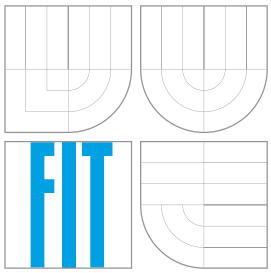
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

VYSÍLÁNÍ VIDEO STREAMU NA WINDOWS PHONE

TRANSMITTING STREAMED VIDEO ON WINDOWS PHONE

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

ADAM JEŽ

VEDOUCÍ PRÁCE
SUPERVISOR

Doc. Ing. ADAM HEROUT, Ph.D.

BRNO 2015

Abstrakt

Hlavním cílem této bakalářské práce je návrh a realizace aplikace pro mobilní systém Windows Phone, která bude schopna vysílat multimediální tok z kamery mobilního zařízení prostřednictvím paketové sítě. Pro navázání spojení a sjednání podmínek přenosu je využitý protokol RTSP, samotná multimediální data jsou posílána prostřednictvím protokolu RTP. V práci jsou popsány tyto protokoly a navazující protokoly jako RTCP a SDP. Stručně popsaný je také vývoj multimediální aplikace pro Windows Phone. V poslední části práce je rozebrán dopad transportního protokolu na kvalitu videa. Hlavním výsledkem práce je aplikace, která je téměř připravena pro publikování.

Abstract

The main goal of this bachelor thesis is the design and implementation of an application for mobile system Windows Phone which will be able to broadcast a multimedia stream from the mobile device camera via a packet network. The RTSP protocol is used to establish the connection and to negotiate the transmission conditions. The multimedia data themselves are sent via the RTP protocol. The aforementioned protocols and follow protocols such as RTCP and SDP are described in the thesis. Also the development of the multimedia application for Windows Phone is briefly described. The last part of the thesis deals with the impact of the transportation protocol on the quality of the video. The main result of the thesis is an application almost ready to be published.

Klíčová slova

mobilní aplikace, multimediální aplikace, protokoly pro přenos multimédií, Windows Phone, RTSP, RTP, RTCP, SDP, H.264

Keywords

mobile application, multimedia application, streaming Protocols, Windows Phone, RTSP, RTP, RTCP, SDP, H.264

Citace

Adam Jež: Vysílání video streamu na Windows Phone, bakalářská práce, Brno, FIT VUT v Brně, 2015

Vysílání video streamu na Windows Phone

Prohlášení

Čestně prohlašuji, že na bakalářské práci jsem pracoval samostatně na základě vlastních teoretických a praktických poznatků, konzultací a s použitím uvedených informačních zdrojů.

.....
Adam Jež
19. května 2015

Poděkování

Děkuji Doc. Ing. Adamu Heroutovi, Ph.D za pomoc při vedení bakalářské práce. Mé poděkování patří též panu Robertu Prokešovi za spolupráci při zavádění mé výsledné práce do již fungujícího systému.

© Adam Jež, 2015.
Tato práce vznikla jako školní dílo na Vysokém učení technickém v Brně, Fakultě informačních technologií. Práce je chráněna autorským zákonem a její užití bez udělení oprávnění autorem je nezákonné, s výjimkou zákonem definovaných případů.

Obsah

1 Úvod **2**

2 Protokoly pro přenos videa **3**
2.1 Protokol TCP/IP . . . 3
2.2 Real-Time Streaming Protocol (RTSP) . . . 5
2.3 Real-Time Protocol (RTP) . . . 10
2.4 Real-Time Control Protocol (RTCP) . . . 13
2.5 Session Description Protocol (SDP) . . . 14
2.6 Real Time Messaging Protocol (RTMP) . . . 15
2.7 H.264 . . . 16

3 Vývoj aplikací na Windows Phone **20**
3.1 Windows Runtime . . . 21
3.2 Microsoft Media Foundation . . . 21
3.3 Enkodér H.264 . . . 23

4 Návrh aplikace pro vysílání videostreamu **25**
4.1 Specifikace požadavků . . . 25
4.2 Návrh grafického uživatelského rozhraní . . . 26
4.3 Zvolené protokoly . . . 27

5 Implementace aplikace **28**
5.1 Přímé připojení k mobilní aplikaci . . . 28
5.2 Využití serveru pro registraci vysílání . . . 29
5.3 Využití CDN . . . 30
5.4 Výsledná aplikace . . . 30
5.5 Popis implementace knihovny . . . 34
5.6 Testování . . . 36

6 Závěr **40**

A Instalační příručka **43**
A.1 Obsah přiloženého média . . . 43
A.2 Instalace aplikace . . . 43
A.3 Spuštění registračního serveru . . . 44

Kapitola 1

Úvod

Dnešní mobilní telefony nabízí mnohem více možností než jen uskutečnění hovoru nebo zaslání textových zpráv. Jejich výpočetní výkon a kvalita dalších komponent umožňuje nová využití nebo nahrazení specializovaných zařízení. Jedním z příkladů jsou levnější fotoaparáty, u kterých je kvalita výsledných fotografií srovnatelná s mobilními fotoaparáty. IP kamera by mohla být dalším zařízením, jež by mohlo být nahrazeno. Kvalitnější IP kamery mohou být finančně náročnější a uživatel ve výsledku nemusí být se zakoupeným zařízením spokojen. Aplikací pro mobilní zařízení si lze ověřit reálné využití IP kamery. Pro mobilní operační systém Windows Phone 8.1 doposud neexistuje žádná aplikace s podobnou funkcionalitou.

Cílem této práce je vytvořit mobilní aplikaci pro operační systém Windows Phone. Aplikace má za úkol vysílat multimediální data z kamery mobilního zařízení k uživatelům, kteří chtějí záběry sledovat. Aplikace má poskytnout uživateli co nejjednodušší přístup k výsledným záběrům z kamerového zařízení. Uživatelské rozhraní aplikace má dovolit snadný a jednoduchý přístup k nastavení výsledné kvality videa.

Dokument obsahuje více kapitol. Kapitola číslo 2 je věnována analýze protokolů sloužících pro přenos videa po paketových sítích. V této části jsou zpracovány protokoly potřebné k implementaci serveru přenášející streamované video. Taktéž jsou zmíněné problémy při poskytování streamovaného obsahu z neveřejné IP adresy. V kapitole 3 jsou rozebrány technologie potřebné k implementaci aplikace využívající video z fotoaparátu na systému Windows Phone. Další kapitoly se budou věnovat návrhu, implementaci a testování vytvořené aplikace. Poslední kapitolou je závěr obsahující zhodnocení a možné rozšíření projektu. Součástí dokumentu je návod k instalaci aplikace.

Kapitola 2

Protokoly pro přenos videa

V této kapitole budou popsány protokoly, které jsou využívány pro přenos videa přes paketové sítě. Nejprve budou stručně popsány nižší vrstvy IP protokolu a techniky, které následně ovlivňují samotnou implementaci aplikace využívající tyto protokoly. Dále jsou popsány protokoly aplikační vrstvy, které ustavují spojení mezi klienty (RTSP), přenášejí samotná multimediální data (RTP), kontrolují přenos dat (RTCP), popisují přenášené multimediální informace (SDP) a komprimují multimediální data (H.264).

2.1 Protokol TCP/IP

Protokol TCP/IP je založen na čtyřvrstvém referenčním modelu. Čtyři vrstvy modelu jsou síťové rozhraní, síťová, transportní a aplikační vrstva. Každá vrstva modelu struktury protokolu TCP/IP odpovídá jedné nebo více vrstvám sedmivrstvého referenčního modelu ISO/OSI. Popsány budou síťová a transportní vrstva, které mají dopad na implementaci výsledné aplikace.

2.1.1 Síťová vrstva

Nejdůležitějším úkolem síťové vrstvy je směrování paketu, což zajišťuje protokol IP. Protokol poskytuje nespolehlivou datagramovou službu. Z pohledu posílání multimediálních dat po síti je důležitá fragmentace paketů a překlad adres [13].

Fragmentace

Fragmentace IP datagramu se provádí ve chvíli, kdy je jeho velikost větší než největší přenosová jednotka linkové vrstvy (MTU – Maximum Transmission Unit). Datagram je fragmentován směrovačem a je sestavován až koncovým hostem [16]. Nadměrná fragmentace vede k snížení efektivity přenosu. Rozdělení přenosových technologií na linkové vrstvě podle velikosti MTU (v bajtech) [16]:

- Ethernet – 1500,
- IEEE 802.3 – 1492,
- Token Ring – od 4440 do 17940,
- FDDI – 4352.

Překlad adres

Jedná se o metodu, která zajišťuje úpravu IP datagramu za účelem komunikace na internetu bez potřeby vlastnit veřejnou IP adresu [16]. Metoda je dnes hojně používaná díky nesporným výhodám:

- není potřeba vlastnit rozsah veřejných adres pro lokální síť,
- lze měnit adresy v lokální síti bez potřebných souhlasů internetového poskytovatele,
- lze změnit poskytovatele bez nutnosti úpravy nastavení lokální sítě.

Nevýhodou pro poskytování nějakého obsahu ze zařízení v lokální síti je nemožnost explicitně adresovat toto zařízení (neboli komunikaci nelze iniciovat zařízením cílícím na zařízení v lokální síti s privátní adresou). Tento problém lze částečně řešit technikou zvanou *forwarding*, která mapuje veřejnou IP adresu a port na lokální IP adresu a port.

2.1.2 Transportní vrstva

Je zahrnuta jak v referenčním ISO/OSI modelu, tak v modelu TCP/IP. Zajišťuje komunikaci mezi libovolnými dvěma uzly sítě. Architektura TCP/IP obsahuje dva základní protokoly na této vrstvě: TCP (Transmission Control Protocol) a UDP (User Datagram Protocol) [13].

Každé síťové rozhraní je identifikováno IP adresou a jednotlivá spojení jsou identifikována číslem portu. K plné identifikaci spojení v TCP/IP architektuře tedy slouží tato pětice:

- zdrojová IP adresa,
- cílová IP adresa,
- zdrojový port,
- cílový port,
- protokol transportní vrstvy.

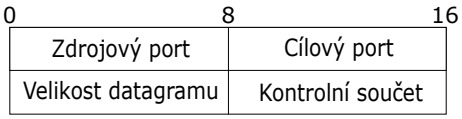
Porty jsou reprezentovány jako 16bitová celočíselná nezáporná čísla a dělí se na [17]:

- systémové porty (0 až 1023), které jsou přiřazeny základním internetovým službám organizací IANA;
- uživatelské porty (1024 až 49151), které taktéž registruje organizace IANA;
- dynamické anebo privátní porty (49152 až 65535), které nejsou přiřazovány žádným službám a slouží pro proprietární použití.

Nejznámější mechanismus pro přístup k službám transportní vrstvy jsou tzv. *BSD sockety* původně vyvinuté pro BSD (Berkeley Software Distribution) Unix systém, dnes používané ve většině operačních systémů. Tyto sockety zaobalují plnou identifikaci spojení.

UDP

Protokol nabízí službu bez záruky doručení a bez záruky dodržení pořadí paketů [15]. Jde o nespojovanou službu. Jediné, co UDP protokol zajišťuje, je zabalení a rozbalení UDP hlavičky. Zodpovědnost za správné pořadí doručených datagramů musí zajistit služba na aplikační vrstvě podle své potřeby. Protokol zpřístupňuje rychlé a efektivní přenosové služby síťové vrstvy. Je tedy vhodný pro tzv. *realtime* aplikace, pro něž je podstatné včasné doručení datagramu.



Obrázek 2.1: UDP hlavička

Jak lze z obrázku vyčíst, UDP hlavička má fixní velikost 8 bajtů. Hlavička TCP protokolu je variabilní s minimální velikostí 20 bajtů a maximální 60 bajtů. Při množství dat, které vyprodukuje multimediální obsah, se jeví UDP protokol jako ideální.

TCP

Protokol zajišťuje spolehlivý přenos dat se zachováním pořadí streamu [14]. Jedná se o spojovanou službu. Před zahájením komunikace mezi dvěma uzly je nutné nejprve inicializovat spojení tzv. *handshakem*. Protokol TCP očekává od aplikační vrstvy jednotlivé bajty, naproti tomu UDP očekává bloky bajtů. Protokol využívá vhodnou vyrovnávací paměť, po jejíž naplnění jsou data odeslána. Existuje mechanismus pro vynucení odeslání dat z vyrovnávací paměti. Na straně příjemce jsou data ukládána také do vyrovnávací paměti a poskytována po jednotlivých bajtech aplikační vrstvě. Další vlastnosti protokolu jsou:

- plně duplexní přenos,
- kumulativní potvrzování,
- efektivní využití přenosového pásma.

2.2 Real-Time Streaming Protocol (RTSP)

Real-Time Streaming Protocol je protokol na aplikační vrstvě, který ovládá doručování dat reálného času. RTSP poskytuje rozšíření pro podporu doručení dat na vyžádání, například audio nebo videosoubory. Protokol je popsán v RFC 2326 z roku 1998 ve verzi 1.0 [19]. V současné době je vyvíjena verze 2.0, která není zpětně kompatibilní.[1] RTSP protokol je v mnoha ohledech podobný protokolu HTTP. Jedná se také o textový protokol s architekturou *dotaz odpověď*. Oproti protokolu HTTP je protokol RTSP stavový a zprávu typu dotaz může vyvolat server, také samotný obsah je oproti HTTP přenášen jiným protokolem. Výchozí číslo portu RTSP protokolu je 554 a ve výchozím stavu využívá služeb protokolu TCP.

[1] http://tools.ietf.org/html/draft-ietf-mmusic-rfc2326bis-36

2.2.1 Základní vlastnosti protokolu

Každý multimediální stream nebo prezentace skládající se z více multimediálních streamů může být určen RTSP URL ve tvaru:

```
('rtsp'|'rtspu')'://'host[':'port][absolutní_cesta]
```

Schéma *rtsp* požaduje komunikaci s využitím TCP protokolu a naopak schéma *rtspu* vyžaduje využití nespolehlivého protokol UDP. Interpretace absolutní cesty záleží na konkrétním serveru. Může se například jednat o interpretace v rámci souborového systému při nabízení audiovizuálního obsahu na požádání. V rámci komunikace se musí ustanovit, v jakém módu bude posílání dat probíhat. Lze volit z následujících módů:

- *unicast* – multimediální data jsou posílána ke klientovi na port, který se vybere, případně k tomu může být využito již vytvořené spojení, na kterém probíhá komunikace RTSP protokolu;
- *multicast* – kde server sdělí multicastovou adresu a port, kde jsou data vysílána;
- *multicast* – kde server je vyzván k připojení k již existující multicastové konferenci.

2.2.2 Formát zpráv

Text RTSP zpráv používá kódování znakovou sadou *UTF-8*. Jednotlivé řádky jsou ukončeny dvojicí znaků `CR LF`. Zpráva obsahuje hlavičku a tělo zprávy, které může mít nulovou délku. Konec hlavičky zprávy je indikován dvojicí znaků `CR LF`. Hlavička se skládá z prvního řádku, který závisí na typu zprávy *dotaz* nebo *odpověď*, následovaným řádky s parametry dotazu nebo odpovědi. V případě, že není obsažen v hlavičce parametr *Content-Length*, se předpokládá nulová velikost těla zprávy. Každá zpráva musí obsahovat parametr *CSeq*. Ten obsahuje číslo identifikující dotaz. Odpověď na tento dotaz musí obsahovat stejné číslo v parametru *CSeq*. Podle standardu musí další dotaz obsahovat číslo z předcházejícího dotazu inkrementované o 1.

Zpráva typu dotaz obsahuje na prvním řádku tzv. dotazovací řádek (request-line), identický jako v protokolu HTTP 1.0. V něm se nachází název metody, požadovaná *RTSP URL* a verze RTSP protokolu. Příklad dotazovacího řádku:

```
OPTIONS rtsp://192.168.1.1/stream RTSP/1.0\r\n
```

Ve zprávě typu *odpověď* se nachází na prvním řádku tzv. *odpovídací řádek* (anglicky response-line), který obsahuje verzi protokolu, trojčíselný stavový kód, který představuje úspěšnost splnění dotazu, a krátké textové vyjádření stavového kódu. Příklad:

```
RTSP/1.0 200 OK\r\n
```

2.2.3 Metody

Aplikace využívající protokol RTSP musejí implementovat pouze povinné metody. Ostatní jsou volitelné podle potřeb aplikace. Možnosti rozšíření protokolu [19]:

- existující metody mohou být rozšířeny novými parametry, jestliže lze bezpečně tyto parametry ignorovat příjemcem;
- mohou být přidány nové metody, když příjemce nerozumí požadavku, odešle odpověď s chybovým kódem 501 (neimplementováno) a odesílatel by se neměl pokoušet odeslat metodu znovu;
- nová verze protokolu může být definována.

Minimální implementace serveru musí být schopná porozumět následujícím metodám [19]: `SETUP`, `TEARDOWN`, `OPTIONS` a buď `PLAY`, nebo `RECORD`. RTSP klient musí umět generovat metody: `SETUP`, `TEARDOWN` a buď `PLAY`, nebo `RECORD`. V případě implementace metody `RECORD` musí implementovat klient taktéž metodu `ANNOUNCE`.

Název metody	Směr komunikace	Objekt	Požadavek
`DESCRIBE`	K → S	P, S	doporučeno
`ANNOUNCE`	K → S, S → K	P, S	volitelné
`GET_PARAMETER`	K → S, S → K	P, S	volitelné
`OPTIONS`	K → S, S → K	P, S	povinné (S ->K: volitelné)
`PAUSE`	K → S	P, S	doporučeno
`PLAY`	K → S	P, S	povinné
`RECORD`	K → S	P, S	volitelné
`REDIRECT`	S → K	P, S	volitelné
`SETUP`	K → S	S	povinné
`SET_PARAMETER`	K → S, S → K	P, S	volitelné
`TEARDOWN`	K → S	P, S	povinné
`REGISTER`	S → K	P, S	volitelné (neobsaženo ve verzi RTSP 1.0 [4])

Tabulka 2.1: Přehled RTSP metod, jejich směru (K: klient, S: server) a objektu (P: prezentace, S: multimediální proud dat), nad kterým se vykonávají [19]

2.2.4 Popis jednotlivých metod

DESCRIBE

Metoda slouží k získání popisu prezentace nebo multimediálního proudu dat identifikovaného pomocí URL. V parametru *Accept* lze specifikovat formát, kterému klient rozumí. Specifikace je zapsána ve formátu *MIME*. Server odpoví s popisem požadovaného zdroje zadaným formátem. Metoda se používá ve fázi inicializace a nijak nemění stav RTSP komunikace.

ANNOUNCE

Metoda slouží ke dvěma účelům podle směru požadavku:

- **ve směru K → S** posílá popis prezentace nebo multimediálního proudu dat identifikovaného pomocí URL,

- **ve směru S → K** aktualizuje popis již ustaveného *sezení* (doba od připojení klienta k serveru až po ukončení komunikace).

GET_PARAMETER

Metodou se žádá o získání hodnot parametrů prezentace nebo multimediálního proudu dat identifikovaného pomocí URL. Jednotlivé parametry jsou definovány v těle požadavku. Forma a obsah odpovědi jsou ponechány na konkrétní implementaci. Bez uvedení parametrů může být metoda použita pro ověření dostupnosti koncového bodu (*ping*).

SET_PARAMETER

Metoda slouží k nastavení hodnot parametrů prezentace nebo multimediálního proudu dat identifikovaného pomocí URL. Dotaz by měl obsahovat pouze jeden parametr tak, aby se dalo určit, proč daný dotaz selhal. Když obsahuje dotaz více parametrů, musí být nastaveny buď všechny, nebo žádný. Parametry definující síťové nastavení mohou být nastaveny pouze pomocí metody *SETUP*.

SETUP

Metoda slouží k dohodnutí se na způsobu přenosu multimediálního proudu dat identifikovaného pomocí URL. Klient může požádat o změnu parametrů pro přenos dat, přičemž při již aktivním přenosu může být tato změna serverem odmítnuta. V parametru *Transport* klient specifikuje transportní informace, které si zvolil, a server odpovídá jeho zvolenými parametry. Metodou je změněn stav RTSP komunikace a server alokuje potřebné zdroje pro posílání multimediálního proudu dat.

PAUSE

Metoda způsobí dočasné pozastavení posílání prezentace nebo multimediálního proudu dat specifikovaného pomocí URL. Všechny potřebné zdroje jsou ponechány, dokud není přenos obnoven nebo dokud neuplyne čas specifikovaný v metodě *SETUP* jako parametr *Timeout*. Metoda může obsahovat parametr *Range*, ve kterém je specifikován přesný časový rozsah, kdy bude přenos pozastaven. Metoda může měnit stav RTSP komunikace.

PLAY

Metodou klient žadá server o zahájení posílaní multimediálních dat přes mechanismus specifikovaný metodou *SETUP*. Klient nesmí zaslat metodu *PLAY*, pokud nebyl úspěšně ustanoven přenosový mechanismus metodou *SETUP*. Metoda může obsahovat parametr *Range*, ve kterém je specifikovaný začátek a konec streamovaného proudu dat. Tento rozsah je možné zapsat v časových formátech *SMPTE*, *NPT* a formátu časové značky definované v *ISO 8601*. Při nespecifikování časového rozsahu je streamované médium od jeho začátku. Jestliže byl předtím daný stream pozastaven metodou *PAUSE*, je přehrávání obnoveno. Při aktivním přehrávání může být metoda použita k testování dostupnosti serveru s tím, že nemá žádné důsledky. Metoda může měnit stav RTSP komunikace.

RECORD

Metodou lze zahájit zaznamenávání multimediálního vysílání. V případě obsažení parametru *Range* je zaznamenán pouze vybraný úsek. Při nezahrnutí parametru začne nahrávání okamžitě (v případě aktivního streamování). Při neaktivním vysílání je zaznamenán úsek specifikovaný v metodě *DESCRIBE*. Metoda mění stav RTSP komunikace.

REDIRECT

Požadavek informuje klienta, aby se připojil k jinému serveru. Obsahuje parametr *Location*, ve kterém je uvedena URL adresa zmíněného serveru.

TEARDOWN

Klient posílá požadavek o zastavení doručování streamu definovaného pomocí URI. Server uvolní zdroje spojené s daným proudem. Jestliže se jedná o prezentaci, je celé sezení s klientem ukončeno. Stav RTSP komunikace je změněn na neinicializovanou a nachází se v počátečním stavu. Pro obnovení vysílání je potřeba znovu provést požadavek metodou *SETUP*.

REGISTER

Metoda není součástí RTSP verze 1.0. Je součástí návrhu [4], který vypršel v roce 2014, ale je podporován populárními implementacemi RTSP serveru, např. LIVE555 [5]. Metoda slouží k informování koncového bodu o existující prezentaci nebo proudu dat identifikovaných URL adresou. Metoda obsahuje speciální volitelné parametry: *reuse_connection* a *preferred_delivery_protocol*. První zmiňovaný parametr navrhuje využití již existujícího spojení s klientem k další komunikaci. Druhý zmiňovaný parametr specifikuje preferovaný transportní protokol pro samotný multimediální stream. Kromě UDP protokolu lze zvolit tzv. *prolínání přes protokol TCP* (multimediální data jsou posílána po stejném spojení jako samotná RTSP komunikace).

2.2.5 Stavy RTSP serveru

Server se může nacházet ve 4 stavech[19]:

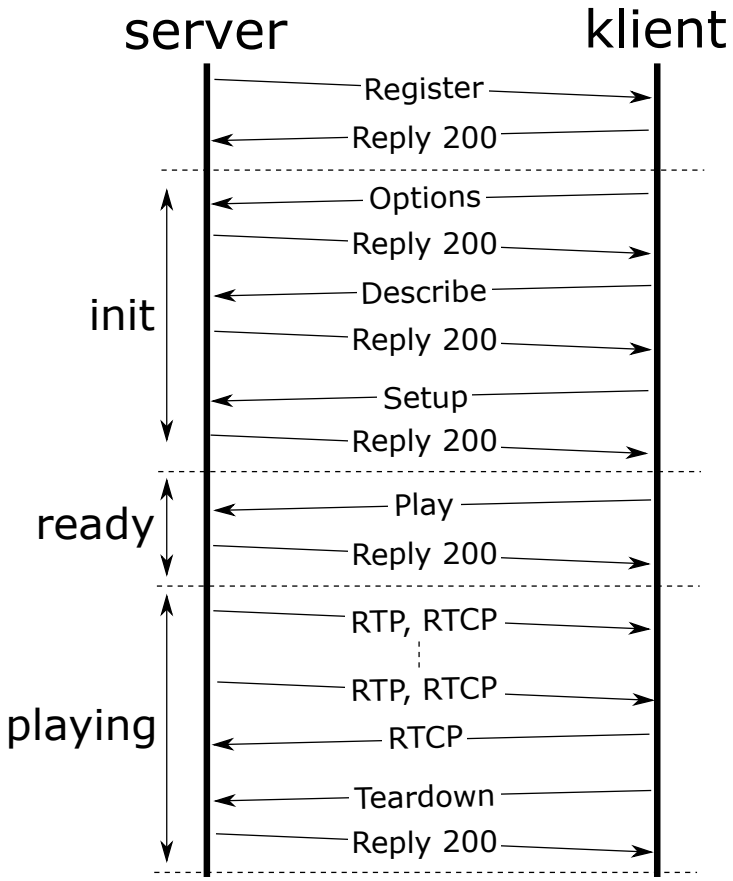
- *init* – počáteční stav,
- *ready* – server s klientem mají ustavené sezení a server čeká na jeden z příkazů `PLAY`, `RECORD`, `TEARDOWN`,
- *playing* – server posílá ujednaná data,
- *recording* – server nahrává ujednaná data.

2.2.6 Způsoby zasílání multimediálních dat

K dohodnutí způsobu zasílání dat slouží metoda SETUP a může být využitý jeden z následujících:

- **UDP nebo TCP** – zasílání dat probíhá přes samostatné spojení vybraným transportním protokolem,

- **prolínání přes protokol TCP** – multimediální data jsou zasílaná po stejném spojení, na kterém probíhá RTSP komunikace. K rozeznání odlišné komunikace po stejném spojení je před každý paket obsahující multimediální data vložen znak dolaru '$' následovaný jednobajtovým identifikátorem daného proudu. Identifikátory jsou přiřazovány v metodě `SETUP`. Za identifikátorem se nachází dvoubajtové číslo reprezentující velikost obsaženého paketu.



Obrázek 2.2: Příklad ustavení sezení a zasílání jednoho streamu. Stav RTSP komunikace při metodě REGISTER není definován, jelikož metoda není obsažena v původní verzi 1.0

2.3 Real-Time Protocol (RTP)

Protokol RTP slouží k přenosu multimediálních dat v reálném čase a působí na aplikační vrstvě TCP/IP modelu. Protokol je nezávislý na transportní vrstvě. Původní verze protokolu byla vydána v RFC 1889 v roce 1996 a poté nahrazena aktualizovanou verzí RFC 3550 [18] z roku 2003. Pro zajištění statistik přenášených dat, kvality služby a synchronizace více RTP streamu se používá ve spojení s protokolem RTCP, který bude rozebrán v samostatné kapitole.

Před zahájením vysílání dat je potřeba ustanovit sezení a sjednat podmínky, za kterých bude k přenosu docházet. K tomuto účelu jsou určeny signalizační protokoly, mezi které patří protokoly H.323, SIP nebo XMPP. Sezení se skládá z IP adresy (unicastové nebo multicastové) a párů čísla portů pro RTP a RTCP přenos. Každé sezení obsahuje pouze

jeden typ multimediálních dat, což umožňuje příjemci vybrat si pouze stream, který chce dostávat a následně přehrávat.

Protokol RTP byl navržen tak, aby mohl nést řadu různých multimediálních formátů a bylo možné další přidat bez potřeby aktualizovat standart RTP. Toho bylo docíleno definováním profilů a formátu multimediálních dat obsažených v RTP paketu. Během jednoho sezení je využíván pouze jeden profil, který je buď staticky definovaný v RFC 3551, nebo je dynamicky vytvořen jiným protokolem např. SDP, který bude později popsán. Profil specifikuje typ dat (audio, video atp.), kodek a další informace navázané na kodek. V závislosti na zvoleném profilu se může měnit sémantika některých částí RTP hlavičky. Formát multimediálních dat je různý dle zvoleného profilu.

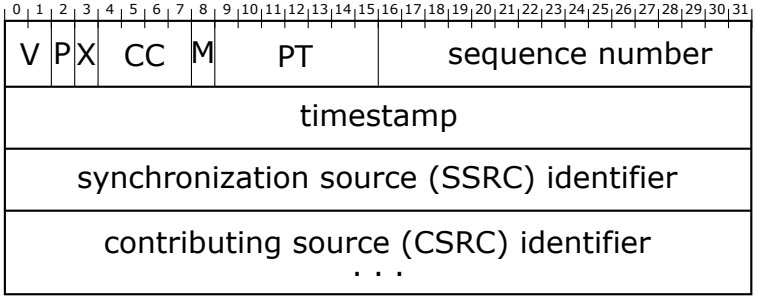
2.3.1 Identifikace zdroje dat

K identifikaci zdroje multimediálních dat se používá jeden ze dvou identifikátorů [18]:

- *Synchronization source (SSRC)*: Každý zdroj RTP paketů je identifikován 32bitovým číslem SSRC, které je obsaženo v každé RTP hlavičce. Identifikátor je součástí RTP sezení. Číslo je zvoleno náhodně a pro každé RTP sezení může být jiné.
- *Contributing source (CSRC)*: Jedná se o identifikátory SSRC, které přispívají k výslednému multimediálnímu proudu dat vytvořeného zařízením zvaným směšovač (popsaný níže).

2.3.2 Formát RTP paketu

V každém RTP paketu je obsažena hlavička s velikostí 12 bajtů. Velikost hlavičky může být větší v závislosti na počtu identifikátorů (CSRC), které jsou přítomny při průchodu více streamů směšovačem, nebo jestli je obsaženo rozšíření, které následuje za fixní hlavičkou. V této části popíši jednotlivé informace obsaženy v RTP hlavičce [18].



Obrázek 2.3: RTP hlavička

- **Verze** (V): Pole identifikuje RTP verzi. Pro specifikaci RFC 3550 obsahuje pole číslo 2.
- **Zarovnání** (P): Jestli je bit nastaven, paket obsahuje na konci další bajty, které nejsou součástí samotných dat, a slouží pro zarovnání velikosti celého paketu. Poslední bajt za multimediálními daty obsahuje velikost zarovnání, které má příjemce ignorovat. Zarovnání je potřebné pro některé šifrovací algoritmy.

- **Rozšíření** (X): Jestli je bit nastaven, paket obsahuje za hlavičkou rozšiřující informace. Formát rozšíření je nad rámec tohoto dokumentu. Mechanismus rozšíření je k dispozici pro nové formáty datových částí, které vyžadují, aby RTP paket obsahoval další informace.
- **Počet identifikátorů CSRC (Contributing source)** (CC): Obsahuje počet identifikátorů CSRC, které následují za fixní RTP hlavičkou.
- **Značka** (M): Interpretace značky závisí na zvoleném profilu multimediálních dat. Například může označovat poslední RTP paket potřebný k sestavení snímku.
- **Formát dat** (PT): Pole identifikuje formát multimediálních dat a určuje jejich interpretaci aplikací. Profil může určit statické mapování typu dat na formát dat. Další typy dat mohou být definovány dynamicky prostřednictvím jiných protokolů, např. SDP. Výchozí mapování pro audio a video je určeno v dokumentu RFC 3551. Během RTP sezení se může typ dat změnit.
- **Pořadové číslo** (sequence number): Pořadové číslo je inkrementováno o jedna pro každý RTP paket. Slouží ke zjištění ztráty paketu nebo nesprávného pořadí přijetí. Počáteční hodnota čísla by měla být náhodná pro zlepšení odolnosti proti útokům.
- **Časové razítko** (timestamp): Hodnota je získána na základě frekvence, která je určena profilem daného multimediálního formátu. Frekvence hodin by měla být dostatečně veliká, aby příjemce mohl bez problémů synchronizovat multimediální obsah a následně vypočítat rozptyl. Počáteční hodnota razítka je určena náhodně kvůli bezpečnosti. Časová razítka z různých multimediálních datových proudů jsou nezávislá a obvykle mají jinou počáteční hodnotu. Pro jejich synchronizaci se využívá RTCP protokol popsaný dále v dokumentu.
- **SSRC** (synchronization source): Pole obsahuje identifikátor zdroje dat, popsaný v předcházející části.
- **CSRC** (contributing source): Seznam identifikující zdroje přispívající do výsledného multimediálního proudu dat. Seznam může být prázdný, nebo obsahovat až 15 identifikátorů.

2.3.3 Systémy v RTP architektuře

U RTP jsou kromě koncových systémů definované také dva typy systémů, které mají speciální využití:

- Translátor – Předává RTP pakety s nezměněnými identifikátory (SSRC), což umožňuje určit zdroj streamu. Některé typy translátorů mohou nechat paket nezměněný, např. pro překonání firewallů. Hlavní účel translátorů je upravit formát multimediálních dat, s tím souvisí i změna profilů a také časové značky v RTP paketu.
- **Směšovač (angl. mixer)** – Přijímá stream RTP datových paketů z jednoho nebo více zdrojů, případně změní formát dat, kombinuje streamy a poté předá kombinovaný stream. Jelikož časování jednotlivých streamů nebude obecně synchronizované, směšovač provede úpravu časování a vygeneruje vlastní časování, proto musí být mixer označen vlastním identifikátorem (SSRC), který je poté obsažen v RTP paketu. Identifikátory původních zdrojů multimediálních dat jsou obsaženy v seznamu identifikátorů (CSRC) v jednolitých RTP paketech.

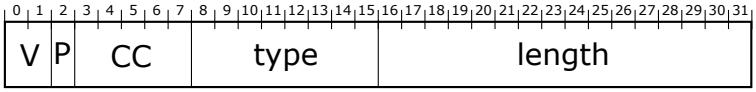
2.4 Real-Time Control Protocol (RTCP)

Protokol RTCP je komplementární k RTP protokolu. Je definován ve stejném RFC 3550 [18] z roku 2003. Protokol je založen na periodickém odesílání kontrolních paketů všem příjemcům, se kterými je ustanoveno sezení. Typicky jsou kontrolní zprávy posílány na stejné transportní vrstvě jako RTP pakety, ale s jiným číslem portu. Protokol má tyto základní funkce [18]:

- Hlavní funkcí je poskytnout zpětnou vazbu o kvalitě distribuce dat. Zpětná vazba může být využita pro adaptivní kvalitu vysílaných médií, ale také pro diagnózu chyb v distribuci multicástových streamů.
- Pakety RTCP v sobě obsahují identifikátor v podobně kanonického jména. Takže i přes změnu SSRC identifikátoru lze určit příjemce.
- Jelikož každý účastník RTP komunikace musí posílat RTCP pakety všem ostatním, má každý nezávisle na sobě přehled o počtu účastníků. Tento počet je využitý pro vyhodnocení, jak často se budou kontrolní pakety posílat.

2.4.1 Formát RTCP paketu

Podobně jako u RTP paketu je taky u RTCP paketu fixní hlavička o velikosti 4 bajty. Každý RTCP paket musí být zarovnán na 32 bitů, a to kvůli možnosti skládání paketů za sebou bez potřeby přidání nějakého dělícího prvku. Za hlavičkou se nachází pevně strukturovaný formát definovaný typem RTCP paketu.



Obrázek 2.4: RTCP hlavička

- **Verze** (V): Pole identifikuje RCTP verzi, která je stejná jako u RTP, tedy 2.
- **Zarovnání** (P): Pole má stejný význam jako u RTP hlavičky.
- **Počet identifikátorů CSRC** (CC): Pole má stejný význam jako u RTP hlavičky.
- **Typ RTCP paketu** (type): Obsahuje číslo typu RTCP paketu.
- **Velikost** (length): Obsahuje velikost RTCP paketu včetně hlavičky a zarovnání, vypočítanou vzorcem viz. 2.1.

$$velikost\_v\_RTCP\_hlavičce = \frac{velikost\_RTCP\_paketu\_v\_bajtech}{4} - 1 \tag{2.1}$$

Typy RTCP paketu

Dokument RFC 3550 definuje následující typy RTCP paketů [18]:

- **Sender report**: Tento typ zprávy posílají pouze stanice, které vysílají data. Zpráva obsahuje statistiky pro všechny poslané RTP pakety. Tyto informace umožní přijímači odhadovat kvalitu přenosu. Jsou to informace jako celkový počet vyslaných paketů a celková velikost poslaných dat. Mezi důležitý obsah paketu také patří aktuální RTP časové razítko a s ním také aktuální čas ve formátu *Network Time Protocol*. Pro server není nutné tento protokol implementovat, stačí jiný zdroj aktuálního času s tím, že bude převeden do patřičného formátu. Tohle mapování RTP časového razítka na NTP formát aktuálního času je nutné kvůli synchronizaci více multimediálních streamů, typicky videa a audia.
- **Receiver report**: Tento typ zpráv posílají příjemci multimediální dat vysílačům. Ve zprávě se nachází informace o kvalitě využívané služby. Mezi tyto informace patří počet ztracených paketů jak celkový, tak mezi vysláním předchozího hlášení, pořadové číslo posledního získaného RTP paketu a informace o kolísání zpoždění (anglicky jitter) u přijímače. Na základě těchto informací může aplikace vysílající multimediální data upravit kvalitu poskytovaného obsahu.
- **Source Description Message**: Ve zprávě tohoto typu se nachází informace o odesílateli, což je každý účastník RTP komunikace. Zpráva může obsahovat tyto druhy informací[18]:
  - **CNAME**: kanonické jméno (například může být zvoleno doménové jméno nebo IP adresa),
  - **NAME**: skutečné jméno,
  - **EMAIL**: emailová adresa,
  - **PHONE**: telefonní číslo,
  - **LOC**: geografická poloha uživatele,
  - **NOTE**: aktuální stav uživatele,
  - **PRIV**: určeno pro rozšíření,
  - **TOOL**: jméno a popřípadě verze aplikace, která odesílá multimediální data,
  - **END**: určuje konec seznamu informací.
- **Goodbye Message**: Účastník komunikace oznamuje ukončení spojení.
- **Application-Specific Message**: Zpráva tohoto typu je určena pro zasílání informací, které nejsou definované ve standardu, a může být využita k experimentálním účelům.

2.5 Session Description Protocol (SDP)

Session Description Protocol slouží k popisu multimediálního sezení před jeho začátkem. Nejaktuálnější verze protokolu číslo 0 je standardizována v RFC 4566 z roku 2006 [6]. Jedná se o textový protokol, který využívá kódování *UTF-8*. Původně protokol spolupracoval pouze s protokolem SAP, ale později našel využití s dalšími protokoly jako jsou RTSP, SIP, HTTP nebo přes e-mail využitím rozšíření *MIME*. Popis v tomto protokolu je označený typem média `application/sdp` [6].

2.5.1 Formát zprávy

Popis protokolem SDP se skládá z jednotlivých řádků textu ve formátu typ=hodnota, kde typ musí být jeden znak a hodnota je strukturovaný text, který závisí na uvedeném typu. Většinou je hodnota buď několik polí oddělených jednou mezerou, nebo volně strukturovaný text. Všechny typy a pole využívají pouze podmnožinu UTF-8 kódování US-ASCII, ale atributy a text mohou využít celou množinu znaků. Typy řádků mají pevně stanovené pořadí kvůli lepší detekci chyb. Některé typy jsou povinné a některé volitelné.

Ukázka popisu v protokolu SDP[6]:

```
v=0
o=jdoe 2890844526 2890842807 IN IP4 10.47.16.5
s=SDP Seminar
i=A Seminar on the session description protocol
u=http://www.example.com/seminars/sdp.pdf
e=j.doe@example.com (Jane Doe)
c=IN IP4 224.2.17.12/127
t=2873397496 2873404696
a=recvonly
m=audio 49170 RTP/AVP 0
m=video 51372 RTP/AVP 99
a=rtpmap:99 h263-1998/90000
```

Následující informace musí být obsaženy v každém popisu protokolem SDP. Uveden je znak, kterým lze tento typ informace zapsat:

- `v`: verze protokolu,
- `o`: informace o vlastníkovi popisu,
- `s`: jméno sezení,
- `t`: doba, po kterou je sezení aktivní.

2.6 Real Time Messaging Protocol (RTMP)

Protokol RTMP byl původně proprietární protokol vyvinutý společností Macromedia. Společnost Macromedia byla koupená firmou Adobe, které vydala specifikace protokolu RTMP ve verzi 1.0 [2]. Jedná se o aplikační protokol navržený pro multiplexování a paketizování multimediálního streamu přes vhodný transportní protokol, jako je TCP. Skrze protokol jsou paralelně přenášeny video, audio a datové zprávy s přidruženými časovými informacemi.

K inicializování, kontrolování a přenášení samotných multimediálních dat je využito pouze jedno spojení. Protokol definuje virtuální kanály, na kterých jsou pakety zasílány a přijímány nezávisle na sobě. Audio, video a kontrolní zprávy mají určené různé virtuální kanály. Jednotlivé pakety jsou rozdělovány do fragmentů, jejichž velikost je dynamicky dohodnuta klientem a serverem. Kontrolní a datové zprávy jsou kódovány formátem Action Message Format (binární formát používaný k serializaci objektů vyvinutý společnosti Adobe). Komunikace je započata tzv. handshakem mezi serverem a klientem.

2.7 H.264

H.264, jiným názvem MPEG-4 Part 10 nebo taky Advanced Video Coding (MPEG-4 AVC), je formát určený pro kompresi videa, který je v současnosti jedním z nejběžněji používaných formátů. První verze byla dokončena v roce 2003 organizacemi ITU-T a ISO/IEC. Standardizována je v dokumentu ISO/IEC 14496-10[1]. Nejnovější revize standardu je z roku 2014. H.264 se obvykle používá pro ztrátovou kompresi. Standard H.264 lze považovat za „rodinu standardů“ složenou z profilů popsaných níže. Dekodér nebo enkodér musí umět pracovat s alespoň jedním profilem, ale ne nutně se všemi.

2.7.1 Profily a úrovně

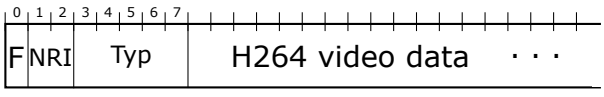
Profily definují, jaké techniky jsou použity kodérem při kompresi videa. Specifikováním profilu se zajišťuje kompatibilita mezi zařízeními, které mají různé dekódovací schopnosti. Profily nabízejí využití formátu H.264 pro různé kategorie využití (např. videokonference, HD televizní vysílání atp.). Jsou označeny číselným kódem. Profilů je v současné době 14. Zde je výpis nejpoužívanějších s jejich typickým využitím:

- `Baseline Profile (66)` je určen primárně pro nenáročné aplikace, nejčastěji se tento profil používá ve videokonferencích a mobilních aplikacích.
- `Main Profile (77)` se využívá se pro televizní digitální vysílání, není ovšem určen pro vysílání ve vysokém rozlišení.
- `Extended Profile (88)` je určený pro streamované video s relativně velkou kompresí a obsahující mechaniky, které zvětší odolnost proti ztrátám.
- `High Profile (100)` je vhodný pro televizní vysílání ve vysokém rozlišení a ukládání na Blue-Ray disky.

Úroveň indikuje stupeň možnosti požadovaného profilu pro dekodér. Úroveň podpory v rámci profilu určuje maximální rozlišení, frekvenci snímku a přenosovou rychlost, kterou dekodér může využít. Dekodér, který zvládá zpracovávat danou úroveň musí být schopen dekódovat všechny videostreamy zákodované danou úrovní a všemi nižšími úrovněmi.

2.7.2 Síťová abstraktní vrstva

Tato vrstva (angl. Network Abstraction Layer, zkráceně NAL) je součástí standardu a hlavním cílem je poskytnutí síťově přátelského řešení reprezentace videodat [22]. Zakódovaná videodata jsou organizovaná do NAL jednotek. První oktet každé NAL jednotky obsahuje hlavičku s popisem, o jaký typ NAL jednotky se jedná. Zbývající bajty NAL jednotky jsou samotná data podle typu definovaném v hlavičce.



Obrázek 2.5: Formát NAL jednotky[1]

- `F` – bit s vždy nulovou hodnotou,
- `NRI` – podpůrné informace pro dekodér,

- `TYP` – kód typu NAL jednotky.

V současné době existuje 19 typů NAL jednotek rozdělených do dvou kategorií (všechny typy jsou definované v tabulce 7-1 v [1]):

- **VCL (Video Coding Layer)** - obsahuje samotné vizuální informace,
- **Non-VCL**: - obsahuje metadata, která mohou nebo nemusí být vyžadována k dekódování videa.

2.7.3 Access Units

Sada NAL jednotek ve specifickém tvaru se označuje jako *Access Units* (zk. AU) [22]. Dekódování každé AU vede k jednomu dekódovanému snímku. Jelikož NAL jednotky neobsahují svou délku, je potřeba nějak rozlišit jednotlivé NAL jednotky. K tomu se využívají různé techniky. Součástí standardu je příloha (angl. Annex), ve které se nachází jeden možný formát, ale není podmínkou ho podporovat. Nejpoužívanější techniky jsou:

- **Annex B** – před každou NAL jednotkou se nachází tyto řetězce '0x000001' nebo '0x00000001' (pozn. Tyto řetězce nejsou přítomny v samotném obsahu NAL jednotek.),
- **AVC** – před každou NAL jednotkou je uložena její velikost. Velikost může být uložena v 1, 2 nebo 4 bajtech. Informace o počtu bajtů, ve kterých je velikost paketu uložena, je přenášena zvlášť s dalšími informacemi o použitém kodéru. Formát je definovaný v ISO/IEC 14496-15.

2.7.4 H264 přenášené v RTP paketu

Pro posílání NAL jednotek skrz RTP protokol je potřeba dodržet formát definovaný v RFC 3984 [21]. RTP protokol dovoluje paketizování jedné nebo více NAL jednotky do RTP paketu

2.7.5 Paketizace H264 NAL jednotek

První bajt za RTP hlavičkou identifikuje, který způsob uložení NAL jednotek byl zvolen.

Obrázek 2.6: Hlavička datové části RTP paketu

- `F` a `NRI` mají stejný význam jako u hlavičky NAL jednotky,
- `TYP` – obsahuje typ NAL jednotky při obsažení jedné celé jednotky nebo typ provedené paketizace. Rozdělení podle kódu:
  - `0` – nedefinováno,
  - `1-23` – typ NAL jednotky,
  - `24` – STAP-A ,

- `25` – STAP-B,
- `26` – MTAP16,
- `27` – MTAP24,
- `28` – FU-A,
- `29` – FU-B,
- `30-31` – nedefinováno.

Existují tři možné způsoby uložení NAL jednotek[21]:

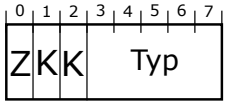
- `Paket s jedinou NAL jednotkou` – obsahuje pouze jednu NAL jednotku. Hlavička NAL je stejná s hlavičkou datové části RTP paketu, jelikož NAL jednotka má definované typy v rozmezí 1-23 (ne všechny jsou v současné době využity).
- `Agregační paket` – používá se k agregaci NAL jednotek do jedné datové části RTP paketu. Tento způsob existuje ve čtyřech verzích: STAP-A, STAP-B, MTAP16, MTAP24.
- `Fragmentace NAL jednotek` – používá se k fragmentaci NAL jednotky přes několik RTP paketů. Existují dvě verze: FU-A a FU-B.

Rozlišují se také dva módy: prokládaný a neprokládaný. Určení módu se provádí v SDP popisu. Při prokládaném módu lze posílat NAL jednotky v pořadí, které není dekódovací, a je tedy nutné toto pořadí uvést v RTP paketu v daném formátu. Pro využití neprokládané je potřeba použít způsob uložení buď jedné NAL jednotky, STAP-A nebo FU-A. Jelikož jsem ve své práci využil pouze způsob FU-A, popíšu jej jako jediný. Nejprve uvedu důvody, proč použít fragmentaci NAL jednotek, proč nestačí posílat pouze paket s jedinou NAL jednotkou:

- NAL jednotky mohou být větší než 65535 bitů, což je maximální velikost IPv4 paketu a často taky bufferu pro síťové rozhraní v operačním systému (při pokusu o zápis většího množství dat může být vyhozena výjimka).
- Při paketu větším než MTU může síťový prvek tento paket zahodit nebo rozdělit do menších částí, čímž vzniká duplikace záhlaví paketů a zbytečné datové přenosy navíc.

Fragmentační jednotka (FU-A)

Jak již bylo zmíněno, jedná se o způsob fragmentace velkých NAL jednotek do několika RTP paketů. Tento způsob je označen kódem 28 a předpokládá výstupní pořadí NAL jednotek stejné jako dekódovací pořadí. Za hlavičkou datové části RTP paketu, která indikuje použití tohoto způsobu, se nachází *Fragmentation Unit* (zk. FU) hlavička.



Obrázek 2.7: FU hlavička[21]

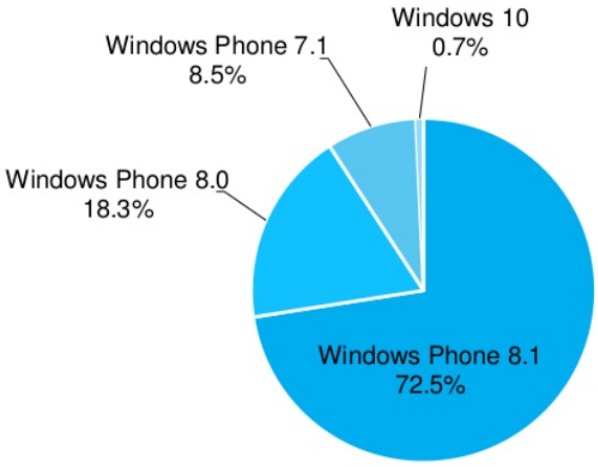
- `S` – když je nastaven na 1, indikuje začátek fragmentované NAL jednotky,

- `K` – když je nastaven na 1, indikuje konec fragmentované NAL jednotky,
- `R` – rezervovaný bit musí být nastaven na 0,
- `TYP` – obsahuje kód typu NAL jednotky.

Kapitola 3

Vývoj aplikací na Windows Phone

Windows Phone 8.1 je v současné době nejnovějším operačním systémem pro mobilní telefony od společnosti Microsoft Corporation. Vydán byl 14. dubna 2014[1] pro vývojáře. Přechod ze starší verze operačního systému 8.0 je možný pro všechny telefony zdarma. Statistika vytvořená společností AdDuplex z 23. dubna 2015 ukazuje největší zastoupení verze 8.1 operačního systému Windows Phone [2].



Obrázek 3.1: Statistika zastoupení verzí operačního systému Windows Phone [7]

Microsoft nabízí 2 sady API pro Windows Phone 8.1:

- **Silverlight** – starší API, které není v plánu v budoucích verzích operačního systému rozšiřovat[3] (některé funkce systému jsou dostupné pouze přes Silverlight API, např. *VoIP*, *Clipboard* nebo *Lens*).
- **Windows Runtime** – novější API, které si klade za cíl sjednotit API napříč Windows a Windows Phone. Lze tedy vyvíjet aplikace s identickým zdrojovým kódem pro obě

[1] http://www.theverge.com/2014/4/14/5612322/windows-phone-8-1-download-features

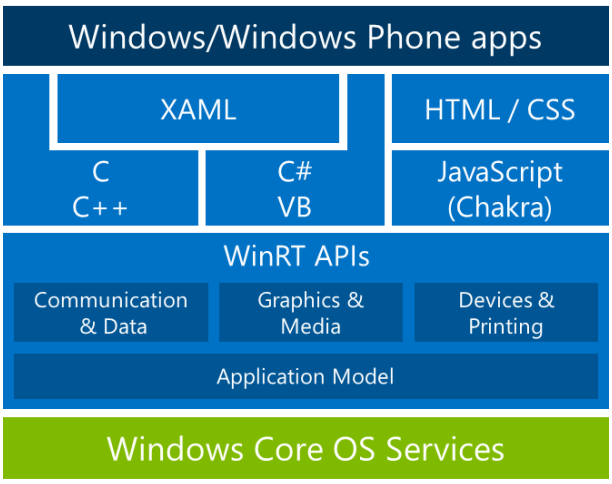
[2] Společnost AdDubplex získala tyto data na základě aplikací, které obsahují AdDuplex SDK sloužící reklamním účelům.

[3] http://blogs.windows.com/buildingapps/2014/12/17/bring-your-windows-phone-silverlight-apps-to-windows-runtime-xaml-prepare-for-universal-app-development-in-windows-10/

platformy. Samozřejmě existují případy nebo podporované funkce, ve kterých menší rozdíly jsou.

3.1 Windows Runtime

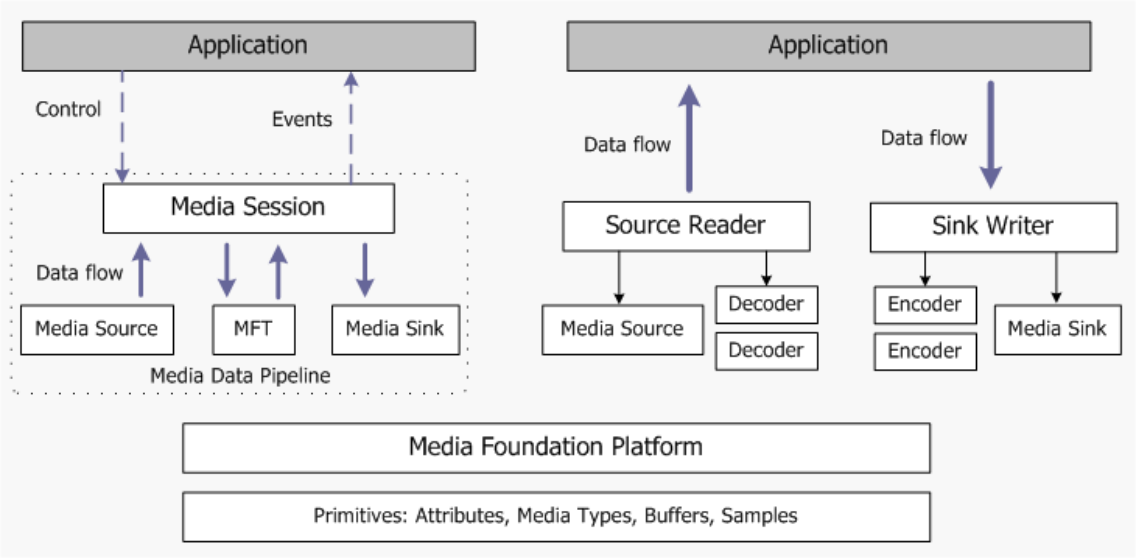
Aplikace vyvíjené pro Windows Runtime (zkráceně WinRT) podporují nativně architektury x86 a ARM. Aplikace běží v tzv. prostředí *sandbox*, které zajišťuje větší bezpečnost a stabilitu pro operační systém. Výhodou WinRT je podpora řady programovacích jazyků. WinRT komponenty jsou navrženy tak, aby dovolily interoperabilitu jazyků (zahrnující nativní C++, skriptovací Javascript a C#). Samotné API je implementované v C++ [8].



Obrázek 3.2: Windows Runtime architektura [3]

3.2 Microsoft Media Foundation

Microsoft Media Foundation (zkráceně MF) je multimediální platforma určená pro systémy Windows od verze Vista, která nahradila DirectShow [12]. MF je založená na COM rozhraní (COM je technologie, která umožňuje komunikaci objektů bez toho, aniž by znaly svoji vnitřní strukturu a detaily, byla vytvořena společností Microsoft).



Obrázek 3.3: Windows Media Foundation architektura [11]

MF poskytuje dva odlišné modely programování. V prvním modelu, na levé straně diagramu, aplikace kontroluje streamování voláním metod. V druhém modelu aplikace čte data ze zdroje (mikrofon, kamera) nebo zapisuje data do komponenty, která může stream přehrát, nebo dělá obojí. Druhý model je užitečný, když je nutné mít přímý přístup k samotným datům. Popis jednotlivých částí modelu [11]:

- *Primitives* – pomocné objekty, které se používají v MF API,
- *Media Foundation Platform* – poskytuje základní funkce využívané v architektuře MF (např. asynchronní volání, speciální fronty pro asynchronní operace na jiném vlákně atd.),
- *Media Pipeline* – skládá se ze tří základních objektů, které mohou být implementovány:
  - *Media Source* – generuje multimediální data ze souboru, síťového streamu nebo zaznamenávacího zařízení,
  - *Media Foundation Transforms* – zpracovává multimediální data a dále je poskytuje (slouží k implementaci např. dekodérů a enkodérů),
  - *Media Sinks* – využívá data k následnému zobrazení, uložení do souboru nebo poslání po síti.
- *Media Session* – řídí tok multimediálních dat a zpracovává úkoly, jako je synchronizace zvuku a videa,
- *Source Reader and Sink Writer* – poskytují alternativní způsob, jak využít základní komponenty Media Foundation a získat data z multimediálního zdroje nebo zapsat data do komponenty media sink. Pro některé aplikace může být vhodnější tento způsob, jelikož nevyžaduje tak velkou znalost MF komponent. Tento model však není vhodný pro použití v časově závislých operacích.

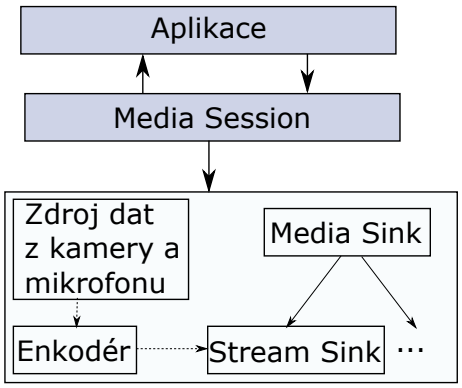
Pro implementování komponent MF pro operační systém Windows Phone je potřeba implementovat COM a Windows Runtime rozhraní. K tomu slouží Windows Runtime C++ Template Library (WRL). WRL je šablona pro knihovnu, která poskytuje nízkoúrovňový přístup, jak vytvářet a používat Windows Runtime komponenty.

3.2.1 Media Sink

Media Sink je komponenta, která přijímá data z jednoho nebo více multimediálních streamů. Media Sink se dělí do dvou kategorií[10]:

- *Renderer* – předkládá data pro přehrání,
- *Archive Sink* – zapisuje data do souboru nebo jiných uložišť.

Hlavní rozdíl mezi nimi je, že Archive Sink zpracovává data tak rychle, jak zvládá. Renderer oproti tomu zpracovává data s pevnou rychlostí přehrávání. Komponentu Media Sink lze implementovat skrz rozhraní `IMFMediaSink`[4]. Každý Media Sink obsahuje jeden nebo více tzv. *Stream Sink*. Každý Stream Sink získává data z právě jednoho multimediálního toku. Stream Sink lze implementovat přes rozhraní `IMFStreamSink`[5]. Obvykle aplikace nevytváří přímo Media Sink. Místo toho aplikace vytvoří tzv. aktivační objekty, které využijí Media Session k vytvoření samotné komponenty Media Sink. Další operace s Media Sink jsou prováděny přes Media Session. Samotná aplikace tedy nevolá žádné metody objektů Media Sink nebo Stream Sink. Stream Sink žádá o multimedální data, když je potřebuje. Aplikace by tedy měla žádat v časovém limitu, aby se zabránilo trhání zvuku nebo obrazu [10].



Obrázek 3.4: Diagram ukazuje, jak putují data (čárkovaná čára) a které objekty ovládají které (plná čára)

3.3 Enkodér H.264

Enkodér H.264 implementovaný v Media Foundation podporuje následující H.264 profily [9]:

- **Baseline Profile**,
- **Main Profile**,

[4] https://msdn.microsoft.com/en-us/library/windows/desktop/ms694262(v=vs.85).aspx
[5] https://msdn.microsoft.com/en-us/library/windows/desktop/ms705657(v=vs.85).aspx

- **High Profile** (vyžaduje Windows 8).

Enkodéru lze nastavit řadu atributů pro H.264 výstupní multimediální tok [9]:

- průměrná přenosová rychlost výstupního toku,
- počet snímků za sekundu,
- šířka a výška snímku v pixelech,
- technika prokládání snímků,
- H.264 profil,
- kvalita výsledného videa (jde o číslo od 0 do 100, přičemž 100 je nejlepší kvalita).

Poslední zmíněný atribut lze nastavovat během enkodování. Změny jsou provedeny ihned. Enkodér generuje Access Unit ve formátu Annex B, který byl dříve popsán.

Kapitola 4

Návrh aplikace pro vysílání videostreamu

Výsledná aplikace pro mobilní telefony má sloužit jako dočasná náhrada IP kamer. Ty slouží k monitorování zejména venkovních prostor. Záběry z kamer mohou být dále zpracovávány v reálném čase a vyhodnocovány. V některých situacích by se dala aplikace použít pro vysílání výjimečných událostí pro větší počet lidí.

Většina domácích sítí disponuje jednou nebo žádnou veřejnou IP adresou. Je tedy potřeba počítat s tím, že na mobilní telefon připojený na domácí Wi-Fi síť nelze zahájit komunikaci odkudkoliv z internetu, jelikož nebude mít přidělenou veřejnou IP adresu. Další problém, který je potřeba vyřešit, je vícenásobný přístup k výslednému mutlimediálnímu toku z kamery. Mobilní telefony nedisponují dostatečně výkonným hardwarem, aby zvládaly přístup mnoha klientů. Nabízí se tedy řešení ve vytvoření "prostředníka" mezi mobilní aplikací a uživateli, kteří chtějí obsah sledovat. Aplikace se při aktivním vysílání zaregistruje u serveru a ten začne přijímat multimediální stream, který následně bude distribuovat klientům přes webový prohlížeč.

4.1 Specifikace požadavků

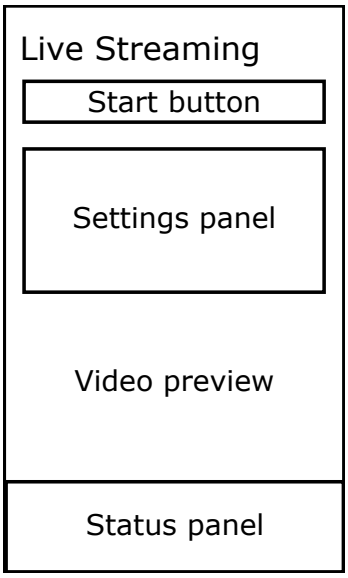
Při specifikaci požadavků bylo vycházeno z předpokladu, že uživateli vytvořené aplikace budou případní zájemci nebo vlastníci IP kamer. Jedná se tedy o běžné uživatele bez větších technických znalostí o tom, jak výsledný stream funguje. S ohledem na tento fakt nebylo zpřístupněno hlubší nastavení kvality výsledného videostreamu.

Název	Vytvoření vysílání	
Popis	Uživatel začne vysílat videostream z kamery mobilního telefonu a zobrazí výsledný stream v prohlížeči	
Vstupní podmínky	Uživatel má stáhnutou aplikaci v telefonu	
Výstupní podímky	Webová stránka zobrazí v přehrávači záběry z kamery mobilního zařízení	
Základní posloupnost	Krok	Činnost
	1	Případ užití začíná zvolením kvality výsledného videostreamu podle kvality internetového připojení
	2	Uživatel zmáčkne tlačítko pro začátek streamování
	3	Aplikace zobrazí pokyny pro sledování výsledného streamu přes webový prohlížeč
	4	Uživatel si podle pokynů zobrazí stream z kamery ve webovém prohlížeči
Poznámky	Výchozí kvalita videa je zvolena a uživatel ji může a nemusí měnit	
	Uživatel má možnost využít kamerové funkce během vysílání k zlepšení kvality streamu	

Tabulka 4.1: Případ užití

4.2 Návrh grafického uživatelského rozhraní

Při vytváření uživatelského rozhraní bylo dbáno na zásady správného postupu tvorby uživatelského rozhraní pro Windows Phone 8.1 definované v dokumentu [1]. Mezi hlavní kritéria rozhraní patří jednoduchost a přehlednost. Grafické rozhraní se skládá z jedné stránky, ve které jsou jednotlivé grafické prvky rozděleny podle návrhu na obrázku 4.1.



Obrázek 4.1: Návrh grafického uživatelského rozhraní aplikace

[1] http://www.microsoft.com/en-us/download/confirmation.aspx?id=30704

Jednotlivé prvky grafického rozhraní jsou:

- `Live Streaming` - název aplikace.
- `Start button` - tlačítko Start zapíná vysílání videostreamu z kamery podle zvoleného nastavení. Během vysílání je změněn text tlačítka na Stop a má opačnou funkci.
- `Settings panel` - panel obsahuje ovládací prvky pro nastavení kvality streamu. V případě probíhajícího streamu lze zobrazit ovládací prvky pro kontrolu kamery.
- `Video preview` - v této pozici je zobrazen náhled kamery pro přesné umístění a zaměření záběru kamery.
- `Status panel` - panel obsahuje informace o aktuálním stavu vysílání, popřípadě jak lze vysílání sledovat.

4.3 Zvolené protokoly

Při vytváření aplikace pro vysílání multimediálního streamu z kamery mobilního telefonu bylo potřeba nejprve rozhodnout, jakým protokolem bude komunikace incializována a poté samotná data přenášena. Jelikož neexistují žádné knihovny, které by vývoj takové aplikace ulehčily, byl zvolen protokol RTSP a s ním spjaté protokoly RTP, RTCP a SDP, a to z následujících důvodů:

- specifikace zvolených protokolů jsou volně dostupné v přívětivé formě,
- protokoly jsou navrženy s ohledem na odlišné potřeby aplikací,
- další z kandidátů, protokol RTMP, nespecifikuje minimální implementaci serveru a při snaze jej implementovat nebylo dosaženo tížených výsledků,
- zvolené protokoly jsou podporovány společností Angelcam a bylo možné je zakomponovat do jejich služeb.

Kapitola 5

Implementace aplikace

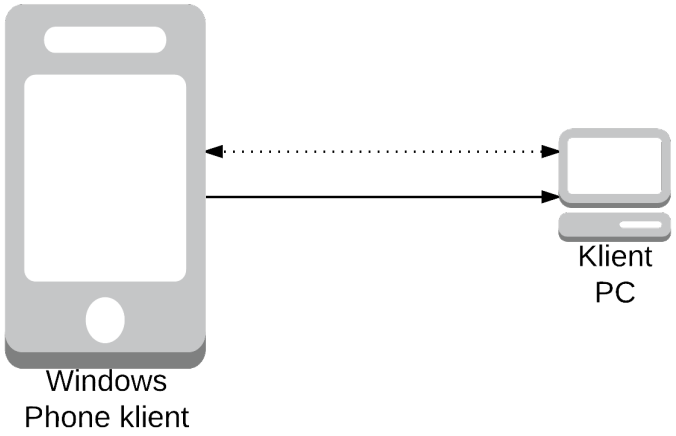
Implementace výsledného serveru je založená na příkladu vytvořeném společností Microsoft Corp. [1] pro demonstraci komunikace v reálném čase. Příklad je také funkční na Windows 8.1, ale jelikož by nemělo využití výsledné aplikace na stolních počítačích smysl, nebyla při vývoji zachována kompatibilita s Windows operačním systémem. Příklad je distribuován pod licencí *Microsoft Limited Public Licence* (zk. MS-LPL), která dovoluje jakékoli úpravy vývojářům softwaru, založeném na Microsoft Windows. Výsledné řešení nabízí tři způsoby použití aplikace:

- přímé připojení k mobilní aplikaci,
- využití serveru pro registraci vysílání,
- využití CDN.

5.1 Přímé připojení k mobilní aplikaci

Nejjednodušší, a také nejproblematičtější z hlediska reálného nasazení je přímé připojení klienta, který chce přehrávat záběry z kamery na mobilní aplikaci. Aplikace podporuje pouze jednoho připojeného klienta zároveň. Bez veřejné IP adresy mobilního telefonu lze tento způsob využít pouze v lokální síti. Tento způsob byl využitý hlavně pro testování aplikace.

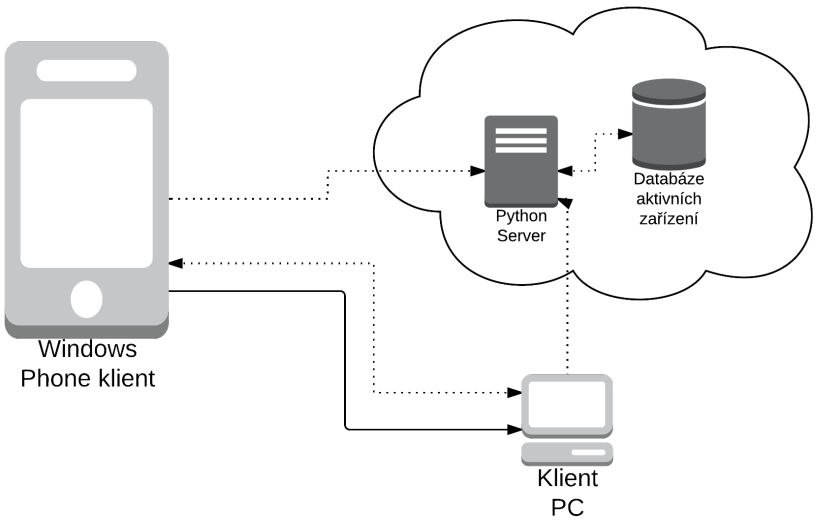
[1] https://code.msdn.microsoft.com/windowsapps/Simple-Communication-Sample-eac73290



Obrázek 5.1: Diagram ukazuje proudění multimediálních dat od mobilního klienta k webovým prohlížečům

5.2 Využití serveru pro registraci vysílání

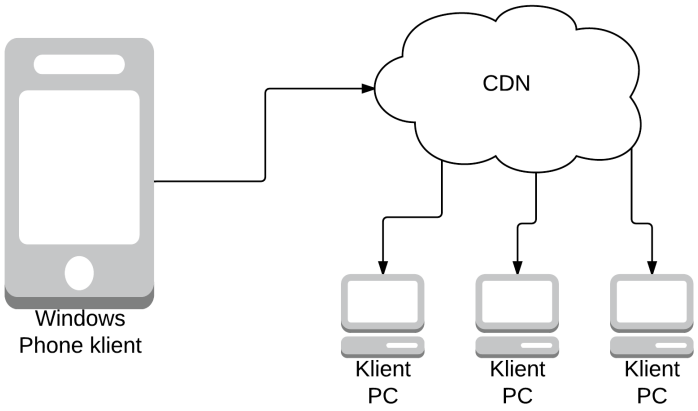
Vylepšení předcházejícího způsobu je ve využití HTTP serveru napsaném v programovacím jazyce Python ve verzi 3.4, který přijímá registrace vysílání od mobilních telefonů. Po úspěšné registraci se v aplikaci zobrazí číselný kód. Pak je potřeba připojit se na server z prohlížeče a zadat vygenerovaný kód. S využitím pluginu do prohlížeče Google Chrome nebo Firefox od společnosti VLC lze po zádání kódu prohlížet vysílání. Plugin se připojuje přímo na mobilní aplikaci prokolem RTSP. Zůstává tedy nevýhoda nutnosti vlastnění veřejné IP adresy nebo připojovat se v lokální síti na mobilní telefon. Server tedy usnadňuje připojení se k multimediálnímu streamu bez nutnosti zadávání IP adresy mobilního telefonu.



Obrázek 5.2: Diagram ukazuje proudění multimediálních dat od mobilního klienta k webovým prohlížečům

5.3 Využití CDN

Nejideálnějším řešením je aplikace s napojením na tzv. CDN (angl. Content Delivery Network – síť serverů, které doručují obsah s vysokými požadavky na přenos a dostupnost klientům). Tento způsob řeší problémy s neveřejnou IP adresou mobilního klienta, protože multimediální stream je dostupný ze serverů, a s nedostatečným výkonem mobilního zařízení, které vysílá pouze jeden stream směrem k serverům. Servery stream zpracovávají a poskytují dál.



Obrázek 5.3: Diagram ukazuje proudění multimediálních dat od mobilního klienta k webovým prohlížečům

5.3.1 Propojení se systémem firmy Angelcam

Společnost Angelcam nabízí služby spojené s IP kamerami. Dovoluje připojení IP kamer k jejich platformě, přes kterou lze sledovat audio-video přenos z kamer odkudkoliv. S přenosem lze dále pracovat přes různá rozšíření, jako je zaznamenávání přenosu, počítání projíždějících aut, počítání lidí apod. Výsledná aplikace se připojuje k serverům této společnosti přes šifrované spojení TLS. K serverům se aplikace připojuje pomocí metody REGISTER protokolu RTSP. K zasílání RTP zpráv je využito TCP spojení prokládané v RTSP spojení.

5.4 Výsledná aplikace

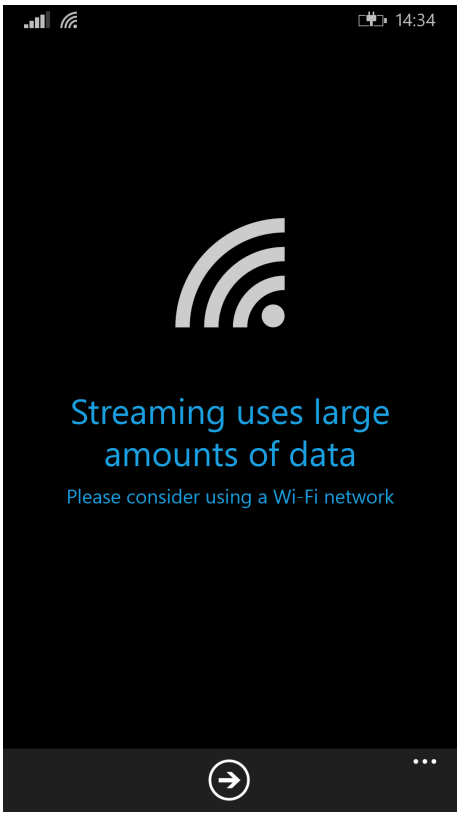
Výsledná aplikace pro mobilní systém Windows Phone 8.1 obsahuje vytvořenou knihovnu `RtspServer.WindowsPhone`, která využívá COM rozhraní pro komunikaci s Media Foundation objekty a také využívá Windows Runtime rozhraní pro interakci se samotnou Windows Store aplikací. Knihovna implementuje RTSP server, který splňuje minimální požadavky na server zmíněné v RFC 2326 sekci D.2. Multimediální data jsou odesílána skrze protokol RTP spolu s kontrolními informacemi protokolu RTCP. K popisu multimediálních dat se využívá popsaný protokol SDP. Server podporuje zasílání dat skrze TCP a UDP protokolu. Posílání dat může probíhat ve dvou režimech při použití TCP: prokládaný (data putují po stejném TCP spojení jako komunikace RTSP protokolem) a samostatný. Rozdílnost transportních protokolů ve výsledném použití bude rozebrána v kapitole o testování. RTSP server zvládá obsluhovat pouze jednoho klienta zároveň. V případě využití serverů

pro zpracování multimediálního streamu z aplikace a poskytnutí dalším uživatelům, je toto řešení dostačující.

Samotná aplikace, která obsahuje referenci na knihovnu, je napsaná v jazyce C# a umožňuje nastavení enkodérů. S tím souvisí zvolení kvality videa a aktivování audiostreamu. Videostream je enkodovaný H.264 formátem a audio je zasíláno ve formátu AAC.

5.4.1 Demonstrace uživatelského rozhraní

Rozhraní bylo vytvořeno prostřednictvím jazyka XAML ve vývojovém prostředí Visual Studio. Při prvním spuštění aplikace jsou zobrazena upozornění, na která by měl uživatel dávat pozor při používání aplikace. Upozornění obsahují textový popis a ikonu. Byly využity ikony z volně dostupného balíčku [2], které zachovávají vzhled základních prvků používaných v systému.



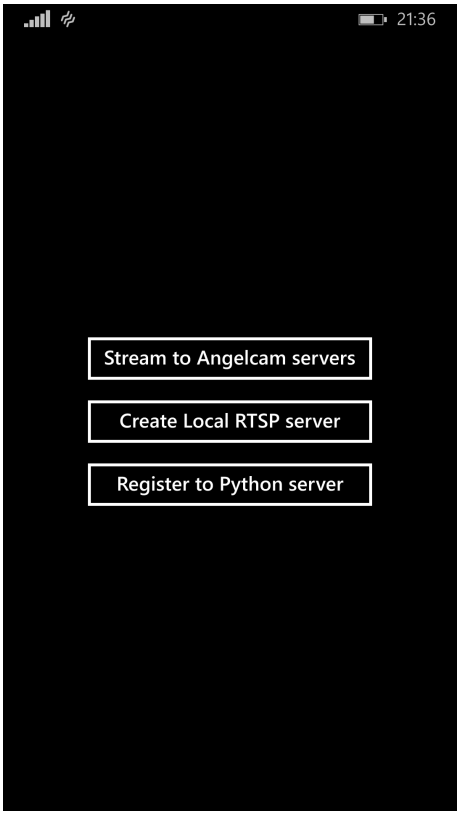
Obrázek 5.4: Jedno ze tří upozornění zobrazené při prvním startu aplikace

Jako logo aplikace byla použita ikona vyobrazující IP kameru. Ikona je šířená pod licencí Public Domain a je dostupná ke stažení skrze tento odkaz [3].

[2] http://modernuiicons.com/
[3] https://thenounproject.com/term/security-camera/2457/

Obrázek 5.5: Logo aplikace zobrazené na tzv. dlaždici

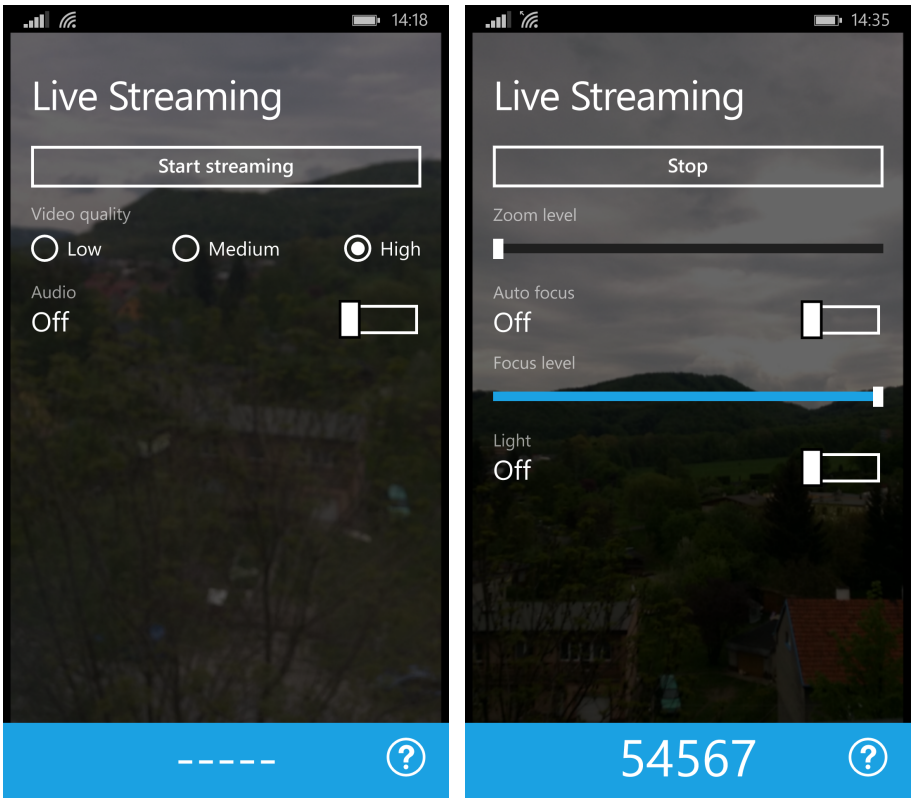
Na obrázku 5.6 lze vidět rozcestník aplikace, který se zobrazí jako první stránka při klasickém startu aplikace. Zvolením jedné ze tří možností je uživatel přesměrován na hlavní obrazovku. Pro všechny tři možnosti vypadá výsledná stránka stejně. V případě publikování aplikace by byl vybrán pouze jeden způsob fungování a stránka by nebyla zobrazena.



Obrázek 5.6: Stránka sloužící jako rozcestník pro tři různé způsoby použití aplikace

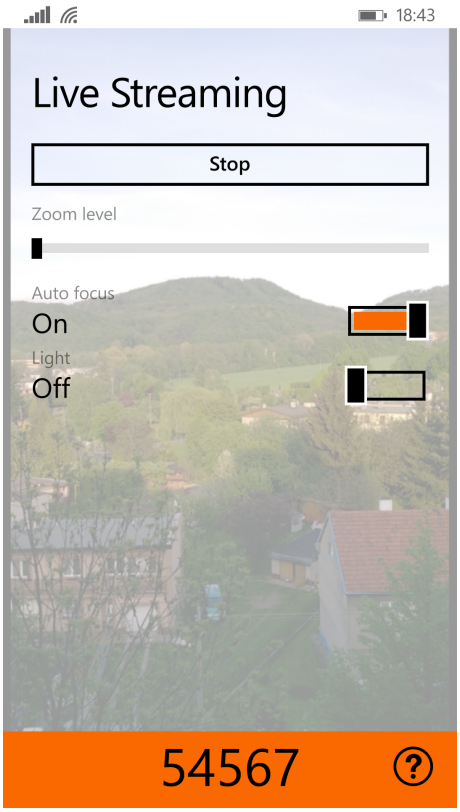
Hlavní a jediná obrazovka pracuje ve dvou režimech. V prvním režimu se nachází před začátkem streamování. Lze zde nastavit kvalitu výsledného videostreamu a případně zapnout nebo vypnout nahrávání zvuku. Kvalita videa je reprezentována třemi stupni: vysoká, střední a nízká. Po zmáčknutí tlačítka “Start streaming” se animací nahradí prvky ovládající formát výstupního streamu prvky ovládající kamerové zařízení, které reprezentují druhý režim. V druhém režimu lze nastavit digitální přiblížení kamerového výstupu, automatické zaostřování nebo zapnutí přisvětlovacího světla na telefonu. V případě, kdy telefon nedisponuje zmíněnou funkcionalitou, nejsou jednotlivé ovládací prvky zobrazeny. Ve spodním informačním panelu se nachází kód, který slouží pro přístup k videostreamu přes

prohlížeč. Napravo od kódu je tlačítko pro nápovědu. Po zmáčknutí se vysune menší textová nápověda ze spodního okraje. Na pozadí v obou režimech je zobrazen náhled kamery.



Obrázek 5.7: Vlevo na obrázku je zobrazený režim nastavování výstupního streamu. Vpravo lze vidět režim ovládání kamerového zařízení při probíhajícím vysílání dat. Na pozadí aplikace se zobrazuje náhled snímaného záběru

Systém Windows Phone obsahuje unikátní funkci, kdy si uživatel může zvolit pozadí systému (bílé nebo černé) a ladění prvků do jedné z šestnácti přednastavených barev. Aplikace by měly tato nastavení respektovat. Na obrázku 5.8 je zobrazena aplikace při rozdílném nastavení systému.



Obrázek 5.8: Aplikace při zvolené bílé barvě pozadí a oranžové barvě pro ladění prvků

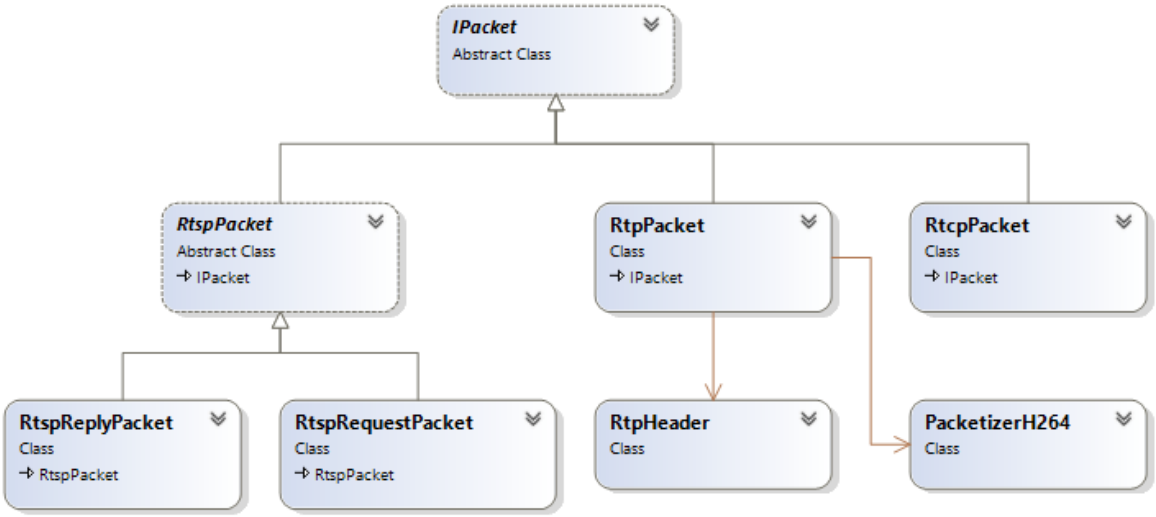
5.5 Popis implementace knihovny

Knihovna umožňuje uskutečnit spojení s klientem, dohodnout se na podmínkách vysílání a začít vysílat multimediální stream. Knihovna je napsaná v jazyce C++ a využívá Windows Runtime C++ Template Library k přístupu ke COM a WinRT rozhraní. Následuje popis důležitých tříd, který neobsahuje vytvořené abstraktní třídy ani rozhraní:

- `CMediaSink` – implementuje rozhraní `IMFMediaSink`[4]. Třída má na starost vytváření tříd `CStreamSink` a jejich zinicializování. Třída obsahuje veškerou logiku pro zpracování a zaslání zpráv RTSP protokolem. Také má na starost ovládání prezentačního času, které slouží pro synchronizaci více mutlimediálních streamů.
- `CStreamSink` – implementuje rozhraní `IMFStreamSink`[5]. Každá instance této třídy reprezentuje jeden multimediální stream (video nebo audio). Metoda `ProcessSample` obsahuje jeden argument typu `IMFSample`, což je kontejner pro samotná mediální data, a je volána z Media Foundation komponent generující data. Třída se tedy stará o zasílání jednotlivých RTP paketů obsahujících multimediální data a také o zasílání kontrolních RTCP paketů.
- `NetworkServer` – třída se stará o vytvoření a inicializování síťového socketu, který čeká na spojení na určitém portu.

[4] https://msdn.microsoft.com/en-us/library/windows/desktop/ms694262(v=vs.85).aspx
[5] https://msdn.microsoft.com/en-us/library/windows/desktop/ms705657(v=vs.85).aspx

- `NetworkTcpClient` – třída vytváří socket, pomocí kterého se připojí přes TCP protokol k zadané adrese, a využívá šifrované spojení TLS1.2.
- `NetworkUdpClient` – třída vytváří socket, který se napojí na lokální port a připojí se na vzdálenou adresu.
- `NetworkChannel` – třída se stará o zasílání a přijímání síťových zpráv využívajících TCP nebo UDP protokol.
- `SessionDescription` – obsahuje jednotlivé informace spojené se SDP. Informace jsou uložené v seznamu objektů, které implementují rozhraní `ISessionDescriptionLines`.
- `SessionDescriptionLines` – slouží k vytvoření základních informací, které musí SDP zpráva obsahovat a pro které neexistuje speciální třída. Implementuje rozhraní `ISessionDescriptionLines`.
- `MediaDescription` – slouží k vytvoření popisu média ve formátu SDP. Implementuje rozhraní `ISessionDescriptionLines`.
- `RtspPacketRequest` – vytváří a zpracovává dotazovací zprávy protokolu RTSP. Třída umí vytvořit pouze dotaz typu `REGISTER`.
- `RtspPacketReply` – vytváří a zpracovává zprávy, které slouží jako odpověď na RTSP dotazovací zprávy.
- `RtpPacket` – přijímá multimediální data a přidává k nim RTP hlavičku. K tomu využívá třídu `RtpHeader`.
- `RtpHeader` – vytváří fixní hlavičku pro RTP paket.
- `RtpHeader` – vytváří kontrolní RTCP pakety typ SenderReport, SourceDescription a Goodbye.
- `PacketizerH264` – zpracovává mediální data ve formátu H264 a rozděluje je do paketů pomocí metody FU-A, aby nepřesahovala MTU ethernetové sítě.



Obrázek 5.9: Diagram tříd, které vytvářejí a zpracovávají jednotlivé pakety protokolů RTSP, RTP, RTCP

5.6 Testování

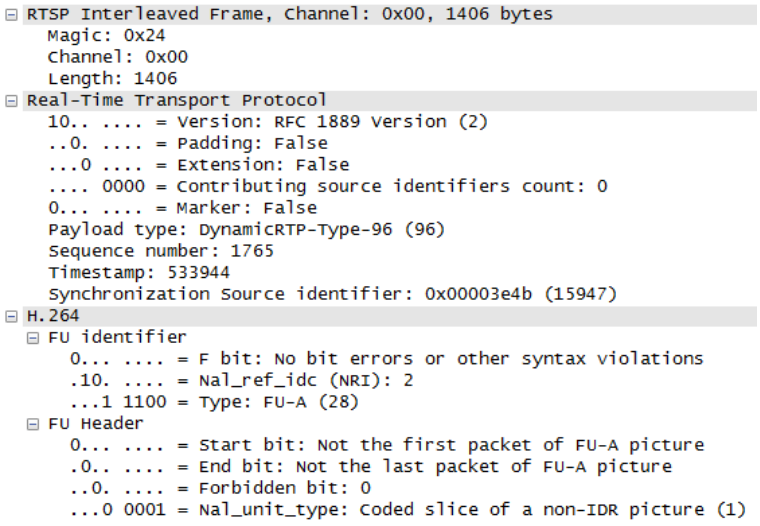
Aplikace byla testována na dvou zařízeních se systémem Windows Phone 8.1, konkrétně na zařízení Nokia Lumia 930 a Nokia Lumia 820 s následujícími specifikacemi:

- `Lumia 930` – chipset Qualcomm MSM8974 Snapdragon 800, procesor Quad-core 2.2 GHz, ram paměť 2GB;
- `Lumia 820` – chipset Qualcomm MSM8960 Snapdragon, procesor Dual-core 1.5 GHz, ram paměť 1GB.

Výkonnější telefon Lumia 930 zvládal bez problémů všechny nabízené kvality videa. Méně výkonný telefon Lumia 820 nezvládal plynule streamovat nejvyšší nabízenou kvalitu s následujícími parametry enkodéru videa:

- `rozlišení v pixelech` – 1280 x 720,
- `počet snímků za sekundu` – 30,
- `průměrný bit rate` – 9000000 b/s,
- `H.264 profil` – Main (77).

Pro záznam síťové komunikace a její následný rozbor byl použit síťový analyzátor Wireshark [23]. Na následujícím obrázku lze vidět případ analyzovaného RTSP paketu, ve kterém se nachází RTP paket obsahující H.264 NAL jednotku paketizovanou způsobem FU-A.



Obrázek 5.10: Analyzovaný RTSP paket programem Wireshark

Pro přijímání videostreamu v lokální síti byl použit program VLC Media Player [6] ve verzi 2.2.0 (VLC je multiplatformní přehrávač multimédií vyvíjený pod Open Source licencí) běžící pod systémem Windows 8.1. V nastavení programu lze zvolit preferovaný způsob zasílání RTP paketů. Jako výchozí je zvolený protokol UDP. Při testování byla ponechána výchozí velikost mezipaměti pro multimediální data 1 sekunda.

Při používání mobilního zařízení jako zdroje vysílání multimediálních dat nelze vždy počítat s kvalitním Wi-Fi signálem nebo kvalitním a rychlým internetovým připojením. Proto bylo zvoleno testování výsledné aplikace při zhoršených podmínkách. Pro simulování podmínek byl využitý program Clumsy ve verzi 0.2 [20]. Program dovoluje vybrat příchozí nebo odchozí pakety a provést s nimi jednu z následujících možností: zahození, zpoždění, duplikování apod. V nespolehlivých paketových sítích se může stávat, že paket nedorazí, nejčastěji z důvodu zajištění kvality služeb od poskytovatelů internetu.

Pro zhodnocení kvality výsledného videostreamu za zhoršených podmínek simulovanými náhodným zahazováním paketů byla zvolena stupnice od 1 do 5 s tím, že 5 je nejlepší kvalita. Testování bylo provedeno s využitím transportních protokolů UDP i TCP na telefonu Lumia 930. Výsledek testování má ukázat jejich rozdílné vlastnosti a dopad na videostream. Streamování a následné zhodnocení kvality videostreamu probíhalo ve 30sekundových intervalech. Po uskutečněném jednom kole následovalo další pro zmírnění dopadu nahodilých jevů v paketové síti. Výsledné hodnoty byly aritmeticky zprůměrovány.

[6] http://www.videolan.org/vlc/

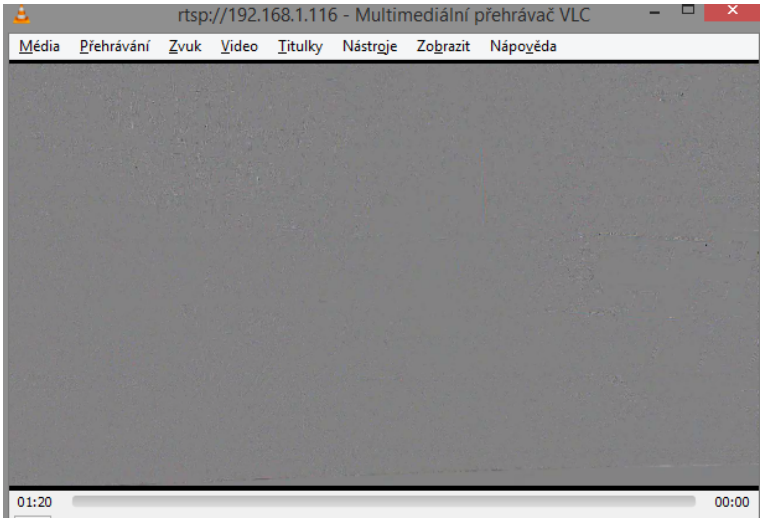
Počet zahozených paketů	hodnocení při využití TCP	hodnocení při využití UDP
0 %	5	5
0.2 %	5	4
0.4 %	4.667	3.334
0.6 %	4.667	2
0.8 %	5	1.334
1.0 %	4.667	1.334
2 %	4.334	0.334
3 %	2.667	0
4 %	2	0
5 %	1.334	0
10 %	0	0

Tabulka 5.1: Výsledky testování videostreamu z aplikace za simulovaných zhoršených síťových podmínek

Z výsledků testů lze vyvodit, že videostream přenášený protokolem TCP si lépe poradí se zahozenými pakety (což reflektuje vlastnosti tohoto protokolu) až do 5 %. Poté je kvalita videostreamu u obou protokolů velice nízká. V případě, kdy je důležitá aktuálnost videostreamu, dosahuje protokol UDP lepších výsledků. Protokol udržuje konstantní odezvu mezi posílaným videostreamem a přehráváním. V případě protokolu TCP se odezva zvětšuje s postupem času kvůli zvětšování vstupního bufferu přehrávače. Kromě klasického zasekávání videostreamu byly vypozorovány dva specifické úkazy. U protokolu UDP to byly artefakty 5.11, které vznikají snahou kodeku H.264 kompenzovat ztracené snímky.

Obrázek 5.11: Artefakty objevující se při zahozených paketech přenášejících video přes UDP

Při využití protokolu TCP ve zhoršených síťových podmínkach se jednou za čas objevila šedá obrazovka 5.12, která vznikne včasným nepřijetím *klíčových snímků* (angl. key frame), a přehrávač tedy nemá, co by zobrazil.



Obrázek 5.12: Šedá obrazovka způsobená chybějícím snímkem

Kapitola 6

Závěr

V bakalářské práci byly analyzované síťové a multimediální protokoly potřebné k realizaci aplikace pro vysílání multimediálních dat. Stručně byl popsán vývoj aplikací pracujících s multimediálními daty pro systém Windows Phone 8.1. Při návrhu aplikace byl zvolen programovací jazyk C# a pro implementaci uživatelského prostředí jazyk XAML. Jelikož neexistuje knihovna pro účely vysílání videa z kamery, byla implementována knihovna s omezenou funkcionalitou. Pro vývoj knihovny, která zpracovává samotný multimediální přenos a komunikaci se vzdáleným klientem, bylo využito šablony WRL.

Tvorba výsledné aplikace byla rozdělena do třech částí. V první části byla vytvořena aplikace komunikující s VLC Media Player využívající transportního protokolu UDP. V následující části bylo totéž uskutečněno přes TCP spojení. V další části byla komunikace uskutečněna s již existujícím řešením firmy Angelcam, kdy komunikace byla inicializována aplikací přes šifrované spojení. V poslední části bylo navrženo a následně implementováno uživatelské rozhraní.

Testování vytvořené aplikace odhalilo velkou náročnost na systémové prostředky mobilního zařízení. Při testování streamování ve zhoršených síťových podmínkách se projevily rozdílné vlastnosti TCP a UDP spojení. Při nestabilních podmínkách paketové sítě se dosahuje použitím TCP spojení lepších výsledků.

Cíle bakalářské práce, které jsou uvedeny v zadání práce, byly splněny. Vytvořená aplikace vysílá multimediální tok z digitální kamery mobilního zařízení ke vzdálenému klientovi. Přenášeno může být video ve zvolené kvalitě a také audio z mikrofonu. Před publikováním aplikace by bylo potřeba otestovat aplikaci na větším množství mobilních telefonů, jelikož aplikace pracuje s mnoha hardwarově specifickými funkcemi. Pro nasazení aplikace v reálném prostředí by bylo vhodné implementovat dynamickou kvalitu multimediálního toku podle kapacity počítačové sítě. K tomu je možné využít kontrolních paketů protokolu RTCP o probíhající komunikaci. Dalším vhodným rozšířením by bylo přidání tzv. *odborného módu*, kde by zkušený uživatel mohl podrobněji nastavit výslednou kvalitu videa. IP kamery také často zobrazují aktuální čas na výsledném videu, což by taktéž mohlo být přidáno do aplikace.

Literatura

[1] Information technology – Coding of audio-visual objects – Part 10: Advanced Video Coding. ISO ISO/IEC 14496-10:2014, 2014.

[2] Adobe Systems Incorporated: Real-Time Messaging Protocol (RTMP) specification [online]. http://www.adobe.com/devnet/rtmp.html, 2012 [cit. 2015-4-28].

[3] Bryntesson, P.: Windows Phone 8.1 for Developers-Converging the App Models [online]. http://blogs.msdn.com/b/thunbrynt/archive/2014/03/31/windows-phone-8-1-for-developers-converging-the-app-models.aspx, 2014 [cit. 2015-4-27].

[4] Finlayson, R.: Notifying RTSP Clients and Proxy Servers About Streams: The RTSP „REGISTER“ Command [online]. https://tools.ietf.org/html/draft-finlayson-rtsp-register-command-00, 2013 [cit. 2015-4-14].

[5] Finlayson, R.: LIVE555.COM [online]. http://www.live555.com/, 2015 [cit. 2015-4-15].

[6] Handley, M.; Jacobson, V.; Perkins, C.: SDP: Session Description Protocol [online]. http://tools.ietf.org/html/rfc4566, 2003 [cit. 2015-4-24].

[7] Juozaityte, L.; Lacey, M.; Mendelevich, A.: AdDuplex Windows Phone Device Statistics Report for April, 2015 [online]. http://blog.adduplex.com/2015/04/adduplex-windows-phone-device.html, 2015 [cit. 2015-4-27].

[8] Microsoft Corporation: Windows Phone API reference [online]. https://msdn.microsoft.com/library/windows/apps/ff626516.aspx, 2014 [cit. 2015-4-27].

[9] Microsoft Corporation: H.264 Video Encoder [online]. https://msdn.microsoft.com/en-us/library/windows/desktop/dd797816.aspx, 2015 [cit. 2015-4-27].

[10] Microsoft Corporation: Media Sinks [online]. https://msdn.microsoft.com/en-us/library/windows/desktop/ms701626.aspx, 2015 [cit. 2015-4-27].

[11] Microsoft Corporation: Overview of the Media Foundation Architecture [online]. https://msdn.microsoft.com/en-us/library/windows/desktop/ff819455.aspx, 2015 [cit. 2015-4-27].

[12] Microsoft Corporation: What's New for Media Foundation [online]. https://msdn.microsoft.com/en-us/library/windows/desktop/bb970511.aspx, 2015 [cit. 2015-4-27].

[13] Peterka, J.: Síťový model TCP/IP [online]. http://www.earchiv.cz/a92/a231c110.php3/, 1992 [cit. 2015-4-21].

[14] Peterka, J.: Protokol TCP - I. [online]. http://www.earchiv.cz/a93/a305c110.php3, 2011 [cit. 2015-4-21].

[15] Peterka, J.: Protokol UDP [online]. http://www.earchiv.cz/a93/a303c110.php3, 2011 [cit. 2015-4-21].

[16] Peterka, J.: Rodina protokolů TCP/IP, verze 3.0 [online]. http://www.earchiv.cz/l225/nahled.php3, 2013 [cit. 2015-4-21].

[17] Reynolds, J. K.; Postel, J.: ASSIGNED NUMBERS [online]. http://ietf.org/rfc/rfc1340.txt, 1992 [cit. 2015-4-21].

[18] Schulzrinne, H.; Casner, S. L.; Frederick, R.; aj.: RTP: A Transport Protocol for Real-Time Applications [online]. https://tools.ietf.org/html/rfc3550, 2003 [cit. 2015-4-22].

[19] Schulzrinne, H.; Rao, A.; Lanphier, R.: Real Time Streaming Protocol (RTSP) [online]. http://www.ietf.org/rfc/rfc2326.txt, 1998 [cit. 2015-4-14].

[20] Tao, C.: clumsy [online]. http://jagt.github.io/clumsy/index.html, 2015 [cit. 2015-5-02].

[21] Wang, Y.-K.; Even, R.; Kristensen, T.; aj.: RTP Payload Format for H.264 Videol [online]. https://tools.ietf.org/html/rfc6184, 2011 [cit. 2015-4-24].

[22] Wiegand, T.; Sullivan, G. J.; Bjontegaard, G.; aj.: Overview of the H.264/AVC video coding standard. *Circuits and Systems for Video Technology, IEEE Transactions on*, ročník 13, 2003 [cit. 2015-4-26].

[23] Wireshark Foundation: User Documentation [online]. https://www.wireshark.org/docs/, 2015 [cit. 2015-5-02].

Dodatek A

Instalační příručka

A.1 Obsah přiloženého média

- `/application/binary` – appx balíček výsledné aplikace pro architekturu procesorů ARM
- `/application/source` – obsahuje projekt pro vývojové prostředí Visual Studio 2013
- `/application/registerServer` – obsahuje HTTP server v Pythonu pro registraci vysílání
- `/document/doc` – dokument v elektronické podobě
- `/document/poster` – plakát
- `/document/video` – prezentační video

A.2 Instalace aplikace

- 1. způsob – Před nainstalováním aplikace na mobilní telefon nebo emulátor s operačním systémem Windows Phone 8.1 je potřeba mít nainstalované Visual Studio 2013 Update 2 nebo novější verzi. Emulátor systému lze stáhnout z následujícího odkazu [1]. Přes aplikace `Windows Phone Application Deployment (8.1)` lze zvolit přiložený appx balíček a nahrát do mobilního telefonu nebo emulátoru.
- 2. způsob – Potřeba je otevřít projekt ve vývojovém prostředí Visual Studio 2013 Update 2 nebo novějším. Vybrat zařízení nebo emulátor a zvolit `Deploy Solution` nebo `Start Debugging`. V případě mobilního zařízení je před nahráním projektu nutné odemknout telefon pro debugování nástrojem `Windows Phone Developer Registration (8.1)`. Pro použití nástroje je potřeba se přihlásit Microsoft účtem s vývojářskou licencí.

  Pozn.: Pro změnu IP adres vzdálených serverů je potřeba upravit řetězce v souboru "ServerAdress.cs", který se nachází v kořenovém adresáři projektu aplikace. Následně je nutné projekt zkompilovat.

[1] https://dev.windows.com/en-us/develop/download-phone-sdk

A.3 Spuštění registračního serveru

Pro spuštění serveru je nutné mít nainstalovaný interpret jazyka Python nejníž ve verzi 3.3.5. Program má jeden nepovinný parametr číslo portu, na kterém HTTP server bude naslouchat. Při nezvolení čísla portu je výchozí číslo 8080.