Connection Handover Protocol

As mentioned in the above sections, the Connection Handover Specification [5] defines a message structure to connect an NFC Forum device over another communication carrier. The Alternative Carrier will be chosen after negotiation and used for various kinds of data exchange between devices such as sending a photo to a printer or music streaming to a WLAN television or to an NFC supported wireless headphone.

An NDEF message will be used for negotiation initiated by a Handover Requester for alternative communication carrier with a Handover Selector over NFC link. The Handover Selector will also be able to retrieve the available alternative communication carrier(s) from the NFC tag. It remains passive until requested by the Handover Requester device and it will not start any activity by itself.

2.5.4 Negotiated Handover

A negotiated handover use case is described in Figure 6. A Handover Requester uses the NFC Forum Device to exchange connection handover information to select a matching alternative carrier. The Handover Requester announces it is supported alternative carriers to the Handover Selector and then receives the selected carrier information.

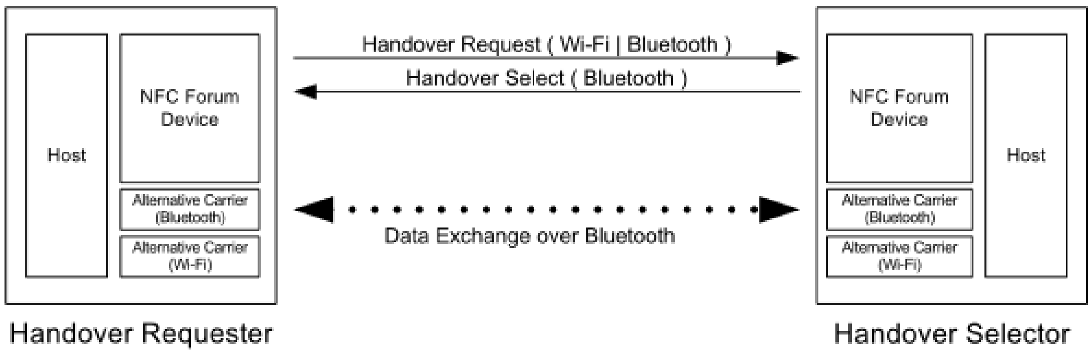
Handover Request (Wi-Fi | Bluetooth)
Handover Select (Bluetooth)
Host
NFC Forum Device
Alternative Carrier (Bluetooth)
Alternative Carrier (Wi-Fi)
Data Exchange over Bluetooth
Handover Requester
Handover Selector

Figure 6: Negotiated Handover Usecase [5].

If the Handover Selector returns more than one Alternative Carrier, the Handover Requester will have to find the order of the supported Carrier Configuration Data and it can choose the first preferred one. Handover Requester can also ignore the preferred one and select carrier, as shown in Figure 7.

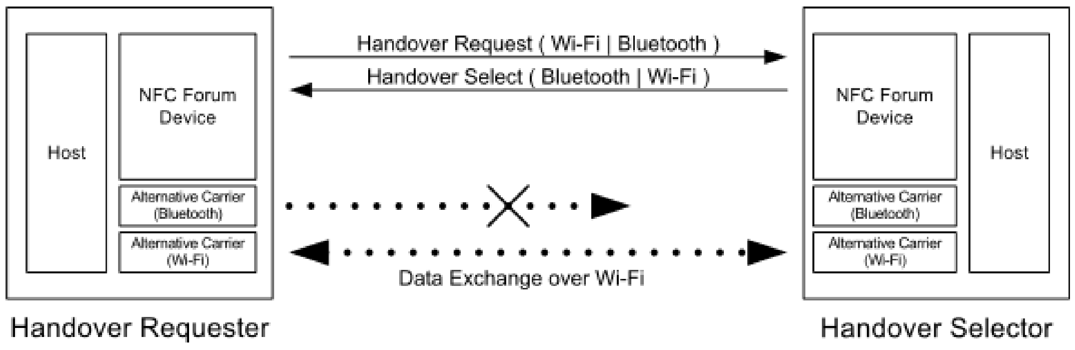
Handover Request (Wi-Fi | Bluetooth)
Handover Select (Bluetooth | Wi-Fi)
Host
NFC Forum Device
Alternative Carrier (Bluetooth)
Alternative Carrier (Wi-Fi)
Data Exchange over Wi-Fi
Handover Requester
Handover Selector

Figure 7: Negotiated Handover - selection of second alternative carrier [5].

In power constrained devices such as battery powered, the negotiation needs another set of message exchange. In low powered devices, the Handover Selector may not want to activate all available carriers. It returns the Handover Select message with Carrier Power State set to zero in the Alternative Carrier Records field. Handover Requester, after choosing the carrier, need to send the Handover Request message again with select carrier information. Upon receiving the selected carrier information, Handover Selector can activate the carrier and send back the

Handover Select message with chosen carrier information to complete the negotiation sequence. Figure 8 shows the handover selection on a power constrained device.

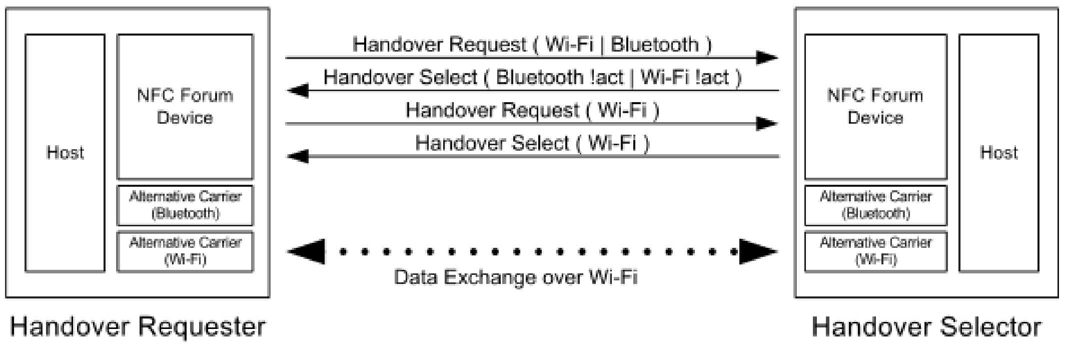
Host
NFC Forum Device
Alternative Carrier (Bluetooth)
Alternative Carrier (Wi-Fi)
Handover Request (Wi-Fi | Bluetooth)
Handover Select (Bluetooth !act | Wi-Fi !act)
Handover Request (Wi-Fi)
Handover Select (Wi-Fi)
Data Exchange over Wi-Fi
NFC Forum Device
Alternative Carrier (Bluetooth)
Alternative Carrier (Wi-Fi)
Host
Handover Requester
Handover Selector

Figure 8: Negotiated Handover - selection of alternative carrier on low power device [5].

Thus, the complete negotiation sequence includes two steps, as shown in Figure 9.

The Handover Selector device sends the Handover Select message, and gets the Handover Select response from Handover Selector with proposed carrier options which may specify the power state of the carrier. If power state is not mentioned, the sequence is already complete and the device pairing can be started using the chosen carrier. Else, it needs another Handover Request message from Handover Requester with chosen carrier which needs an acknowledgement from Handover Selector with chosen carrier information in which carrier power state should be set.

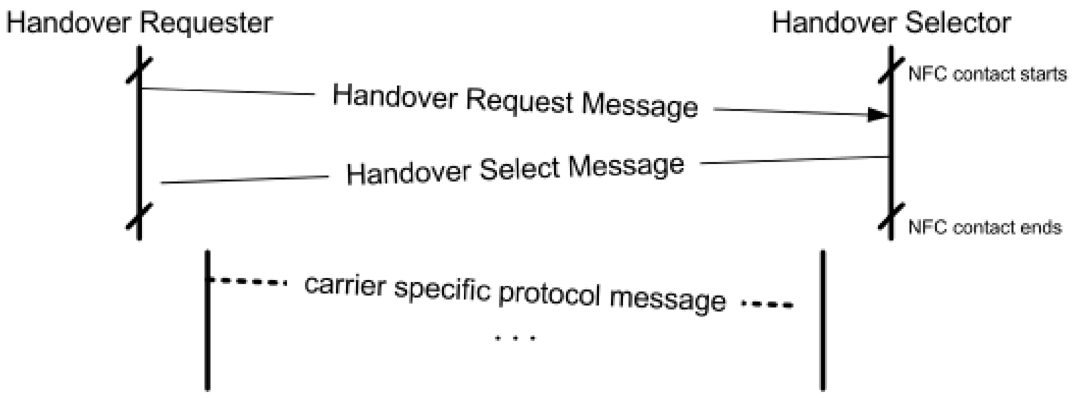

Figure 9: Negotiated Handover - complete sequence [5].

2.5.5 Static Handover

A Static Handover use case is described in Figure 10. It is used when the counterpart device is not an NFC Forum device but an NFC tag which only has storage, but no host controller. The NFC tags have no capability to power themselves and only provide storage where data could be written and read. Read only or one time programmable tags are possible. These tags meant to be cheaper solutions with a limited or no power capable environment. NFC tags could be embedded into the host device with no communication means. The NFC Requester device powers on the NFC tag using its RF field and reads the data on the tag which has the Carrier Configuration Data in the form of NDEF message. NFC tag will not be able to receive any Handover Request message or it can process the message to reply with a dynamic response.

The host device of the NFC tag (Handover Selector) will not be able to activate its alternative carrier as part of the handover process. The carriers need to be always active or need to be activated manually by the user before the handover.

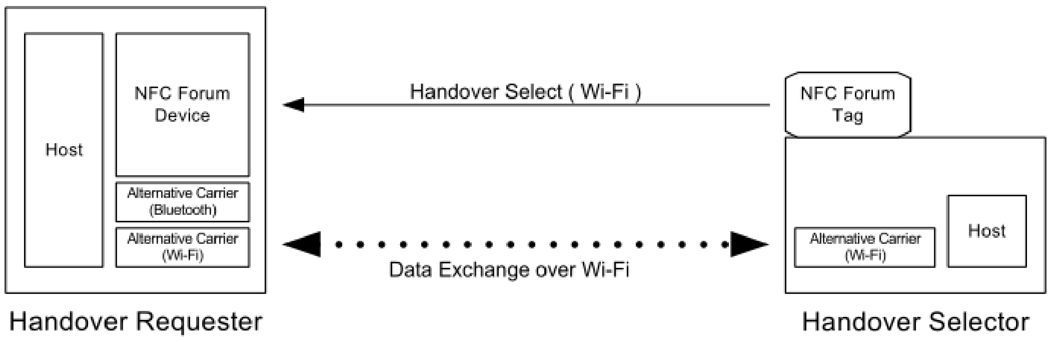
Host
NFC Forum Device
Alternative Carrier (Bluetooth)
Alternative Carrier (Wi-Fi)
Handover Select (Wi-Fi)
Data Exchange over Wi-Fi
NFC Forum Tag
Alternative Carrier (Wi-Fi)
Host
Handover Requester
Handover Selector

Figure 10: Static Handover usecase [5].

2.5.6 Message composition in Connection Handover Protocol

Handover message includes Handover Select Record or Handover Request Record followed by arbitrary number of NDEF records. NFC Forum global type for Handover Select Record is “Hs” and Handover Request Record is “Hr”. Handover Select Record or Handover Request Record can include sequence of Alternative Carrier

Records, referred by NFC Forum local type “ac”. Alternative Carrier Records provide references to the alternative carriers and data will be included in the NDEF records which will follow the Alternative Carrier Records.

The record references are based on the URI based Payload Identification mechanism as defined in the NDEF specification. Relative URIs shall be encoded with the virtual base defined as “urn:nfc:handover:”

Figure 11 shows the general structure of connection handover message composition. It is a collection of NDEF records and a single Handover Select or Request record.

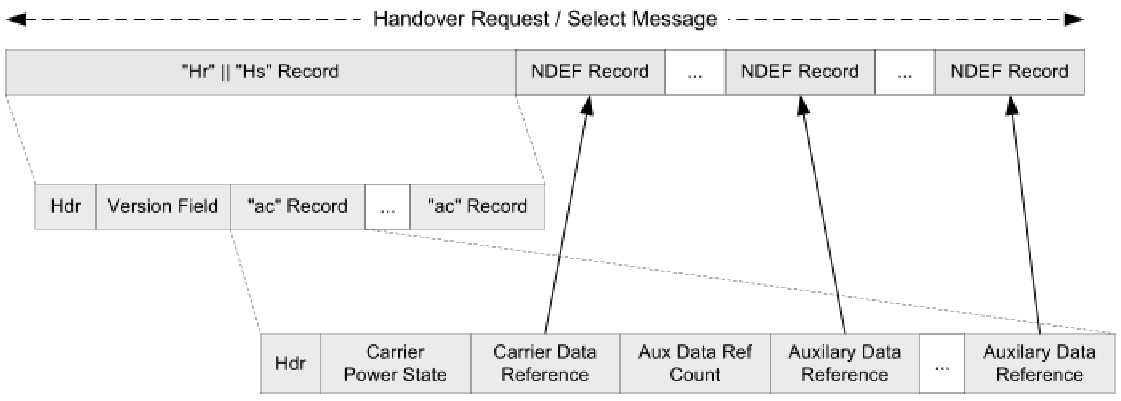

Figure 11: Message composition [5].

2.5.7 NDEF messages in Connection Handover Protocol

Figure 12 shows the format of Handover Request message. Handover Request record has the message beginning bit (MB) bit set and the last NDEF record has the message end (ME) bit set.

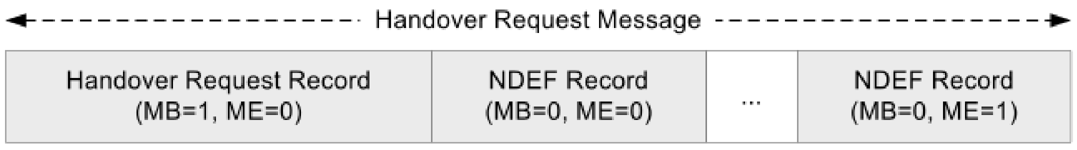

Figure 12: Handover Request Message structure [5].

Figure 13 shows the format of Handover Select message. Handover Select record has the message beginning (MB) bit set and the last NDEF record has the message end (ME) bit set.

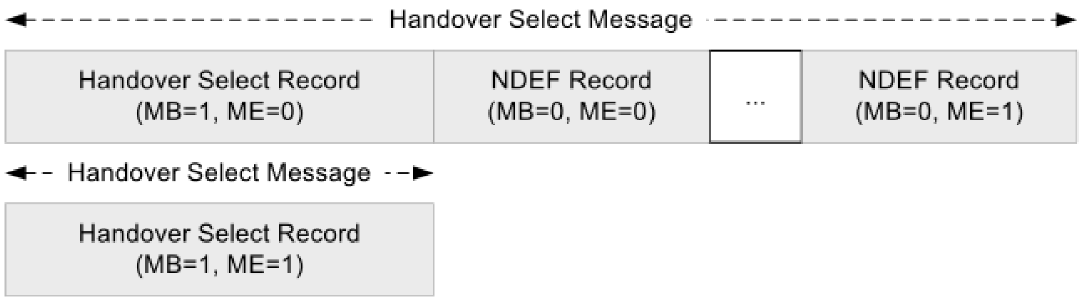

Figure 13: Handover Select Message [5].

Figure 14 shows the structure of a Handover Select Record. This record is a collection of Alternative Carrier records with the carrier configuration data from NFC target device.

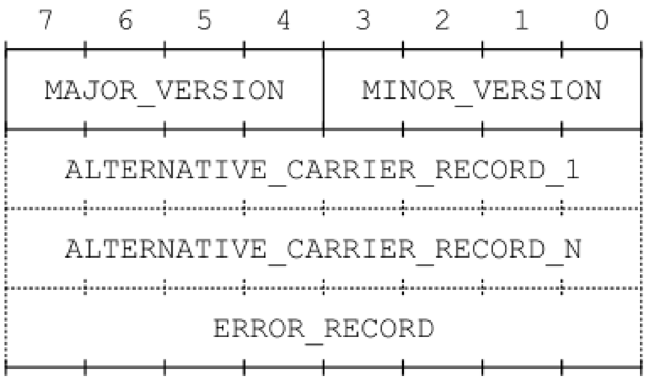

Figure 14: Handover Select Record [5].

Figure 15 shows the structure of a Handover Carrier Record. The record is used to hold a single carrier configuration data such as Bluetooth MAC address, local name, etc.

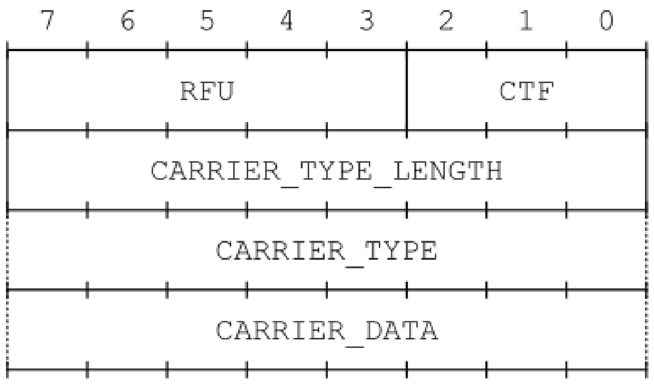

Figure 15: Handover Carrier Record [5].

2.5.8 Version Handling in connection handover

The version number field in the Handover Request Message shall be equal to the version number field in the Handover Select Message for interoperability. The message content is decoded after checking the version number. The version number is divided into a major version and a minor version. If the minor version is different, the message can still be decoded and it enables backward compatibility. If major version is different, the device should not try to decode further and the devices are considered as incompatible. A difference in the major version indicates that there could be difference in syntax itself and the messages can have different format and decoding procedures due to specification changes.

If the Handover Selector receives Handover Request Message with difference only in the minor number and if it has the higher value of minor version, it should return the Handover Select Select Message with its own version number.

If the Handover Selector device receives a Handover Request Message with difference in the major version number and if it is higher than its own version number, it should return an empty Handover Select Message without any alternative carrier records.

If the Handover Selector device receives a Handover Request Message with difference in the major version and lower than its version, it can reply with empty message or it can return the Handover Select Message with the version number it received in Handover Request Message.

3 Means for effective application of NFC technology

3.1 Application scenario

In mobile and portable devices, different technologies have been used to support data exchange. The end users choose the technology based on various factors. The most important ones include the amount of resources like power to be spent, the complexity level of setting up the communication between devices, the additional requirements of resources like physical communication links, etc. The most commonly used data exchange technologies include the Wireless and the Bluetooth technologies. Both technologies are based on radio frequency. NFC has advantages and disadvantages against both of them.

Table 5 shows the NFC characteristics, in comparison with other technologies. Values in the table are approximate, and to provide a comparative view. NFC is slower in terms of data transfer speed when comparing with Bluetooth and Wireless LAN. Normally it supports speeds like 212/424/848 kbit/s whereas Bluetooth supports 2.1 Mbit/s and Wireless LAN supports in terms of Mbit/s based on the band.

Operating Technology	**Data Rate**	**Operating distance**
NFC	< 1MB (maximum 848kb/s)	10 cm
Bluetooth	24 MB	60 meters
WiFi	> 100 MB	100 meters
WiMax	100 MB	10 kilometers

Table 5: NFC characteristics compared to other technologies.

If we have to transfer some data using the Bluetooth technology, before starting the transferring session, we have to search for the devices nearby and pair them by entering a PIN code for authorization. This requires certain amount of setup time. In the similar way, Wireless LAN also involves setup process before starting any data exchange. Apart from this setup delay, power consumption during data transfer should also be considered. Always we need this setup process, if we have to exchange data quickly with some new devices which were not paired before.

When using the NFC technology, the devices need to be in close proximity such as within 10cm range. The devices can be paired by simply 'touching' each other. Power consumption is negligible for this pairing process when compared to Bluetooth and WLAN. But the data transfer speed of NFC is very limited when comparing against those technologies.

NFC could be combined with either one of those technologies to achieve both simplicity of setup process as well as higher data transfer speed. User can simply print a picture from digital camera by touching its NFC enabled part to the NFC enabled printer. Another use case would be synchronizing new music files from home media server by touching the PC with the mobile device. In the above two use cases, when the user touches one device with another, pairing will be completed between devices with the help of NFC, which will take less than a second. Data transfer will be started using the next available technology, which could be either Bluetooth or WLAN. User can move the device after touching, and the transfer will be completed as long as user keeps the device within the range of the other technology. Bluetooth supports operating range up to 50 meters and WLAN depends on the operating frequency.

There are various multimedia devices available in the market which can be used with smart phones. Typical examples are wireless printer, Bluetooth head set, NFC based storage devices, etc. These multimedia devices and hand held devices are from different vendors. If all the vendors are supporting the connection handover protocol, any multimedia device could be seamlessly used with hand-held devices.

Interoperability provides the freedom to the end user to pair devices from different vendors. But it depends on the device manufacturer to support the connection handover protocol, which could be influenced by business reasons and security considerations.

The current state of the connection handover protocol, level of support from different device manufacturers, availability of specifications and other manuals, security considerations need to be analyzed.

Currently many smart phone operating systems include NFC software protocol stack, but there are not many applications available to the user. Also connection handover support across multiple smart phone operating systems is not yet available at the moment and it is not sure whether it will be available in future from vendors since this includes commercial aspects. This thesis work explores the possibility of making an application prototype which can support connection handover between devices from multiple vendors.

Security is an important aspect which should be considered during connection handover. It might be needed that only authorized devices can make connection handover request. Transmission of Carrier Configuration Data such as Bluetooth device identification data needs to be secured. Connection handover specification mentions it as due to the close proximity nature -typically 4/10 cm distance - the Handover Requester device could be legitimate. But it does not rule out the possibility of eavesdropping.

3.2 NFC based data transfer in Android

Android [20] is an operating system which is based on the Linux kernel, targeted to be used in mobile devices, specifically in smartphones and tablets devices. It was developed by a company called Android, which was later acquired by Google.

Android was announced during 2007 [21] by Google and its source code is based on Apache license which allows device manufacturers to modify and distribute in their devices. Most of the Android devices available today contain combination open source and proprietary software. Google provides an application ecosystem, with an application store, called Google Play, which allows developers to distribute their applications commercially or for free or cost. End uses can download the applications on their Android based smartphones. Device manufactures can also provide their own application store.

Android is based on open source based Linux Kernel. The core of the operating system is developed using Linux Kernel which is written in C. C++ language is also used for development and user interface (UI) is based on Java. The Android application ecosystem enables software developers to develop their applications based on Android SDK using JAVA as programming language.

3.2.1 Android API support for NFC

Android Framework APIs are based on NFC Forum standard [22]. An Android based smartphone may include an NFC controller from any vendor, such as NXP [30] or QUALCOMM [31]. The Android software stack includes the required components to control the NFC hardware including the Hardware Abstraction Layer, the middleware which implements the NFC specific protocols and the APIs for application development. The APIs are mostly based on NFC Forum standard for NDEF.

Android based devices support mostly the following NFC operating modes.

- **Reader/Write mode** which allows the mobile device to read and write data in NFC tags and stickers.
- **Peer to Peer mode (P2P)** which allows data exchange with another NFC based mobile device. This mode has the application also, called Beam, which is supported in recent versions of Android.
- **Card emulation mode** which allows the mobile device to emulate itself

NFC tag and it can be accessed by an external reader, such as a point-of-sale terminal to do the transaction.

Table 6 lists the classes which implement the NFC functionality in Android. These classes provide functions to control the NFC controller, exchange data, parse the NFC data, etc.

NfcManager	The top level NFC class, provides means to access the NFC instance, through *NfcAdapter*. Static helper *getDefaultAdapter(android.content.Context)* can also be used to get the local instance.
NfcAdapter	This represents the device's NFC adapter, which is the entry point of performing NFC operations.
NdefMessage	This represents the NDEF data message, as defined in the NDEF specification. This class uses the NdefRecord class to create NDEF messages which will be sent to the peer device or to the NFC tag.
NdefRecord	Represents the NDEF record, as defined in the NDEF specification and this class will be used by the NdefMessage class to create complete NDEF message.

Table 6: List of NFC related APIs in Android.

In the following sections, we briefly describe the three technologies, i.e. Bluetooth, Wireless LAN and NFC, along with their API level support in the Android OS SDK.

3.3 Bluetooth application development in Android

Bluetooth is a short range wireless standard for data exchange between devices including mobile based. This technology is supported in most of the mobile devices for data synchronization. It is defined and maintained by Special Interest Group (SIG), a global standards organization. Bluetooth is based on IEEE specification, IEEE 802.15.1 and it operates on 2.4 to 2.485 GHz.

3.3.1 Bluetooth stack in Android

A Bluetooth network stack is available in Android OS to exchange data over wireless. The stack also provides public API functions to develop applications that can use Bluetooth devices. Application developers can use the public API functions in their applications to control and use the Bluetooth devices using the Bluetooth network stack. The API functions provide support for the following functionalities [23].

- Check for available Bluetooth devices through RF scanning.
- Request the local Bluetooth adapter to provide the list of already paired devices, if any.
- Establish RFCOMM (Radio frequency communication).
- Establish connection with other devices through service discovery.
- Exchange data between devices.
- Manage several device connections.

The above features are provided as 'Classic Bluetooth'. Bluetooth specification also supports a feature called as 'Bluetooth Low Energy' which is intended for power saving while using Bluetooth technology on devices. Android supports Bluetooth Low Energy (BLE) feature, and also provides API functions. The application prototype to be developed in the thesis work only uses Classic Bluetooth, as BLE is not needed for performing connection handover.

3.3.2 Bluetooth API support in Android

Bluetooth specific APIs are available at android.bluetooth package. Table 7 shows API classes that are used for the connection handover application development [23].

BluetoothAdapter	Represents the Bluetooth device on the system (Bluetooth radio hardware). This can be used to - query the Bluetooth devices that are available nearby, query the list of already paired devices, create an instance of available Bluetooth device using a known MAC address of the device, create server socket to listen communications from the other device.
BluetoothDevice	Represents the instance of remote Bluetooth device. This will be used to request connection with the remote device using *BluetoothSocket*, or query information about the remote device such as device name, address, device class and pairing state.
BluetoothSocket	This represents the connection point from where data could be exchanged with the remote device, similar to a *TCP Socket*, using *InputStream* or *OutputStream*.
BluetoothServerSocket	This represents the server socket which listens and accepts the connection request from the remote device. When connection is accepted withe remote device, this returns the instance of *BluetoothSocket*.
BluetoothClass	Represents the read-only set of properties which describe the general characteristics of the device such as major and minor device classes and its services.
BluetoothProfile	Represents the *Bluetooth profile*, is a wireless interface specification for Bluetooth based communication between devices, for example, a Bluetooth headset or Bluetooth health device.
BluetoothHeadset	Represents support Bluetooth headset devices to be used with mobile phones.
BluetoothA2DP	Advanced audio distribution profile, which defines how high qulity audio streaming can be done between one device to another over Bluetooth connection.

Table 7: List of Android APIs that are used for Bluetooth connectivity [23].

The above classes are mostly used to test the NFC connection handover.

3.3.3 Bluetooth permissions

Permissions need to be obtained from the Android system, if any application wants to use the Bluetooth hardware. The application which needs to perform any Bluetooth communication such as requesting and accepting connections, should declare this in the project settings file, i.e. BLUETOOTH and BLUETOOTH_ADMIN permissions need be declared in the manifest file to get the needed permissions. When the end user installs the application, dialog will be presented requesting permission to use the Bluetooth device.

3.4 Wireless LAN application development in Android

Wireless Local Area Network (WLAN) is defined by IEEE 802.11 standard which is a network protocol standard used to interconnect two devices over wireless. It also defines the devices to move between network seamlessly.

Typical wireless network contains two different components, wireless access points and client devices. Wireless access points are basically wireless routers which are also called wireless base stations used to send and receive data in a pre-defined radio frequency to enable interconnection between different client devices.

Though there are other standards exist for WLANs, IEE802.11 is the widely used WLAN standard and the devices that support are marketed under a brand name called WiFi.

Client devices include any mobile or normal device which need to communicate with another device over wireless. Smart phones, personal computing devices and several other consumer devices support the WiFi technology and can communicate with another device using wireless access points.

WLAN can operate in two modes, namely ad hoc and infrastructure. In an ad hoc mode, the devices will communicate with each other in peer to peer mode without access points. In an infrastructure mode, the device exchange date through wireless access point which acts as a bridge to transmit and receive data.

Infrastructure WLAN network can have one or more access points which can co-exist and help devices to move between different access points. Group of wireless stations are collectively called as Basic Service Set (BSS) and identified using BSS ID which is normally the MAC address of the wireless router. A set of connected BSSs called Extended Service set (ESS), extended service set which is identified by a 32 byte character string, called as ESS ID.

Individual wireless client devices can connect to an ESS using the ESS ID and authenticate themselves based on the policies defined on the ESS.

Android supports the low level wireless stack, which provides Wi-Fi network access. For Developers, the APIs are in android.net.wifi package.

Authentication in wireless networks is performed by the supplicant component. Using supplicant the user can submit the authentication information and also retrieve the network status such as IP address, negotiation state and link speed.

Due to availability of hardware, the prototype does not include any features to test wireless LAN based handover, but it could be extended by adding the authentication part of WLAN association process, as the MAC address information will be read from the embedded NFC tag for static handover.

4 Prototype design and implementation

The Connection Handover specification version 1.2 [5] was used as a reference for developing the application prototype. The application prototype was built for an Android mobile operating system with the NFC support. Implementation was done using Eclipse IDE and tested in an Android based phone, i.e. Google Nexus S [24]. The prototype does not have a usable user interface (UI) as it needs to run as a background activity in a real world environment to detect the tags and pair the devices.

An Android SDK [25] provides the NFC API support for development as well as APIs for Bluetooth and Wireless LAN protocols. The application is tested against another NFC supported device. Nokia BH-505 [26] Bluetooth headset supports NFC and it was used as a target multimedia device to do the Bluetooth connection handover.

As discussed in the above chapters, NFC Connection handover specification [5] defines two types of handover methods, Static Handover and Negotiated Handover. The Negotiated handover supports choosing a specific carrier when more than one is available. The application prototype will be implemented with static handover targeted with NFC based Bluetooth multimedia device.

The Connection handover specification defines NFC Data Exchange Format (NDEF) messages which will be exchanged between the two devices to setup pairing and negotiate the alternative communication carrier for further data transfer. The device which requests is a Handover Requester and the other device is a Handover Selector. The Handover Selector device remains passive until it gets the handover request. In physical terms, the Handover Selector device does not generate any active RF field to look for counterpart devices.

In Negotiated Handover, during request, the Handover Selector will announce the available technologies with priority information, and a Handover Requester will choose one of them. A Handover Requester informs the carrier selection to the Selector.

If the pairing could not be done using the chosen technology, the Requester moves to the second available carrier and starts the pairing process. For example, if the chosen one is Bluetooth and if the user has already moved out of the operating range, Requester will try to proceed with the next one.

The NFC tag contains all the necessary information for pairing, which will be in the form of Handover Select Record, as defined in the specification. This information will be static and may not be suitable for all use cases, for example, providing a dynamic IP address, in case of WLAN pairing. It could be used without any issues in case of Bluetooth as it does not need any dynamic information for pairing.

The prototype application uses a Static Handover approach and implements pairing using Bluetooth. During development, it is observed that there are security limitations in both modes of pairing. The Handover Record which is required for pairing, contains the network access data to be used for carrier configuration. This might include credentials along with other sensitive information, like IP and MAC addresses.

Current version of the connection handover protocol assumes that the devices that can be brought to close proximity are considered as legitimate. If this is not the case, the specification leaves open for further analysis of the operating environment to implement the needed security. By implementing the prototype, it is aimed to analyze the above security issues as well as the interoperability issues.

4.1 Android Software Development Kit

Android software development kit (SDK) was used for the prototype development. The Android SDK includes a comprehensive list of software tools to develop, debug and deploy Android applications. These applications can be submitted to the application store for review and approval. Users can download and install once it is available in the store.

The SDK includes compiler, debugger, software libraries, emulator, example source code and documentation, etc. Each SDK release includes updated APIs that applications can use to interact with the underlying Android System. Each new revision of APIs are referred with an integer, called API level. When an application is developed using a certain API level, it may or may not support the future API levels, since the same API could be be changed to provide a new functionality in the next release. This might require the application to be maintained continuously to get it ported to the new API level.

Generally an Android framework API includes the following components.

- Core set of packages and classes which provide the required functions for the application development.
- A set of XML elements and attributes for declaring the manifest file which describes the application properties.
- A set of XML elements and attributes for declaring and accessing resources.
- A set of intents which provide the messaging objects that facilitate communication between components.

- A set of permissions that the application can request, as well as the permission enforcements included in the system.

Open source based Eclipse is the officially supported Integrated development environment (IDE) for application development and Google also provides its own IDE called Android Studio, which is currently offered as preview version. Eclipse was used for the development of Connection Handover Application. Eclipse is a freely available open source IDE which can be used to develop software tools using C++, JAVA, C and various other languages.

Eclipse IDE supports Android application development using Android Development Tools (ADT) plugin in JAVA. Android core provides JAVA virtual machine called Dalvik, which allows executing Android applications. Other than JAVA, C and C++ are also supported for development. Native applications which can run without the support of any virtual machine could be developed using C and C++.

Eclipse IDE supports creating Android projects that contain the source code of Android application which needs to be developed, with the support of Android SDK. The Android SDK includes the API libraries and developer tools which are needed to build, test and debug applications.

To simplify the application development, Google provides ADT bundle to quickly start developing applications. The ADT bundle which can be freely downloaded and used, includes essential SDK components and Eclipse IDE with built-in Android Developer Tools (ADT) plugin.

In summary, the ADT bundle includes Android SDK Tools, Android Platform Tools, current Android Platform Image which is needed for emulator and recent version of Eclipse IDE and ADT plugin. Table 8 shows the operating system and development environment specific system requirements for Android based development.

Operating systems	Windows XP (32-bit), Vista/Windows 7 (32/64 bit)
Eclipse IDE	Eclipse 3.7.2 (Indigo) or later with JAVA SDK version 6 or later

Table 8: System requirements for Android development.

4.2 Implementing an NFC and Bluetooth handover prototype

The prototype aims to implement the basic connection handover, thus exploring the specification features such as interoperability and security. The environment uses Nexus S [24] Android phone and Nokia BH-505 [26] Bluetooth NFC headset. The hardware setup puts limitation in performing negotiated connection handover. Since the counterpart device is a Bluetooth headset which only has a passive NFC tag, static connection handover could only be performed. A simplified sequence of the implemented prototype is as in Figure 16.

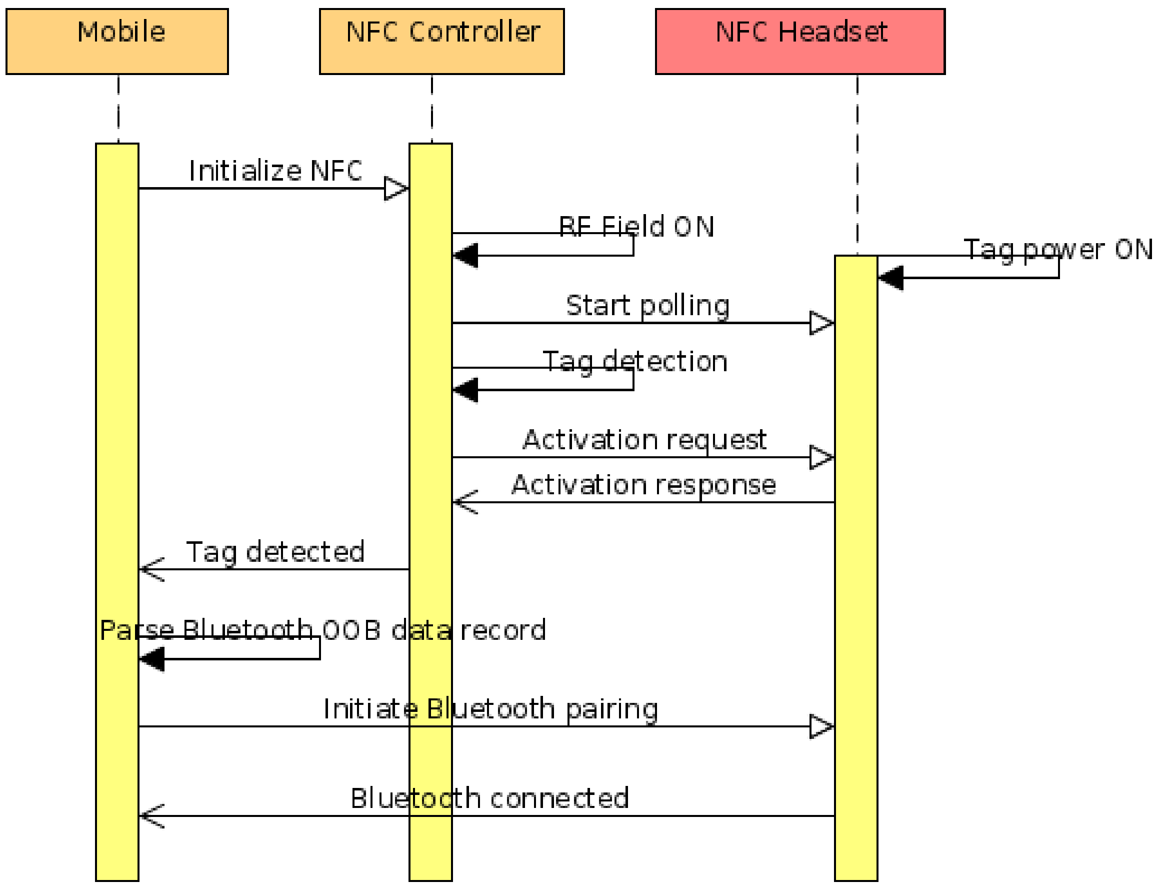

Figure 16: Application handover sequence.

The prototype implementation comprises the following steps: Initializing the NFC in Reader / Writer mode and scanning the NFC devices or tag in the close proximity. If NFC tags or NFC devices are found, a connection handover message sequence is performed. The application initiates the static handover process, by looking for the NFC tag in the close proximity, which will be the Nokia BH-505 [26] headset. After detecting the tag by using NFC-A polling sequence, the NDEF data needs to be retrieved and parsed. This NDEF data has the Connection Handover Message with carrier configuration data. The carrier configuration data will be parsed and after performing the necessary steps such as version checking and carrier power state.

The Handover Select Message stored in an NFC Forum tag is similar to the Handover Select Message returned by active NFC Forum device as it is in the negotiated handover. This message will have all carrier configuration data and it will not support any dynamic data. Figure 17 shows the Bluetooth configuration data stored in an NFC tag.

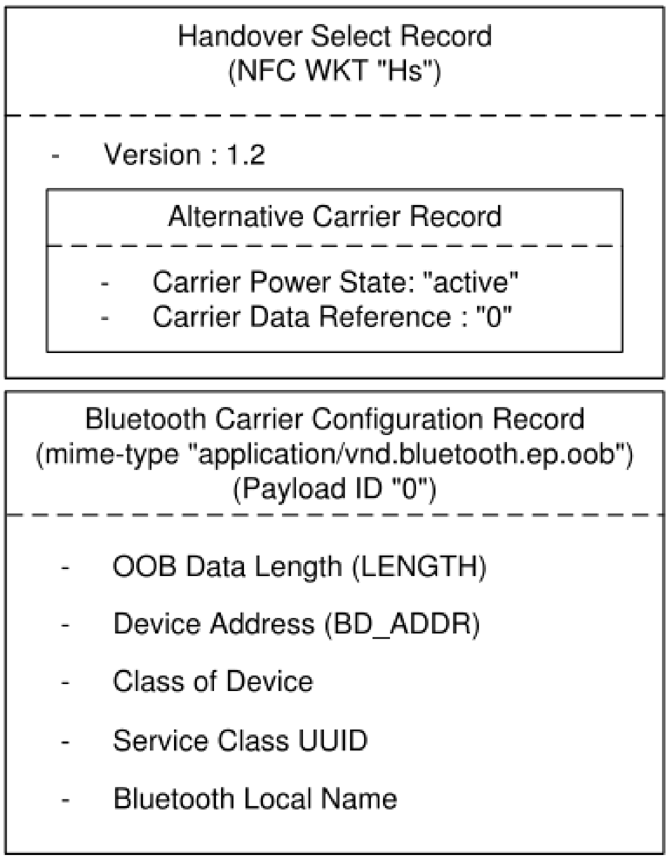
Handover Select Record
(NFC WKT "Hs")
- Version : 1.2
Alternative Carrier Record
- Carrier Power State: "active"
- Carrier Data Reference : "0"
Bluetooth Carrier Configuration Record
(mime-type "application/vnd.bluetooth.ep.oob")
(Payload ID "0")
- OOB Data Length (LENGTH)
- Device Address (BD_ADDR)
- Class of Device
- Service Class UUID
- Bluetooth Local Name

Figure 17: Bluetooth Configuration Data on a NFC tag [5].

In Figure 17, the power state is mentioned as 'active', which indicates that the Bluetooth should be powered on when performing this negotiation. Other power states such as 'inactive' or 'unknown' will lead to undefined behavior and it needs separate request from the Handover Requester to activate the carrier.

If the Handover Selector device will provide only one alternative carrier, a simplified format of Handover Selector Record could be used. In this case the NFC

Forum tag contains only the Bluetooth OOB information, which only provides the essential information needed for handover, such as Bluetooth MAC address, local device name and class of device, as shown in Figure 18.

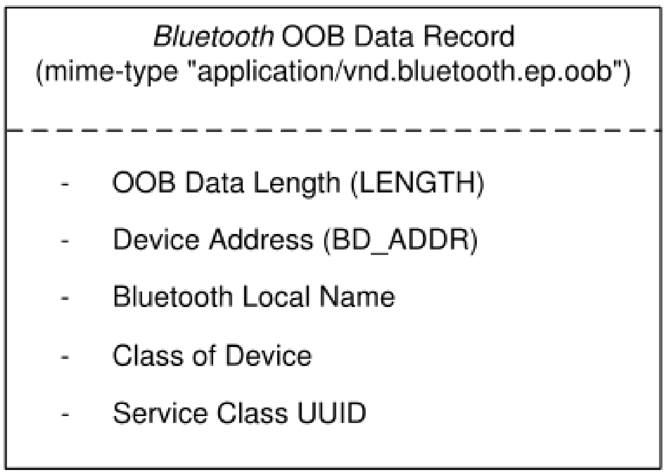
Bluetooth OOB Data Record
(mime-type "application/vnd.bluetooth.ep.oob")
- OOB Data Length (LENGTH)
- Device Address (BD_ADDR)
- Bluetooth Local Name
- Class of Device
- Service Class UUID

Figure 18: Bluetooth OOB data on a NFC tag - Simplified tag format with one carrier [5].

The Nokia BH-505 [26] supports the simplified tag format. The Buletooth headset has the embedded NFC tag made by Infineon Technologies [27], called my-d move (SLE66R01P).

The characteristics of the NFC tag IC are shown in Table 9.

Standards supported	ISO/IEC 14443-3 Type A, ISO/IEC 18092 and NFC ForumTM Type 2 Tag Specification
Data rates supported	106 Kbits/s
Storage	128 byte programmable memory

Table 9: Characteristics of NFC tag in BH-505.

The code snippet in Appendix A shows how to initialize the NFC device for tag detection. The source code in Appendix A performs the following steps in sequence.

1. Get the default instance of the NFC adapter present in the mobile device. NFC adapter class represents the physical NFC controller present in the mobile which has the NFC Firmware and it is controlled through the Android OS hardware abstraction layer and device drivers. It is assumed that the NFC adapter is enabled before retrieving the instance.

2. Register an intent to receive NFC related event notifications.

3. Set up intent filtering. This will filter the event whenever a new NFC tag is brought to proximity and detected by mobile.

The code snippet in Appendix B shows how to read the NFC tag data when the mobile device is brought close to the Bluetooth BH-505 headset. The source code in Appendix B performs the following steps in sequence.

1. When new intent is discovered, the intent type is checked and filtered for either one of these three events - ACTION_TECH_DISCOVERED, ACTION_TAG_DISCOVERED, ACTION_NDEF_DISCOVERED. These events indicate that a new tag technology is found, new NFC tag is discovered, new NDEF data is available respectively.

2. After getting the intent, the data is parsed and checked for the correct NFC tag type. Android provides classes for all tag types to create or parse each NFC tag specific data.

Following shows the parsed Handover Selector Record, which has the simplified tag format and contains single Bluetooth carrier record from the NFC tag embedded on the Nokia BH-505 headset:

```
** TagInfo scan (version 1.40) 2012-12-05 12:05:17 **
# IC manufacturer:
Infineon Technologies AG
# IC type:
my-d move (SLE66R01P)

# NFC Forum NDEF-compliant tag:
Type 2 Tag
--------------------------------------
# NFC data set information:
Current message size: 68 bytes
Maximum message size: 110 bytes
NFC data set access: Read-Only

# Bluetooth Secure Simple Pairing record:
type: "application/vnd.bluetooth.ep.oob"
payload length: 33 bytes
payload data:
[0000] 1F 00 EE 96 29 DE 1E 00 |....)...|
[0008] 0D 09 4E 6F 6B 69 61 20 |..Nokia |
[0010] 42 48 2D 35 30 35 04 0D |BH-505..|
[0018] 04 04 20 05 03 18 11 23 |.. ....#|
[0020] 11                      |.       |
--------------------------------------
# Technologies supported:
ISO/IEC 14443-3 (Type A) compatible
ISO/IEC 14443-2 (Type A) compatible

# Android technology information:
android.nfc.tech.Ndef
android.nfc.tech.NfcA
* Maximum transceive length: 253 bytes
* Default maximum transceive time-out: 618 ms
Tag description:
* TAG: Tech [android.nfc.tech.NfcA, android.nfc.tech.Ndef]

# Detailed protocol information:
ID: 05:34:00:00:57:FA:24
ATQA: 0x4400
SAK: 0x00
```

The detailed contents of the above NDEF message are as in Table 10.

Offset [octets]	Content	Length [octets]	Description
0	0xD2	1	NDEF Record Header: MB=1b, ME=1b, CF=0b, SR=1b, IL=0b, TNF=010b
1	0x20	1	Record Type Length: 32 octets
2	0x21	1	Payload size: 32 octets
3	`0x61 0x70 0x70 0x6C 0x69 0x63 0x61 0x74 0x69 0x6F 0x6E 0x2F 0x76 0x6E 0x64 0x2E 0x62 0x6C 0x75 0x65 0x74 0x6F 0x6F 0x74 0x68 0x2E 0x65 0x70 0x2E 0x6F 0x6F 0x62`	32	Record Type Name: application/vnd.bluetooth.ep.oob
35	0x1F 0x00	2	OOB Optional data length: 33 octets
37	0xEE 0x96 0x29 0xDE 0x1E 0x00	6	Bluetooth MAC address : 6 octets
43	0x0D	1	EIR data length, 13 octets
44	0x09	1	EIR data type
45	`4E 6F 6B 69 61 20 42 48 2D 35 30 35`	12	Complete local name 'Nokia BH-505'
57	0x04	1	EIR Data Length: 4 octets
58	0x0D	1	EIR Data type: class of device
59	0x04 0x04 0x20	3	0x20 - Audio service class, 0x04 - Major device class (audio/video), 0x04 - Minor device class (wearable headset device)
62	0x05	1	EIR Data length: 5 octets
63	0x03	1	EIR Data type: 16 bit service class UUID
64	0x18 0x11 0x23 0x11	4	0x1118 - HPF HF, 0x1123 - A2DP profile

Table 10: NDEF message contents in BH-505 NFC Tag

The carrier configuration data has the Bluetooth MAC address and the device name which will be used for the connection handover. After having the carrier configuration data, the handover process needs to be initiated, by scanning for the availability of the Bluetooth device using Bluetooth MAC address. After detecting the Bluetooth device, usual Bluetooth pairing has to be done.

The code snippet in Appendix C shows the NDEF data parsing of Bluetooth MAC address and the handover part as mentioned in the following sequence.

1. Extract the Bluetooth adapter MAC address from the received NDEF record.
2. Get the instance of the Bluetooth adapter. It is assumed that the Bluetooth adapter in the mobile device is powered on before this.
3. Start the discovery activity and try to detect the Bluetooth device using the parsed MAC address.
4. If discovery is successful, pair the device automatically.

5 Testing and analysis of results

5.1 Requirements for interoperability

The main goal of interoperability is that any NFC enabled device with alternative carrier options implements and uses connection handover in an interoperable way to provide easy to use spontaneous data sharing between different types of NFC devices. This can be achieved in any NFC device with the alternative carrier technologies such as Bluetooth or WiFi by implementing Connection Handover specification even when they are from different vendors. Connection handover application tests this level of interoperability support in a multi-party hardware and software environment.

Connection handover application is expected to read NDEF message from the NFC tag presented by the counter part multimedia device and verify that the NDEF message is a Handover Select message with required Bluetooth configuration data which contains Bluetooth device address, local device name, class of device, etc. It should also verify that the Bluetooth connection can be established with the counterpart device.

Interoperability test requirements for the static handover can be summarized as shown in the following. These requirements are applicable to the Bluetooth and NFC connection handover process.

REQ-1: The mobile device shall enable the NFC functionality and read the carrier configuration data from the NFC tag which is embedded into the target multimedia device, the Nokia BH-505 headset. This requirement shows that the NFC data on the target can be retrieved from any NFC enabled mobile device. The retrieved data shall be verified that it has a valid Handover Select message which contains Bluetooth device address and device name.

REQ-2: The headset shall power on the alternative carrier, the Bluetooth service. This requirement enables the mobile device to start connecting the alternative technology after parsing and verifying the NFC data. Since the dynamic power mode handling can only be achieved in negotiated handover, the low powered devices which use the static handover shall remain powered on before the read operation or shall enable power during the NFC read activity.

REQ-3: The mobile device shall be able to complete the Bluetooth pairing automatically using the carrier configuration data without any user interaction. The prototype application can also enforce a security code based authorization for pairing. The NFC functionality shall be disabled after connection established over the Bluetooth.

REQ-4: After successful pairing, the mobile device shall be able to stream multimedia content to the headset.

The table 11 has the list of test cases for these requirements.

5.2 Test environment and test cases

The test environment includes the Eclipse development environment with ADT plugin, a Nexus S device with the application prototype installed, a Samsung Galaxy S4 and a Nokia N9. The Nokia N9 is used to test the vendor specific behavior. This device is officially supported by Nokia to be compatible with its BH-505 bluetooth headset. The Samsung Galaxy S4 is used for interoperability testing with BH-505. This device is also based on Android as Nexus S, except that it supports the most recent version of Android which has a higher API level. The Eclipse integrated development environment with ADT plugin has been used to debug and install the application on the mobile device. Windows 7 and Ubuntu Linux operating systems have been used for the prototype development.

The Table 11 has the list of test cases that have been used to check the connection handover behavior with multiple devices. The test cases are listed along with their steps and results. The testing does not include all possible functional and non functional cases, since the aim was only to check the interoperable behavior.

Test case and steps	**Results**
1. Read the NFC tag contents **Step 1.** Bring Nexus S and Nokia BH-505 (power on state) headset to close proximity. **Step 2.** Read NFC tag contents through debug log.	The application was able to read the Bluetooth MAC address of the BH-505 headset (0xEE 0x96 0x29 0xDE 6 0x1E 0x00).
2. Power mode handling **Step 1.** Bring Nexus S and Nokia BH-505 (power off state) headset to close proximity. **Step 2.** Check the power state of the Bluetooth headset. **Step 3.** Repeat the above steps with Nokia N9 and Samsung Galaxy S4.	The power state changes with Samsung Galaxy S4 and Nokia N9 (Powered off -> Powered on). No power state change observed with Nexus S.
3. Connection Handover **Step 1.** Bring Nexus S and Nokia BH-505 (powered on state) headset to close proximity. **Step 2.** Bluetooth pairing should be done without user confirmation. **Step 3.** Repeat the above steps with Nokia N9 and Samsung Galaxy S4 (headset in powered off state).	All the three devices can perform handover. Nokia N9 has vendor specific support for Nokia headset. Nexus S (with Android 4.1 API level 16) achieves connection handover with the help of prototype application. Samsung Galaxy S4 (with Android 4.4.2 API level 19) is able to connect after user confirmation.
4. Automatic pairing **Step 1.** Bring Nexus S and Nokia BH-505 (powered on state) headset to close proximity. **Step 2.** Bluetooth pairing should be done with without user confirmation.	This tests the Bluetooth pairing behavior. The pairing should be done without user confirmation as implemented in the prototype. Nexus S was able to pair with the headset in 2-3 seconds. Pairing the headset using only with Bluetooth took around 1-2 minutes, including scanning the available Bluetooth devices and pairing the selected one after user confirmation.
5. Media streaming **Step 1.** Bring Nexus S and Nokia BH-505 (powered on state) headset to close proximity. **Step 2.** Bluetooth pairing should be done with without user confirmation. **Step 2.** Play some audio in the mobile and check them with headset.	This test confirms that the prototype was able to perform the connection handover from NFC to Bluetooth successfully.

Table 11: Test cases list and the results.

5.3 Key assumptions and limitations

It is assumed that the connection handover specification is subject to change in future revisions. Current stable version is version 1.2 and a candidate specification is available for version 1.3.

Connection handover protocol implementation on the software stack of various smartphone operating systems are not clearly documented. They may be providing only minimum APIs which may or may not support interoperability.

Also, connection handover support may not be implemented completely according to the specification as this is not mandatory for the vendor. This support might be needed only when they need to market devices that are NFC-Forum certified.

The support of connection handover protocol may vary between different releases of smartphone operating systems. For example, Android version 4.4, i.e. Kitkat, has enhanced support of this protocol than its older version 2.3, i.e. Gingerbread.

Over Bluetooth and Wireless LAN, NFC enables more user friendly touch based communication between end user devices. Also, the time to set up the device communication is usually in terms of hundreds of milliseconds in NFC, whereas in Bluetooth and Wireless LAN, it takes several seconds.

5.4 Issues in vendor interoperability

Connection handover between Google Nexus S with Nokia BH-505 does not happen seamlessly. The handover needs to be forced by the connection handover application, which does the static handover by reading the Connection Handover Record. Bluetooth pairing could be done automatically without user permission or it could also be forced with user permission. The limitation was observed due to lack of protocol support in the Android stack. Nexus S is running the Icecream Sandwich version of Android Operating system, which does not provide any NFC connection handover APIs explicitly. Android Operating system is being constantly upgraded and released to smartphone devices.

NFC functionality has been introduced with Android 2.3–2.3.2 Gingerbread (API level 9). Nexus S was initially having Android 2.3–2.3.2 Gingerbread (API level 9) and further upgraded through Android 2.3.3–2.3.7 Gingerbread (API level 10), Android 3.0 Honeycomb (API level 11), Android 3.1 Honeycomb (API level 12), Android 3.2 Honeycomb (API level 13) and finally to Android 4.0–4.0.2 Ice Cream Sandwich (API level 14). The prototype has been tested with Android 4.0 and Android 4.0.2. Even though API level is same through 4.0 to 4.0.2, there were API changes and difference in behavior has been observed. Google supported Android upgrade until Android 4.1 Jelly Bean (API level 16) for Nexus S, but afterwards no updates were released to Nexus S, due to hardware limitations.

Samsung S4 [28] has been used for testing, which has Android 4.3 Jelly Bean (API level 18), but it did not detect the NFC tag or performed any handover automatically. It was supporting NFC Beam, a feature which provides NFC peer to peer communication between two NFC devices.

The latest version Android 4.4 KitKat with API level 19, does not support older hardware.

In summary, there were no public APIs available in Android Operating system, which can be used by application developers to implement connection handover applications. It is also observed that Nokia BH-505 performs static connection handover automatically only with Nokia N9 [29], which was advertised as target device for the Bluetooth NFC headset.

5.5 Protocol specification level issues

Connection Handover Specification states that transmission of carrier configuration data to the devices that can be brought to close proximity are deemed legitimate within the scope of the specification. It also states that security attacks like eavesdropping of carrier configuration data is difficult, though it is possible. This leaves the application developers to consider extra measures to protect the carrier configuration data.

The NFC tag which has the carrier configuration data (Bluetooth MAC address and other information) in BH-505 could be read by any device which has the NFC capability. Although the tag is write protected, which secures from the tag data from tampering, it still provides the possibility of getting connected by any NFC device which does not belong to the legitimate owner of the device.

Due to the above issue, protecting the carrier configuration data becomes a limitation and this provides a scope for vendor locked devices, instead of open device which can work with other vendor's products. This carrier configuration data could be protected by storing the data in encrypted format which can only be able to decrypted and read by another vendor specific device. The above use case which has a mobile device and a headset may not pose a high security risk, but it can be different considering the use case such as a wireless printer which has NFC support, in which case, it allows anyone who can be close enough to the printer can print.

5.6 Operating system specific issues

The leading mobile operating systems such as Android and Windows Mobile support NFC but they do not provide any publicly available API information that can be used for connection handover. Due to this, application developers can not use a common interface and it leads to more device specific applications. This is observed when testing Samsung Galaxy S4 with Nokia BH-505, in which the mobile could only read the NFC tag content and it has failed to perform connection handover successfully.

6 Summary and conclusion

NFC technology is already being taken into different domains of end-user applications including banking, security, etc., where most of them are based on hand-held devices. The prototype explores the possibility of having mobile based applications which can recognize the NFC supported multimedia devices from different vendors and connecting them with the mobile for further data transfer and also provide adequate security options.

NFC support has become ubiquitous in most of the recent devices, which enables the possibility of using the technology for other purposes than the intended ones. A NFC enabled mobile phone could provide banking applications as well as connection handover support with other multimedia devices, thus the feature is not restricted to a specific application.

The idea of interoperability is being explored, thus analyzing the possibility of using NFC enabled devices from different vendors interacting with each other to connect and share data. A wide range of multi media devices and mobile devices could communicate with each other irrespective of vendor specific operating system or application.

Based on the current version of the connection handover protocol, the state of interoperability is at the beginning level. While the vendor specific counterpart devices are working seamlessly, interoperability between different vendor devices are not working completely based on connection handover protocol. It is also observed that the connection handover support is being improved in each new release of the software stack.

Interoperability also brings the issue of security. Since many of the multimedia devices like the Bluetooth headset do not have the option to control whether it can be attached with another device for example with a mobile. In this situation the mobile device automatically requests and pairs the headset without requesting permissions from the headset. This enables the possibility of being paired with any device which is on the proximity. This could become a security issue, if the multimedia device holds some confidential data or provides route to access such data.

The future work could explore the possibility of a connection handover application which can support a range of NFC based multimedia devices from different vendors, by implementing support for static and negotiated handover in a generic way using Bluetooth and Wireless LAN. In case of the Android, the NFC application can take complete control of NFC activities on the device and this provides the possibility of generic handling of the connection handover irrespective of the level of support added in the Android NFC stack.

References

[1] Tom Igoe, Don Coleman and Brian Jepson, *Beginning NFC.* O'Reilly Media, 2014.

[2] Bill Glover, Himanshu Bhatt, *RFID Essentials.* O'Reilly Media, 2006.

[3] *NFC Forum.* `http://www.nfcforum.org/` [Accessed 2015-01-28]

[4] *NFC Forum Technical Specifications.* 2006, `http://members.nfc-forum.org/specs/spec_list/` [Accessed 2015-01-28]

[5] *NFC Forum Connection Handover Technical Specification.* Version 1.2, NFC Forum, 2010, `http://www.nfc-forum.org/specs/spec_list/#refapps` [Accessed 2015-01-28]

[6] *Interoperability in Bluetooth NFC sharing.* `http://members.nfc-forum.org/apps/group_public/download.php/18688/NFCForum-AD-BTSSP_1_1.pdf` [Accessed 2015-01-28]

[7] *ISO/IEC 9126 Software engineering — Product quality / ISO/IEC 25010:2011.* `http://www.iso.org/iso/catalogue_detail.htm?csnumber=35733` [Accessed 2015-01-28]

[8] Vedat Coskun, Kerem Ok and Busra Ozdenizci, *Professional NFC Application Development for Android.* Wrox, 2013.

[9] *Apple Pay NFC payment system.* `https://www.apple.com/apple-pay/` [Accessed 2015-01-28]

[10] *ECMA-340: Near Field Communication – Interface and Protocol (NFCIP-1) which is based on ISO18092*, `http://www.ecma-international.org/publications/standards/Ecma-340.htm`, `http://www.ecma-international.org/publications/files/ECMA-ST/Ecma-340.pdf`, 2013 [Accessed 2015-01-28]

[11] *ECMA-352: Near Field Communication Interface and Protocol -2 (NFCIP-2).* `http://www.ecma-international.org/publications/standards/Ecma-352.htm`, `http://www.ecma-international.org/publications/files/ECMA-ST/ECMA-352.pdf`, 2013 [Accessed 2015-01-28]

[12] *NFC Forum Tag Type Technical Specifications.* NFC Forum, `http://members.nfc-forum.org/specs/spec_list/`, 2007 Accessed 2015-01-28

[13] *ISO/IEC 14443-4:2008 - Identification cards – Contactless integrated circuit cards – Proximity cards – Part 4: Transmission protocol.* `http://www.iso.org/iso/catalogue_detail.htm?csnumber=50648`, 2008 [Accessed 2015-01-28]

[14] *NFC Forum Standards architecture.* `http://nfc-forum.org/our-work/specifications-and-application-documents/specifications/` [Accessed 2015-01-28]

[15] *Tagawa, Koichi*, Bringing NFC to Market, 2010, `http://members.nfc-forum.org/resources/presentations/contactless_2010.pdf` [Accessed 2015-01-28]

[16] *NFC Digital Protocol Technical Specification.* Version 1.0, NFC Forum, 2010, `http://www.nfc-forum.org/specs/spec_list/` [Accessed 2015-01-28]

[17] *NFC Activity Technical Specification.* Version 1.1, NFC Forum, 2014, `http://www.nfc-forum.org/specs/spec_list/` [Accessed 2015-01-28]

[18] *NFC Controller Interface (NCI) Technical Specification.* Version 1.1, NFC Forum, 2014, `http://www.nfc-forum.org/specs/spec_list/` [Accessed 2015-01-28]

[19] *NFC Data Exchange Format (NDEF).* Version 1.0, NFC Forum, 2006, `http://www.nfc-forum.org/specs/spec_list/` [Accessed 2015-01-28]

[20] *Android Mobile Operating System.* `http://www.android.com/` [Accessed 2015-01-28]

[21] *Android Mobile Operating System version information.* `https://developer.android.com/about/dashboards/index.html` [Accessed 2015-01-28]

[22] *NFC Developer guide in Android.* `https://developer.android.com/guide/topics/connectivity/nfc/index.html` [Accessed 2015-01-28]

[23] *Bluetooth API support in Android.* `https://developer.android.com/guide/topics/connectivity/bluetooth.html` [Accessed 2015-01-28]

[24] *Google Nexus.* `http://www.android.com/devices/detail/nexus-s` [Accessed 2015-01-28]

[25] *NFC support in Android SDK.* `https://developer.android.com/reference/android/nfc/package-summary.html` [Accessed 2015-01-28]

[26] *Nokia BH-505 NFC Bluetooth headset.* http://www.nokia.com/global/products/accessory/bh-505/ [Accessed 2015-01-28]

[27] *NFC Tag from Infineon Technologies.* http://www.infineon.com/cms/en/product/smart-card-ic/transport-and-ticketing-ic/mifare-compatible-my-d-tm-proximity-and-ticketing-products/SLE+66R01P+my-d-tm+move/productType.html?productType=db3a30433fcce646013fd2391aee23e1, https://developer.android.com/reference/android/nfc/package-summary.html [Accessed 2015-01-28]

[28] *NFC support in Galaxy S4 mobile.* https://www.samsung.com/global/microsite/galaxys4/ [Accessed 2015-01-28]

[29] *NFC in Nokia devices.* http://developer.nokia.com/community/wiki/Category:Near_Field_Communication_%28NFC%29 [Accessed 2015-01-28]

[30] *NXP Semiconductors.* www.nxp.com [Accessed 2015-01-28]

[31] *Qualcomm incorporated.* www.qualcomm.com [Accessed 2015-01-28]

Glossary

Active Communication A communication mode in which each device generates its own RF field to send a message to another device.

Alternative Carrier A wireless communication technology that can be used for data transfers between a Handover Requester and a Handover Selector.

Carrier Configuration Data The information needed to connect to an alternative carrier. The exact amount of information depends on the carrier technology.

Handover Requester An NFC Forum Device that begins the Handover Protocol by issuing a Handover Request Message to another NFC Forum Device.

Handover Selector An NFC Forum Device that constructs and replies to a Handover Select Message as a result of a previously received Handover Request Message or an NFC Forum Tag that provides a pre-set Handover Select Message for reading.

Negotiated Handover An exchange of NDEF messages that allows two NFC Forum Devices to agree on a (set of) alternative carrier(s) to be used for further data exchange.

NFC Forum Device A device that supports the following Modus Operandi: Initiator, Target, and Reader/Writer. It may also support Card Emulator.

NFC Tag A contactless tag or (smart) card supporting NDEF over Passive Communication..

Passive Communication A communication mode in which one device generates an RF field and sends Commands to a second device. To respond, this second device uses load modulation (i.e., it does not generate an RF field but it draws more or less power from the RF field).

Static Handover Provision of an NDEF message on an NFC Forum Tag that allows a reading NFC Forum Device to select and use alternative carriers for further data exchange.

Acronyms

ECMA	European Computer Manufacturers Association, www.ecma-international.org.
ETSI	European Telecommunications Standards Institute [www.etsi.org].
ISO/IEC	International Organization for Standardization [www.iso.org],International Electro technical Commision [www.iec.ch].
NDEF	NFC Data exchange format.
NFC	Near Field communication.
RFID	Radio-frequency identification.

Appendices

A Source code : Initializing NFC

Following code snippet shows initializing the NFC for tag detection.

```
// Get the instance of local NFC Adapter

mAdapter = NfcAdapter.getDefaultAdapter(this);

// Register an intent to receive the NFC related events

mPendingIntent = PendingIntent.getActivity(this, 0,
 new Intent(this, getClass()).addFlags(Intent.FLAG_ACTIVITY_SINGLE_TOP), 0);

// Setup an intent filter for all MIME based dispatches
// This enables tag discovery notification given to this android
// application

IntentFilter ndef = new IntentFilter(NfcAdapter.ACTION_NDEF_DISCOVERED);
try {
          ndef.addDataType("*/*");
} catch (MalformedMimeTypeException e) {
          throw new RuntimeException("fail", e);
}

// Add the NDEF format as the only format to be filtered
// through intent filter

mFilters = new IntentFilter[] {ndef};
```

B Source code : NFC tag detection

Following code snippet receives the NFC tag data when the mobile device is brought close proximit

```
@Override
public void onNewIntent(Intent intent) {
Log.i("Connection handover", "Discovered tag with intent: " + intent);

intent.setFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
boolean handled = false;

// Parse the intent
final String action = intent.getAction();
if (NfcAdapter.ACTION_TECH_DISCOVERED.equals(action) ||
NfcAdapter.ACTION_TAG_DISCOVERED.equals(action) ||
NfcAdapter.ACTION_NDEF_DISCOVERED.equals(action) ) {


// When tag is discovered, the NDEF data needs to be retrieved and parsed.
Tag nfctag = intent.getParcelableExtra(NfcAdapter.EXTRA_TAG);
   if (nfctag != null) {
// Retrieve the tag contents
typeATag = NfcA.get(nfctag);
try {
typeATag.connect();
} catch (Exception e) {
Log.w("Connection handover", "Type A Connect() failed");
}
Log.i("Connection handover", "Type A tag connected");

int saK = typeATag.getSak();

try {
typeATag.close();
}catch (Exception e ){
Log.w("Connection handover", "Type A Close() failed");
}
Log.i("Connection handover", "Type A tag disconnected");
handled = true;
   }
}

if (!handled) {
Log.w("Connection handover", "Unknown intent " + intent);
finish();
return;
}
```

C Source code : Parsing NDEF

The following code snippet shows the NDEF data parsing of Bluetooth MAC address and handover part.

```
// Parse NDEF message

Parcelable[] rawMsgs = intent.getParcelableArrayExtra
(NfcAdapter.EXTRA_NDEF_MESSAGES);
NdefMessage[] msgs;
if (rawMsgs != null) {
msgs = new NdefMessage[rawMsgs.length];
for (int i = 0; i < rawMsgs.length; i++) {
msgs[i] = (NdefMessage) rawMsgs[i];
Log.i("Parsed NDEF message ", "NDEF record: " + msgs[i]);
}
NdefRecord[] records_list = msgs[0].getRecords();
Log.i("Parsed NDEF record ", "First record: " + records_list[0]);
byte[] payload = records_list[0].getPayload();

//for (int i=0; i< payload.length; i++) {
//Log.i("Parsed NDEF record ", "payload data : " + i + " " + payload[i]);
//}

try {
byte[] bMacAddress = new byte [6];
//EE9629DE1E00

bMacAddress[0] = (byte)paylod[37];
bMacAddress[1] = (byte)paylod[38];
bMacAddress[2] = (byte)paylod[39];
bMacAddress[3] = (byte)paylod[40];
bMacAddress[4] = (byte)paylod[41];
bMacAddress[5] = (byte)paylod[42];

mBluetoothAdapter.enable();
//mBluetoothAdapter.startDiscovery();
BluetoothDevice bDev =
mBluetoothAdapter.getRemoteDevice(bMacAddress);

//ParcelUuid[] pUIid = bDev.getUuids();
//Log.e("Bluetooth test", "remote UUID " + pUIid[0]);
//mmServerSocket =
mBluetoothAdapter.listenUsingInsecureRfcommWithServiceRecord(
//        NAME_INSECURE, MY_UUID_INSECURE);
//mmServerSocket.accept();

BluetoothSocket tmp = null;
tmp = bDev.createInsecureRfcommSocketToServiceRecord(MY_UUID_INSECURE);

mBluetoothAdapter.cancelDiscovery();
tmp.connect();

try {
Method m = bDev.getClass()
.getMethod("createBond", (Class[]) null);
m.invoke(bDev, (Object[]) null);
}catch (Exception ie) {
Log.e("Bluetooth error", "Pairing  failed", ie);
}

tmp.connect();
Log.e("Bluetooth test", "Pairing  succeeded");

} catch (IOException e) {
Log.e("Bluetooth error", "Pairing  failed", e);
}
```