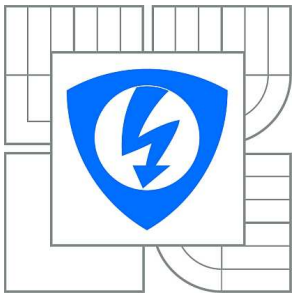

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY


FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

DEPARTMENT OF TELECOMMUNICATIONS


# TECHNOLOGIE NFC A JEJÍ ZABEZPEČENÍ

NFC TECHNOLOGY AND ITS SECURITY

DIPLOMOVÁ PRÁCE

MASTER'S THESIS


AUTOR PRÁCE — Bc. TOMÁŠ MERTLÍK

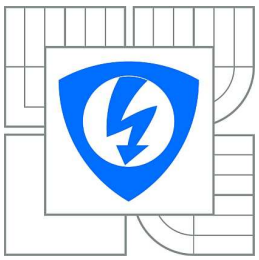
AUTHOR

VEDOUCÍ PRÁCE — Ing. MARTIN ROSENBERG

SUPERVISOR



BRNO 2013



**VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ**

**Fakulta elektrotechniky a komunikačních technologií**

**Ústav telekomunikací**

# Diplomová práce

magisterský navazující studijní obor
**Telekomunikační a informační technika**

***Student:*** Bc. Tomáš Mertlík ***ID:*** 112052
***Ročník:*** 2 ***Akademický rok:*** 2012/2013

**NÁZEV TÉMATU:**

## Technologie NFC a její zabezpečení

**POKYNY PRO VYPRACOVÁNÍ:**

Úlohou studenta je v diplomové práci popsat technologii NFC (Near field communication) a operační systém Android. Důraz bude kladen na rozbor bezpečnosti technologie a její návaznost na operační systém. Student se zaměří na možnosti využití technologie NFC pro přenos menších souborů (jednotky MB). Výstupem bude návrh aplikace, která bude umožňovat přenos libovolných dat mezi dvěma zařízeními podporující technologii NFC. Student následně navrženou aplikaci vytvoří. K dispozici jsou dva telefony Galaxy Nexus a čtečka NFC.

**DOPORUČENÁ LITERATURA:**

[1] MEDNIEKS, Z; DORNIN, L.; NAKAMURA, M. Programming Android, OREILLY, 2011. 502 S. ISBN: 978-1-449-38969-7
[2] Gallo, F. NFC Tags A technical introduction, applications and products [online], Rev. 1.3, 1 December 2011. Dostupné z URL: http://www.nxp.com/documents/other/R_10014.pdf

***Termín zadání:*** 11.2.2013 ***Termín odevzdání:*** 29.5.2013

***Vedoucí práce:*** Ing. Martin Rosenberg
***Konzultanti diplomové práce:***

**prof. Ing. Kamil Vrba, CSc.**
*Předseda oborové rady*

**UPOZORNĚNÍ:**

## ABSTRAKT

Prvním cílem této diplomové práce je popis mobilního operačního systému Android. Je zde uvedena jeho historie, architektura a verze operačního systému od počátku do současnosti. Důležitost byla také kladena na popis jeho zabezpečení. Další téma této práce je popis bezkontaktního systému NFC (Near Field Communication). Ten však navazuje na standardy staršího bezkontaktního systému RFID (Radio Frequency Identification), který je zde pro lepší přehlednost a srozumitelnost také popsán. U NFC jsou uvedeny způsoby využití, historie a technická specifikace. Ta popisuje fyzickou a linkovou vrstvu NFC, režimy přenosu dat a speciální formát pro přenos zpráv NDEF (NFC Data Exchange Format). Ten je využit pro přenos jakýkoliv dat přes NFC. Dále byl kladen důraz na rozbor všech možných útoků na NFC. V neposlední řadě jsou zde popsány způsoby a doporučení na ochranu před těmito útoky. Výstupem této diplomové práce je návrh a vytvoření aplikace, která umožňuje přenos libovolných dat mezi dvěma zařízeními podporující technologii NFC. Pro přenos objemnějších dat je v aplikaci využito spojení NFC a Bluetooth, kde se pomocí NFC přístroje spárují a data se posílají technologií Bluetooth.

## KLÍČOVÁ SLOVA

Aktivity, Android, bezpečnost Androidu, API, APK, ARM, bezpečnost NFC, DEX, Google, intent, Java, JDK, Linux, NFC, RFID, SDK, virtuální stroj.

## ABSTRACT

The first objective of this paper is to describe the Android operating system. The chapter presents history and architecture regarding to all versions of the operating system. The significant part of chapter is focused on security. The second main topic of this paper is analysing the NFC (Near Field Communication) technology. This technology is based on an older contactless system RFID (Radio Frequency Communication), so RFID is described either. The NFC technology chapter contains the description of usability, history and the definitions of physical layer, link layer and NDEF (NFC Data Exchange) data format, used for data transmitting. The next chapter is focusing on the analysis of the NFC vulnerability. It contains a possible attacks methods and solutions how to prevent them. The output of this paper is the projection and creation of an application, which allow sending an arbitrary data between two devices using the NFC technology. Additional Bluetooth technology can be used for larger files. In this case, NFC helps to create Bluetooth communication channel which is utilized for data transfer.

## KEYWORDS

Activity, Android, Android security, API, APK, ARM, NFC security, DEX, Google, intent, Java, JDK, Linux, NFC, RFID, SDK, virtual machine.

MERTLÍK, Tomáš *Popis technologie NFC a jejího zabezpečení*: diplomová práce. BRNO: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací, 2013. 116 s. Vedoucí práce byl Ing. Martin Rosenberg

# PROHLÁŠENÍ

Prohlašuji, že svou diplomovou práci na téma „Popis technologie NFC a jejího zabezpečení“ jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené diplomové práce dále prohlašuji, že v souvislosti s vytvořením této diplomové práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

BRNO ..... .....

(podpis autora)

# PODĚKOVÁNÍ

Rád bych poděkoval vedoucímu diplomové práce panu Ing. Martinu Rosenbergovi za odborné vedení, konzultace, trpělivost a podnětné návrhy k práci.


BRNO ..... .....

(podpis autora)

# OBSAH

# SEZNAM OBRÁZKŮ

# SEZNAM TABULEK

# ÚVOD

Android je poměrně mladou platformou, která pohání stovky miliónů mobilních zařízení ve více než 190 zemích po celém světě. Slovo Android v dnešní době zná už asi každý, ale málokdo ví, jak to všechno začalo. Proto je zde uveden vznik i historie operačního systému Android spolu s jeho verzemi. Dále je zde rozebrána jeho architektura, protože má zcela odlišné vlastnosti od ostatních operačních systémů. Řada popsaných součástí z architektury bude použita při tvorbě vlastní aplikace na přenos dat přes NFC (Near Field Communication). Poslední kapitola o Androidu pojednává o zabezpečení operačního systému a samotných aplikací.

Druhé téma této práce je technologie RFID (Radio Frequency Identification), protože bez znalostí této technologie by bylo těžké pochopit technologii NFC, jelikož navazuje na standardy specifikované pro RFID. Tato návaznost umožňuje zpětnou kompatibilitu těchto systémů, ale jen na pracovním kmitočtu 13,56 MHz, na kterém je NFC založeno. RFID je bezdrátový bezkontaktní systém, který používá elektromagnetické pole pro přenos dat pro identifikaci a sledování zboží.

Další část je věnována již zmíněné technologii NFC, která popisuje technické specifikace a zabezpečení. NFC je bezdrátový bezkontaktní sytém propojující dvě zařízení na krátkou vzdálenost do jednotek centimetrů pracující na kmitočtu 13,56 Mhz. Jedná se o poměrně mladou technologii, kde její využití rychle stoupá. NFC se prosazuje díky své jednoduchosti a nízké energetické náročnosti, proto se hodí do mobilních zařízení, které mají omezený zdroj energie. U technické specifikace NFC je popsána fyzická a linková vrstva, komunikační standardy, režimy přenosu dat, různé metody implementace NFC do zařízení a speciální datový formát NDEF (NFC Data Exchange Format). Poslední část této kapitoly je zaměřena na bezpečnost systému NFC a popis různých útoků. Dále jsou uvedeny způsoby zabezpečení a doporučení před popsanými útoky.

Předposlední téma se věnuje návrhu aplikace na operační systém Android pro přenos dat pomocí NFC. Bude zde využit datový formát NDEF, který podporuje jakýkoliv typ přenášeného souboru. Jelikož NFC podporuje relativně pomalé spojení a musí být zařízení pro komunikaci v těsné blízkosti, bude na samotný přenos objemnějších dat využita jiná technologie (Bluetooth nebo Wi-Fi) a pomocí NFC se přístroje spárují. Uživatel si pak bude moci vybrat jakou technologii bude pro přenos dat využívat.

Závěrečná část obsahuje popis vytvořené aplikace NFC File Transfer. Jsou zde popsány důležité funkce aplikace pro získání dat od uživatele, odesílání a příjem dat.

# 1 OPERAČNÍ SYSTÉM ANDROID

## 1.1 Úvod do OS ANDROID

Android pohání stovky miliónů mobilních zařízení ve více než 190 zemích po celém světě. Má prvotřídní platformu pro vytváření aplikací a her spojenou se službou Google Play, pro téměř okamžité šíření digitálního obsahu všem uživatelům. V návaznosti na příspěvcích open-source komunitě Linux a více než 300 hardware a software partnerů se Android rychle stal nejrychleji rostoucím mobilním OS. Android se svými partnery neustále posouvá hranice hardwaru a softwaru dopředu, aby měli uživatelé nové možnosti k vývoji OS [4].
Zajímavosti o OS Android [4]:

- Google Play obsahuje více jak 700 000 aplikací, her, elektronických knih, muziky a mnoho dalšího obsahu podporující OS Android.
- Uživatelé OS Android stahují každý měsíc více než 1,5 miliardy aplikací a her z Google Play.
- Celkový počet stažených aplikací z Google Play se odhaduje okolo 25 miliard.
- Každý den je více než 1,3 milionu nově aktivovaných zařízení po celém světě.
- Android OS má již 54% zastoupení na trhu s OS pro „chytré“ telefony, Apple iOS 36 % a Windows Phone 2 % (září 2012).
- Počet prodaných telefonů s Androidem meziročně vzrostl o 379 % na téměř 52 milionů prodaných kusů (září 2012).

Android jako takový je pod open-source licencemi, což umožňuje jednotlivým společnostem i jednotlivcům snadné úpravy v systému bez nutnosti dokupování dalších licencí či patentů. Jinak je také Android označován jako Android Open Source Project (AOSP). Android OS je založen na Linuxu a je primárně určen pro zařízení s dotykovým displejem, jako jsou smartphony, tablety, navigace a PDA. Používá procesorovou architekturu ARM (více v kap. 1.4) Při vývoji systému byl brán ohled na menší výkon přístrojů, méně dostupné paměti a hlavně na výdrž baterie [3].

Platforma se kromě linuxového jádra skládá z řady C/C++ knihoven – např. SQLite knihovny, knihovny grafického enginu, knihovny enginu webového prohlížeče, knihovny pro práci s médii, systémová libc a mnoho dalších. Dále obsahuje Dalvik Virtual Machine, který slouží pro vykonávání bytecodu, na kterém jsou postaveny vyšší vrstvy systému. Dalvik Virtual Machine není stejný jako Java Virtual Machine a používaný bytecode není Java bytecode. Aplikace pro Android se sice píší v Java kódu (.class), ale ten se převede do Dalvik kódu (.dex) pomocí aplikace dx z balíčku dodávaného Android týmem pro vývoj OS. Navíc se nejedná o čisté programování v Java kódu, programování je mírně odlišné [13].

## 1.2 Historie OS Android

Slovo Android v dnešní době zná už asi každý, ale málokdo ví, jak to všechno začalo. Android je jeden z nejpoužívanějších operačních systémů na světě. Dříve nebyla žádná volba OS do „chytrých telefonů“. Ve většině případů byl obvykle OS Symbian[1]. V krátkém časovém rozpětí si Android vydobyl velké místo na trhu.

Android Inc. byl založen 26. října 2003 v Kalifornii v městečku Palo Alto. Spoluzakladatelé byly tehdy: Andy Rubin, Rich Miner, Nick Sears, Chris White. Od začátku firmy Android Inc. všichni pracovníci pracovali v utajení, odhalili jen to, že pracují na softwaru pro mobilní telefony. V ten samý rok Andymu Rubinovi došly peníze. Steve Perlman, blízký přítel Andyho, mu přinesl 10 000 dolarů v obálce. To firmu zachránilo před zánikem. Google Inc. dne 17. srpna roku 2005 odkoupil v té době takřka neznámou firmu Android Inc. a udělal z ní svoji dceřinou společnost. Všichni zaměstnanci včetně spoluzakladatelů zůstali pracovat ve společnosti [2].

Po odkupu společnosti, tým Googlu pod vedením Andyho Rubina vyvinul platformu založenou na Linuxovém jádře a v září roku 2007 Google získal několik patentů v oblasti mobilních technologií. Odborná veřejnost začala po akvizici spekulovat, že Google chce tímto krokem vstoupit na trh „chytrých“ mobilních telefonů a chystá vydání vlastního telefonu [3].


Obr. 1.1: Oficiální logo OS Android [4].

Společnost Google Inc. začala nabízet OS Android výrobcům mobilních telefonů, aby mohla shromáždit potřebné hardwarové komponenty na výrobu jejich OS. Dne 5. listopadu 2007 bylo vytvořeno uskupení Open Handset Alliance [2]. Open Handset Alliance v dnešní době obsahuje mnoho firem např. Google, HTC, Intel, LG, Motorola, NVidia, Samsung, Texas Instruments a další. Cílem tohoto konsorcia bylo

[1] Svobodný operační systém založen na Javě pro chytré telefony. Nejčastěji běžel na mobilních telefonech Nokia, dnes je již tento OS zastaralý a jeho vývoj nejde tak rychle kupředu s rostoucími požadavky zákazníků. Zdroj: <http://cs.wikipedia.org/wiki/Symbian_OS>

[2] Jedná se o uskupení výrobců mobilních telefonů,telekomunikačních operátorů, technologických firem, které podporují vývoj operačního systému Android pro chytré telefony.

vyvinout otevřený standard pro mobilní zařízení. Ve stejný den konsorcium Open Handset Alliance odhalila svůj první produkt, mobilní platformu Android založenou na Linuxovém jádře 2.6 [1], [2].

Eric Schmidt dal při této příležitosti ve svém projevu najevo, že má Google s platformou Android velké plány: „Dnešní oznámení je mnohem ambicióznější, než pouhý Google telefon, jak spekuloval tisk v posledních několika týdnech. Naší vizí je, že platforma, kterou představujeme, bude moci být použita na tisících rozdílných telefonních modelech.“ O týden později byl vydán první Android SDK (Software Development Kit) pro vývojáře pod licencí open-source [3].


Obr. 1.2: První prototyp s Androidem [11].

Vývoj šel stále kupředu a novináři začali spekulovat o GPhonu – telefon z dílen Google (obr. 1.3).


Obr. 1.3: Jeden z údajných GPhonů [11].

První komerčně dostupný chytrý telefon byl T-Mobile G1(známý také jako HTC Dream) s Android verzí 1.0, uvolněný 22. října 2008 (v České Republice byl uveden v lednu 2009). Mobilní telefon G1 přišel s řadou novinek např.: podpora GPS funkcí, 3,1 MPx fotoaparátem, hardwarovou klávesnicí, HVGA[3] dotykovým displejem [1].

[3]Tzv. Half-Size VGA o rozlišení 480x320 pixelů. Zdroj: http://cs.wikipedia.org/wiki/Rozlišení

Obr. 1.4: První telefon s OS Android T-Mobile G1 (HTC Dream) [11].

Koncem srpna roku 2008 byl oficiálně představen Android Market[4], avšak uživatelům byl dostupný až o celé dva měsíce později – 22. října 2008. V té době obsahoval asi 35 aplikací [11].

Počet zařízení na této platformě začal prudce stoupat a ke konci roku 2009 nastoupila druhá generace zařízení s verzemi Android 2.x a výkonnými ARM procesory Qualcomm architektury Snapdragon taktovanými minimálně na 1GHz. Na začátku roku 2010 Google spolupracoval s HTC k uvolnění vlajkové lodě chytrých telefonů HTC Nexux One. Hned následovali další řady chytrých telefonů, např. společnost Samsung představila řadu telefonů Galaxy [11].

## 1.3 Verze operačního systému Android

Za posledních pár let vzniklo několik verzí OS Android. Tato kapitola znázorní jejich přehled a stručně tu bude pojednáno o nejlepších vylepšeních v každé z nich.

Jelikož se systém Android neustále vyvíjí, tak musel být zaveden standard na psaní aplikací pro konkrétní verzi OS, tzv. API (Application programming interface) level. Jedná se tedy o celočíselnou hodnotu, která označuje verzi revize programového vybavení pro kompatibilitu s OS Android verzemi. Pokud je aplikace napsaná v API level 16, nelze ji provozovat na OS Androidu nižším než je verze 4.1. Zatím co zpětná kompatibilita byla zachovaná. Když bude aplikace psaná v API level 8, lze tuto aplikaci spustit na novějším OS Android než je verze 2.2 [4].

Tabulka 1.1 znázorňuje stručný přehled vývoje OS Android, API level, datum vydání OS a podíl na trhu platný k 4. 9. 2012.

[4]Dnes známý pod názvem Google Play. Slouží především ke sdílení aplikací, her, elektronických knih, muziky a mnoho dalšího obsahu podporující OS Android. A to ve variantách placených tak i neplacených.

Tab. 1.1: Stručný přehled verzí androidu [4].

| Verze | Kódové označení | API | Datum vydání | Podíl na trhu |
|---|---|---|---|---|
| 1.0 | Apple Pie | 1 | 23. září 2008 | 0 % |
| 1.1 | Banana bread | 2 | 9. února 2009 | 0 % |
| 1.5 | Cupcake | 3 | 30. duben 2009 | 0.2 % |
| 1.6 | Donut | 4 | 15. září 2009 | 0.4 % |
| 2.0, 2.1 | Eclair | 7 | 26. října 2009 | 3.7 % |
| 2.2 | Froyo | 8 | 20. května 2010 | 14 % |
| 2.3.x | Gingerbread | 9 – 10 | 6. prosince 2010 | 57.5 % |
| 3.x.x | Honeycomb | 11 – 13 | 22. února 2011 | 2.1 % |
| 4.0.x | Ice Cream Sandwich | 14 – 15 | 19. říjen 2011 | 20.9 % |
| 4.1.x | Jelly Bean | 16 | 9. července 2012 | 1.2 % |

Jak je zřejmé z tabulky 1.1, verze OS Android mají názvy sladkých pochutin a to v abecedním pořadí podle data vydání. České překlady jsou v tabulce 1.2. Kódové označení nové verze OS Android bude tedy nejspíše začínat písmenem K.

Tab. 1.2: Překlady názvů OS Android do češtiny.

| Verze OS | Oficiální název | Český překlad |
|---|---|---|
| 1.0 | Apple Pie | Jablečný koláč |
| 1.1 | Banana bread | Banánový chléb |
| 1.5 | Cupcake | Košíček |
| 1.6 | Donut | Kobliha |
| 2.0, 2.1 | Eclair | Piškotek |
| 2.2 | Froyo | Mražený jogurt |
| 2.3.x | Gingerbread | Perníček |
| 3.x.x | Honeycomb | Medová plástev |
| 4.0.x | Ice Cream Sandwich | Zmrzlinový sendvič |
| 4.x.x | Jelly Bean | Želé bonbon |

### 1.3.1 Android 1.0 – Apple Pie

Jak již bylo řečeno o kapitolu výše, tato verze byla jako první komerčně dostupná 23. září 2008 v mobilním telefonu T-Mobile G1 (známé také jako HTC Dream). Kódové označení Apple Pie je zpětné neoficiální pojmenování. Přinášela řadu novinek do světa operačních systémů pro chytré telefony, jako jsou například [4]:

- Android Market aplikaci pro stahování a aktualizaci softwaru přes internet,
- internetový prohlížeč – podporoval přiblížení, oddálení, prohlížení HTML a XHTML stránek a více současně otevřených oken,
- přístup k emailovým serverům, podporující POP3, IMAP4 a SMTP,
- synchronizace emailů (Gmail), kontaktů (People), kalendáře (Calendar),
- Google mapy s podporou Street View využívající k navigaci GPS systém,
- podpora SMS, MMS, nultimediální přehrávač, Wi-Fi, Bluetooth,
- oznamovací akce ve stavovém řádku, hlasové vytáčení kontaktu,
- YouTube video přehrávač, alarm, kalkulačka, galerie a mnoho dalšího.

### 1.3.2 Android 1.1 – Bananna bread

Tato verze byla uvolněna 9. února 2009 zpočátku pouze pro HTC Dream. Kódové označení Banana bread je pouze neoficiální. V tuto dobu byl Android stále považován za marný pokus a byla mu předpovídána krátká budoucnost. V Google Marketu bylo něco okolo 1700 aplikací. Tato aktualizace opravovala mnoho chyb z předchozí verze a přidala pár nových vlastností například [4]:

- propracovanější Google mapy, které zobrazují recenze, obchody atd.,
- možnost uložit přílohy ve zprávách.

### 1.3.3 Android 1.5 – Cupcake

Tato aktualizace vyšla 30. dubna 2009. Obsahovala nové Linuxové jádro 2.6.27. Počet aplikací na Google Marketu se mezitím zdvojnásobil na 3543. Tato verze vylepšuje mnoho funkcí z předešlé verze a přidává několik nových funkcí [4]:

- možnost nahrávat a sledovat videa z kamery,
- nová softwarová klávesnice s automatickým dokončováním slov,
- nahrávání videí na YouTube a fotografií na Picasu přímo z telefonu,
- bluetooth — podpora A2DP (Advanced Audio Distribution Profile)[5],
- možnost automaticky připojit Bluetooth headset,
- nové widgety a složky,
- animace při přechodu mezi obrazovkami,
- rozšířena funkce kopírovat a vložit.

Obr. 1.5: Logo Android 1.5 – Cupcake [4].

[5]Tento profil definuje, jak vysoká kvalita zvuku (mono nebo stereo) může být přenášena z jednoho zařízení do druhého přes Bluetooth. Zdroj: <http://en.wikipedia.org/wiki/Bluetooth_profile>

### 1.3.4 Android 1.6 – Donut

Verze 1.6 s označením Donut vyšla 15. září 2009 s novým Linuxovým jádrem 2.6.29. Počet aplikací na Google Marketu se držel pod hranicí 4000 aplikací. S novou verzí Android 1.6 byl také představen nový chytrý telefon HTC Hero, který Androida doslova katapultoval v žebříčku popularity. Díky tomu začala OS Android brát vážně i jeho konkurence jako je Microsoft (OS Windows Phone) nebo Apple (OS Apple iOS) [11]. Dále jsou shrnuty nejdůležitější nové funkce [4]:

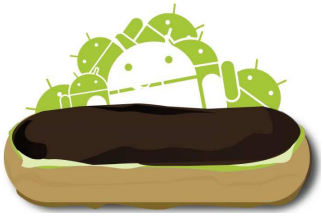
- aktualizované vyhledávání hlasem,
- vylepšení rychlosti vyhledávání a kamery,
- podpora pro rozlišení displeje WVGA[6],
- podpora pro technologie CDMA/EV-DO, 802.1x, VPN, Gesta a syntéza řeči,
- nové prostředí fotoaparátu, kamery a galerie,
- Quick Search Box – umožňuje vyhledávat historii, kontakty a na webu z domovské obrazovky.

Obr. 1.6: Logo Android 1.6 – Donut [4].

### 1.3.5 Android 2.0 – Eclair

Dne 26. října 2009 vyšla zcela nová verze Androida (Eclair) a dosud největší aktualizace. Android poskočil na verzi 2.0 a obsahoval stejné Linuxové jádro, jako verze 1.6. Android Market v tu dobu obsahoval bezmála 14 000 aplikací. V prosinci téhož roku bylo do světa vypuštěno SDK verze 2.0.1 a Android Market pokořil hranici 20 000 aplikací. Dne 5. ledna 2010 vyšel nový mobilní telefon Google Nexus One. Říkalo se mu také „Zabiják iPhonu“. Avšak neuspěl a po 6 měsících prodeje byl odstaven a nadále se prodává pouze jako vývojářský telefon. V březnu uvedla společnost Samsung svoji vlajkovou loď, první telefon ze série Galaxy. Pouhého půl roku po uvedení prodal Samsung celosvětově přes 5 miliónu kusů. Galaxy S také vyhrál cenu za Evropský smartphone roku, udělovanou asociací EISA [11].
Následuje přehled nových funkcí ve verzi Android 2.0 [4]:

- zdokonalené uživatelské prostředí,
- podpora pro více velikostí a rozlišení displeje,
- větší rychlost hardwaru (HW akcelerace),
- podpora pro Microsoft Exchange,
- podpora přisvětlovací diody,
- digitální zoom (fotoaparát),
- podpora pro Bluetooth 2.1,
- animované tapety na domovské stránce.



Obr. 1.7: Logo Android 2.0 – Eclair [4].

[6]Tzv. Wide VGA o rozlišení 800x480 pixelů. Zdroj: <http://cs.wikipedia.org/wiki/Rozlišení>

### 1.3.6 Android 2.2 – Froyo

Verzi s označením Froyo zná už asi každý. Tato aktualizace vyšla 20. května 2010 a přinesla řadu novinek. Obzvlášť největší novinka je začlenění JIT (Just-in-time)[7] kompilátoru, díky kterému aplikace běží až 5x rychleji než předtím. Tato verze OS Android měla již nové Linuxové jádro, konkrétně verzi 2.6.32. V Google Marketu bylo okolo 97 000 aplikací a počet stažených aplikací již přesáhlo 1 miliardu. Čtvrtletní čísla prodeje Androida poprvé přesáhly prodeje iPhonu. V červenci téhož roku byl oznámen první tablet s Androidem 2.2, byl to Toshiba SmartPad [14].

Koncem srpna se objevila první velká žaloba od společnosti Oracle, která koupila společnost Sun Microsystems a tím také práva na jazyk Java, který Android využívá pro běh aplikací. Oracle kromě finanční kompenzace od Google také požaduje, aby „všechny kopie děl souvisejících s Javou byly zabaveny, zničeny nebo jinak odstraněny“. Firma Oracle naštěstí neuspěla a po několika měsících soudních sporů navíc Oracle musel Googlu proplatit všechny soudní výlohy [15].

Koncem léta společnost Google spustila nový projekt s názvem Google App Inventor. Jedná se o programování jednoduchých aplikací přes webové rozhraní. Programování je velice jednoduché a zvládnou ho i úplní začátečníci. Hlavní výhoda je ta, že je vše přes internet a není potřeba nic instalovat. Přístupnost je pak odkudkoliv. Při testování aplikací lze takto vytvořenou aplikaci poslat ihned do mobilního telefonu [12].

V září Samsung oznamuje svůj Galaxy Tab, HTC začal prodávat Desire HD a T-Mobile uvádí další telefon pojmenovaný T-Mobile G2. V té době se OS Android vyšplhal na druhý nejprodávanější systém světa, prvním je OS Symbian, ale jeho podíl klesá rychle dolů. V říjnu Google Market překročil 100 000 aplikací a zároveň se OS Android stává nejpopulárnější systém na světě, předběhl i Apple iOS [11]. Následuje souhrn nových vlastností ve verzi 2.2 [4]:

- lepší správa paměti RAM, podpora Flash 10.1,
- možnost instalovat aplikace na paměťovou kartu.
- kompilátor JIT (Just-in-time), tzv. Dalvik Turbo,
- více nastavení fotoaparátu a kamery,
- možnost vytvořit z telefonu WiFi hotspot, nebo sdílet internetové připojení přes kabel USB,
- nové režimy — „režim v autě“ a „noční režim“,
- přidána podpora pro OpenGL ES 2.0[8],
- více nastavení fotoaparátu a kamery.

Obr. 1.8: Logo Android 2.2 – Froyo [14].

[7] Speciální metoda kompilátoru kódu, který využívá různé techniky pro urychlení běhu programů přeložených do mezikódu. Zdroj: <http://cs.wikipedia.org/wiki/JIT>

[8] Standard, který specifikuje multiplatformní rozhraní pro tvorbu počítačové grafiky a aplikací.

### 1.3.7 Android 2.3 – Gingerbread

Tato nejznámější verze OS Android spatřila světlo 6. prosince 2010 a přidala nesčetně vylepšení, oprav a hlavně výrazné zrychlení celého systému díky novému Garbage Collectoru[9]. Gingerbread existuje ve verzích 2.3 – 2.3.2 (API level 9) a 2.3.3 – 2.3.7 (API level 10) a pohání ho nové Linuxové jádro 2.6.35. Počet aplikací na Google Marketu tehdy překročil 130 000 a prudce tento trend stoupá. Gingerbread se řadí mezi nejpoužívanější OS Android a i v dnešní době má 57% zastoupení na trhu [2].

Pár týdnů po vydání Gingerbreadu se na internetu objevuje první malware na OS Android s označením Geinimi. Ten se chová jako botnet a pokud infikuje mobilní telefon tak je schopný odcizit spoustu osobních dat. Geinimi byl rozšířen hlavně v Číně. Podle průzkumů se v letech 2009 – 2010 zvýšilo množství útoků o 33 % a s růstem celé platformy se dá očekávat ještě prudší nárůst v následujících letech [15]. Nové funkce v OS Android 2.3 jsou [1]:

- změna z YAFFS na ext4 na novějších zařízeních[10],
- vylepšená podpora pro nativní vývoj kódu,
- podpora kodeků WebM/VP8 přehrávání videa, AAC audio kódování
- podpora Near Field Communication (NFC),
- nativní podpora pro SIP a VoIP telefonie,
- vylepšený Garbage Collector,
- nativní podpora pro více senzorů (např. gyroskopy a barometry), dokonalejší správa napájení,
- hlasová a video komunikace pomocí Google Talk,
- Google Maps 5 s 3D, správce stahování,
- rozšířená funkce kopíruj/vlož.

Obr. 1.9: Logo Android 2.3 – Gingerbread [14].

### 1.3.8 Android 3.0 – Honeycomb

Android 3.0 Honeycomb je verze platformy Android z 22. února 2011, která je od základu navržena pro zařízení s vetší velikostí obrazovky, zejména pro tablety. Zavádí nové „holografické“ prostředí a staví na tom co mají lidé rádi z Androidu – pokročilý multitasking, vylepšené oznamování akcí, widgety a mnoho dalšího. Kromě toho je OS Android 3.0 také speciálně navržen tak, aby dal vývojářům nástroje a funkce, které potřebuje k vytvoření skvělých aplikací. Tato verze OS má vysoce výkonné 2D

[9] Část programovacího jazyka, která má za úkol automaticky určovat nepoužívané části paměti a připravit je na další použití.

[10] YAFFS – Yet Another Flash File System. Jedná se o open-source souborový systém. Zdroj: <http://www.yaffs.net/>

ext4 – fourth extended filesystem. Je to žurnálovací souborový systém vyvinutý pro Linux. Zdroj: <http://cs.wikipedia.org/wiki/Ext4>

a 3D prostředí díky vylepšenému OpenGL renederu a novému grafickému 3D enginu s názvem Renderscript. Android Honeycomb je ve verzích 3.0 – 3.2.6. Linuxové jádro poskočilo na verzi 2.6.36. Google Market byl přejmenován na Android Market a v té době přesahoval hranici 170 000 aplikací [4].

V této době byl Android postihnut další žalobou. Tentokrát to byla patentová válka se společností Microsoft, kde Microsoft uvádí výčet několika desítek používaných patentů v OS Android bez schválení licenčního ujednání. Po dohodě Microsoftu s velkými firmami podporující OS Android ve svých zařízeních, jako jsou například HTC, Samsung, LG, Wistron a další, musí zaplatit z každého prodaného výrobku částku 7,5 – 15 dolarů. Což znamená, že při velkém prodeji zařízení s OS Android Microsoft obdrží až stovky miliónů jen za svoje patenty a to je mnohem více něž Microsoft vydělá na svém produktu Windows Phone [16].
Následuje výčet novinek ve verzi Honeycomb [4]:

- podpora pro multi-core procesory,
- schopnost šifrování uživatelských dat,
- filesystem v uživatelském prostoru (FUSE)[11],
- multitasking s rychlým přepínáním aplikací,
- podpora 3G a 4G tablety,
- hardwarová akcelerace,
- Podpora pro HTTP proxy,
- podpora USB zařízení,
- nová verze Google Books.

Obr. 1.10: Logo Android 3.0 – Honeycomb [14].

### 1.3.9 Android 4.0 – Ice Cream Sandwich

Po necelém roce vylepšování verze Gingerbread byla uvolněna zcela nová platforma OS Android 4.0 na trh dne 19. října 2011 se zcela novým Linuxovým jádrem 3.0.1. Verze Ice Cream Sandwich přináší vytříbené, jednotné uživatelské prostředí pro telefony, tablety a další. Tato nová platforma navazuje na to, co mají lidé nejvíce rádi na Androidu např.: efektivní multitasking, bohaté oznamování, přizpůsobení domovské obrazovky, různá velikost widgetů a mnoho dalšího [4].

Android Market v té době spustil své webové stránky pro počítače a disponoval s více jak 319 000 aplikací. Stačilo se přihlásit, vybrat si aplikaci a dát instalovat. Do telefonu se začala stahovat aplikace a posléze se i nainstalovala. Počet stažených aplikací byl okolo 7 miliard [2].

[11] FUSE – Filesystem in Userspace je open source modul pro jádra unixových OS, umožňující vytvářet vlastní souborové systémy bez potřeby privilegovaného uživatele. Zdroj: <`http://fuse.sourceforge.net/`>.

Ice Cream Sandwich přináší revoluční novinku ve sdílení informací – Android Beam. Je to funkce, která umožňuje uživatelům okamžitě sdílet informace o aplikacích, které používají jen tím, že se přiblíží dvěma telefony s podporou NFC (Near field communication). Pokud jsou tyto zařízení od sebe v rozsahu několik centimetrů, systém zřídí NFC spojení a zobrazí nabídku ke sdílení informací [4].
Dále jsou sepsány novinky ve verzi 4.0 [1]:

- zlepšení grafiky a funkcí Bluetooth,
- Data Usage – kontrola limitu datového připojení,
- Wi-Fi Direct – propojení zařízení bez hotspotu,
- vylepšené hlasové ovládání a překlad do textu,
- VPN klient s podporou L2TP a IPSec protokolů,
- podpora video záznamu ve Full HD (1080p),
- podpora stylusu a myši,
- Face Unlock – odemknutí zařízení pomocí rozpoznávání obličeje.

Obr. 1.11: Logo Android 4.0 – Ice Cream Sandwich [14].

### 1.3.10 Android 4.1 – Jelly Bean

Doposud nejnovější na trhu je méně známá verze 4.1 nazývaná jako Jelly Bean, která vyšla 9. července 2012 na novém Linuxovém jádře 3.0.31. Tato verze se označuje jako nejrychlejší a nejplynulejší verzí OS Android. Díky vylepšení funkce Vsync (funkce na synchronizaci obrazu) v celém systému vše probíhá synchronizovaně po dobu 16 ms (60 snímků za sekundu), takže rámce na obnovu obrazu už nemohou přicházet dříve či později. Tento OS přidává funkci trojitého vyhlazování v grafickém procesoru pro více konzistentní vykreslování, které dělá hladší hrany a přechody v obraze. Dále snižuje reakci dotykového displaye díky predikci pohybu prstů v době aktualizace obrazovky. Byl přidán speciální nástroj zvaný SysTrace, který sleduje, co se v systému děje a dokáže říct, kde systém či aplikace vázne [4].

Android Market byl přejmenován 6. března 2012 na Google Play a obsahoval více jak 650 000 aplikací. Počet stažení překročil 22 miliard [2].
Shrnutí novinek verze Jelly Bean [1]:

- vysoké rozlišení fotek u kontaktů,
- podpora arabštiny a hebrejštiny,
- překládání z hlasu do textu funguje offline,
- hlasové vyhledávání a hlasové odpovědi,
- podpora zařízení pro vstup Braillova písma,
- Bluetooth přenos dat pro Android Beam,
- vícekanálový zvuk,
- Google Chrome a šifrování.

Obr. 1.12: Logo Android 4.1 – Jelly Bean [14].

## 1.4 ARM procesory

ARM (Acron RISC Machine, dnes již název Advanced RISC Machine) je značka procesoru vyvíjená ve Velké Británii firmou ARM Limited od roku 1980. Dříve firma ARM Limited vyráběla a distribuovala procesory sama, ale v nynější době se soustředí především jen na jejich vývoj a vybírá licence od výrobců hardware za jeho použití. Nejznámější výrobci procesorů jsou firmy: Texas Instruments, Marvell, Apple, LG, Nintendo, Nvidia, Sony, Samsung, Yamaha a mnoho dalších [6].

Procesory ARM je v dnešní době možné najít takřka ve všech odvětvích spotřební elektroniky např: herní konzole, směrovače, pevné disky, multimediální přehrávače, kalkulačky, PDA a hlavně skoro ve všech mobilních telefonech. Prostě všude tam, kde je zapotřebí rychlost bez složitého chlazení a mobilita s menšími energetickými nároky. ARM procesor začínal s frekvencí 8 Mhz na jedno jádro a v nynějších dobách přes řadu vývojů se posunul až na čtyř jádrový procesor s taktovacím kmitočtem 2500 Mhz [6].

Tato architektura procesorů způsobila revoluci v informačních technologiích, protože používá redukovanou instrukční sadu procesoru (RISC) spolu s 16 bitovou nebo 32 bitovou šířkou instrukce (například pro levné kalkulačky stačí pouze 4 nebo 8 bitová šířka instrukce). Instrukční sada je omezena na nezbytné minimum jednoduchých instrukcí, proto také procesor rychleji reaguje na vstupní požadavky. Např. u RISC procesorů neexistuje instrukce na násobení, ale realizuje se softwarově pomocí jednoduchých instrukcí sčítání a bitových posuvů. Mikroinstrukce jsou již hardwarově implementovány na procesoru, čímž se velmi výrazně zvýší rychlost jejich provádění. Délka provádění jedné instrukce je vždy jeden cyklus (bitová délka instrukcí je vždy stejná). Pro řetězení instrukcí využívá tzv. Pipelining, kde se rozděluje instrukce mezi různé části procesoru a tím je dosažena možnost zpracování více instrukcí najednou [6].

Pro mobilní zařízení, kde je kladen důraz na výdrž baterie, si procesory dokážou sami řídit taktovací kmitočet a přísun energie, aby nespotřebovávali velké množství energie při jejich nečinnosti nebo malém vytížení. Také se dokážou přepnout do tzv. „režimu spánku“, kde sníží svůj taktovací kmitočet natolik, že pouze čekají na přerušení. Pokud přerušení přijde, procesor se „probudí“ a začne obsluhovat přerušení. Tohoto jevu se hodně využívá například u WiFi adaptéru [6].


Obr. 1.13: ARM Cortex M4F procesor od firmy Texas Instruments [6].

## 1.5 Android-x86 projekt

Již od počátku OS Android byla snaha, aby byl tento OS multiplatformní. Jelikož je tento systém šířen pod volnou licencí, tak začaly vznikat různé odvětví tohoto OS. Jedna z mnoha se věnuje možnosti mít OS Android na běžném procesoru řady x86 (32 bitové Intel nebo AMD), protože je OS Android stavěný pouze na procesorovou architekturu ARM (popsáno v kapitole 1.4). Tento projekt vznikl 6. 7. 2009 na verzi Android 1.5 (Cupcake) a prodělal již mnoho změn. V současné podobě je tento projekt ve verzi Android 4.0 (Ice Cream Sandwich) a přispívá do něho mnoho lidí po celém světě. Bohužel na tento OS není ještě taková podpora ovladačů od výrobců hardwaru jako např. pro Microsoft Windows, tak není zaručena kompatibilita na všech strojích poháněných procesorem z řady x86 a vývoj jde bez této podpory pomalu [7].
Následuje výpis platforem, na kterých je projekt Android x86 odzkoušen a je na nich otestován:

- ASUS Eee počítače/notebooky,
- Viewsonic Viewpad 10,
- Dell Inspiron Mini Duo,
- Samsung Q1U,
- Viliv S5,
- Lenovo ThinkPad x61 Tablet.


Obr. 1.14: Logo projektu Android x86 [7].

Jak je již zřejmé z výpisu podporovaných zařízení, hlavní snaha tohoto projektu je přebudovat OS na platformu rodiny procesorů Intel Atom, protože dosahuje vysokých výkonů s velmi nízkou spotřebou. Intel Atom má naprosto odlišnou instrukční sadu oproti ARM. Hlavní odlišnost je v tom, že Intel Atom má komplexní instrukční sadu (CISC) a ARM ji má pouze redukovanou (RISC). Proto na OS Android běžící na x86 se musí přebudovat kernel (jádro systému) s Dalvik Virtual Machine a ostatní zůstává stejné jako doposud. Tím je zaručen standardní vývoj aplikací a OS.

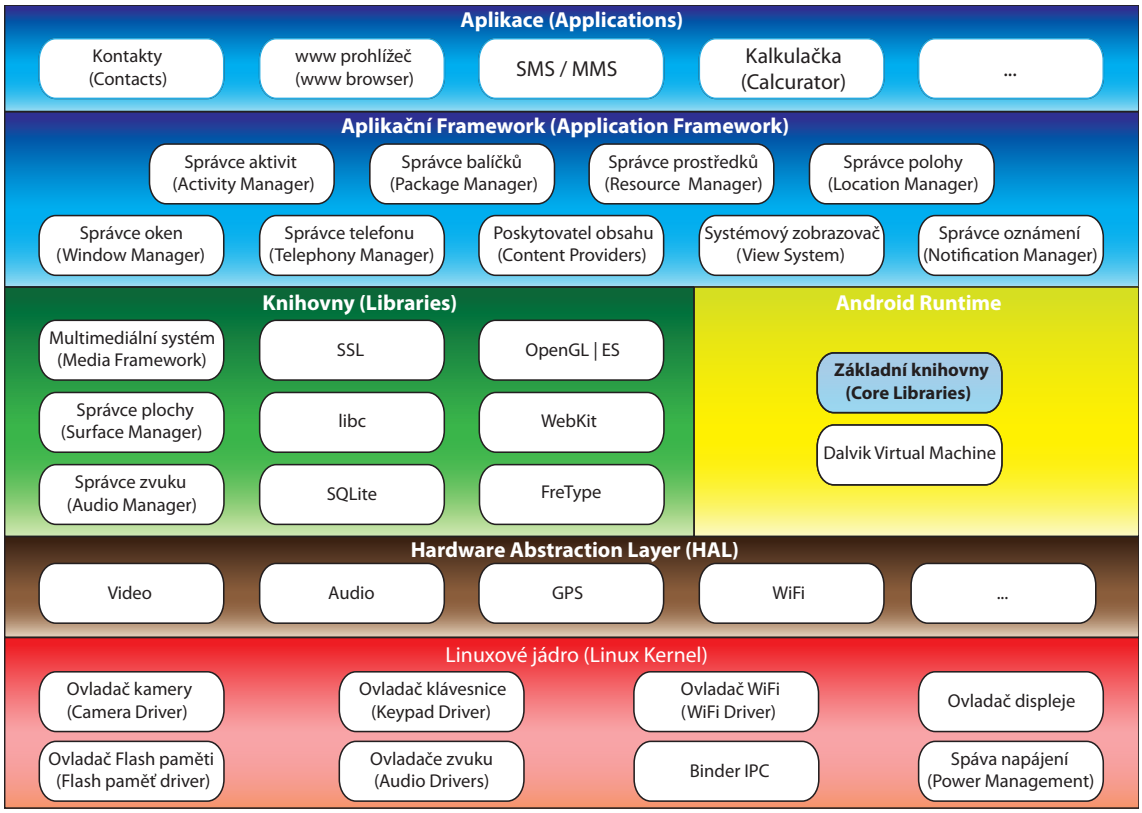
Zcela odlišný návrh přidala společnost Intel, která se snaží prosadit na trhu „chytrých“ telefonů a tabletů. Vydala svůj první chytrý telefon 27. 2. 2012 s označením Santa Clara od firmy Orange s Androidem 2.3 (Gingerbread). Tento telefon disponoval procesorem x86 Intel Atom taktovaný na frekvenci 1,6 Ghz s Intel Burst Performance [12] navýšení kmitočtu na 2 Ghz a podporou Hyper-Threading [13]. Výkonem dosahuje na dnešní procesory ARM. Intel také uvedl verze 2.3.3, 4.0 a 4.1 pro platformu x86 Atom volně ke stažení pro vývojáře na psaní a ladění aplikací [9].

[12] Dynamické navýšení taktu procesoru při požadavku na větší výkon používaný firmou Intel.
[13] Tato technologie vytváří z jednoho jádra procesoru více virtuálních pro rychlejší obsluhu paralelních vláken. Opět od firmy Intel.

## 1.6 Architektura OS Android

Na následujícím obrázku 1.15 je znázorněno celkové schéma architektury OS Android. Důležité je si uvědomit použití barev v obrázku, kde vrstvy červená s hnědou jsou psány v jazyku C, žlutá se zelenou v jazyku C/C++ a modrá v jazyku JAVA. Každá vrstva má soje speciální funkce v OS. Nejjednodušší vysvětlení vrstveného modelu je následující: určitá vrstva poskytuje služby vrstvám nad ní a přijímá požadavky na služby z vrstev nižších.

Všechny vrstvy a jejich komponenty budou popsány pro přehlednost v samostatných kapitolách.




Obr. 1.15: Architektura OS Android [8].

### 1.6.1 Linuxové jádro (Linux Kernel)

Jedná se o nejnižší vrstvu architektury operačního systému z obrázku 1.15, která tvoří abstraktní vrstvu mezi používaným hardwarem a zbytkem softwaru ve vyšších vrstvách. Jádro tedy obsahuje nezbytně nutné ovladače hardwaru. Ovladače jsou programy, které řídí komunikaci s hardwarem [8].

Android je sice postaven na Linuxovém jádru (Linux kernel), ale Android není Linux. Není zde nativní X Window System, který v Linuxu vytváří grafické uživatelské prostředí. Chybí zde také kompletní standardní knihovna GNU C (glibc), kterou v systému Linux používaly programy psané ve standardním jazyku C/C++. Přístup do systému je zde omezen pouze na uživatelský přístup (user-space) a to přes aplikační framework. Na většině zařízení však lze získat superuživatelské oprávnění (tzv. root) a díky tomu je možné plně využít možnosti, které poskytuje linuxové jádro. Android začínal na jádru ve verzi 2.6.24, nyní je již ve verzi 3.0.31 [40].
Otázkou však je, proč bylo zvoleno právě Linuxové jádro? [8]

1. Poskytuje skvělé řízení paměti a procesů.
2. Podpora sdílených knihoven.
3. Známé prostředí pro vývojáře, základ na standardním jazyku C/C++.
4. Poskytuje propracovaný bezpečnostní model.
5. Osvědčený model ovladačů, který poskytuje skvělou abstraktní vrstvu mezi hardwarem a softwarem.
6. Open-source model, který vylepšuje milióny lidí po celém světě a navíc je pod volnou licencí.

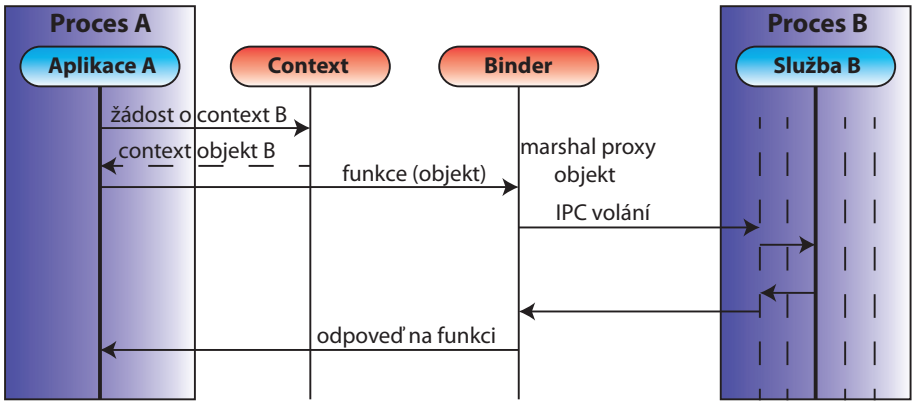
Vylepšení Linuxového jádra pro Android jsou popsány níže.

**Alarm** – používá se pro plánování událostí nebo služby v nastaveném čase či intervalu i při stavu procesoru v režimu „spánku“. Použití např.: budík, oznámení z kalendáře, spuštění aplikace v daný okamžik, atd. [8].

**Ashmem** (Android shared memory) – poskytuje sdílení paměti samostatným aplikacím. Pokud více aplikací využívá stejnou oblast paměti, nebude odstraněna dokud ji všechny aplikace řádně neopustí [8].

**Binder IPC** (Binder Inter-process communication) – zařizuje meziprocesní komunikaci, aby mohly aplikace a služby běžet v samostatných procesech se svoji přidělenou pamětí, přičemž musí mezi sebou být schopni komunikovat a sdílet si svoje prostředky. Binder IPC je odlehčená varianta RPC (Remote Procedure Communication). V Linuxu se tato mezi procesní komunikace řeší pomocí signálů (Signal), rour (Pipe), semaforu (Semaphore) a soketu (Socket) [8].

Komunikace probíhá dle obrázku 1.16 následovně. Každý proces se při vytváření musí registrovat u Service Manageru. Následně od něj obdrží tzv. context objekt, který obsahuje rozhraní pro globální informace o aplikačním prostředí. Řekněme, že aplikace A potřebuje komunikovat se službou B, proto A vyšle žádost o context objekt pro B. Tím se A dozví o tom, co všechno B umožňuje. Následuje volání vybrané funkce od A pro B. Tuto žádost zachytí tzv. Marshal proxy objekt z Binderu. Je nutno podotknout, že funkce předává parametry odkazem a tím není nutno vytvářet kopii objektu. Na obrázku je služba B dělena do více vláken, kde Binder zná rozsah použitých vláken. Pomocí instance IPC (mezi procesní komunikaci) kontaktuje




Obr. 1.16: Binder IPC – komunikace mezi procesy [8].

službu B. Nějaké vlákno ze služby B obdrží příchozí volání. Toto vlákno přepošle požadavek tam kde bude požadavek obsloužen. Po vyřízení požadavku, se ze služby B vrací postupně odpověď na funkci až do aplikace A [10].

**Správa napájení** (Power Management) – postaven na standardu správa napájení z Linuxu. Pokud části zařízení nejsou používány, tak je vypne nebo uspí, aby se šetřila energie baterie. Aplikace se dotazují správy napájení zda-li může pro ně zařídit požadované prostředky. Při ukončení aplikace se dá vědět správě napájení a ta je uvolní. Například pokud není od žádné aplikace žádost na procesor, tak se procesor „uspí“, aby nebral zbytečnou energii [8].

**Low Memory Killer** – jelikož je mobilní zařízení s nedostatkem paměti RAM, musí existovat mechanizmy jak tuto paměť efektivně využít. Při nedostatku paměti tento modul odstraní z paměti aplikace, které nebyly dlouho použity nebo ty, které mají nízkou prioritu. Proto má název Killer (zabiják) [8].

### 1.6.2 HAL (Hardware Abstraction Layer)

Tato vrstva specifikuje proprietární ovladače výrobců zařízení jako jsou např.: WiFi, GPS, audio čip, video čip, atd. Oddělená vrstva je díky politice Linuxu, který je pod licencí open-source a to se výrobcům hardwaru nelíbilo. Proto byla zavedena tato mezivrstva, která by jinak byla integrována přímo do jádra Androidu [8].

### 1.6.3 Knihovny (Libraries)

Zde jsou specifikovány nativní knihovny pro OS Android, které jsou napsány v jazyku C/C++. Přistupuje se k nim pomocí aplikačního frameworku.
Patří sem zejména následující knihovny [40], [8], [4]:

**Multimediální systém** (Media Framework) – je založen na platformě PacketVideo, kde jeho knihovny zajišťují podporu pro přehrávání a nahrávání mnoha populárních audio a video formátů. Obsahuje podporu pro MPEG4, H.264, MP3, AAC, AMR, JPG, PNG a mnoho dalších.

**Správce plochy** (Surface Manager) – je zodpovědný za tvorbu grafického složení plochy v paměti systému od více aplikací najednou, které běží v odlišných procesech pomocí Binder IPC. Lze kombinovat 2D a 3D prostředí. Podporuje OpenGL ES (knihovny pro 2D a 3D grafiku) a 2D hardwarovou akceleraci. Používá systém Double-buffering[14] s využitím page-flip[15]. Tím se docílí toho, že se nikdy nezačne vykreslovat necelá plocha. Umí prokládat okna z různých aplikací. Například pokud je spuštěna aplikace, ale na pozadí je stále vidět síla signálu telefonu, kapacita baterie a oznamovací řádek.

**Správce zvuku** (Audio Manager) – přehrává veškeré zvuky ze systému, her a ostatních aplikací pomocí Binder IPC. Dokáže rozdělovat toky na jednotlivé výstupy (Bluetooth, reproduktor, sluchátka) nebo je umí slučovat do jednoho toku.

**SSL** (Secure Sockets Layer) – protokol, který poskytuje zabezpečení komunikace pomocí šifrování a autentizace komunikujících stran.

**Bionic libc** – vlastní knihovna pro C kompilátor (libc). Implementace odvozena z BSD (Berkeley Software Distribution) kódu standardní knihovny C. Jelikož se libc spouští u každého procesu, tak byla snaha odebrat nepotřebné funkce a tím zmenšit i jeho velikost. Bionic libc neobsahuje například podporu pro výjimky (Exception) z C++ a další věci, které OS Android nevyužije.

**SQLite** – je odlehčená relační databáze, která ukládá potřebná data do telefonu v definovaném pořádku. V Androidu ji používá většina aplikací jako jsou kontakty, sms/mms atd. Podporuje standardní SQL příkazy, které jsou kompatibilní se standardem SQL92.

**OpenGL | ES** (Open Graphics Library) – hardware mobilních telefonů dnešní doby je dostatečně výkonný pro provozování náročných multimediálních a grafických systémů. Na mobilních zařízeních byla vytvořena vlastní verze grafického standardu OpenGL nazývaná OpenGL | ES (for Embedded Systems).

**WebKit** – jádro open source webového prohlížeče od firmy Apple, který je využíván ve většině prohlížečích. Vykresluje www stránky stejně jako na normálním počítači. Podporuje HTML, CSS, Javascript, AJAX (Asynchronous JavaScript and XML) atd.

**FreeType** – knihovna pro rendering bitmapových a vektorových fontů napsaná v jazyku C.

[14]Technika na vykreslení plochy. Má dvě vyrovnávací paměti. Do jedné se kreslí, jiná se zobrazuje.

[15]Technika na rychlé hardwarové přepínání vyrovnávacích pamětí.

Zdroj: <`http://en.wikipedia.org/wiki/Multiple_buffering`>

### 1.6.4 Aplikační framework (Application framework)

Aplikační framework je bohaté prostředí, které poskytuje širokou škálu služeb, které pomáhají vývojářům k ulehčení vývoji aplikací. Na této vrstvě lze najít mnoho knihoven napsaných v jazyku JAVA speciálně upravené pro OS Android.
Základní sada služeb zahrnuje především [40], [8], [4]:

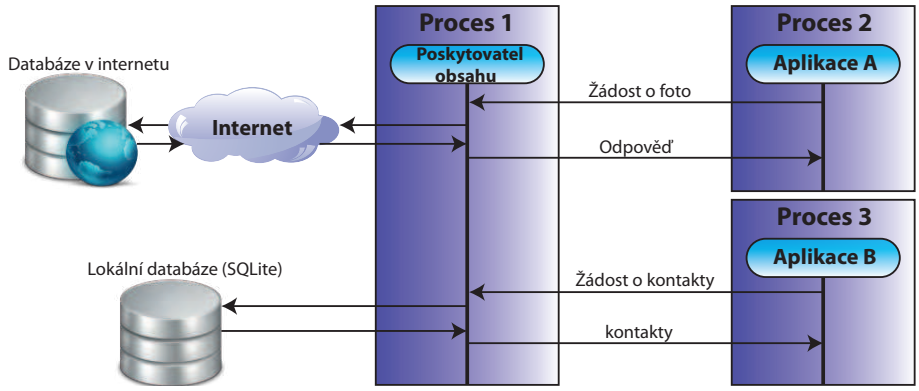
**Správce aktivit** (Activity manager) – řídí činnost životního cyklu aplikací, pro lepší vysvětlení je zahrnut do samostatné kapitoly 1.6.7.

**Správce balíčků** (Package manager) – slouží pro vyhledávání různých druhů informací souvisejících s aplikacemi, které jsou aktuálně nainstalovány v zařízení.

**Správce oken** (Window manager) – řídí umístění a vzhled oken v rámci systému. Je zodpovědný za pohnutí a změnu velikosti okna.

**Správce prostředků** (Resource manager) – eviduje všechny zdroje potřebné k aplikaci. Jako jsou např.: přídavné soubory, obrázky, atd. Díky tomu lze udělat aplikaci pro více jazykových mutací.

**Poskytovatel obsahu** (Content providers) – poskytuje sdílení dat mezi aplikacemi. Ve výchozím nastavení Android OS provozuje každou aplikaci samostatnou, aby všechny údaje, které patří aplikaci byly zcela odděleny od ostatních aplikací. Proto byl zaveden poskytovatel obsahu, který hledá data na centrálním úložišti, ať už se jedná o nějakou lokální databázi uloženou na telefonu nebo vzdálenou databázi uloženou na internetu, a zpřístupní je více aplikacím.




Obr. 1.17: Ukázka komunikace aplikací s poskytovatelem obsahu [40].

Například centrální úložiště kontaktů, pokud by nebylo, tak by jednotlivé aplikace měly svoje kontakty různé a zcela jen ve své moci. Takto jsou kontakty přístupné pro jakoukoliv jinou aplikaci, která je používá (např.: sms/mms, správce kontaktů, email klient atd.). Oddělení dat od uživatelského rozhraní nabízí možnost nahrazení výchozích aplikací alternativními. Toto ale neplatí jen pro textovou formu, ale i pro obrázky, muziku, video a další.

Poskytovatel obsahu má relativně jednoduché rozhraní se standardními metodami SQL (insert(), update(), delete(), query()). Oddělení dat od uživatelského rozhraní nabízí větší flexibilitu. Navíc i pro vývojáře aplikací je lepší použít poskytovatele obsahu, než psát svůj vlastní kód. Dokonce i widgety používají poskytovatele obsahu, přes kterého sdílí data a funkce s rodičovskou aplikací.

**Systémový zobrazovač** (View system) – je odpovědný za organizaci obsahu obrazovky. Určuje rozložení prvků na obrazovce tzv. layout, například na jakých souřadnicích je umístěno tlačítko.

**Správce polohy** (Location Manager) – pomocí GPS (Global Positioning System) nebo mobilních stanic určuje polohu uživatele a sdílí ji aplikacím.

**Správce oznámení** (Notification manager) – umožňuje všem aplikacím zobrazit vlastní upozornění ve stavovém řádku.

**Správce telefonu** (Telephony manager) – poskytuje přístup k informacím o telefonních službách a komunikuje se SIM (Subscriber Identity Module) kartou.

### 1.6.5 Android Runtime

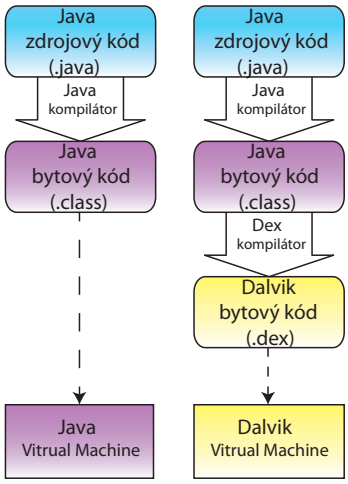
Tato vrstva obsahuje dvě nejdůležitější komponenty Dalvik Virtual Machine (dále jen Dalvik VM) a základní knihovny Java (Core libraries). Vznik Dalvik VM je hned ze dvou důvodů. První důvod jsou licenční práva na jazyk Java, kde jeho knihovny jsou volně šiřitelné, ale Java VM není. Druhý důvod je optimalizace virtuálního stroje pro mobilní zařízení [3].

**Základní knihovny** (Core libraries) – obsahuje základní knihovny programovacího jazyka Java. Obsah knihoven je přebrán z platformy Java SE (Standard Edition), akorát postrádá knihovny pro uživatelské rozhraní (Swing a AWT), které jsou nahrazeny knihovnami uživatelského rozhraní pro Android. Dále tyto knihovny obsahují například knihovny Apache pro práci se sítí [3], [40].

**Dalvik Virtual Machine** – typ virtuálního stroje vyvíjen od roku 2005, který je používán v zařízeních s OS Android ke spuštění aplikací a služeb. Je optimalizován pro nízký výpočetní výkon a nízké paměťové nároky [3].

Pro OS Android se aplikace píší v Java kódu, jak již bylo řečeno dříve, ale Dalvik VM tento kód nepodporuje. Na rozdíl od Java Virtual Machine, Dalvik nepoužívá soubory .class, ale místo toho používá soubory s příponou .dex (Dalvik EXetuble), které vznikají ze souborů .class při použití kompilátoru DX. Kompilátor DX se nachází v balíčku SDK pro Android [40]. Obrázek 1.18 ilustruje toto srovnání.

Ve srovnání velikostí bytekódu class a bytekódu dex, je dex mnohem úspornější, protože hledá duplicitní kód v class souborech a místo duplicity ho nahradí pouze ukazatelem. Také slučuje více class souborů do jednoho DEX souboru. Díky virtuálnímu stroji Dalvik jsou aplikace přenositelné na všechny telefony podporující




Obr. 1.18: Rozdíl mezi Java VM a Dalvik VM [40].

OS Android. To znamená, že nezáleží na použitém hardwaru. Pro každý proces je spuštěna jedna instance Dalvik VM [40].

Novinkou v Dalvik VM bylo použití JIT (Just In Time) kompilátoru, který využívá různé techniky pro urychlení běhu programů přeložených do mezikódu. Program využívající JIT se při spuštění analyzuje a převede se do nativních[16] instrukcí fyzického procesoru daného zařízení. Tato kompilace probíhá postupně přímo za běhu programu a tím přináší zrychlení provádění programu. Negativní je jen prodleva JIT kompilátoru, kterou stráví při překladu do nativního kódu a proto se překládají jen často volané úseky programu [4].

### 1.6.6 Applications (Aplikace)

**Aplikace** (Applications) – nejvyšší vrstva z modelu. Zde jsou zahrnuty všechny aplikace a hry obsažené v mobilním zařízení. Některé aplikace jsou již distribuovány s OS jako jsou např.: SMS/MMS klient, správce kontaktů, webový prohlížeč, kalkulačka atd. Aplikace jsou psány v Java kódu (.class) a ten je následně zkompilován do dex (Dalvik Executable) bytekódu, který používá Dalvik. Pro instalaci aplikací do systému Android se využívají balíčky APK (Android application package), které obsahují zabalené soubory nutné pro běh aplikace (obrázky, hudbu, knihovny atd.). Pomocí souborů APK lze jednoduše aplikaci nainstalovat [40].

### 1.6.7 Správce aktivit (Activity Manager)

Revoluční novinkou ve světě operačních systémů a tvorbou aplikací bylo přidání tzv. aktivit, kde si aktivitu můžeme představovat jako jedno vykreslené okno aplikace.

[16] nativní = vlastní pro danou platformu, specifický pro daný systém a procesor.

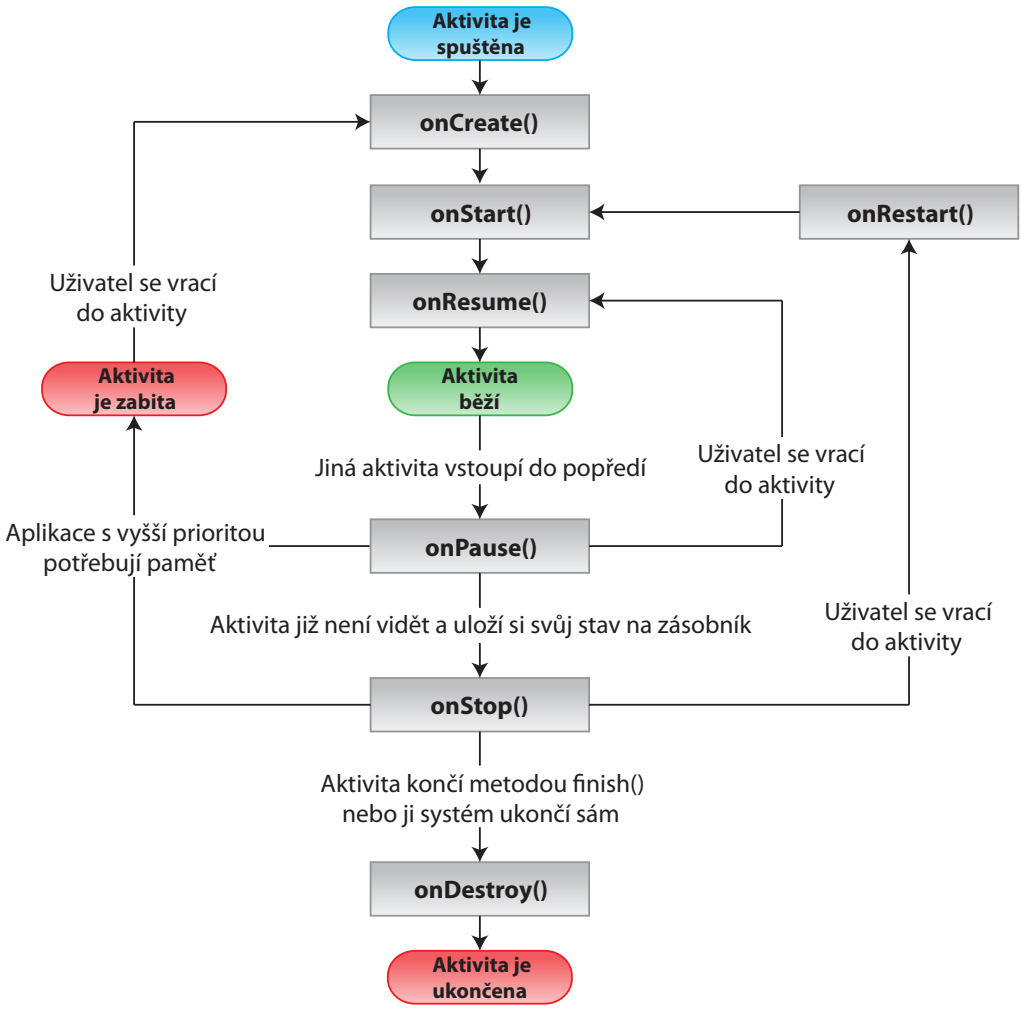
Aplikace se obvykle skládá z několika aktivit, které jsou k sobě navzájem volně vázány. Ve většině případů je jedna aktivita aplikace specifikována jako „hlavní“, která je uvedena pro uživatele při prvním spuštění. Každá další aktivita pak může zahájit svou vlastní činnost, aby se prováděly různé akce. Po každé, když nová aktivita začíná, předchozí aktivita se musí zastavit (uspat), přičemž je tato uspaná aktivita uložena na zásobníku aktivit (tzv. „back stack“) pro její pozdější použití. Tím jak jsou ukládány aktivity na zásobník se lze vracet v aplikaci směrem k „hlavní“ aktivitě tlačítkem ZPĚT. Při návratu se ke starší aktivitě je aktuální aktivita zničena a odebrána ze zásobníku. Při zmáčknutí tlačítka DOMŮ, které vás vrátí do domovské obrazovky OS, se poslední aktivita uloží na zásobník a uspí se. Při opětovném spuštění aplikace se vyhledá poslední známa aktivita této aplikace a použije se. Pokud je nedostatek paměti, tak pomocí Low Memory Killer se odstraňují nejstarší aktivity aplikace [4].

### Životní cyklus aktivit

Zde je popsán životní cyklus aktivit od jejich vzniku do jejich zániku a průchody mezi stavy z obrázku 1.19. Každá aktivita musí mít zadaný konkrétní stav, podle kterého s ní OS zachází. Aktivity však nejsou určeny k dlouhodobým operacím a výpočtům, protože při odložení aktivity do pozadí se aktivita uspí a probudí se jen při opětovném požadavku. Pokud by bylo potřeba vytvořit aplikaci, která na pozadí vykonává nějakou činnost (např. pravidelné stahování pošty či přehrávání hudby), se musí použít samostatná vlákna nebo služby (služby jsou popsány v následující kapitole 1.6.8) [4].
Následuje výčet stavů a přechodů životního cyklu aktivit z obrázku 1.19 [4]:

- **`onCreate()`** – Volá se jen jednou při spouštění aktivity. Zde je nutné inicializovat podstatné součásti aktivity (co bude používat, s čím bude pracovat atd.). Další důležitá věc je definovat rozložení okna pro systémový zobrazovač. Po tomto stavu vždy následuje stav `onStart()`.
- **`onStart()`** – Používá se pro zviditelnění aktivity.
- **`onResume()`** – Tato metoda se volá, když je potřeba zviditelnit aktivitu, která je ve stavu kde se jí odebírají přidělené systémové prostředky (`Pause`).
- **`onPause()`** – OS vyvolá tuto metodu, když uživatel opouští tuto aktivitu a přechází k nadřazené. Zde se musí ukončit akce aktivity, které by využívaly zbytečně procesor (např. animace). Po ukončení akcí se aktivita přepne do stavu `onStop()`.
- **`onRestart()`** – Volá se tehdy, pokud se má aktivita znovu zobrazit. To nastává pokud se uživatel vrátí k aplikaci, která byla uspána odložena na zásobník.

- `onStop()` – Je volána, když už aktivita není zapotřebí a uvolnila všechny přidělené systémové prostředky. Uloží si svůj stav na zásobník a čeká v paměti. Při nedostatku paměti může být odstraněna. Pokud si tuto aktivitu uživatel vyžádá, obnoví se přes metodu `onRestart()`.
- `onDestroy()` – Volá se v případě, že je potřeba aktivitu zničit. To může nastat voláním metody `finish()`, kde se oznamuje OS, že aktivita dodělala svou činnost nebo čištěním paměti pomocí Low Memory Killer. Také lze aktivitu ukončit tzv. „na tvrdo“, kde se OS řekne, že ji má ukončit.




Obr. 1.19: Životní cyklus aktivit [4].

Aktivita je brána jako jedno okno na obrazovce, díky tomu na větších obrazovkách (např. tablet) nebyla možnost mít více oken vedle sebe. Proto vznikl fragment. Ten si můžeme představit jako okno, které má svůj vlastní životní cyklus podobný aktivitě a spadá plně pod režii aktivit. Tím je u větších rozměrů obrazovky dosaženo toho, že může obsahovat více oken (teď jsou to fragmenty) a ty spadají pod jednu aktivitu, která je řídí. Tento přínos byl zaveden až ve verzi Android 3.0 [4].

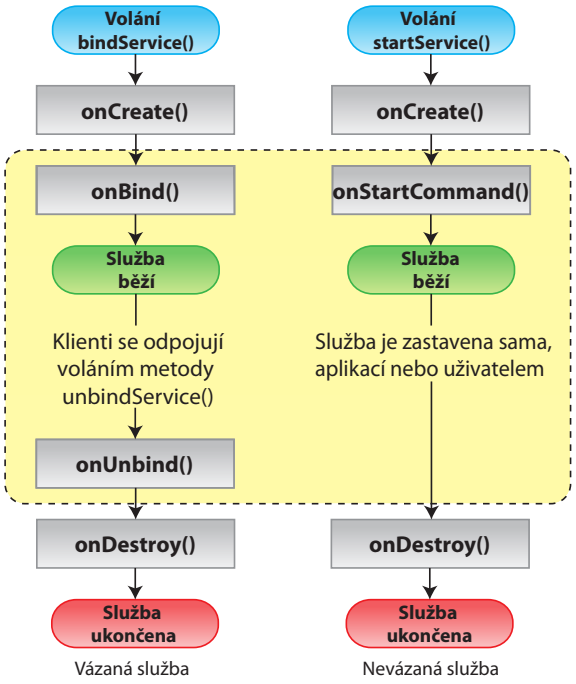
### 1.6.8 Služby (Services)

Služby jsou součásti aplikace, které mohou provádět dlouhotrvající operace na pozadí (stahování souboru či přehrávání muziky) a neposkytují grafické uživatelské rozhraní. V aplikaci stačí spustit službu, která bude běžet nadále na pozadí bez toho, aniž by byla odložena nebo uspána na zásobníku aktivit. Pravděpodobnost toho, že bude aplikace se službou na pozadí odstraněna z paměti je menší než u aplikace, která má na pozadí pouze aktivity. Se službou se komunikuje zpravidla pomocí Binder IPC (viz. kapitola 1.6.1). Služba standardně není samostatný proces ani samostatné vlákno, protože je součástí procesu, který ji vytvořil. Lze ji však spustit pod samostatným procesem, který oddělí hlavní proces (uživatelské rozhraní) od procesu služby a tím lze předcházet zamrznutí aplikace a služby najednou [4].
Jsou definovány dva základní druhy služeb [4]:

1. **Nevázaná** (Unbounded service) – je volána metodou `onStartCommand()`. Tím se spustí a běží na pozadí na dobu neurčitou, i když je odstraněna součást, která ji spustila. Obvykle provádí jen jednu operaci a nevrací výsledek tomu kdo ji spustil. Tato služba může komunikovat pouze s procesem, který ji spustil. Tento typ se dá využít například pro stáhnutí nebo nahrání souboru přes síť. Po dokončení operace se služba ukončí.
2. **Vázaná** (Bounded service) – tento druh služby se spouští metodou `onBind()`. Také běží na pozadí, ale s tím rozdílem, že vytvoří rozhraní typu klient-server, které umožňuje meziprocesní komunikaci se službou (získat výsledky, zaslat žádost atd.). Komunikace mezi procesy probíhá pomocí Binder IPC. Vázaná služba běží pouze tehdy, pokud je na ní alespoň jeden proces vázán. Když již nemá služba žádnou vazbu, ukončí se. Tento druh služby lze využít pro přehrávání zvukových souborů na pozadí podporující komunikaci s widgetem a aplikací.

Tyto dva základní druhy lze kombinovat. Služba může být zavolána metodou `onStartCommand()`, aby běžela neomezeně a poté voláním metody `onBind()` lze docílit mezi procesní komunikaci se službou.
Dále je uveden popis stavů a přechodů životního cyklu služeb z obrázku 1.20:

- **`onCreate()`** – Používá se při vytváření služby. Dále následuje specifikace služby a to vázaná (bind) či nevázaná (unbind).
- **`onStartCommand()`** – Po provedení této metody je služba spuštěna v nevázané specifikaci (unbind) a následně se překlopí do běžícího stavu. Při vytváření se musí jasně definovat kdy služba ukončí svou činnost (ukončovací podmínka). Po provedení své operace služba zavolá metodu `stopSelf()` nebo hlavní proces použije metodu `stopService()`. Další možnost ukončení služby je předčasné ukončení uživatelem.




Obr. 1.20: Životní cyklus služeb [4].

- `onBind()` – Metoda pro vytvoření vázané (bind) služby s mezi procesní komunikací typu klient-server pomocí Binder IPC.
- `onUnbind()` – Metoda, která hlídá počet vázaných procesů na službu. Pokud není žádná vazba tato metoda vyvolá metodu `onDestroy()`.
- `onDestroy()` – Operační systém vyvolává tuto metodu, pokud není služba již delší dobu používána nebo pokud je připravena ukončit svou činnost. Při této metodě si po sobě služba musí uklidit (odebrat prostředky, zrušit vlákno, vyčistit paměť). Nadále služba přechází do závěrečného stavu `Služba ukončena`.

### 1.6.9 Intenty a intentové filtry (Intents and Intent Filters)

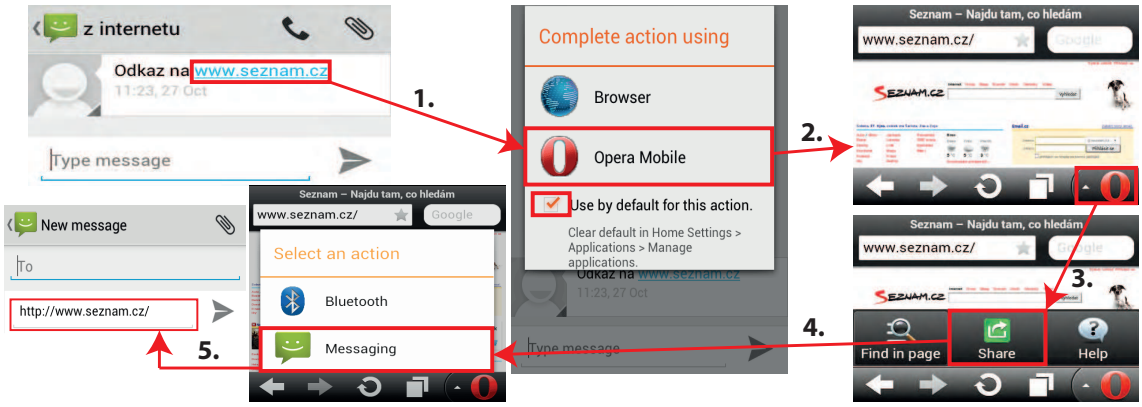
Intenty jsou zprávy, které vyvolává uživatel a jsou odesílány mezi hlavními stavebními bloky, jako jsou aktivity (Activity, kap. 1.6.7), služby (Services, kap. 1.6.8) a přijímači vysílání (Broadcast receivers 1.6.10). Podněcují aktivity k zahájení jejich činností, říkají službám kdy se mají spustit nebo zastavit a odesílají přijímačům zprávy. Intenty používají asynchronní komunikaci, což znamená, že odesílatel nemusí čekat na dokončení akce. Intenty mohou být explicitní nebo implicitní. V explicitním intentu odesílatel jasně uvádí příjemce. V implicitní intentu odesílatel určuje typ příjemce [40], [4].

Každý příjemce intentu může mít přidělený filtr, který specifikuje jaké intenty může přijmout a jaké akce jsou přes ně dovoleny. Lze filtrovat pomocí odesílatele,

příjemce, specifické akce, kategorie (např. prohlížeče webu), podle typu dat (video, audio). Navíc se tyto filtry mohou na sebe libovolně řetězit. Jako příklad poslouží editor poznámek z OS. Tato aplikace má dva základní filtry, první na spuštění s konkrétní poznámkou a druhý pro spuštění s novou (prázdnou) poznámkou. Na jiné intenty nebude reagovat [4].

Popis fungování intentů z obrázku 1.21 je následující:

1. Uživateli přišla SMSka, která obsahuje odkaz na stránku www.seznam.cz. Jak na ní klikne, vyvolá tím implicitní intent s filtrem příjemců, kteří podporují webový obsah.
2. Pokud má uživatel nainstalovaných více webových prohlížečů (na tomto obrázku jsou dva), OS umožní výběr prohlížeče k dokončení akce. Zde byla vybrána aplikace Opera Mobile a navíc byla zaškrtnuta volba pro standardní používání[17] aplikace na tyto intenty. Poslaný intent vyvolal akci na spuštění procesu a zahájení aktivity aplikace Opera Mobile.
3. Dále bylo v aplikaci vybráno sdílení odkazu na webovou stránku.
4. OS opět vyvolal výběr, kde nabízí všechny aplikace podporující tuto akci s různě nastavenými filtry intentů. Sdílení odkazu bylo vybráno pomocí SMSky, kde se tentokrát vyvolal explicitní intent, protože byl vybrán konkrétní příjemce.
5. Intent spustí novou aktivitu, která vytvořila novou zprávu a zkopírovala do obsahu sdílený odkaz.




Obr. 1.21: Ukázka používání intentů.

Podobně toto funguje i v OS Microsoft Windows, kde může mít uživatel více programů na stejnou akci (např. Internet Explorer, Firefox, Safari, Opera atd.) a OS mu pak dá vybrat v jaké aplikaci to chce spustit nebo při nastavení jako výchozí přejde přímo k vykonávání [40].

[17] Při dalším spuštění se OS nebude ptát na výběr prohlížeče, ale přímo spustí již dříve vybranou aplikaci.

### 1.6.10 Přijímače vysílání (Broadcast receivers)

Přijímači vysílání jsou odvozeny od intentů. Hlavní rozdíl je v tom, že u tohoto typu komunikace nevyvolává akci uživatel, ale OS. Každá aplikace, která potřebuje být přijímač, se musí v OS zaregistrovat, aby tyto zprávy mohla obdržet. To může udělat staticky (při startu aplikace) nebo dynamicky (při běhu aplikace). Při požadavku na odeslání zprávy na specifickou skupinu přijímačů, se OS podívá na zaregistrované příjemce a pošle jim zprávu typu intent, která již může zahájit aktivitu nebo službu. Přijímače vysílání také mohou mít zadané filtry jen na specifické události. Tato komunikace připomíná některé typy komunikací po multicast sítích, kde se také jednotlivé stanice registrují do přijímací skupiny [4], [41].

Android díky přijímačům vysílání oznamuje systémové události, jako jsou změny v připojení k internetu, slabá baterie, příjem SMS/MMS zprávy, zapnutí nebo vypnutí displeje, příchozí volání atd. Díky tomu umožňuje uživateli vyměnit nativní aplikace v OS ( například SMS/MMS aplikaci nebo jinou aplikaci na příjem hovorů). Zpravidla se přijímače vysílání používají pouze k vytvoření systémové notifikace [40].

## 1.7 Bezpečnost Androidu (Android Security)

V poslední době se tato otázka okolo mobilních OS řeší čím dál častěji. Proto, aby mohl Android proniknout do podniků musí obsahovat důkladné zabezpečení a výbornou kontrolu nad aplikačním prostředím. Tímto se odlišují poslední dvě verze OS systému, které jsou takřka identické až na pár nových aplikací, ale verze Jelly Bean přináší mnoho novinek v oblasti bezpečnosti.

Android je moderní mobilní platforma, která byla navržena tak, aby byla otevřená pro širokou veřejnost. Zajištění otevřenosti platformy vyžaduje robustní bezpečnostní architekturu a propracované bezpečnostní programy. Proto Android přichází s více vrstvím zabezpečením, které poskytuje flexibilitu potřebnou pro otevřenou platformu a zároveň poskytuje ochranu pro všechny uživatele [5], [42].
Zabezpečení OS Android zajišťuje aby [5]:

- chránil uživatelská data,
- chránil systémové prostředky (včetně sítě),
- zajistil izolaci aplikací.

K dosažení takovéto ochrany Android poskytuje tyto klíčové bezpečnostní prvky [4]:

- robustní zabezpečení prostřednictvím linuxového jádra,
- povinné použití aplikační karantény pro všechny aplikace,
- bezpečná meziprocesní komunikace pomocí Binder IPC,
- podpisy aplikací u certifikační autority Android,
- přidělování oprávnění pro komunikaci s chráněnou API.

Následující oddíly této kapitoly popisují vrstvy zabezpečení od samotného linuxového jádra po zabezpečení aplikací.

### 1.7.1 Zabezpečení na úrovni systému a jádra (System and Kernel Level Security)

#### Bezpečnost Linuxu (Linux Security)

Základ platformy Android je již několikrát zmíněné Linuxové jádro. Toto jádro je již vyvíjeno od roku 1991 a je distribuováno v open source licenci, kde si každý může stáhnout zdrojové kódy a upravovat si je podle libosti.

Na úrovni OS, platforma Android poskytuje bezpečnost linuxového jádra, stejně jako bezpečnost pro komunikaci mezi procesy (IPC). Tyto funkce zajistí, aby i nativní kód byl omezen na aplikační karanténu (Application Sandbox). Díky tomu, že má Android oddělené procesy a komunikace probíhá výhradně přes jádro, které kontroluje tuto komunikaci, je model zabezpečení hned větší [5], [42].
Klíčové bezpečností funkce z Linuxu jsou [5]:

- přístup do systému je omezen pouze na uživatelský přístup (user-space),
- úplná izolace procesů,
- rozšířitelný mechanizmus pro mezi procesní komunikaci (IPC),
- schopnost odstraňovat potenciálně nebezpečné a zbytečné části jádra.

#### Aplikační karanténa (The Application Sandbox)

Android využívá uživatelský přístup z OS Linux, kde se přiřadí jednoznačné identifikační číslo tzv. UID (Unique ID) pro každou aplikaci, která běží v samostatném procesu. Tento přístup se liší od ostatních OS tím, že v ostatních OS se spouští více aplikací se stejnými uživatelskými právy. Ve výchozím nastavení nemohou aplikace mezi sebou přímo komunikovat a mají omezený přístup k OS. Pokud by se aplikace pokusila například číst data druhé aplikace nebo vytočit telefon bez povolení, pak OS zaregistruje tuto akci a zamítne ji, protože aplikace musí mít odpovídající uživatelské oprávnění k provedení určité akce [5].

Všechen software nad linuxovým jádrem, včetně knihoven OS, aplikačního frameworku a samotných aplikací, spadá do této aplikační karantény a navíc je její používání monitorováno [5].

### Systémový oddíl a nouzový režim (System Partition and Safe Mode)

Systémový oddíl obsahuje Linuxové jádro, knihovny, aplikační framework a další součásti pro OS. Tento oddíl je určen pouze pro čtení běžnému uživateli[18] a superuživatel (root) může zde i zapisovat [5].

Android také podporuje tzv. nouzový režim (safe mode), kde se spustí jen operační systém bez nainstalovaných aplikací třetích stran. Tím lze jednoduše odstranit aplikaci ze systému, pokud špatně funguje na normálním režimu OS [5].

### Oprávnění souborového systému (File system Permissions)

Standardně se aplikace instalují do speciálního adresáře podle jména balíčku aplikace (např. /data/data/com.adobe.reader). Tyto složky jsou nastaveny tak, že aplikace s přidruženým UID jsou majitelé pouze svých souborů a složek a jiná aplikace s jiným UID nebude mít přístup do jejich složky. Tím je zaručeno oddělení aplikací na datovém úložišti. Pokud by aplikace chtěla sdílet data ostatním aplikacím, musí se k souborům a složkám udělit dodatečné oprávnění [42].

Veškeré data uložené na externí kartě nedisponují žádnými oprávněními souborového systému, protože jsou naformátována pomocí souborového systému, který nepodporuje standardní Linuxové oprávnění. Jsou tak přístupná všem bez ohledu na vlastníka. Pro ochranu těchto dat lze využít šifrování [42].

### Šifrování souborového systému (Filesystem Encryption)

Šifrování souborového systému podporují verze Android 3.0 a novější. Samotné šifrování je založeno na použití symetrických šifer (pro šifrování i dešifrování využívá stejný klíč s pevně danou délkou šifrovacího klíče). Používá se algoritmus AES (Advanced Encryption Standard) se 128 bitovou délkou klíče společně s hašovací funkcí SHA1 (Secure Hash Algorithm) s 256 bitovým otiskem. Šifrovací klíč je chráněn pomocí AES se 128 bitovým klíčem odvozený od uživatelského hesla. Výhoda šifrování tkví v tom, že pokud by byl mobilní telefon odcizen, tak se nálezce nedostane k datům bez zadání hesla. Nevýhodou je jen zapomenutí hesla, protože tam již neexistuje žádná varianta jak se dostat bez toho hesla ke svým datům [5].

Pro zajištění odolnosti proti útokům (např. brute force[19] či rainbow tables[20] ) se heslo kombinuje s kryptografickou solí[21], které se následně hašuje [5], [42].

[18] Některé soubory a složky v systémovém oddílu nejsou pro běžného uživatele přístupné.

[19] Zkouší se systematicky celý možný prostor hesel.

[20] Tabulky s předem vypočtenými hodnotami pro prolomení hašovacích funkcí, nejčastěji pro zjištění hašovaných hesel. Zdroj: http://en.wikipedia.org/wiki/Rainbow_table

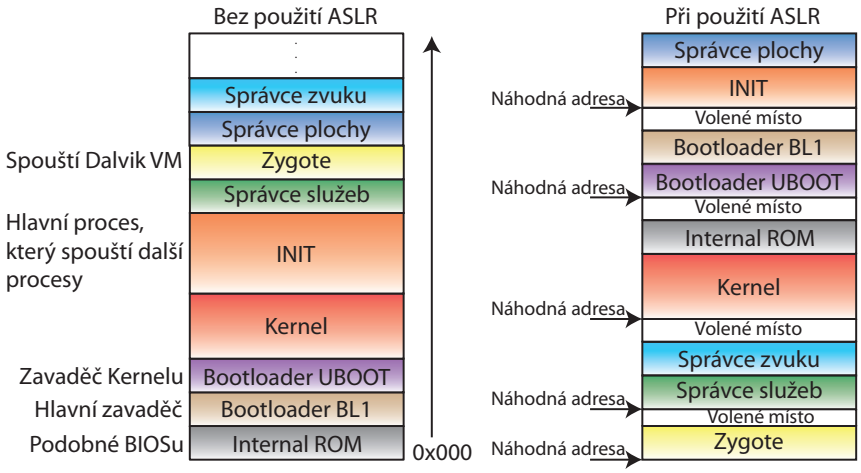
[21] Několik náhodných bitů pro jednosměrnou funkci, které umožňují, aby její výstup měl mnoho možných variant. Zdroj: http://cs.wikipedia.org/wiki/Kryptografická_sůl

Pro zajištění odolnosti proti slovníkovému útoku[22] Android stanoví složitost hesla jako je minimální délka, použití malých a velkých písmen, čísel a speciálních znaků.

### Správa zabezpečení paměti (Memory Management Security)

Jelikož se v OS Android používá sdílená paměť, je nutno zajistit ochranu dat aplikací před zneužitím. Proto každá verze OS Android přináší mnoho vylepšení v této oblasti. Nejzajímavější vylepšení přinesly verze 4.0 a 4.1.

Android 4.0 přinesl novou technologii ASLR (Address Space Layout Randomization – náhodné rozložení adresního prostoru). Jedná se o metodu zabezpečení, která zahrnuje pseudonáhodné uspořádání pozic klíčových prvků v operační paměti, jako je jádro, knihovny, moduly aplikačního frameworku a další potřebné procesy pro chod OS. Také podporuje náhodné umístění i pro všechny ostatní procesy. Tato metoda funguje tak, že při každém spuštění OS budou tyto procesy na různých místech než předtím. Tím výrazně ztíží (nebo dokonce zabrání) útok pomocí známých bezpečnostních děr systému, protože útočník nebude vědět, kde se data v paměti přesně nachází. Tuto metodu zabezpečení již využívá řada OS jako je Microsoft Windows či Linux. Bez této metody se při zavádění OS nahrávala data postupně za sebou od začátku operační paměti [4], [5].
Porovnání této metody zabezpečení a klasické uspořádání dat v operační paměti je vidět na obrázku 1.22.

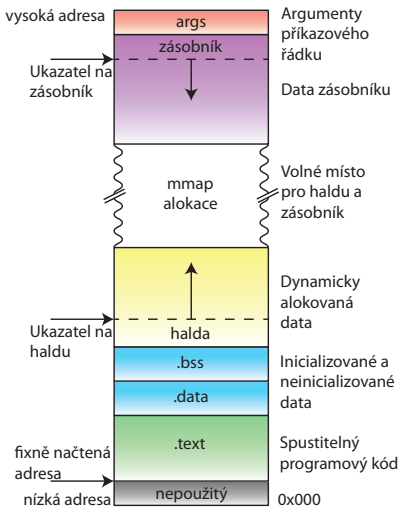



Obr. 1.22: Porovnání použití ASLR se starším způsobem [5].

Verze 4.1 podporuje technologii PIE (Position Independent Executable – pozičně nezávislý spustitelný soubor). Strojový kód, který je možno vykonat nezávisle na umístění v operační paměti. Tuto metodu musí podporovat procesor, musí být zapnutá podpora ASLR a program musí být přeložen speciálním kompilátorem. Na

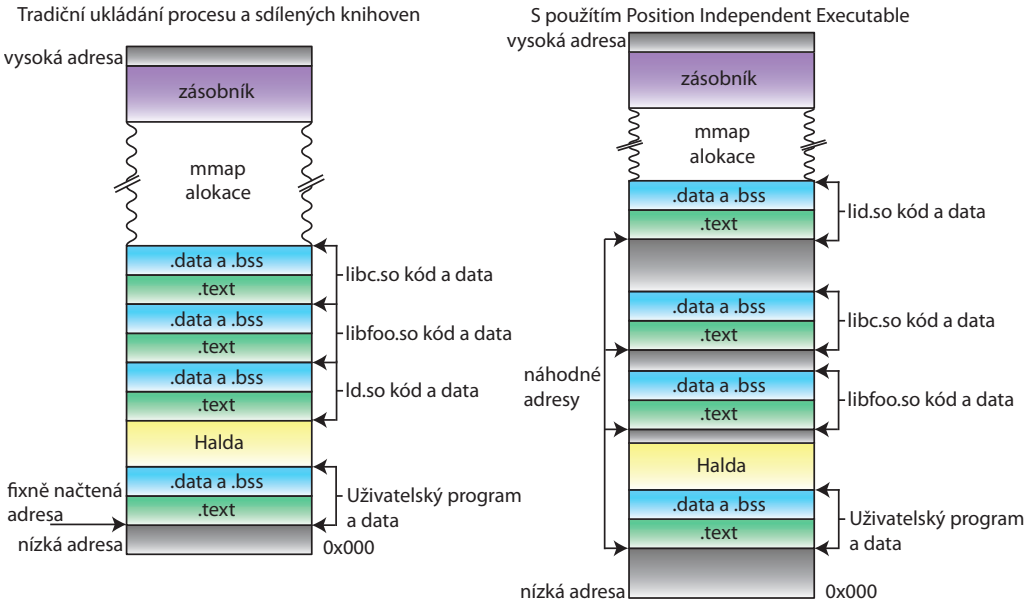
[22] Útok založený na zkoušení hesel ze slovníku.

obrázku 1.23 je vidět standardní rozložení programu v operační paměti bez použití sdílených knihoven. Jednotlivé části procesu se skládají za sebe [17].




Obr. 1.23: Tradiční rozložení procesu v operační paměti – statický program [17].

Obrázek 1.24 znázorňuje případ se sdílenými knihovnami (libc.so, libfoo.so a ld.so) bez a s použitím techniky PIE. Na rozdíl od klasického chování procesů v operační paměti se jednotlivé části procesů ukládají na náhodné místo, které určí OS.




Obr. 1.24: Rozložení procesu se sdílenými knihovnami bez a s použitím PIE [17].

### 1.7.2 Zabezpečení aplikací Androidu (Android Application Security)

Každá aplikace (soubor .apk) musí obsahovat základní prvky jako jsou Android Manifest a hlavní aktivita. Přídavné prvky jsou další aktivity, služby a přijímače vysílání, které nemusí být přítomny. Zabezpečení mezi procesní komunikace řeší Binder IPC (kapitola 1.6.1) pomocí ověřování přidělených práv [5].
Nejdůležitější je již zmiňovaný soubor Android Manifest, který obsahuje informace o aplikaci a provádí následující úkony [4]:

- Pojmenovává balíček Java aplikace. Název balíčku slouží jako jedinečný identifikátor aplikace (např. com.adobe.reader).
- Popisuje komponenty aplikace (aktivity, služby, přijímače vysílání a poskytovatelé obsahu) a přiděluje potřebné práva pro komunikaci, filtry atd.
- Určuje která oprávnění aplikace dostane pro přístup k chráněným částem API rozhraní a pro komunikaci s ostatními aplikacemi.
- Definuje minimální úroveň použitelného API rozhraní.
- Uvádí jaké knihovny jsou propojené k aplikaci.

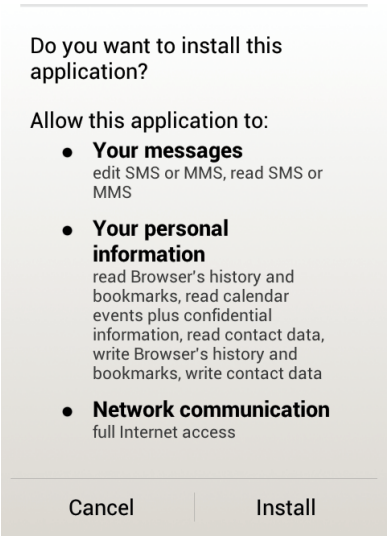
#### Přístup k chráněné části API (Accessing Protected APIs)

Ve výchozím nastavení má aplikace zakázán přístup k většině zdrojům. Pokud nejsou v souboru Android Manifest přiděleny žádné práva, tak je aplikace schopna komunikovat pouze sama se sebou. Některé přístupové práva nelze přidělit z důvodu bezpečnosti (např. nedostane přístup do chráněné oblasti na SIM kartě). Pokud aplikace potřebuje využívat něco z chráněné API, musí mít definovány přístupové práva. Chráněná API jsou přístupná pouze přes OS [5].
Do chráněné oblasti API patří [5]:

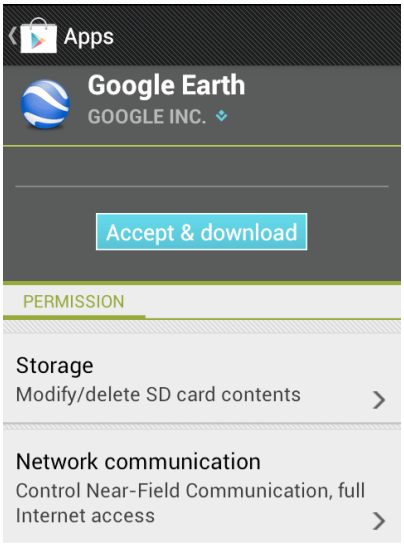
- funkce fotoaparátu,
- funkce Bluetooth,
- funkce telefonu,
- funkce SMS/MMS,
- lokalizační údaje (GPS),
- funkce sítí (WiFi, GPS, UMTS atd.).

Před samotnou instalací aplikace OS zobrazí dialogové okno pro uživatele, které vypisuje požadovaná oprávnění aplikace (viz. obr. 1.25). Pokud uživatel této aplikaci povolí vyžadovaná práva, OS si uloží informace o přidělení práv a dovolí aplikaci instalaci. Uživatel bohužel nemůže udělit nebo odepřít aplikaci individuální práva, ale musí souhlasit nebo zamítnout všechna požadovaná práva. Práva jsou aplikaci přidělena tak dlouho jak dlouho je nainstalovaná. Při odstranění aplikace se práva odebírají a při opětovné instalaci se opět uživatele OS ptá na přidělení práv [4], [5].

V případě, že se aplikace pokusí použít chráněnou funkci, která není uvedena v manifestu, bude přístup odepřen. Vyžadování přístupových práv po uživateli jako je na obrázku 1.25 je typické pro instalaci aplikace přímo v mobilním telefonu[23]. Většinou však uživatel instaluje z Google Play, kde se souhlasí s přidělováním práv a po potvrzení se aplikace začne stahovat a instalovat (obr. 1.26) [4], [5].




Obr. 1.25: Povolení přidělení práv aplikaci v mobilním telefonu.




Obr. 1.26: Povolení přidělení práv aplikaci z Google Play.

### Podpis aplikací (Application Signing)

Všechny Android aplikace (.apk) musí být podepsané certifikátem, jehož soukromý klíč je ve vlastnictví vývojáře. Pokud aplikace nedisponuje podpisem autora nelze tuto aplikaci nikde nainstalovat. Tento certifikát prokazuje identifikaci autora aplikace a stanovení vztahů důvěryhodnosti mezi aplikacemi. Podepsáním se umožní jednoduchá aktualizace aplikace, bez vytváření složitých rozhraní a oprávnění [4].

Aplikace, která je z Google Play (Android Market) musí mít tento certifikát podepsaný u certifikační autority Android. Certifikát nemusí být vždy podepsán certifikační autoritou, ale stačí tzv. self-signed, kde si certifikát podepíše sám autor. Tyto aplikace nelze distribuovat pomocí Google Play a pro instalaci do zařízení je nutno povolit instalaci aplikací z neznámého zdroje [4].

## 1.7.3 Bezpečnost vs. otevřenost systému

Jak již zde bylo mnohokrát řečeno, OS Android má otevřenou architekturu, která umožňuje komukoliv čtení a úpravu celého OS. S touto možností vzniká mnoho

[23] Otevírá se soubor .apk – většinou se jedná o aplikace ze třetích stran nebo stažené aplikace.

modifikací operačního systému, tzv. custom ROM[24], a Linuxových jader. Oficiální OS se nazývá Stock ROM. [18].

### Výhody custom ROM

Custom romky se vyvíjí hlavně za účelem zrychlení mobilního telefonu. Odstraňují se zbytečné aplikace, které jsou implementovány přímo do oficiálních verzí OS. Přidávají se nové aplikace pro lepší ovládání zařízení. Často se také upravuje vzhled OS pro lepší plynulost a chod. Vývojáři se snaží opravit chyby z oficiálních verzí a vylepšují některé funkce OS. Mezi další výhody se řadí frekvence aktualizací. Pokud se zjistí chyba OS tak je většinou do druhého dne opravena a distribuována v nové verzi custom ROM. Další vylepšení je instalace aplikací na SD kartu a přímý přístup do souborů OS pomocí uživatele root. Největší výhoda ovšem je ta, že i na některé starší, již oficiálně nepodporované zařízení, lze nahrát nejnovější OS pokud takovou romku někdo vytvořil [18].

### Nevýhody custom ROM

Tím, že vzniká mnoho modifikací, kde je ve většině případů OS pozměněn tak, aby měly aplikace práva root, klesá rapidně bezpečnost celého OS a představuje tak bezpečnostní hrozbu. Při modifikaci mohou vývojáři zavést vážnou chybu systému nebo ohrozit nějakou podstatnou část ochrany. Není žádná oficiální podpora od Android. Při instalaci custom ROM se smažou všechna data z telefonu. Asi jako hlavní nevýhoda je ta, že při flashování (nahrávání OS) custom rom se ztrácí záruka u daného zařízení. Také mohou při flashování nastat nějaké potíže a může se zařízení nenávratně poškodit [18].

### Vývojáři custom ROM a Linuxových jader

Většina vývoje custom ROM a Linuxových jader je na fóru XDA Developers, kde je komunita s více než čtyřmi milióny uživatelů z celého světa. Hlavním cílem je diskuse, řešení problémů a rozvoj Android. Mezi neznámější custom ROM patří skupiny CM (CyanogenMod), AOKP (Android Open Kang Project), Darky ROM a MIUI ROM, která má předělané grafické rozhraní připomínající iPhone. MIUI ROM je rozšířena na tolik, že už vydává svůj vlastní telefon.

Linuxové jádra mají také svoji modifikaci a nejznámější jsou od Semaphore, Devil kernel, SpeedMod a Midnight [18].

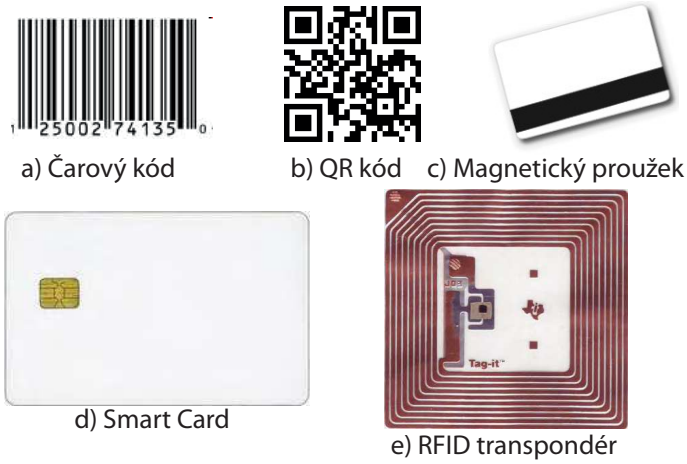
[24]Speciální část paměti ve které je uložen vlastní operační systém. Proto se slovem ROM také označují vlastní soubory s OS.

# 2 RADIO FREQUENCY IDENTIFICATION

Radio Frequency Identification (RFID) nebo-li česky identifikace na rádiové frekvenci. Bez znalostí této technologie by bylo těžké pochopit technologii NFC (Near field communication – kap. 3), protože NFC navazuje na standardy specifikované pro RFID. Tato kapitola se tedy zabývá výhradně popisem technologie RFID. Jelikož je tato diplomová práce zaměřena na technologii NFC, jsou zde popsány jen základy funkce pro systém RFID bez popisu historie a bezpečnosti.

RFID je bezdrátový bezkontaktní systém, který používá elektromagnetické pole pro přenos dat pro účely automatické identifikace a sledování zboží. Nahrazuje systém magnetických proužků, smart card a čárových kódů, kde musela být čtečka v přímé viditelnosti a těsné blízkosti pro čtení zprávy (obr.2.1). RFID oproti čárovým kódům je rychlejší, spolehlivější a poskytuje jednodušší identifikaci a sledování objektů. Nepotřebuje přímou viditelnost a může obsloužit pomocí čtečky až stovky požadavků za sekundu. Většinou se používají v kombinaci s čárovým kódem [19].

Informace jsou přenášeny pomocí transpondérů, též známé jako RFID tagy či RFID štítky (viz. obr. 2.2). Slovo transpondér vzniklo ze slov transmit (přenos) a responce (odpoveď). Základní funkcí je uložení informací do vnitřní paměti a při požadavku je distribuovat ke čtecímu zařízení [20].

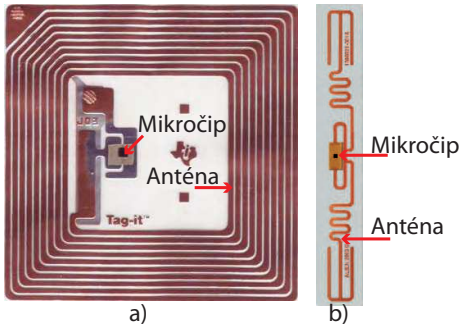



Obr. 2.1: Srovnání technologií [20].

V logistice se staly čárové kódy osvědčeným standardem a fungují již dlouhou řadu let. Nicméně se stále více nahrazují technologií RFID zejména kvůli velkému úložnému prostoru informací a schopnosti přenosu dat mezi čtečkou a transpondérem ve vzdálenosti několika centimetrů až desítek metrů. Další zajímavý přínos je sledování objektu v reálném čase pomocí aktivních transpondérů kteří posílají informaci o poloze. Jediné proč se stále používají čárové kódy je jejich výrobní cena, kde RFID transpondér stojí mnohem více než kus potištěného papíru [21].

## 2.1 RFID transpondér a čtecí zařízení

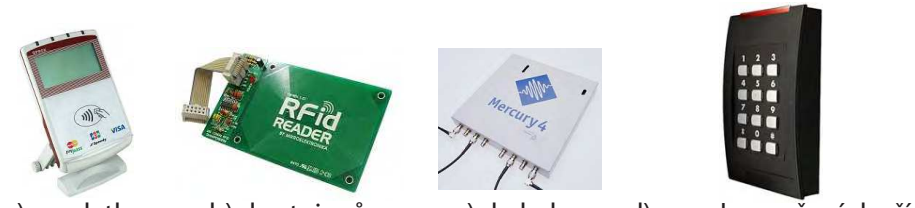
Transpondér obsahuje mikročip a komunikační anténu (obr. 2.2). Samotný čip může být pouze 1 mm velký. Velikost transpondérů záleží na použité frekvenci a s tou souvisí použitá anténa. Obvykle platí, že čím je vetší frekvence tím je menší anténa. Čip obsahuje různé typy a velikosti pamětí pro ukládání informací. V této paměti může být uloženo například místo výroby, datum výroby a spotřeby, způsob přepravy a další informace. RFID transpondéry se hojně v dnešní době používají pro identifikaci osob v docházkovém systému, pro přístup k bankovnímu účtu, na elektronickou jízdenku a na umožnění oprávněné osobě vstoupit do budovy [21].




Obr. 2.2: Transpondér pro: a) nízké a vysoké frekvence, b) velmi vysoké a mikrovlnné frekvence [20].

V dnešní době se vyrábí mnoho velikostí a typů RFID transpondérů pro různé účely. V následující kapitole je popsáno základní dělení podle typu: pamětí, použití, komunikace a používané frekvence.

Čtecí zařízení má také již mnoho podob (viz. obr. 2.3). Od platebního systému po stojany u pokladen obchodů či čtečky do velkých hal při použité vysoké frekvenci. Čtecí zařízení dodává energii pasivním transpondérům, zachytává a demoduluje informace od transpondérů. Zapisuje data do programovatelných pamětí transpondérů. Také musí řešit kolize při čtení či zápisu několika transpondérů najednou a ověřuje je, aby se zabránil neoprávněný přístup do systému RFID. Podporuje i šifrování a integritu přenášených dat [19].




Obr. 2.3: Druhy čteček RFID [20].

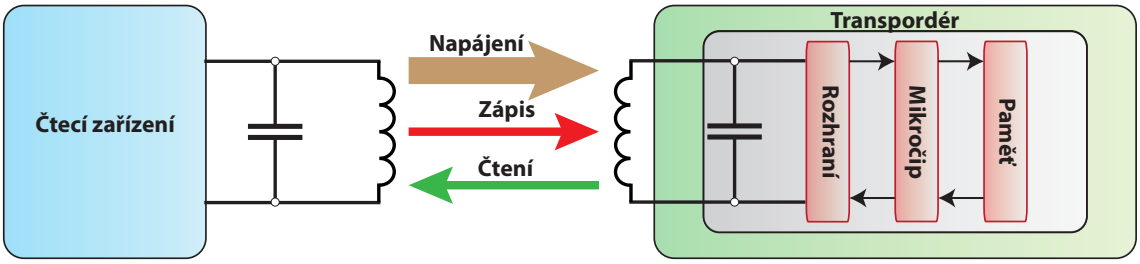
## 2.2 Princip komunikace RFID

Pro jednoduchost tu bude popsána komunikace RFID čtečky s pasivním RFID transpondérem (nemá vlastní napájení, více viz. kap. 2.3.3), který je více používaný.

Čtečka vysílá prostřednictvím své antény do okolí periodicky na svém nosném kmitočtu elektromagnetickou vlnu. Pokud se objeví transpondér v dosahu této vlny, který je naladěn na stejnou nosnou frekvenci, je jím tato vlna přijata. Na anténě transpondéru se indukuje napětí a tím se vyvolá střídavý elektrický proud. Střídavý proud se usměrní a prochází kapacitorem (kondenzátorem), který se postupně nabíjí. Tato akumulovaná energie je použita pro napájení elektronických obvodů v transpondéru. Jakmile akumulovaná energie na kondenzátoru dosáhne minimální potřebnou úroveň, spustí se elektronické obvody na transpondéru a pomocí antény začne odpovídat čtecímu zařízení [21], [22].

Transpondér zpravidla používá dvoustavové amplitudové klíčování (ASK – Amplitude Shifting Key), které je realizováno změnou zakončovací impedance antény transpondéru. Pomocí vysílaných vln ze čtecího zařízení lze do transpondéru i zapisovat data (pokud transpondér disponuje programovatelnou pamětí) [19].

Ve čtecím zařízení se vyslaný signál od transpondéru demoduluje a začne se zpracovávat. Rychlost a velikost akumulované energie v kapacitoru transpondéru závisí na vzdálenosti čtecího zařízení. Při větších vzdálenostech a překážkách je nabíjení kapacitoru popř. dobrá detekce vysílaného signálu mnohem horší [20].




Obr. 2.4: Princip RFID komunikace [21].

## 2.3 Dělení transpondérů (tagů)

Transpondér má již mnoho modifikací co se týče velikostí a tvarů. Tato část se zabývá dělením podle: použité paměti, způsobu použití, typu komunikace, napájení a použité nosné frekvence.

### 2.3.1 Podle použité paměti

#### RO (Read Only)

Je o typ pamětí, které jsou již naprogramovány z výroby a nelze je přeprogramovat. Jsou určeny pouze pro čtení podobně jako vylisované CD-ROM. Většinou obsahují pouze neměnné číslo v podobě identifikačního čísla zvířat, osoby či věci. Jsou obdobné čárovým kódům. Kapacita paměti je obvykle od 40 do 512 bitů [22].

#### WORM (Write Once Read Many)

Tento druh je pouze jednou programovatelný. Z výroby je prázdný a programuje se až u dodavatele. Tato paměť je podobná prázdnému CD-ROM, kde po vypálení dat je nelze změnit. Po naprogramování pak slouží jen pro čtení, protože nelze paměť přepsat. Někteří výrobci udávají, že je tento druh pamětí programovatelný vícekrát, ale bez záruky správného přeprogramování. Kapacita paměti je stejná jako u Read Only paměti [21], [22].

#### RW (Read Write)

Tyto paměti mohou uchovávat velké množství dat, protože používají adresovanou paměť. Pasivní transpondéry disponují pamětí od 386 bitů do 8 kbit a aktivní mají většinou od 16 kbit až do 2 Mbit. Počet cyklů přepisu paměti se uvádí až tisíckrát. Tento druh paměti je podobný nosiči CD-RW, kde lze libovolně data přepisovat [22].

#### Kombinace Read Only a Read Write

Mohou se používat kombinace pamětí pouze pro čtení a přepisovatelných pamětí. Jako příklad poslouží paleta, kde paměť pouze pro čtení bude označovat pořadové číslo palety po celou dobu její životnosti. Přepisovatelná paměť bude obsahovat soupis zboží na této paletě, které se neustále bude měnit [22].

### 2.3.2 Podle typu komunikace

#### Induktivní metoda

Tato metoda dosahuje čtecí vzdálenosti v řádech desítek centimetrů a je označována jako Low Range nebo také Near Field Comunication (NFC, komunikace v blízkém poli) [19].

Transpondér v sobě obsahuje čip pro uchování dat a cívku, která funguje jako anténa. Čtečka vysílá do prostředí vysokofrekvenční magnetické pole, které zachytí anténa transpondéru (funguje jako cívka – induktor). Tím se na závitech cívky

vytvoří napětí, které se usměrní a dále se akumuluje na kapacitoru (kondenzátoru). Po vytvoření dostatečného napětí se spustí elektronika transpondéru a vygeneruje odpověď. Tento princip je založen na vzájemné indukci dvou cívek (induktorů), kde primární cívku obsahuje čtecí zařízení a sekundární transpondér (viz. obr. 2.4) [22].

#### Odrazová metoda

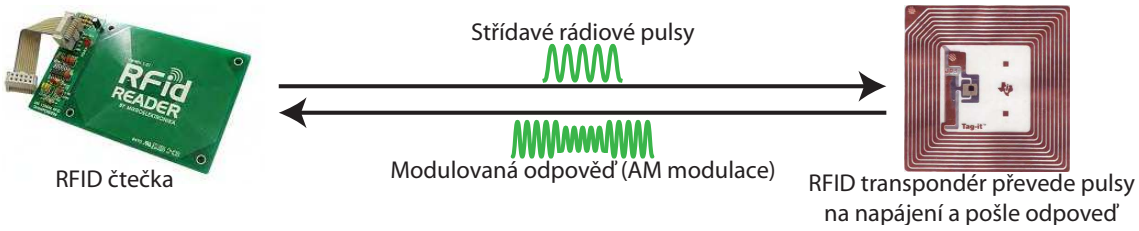
Tato metoda má větší dosah, řádově jednotky metrů a označuje se jako Far Field (komunikace vzdáleným polem). Využívá podobného principu jako radar, kde část energie vyzařována anténou čtecího zařízení dorazí k transpondéru ve formě vysokofrekvenčního signálu. Ten je po úpravě použit pro čip na řízení rezistoru (odpor), který mění impedanci antény. Samotná změna impedance antény vyvolá odraz signálu směrem ke čtecímu zařízení, které tuto zprávu musí detekovat [22].

### 2.3.3 Podle napájení

#### Pasivní

Pasivní transpondéry jsou mnohem běžnější než aktivní. Jsou levnější, protože neobsahují vlastní zdroj elektrické energie. Veškerou energii pro funkci elektronických obvodů získávají elektromagnetickou indukcí z pole vyzařovaného čtecím zařízením. To znamená, že bez čtecího zařízení transpondér nefunguje. Dosahují malých čtecích vzdáleností (od pár centimetrů do jednoho metru) a velmi dlouhou životností. Většinou se používají k jednorázovým účelům jako je označení dražšího zboží [22].

Komunikace mezi čtecím zařízením a pasivním transpondérem je na následujícím obrázku.




Obr. 2.5: Komunikace mezi pasivním transpondérem a čtecím zařízení [20], [22].

#### Aktivní

Tento druh transpondérů obsahuje vlastní zdroj elektrické energie ve formě baterie pro napájení mikročipu a posílení vysílaného signálu. Dosahují větší čtecí vzdálenosti oproti pasivním transpondérům, běžně do 100 metrů (podle velikosti vysílaného výkonu a použité frekvence). Dají se vhodně doplnit různými fyzikálními senzory

a tím mohou poskytovat informace o objektech, ke kterým jsou připevněny. Jsou schopny pracovat i bez dosahu čtecího zařízení. Pokud detekují čtecí zařízení ihned začnou vysílat svoje data. Využívají se spíše při lokalizaci objektu. Nevýhoda je ovšem u tohoto typu vysoká cena a kratší životní cyklus díky kapacitě baterie [22].

Princip komunikace je zřejmá z obr. 2.6.


Obr. 2.6: Komunikace mezi aktivním transpondérem a čtecím zařízení [20], [22].

#### Semi-pasivní

Kombinuje výhody pasivních a aktivních transpondérů. Pracuje na stejném principu jako pasivní transpondér s tím rozdílem, že obsahuje miniaturní baterii, která zlepšuje kvalitu vysílané odpovědi. Nebo může obsahovat nabíjecí baterii, která uchová energii od čtecího zařízení pro budoucí komunikaci. Mají větší citlivost oproti pasivním transpondérům a proto dosahují delších čtecích vzdáleností. Tento způsob se využívá například u elektronického mýtného [19], [22].

### 2.3.4 Podle používané frekvence

Systém RFID využívá elektromagnetické vlny o různých vlnových délkách. Jsou tvořeny pohybujícími elektrony, které se skládají z oscilujících, navzájem kolmých, elektrických a magnetických polí. Podle frekvence, resp. délce rádiové vlny, mohou nebo nemusí projít různými druhy materiálů. Pracující kmitočet je důležitým parametrem pro čtecí dosah a vzájemné působení s okolním prostředím. Větší frekvence znamená rychlejší přenos dat na delší vzdálenosti na úkor vetší citlivosti na komunikující prostředí (různé materiály, rušení, odrazy atd.) [22].

Proto je nedůležitější volba frekvence, kterou bude systém RFID využívat. Při zvolené frekvenci se také musí uvažovat maximální výkon, který je určován příslušnými normami a zákonem daného státu. Výhody větších frekvencí jsou malé rozměry transpondérů s tím spojené menší náklady na výrobu [22].

Nízké a vysoké frekvence používají induktivní vztah mezi anténami, kde je nositelem informace elektromagnetické pole [20].

Tyto frekvenční pásma zastřešuje norma ISO 1800.

### Nízká frekvence (Low Frequency)

Toto frekvenční pásmo má velmi krátkou vzdálenost komunikace (do 20 cm) a nízkou přenosovou rychlost. Frekvence se pohybuje se od 125 – 134 kHz. Pásmo se využívá převážně v identifikačních průkazech, pro identifikaci výrobku a na evidenci domácích zvířat. Převážně se s tímto frekvenčním pásmem používají pasivní transpondéry, které se skládají z krouceného měděného drátu a nepřepisovatelné paměti [22].

### Vysoká frekvence (High Frequency)

Používaná frekvence je zde 13,56 MHz. Dosahuje větší čtecí vzdálenosti oproti nižším frekvencím (do 1 metru). Při použití aktivních nebo semi-aktivních transpondérů naroste čtecí vzdálenost až do 4 metrů. Přesto se výhradně využívají pasivní transpondéry. Tato frekvence je nejvhodnější volbou pro rušené nebo stíněné prostředí (dokáže projít kovem i tekutinou). Antény transpondérů mohou být opět pomocí měděného krouceného drátu nebo pomocí vodivého inkoustu vytištěného na papír. Zde se používají transpondéry nejen pro čtení, ale také i pro zápis s kapacitou do několika kilobytů. Nejčastější využití tohoto pásma je pro knihovní systémy a pro identifikační karty (elektronické peněženky a přístupové systémy). Toto pásmo je standardizováno v normě ISO 14443, kterou využívá také systém NFC [20], [22].

### Ultra vysoká frekvence (Ultra High Frequency)

Pásmo ultra vysoké frekvence je od 860 – 960 MHz a umožňuje rychlejší přenos na delší vzdálenosti do jednotek metrů. Toto frekvenční pásmo je nejvíce využívané pro identifikaci zboží a logistických jednotek a má přiděleny různé frekvence (subpásma) v různých zemích světa. Například pr Evropu je přiděleno 865,6 – 867.6 MHz a pro spojené státy Americké s Kanadou je přiděleno 902 – 928 MHz. Tento systém má zaveden jednotný číselný standard EPC (Electronic Product Code), který je spravován světovou organizací EPC Global. Ta přiděluje určité rozsahy použitelných čísel lokálním registrátorům (EAN[1] ČR). Ty to pak rozdělí výrobcům pro jednoznačnou identifikaci výrobku v rámci celého světa [21], [22].

### Mikrovlnné pásmo (Microwave Frequency)

Toto pásmo využívá frekvenci 2,45 GHz, které je blízké u často používané WiFi sítě. Mají velkou čtecí rychlost na velké vzdálenosti (desítky metrů), ale s velmi velkou citlivostí na prostředí a překážky. V tomto pásmu se využívají většinou aktivní transpondéry, protože pasivní by neměli dostatečný výkon na komunikaci. Využití je například pro identifikaci vozidel či lokalizace pohybujících se předmětů [20], [22].

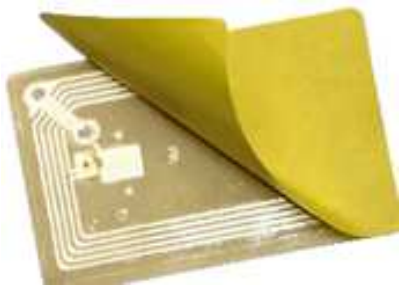
[1] Dříve European Article Number a po přejmenování International Article Number.

### 2.3.5 Podle způsobu použití

Provedení tvaru transpondéru úzce souvisí s tím jak bude využit. Některé typy musí být odolné velkým teplotám a vlhkosti, jiné zase musí mít malé rozměry nebo vysokou mechanickou odolnost.

#### RFID transpondér lepící

U tohoto typu jsou RFID transpondéry přilepeny na substrát z jedné strany a ze strany druhé mají lepící vrstvu. Mohou tak být jednoduše připevněny na produkt pomocí lepicí strany. Lze je kombinovat s čárovými či QR kódy, které se vytisknou na stranu bez lepidla [20]. Ukázka je na následujícím obrázku.



Obr. 2.7: RFID transpondér lepící [20].

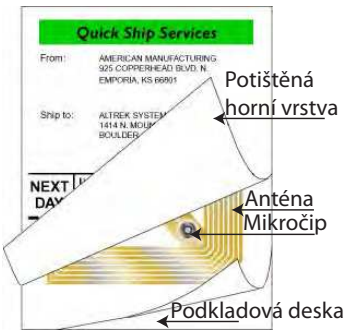
#### RFID chytrá karta (RFID Smart Card)

Karta uvedená na obrázku 2.8 kombinuje dotykovou technologii a bezdotykovou. Dotyková je starší, ale stále používaná. Je složena z pozlacených nebo postříbřených plošek (na obrázku vlevo uprostřed), které poskytují elektrickou konektivitu při vložení do čtecího zařízení. Bezdotyková technologie je ovšem RFID, která má na kartě symbol vpravo nahoře. RFID transpondér je zabudován mezi vrstvy plastu. Díky standardním rozměrům karty může mít RFID poměrně velkou anténu, která tak zvětšuje dosah komunikace. Používají se pro bezpečnostní a platební systém [22].


Obr. 2.8: RFID chytrá karta (RFID Smart Card) [20].

### RFID chytrá etiketa/plakát (RFID Smart Label/Poster)

Jedná se o papírovou či plastovou potištěnou etiketu se zabudovaným pasivním nebo semi-pasivním transpondérem. Jako podklad se často volí silikonová podložka (často i lepící). Tento druh lze také kombinovat s čárovými nebo QR kódy. Některé chytré etikety/plakáty mají odnímatelnou horní vrstvu a programovatelný transpondér pro vícenásobné využití [22].




Obr. 2.9: RFID chytrá etiketa (RFID Smart Label) [22].

### Skleněné RFID transpondéry

Skleněné transpondéry jsou pro identifikaci zvířat a osob. Tím jak jsou malé (10 až 30 mm) je lze lehce implantovat pod kůži. Obal je skleněný nebo silikonový [22].

Obr. 2.10: Skleněné RFID transpondéry [22].

### RFID klíčenky, knoflíky a ostatní

Existuje celá řada tvarů a provedení. Pro názornou ukázku slouží následující obrázky.


Obr. 2.11: Ostatní druhy RFID transpondérů [20].

# 3 NFC (NEAR FIELD COMMUNICATION)

Near field communication známé pod zkratkou NFC je sada standardů, která definuje bezdrátovou technologii na velmi krátkou vzdálenost (obvykle jen jednotky cm) propojující dvě zařízení. Přestože je NFC poměrně mladá technologie, její využití den co den rychle stoupá a potenciál do budoucna je obrovský. NFC je velice úsporná technologie pro přenos dat a proto se hodí do mobilních a přenosných zařízení [24].

Tato technologie nedisponuje takovými rychlostmi jako je tomu například u technologie WiFi v mobilních zařízeních (až 54 Mbit/s), ale podporuje přenos dat od 106 kbit/s jen do 424 kbit/s. Rychlost přenosu dat záleží na použitém standardu. Tato technologie se nepoužívá k přenosu velkých objemů dat, ale lze využít jinou technologii například Bluetooth, kde se pomocí NFC spárují přístroje a samotný přenos dat jde přes Bluetooth [24].

Tato technologie je založena na standardech RFID (Radio Frequency Identification, viz. kap. 2) zahrnující standardy ISO/IEC 14443 a FeliCa. Tyto normy zahrnují ISO/IEC 18092 definované neziskovou organizací NFC Fórum [24]. Přehled použitých norem a návaznost na normy definované pro RFID jsou popsány na začátku kapitoly 3.3.

Obr. 3.1: NFC logo pro certifikovaná zařízení [27].

## 3.1 Využití NFC

Nejpraktičtější využití NFC technologie je tzv. peněženka v mobilním telefonu. Tuto funkci obstarávají například aplikace Google Wallet nebo O2 Wallet, které zajistí správu debetních/platebních karet. Peněženka v mobilním telefonu se velice rychle rozšiřuje a to dokonce i v ČR, kde některé hypermarkety či banky již podporují tento způsob platby. Při placení stačí přiložit zařízení podporující NFC k terminálu, kde si aplikace posléze vyžádá zadání bezpečnostního kódu. Po zadání kódu se opět přiloží zařízení k terminálu, kde se začnou odesílat autorizační data a po krátké chvilce se potvrdí transakce [26], [25].

V ČR zatím není Google Wallet oficiálně podporovaná, ale existují neoficiální modifikace, které umožňují nasazení přes PayPass systém. O2 Wallet je podporo-

vána v nákupních řetězcích Interspar a Globus. Ovšem pro podporu O2 Wallet je nutno mít speciální SIM kartu s bezpečnostním čipem, kde se uchovávají informace o emulované platební kartě [25].

Probíhají diskuze ohledně ochrany proti zneužití tohoto způsobu využití. Samotná aplikace po přiložení k terminálu NFC požaduje heslo pro spuštění. Pro zaplacení útraty se využívá systém, kde terminál vyžaduje PIN náhodně při nákupu do 500 Kč a nad tuto částku se musí PIN pro transakci zadávat vždy [25].

Další využití NFC je na bázi sjednocení průkazů, jízdenek a kupónů na jedno místo v kombinaci s peněženkou. Mezi průkopníky v České Republice se řadí město Plzeň, kde uživatelé mohou využívat některé z řady služeb. Pomocí telefonu s NFC ho lze využít jako jízdenku městské hromadné dopravy, slevový kupón či průkaz do knihovny. Plzeň však pokročila dál a vybavila zastávky terminály s podporou technologie NFC, kde si uživatel může pouhým přiložením zjistit aktuální jízdní řád [25].

Pomocí NFC technologie lze zabezpečit dům a otvírání všech dveří popř. použít jako klíč od vozu. Odpadá tak nutnost nosit u sebe různé klíče, přístupové karty, či zadávání přístupových kódů. [25].
NFC technologii lze využít jako [24], [25]:

- občanský průkaz, řidičský průkaz, pas,
- klíč od vozu a jiných dopravních prostředků,
- elektronický klíč, například k vašemu PC,
- vzájemná konfigurace Bluetooth a Wi-Fi zařízení,
- přístup do WiFi zabezpečené technologií WPS[1],
- sdílení kontaktů, fotografií, videí nebo souborů,
- a mnoho dalšího ...

Z této kapitoly vyplývá přínos NFC pro pohodlí člověka a velký potenciál využitelnosti do budoucna. Problém je vybití baterky při integrovaných službách do mobilního zařízení, to pak uživatel ztratí přístup k většině možnostem NFC. Také se uvádí fakt, že pokud bude zařízení s NFC odcizeno, dojde tím ke ztrátě všech důležitých věcí integrovaných v zařízení (občanský průkaz, elektronický klíč, pas atd.). Samozřejmě NFC podporuje i šifrování a většina aplikací ještě před spuštěním požaduje heslo na přístup.

Detailnější rozbor bezpečnosti NFC technologie je popsán v kapitole 3.4.

[1]WPS – Wi-Fi Protected Setup. Jedná se o jednoduché nastavení bezdrátové WiFi sítě pomocí tlačítka a PINu na zařízení. Zdroj: <`http://en.wikipedia.org/wiki/Wi-Fi_Protected_Setup`>

## 3.2 Historie NFC a RFID

První patent byl uznán Charlesu Waltonovi roku 1983 se zkratkou RFID. Další léta se podílely na řadě vylepšení a standardů technologie RFID. Až v roce 2003 bylo schváleno NFC jako ISO/IEC standard. Rok na to zakládají firmy Nokia, Philips a Sony neziskovou organizaci NFC Fórum, která navrhuje standardy na komunikaci pomocí NFC. V dnešní době NFC Fórum má přes 200 členů a s rozmachem NFC stále narůstá.V letech 2006–2007 byly vytvořeny počáteční specifikace pro NFC, avšak nemá velký úspěch, protože byla technologie nákladná a neměla dostatečnou podporu na trhu [23], [24].

První mobilní telefon s podporou NFC byla Nokia 6131 v roce 2006. První Android s podporou NFC je verze Gingerbread, který byl představen na Samsung Nexus S v roce 2010. Do propagace NFC se přidává Google Inc., kde NFC demonstruje k zahajování her, sdílení kontaktů, URL, aplikací a dalšího obsahu pomocí Android Beam. V roce 2011 NFC podporuje i OS Symbian. Ve stejném roce získala firma RIM certifikaci na podporu MasterCard. Tento rok se označuje jako rok Near Field komunikace, protože se pořádalo mnoho konferencí, výrobci mobilních telefonů začali plně podporovat a propagovat NFC a největší přínosy měli pilotní projekty na platby pomocí NFC [23], [24].

## 3.3 Technické specifikace NFC

Na nejnižších vrstvách jsou NFC standardy odvozeny od skupiny standardů bezkontaktních karet, které specifikují kódování, modulační schémata, přenosové rychlosti a technologické specifikace RFID transpondérů (známé jako RFID tag či RFID štítek). Zejména sem patří standardy ISO/IEC 14443, JIS X 6319 (Sony FeliCa), které pracují na frekvenci 13,56 MHz s přenosovými rychlostmi od 106 kbit/s do 424 kbit/s a to do vzdálenosti několika centimetrů. Další použitý standard je ISO/IEC 15693. Ten dosahuje vzdáleností výrazně vyšších a to až do 1,5 metru. Při této vzdálenosti je ovšem pomalejší přenos dat a to maximálně do 26 kbit/s [24], [26].

### 3.3.1 Fyzická a spojová vrstva NFC

Fyzická vrstva odpovídá standardům NFCIP-1 (Near-Field Communication Interface and Protocol 1 - ISO/IEC 18092) publikován v roce 2004 a novější NFCIP-2 (ISO/IEC 21481) publikován v roce 2012. Ty jsou odvozeny od ISO/IEC 14443 (sada standardů pro komunikaci v systému RFID). NFC také využívá pasivní (bez vlastního napájení) a aktivní (má vlastní napájení) transpondéry stejně jako RFID

(kap. 2.3.3). Využívá induktivní metodu, která je popsána v kap. 2.3.2 a používá pásmo vysoké frekvence popsané v kap. 2.3.4 stejně jako RFID [27].

### ISO/IEC 14443

Tento standard definuje základní elementy v komunikaci, základní požadavky, fyzické vlastnosti, maximální vysílací výkony a protokoly pro iniciaci komunikace, antikolizní protokoly a přenosové protokoly [24].

Standard ISO/IEC 14443 definuje dva typy komunikačních rozhraní mezi čtecím zařízením a transpondérem [24]:

- **Rozhraní A** – Pro komunikaci od čtecího zařízení k transpondéru se používá ASK [2] se 100% hloubkou modulace. Data jsou kódována pomocí modifikovaného Millerova kódování. Pro komunikaci od transpondéru ke čtecímu zařízení je použita OOK[3] modulace s kódováním typu Manchester[4].
- **Rozhraní B** – Pro komunikaci od čtecího zařízení k transpondéru se používá ASK s 10% hloubkou modulace. Používá se zde kódování NRZ-L[5]. Pro komunikaci od transpondéru ke čtecímu zařízení je použita BPSK[6] modulace s kódováním NRZ-L.

Dále se standard ISO/IEC 14443 dělí na 4 části, jak pro rozhraní A tak i pro rozhraní B, následovně [24]:

- **Část 1** – Fyzikální charakteristiky bezkontaktních čipových karet – Definuje fyzikální charakteristiky bezkontaktních čipových karet a jejich požadavky na ně.
- **Část 2** – Vysílací výkony a signálové rozhraní – Definuje vysílací výkony, způsob napájení čipu z radiofrekvenčního magnetického pole, signalizační rozhraní, jejich signalizační schémata a typy modulací pro oba směry komunikace (čtečka -> transpondér a naopak), a to jak pro pasivní a aktivní transpondéry.
- **Část 3** – Inicializační a antikolizní protokoly – Definuje inicializační a antikolizní protokoly pro oba typy transpondérů (aktivní a pasivní), společně s antikolizními příkazy, odpověďmi, datovými rámci a časováním.

[2] ASK (Amplitude Shift Keying) – přenáší binární informaci pomocí změn amplitudy modulovaného signálu. Zdroj: <http://en.wikipedia.org/wiki/Amplitude-shift_keying>.

[3] OOK (On-Off Keying) modulace, která označuje základní dvoustavovou ASK modulaci. Zdroj: <http://en.wikipedia.org/wiki/On-off_keying>.

[4] Přechod z vysoké úrovně na nízkou úroveň znamená 1. Naopak přechod z nízké do vysoké úrovně znamená 0. Zdroj: <http://en.wikipedia.org/wiki/Manchester_code>.

[5] Non Return to Zero Level – kódování, kde „1“ znamená vyšší úroveň a „0“ nižší úroveň, ale nenulovou. Zdroj: <http://ckp.made-it.com/encodingschemes.html>

[6] Binary-Phase Shift Keying – založena na posunutí fáze harmonické nosné pro 0° nebo 180° s unipolárním binárním signálem. Zdroj: <http://cs.wikipedia.org/wiki/BPSK>

- **Část 4** – Protokoly pro přenos – Určuje, které protokoly jsou určeny pro vysokoúrovňový přenos dat. Všechny tyto protokoly v rámci této části standardu jsou volitelné.

#### Standardy NFCIP

Standard NFCIP-1 vznikl rozšířením standardu ISO/IEC 14443 a přizpůsobením pro komunikaci NFC. Standard NFCIP je ekvivalent pro komunikaci pomocí skupiny protokolů TCP/IP. Je rozšířen o komunikační režimy, transportní protokoly a protokoly pro přenos dat. Jsou zde navrženy normy pro modulační schémata, přenosové rychlosti, strukturu a detekci rámce, detekci chyb, detekci zařízení, výběr komunikačních parametrů a protokolů a nakonec i samotnou výměnu dat [24], [27].

Tento standard zahrnuje LLCP (Logical Link Control Protocol) pracující na spojové vrstvě NFC pro podporu peer-to-peer komunikace (více viz. 3.3.2). LLCP je odvozen od standardu IEEE 802.2. Specifikace definuje dva typy služeb, spojově orientovanou (podobné TCP – Transmission Control Protocol) a nespojově orientovanou (podobné UDP – User Datagram Protocol). Služba založená na spojové komunikaci (navazuje logické spojení mezi zařízeními před samotným posíláním dat) podporuje číslování rámců, spolehlivou komunikaci, řízení toku dat a mechanizmus na opravu chyb. Oproti tomu služba bez navazování spojení nepodporuje spolehlivé doručení ani řízení toku dat [27].

Všechna zařízení, která podporují standard NFCIP-1 musí operovat na přenosových rychlostech 106, 212 nebo 424 kbit/s. Dle potřeby mohou zařízení komunikační rychlost přepínat. Změna rychlosti v rámci komunikace je jednoduše realizována pomocí změny parametrů procedury. Na rozdíl od rychlosti nelze měnit režim komunikace z aktivního na pasivní a naopak [24].

Standard NFCIP-2 vznikl rozšířením stávajícího standardu NFCIP-1. Specifikuje mechanizmy výběru správného komunikačního režimu s ohledem na již existující komunikaci RFID, kterou nesmí rušit. Zařízení podporující tento standard musí podporovat všechny specifikace ze standardů ISO/IEC 14443 a ISO/IEC 15936, aby byla zaručena zpětná kompatibilita s již existujícími bezkontaktními systémy [24].

### 3.3.2 Režimy přenosu

NFC je víceúčelová technologie pro přenos jakýchkoliv dat oproti RFID, kde se přenášely pouze informace o výrobku nebo se posílalo jen identifikační číslo. NFC umožňuje další režimy přenosů pro komunikaci mezi zařízeními. Navíc uchovalo zpětnou kompatibilitu k systému RFID na kmitočtu 13,56 MHz.

#### Čtení/zápis (Read/Write)

Režim čtení/zápis umožňuje nahrávat a číst data transpondérů pomocí čtecího zařízení. Transpondéry mohou být aktivní i pasivní. Tento režim komunikace se převzal ze standardů ISO/IEC 14443 (typu A či B) a FeliCa (JIS X 6319). Není zde zapotřebí žádné zabezpečení neboť komunikace většinou neobsahuje důležitá data, která je potřeba přenášet bezpečnou cestou (šifrování a integrita). Ani transpondéry nedisponují bezpečným úložištěm svých dat. Maximální rychlost pro zápis se udává do 106 kbit/s [24].

#### Emulace karty (Card emulation)

Tento režim emuluje NFC čipovou kartu na aktivním zařízení s NFC podporou. Při zapnutí tohoto režimu se zařízení začne chovat jako pasivní NFC čipová karta, která je definována ve standardu ISO/IEC 14443. V tomto režimu musí začít komunikaci čtecí zařízení. Běžný případ použití je například jako jízdenka, vstupenka, debetní/kreditní karta či libovolná forma autentizačního faktoru ve formě pasivního čipu. Zde jsou poskytovány služby bezpečné komunikace a bezpečné úložiště citlivých dat [24].

#### Rovný s rovným (Peer-to-peer/P2P)

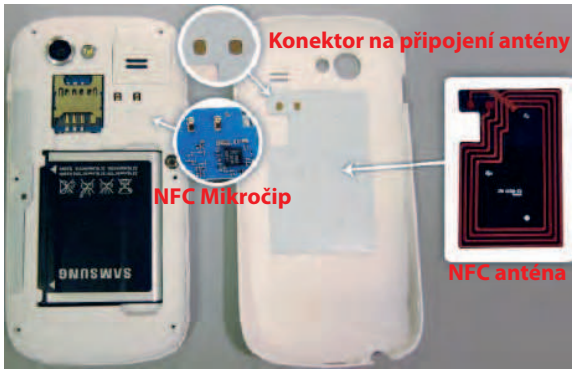
Režim P2P je novinkou ve způsobu přenosu dat mezi zařízeními podporující NFC. Je definován ve standardu NFCIP-1. Tento režim je podobný režimu P2P z počítačových sítí založených na sadě protokolů TCP/IP, který umožňuje obousměrnou výměnu dat mezi zařízeními. Komunikace probíhá v režimu half-duplex, kde v daném okamžiku může komunikace probíhat pouze v jednom směru (směr přenosu se rychle střídá). Také se předpokládá, že v tomto režimu jsou zařízení aktivní po celou dobu komunikace. Maximální přenosové rychlosti dosahují 424 kbit/s [24], [27].

### 3.3.3 Implementace NFC do mobilního zařízení

Zde jsou popsány možnosti implementace NFC technologie do zařízení. U nových zařízení jsou NFC mikročipy implementovány přímo v těle zařízení. U starších bohužel NFC nebylo k dispozici, avšak lidé chtěli tuto technologii využívat bez nutnosti koupě nového zařízení. Pro zprovoznění technologie NFC na zařízeních, které neměli zabudovanou podporu NFC, byly vymyšleny alternativy v podobě umístění NFC mikročipu do SD karty nebo SIM karty [28].

### V mobilním zařízení

Novější typy zařízení mají NFC čip integrovaný přímo na základní desce přístroje. Problém však byl v umístění antény. Jelikož NFC pracuje na frekvenci 13,56 MHz, musí být anténa poměrně velká a většinou není možnost ji instalovat přímo na základní desku přístroje (myšleno u mobilních telefonů kde se šetří místem). Proto byly vymyšleny alternativy, kde se NFC anténa instalovala na zadní kryt zařízení (obr. 3.2) nebo na baterii (obr. 3.3) [43]. Další umístění antény může být zabudovaná v krytu okolo přístroje [28].



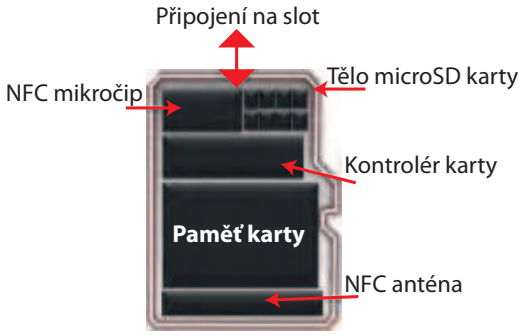

Obr. 3.2: Implementace NFC v mobilním telefonu Nexus S [43].


Obr. 3.3: Implementace NFC v mobilním telefonu Galaxy Nexus [43].

### Na SD kartě

Toto je první alternativní řešení implementace technologie NFC, například od výrobce Tyfone, kde přidal NFC mikročip spolu s anténou do paměťové karty (obr. 3.4). Nevýhoda je ta, že díky malé anténě je dosah NFC příliš malý. Mobilní telefon se musí přiložit přímo ke čtecímu zařízení. Pro správnou funkci musí být v zařízení

nainstalovaná aplikace, která spolupracuje s tímto NFC mikročipem. Samozřejmě toto řešení disponuje bezpečným úložištěm pro citlivá data. Ve většině případů toto řešení funguje jen pro režim přenosu emulace karty (kap. 3.3.2), avšak jsou již řešení, které podporují i ostatní režimy přenosu. Mezi další nevýhody by se mohlo zařadit to, že ne všechny zařízení mají slot na karty. Výrobce Tyfone podporuje řešení pro SD, miniSD a microSD karty. Navíc se ve specifikacích uvádí, že pokud má zařízení kovový slot na kartu nebo je karta umístěna pod baterií či z boku zařízení, není zaručena správná funkce NFC [29].

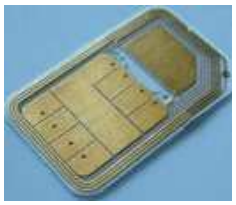



Obr. 3.4: Technologie NFC na paměťové kartě microSD [29].

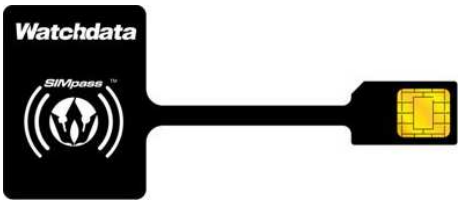

Obr. 3.5: MicroSD podporující NFC [29].

### Na Sim kartě

Další alternativa zabudování technologie NFC přišla s implementací mikročipu na SIM kartu, protože SIM kartu mají všechny mobilní telefony. Tím se vyhne nevýhodě u SD karet, kde přistroj nemusel mít patřičný slot na kartu. Při požadavku na SIM katu s NFC technologií vaši stávající SIM kartu naklonují do nového pouzdra obsahující NFC mikročip. Jsou zde opět nevýhody tohoto řešení jako u SD karet, kde některé přístroje mají umístění SIM karty pod kovovým rámečkem nebo pod baterkou. Existují dvě varianty SIM karet. První varianta má okolo svého těla zabudovanou anténu (obr. 3.6), avšak toto řešení nemá opět dobrý dosah. Druhá varianta je SIM karta s vyvedenou anténou, která se přilepí například na baterii nebo na zadní kryt zařízení (obr. 3.7). Tato varianta odstraňuje potíže s dosahem NFC a s umístěním SIM karty v zařízení [30], [31].



Obr. 3.6: Technologie NFC zabudovaná na SIM kartě s interní anténou [30].




Obr. 3.7: Technologie NFC zabudovaná na SIM kartě s externí anténou [31].

### 3.3.4 Druhy NFC transpondérů (tagů)

V současné době je známých víc druhů transpondérů podporujících NFC technologii. Většina z nich je založena na starších standardech pro RFID komunikaci. Nezisková organizace NFC fórum definuje 4 typy transpondérů, které podporují standardy NFCIP-1 a NFCIP-2 spolu s datovou strukturou NDEF (NFC Data Exchange Format) pro přenos jakýchkoliv dat. Rozdíly transpondérů jsou v komunikaci, použití, možnosti šifrování, rychlosti přenosu dat a v neposlední řadě i ceně. Následující tabulka shrnuje 4 typy transpondérů definované NFC fórem. [27].

Tab. 3.1: Druhy transpondérů definované NFC fórem [32].

| | **Typ 1** | **Typ 2** | **Typ 3** | **Typ 4** |
|---|---|---|---|---|
| **Založeno na standardu** | ISO/IEC 14443 Typ A | ISO/IEC 14443 Typ A | FeliCa | ISO/IEC 14443 Typ A, typ B |
| **Název čipu** | Topaz | MIFARE | FeliCa | DESFire, SmartMX-JCOP |
| **Velikost paměti** | do 1 kB | do 2 kB | do 1 MB | do 64 kB |
| **Přenosová rychlost** | 106 kbit/s | 106 kbit/s | 212 kbit/s | 424 kbit/s |
| **Zabezpečení** | 16 nebo 32bitový digitální podpis | nezabezpečeno | 16 nebo 32bitový digitální podpis | volitelně |
| **Cena** | nízká | nízká | vysoká | průměrná až vysoká |
| **Poskytované výrobci** | Innovision Research and Technology | Philips/NXP | Sony | různí výrobci |
| **Případy užití** | Jednoúčelové tagy | Jednoúčelové tagy | Flexibilní tagy s širokými možnostmi užití | Flexibilní tagy s širokými možnostmi užití |

Další označení transpondérů je následující [27]:

- NFC-A – standard ISO/IEC 14443-3A (RFID A),
- NFC-B – standard ISO/IEC 14443-3B (RFID B),
- NFC-F – standard JIS X 6319-4 (FeliCa),

- NFC-V – standard ISO/IEC 15693,
- ISO-DEP – standard ISO/IEC 14443-4,
- MIFARE – standard ISO/IEC 14443 A,

Tyto typy však nedisponují standardem NCFIP-2 ani NFCIP-1, tudíž nevyužívají datový formát NDEF pro komunikaci.

### 3.3.5 NDEF – NFC Data Exchange Format

Dříve zavedené standardy pro přenos dat pomocí RFID podporovaly pouze přenos identifikačních dat (URI – Uniform Resource Identifier či znaky z ASCII tabulky). Proto byl zaveden rozšiřující standard pro přenos jakýchkoliv dat. S tím přišla organizace NFC fórum, která specifikovala datový formát NDEF (NFC Data Exchange Format). NDEF tedy definuje formát zapouzdření zpráv pro výměnu dat mezi NFC zařízeními. Podporuje přenosy mezi dvěma aktivními zařízeními nebo aktivním a pasivním (popř. semi-pasivním) [32].

Cílem specifikace je definice datových jednotek a pravidel pro přenos pomocí NFC. Tato specifikace staví na spolehlivé a spojově orientované komunikaci, která je definována ve standardu NFCIP-1 (ISO/IEC 18092) na podvrstvě LLCP [27].

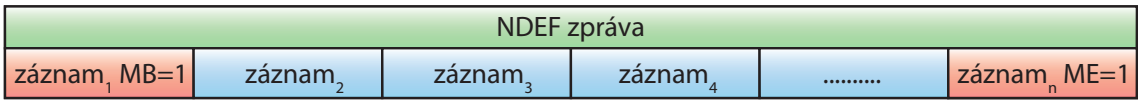
NDEF specifikace podporuje následující [32], [27]:

- Libovolné typy dat (GIF, JPEG, atd.). Poskytuje i přenos šifrovaných dat.
- Agregaci více typů dat, které spolu logicky souvisí do jedné zprávy.
- Sloučení malých bloků dat do jedné zprávy za účelem ušetření režie při přenosu.
- Libovolné velikosti souborů, dokonce i o předem neznámé velikosti (dynamicky generovaná data).

#### NDEF zpráva

NDEF zpráva se skládá z jednoho či více NDEF záznamů, kde první záznam je označen příznakem MB (Message Begin – počátek zprávy) a poslední záznam je označen příznakem ME (Message End – konec zprávy). Minimální délka zprávy je omezena na jeden záznam, které se dosáhne tím, že příznaky MB a ME se zapíší do stejného záznamu. Maximální počet záznamů ve zprávě NDEF není omezen [27].

NDEF záznamy se ve zprávě nesmí překrývat proto byl zaveden formát, který je uveden na obrázku 3.8. První záznam má index 1 (záznam MB=1) pak následují číslované záznamy nesoucí data a poslední záznam má index n (záznam ME=1), který označuje konec NDEF zprávy [27].

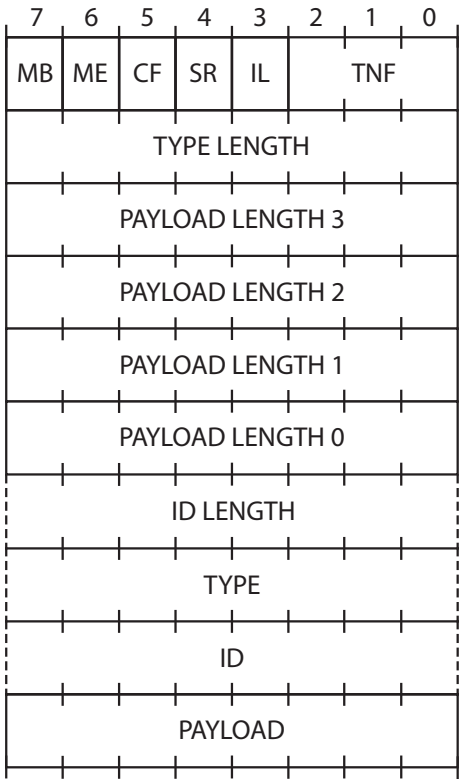



Obr. 3.8: Struktura NDEF zprávy [27].

### NDEF záznam

Všechny NDEF záznamy, které nesou data, musí obsahovat [32], [27]:

1. **Velikost dat** (Payload length) – reprezentuje velikost užitečných dat zapouzdřených v NDEF záznamu. Pro krátké záznamy je vyhrazen jeden oktet a pro normální zprávy jsou vyhrazeny čtyři oktety. Krátké záznamy jsou identifikovány bitem SR (Short Record). Nula je platná délka.
2. **Typ přenášených dat** (Payload type) – indikuje typ dat v NDEF záznamu. Konkrétní typ nastavuje uživatelská aplikace. V nabídce je hned několik typů dat, které se nastavují v poli TYPE.
3. **Volitelný identifikátor** (Payload identification) – je určen aplikacím k dodatečné identifikaci NDEF záznamu.




Obr. 3.9: Struktura NDEF záznamu [27].

Následuje znázornění struktury hlavičky NDEF záznamu a vysvětlení jednotlivých polí z obrázku 3.9 [32], [27].

- **MB (Message Begin)** – jednobitové pole, které značí začátek NDEF zprávy.
- **ME (Message End)** – jednobitový příznak, který značí konec NDEF zprávy.
- **CF (Chunk Flag)** – jednobitový příznak, který indikuje zda-li se jedná o rozkouskovaná (fragmentovaná) data do více záznamů nebo záznam obsahuje celou zprávu najednou.
- **SR (Short Record)** – opět jednobitový příznak indikující krátkou zprávu (nastaven na 1) nebo normální zprávu (nastaven na 0). Při použití krátké zprávy je vyhrazen jeden oktet na určení velikosti užitečných dat (pole PAYLOAD LENGTH) a při normální velikosti záznamu jsou použity čtyři oktety.
- **IL (ID Length)** – jednobitový příznak oznamující přítomnost polí ID a ID LENGTH v hlavičce NDEF záznamu. Při nastavení na 1 jsou pole ID a ID LENGTH uvedeny v hlavičce a každý zabírá jeden oktet. Pokud je nastavena 0 jsou tyto pole z hlavičky vynechány.
- **TNF (Type Name Format)** – jedná se o tříbitové pole, které definuje druh přenášených dat. Hodnoty jsou následující:
  - ***0x00*** – prázdný typ dat, při použití musí být pole TYPE LENGTH, ID LENGTH a PAYLOAD LENGTH nulové a pole TYPE, ID a PAYLOAD jsou vynechány ze záznamu. Používá se pokud uživatel již nemá data.
  - ***0x01*** – typ definovaný NFC fórem ve specifikaci NFC RTD (Record Type Definition). Potom pole TYPE obsahuje jméno RTD typu.
  - ***0x02*** – označuje, že se přenáší média (MIME – Multipurpose Internet Mail Extensions) a pole TYPE poté definuje konkrétní typ média dle specifikace RFC 2046.
  - ***0x03*** – nastavuje typ přenášených dat na absolutní URI (Uniform Resource Identifier). Pole Type obsahuje typ URI dle specifikace RFC 3986.
  - ***0x04*** – značí externí typ NFC fóra. Pole TYPE obsahuje jméno ze specifikace NFC RTD.
  - ***0x05*** – označuje neznámý typ dat. Při nastavení musí být pole TYPE LENGTH nulové a pole TYPE je vynecháno ze záznamu.
  - ***0x06*** – definuje nezměněný typ dat. Jedná se o typ určený pouze prostředním záznamům, který definuje, že má stejný identifikátor, jako počáteční kus záznamu. Pokud je nastaven, pak TYPE LENGTH musí být nulový a pole TYPE je vynecháno ze záznamu.
  - ***0x07*** – rezervováno pro budoucí účely.
- **TYPE LENGTH** – obsahuje osmibitové celé číslo, které definuje délku pole TYPE v bytech. Toto pole je vždy nulové pro specifické hodnoty pole TNF.
- **ID LENGTH** – obsahuje osmibitové celé číslo, které definuje délku pole ID v bytech. Toto pole je přítomno pouze v případech, kdy je nastaven příznak IL na 1. Pokud je pole ID LENGTH nastaveno na 0 bytů, pak je pole ID

vynecháno ze záznamu.

- **PAYLOAD LENGTH** – definuje délku použitých dat. Velikost tohoto pole záleží na nastavení bitu SR, kde 0 značí použití čtyř oktetů (32 bitů) a 1 značí jeden oktet (8 bitů). Jsou-li data větší jak 231 bytů musí se rozdělit (fragmentovat) a posléze u příjemce opět složit. Pokud je velikost nulová potom pole PAYLOAD je vynecháno ze záznamu.
- **TYPE** – pole o velikosti jednoho oktetu, které definuje konkrétní typ přenášených dat dle specifikace z pole TNF.
- **ID** – přiděluje jednoznačný identifikátor zprávy. Jednoznačnost zajišťuje strana generující data. Toto pole musí mít nastaven pouze první záznam, v případě fragmentované komunikace prostřední a ukončující kus záznamu nesmí mít toto pole nastaveno.
- **PAYLOAD** – pole nesoucí užitečná data aplikace.

## 3.4 Bezpečnost NFC

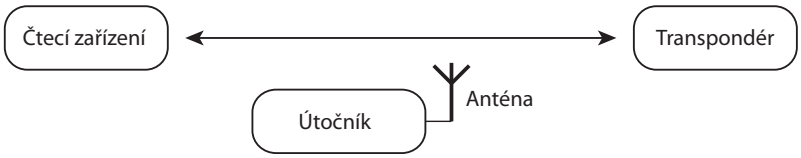
Prvním bezpečnostním aspektem technologie NFC je její krátký dosah, který je v řádek jednotek centimetrů. Ale i na tak krátké vzdálenosti se vyskytují různé bezpečnostní rizika. Technologie NFC má nezabezpečenou komunikaci, kde si musí zabezpečit průběh komunikace komunikující strany na vyšších vrstvách nebo musí použít přídavné algoritmy pro bezpečnou komunikaci. NFC se prosadila díky své jednoduchosti. Není zde ani žádná ochrana proti odposlechu komunikace, a tím může být zranitelná vůči modifikaci dat při komunikaci [24].

### 3.4.1 Způsoby narušování bezpečnosti NFC během komunikace

#### Odposlech (Eavesdropping)

Vzhledem k tomu, že NFC využívá bezdrátové komunikační rozhraní, je zřejmé, že existuje možnost komunikaci odposlechnout. Pomocí antény, zesilovače a dekódovacího zařízení lze realizovat odposlech radiofrekvenčního signálu komunikujících zařízení (obr. 3.10). NFC nijak nezabezpečuje komunikaci proti tomuto typu narušení. [24], [36].
Odposlech záleží na několika faktorech [36]:

- Radiofrekvenční charakteristika (tj. vyzařovací charakteristika) odesílatele,
- radiofrekvenční charakteristika útočníka,
- kvalita útočníkova přijímače,
- kvalita útočníkova dekodéru,




Obr. 3.10: Odposlech NFC komunikace [36].

- Místo kde se realizuje útok (prostředí, překážky, stěny, šum),
- výkon zúčastněných NFC zařízení.

Odposlech se dělí do dvou kategorií, kde záleží na typu napadeného transpondéru.

U pasivních transpondérů je situace s odposlechem celkem složitá, protože generují svoji odpověď jen při existenci čtecího zařízení v jeho okolí, které jim dodává energii. Navíc je odpověď odesílána s relativně nízkým ziskem a proto komunikace nedosahuje velkých vzdáleností. Tím se možnost odposlechu snižuje maximálně do vzdálenosti jednoho metru [36].

V případě odposlechu aktivního zařízení je situace lepší, protože dokáže vysílat svoje data i bez čtecího zařízení s větším ziskem díky vlastnímu napájení. Tím se možnost odposlechu zvětšuje na vzdálenost až 10 metrů. [36].

### Modifikace dat (Data Modification)

Narušování komunikace není složité. Stačí, aby zařízení, které pracuje na kmitočtu 13,56 MHz s větším výkonem, rušilo přenášenou posloupnost bitů (obr. 3.10). Tím bude průběh NFC komunikace náhodně modifikován a bude docházet ke špatnému vyhodnocování dat na straně příjemce. Zatím není a asi ani nebude žádná možnost jak rušení zabránit. Avšak detekce rušení komunikace je možná, protože NFC zařízení kontrolují elektromagnetické pole ve svém okolí [36].

S tím také souvisí úmyslná modifikace dat. Útočník se snaží podstrčit příjemci modifikovaná data tak, aby se mu data jevila jako validní. Tento typ je výrazně složitější a záleží na použitém kódování a hloubce modulace. Pokud chce útočník změnit přenášená data, tak musí modifikovat jednotlivé bity radiofrekvenčního signálu v přesně daném okamžiku a navíc musí použít větší vysílací výkon než vysílací zařízení, aby jednotlivé vyslané bity mohl změnit. Úspěšnost modifikace bitů (změna z 0 na 1 a naopak) je závislá na hloubce amplitudové modulace [36].

Jako příklad poslouží modifikované Millerovo kódování s hloubkou modulace 100 %, kde jen některé bity mohou být modifikovány. Hloubka modulace 100 % umožní eliminovat pauzy v radiofrekvenčním signálu, ale neumožňuje generovat pauzy, kde předtím nebyly. Tím je docíleno, že pouze bity s hodnotou 1 následované bitem s hodnotou 1 lze změnit. Všechny ostatní bity nelze měnit, jinak by

se narušilo kódování a přijímací strana by data nepřijala. U kódování typu Manchester s hloubkou modulace 10 % umožňuje útočníkovi modifikovat jakýkoliv bit v přenášené posloupnosti [24], [36].

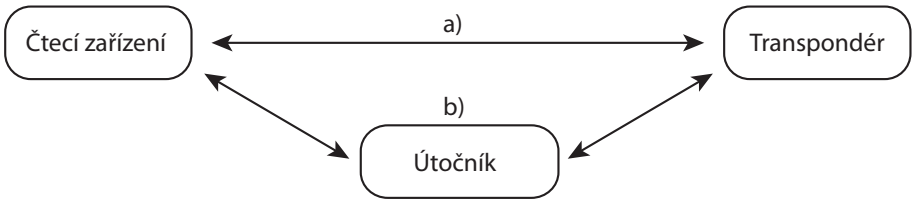
### Vkládání dat (Data Insertion)

Tento útok znamená, že útočník vloží svoji zprávu do komunikace mezi NFC zařízeními. Toto je možné jen v případě, že NFC transpondér potřebuje dlouhou dobu na odpověď ke čtecímu zařízení. Útočník pak může poslat odpověď dříve než dotazovaný transpondér. Úspěšnost závisí na rychlosti odpovědi útočníka a čekací době transpondéru. Pokud by se datové toky překrývaly, tak budou data poškozena a čtecí zařízení je vyhodnotí jako chybné [36].

### Přepojovaný útok (Relay Attack)

U tohoto typu útoku musí útočící zařízení přijmout požadavky od čtecího zařízení a následně je přeposílat k oběti útoku. Oběť po přijetí požadavků začne odesílat data (odpověď), která opět zachytí útočící zařízení a přepošle je zpět ke čtecímu zařízení. Útočící zařízení může tyto data odposlouchávat popř. i modifikovat. Vše však musí probíhat v reálném čase, aby komunikující strany nezjistily narušení komunikace [24].

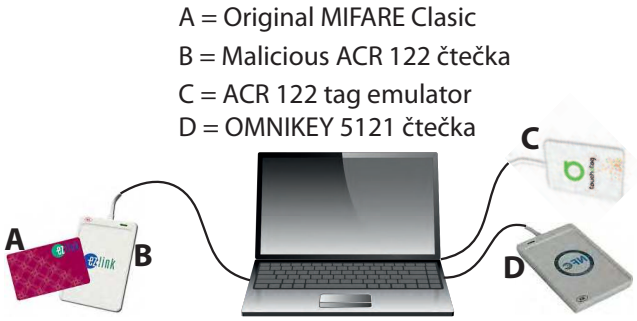
Za předpokladu, že čtecí zařízení je v aktivním režimu a transpondér pouze v pasivním režimu, následuje situace. Čtecí zařízení generuje elektromagnetické pole směrem k transpondéru, který si z toho vytvoří potřebné napájení pro odpověď. V případě, že je útočník dost blízko, může odposlouchávat komunikaci. Navíc musí generovat směrem ke čtecímu zařízení rušivé elektromagnetické vlnění, aby znemožnil komunikaci směrem od transpondéru. Čtecí zařízení však může detekovat toto rušení a odstoupit od komunikace. Dále je počítáno s tím, že čtecí zařízení nezjistilo narušení komunikace. Útočník zachytí data od transpondéru a přepošle je (už modifikované) jako odpověď čtecímu zařízení. Toto je pouze teoretické vysvětlení, které v praxi nelze jednoduše realizovat. Protože by komunikující strany detekovaly narušení a odstoupily by od komunikace [36].




Obr. 3.11: Komunikace pomocí NFC: a) normální, b) přepojovaná [36].

Další možnost je ta, že obě komunikující strany jsou v aktivním režimu. Jedna komunikující strana pošle data té druhé. Při tom musí opět útočník zachytit data a patřičným rušením zabránit jejímu doručení. Opět by tuto skutečnost mohly komunikující strany detekovat a ukončit přenos. Dále se opět předpokládá, že komunikující strany nezjistily rušení. Útočník by pak musel rychle střídat směry na odposlech a při tom by musel provádět i rušení, aby zabránil přímé komunikaci. Tento způsob by taky nešel jednoduše realizovat, protože by útočník musel vysílat na jednu stranu rušení a zároveň by druhou stranu odposlouchával [36].

Prakticky by tyto typy přepojovaných útoků v radiofrekvenčním pásmu byly zcela nemožné. Pokusy přepojovaného útoku byly realizovány[7], kde útočník byl počítač (popř. notebook) s USB NFC aktivními zařízeními. Útok spočíval v tom, že originální karta komunikovala se čtecím zařízením připojeném k notebooku. Ten následně měl k dispozici aktivní transpondér v režimu emulace karty do kterého kopíroval přijatá data od originální karty. Další čtecí zařízení si již myslí, že komunikuje s originální kartou, ale při tom komunikuje pouze s transpondérem v režimu emulace karty [36], [33].




Obr. 3.12: Přepojovaný útok pomocí notebooku s aktivními zařízeními [20], [33].

### Přerušení spojení (Interrupted connection)

Komunikace se zabezpečenými funkcemi NFC (například paměť, která obsahuje citlivá data) je chráněna časovačem, který po vypršení neumožní přístup a požaduje znovu autentizaci zařízení (uživatele). Odpočet časovače se sepne v případě, kdy nedetekuje zařízení aktivitu již autentizovaného zařízení. Pokud zařízení, které odstupuje od komunikace, neuzavře komunikační kanál, může útočník navázat komunikaci tam, kde přestalo odstupující zařízení bez opětovné autentizace [24], [37].

[7]Demonstrovány v ukázkové knihovně *libnfc* při použití dvou běžně dostupných NFC zařízení.

#### Opakované přenášení (Replay Attack)

Tento způsob narušení bezpečnosti spočívá v opakovaném přenosu originálních dat. Útočník může odposlechnout zahajování komunikace mezi zařízeními. Aniž by této komunikaci musel rozumět nebo ji nějakým způsobem měnit, uloží si ji pro pozdější opakovaný přenos. Tím docílí toho, že při vyslání takto uložených dat se může vydávat za originální transpondér a získat tak kontrolu nad komunikací. Například u NFC karty na MHD, kde stačí zachytit prvotní komunikaci a později ji pouze přehrát čtecímu zařízení a jezdit tak na účet někoho jiného. Nebo u NFC debetní/kreditní karty lze tímto způsobem odposlechnout ověřovací sekvenci uživatele při navazování spojení a posléze ji stačí pouze přehrát čtecímu zařízení a útočník pak bude mít přístup na bankovní účet [37].

### 3.4.2 Ostatní způsoby narušování bezpečnosti NFC

#### Získání citlivých dat

Toto je jedno z nejdiskutovanějších témat poslední doby. Zde vystupují hlavně debetní/kreditní popř. dopravní bezkontaktní karty a jejich ochrana před získáním citlivých dat. Dříve bylo propracované zabezpečení na RFID zařízení pomocí standardu ISO 7816, který neuchovával žádné citlivé osobní údaje na kartě, podporoval dobré šifrování a autentizaci spolu s digitálním podpisem. Jenže nastala éra NFC, který neřeší šifrování ani autentizaci uživatele. Veškeré zabezpečení je přesunuto na vyšší vrstvy, čili si zabezpečení řeší sama aplikace nebo musejí být zabudované přídavné mechanizmy pro bezpečnou komunikaci. Tím jsou lehce napadnutelné všechny zařízení, které nepoužívají zabezpečené úložiště dat (tzv. secure element[8]) [34].

Pomocí zařízení (například PC, notebook či mobilní telefon) s podporou NFC a vytvořeným programem na odcizení dat, lze celkem snadno získat data z debetních/kreditních či dopravních bezkontaktních karet [34].
Data, která byla úspěšně přečtena v testu bezpečnosti (viz. obrázky 3.13, 3.14) [34]:

- Držitel karty – jméno a příjmení,
- PAN (Primary Account Number) – identifikační číslo účtu,
- datum vypršení platnosti karty,
- transakční historie,
- a všeobecné informace o kartě.

Pomocí takto odcizených údajů pak lze snadno zaplatit zboží kartou na internetu či udělat kopii karty.

[8]Speciální paměť v NFC zařízení, která uchovává data v šifrované podobě a jsou přístupná jen po zadání šifrovacího klíče, který se odvodí od zadaného hesla.


Obr. 3.13: Přečtení NFC debetní karty pomocí PC s NFC USB čtečkou a programem [34].


Obr. 3.14: Přečtení NFC debetní karty pomocí mobilního telefonu s NFC a programem [34].

### Modifikace dat na NFC zařízení

Modifikace dat na NFC zařízení záleží na typu použitého transpondéru, kde se využívají různé techniky na zabezpečení přístupu a přepisování dat v paměti (secure element). Jak je již zvykem, není nic tak dokonalého, aby to nebylo možné prolomit. Obzvlášť při rozvoji technologie jsou největší problémy se zabezpečením, které se následně upravují.

Jako příklad poslouží Plzeňská MHD NFC karta (ovšem tento druh narušení bezpečnosti je znám po celém světě), která byla dodávána od roku 2004 a disponovala transpondérem založeným na technologii Mifare Classic. Ta byla v dřívějších dobách brána jako standard bezpečnosti bezkontaktních systémů, avšak okolo roku 2008 byly bezpečnostní aspekty prolomeny a tím už úložiště dat nebylo bezpečné. Plzeň však pokračovala dále s distribucí tohoto typu karty. Díky tomu stačí zařízení s podporou NFC a chytrá aplikace na komunikaci s touto Plzeňskou kartou pro modifikaci uložených dat na této kartě. Lze si takto připisovat libovolně kredit na jízdy a číst všechna data z karty. Snadno si lze představit situaci v přeplněné tramvaji, kde útočník použije svůj mobilní telefon s NFC a chytrou aplikací pro napadení karet ostatních lidí. Přitom stačí NFC mobilní telefon „pouze“ přiložit ke kartě a lze tímto způsobem například zvyšovat či snižovat aktuální výši kreditu nebo číst citlivé osobní údaje uložené na kartě [35].

#### Ztráta majetku

I ztráta majetku patří do způsobů narušení bezpečnosti. NFC zařízení většinou obsahují typy autentizace, jako jsou zadávání PIN kódu a odemknutí zamknuté obrazovky. Proto případnému nálezci NFC zařízení bez autentizační metody slouží stejně jako majiteli zařízení. U citlivějších aplikací jako jsou například NFC peněženky si musí aplikace sama zařídit autentizační metodu (náhodné dotázání na heslo). Při ztrátě NFC zařízení v režimu emulace karty (např. pro otevírání dveří) slouží nálezci stejně jako majiteli zařízení. Tato situace je podobná, jako když majitel ztratí klíče o bytu nebo kartu s přístupem [24].

### 3.4.3 Způsoby zabezpečení a doporučení pro bezpečnost

#### Odposlech (Eavesdropping)

NFC nemá zabudovaný žádný mechanizmus proti odposlechu. Jediné reálné řešení proti odposlechu je vytvořit bezpečný kanál, který šifruje danou komunikaci (více v kapitole 3.4.4) [36].

#### Modifikace dat (Data Modification)

Ochrana dat proti modifikaci může být dosažena několika způsoby. První způsob je ve využití lepšího kódování a větší hloubce amplitudové modulace. Například u Millerova kódování lze měnit jen některé bity v posloupnosti a u kódování typu Manchester lze měnit všechny bity v posloupnosti. Proto volba lepšího kódování a větší hloubka amplitudové modulace zmenšuje úspěšnost takovéhoto útoku. Další způsob ochrany je v kontrole radiofrekvenčního pole při odesílání dat. To znamená, že vysílací zařízení může průběžně kontrolovat své vyslané data a pokud zjistí narušení, zastaví přenos dat a uzavře komunikaci. Poslední způsob je opět ve využití zabezpečeného kanálu 3.4.4 [36].

#### Vkládání dat (Data Insertion)

První opatření spočívá v rychlosti odpovědi transpondéru čtecímu zařízení. Díky zkrácené době potřebné na odpověď nestihne útočník vložit svá data do komunikace bez toho, aby nebyly narušeny originálními daty. Další opatření je monitorování elektromagnetických vln v okolí transpondéru. Pokud transpondér zjistí, že někdo začal posílat data místo něj, tak detekuje narušení a odstupuje od komunikace. Další možnost zabezpečení je opět pomocí zabezpečeného kanálu viz. kapitola 3.4.4 [36].

### Přepojovaný útok (Relay Attack)

Jak již bylo psáno v kap. 3.4.1 je prakticky nemožné realizovat přepojovaný útok v radiofrekvenčním pásmu. Přepojovanému útoku realizovanému pomocí notebooku (prostředník komunikace) se lze bránit využitím zabezpečeného kanálu spolu s šifrováním [36].

### Přerušení spojení (Interrupted connection)

Při ochraně proti přerušení spojení stačí zmenšit časovač, který se odpočítává v klidovém stavu na kanálu, aby útočník neměl dostatečný čas na navázání komunikace tam kde přestal uživatel. Další řešení je implementace lepšího algoritmu pro ukončování spojení či občasné dotazování na identifikaci pro ověření uživatele. Poslední způsob zabezpečení je v použití zabezpečeného kanálu popsaného v kap. 3.4.4 [36].

### Opakované přenášení (Replay Attack)

Pro ochranu na tento typ útoku jsou nejvhodnější časová razítka, pořadová čísla datových jednotek či čítač transakcí, který se inkrementuje při každém navazování spojení. Složitější, ale nejvíce zabezpečená možnost ochrany proti opakovanému přenosu je zavedení generování náhodného čísla, které se bude měnit při každém navazování komunikace. Toto číslo je pak využito pro odvození šifrovacího klíče, kterým je zabezpečená celá komunikace [37].

### Získání citlivých dat

Pro ochranu získání nešifrovaných citlivých dat existuje speciální peněženka nebo pouzdro na kartu, které blokují elektromagnetické vlnění v pásmu 13,56 MHz. Tím se znemožní útočníkovi získat data pouhým přiložením čtecího zařízení se speciální aplikací pro čtení údajů z NFC karet. Další možnost je využití novějšího transpondéru pro bezpečné uložení citlivých dat spolu s šifrováním [34].

### Modifikace dat na NFC zařízení

Na ochranu před tímto útokem lze opět použít stínící peněženku či pouzdro. Důmyslnější je však využití zpětné kontroly dat transpondéru (například výši kreditu u MHD karty) skrze čtecí zařízení, které musí být registrované u poskytovatele služby. Poskytovatel si pak může namátkově ověřit jestli aktuální zůstatek na transpondéru koresponduje s aktuálním zůstatkem uloženým v systému. Pokud by se aktuální zůstatky lišili, poskytovatel odhalí manipulaci s daty na kartě a může zakázat kartě přístup do systému spolu s žalobou na majitele karty [34], [35].

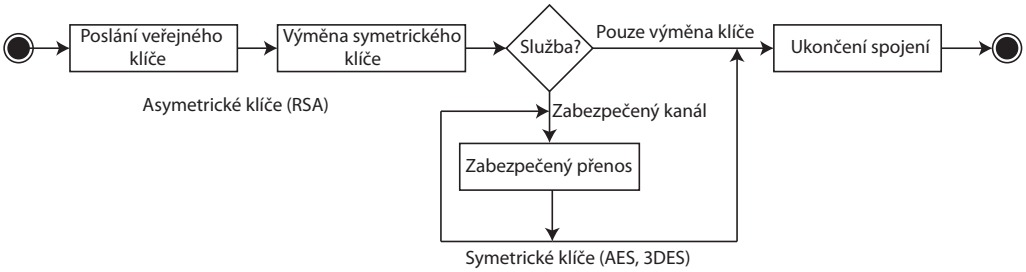
### Ztráta majetku

U tohoto typu lze doporučit používat jen aplikace, které disponují šifrovacími algoritmy spolu se zadáváním silných hesel, které jsou bezpečně uloženy v zařízení. Používání PIN kódu a kódu na odemknutí obrazovky je samozřejmost. U zařízení v režimu emulace karty lze využít přídavné autentizační metody (zadávání hesla nebo biometrických údajů) [24].

## 3.4.4 NFC-SEC: NFCIP-1 Zabezpečené služby a protokol

Standard na zabezpečenou komunikaci pomocí NFC, který je popsán ve standardu ECMA-385 a je navázán na NFCIP-1. Vytvoření zabezpečeného kanálu mezi dvěma NFC zařízeními je jednoznačně nejlepší přístup k ochraně proti většině již popsaných útoků. Standard se skládá ze dvou služeb. První služba je jen na bezpečnou výměnu klíčů pro vlastní potřebu a druhá služba je přímo na sestavení zabezpečeného kanálu. Sestavení zabezpečeného kanálu je rozděleno do několika úrovní viz. obr. 3.15 [38].

Pro výměnu klíčů se využívá standardní protokol o dohodě klíčů jako je algoritmus Diffie-Hellman[9] na základě 192 bitovém asymetrickém klíči RSA[10]. Zahájení přenosu začne posláním veřejného klíče, kterým se zašifruje symetrický klíč (vytvořen z uživatelského hesla). Tento zašifrovaný symetrický klíč se následně pošle zpět odesílateli. Ten díky svému soukromému klíči dešifruje zprávu a tím získá symetrický klíč pro šifrování komunikace. Pro symetrický klíč může být použit algoritmus AES nebo 3DES[11]. [37], [38].
Průběh vytváření zabezpečené komunikace je zobrazen na následujícím obrázku.




Obr. 3.15: Základní kroky pro zabezpečený NFC kanál [38].

[9] Algoritmus na výměnu klíčů, který umožňuje přes nezabezpečený kanál vytvořit mezi komunikujícími stranami šifrované spojení, bez předchozího dohodnutí šifrovacího klíče. Zdroj: <http://cs.wikipedia.org/wiki/Diffie-Hellman>.

[10] RSA – iniciály autorů Rivest, Shamir, Adleman. Asymetrické klíče – soukromý a veřejný. Jedná se o první algoritmus, který je vhodný jak pro podepisování, tak šifrování. Používá se i dnes, přičemž musí mít dostatečnou délku klíče, aby byl považován za bezpečný. Zdroj: <http://cs.wikipedia.org/wiki/RSA>.

[11] Třetí generace šifrovacího algoritmu DES – Data Encryption Standard.

# 4 NÁVRH APLIKACE PRO PŘENOS DAT POMOCÍ NFC

Tato kapitola je věnována návrhu aplikace pro přenos dat pomocí NFC technologie. V přednášce Google I/0 How to NFC [1] zazněl velice dobrý slogan pro NFC, „fyzická akce vyvolá virtuální reakci“. To znamená, že při fyzickém přiblížení zařízení do dosahu NFC se ihned naváže spojení mezi zařízeními a začnou se posílat data (pokud nějaké poslat chtějí). První podpora NFC technologie v OS Android je u verze 2.3 (Gingerbread), čili API level 9, která neměla podporu emulace karty. Od tohoto základu se hodně v OS Android změnilo a v poslední verzi 4.1 (Jelly Bean), API level 16, funguje vše perfektně.

Při programování této aplikace bude s výhodou použita nová funkce OS, Android Beam, která vznikla ve verzi 4.0 (Ice Cream Sandwich), API level 14. Tato funkce usnadňuje programování aplikací založených na přenosové technologii NFC a obsahuje mnoho funkcí pro sdílení obsahu OS Android.

Předpoklad pro přenos dat je zapnutý přístroj a odemčená obrazovka, protože při zamknuté obrazovce (zhasnutý displej) se NFC technologie vypíná. Jedinou výjimkou je režim emulace karty (myšlena přístupová karta, ne bankovní), kde i při vypnutém přístroji NFC stále funguje. To je vcelku logické, protože pokud by byla vybitá baterie a přistroj by již nešel zapnout, nebyla by možnost např. zaplatit účet nebo si otevřít dveře. NFC je velice úsporná technologie, které stačí okolo 7 mA k tomu aby navázala spojení. Díky tomu, že se u režimu emulace karty neposílají velká data, spojení trvá max. pár sekund a tím nespotřebuje velké množství energie.

Návrh aplikace se dělí do dvou směrů. První popisuje logické chování aplikace a druhý se zaměřuje na návrh vzhledu aplikace.

Aplikace bude mít název NFC File Transfer, která pro komunikaci pomocí NFC bude využívat datový formát NDEF a bude komunikovat v režimu P2P. Díky formátu NDEF je zaručena kompatibilita s ostatními zařízeními, které tento standart podporují. Lze díky němu nastavit obsah přenášených dat a tím i přehledně tento obsah filtrovat. Režim P2P implementuje spolehlivou komunikaci mezi zařízeními.

[1] Přednášky od firmy Google přímo pro programátory Androidu. Tuto přednášku uváděl Nick Pelly a Jeff Hamilton, kteří seznamují diváky s průlomovou technologií NFC. Zdroj: <http://www.google.com/events/io/2011/sessions/how-to-nfc.html>.

## 4.1 Popis logického chování aplikace

Tato kapitola se rozděluje na dva stěžejní směry. První popisuje co se všechno musí uskutečnit při odesílání dat druhému zařízení. Potom druhý směr popisuje nutnosti nastavení pro příjem dat od zařízení. Při návrhu byla brána v potaz malá přenosová rychlost NFC, proto pro větší objem přenášených dat se aplikace bude dotazovat uživatele jakou technologii bude chtít využít pro samotný přenos dat. Následující tabulka srovnává použité přenosové technologie.

Tab. 4.1: Srovnání různých technologií na přenos dat [24].

| | NFC | Bluetooth | Wi-Fi Direct |
|---|---|---|---|
| Potřebná doba na sestavení spojení | Žádné prohledávání nebo párování.<br>Menší jak 0,5 sekund | 12 s prohledávání potom nastavení párování.<br>Více jak 30 sekund | Prohledávání, párování, zadávání šifrovacího hesla.<br>Více jak 1 minutu |
| Dosah | 1 – 4 cm | do 15 m | do 150 m |
| Rychlost přenosu dat | 106 – 414 kb/s | 3 – 24 Mb/s | 54 – 108 Mb/s |

NFC samo o sobě není pohodlné na přenos většího objemu dat, protože je poměrně malá rychlost přenosu dat na velmi krátkou vzdálenost. Navíc v celém průběhu výměny dat by musely být přístroje v dosahu NFC. V přednášce Google I/0 How to NFC Nick Pelly uvádí spojení technologií NFC a Bluetooth = Wireless Nirvana, kde se pomocí NFC přístroje spárují a samotný přenos dat probíhá po bluetooth.

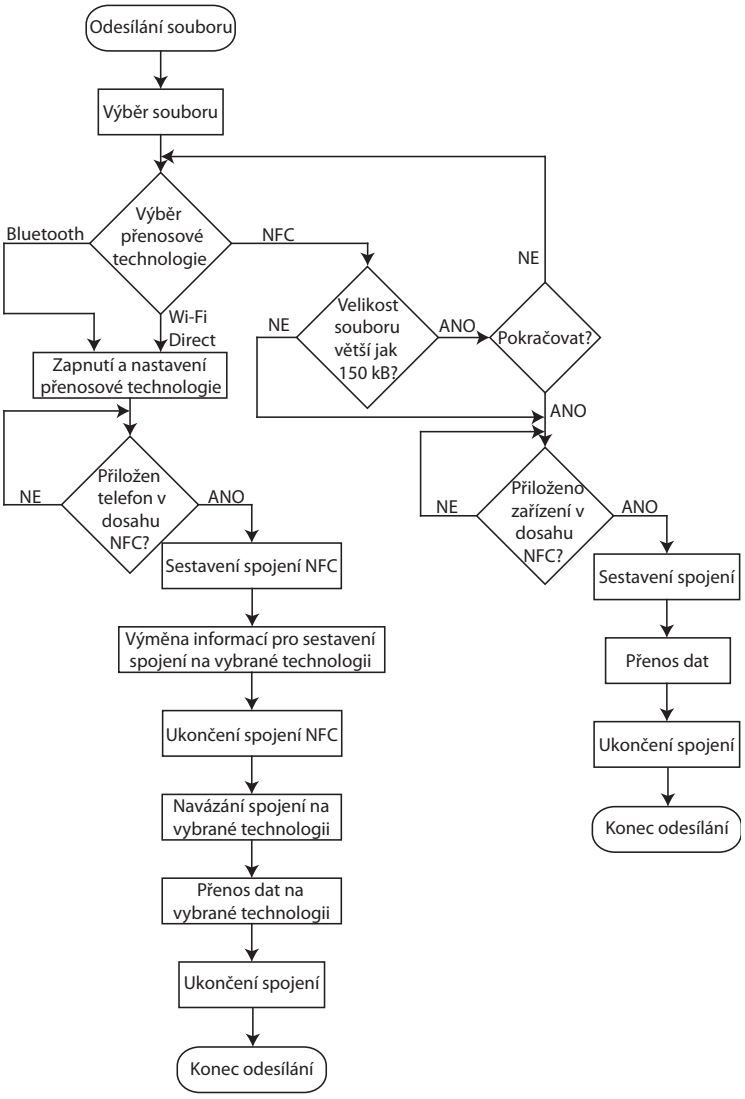
### 4.1.1 Odesílání dat

Na obrázku 4.1 je znázorněn zjednodušený vývojový diagram pro odesílání dat. Digram nezahrnuje žádné chyby v přenosu, vypršení časovačů nebo chyby v párování přístrojů, protože by jeho složitost narostla a stal se tak nečitelný. Dále se předpokládá, že obě zařízení podporují technologie NFC, bluetooth popř. Wi-Fi Direct.

Odesílání souboru začíná výběrem souboru (popř. textu nebo URI), kde se uživateli aplikace nabídne výběr souboru jakéhokoliv typu. Zvolený typ souboru se nastaví do NDEF záznamu pomocí kterého se na straně příjemce budou data identifikovat. Hned po vybrání souboru se bude testovat velikost souboru. Zvolená maximální velikost souboru pro přenos pomocí NFC je nastavena na 150 kB. S ohledem na to, že při menší velikosti je zbytečné sestavovat spojení na jiné technologii, když je soubor odeslán do pár sekund pomocí NFC. Objemnější soubory bude mít uživatel možnost přenášet zvolenou technologií.

Dále se předpokládá, že soubor má menší velikost než 150 kB. Tím se dostáváme do pravé strany vývojového diagramu. Aplikace uživatele upozorní, aby přiložil zařízení do dosahu NFC. Následuje podmínka, která čeká na přiložení telefonu do

dosahu NFC. Pokud telefon není v dosahu NFC jiného zařízení podmínka se neustále opakuje a čeká na přiložení. Při detekci signálu NFC se podmínka vyhodnotí jako pravdivá a začne se sestavovat spojení následované samotným přenosem dat. Po ukončení přenosu dat se spojení ukončí a uživateli se napíše informace o úspěšnosti přenosu souboru.



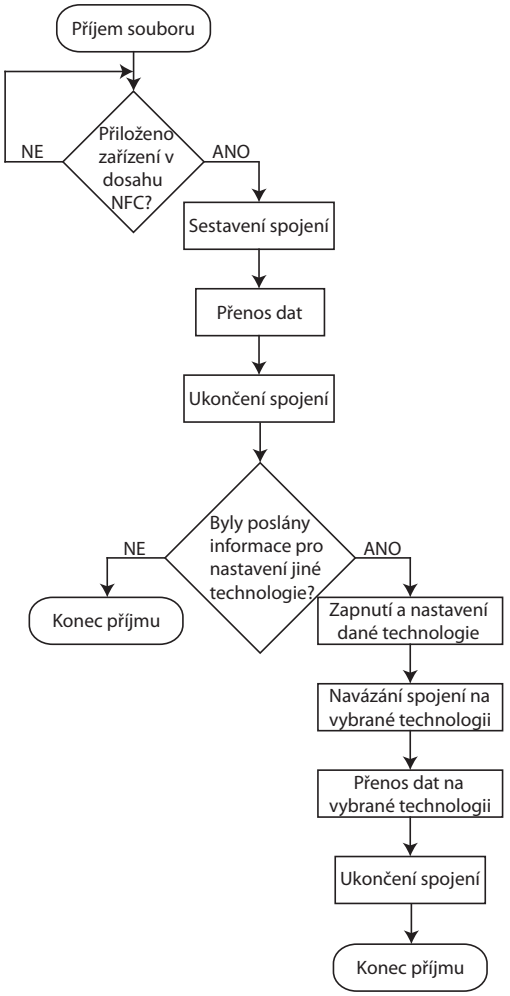

Obr. 4.1: Zjednodušený vývojový diagram pro odesílání dat.

Následuje průchod diagramem při větším souboru jak 150 kB. Aplikace detekuje větší soubor a dá uživateli na výběr přenosovou technologii. Jestliže si uživatel zvolí jako přenosovou technologii NFC i pro objemnější soubor, zobrazí se mu varování, že tento přenos bude trvat delší dobu a po celou dobu přenosu musí být zařízení v dosahu NFC. Pokud si uživatel zvolí pokračování v přenosu, následuje samotný přenos dat pomocí NFC jak už bylo popsáno o odstavec výše.

Pokud si uživatel zvolí jinou přenosovou technologii než-li NFC, vybraná technologie se zapne, nastaví a připraví na přenos dat. Do NDEF záznamu se uloží informace o nastavení přenosové technologie. Následuje opět podmínka na přiložení zařízení do dosahu NFC. Pokud je telefon přiložen v dosahu NFC, sestaví se spojení a proběhne výměna informací o nastavení vybrané přenosové technologie. Díky NFC tak odpadá složitost ručního nastavení přenosové technologie. Následuje ukončení spojení na NFC a sestavování spojení na vybrané technologii. Proběhne výměna dat přes vybranou technologii a spojení se ukončí. Na konec se uživateli napíše informace o úspěšnosti přenosu souboru.

### 4.1.2 Příjem dat

Pro příjem dat je diagram jednodušší. Opět do diagramu nejsou zahrnuté žádné chyby v přenosu, vypršení časovačů nebo chyby v párování přístrojů. Dále se předpokládá, že zařízení podporuje technologie NFC, bluetooth popř. Wi-Fi Direct.




Obr. 4.2: Zjednodušený vývojový diagram pro příjem dat.

Příjem souboru znázorněný na obrázku 4.2 začíná podmínkou jestli je přiloženo zařízení v dosahu NFC. Pokud ne, čeká na přiložení. Pokud ano, sestaví spojení, přenesou se data od odesílatele a ukončí se spojení. Následně si aplikace zkontroluje jaký typ dat byl přenesen v NDEF záznamu. Tím zjistí jestli proběhl přenos vybraného souboru přímo pomocí NFC nebo byly přeneseny pouze informace pro nastavení technologie na přenos konkrétního souboru.

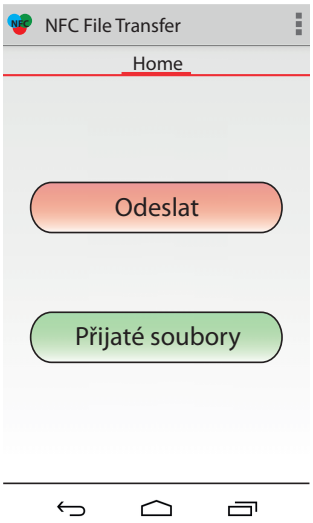
Jestliže NDEF obsahuje pouze informace o nastavení jiné technologie na samotný přenos dat, musí aplikace tuto technologii zapnout, nastavit a připravit na přenos dat. Poté se sestaví spojení na vybrané technologii následujícím samotným přenosem dat. Po přenosu všech dat se spojení na vybrané technologii ukončí, smaže se nastavení a vypne se. Nakonec se uživateli napíše informace o úspěšnosti přenosu dat.

## 4.2 Návrh vzhledu aplikace

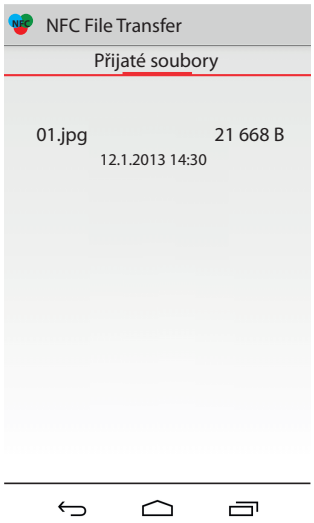
V této kapitole je popsán vzhled a ovládání navržené aplikace. Jelikož jsou displeje u zařízení založeny na černé a tmavých barvách (pro úsporu energie), jsou v následujících obrázcích návrhu aplikace barvy pozměněny tak, aby i po tisku byly obrázky dobře viditelné.

První aktivita (okno po spuštění aplikace), která se uživateli zobrazí je znázorněna na obrázku 4.3. Je na něm vidět název aplikace (NFC File Transfer) a logo (vlevo nahoře). Vpravo nahoře jsou tři čtverečky, které slouží pro nastavení aplikace.

Dále jsou na obrázku znázorněny dvě tlačítka. První tlačítko bude zahajovat výběr dat pro přenos a druhé slouží k zobrazení již úspěšně přenesených souborů. Spodní část obrázku vyobrazuje tlačítka ze zařízení Galaxy Nexus. Obrázek 4.4 znázorňuje vzhled aplikace po stisku tlačítka `Přijaté soubory` z první aktivity.

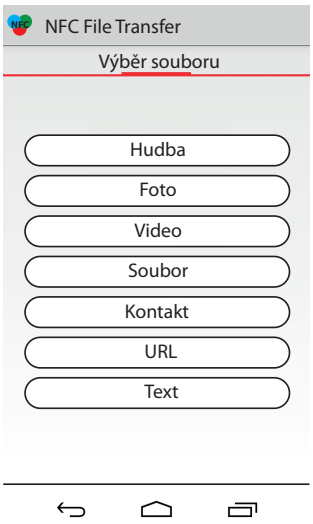



Obr. 4.3: Vzhled první aktivity (první okno po spuštění aplikace).

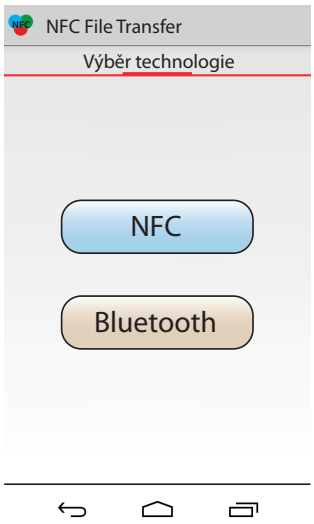



Obr. 4.4: Vzhled aplikace pro přijaté soubory.

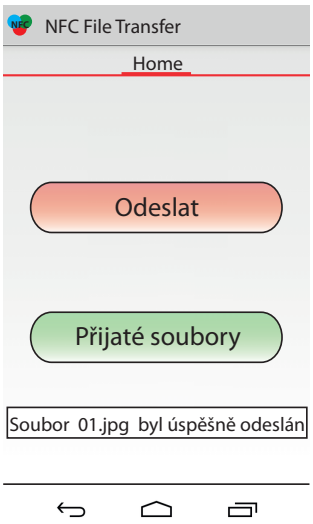
Obrázek 4.5 znázorňuje aktivitu po stisku tlačítka `Odeslat` z první aktivity. Nejdříve se uživateli umožní volba posílaných dat a následně se mu dá na výběr přenosové technologie obr. (4.6). Pro přenos textu, URL a kontaktu je zbytečné sestrojovat spojení na jiné technologii proto se u tohoto typu dat ihned přejde do odesílání pomocí NFC.



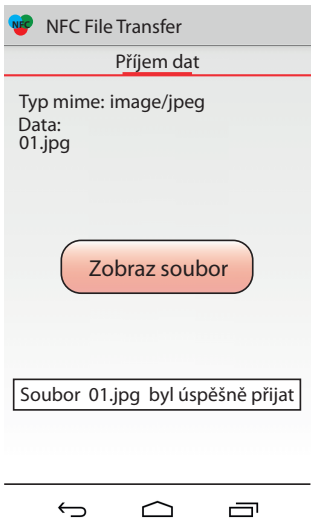

Obr. 4.5: Vzhled aplikace pro výběr souboru.




Obr. 4.6: Vzhled aplikace pro výběr přenosové technologie.




Obr. 4.7: Vzhled aplikace pro potvrzení souboru.




Obr. 4.8: Vzhled aplikace pro potvrzení přijetí souboru.

# 5 APLIKACE NFC FILE TRANSFER

Tato kapitola popisuje vnitřní části nastavení a základní zdrojové kódy aplikace potřebné pro správnou funkci přenosu dat. Jak již bylo popsáno v návrhu, aplikace bude využívat NFC technologii v režimu P2P spolu s datovým formátem NDEF pro přenos jakýkoliv zpráv. Dále bude využita podpora funkce Android Beam. Pro samotný přenos dat se bude aplikace dotazovat uživatele zda-li nechce data přenést pomocí rychlejší technologie (např. Bluetooth).

První a nejdůležitější věc, kterou musí mít každá aplikace psaná pro Android, je správně nastavený soubor Manifest (více o souboru Manifest viz. 1.7.2).
Začátek souboru manifest pro tuto aplikaci vypadá následovně.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android
    "package="cz.vutbr.feec.utko.NFC_FIle_Transfer"
    android:versionCode="1"
    android:versionName="1.18" >
  <uses-sdk
        android:minSdkVersion="16" />
  <uses-feature
        android:name="android.hardware.bluetooth" />
  <uses-feature
        android:name="android.hardware.nfc"
        android:required="true" />
```

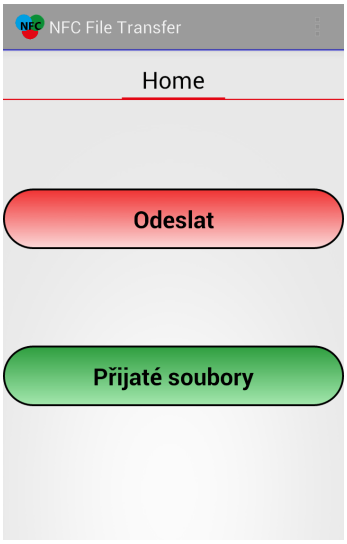
Tento kód definuje minimální spustitelnou verzi OS Android a zároveň verzi OS Android použitou pro překlad do `dex` souboru (ten využívá Dalvik Virtual Machine v OS Android). Oboje je nastaveno jedním řádkem na Jelly Bean (API level 16, verze 4.1). To je kvůli nejnovější podpoře NFC přes funkci Android Beam. Dále je zde vidět definice balíčku ve které je aplikace uložena a označení verze aplikace.

Nejdůležitější jsou však poslední tři řádky, kde aplikace požaduje přítomnost NFC technologie v zařízení. Pokud zařízení nebude disponovat NFC, nepůjde tuto aplikaci nainstalovat a Google Play ji nedá na výběr uživatelům kteří nemají NFC ve svém zařízení. Bluetooth je pouze vyžadován avšak na chod aplikace není nutný.

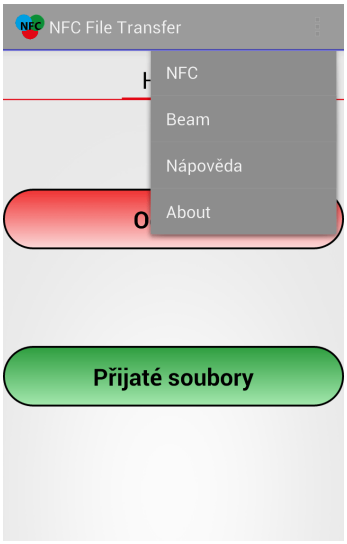
Poté už stačí nastavit potřebná oprávnění pro aplikaci jako je využití funkcí bluetooth spolu s NFC, čtení kontaktů a čtení/zápis do paměti zařízení. Poslední oprávnění znamená, že zařízení „neusne“ během přenosu dat.

```
<uses-permission android:name="android.permission.NFC"/>
<uses-permission android:name="android.permission.BLUETOOTH"/>
<uses-permission android:name="android.permission.READ_CONTACTS"/>
<uses-permission android:name="android.permission.
    READ_EXTERNAL_STORAGE"/>
```

```
<uses-permission android:name="android.permission.
    WRITE_EXTERNAL_STORAGE"/>
<uses-permission android:name="android.permission.WAKE_LOCK"/>
```




Obr. 5.1: Vzhled aplikace první aktivity.




Obr. 5.2: Vzhled menu v první aktivitě.

Obrázek 5.1 ilustruje vzhled první aktivity aplikace NFC File Transfer. Na obrázku 5.2 je vidět uživatelské menu ve kterém je nabídka na zapnutí/vypnutí technologie NFC a funkce Beam. Dále nesmí chybět stručná nápověda k ovládání aplikace a informace o autorovi spolu s licenčními podmínkami užití.

Následující kód kontroluje zapnutí NFC a funkci Android Beam po zmáčknutí tlačítka `ODESLAT`. Pokud nejsou zapnuta, upozorní na to uživatele chybovou hláškou a znemožní mu pokračovat v odesílání. V menu aplikace lze NFC i Android Beam zapnout. Po zapnutí se opět uživateli umožní pokračování v odesílání dat.

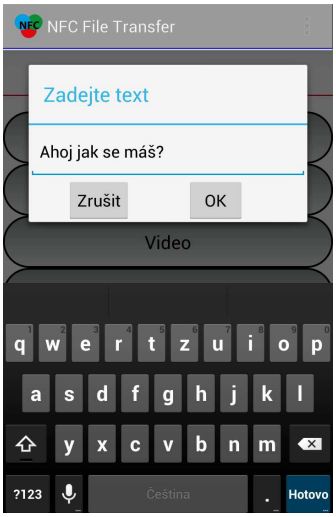
```
boolean test = true;
mNfcAdapter = NfcAdapter.getDefaultAdapter(this);
if (!mNfcAdapter.isEnabled()) {
// Pokud NFC adaptér není zapnut  vypíše zprávu
 Toast.makeText(this,"NFC není zapnuté!",Toast.LENGTH_LONG).show();
 test = false;
}else if (!mNfcAdapter.isNdefPushEnabled()) {
 // Pokud není zapnuta funkce Android Beam
 Toast.makeText(this,"Není zapnut Beam",Toast.LENGTH_LONG).show();
 test = false;
}
return test;
```

## 5.1 Získání dat pro odeslání

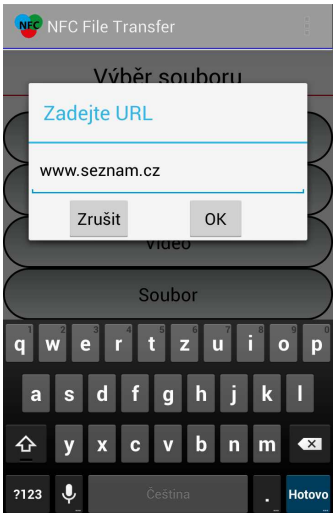
Tato kapitola stručně popisuje jak lze získat data ze zařízení a nabídnout je uživateli pro přenos na jiné zařízení. Jak je vidět na obrázku 5.3 je uživateli nabídnuta široká škála dat k posílání. Každý formát posílaných dat má mírně odlišné zacházení při přenosu i při zpracování na přijímací straně.


Obr. 5.3: Vzhled aplikace pro výběr dat.

Nejdříve zde bude pro jednoduchost probráno získávání textu a URL adresy. Text musí zadat uživatel do dialogu (obr. 5.4), který se objeví po stisku tlačítka `Text`. Tento text je pak přenesen a zobrazen na druhém zařízení. U formátu dat `URL` je pak možné na druhém zařízení otevřít URL adresu přímo v prohlížeči.




Obr. 5.4: Vzhled aplikace pro získání textu.




Obr. 5.5: Vzhled aplikace pro získání URL adresy.

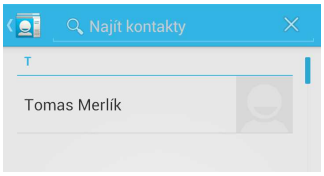
Kód pro zobrazení dialogu na zadání textu a URL vypadá následovně:

```
final Dialog dialog = new Dialog(Context);
// Jak má dialog vypadat - rozložení
dialog.setContentView(R.layout.activity_dialog_write);
// Nadpis dialogu
dialog.setTitle("Zadejte text");
EditText editText = (EditText) dialog.findViewById(R.id.editText);
// Zobrazení dialogu
dialog.show();
```

Po stisku tlačítka `OK` z dialogu vypadá kód následovně.

```
// Vytvoření nového intentu
Intent intent = new Intent(Context, Odesilani.class);
// Skrytí dialogu
dialog.dismiss();
// Přidání dat do intentu
intent.putExtra("format_zpravy", format);
intent.putExtra("data_zpravy", editText.getText().toString());
// Spuštění aktivity daným intentem
startActivity(intent);
```

Nejdříve se vytvoří intent do kterého se následně funkcí `putExtra` přidají data, která se u příjemce intentu zpracují. Datová část `format_zpravy` v intentu označuje číselný formát dat, kterým následně při přípravě k odesílání aplikace detekuje různé typy dat a může tím nastavit patřičnou hodnotu v NDEF zprávě. Datová část `data_zpravy` označuje konkrétně zadaný text nebo URL v intentu. Nakonec se spustí nová aktivita `Odesilani` a začnou se zpracovávat data pro odesílání.




Obr. 5.6: Vzhled aplikace pro získání kontaktu.

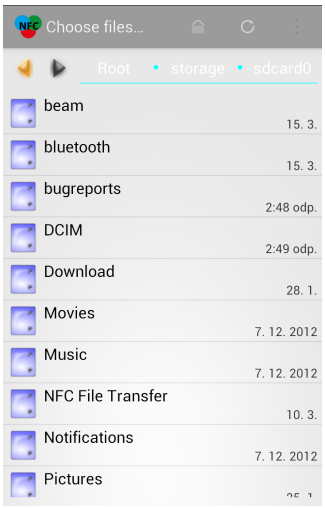
Dalším typem posílaných dat je kontakt. Jeho získání je poněkud složitější, protože je uložen v databázi SQLite. Pro jeho získání je nutno použít poskytovatele obsahu (Content Providers) a ukazatel na záznam v databázi. Poté se vytvoří soubor Vcard ve kterém jsou uloženy informace o kontaktu. Ten se načte do pomocné proměnné a převede se na textový výstup který se pošle druhému zařízení (kód viz. A.3). Formáty Vcard a xVcard jsou přesně definovány standardem pro uložení hodnot kontaktu kvůli jeho přenositelnosti na různé zařízení a operační systémy.

Následujícím kódem lze vyvolat nabídku kontaktů z obrázku 5.6.

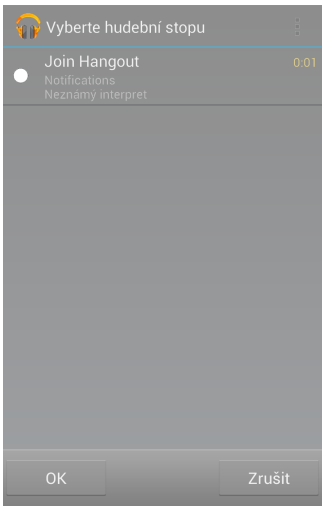
```
// Vytvoření intentu a nastavení, že se jedná o výběr kontaktu
Intent contactPickerIntent = new Intent(Intent.ACTION_PICK,
    Contacts.CONTENT_URI);
// Spustí novou aktivitu a čeká na návratovou hodnotu
startActivityForResult(contactPickerIntent, CONTACT_PICKER_RESULT);
```

Návratové hodnoty všech aktivit spuštěných jako `startActivityForResult` vracejí hodnotu do funkce jménem `onActivityResult`. Ta přijímá data od aktivity, která ji vyvolala. Proměnná `resultCode` znamená úspěšnost operace, `requestCode` je celočíselné rozlišení aktivity (např. 101). Toto číslo se nastavuje při spouštění a při vyhodnocení se dělí pomocí funkce `switch` do svých částí kódu.

```
protected void onActivityResult(int requestCode, int resultCode,
    Intent data) {
     if (resultCode == RESULT_OK) {
       switch (requestCode) {
         case CONTACT_PICKER_RESULT: break;
         case _ReqChooseFile: break;
         case RESULT_LOAD_IMAGE_VIDEO_SOUND: break;
}}}
```




Obr. 5.7: Vzhled aplikace pro výběr souboru.




Obr. 5.8: Vzhled aplikace pro výběr hudby.

Další data pro posílání jsou soubory. Zde může být vybraný jakýkoliv soubor ze zařízení. Pro tento případ byl zvolen volně dostupný prohlížeč souborů android-filechooser od tvůrce Hai Bison[1] který je vidět na obrázku 5.7.

[1]Knihovna je open source a lze ji zdarma stáhnout na adrese: <`http://code.google.com/p/android-filechooser/`>

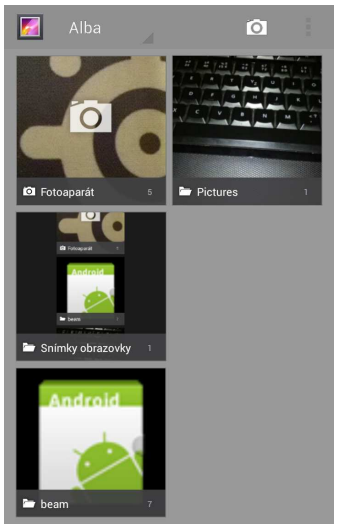
Následujícím kódem lze vyvolat nabídku na prohlížení souborů (File chooser). Ten opět vrací návratovou hodnotu do funkce `onActivityResult` jako odkaz na soubor.

```
Intent intent = new Intent(Context, FileChooserActivity.class);
startActivityForResult(intent, _ReqChooseFile);
```

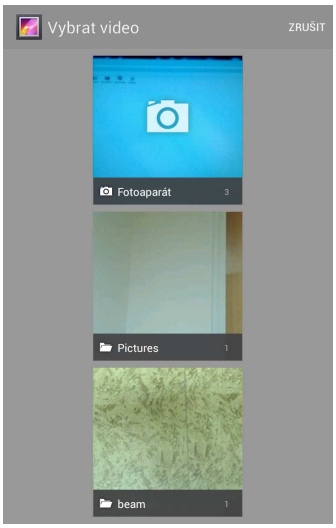
Dále je zde uvedeno vytvoření intentu na výběr technologie[2], přidání extra dat do intentu a spuštění aktivity.

```
Intent intent = new Intent(Context, Vyber_technologie.class);
intent.putExtra("format_zpravy", 3); // format = 3 značí soubor
intent.putExtra("data_zpravy", cesta);
intent.putExtra("jmeno_souboru", jmenoSouboru);
startActivity(intent);
```

Aktivita na výběr hudby je znázorněna na obrázku 5.8. Na následujících obrázcích je vidět vzhled aplikace pro výběr fotek a videa.




Obr. 5.9: Vzhled aplikace pro výběr fotek.




Obr. 5.10: Vzhled aplikace pro výběr videa.

Hudba, foto a video mají velice podobné prohlížeče a spouštějí se následujícím kódem. Po vybrání konkrétního souboru se musejí načíst informace z SQLite databáze, aby se získal odkaz na konkrétní soubor.

```
Intent photoPickerIntent = new Intent(Intent.ACTION_PICK);
photoPickerIntent.setType("image/*"); // ("video/*"), ("audio/*")
startActivityForResult(photoPickerIntent,
    RESULT_LOAD_IMAGE_VIDEO_AUDIO);
```

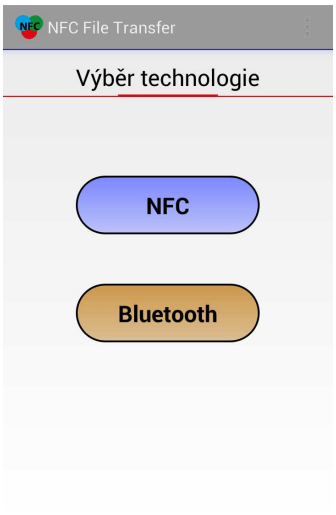
[2]Jedná se o soubor různé velikosti proto se dá uživateli na výběr technologie přenosu. Text, URL a kontakt se ihned posílá pomocí NFC.

V příloze A.4 je ukázka jak lze získat cestu k souboru z databáze SQLlite pomocí poskytovatele obsahu.

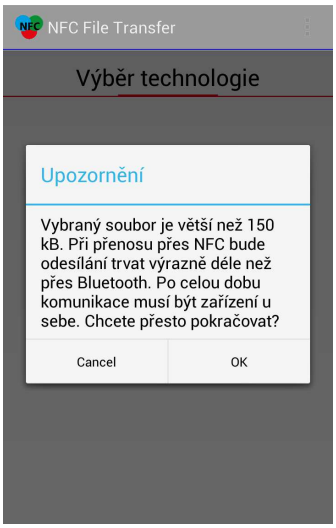
## 5.2 Odesílání dat

Předchozí kapitola popisovala jak data vybrat resp. jak získat konkrétní odkaz na soubor ze zařízení. Tato kapitola se zabývá odesíláním těchto vybraných dat. Nejdříve bude čtenář seznámen s odesíláním pomocí NFC a na konec je tu ukázáno jak data posílat pomocí Bluetooth při párování přístrojů pomocí NFC.

Na obrázku 5.11 je vidět vzhled aplikace na výběr technologie. K dispozici pro přenos dat je NFC a Bluetooth. U NFC se posílají data ihned, avšak je zde podmínka pro maximální velikost souboru nastavena na 150 kB. Tato velikost byla vybrána jako kompromis mezi dobou posílání souboru a chybovosti na NFC spoji [3]. Pokud ovšem soubor má více než zmíněných 150 kB aplikace na tento fakt uživatele upozorní varovným dialogem (viz. obr. 5.12). Uživatel si potom může vybrat jestli opravdu chce posílat vybraný soubor větší než 150 kB pomocí NFC.




Obr. 5.11: Vzhled aplikace pro výběr technologie.




Obr. 5.12: Vzhled aplikace pro upozornění na odesílání velkého souboru pomocí NFC.

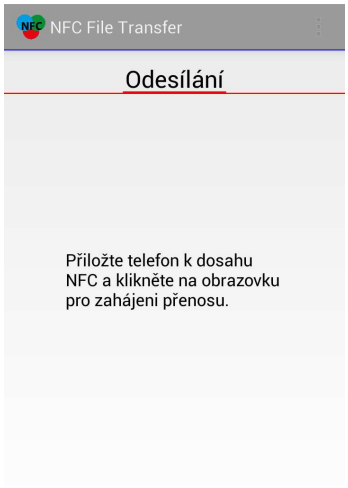
Při výběru technologie Bluetooth se automaticky zapne na zařízení (pokud je k dispozici, pokud ne, vypíše uživateli chybnou hlášku) a díky NFC se přístroje

[3] Chybovost se rapidně zvětšuje se vzdáleností NFC antén a s přítomností rušivého signálu. Navíc musí být NFC antény přiloženy přímo na sebe, protože jejich vyzařovací charakteristika je kolmá k anténě s malým vyzařovacím úhlem.

spárují. Potom veškerý přenos dat probíhá pomocí Bluetooth. Po ukončení přenosu se automaticky Bluetooth na obou zařízeních vypne.

Aktivita odesílání (obr. 5.13) začíná vytvořením třídy `Odesilani` s implementací podpůrných tříd na kontrolu a informování o průběhu přenosu dat pomocí NFC.

```
public class Odesilani extends Activity implements
    CreateNdefMessageCallback, OnNdefPushCompleteCallback {
  NfcAdapter mNfcAdapter = NfcAdapter.getDefaultAdapter(this);
  BluetoothAdapter mBluetoothAdapter = BluetoothAdapter.
     getDefaultAdapter();
  // Informuje aplikaci o průběhu přenosu
  mNfcAdapter.setNdefPushMessageCallback(this, this);
  // Informuje aplikaci o výsledku přenosu
  mNfcAdapter.setOnNdefPushCompleteCallback(this, this);
```




Obr. 5.13: Vzhled aplikace pro odesílání dat.

Následně se musí z intentu, který zavolal tuto aktivitu, vybrat data potřebná pro přenos, jako jsou: formát zprávy, data zprávy, jméno souboru, výběr technologie. Díky těmto datům lze vytvářet individuální NDEF zprávy a určit tak způsob zacházení.

```
Intent intent = getIntent();
fomrat = intent.getIntExtra("format_zpravy", 0);
message = intent.getStringExtra("data_zpravy");
jmenoSouboru = intent.getStringExtra("jmeno_souboru");
vyber_technologie = intent.getIntExtra("vyber_technologie", 0);
```

Pro samotné posílání dat pomocí NFC slouží následující kód.

```
public NdefMessage createNdefMessage(NfcEvent event) {
  switch (fomrat) {
    case 1: // "text/plain"
      NdefMessage msg_text = new NdefMessage(NdefRecord.createMime(
          "text/plain", message.getBytes()));
    return msg_text;
    case 2: // "text/uri"
      NdefMessage msg_uri = new NdefMessage(NdefRecord.createMime(
          "text/uri", message.getBytes()));
    return msg_uri;
    //case 3:  soubor
    //case 4:  kontakt
  default: break;
    }
return null;
}
```

Díky příjmu události `NfcEvent event` se tato funkce vyvolá pokud funkce Android Beam detekuje v dosahu jiné NFC kompatibilní zařízení. Potom již stačí potvrdit odeslání vybrané zprávy a data se začnou posílat.

Aplikace rozlišuje vytváření NFC zprávy pomocí funkce `switch()` a proměnné `format`. V této proměnné se ukládá co se posílá za data přes NFC, aby je mohla aplikace při příjmu třídit a správně vyhodnotit. Proměnná `message` obsahuje data v textové podobě (u přenosu souboru tato proměnná přenáší cestu k souboru). Pomocí funkce `NdefRecord.createMime()` se vytvoří daný NDEF záznam a funkcí `NdefMessage()` se vytvoří NDEF zpráva z jichž vytvořených NDEF záznamů.

Ukázka posílání souboru je v následujícím kódu.

```
// Pokračování SWITCH z funkce createNdefMessage
case 3: // Soubor
  // Otevření souboru a přiřazení do InputStream
  InputStream is = new BufferedInputStream(new FileInputStream(
     message));
  // Vytvoření vyrovnávací paměti
  ByteArrayOutputStream bos = new ByteArrayOutputStream();
  // Načítání souboru do vyrovnávací paměti
  while (is.available() > 0) { bos.write(is.read()); }
  // Vytvoření dvou NDEF záznamů
  NdefRecord[] nDEFz = new NdefRecord[2];
  // První záznam nese jméno souboru
  nDEFz[0]=NdefRecord.createMime("file/*",jmenoSouboru.getBytes());
  // Druhý nese samotná data souboru
  nDEFz[1] =NdefRecord.createMime("file/*",bos.toByteArray());
```

```
  // Vytvoření NDEF zprávy
  NdefMessage msg_soubor = new NdefMessage(nDEFz);
  is.close();  // Uzavření souboru
  bos.close(); // Uzavření vyrovnávací paměti
return msg_soubor;
```

V proměnné `message` je uložena cesta k souboru. Tato část kódu není opatřena zachytáváním chybových stavů `try` a `catch` kvůli přehlednosti v kódu. Díky konstruktoru `FileInputStream()` se soubor otevře a může se načíst do vyrovnávací paměti jeho obsah. Posléze se vytvoří dva NDEF záznamy, kde první ponese jméno souboru a druhý data souboru. Díky nastavení formátu `file/*` aplikace při příjmu pozná, že se jedná o soubor a může ho začít ukládat. Lze uvést libovolný typ NDEF záznamu, ale je nutné nastavit stejný typ při příjmu zprávy u intent filtrů (bude probráno u příjmu dat).
Pro posílání kontaktu je kód následující:

```
// Pokračování SWITCH z funkce createNdefMessage
case 4: // kontakt
  NdefMessage msg_kontakt = new NdefMessage(NdefRecord.createMime("
      text/x-vcard", message.getBytes()));
return msg_kontakt;
```

Zde se data posílají v textové podobě formátu `text/x-vcard`. To je standard na přenositelnost kontaktů. Z telefonu se načtou informace o vybraném kontaktu do textového řetězce `message`. Ten se převede na byty funkcí `getBytes()` a odešle se na druhé zařízení. Operační systém Android tuto zprávu zachytí a díky nastavenému typu `text/x-vcard` ihned přidá přijatý kontakt do seznamu kontaktů. Proto není nutné v aplikaci tento typ dat nijak přijímat.

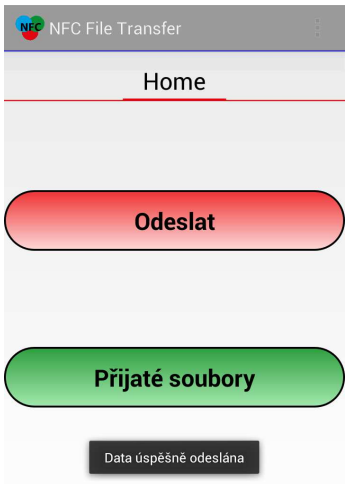
Následující funkce se vyvolá pokud je NDEF zpráva vytvořena a je připravena k odeslání. Díky této funkci je operační systém informován o průběhu přenosu a může řídit i datový tok.

```
public void onNdefPushComplete(NfcEvent arg0) {
    mHandler.obtainMessage(MESSAGE_SENT).sendToTarget();
  }
```

Funkce `mHandler` se vyvolá když je NDEF zpráva úspěšně odeslána na druhé zařízení a aplikace informuje uživatele o úspěšnosti přenosu dat a nastaví aplikaci do úvodní obrazovky (obr. ).

```
private final Handler mHandler = new Handler() {
  public void handleMessage(Message msg) {
    switch (msg.what) {
        case MESSAGE_SENT:
```

```
        // vypíše informaci o úspěšnosti přenosu
            Toast.makeText(getApplicationContext(), "Data úspěšně
                odeslána", Toast.LENGTH_LONG).show();
            Intent intent = new Intent(Odesilani.this, Uvodni.class
                );
            startActivity(intent);
            break;
    }
}};
```




Obr. 5.14: Vzhled aplikace pro úspěšné odeslání dat.

Poslední ukázka poslání souboru je posílání přes Bluetooth. Díky funkci Android Beam, která je ve verzi operačního systému Android 4.1 (API level 16) značně vylepšena, je relativně jednoduché posílat soubory pomocí Bluetooth při párování přes NFC. Následujících pár řádků tuto jednoduchost dokazují.

```
private void odeslatBT() {
File myFile = new File (message);
    mNfcPushUris[0] = Uri.fromFile(myFile);
    mNfcAdapter.setBeamPushUris(mNfcPushUris, this);
}
```

V proměnné `message` je uložena cesta k souboru[4]. Přes NFC se pošlou jen párovací informace a samotná data se začnou posílat po Bluetooth, který se automaticky zapne a po dokončení přenosu vypne. Díky funkci Android Beam není nutno zprávu v aplikaci nijak přijímat a vše si operační systém Android zařídí sám.

[4]Je možno posílat i více souborů najednou, ale tato aplikace podporuje jen jeden soubor na jeden přenos.

## 5.3 Příjem dat

Tato kapitola popisuje jak se v aplikaci detekuje příjem dat a jak se přijatá data uloží popř. zobrazí na druhém zařízení.

Nedílnou součástí je správné nastavení intent filtrů. Intent byl v této aplikaci zatím použit pouze pro lokální vytváření aktivit na jednom zařízení. Intent lze posílat i mezi zařízeními, kde nejdříve tento intent dostane operační systém. Ten potom rozhoduje, podle intent filtru nastaveného v manifestu aplikací, která aplikace (aktivita) bude spuštěna. Zde je příklad nastavení intent filtrů pro příjem NDEF zpráv u aktivity `Prijem`. Je nutné nastavit `mimeType` na typy, které se používají při vytvoření NDEF záznamů a zprávy v aktivitě `Odesilani`.

```
<intent-filter>
  <action android:name="android.nfc.action.NDEF_DISCOVERED" />
  <category android:name="android.intent.category.DEFAULT" />
     <data android:mimeType="text/plain" />
</intent-filter>
<intent-filter>
  <action android:name="android.nfc.action.NDEF_DISCOVERED" />
  <category android:name="android.intent.category.DEFAULT" />
     <data android:mimeType="text/uri" />
</intent-filter>
<intent-filter>
  <action android:name="android.nfc.action.NDEF_DISCOVERED" />
  <category android:name="android.intent.category.DEFAULT" />
     <data android:mimeType="file/*" />
</intent-filter>
```

Předchozí kód informuje operační systém o skutečnosti, že tato aplikace je schopna přijímat NDEF záznamy s nastavenou hodnotou mime type uvedenou v manifestu. Pokud operační systém přijme takto nastavený intent, načte intent filtry všech aplikací (má je uloženy v jednom souboru) a rozhodne která aplikace (popř. více aplikací - dá uživateli na výběr) je schopna tuto událost obsloužit. Po vybrání konkrétní aplikace spustí novou aktivitu a předá ji intent. Aktivita intent přijme, vyhodnotí a začne zpracovávat. V tomto případě je v intentu uložena i NDEF zpráva.

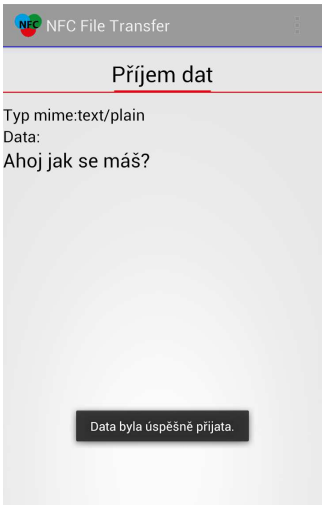
V aktivitě `Prijem` je nutno mít funkci `ukazPrijataData` (na jménu funkce nezáleží, ale musí přijímat zprávy intent). Ta přijímá zprávy intent, které prošly definovaným intent filtrem v manifestu.

```
void ukazPrijataData(Intent intent) {
  // načtení intent zprávy
  Parcelable[] rawMsgs = intent.getParcelableArrayExtra(NfcAdapter.
     EXTRA_NDEF_MESSAGES);
```

```
  // uložení NDEF zprávy do proměnné msg
  NdefMessage msg = (NdefMessage) rawMsgs[0];
}
```

Pomocí funkce `if` lze vytřídit různé typy NDEF zpráv a díky tomu jim lze nastavit různé zacházení. Zpracování NDEF zprávy nastavené na `text/plain` je následující. Nejdříve nastaví do `mimeType` (objekt `TextView`) typ mime nastavený v NDEF záznamu a do `mdata` (objekt `TextView`) nastaví datový obsah NDEF záznamu převedený na textový řetězec. Potom aplikace informuje uživatele o úspěšnosti přenosu.

```
if (msg.getRecords()[0].toMimeType().equals("text/plain")) {
      mimeType.setText(msg.getRecords()[0].toMimeType());
      mdata.setText(new String(msg.getRecords()[0].getPayload()));
      Toast.makeText(this, R.string.text_DataTextURL_REC_OK,
          Toast.LENGTH_LONG).show();
}
```




Obr. 5.15: Vzhled aplikace pro příjem textu.


Obr. 5.16: Vzhled aplikace pro příjem URL

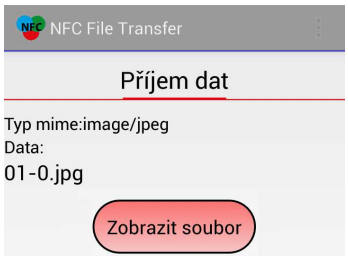
Pro příjem dat typu `text/uri` je příjem stejný jako u typu `text/plain` a navíc je zde možnost zobrazit URL v prohlížeči tlačítkem `Zobraz URL` (obr. 5.16). Kód na zobrazení URL v prohlížeči po stisku tlačítka `Zobraz URL` je následující.

```
// Z textView si vezme URL adresu
String url = mdata.getText().toString();
// Pokud URL nezačíná http:// nebo https:// automaticky se přidá
if (!url.startsWith("https://") && !url.startsWith("http://"))
  {url = "http://" + url;}
Intent myIntentURL = new Intent(Intent.ACTION_VIEW,Uri.parse(url));
startActivity(myIntentURL);
```

Pro přijatý soubor je situace o něco složitější. Ukázka příjmu souboru vypadá následovně.

```
else if (msg.getRecords()[0].toMimeType().equals("soubor/data")) {
  File mediaDir = new File("/sdcard/NFC File Transfer/");
  // Vytvoření složky pro soubor pokud neexistuje
      if (!mediaDir.exists()) {
        mediaDir.mkdir();
      }
  // Jméno souboru je v prvním NDEF záznamu
  String jmeno = new String(msg.getRecords()[0].getPayload());
  // Uložení těla zprávy
  byte[] teloZpravy = msg.getRecords()[1].getPayload();
  File resolveMeSDCard = new File("/sdcard/NFC File Transfer/" +
      jmeno);
    if (!resolveMeSDCard.exists()) {
     // Vytvoření souboru pokud neexistuje
     resolveMeSDCard.createNewFile();
    }
    // Pokud soubor existuje, přepíše ho
    FileOutputStream fos = new FileOutputStream(resolveMeSDCard);
    // Zápis dat do souboru
    fos.write(teloZpravy);
    fos.close();
    // Výpis informace o úspěšném přijetí souboru
    Toast.makeText(this, "Soubor " + jmeno + " byl úspěšně přijat",
        Toast.LENGTH_LONG).show();
```

Hned po příjmu souboru se aplikace snaží vytvořit složku pro ukládání takto přijatých souborů. Pokud neexistuje, vytvoří ji, pokud existuje, nechá ji a pokračuje. Potom si zjistí jméno poslaného souboru, které je uložené v prvním záznamu NDEF zprávy. Následuje vytvoření souboru v paměti mobilního zařízení. Pokud již soubor existoval, přepíše jej. Potom se můžou do otevřeného souboru začít zapisovat přijatá data v bytech. Soubor se uzavře a aplikace zobrazí zprávu o úspěšnosti uložení souboru. Tlačítkem `Zobraz soubor` lze ihned přijatý soubor otevřít.




Obr. 5.17: Vzhled aplikace – potvrzení přijetí souboru.

# 6 ZÁVĚR

V první části této diplomové práce byl popsán operační systém Android. Jelikož poměrně málo lidí zná jeho historii, tak zde byla sepsána od vzniku do dnešní doby. Dále byl čtenář seznámen s verzemi tohoto operačního systému, s kódovým označením a s číslováním API. Následoval stručný popis procesorů s architekturou ARM, na kterých běží OS Android. S tím souvisel popis projektu Android-x86, který se snaží prosadit OS Android pro procesory s architekturou x86, zejména pro Intel Atom. V architektuře OS Android byly vysvětleny rozdíly oproti ostatním OS a funkce, které OS Android podporuje. Poslední část této kapitoly byla věnovaná popisu zabezpečení celého OS od Linuxového jádra po samotné zabezpečení aplikací.

Druhá kapitola byla věnována popisu technologie RFID, na kterou navazuje technologie NFC. U RFID bylo uvedeno porovnání používaných technologií pro logistiku a identifikaci uživatele. Dále byly vysvětleny pojmy transpondér a čtecí zařízení. V neposlední řadě byl čtenář obeznámen s principem komunikace bezdrátového bezkontaktního systému RFID.

V další části bylo popsáno stěžejní téma této diplomové práce, technologie NFC. NFC je poměrně mladá technologie, která má do budoucna velký potenciál v komunikaci, bankovnictví a výměně dat. Byly zde uvedeny standardy a technické specifikace, kde je uvedena funkce fyzické a spojové vrstvy NFC. Podporované režimy přenosu dat pomocí NFC jsou čtení/zápis, emulace karty a rovný s rovným (P2P či Peer-to-Peer). NFC může být do zařízení implementováno třemi způsoby. Prvním způsobem je NFC čip a anténa zabudovaná přímo v telefonu. Další způsob spočívá v existenci NFC čipu a antény na SD kartě. Ve třetím způsobu je NFC čip obsažen na SIM kartě a anténa může být interní nebo externí. Další část tohoto článku byla věnována útokům na zabezpečení při používání technologie NFC a ochrana proti nim. NFC nedisponuje šifrovacím mechanizmem, a tak je nutné, aby bylo implementováno do aplikací vyšších vrstev. S rozvojem této technologie se realizují nejrůznější útoky, a tím se postupně vylepšují standardy pro komunikaci a šifrování spolu s bezpečným úložištěm citlivých dat. Pro nejlepší zabezpečení je dobré použít funkci NFC-SEC, která navazuje na standardy NFC, pro vytvoření zabezpečeného kanálu pro přenos dat.

Předposlední kapitola popisuje návrh aplikace pro přenos dat pomocí NFC. NFC je relativně pomalá technologie na přenos objemnějších dat na velmi krátkou vzdálenost. Proto budou využity rychlejší technologie na samotný přenos dat a pomocí NFC se tyto rychlejší technologie pouze nastaví pro přenos. Méně objemnější data budou přenesena přímo po NFC.

Závěrečná část této diplomové práce popisuje vytvořenou aplikaci NFC File Transfer pro přenos dat mezi zařízeními pomocí NFC.

# LITERATURA


[1] *Android version history. In: Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001–2012, poslední aktualizace 16. 9. 2012 [cit. 20. 9. 2012]. Dostupné z URL: <http://en.wikipedia.org/wiki/Android_version_history>.

[2] *Android (operating system). In: Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001–2012, poslední aktualizace 20. 9. 2012 [cit. 20. 9. 2012]. Dostupné z URL: <http://en.wikipedia.org/wiki/Android_%28operating_system%29>.

[3] *Android (operační systém). In: Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001–2012, poslední aktualizace 20. 9. 2012 [cit. 20. 9. 2012]. Dostupné z URL: <http://cs.wikipedia.org/wiki/Android_%28opera%C4%8Dn%C3%AD_syst%C3%A9m%29>.

[4] *Open Handset Alliance: Android Developer.* [online]. 2012, [cit. 20. 9. 2012]. Dostupné z URL: <http://developer.android.com>.

[5] *Open Handset Alliance: Android Open Source Project.* [online]. 2012, [cit. 20. 10. 2012]. Dostupné z URL: <http://source.android.com>.

[6] *ARM The Architecture for the Digital World* [online]. 4. 5. 2010, 2010–2012 ARM Limited [cit. 5. 10. 2012]. Dostupné z URL: <http://www.arm.com>.

[7] YI Sun *Android-x86 - Porting Android to x86* [online]. 15. 7. 2012, 2009–2012 [cit. 5. 10. 2012]. Dostupné z URL: <http://www.android-x86.org/>.

[8] BRADY, P. *Google I/O 2008 - Anatomy and Physiology of an Android* [online]. 28. 5. 2008, Google I/O 2008–2012 [cit. 7. 10. 2012]. Dostupné z URL: <https://sites.google.com/site/io/anatomy--physiology-of-an-android>.

[9] HOUŠKA P, *První telefon s procesorem od Intelu pro Evropu* [online]. 27. 2. 2012, [cit. 5. 10. 2012]. Dostupné z URL: <www.androidmarket.cz/mobilni-telefony/orange-oficialne>.

[10] GUPTA, V. *Mobile Platforms - Part 1 - Android* [online]. 7. 3. 2011, 2012 Vineet Gupta [cit. 10. 10. 2012]. Dostupné z URL: <http://www.vineetgupta.com/2011/03/mobile-platforms-part-1-android/>.

[11] KLAPETEK, M. *Android slaví 3. narozeniny. Svět Androida.* [online]. 2010, poslední aktualizace 6. 11. 2010 [cit. 20. 9. 2012]. Dostupné z URL: <http://www.svetandroida.cz/android-slavi-3-narozeniny-201011>.

[12] NEPŠINSKÝ, M. *Google App Inventor – snadný vývoj aplikací pro Android.* [online]. 2010, poslední aktualizace 24. 10. 2010 [cit. 21. 9. 2012]. Dostupné z URL: <http://www.svetandroida.cz/google-app-inventor-snadny-vyvoj-aplikaci-pro-android-201010>.

[13] KYPTA, T. *Vyvíjíme pro Android – úvod* [online]. 2011, poslední aktualizace 23. 3. 2011 [cit. 10. 10. 2012]. Dostupné z URL: <http://www.svetandroida.cz/vyvijime-pro-android-1-uvod-201103>.

[14] YADAV, M. *History of Android. In: Tech2crack - Android* [online]. 2011, poslední aktualizace 30. 12. 2011 [cit. 20. 9. 2012]. Dostupné z URL: <http://www.tech2crack.com/history-android/>.

[15] HAVRYLUK, M. *Oracle útočí na podstatu Androidu. Google má několik linií obrany.* [online]. Mobil.cz 2011, poslední aktualizace 22. 8. 2011 [cit. 21. 9. 2012]. Dostupné z URL: <mobil.idnes.cz/mob_tech.aspx?c=A110802_162440_mob_tech_ham>.

[16] KRČMÁŘ, P. *Microsoft vybírá za patenty k Androidu.* [online]. 2011, poslední aktualizace 7. 7. 2011 [cit. 21. 9. 2012]. Dostupné z URL: <http://www.root.cz/clanky/microsoft-vybira-za-patenty-k-androidu/>.

[17] KURT, M. *OpenBSD's Position Independent Executable (PIE) Implementation.* [online]. Presentations: NYCBSDCon 2008, poslední aktualizace 2008 [cit. 30. 10. 2012]. Dostupné z URL: <http://www.openbsd.org/papers/nycbsdcon08-pie/>.

[18] RUSSAKOVSKII, A. *Custom ROMs For Android Explained - Here Is Why You Want Them.* [online]. Android Police 2012, poslední aktualizace 20. 8. 2012 [cit. 8. 11. 2012]. Dostupné z URL: <www.androidpolice.com/2010/05/01/custom-roms-for-android>.

[19] *Radio-frequency identification. In: Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001–2012, poslední aktualizace 25. 9. 2012 [cit. 5. 11. 2012]. Dostupné z URL: <http://en.wikipedia.org/wiki/Radio-frequency_identification>.

[20] LIM SIONG BOON. *NFC (Near Field Communication), RFID* [online]. poslední aktualizace 3. 3. 2012 [cit. 7. 11. 2012]. Dostupné z URL: <http://www.siongboon.com/projects/2012-03-03_rfid/index.html>.

[21] VOJTĚCH, L. *RFID - technologie pro internet věcí.* Pandatron.cz: Elektronický magazín [online]. poslední aktualizace 14. 4. 2009 [cit. 7. 11. 2012]. Dostupné

z URL: <http://pandatron.cz/?733&rfid_-_technologie_pro_internet_veci>.

[22] SOMMEROVÁ, M. *Základy RFID technologií: výukový materiál pro Logistickou akademii* [online]. poslední aktualizace 1. 11. 2012 [cit. 7. 11. 2012]. Dostupné z URL: <http://rfid.vsb.cz/miranda2/export/sites-root/rfid/cs/okruhy/informace/RFID_pro_Logistickou_akademii.pdf>.

[23] *Near field communication. In: Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001–2012, poslední aktualizace 3. 11. 2012 [cit. 5. 11. 2012]. Dostupné z URL: <http://en.wikipedia.org/wiki/Near_field_communication>.

[24] *Near Field Communication. In: Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001–2012, poslední aktualizace 8. 11. 2012 [cit. 5. 11. 2012]. Dostupné z URL: <http://cs.wikipedia.org/wiki/Near_Field_Communication>.

[25] *NFC tech.* [online]. 2012, [cit. 6. 11. 2012]. Dostupné z URL: <http://www.nfctech.cz/>.

[26] KORB, K.; PULTZNER, M.; POSPÍŠIL, J. *Near field.* [online]. 2012, [cit. 8. 11. 2012]. Dostupné z URL: <http://nearfield.cz/>.

[27] *NFC Forum* [online]. 2012, [cit. 7. 11. 2012]. Dostupné z URL: <http://www.nfc-forum.org>.

[28] *Near Field Communication* [online]. 2012, [cit. 12. 11. 2012]. Dostupné z URL: <http://www.nearfieldcommunication.org/>.

[29] *Tyfone connecting money and mobility* [online]. 2012, [cit. 12. 11. 2012]. Dostupné z URL: <http://www.tyfone.com/>.

[30] *Smart Sim Solutions* [online]. 2012, [cit. 12. 11. 2012]. Dostupné z URL: <http://www.smartsim.info/>.

[31] *Watchdata* [online]. 2010, [cit. 12. 11. 2012]. Dostupné z URL: <http://www.watchdata.com/>.

[32] *NFC Data Exchange Format. In: Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001–2012, poslední aktualizace 26. 9. 2012 [cit. 20. 11. 2012]. Dostupné z URL: <http://cs.wikipedia.org/wiki/NDEF>.

[33] *libnfc.org - Public platform independent Near Field Communication (NFC) library* [online]. poslední aktualizace 21. 11. 2011 [cit. 1. 12. 2012]. Dostupné z URL: <http://www.libnfc.org/>.

[34] LIFCHITZ, R. *Hacking the NFC credit cards for fun and debit* [online]. Hackito Ergo - Paris, France, poslední aktualizace 13. 4. 2012 [cit. 1. 12. 2012]. Dostupné z URL: <http://code.google.com/p/readnfccc/downloads/detail?name=hes2012-bt-contactless-payments-insecurity.pdf>.

[35] HRON, M. *Stačí šikovný mobil a v Plzni budete jezdit zadarmo* [online]. 1999—2012 MAFRA a.s., poslední aktualizace 15. 4. 2011 [cit. 1. 12. 2012]. Dostupné z URL: <http://mobil.idnes.cz/mob_tech.aspx?c=A110413_150023_mob_tech_hro>.

[36] HASELSTEINER, E.; BREITFUß, K. *Security in Near Field Communication (NFC)* [online]. Philips Semiconductors Mikronweg 1, Austria, poslední aktualizace 13. 7. 2011 [cit. 1. 12. 2012]. Dostupné z URL: <http://events.iaik.tugraz.at/RFIDSec06/Program/papers/002%20-%20Security%20in%20NFC.pdf>.

[37] *Securing Near Field Communication:* [online]. master's thesis. Norwegian University of Science and Technology Department of Telematics , 12. 6. 2009 135 s. Supervisor: Stig Frode Mjolsnes [cit. 3. 12. 2012]. Dostupné z URL: <http://ntnu.diva-portal.org/smash/get/diva2:347744/FULLTEXT01>.

[38] *NFC-SEC: NFCIP-1 Security Services and Protocol* [online]. 2nd Edition / June 2010, Standard ECMA-385 © Ecma International 2009, [cit. 3. 12. 2012]. Dostupné z URL: <http://www.ecma-international.org/publications/files/ECMA-ST/ECMA-385.pdf>.

[39] MERTLÍK, T.; ROSENBERG, M. *Technologie NFC – popis, bezpečnost a využití.* [online]. Elektrorevue: Časopis pro elektrotechniku. 2013, s. 8. ISSN 1213-1539., [cit. 20. 4. 2013]. Dostupné z URL: <http://www.elektrorevue.cz/cz/clanky/informacni-technologie/0/technologie-nfc---popis--bezpecnost-a-vyuziti/>.

[40] GARGENTA, Marko. *Learning Android.* 1st ed. Sebastopol, Calif.: O'Reilly, c2011, xvii, 245 p. ISBN 14-493-9050-1.

[41] MEDNIEKS, Z., L. DORNIN, M. NAKAMURA a B. MEIKE. *Programming Android.* 1st ed. Sebastopol: O'Reilly, 2011, xvi, 482 s. ISBN 978-1-449-38969-7.

[42] SIX, Jeff. *Application security for the Android platform.* 1st ed. Sebastopol, CA: O'Reilly, 2011c2012, x, 97 p. ISBN 14-493-1507-0.

[43] MILETTE, Greg; STROUD, Adam. *Professional Android sensor programming.* Hoboken, N.J.: Wiley, c2012, xxxiii, 517 p. ISBN 11-181-8348-7.

# SEZNAM SYMBOLŮ, VELIČIN A ZKRATEK

ADT Vývojová sada pro Android – Android Development Tools

AES Pokročilý šifrovací standard – Advanced Encryption Standard

API Rozhraní pro programování aplikací – Application Programming Interface

APK Android aplikační balíček – Android Application Package

ARM Pokročilý stroj s RISC – Advanced RISC Machine

ASK Amplitudové klíčování – Amplitude Shifting Key

ASLR Náhodné rozložení adresního prostoru – Address Space Layout Randomization

AVD Androidové Virtuální zařízení – Android Virtual Device

DES Data Encryption Standard

DEX Dalvik Executable

GPS Globální systém polohy – Global Positioning System

HAL Hardwarová abstraktní vrstva – Hardware Abstraction Layer

IPC Meziprocesní komunikace – Inter process Communication

JDK Vývojová sada pro Javu – Java Development Kit

JIT Just in Time

LLCP Logical Link Control Protocol

MIME Multipurpose Internet Mail Extensions

MMS Multimediální zpráva – Multimedia Messaging Service

NDEF NFC Data Exchange Format

NFC Komunikace v blízkém poli – Near Field Communication

NFCIP Near-Field Communication Interface and Protocol

OS Operační systém – Operating System

PIE Pozičně nezávislý spustitelný soubor – Position Independent Executable

RAM Náhodný přístup k paměti – Random Access Memory

RFID Identifikace na rádiové frekvenci – Radio Frequency Identification

RISC Redukovaná instrukční sada počítače – Reduced Instruction Set Computing

RPC Vzdálené volání procedur – Remote Procedure Call

SDK Programová vývojová sada – Software Development Kit

SHA Bezpečnostní hashovací algoritmus – Secure Hash Algorithm

SIM Identifikační modul účastníka – Subscriber Identity Module

SMS Krátká textová zpráva – Short Message Service

TCP Transmission Control Protocol

UDP User Datagram Protocol

UID Unikátní identifikační číslo – Unique Identification Number

UMTS Univerzální mobilní telekomunikační systém – Universal Mobile Telecommunications System

URI Uniform Resource Identifier

VM Virtuální stroj – Virtual Machine

WiFi Wireless Fidelity

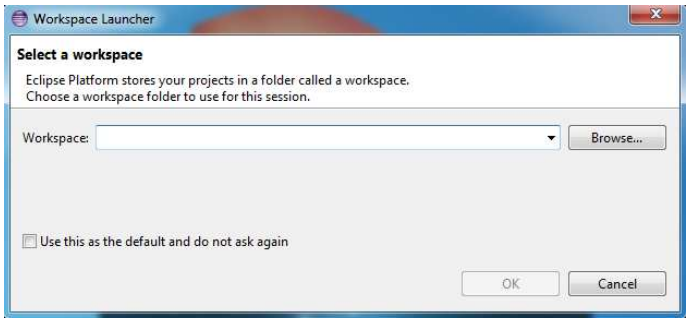
# SEZNAM PŘÍLOH

# A PŘÍLOHY

## A.1 Instalace vývojového prostředí Eclipse a Android SDK

Program Eclipse je jediné oficiálně podporované vývojové prostředí pro platformu Android od Google inc. Je to open source vývojový software[1]. Eclipse je nabízen již v mnoha variantách. Je to nejrozšířenější vývojové prostředí pro programovací jazyk Java. Díky přídavným pluginům lze na Eclipse realizovat i mnohé další např.: C++, C#, PHP, SQL, návrh UML nebo XML.

V dnešní době je Eclipse nabízen ve verzi 4.2 (Juno). Pro programování aplikací na Android bohatě postačí verze Clasic, která se obohatí o další potřebné pluginy. Eclipse lze stáhnout na adrese: <`http://www.eclipse.org/downloads/`> a velikost programu je bezmála 182 MB. Žádná instalace není potřeba, stačí jen rozbalit archiv s vývojovým prostředím. Po spuštění Eclipse bude uživatel vyzván k výběru pracovního prostoru viz. obrázek A.1. Zde stačí zadat složku ve které si bude Eclipse ukládat informace a data o projektech.




Obr. A.1: Výběr pracovního prostoru v Eclipse.

Dále je potřeba stáhnout JDK (Java Development Kit) v základní verzi bez Netbeans[2] , protože Android staví na programovacím jazyku Java. JDK je základním nástrojem pro práci s Javou. Obsahuje virtuální stroj, který je potřeba na běh programů psaných pod Javou. Nehledě na to, jestli se programuje aplikace na běžné počítače nebo pro mobily. JDK lze stáhnout na adrese: <`http://www.oracle.com/technetwork/java/javase/downloads/index.html`>. JDK je již ve verzi 7u7 a zabírá 95 MB. Při instalaci se nenastavuje nic speciálního, jen klasicky „proklikat“.

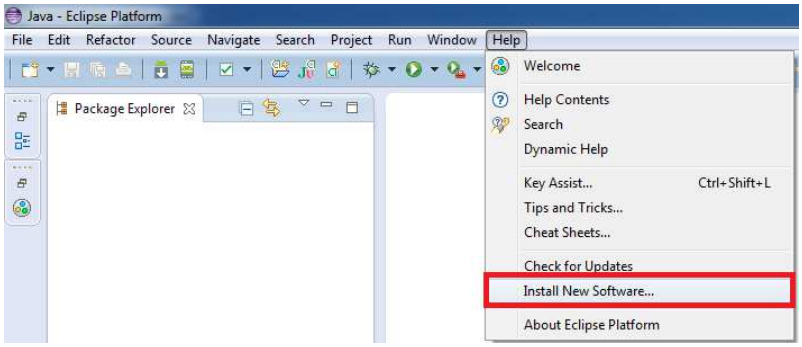
[1]Počítačový software s otevřeným zdrojovým kódem. Otevřenost kódu znamená, že je zdrojový kód dostupný pro širokou veřejnost a při dodržení jistých podmínek, lze tento kód využívat.

[2]Další vývojové prostředí jako je Eclipse.

Následující fáze, je stažení tzv. SDK (Software Development Kit) pro OS (operační systém) Android, což je balíček s různými nástroji na programování aplikací pro OS Android ve vývojovém prostředí Eclipse. Na následující adrese lze stáhnout nejnovější balíček SDK <`developer.android.com/sdk/index.html`>, je již ve verzi 20.0.3 a zabírá téměř 71 MB. Tento balíček má jednoduchou instalaci opět stačí jen „proklikat“ na konec instalace.

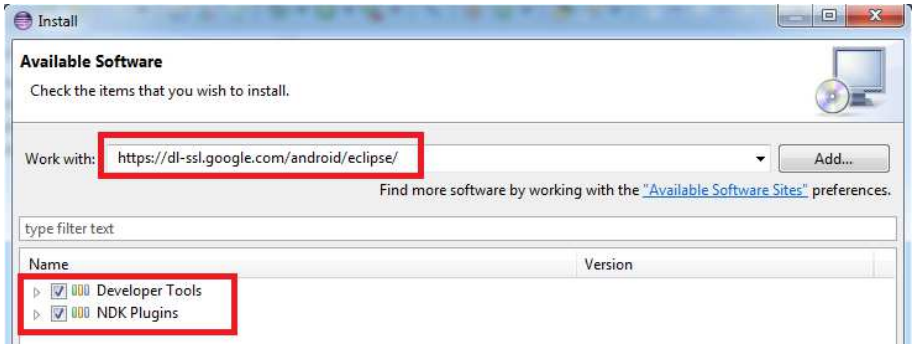
Pokud jsou již všechny předchozí díly skládanky nainstalované, stačí je jen složit pomocí pluginu ADT (Android Development Tools) v prostředí Eclipse, který přidá vývojové prostředí pro Android do Eclipse.
Plugin se stáhne následujícím způsobem:

1. V Eclipse zvolit nabídku Help → Install New Software... (obr. A.2).




Obr. A.2: Instalace nového softwaru do Eclipse.

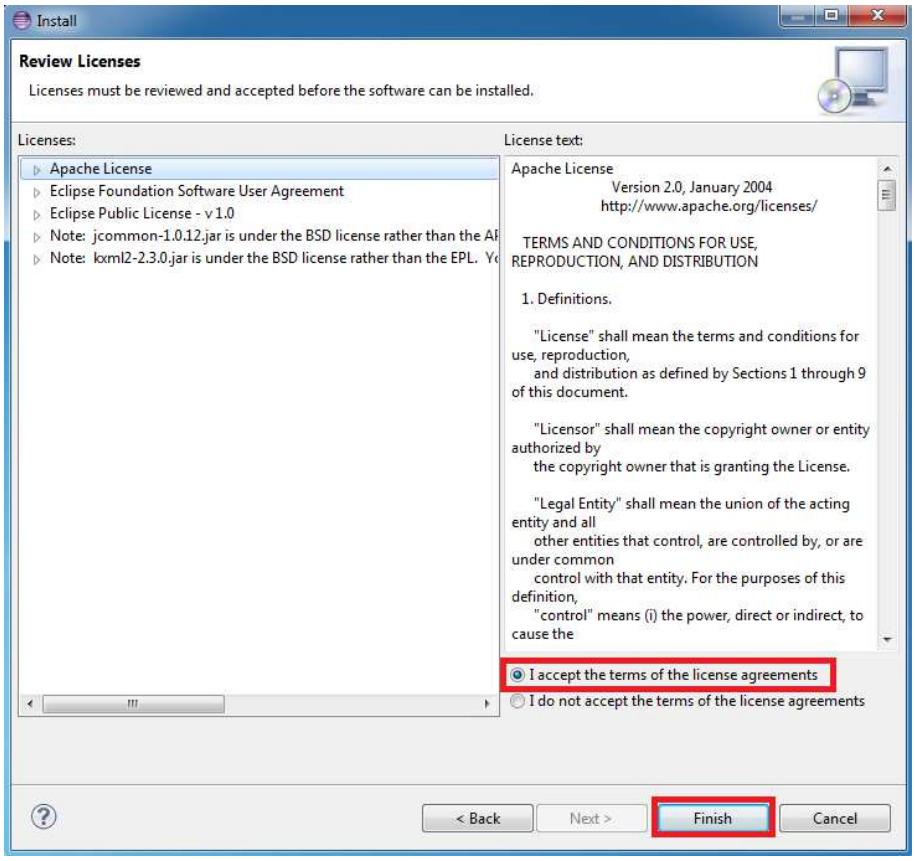
2. Otevře se nové okno pro získání nových pluginů. Do okénka Work with napsat adresu: <`https://dl-ssl.google.com/android/eclipse/`> a potvrdit tlačítkem Enter. V okně se objeví dvě položky. Ty se musí zaškrtnout a dole potvrdit tlačítkem Next. Vše je zobrazeno na obrázku A.3. Ostatní políčka nechat zaškrtnuté tak jak jsou.




Obr. A.3: Instalace ADT (Android Development Tools) do Eclipse.

3. Následující okno stačí jen potvrdit tlačítkem Next. Toto okno jen slouží pro kontrolu instalace a nahlédnutí na detaily instalovaného software.

4. V posledním kroku instalace se musí potvrdit licence používaného software pro vývoj Android. Instalace se dokončí tlačítkem FINISH. Vše je uvedeno na obrázku A.4. Posléze se začne software stahovat a instalovat. Lze to odložit na pozadí tlačítkem RUN IN BACKGROUND. Bude taky nutné potvrdit varovné okno tlačítkem OK, které varuje před instalováním nepodepsaného softwaru. Po dokončení instalace vyskočí okno, které žádá o restartování Eclipse, aby mohl plugin začít fungovat.



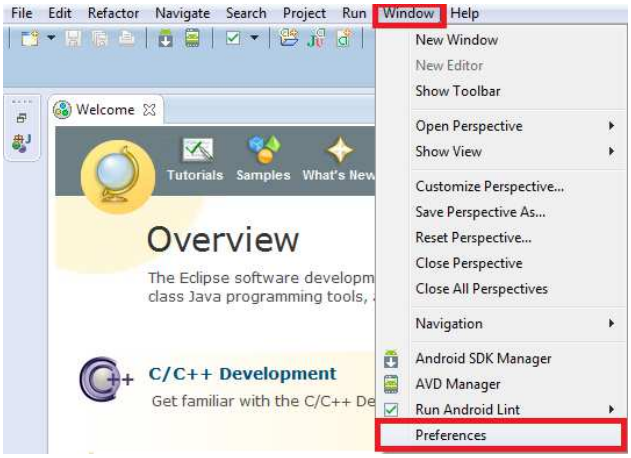

Obr. A.4: Poslední krok instalace ADT do programu Eclipse.

Po dodržení tohoto návodu, lze v Eclipse programovat aplikace pro OS Android. Dále je však nutné stáhnout a nastavit konkrétní verzi Androidu, pro spouštění a testování naprogramovaných aplikací. Jak to vše udělat, je uvedeno v následujícím návodu.

## A.2 Stažení a nastavení verze OS Android pro spuštění a ladění aplikací

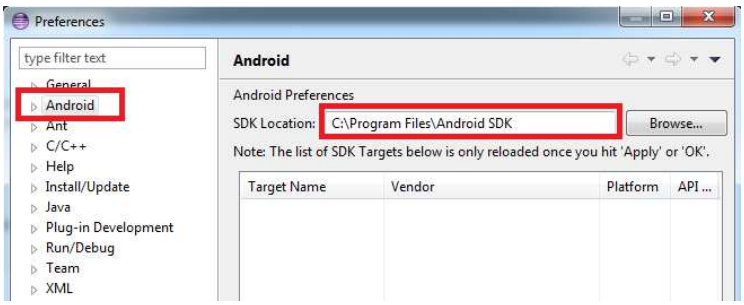
Pro testování vytvořených programů pro OS Android je potřeba virtuální OS Android přímo v PC. Tento návod ukazuje jak ho lze stáhnout a jak ho virtualizovat.

1. Nejprve je nutné nastavit plugin stažený do Eclipse, aby dokázal spolupracovat s vytvořenými emulátory Androidu. Stačí otevřít Eclipse a otevřít nastavení: Windows → Preferences (obr. A.5).




Obr. A.5: Nastavení SDK do Eclipse.

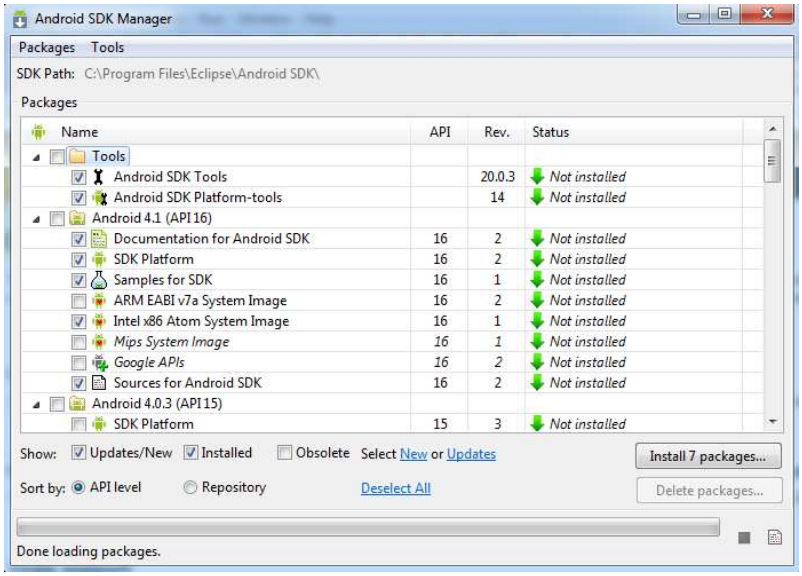
2. V nově otevřeném okně se musí přejít na záložku Android. Stačí zde zadat, pokud si to tam již Eclipse nezadal sám, do políčka SDK Location cestu, kam byl nainstalovaný balíček SDK, nejčastěji to bude právě cesta C:\Program Files\Android SDK (obr. A.6).




Obr. A.6: Nastavení cesty k SDK v Eclipse.

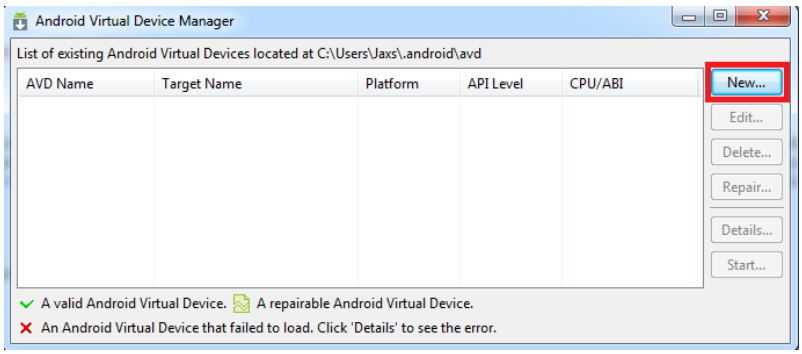
3. Dalším krokem bude stažení verze OS Android pro emulaci na procesoru Intel/AMD. U Eclipse otevřít nabídku Window → Android SDK Manager (podobné z obrázku A.5). Dále se otevře nabídka, ze které lze přímo stahovat verze OS Android do emulátoru a případně další pomůcky. Podle obrázku A.7

byl vybrán nejnovější OS Android ve verzi 4.1 Jelly Bean a ukázky aplikací spolu s dokumentací. Důležité je pro emulaci vybrat ještě balíčky Android SDK Tools a Android SDK Platform-tools. Po vybrání všech balíčků instalace započne stisknutím tlačítka Install (počet balíčků) packages. Tímto způsobem by se vybraly i jiné verze OS Android.




Obr. A.7: Stažení OS Android pro emulaci.

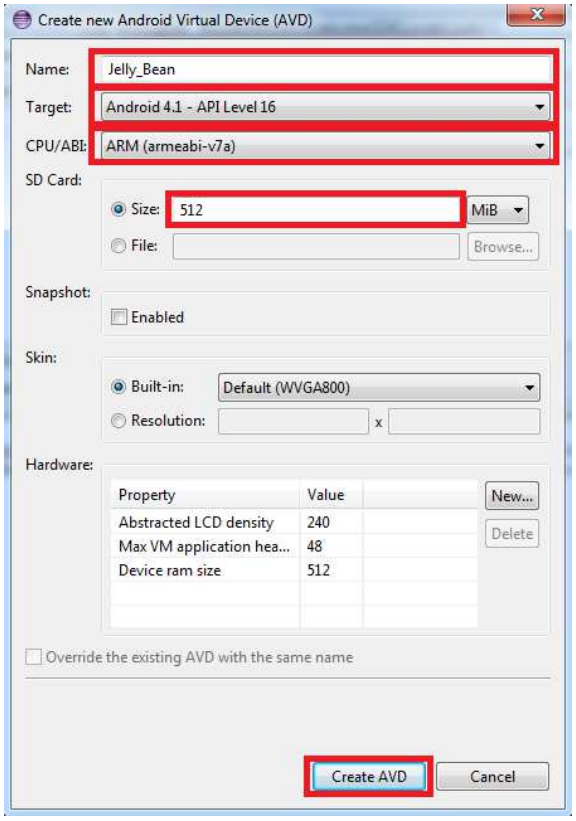
4. Teď už jen stačí vytvořit virtuální OS Android v programu AVD (Android Virtual Device) Manager. V Eclipse nabídka Window → AVD Manager (podobné z obrázku A.5). Otevře se nová nabídka, kde se budou zobrazovat vytvořené virtuální OS Android. Při vytváření nové virtualizace stačí kliknout na tlačítko New ... (obr. A.8).




Obr. A.8: AVD Manager.

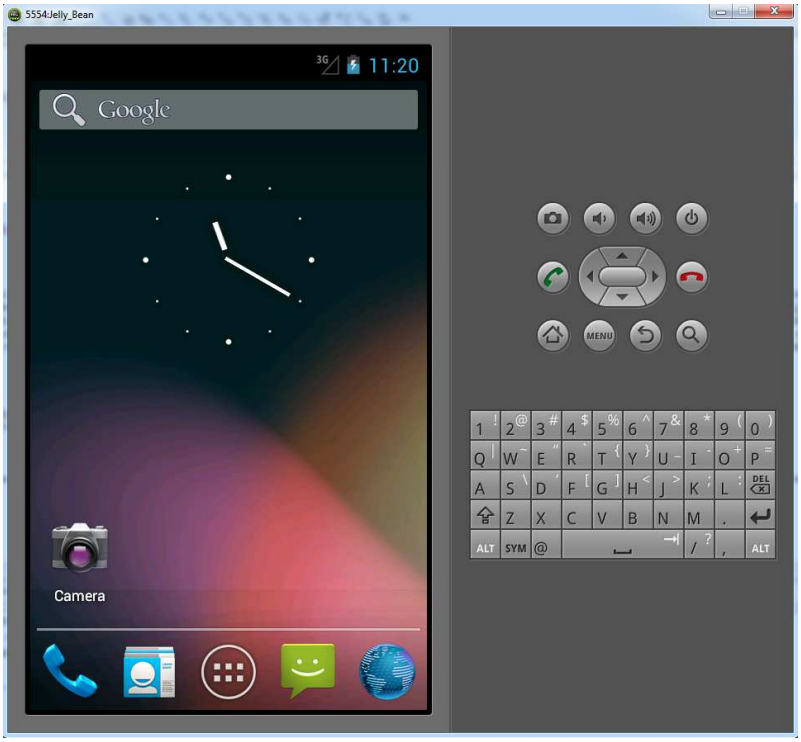
5. Nové okno nabízí řadu nastavení na virtualizaci OS. Podle obrázku A.9 lze učinit základní nastavení pro vytvoření virtualizace. Důležité položky jsou: Name

(jméno, které se bude zobrazovat v AVD Manageru), Target (výběr verze OS, zde je 4.1 Jelly Bean), CPU/ABI (pro jaký procesor se bude virtualizovat, nejčastěji ARM), SD Card (velikost virtuální karty), Snapshot (podpora zachycení tzv. snímku virtuálního stroje), Skin (velikost obrazovky), Hardware (lze virtualizovat hardware např. GPS, GSM, Akcelerometr a další). Po nastavení stačí vytvořit virtuální OS tlačítkem Create AVD. Pokud se spustí aplikace vytvářená v Eclipse, spustí se s ní automaticky i virtuální OS Android (při využití více virtualizovaných verzí OS vyskočí nabídka s výběrem verze OS). Pro samostatné spuštění, lze v AVD Manageru jen kliknout na tlačítko Start... a v následujícím okně nechat základní nastavení. Stačí zmáčknout tlačítko Launch.




Obr. A.9: Nastavení a vytvoření virtuálního OS Android.

Android bude chvilku nabíhat a to jen z důvodu, že se virtualizuje a běží na jiné platformě CPU než je naprogramován. Obrázek A.10 znázorňuje již spuštěný OS Android ve verzi 4.1 Jelly Bean.




Obr. A.10: Virtuální OS Android Jelly Bean.

## A.3 Zdrojový kód pro vytvoření Vcard formátu vybraného kontaktu

```
// Pokračování z funkce onActivityResult
// SWITCH case CONTACT_PICKER_RESULT:
String vcardstring = null;
// Jednoznačný identifikátor na kontakt
Uri contactUri = data.getData();
// Ukazatel na data v databázi SQLite
Cursor cursor = getContentResolver().query(contactUri, null, null,
    null, null);
// Přesune ukazatel na první záznam v kontakt listu
cursor.moveToFirst();
// Zjistí řádek z databáze kontaktů ve kterém je vybraný kontakt
String lookupKey = cursor.getString(cursor.getColumnIndex(
    ContactsContract.Contacts.LOOKUP_KEY));
```

```
// Vytvoří Vcard soubor a vrátí odkaz na něj
Uri uri = Uri.withAppendedPath(ContactsContract.Contacts.
    CONTENT_VCARD_URI, lookupKey);
AssetFileDescriptor fd;
// Otevření Vcard souboru pro čtení
fd = this.getContentResolver().openAssetFileDescriptor(uri,"r");
FileInputStream fis = fd.createInputStream();
byte[] buf = new byte[(int) fd.getDeclaredLength()];
// Načtení souboru do pomocné proměnné
fis.read(buf);
vcardstring = new String(buf); // Převod na text
fis.close();                   // Uzavření souboru
cursor.close();                // Uzavření ukazatele
```

## A.4 Zdrojový kód pro vytvoření odkazu na soubor (hudba, foto, video)

```
// Pokračování z funkce onActivityResult
// SWITCH case RESULT_LOAD_IMAGE_VIDEO_AUDIO:
// Jednoznačný identifikátor na soubor - video, muzika..
Uri selectedFile = data.getData();
// Zjištění umístění souboru z databáze
String[] filePathColumn = { MediaStore.Images.Media.DATA };
// Vytvoření ukazatele na konkrétní soubor v databázi
Cursor cursor2 = getContentResolver().query(selectedFile,
filePathColumn, null, null, null);
// Přesunutí ukazatele na první záznam
cursor2.moveToFirst();
// Zjištění indexu u ukazatele na soubor
int columnIndex = cursor2.getColumnIndex(filePathColumn[0]);
// Zjištění umístění souboru na zařízení
// Načtení cesty do textového řetězce
String filePath = cursor2.getString(columnIndex);
cursor2.close(); // Uzavřeni ukazatele
```

## A.5 Přehled nejznámějších Mime typů

| | Extension | MIME Type |
|---|---|---|
| **Android Application** | .apk | application/vnd.android.package-archive |
| **Text** | .txt | text/plain |
| | .csv | text/csv |
| | .xml | text/xml |
| **Web related** | .htm | text/html |
| | .html | text/html |
| | .php | text/php |
| **Image** | .png | image/png |
| | .gif | image/gif |
| | .jpg | image/jpg |
| | .jpeg | image/jpeg |
| | .bmp | image/bmp |
| **Audio** | .mp3 | audio/mp3 |
| | .wav | audio/wav |
| | .ogg | audio/x-ogg |
| | .mid | audio/mid |
| | .midi | audio/midi |
| | .amr | audio/AMR |
| **Video** | .mpeg | video/mpeg |
| | .3gp | video/3gpp |
| **Package** | .jar | application/java-archive |
| | .zip | application/zip |
| | .rar | application/x-rar-compressed |
| | .gz | application/gzip |

Tab. A.1: Typy Mime (Typy souborů)[4]

## A.6 Obsah DVD

1. Složka **Android SDK** obsahuje potřebné soubory na vývoj pro OS Android (SDK) + AVD.
2. Složka **Dokumentace** obsahuje zdrojové soubory pro program LaTex.
3. Složka **Eclipse** obsahuje spustitelný program Eclipse, ve kterém je už připravený plugin pro vývoj aplikací na operační systém Android.
4. Složka **LaTeX and others** obsahuje instalaci programu LaTex, Sumatry a TeXnicCenter.
5. Složka **Program NFC File Transfer** obsahuje zdrojové soubory aplikace NFC File Transfer pro Eclipse.
6. Soubor **Technologie NFC a její zabezpečení.pdf** je dokumentace k diplomové práci.