VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

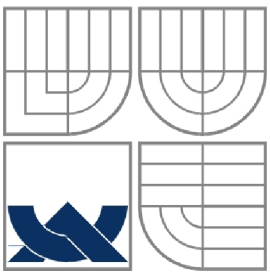
VYUŽITÍ PROTOKOLU RTP PRO DISTRIBUCI PROCESNÍCH DAT V REÁLNÉM ČASE

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

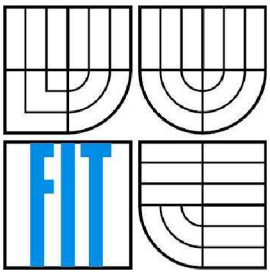
AUTOR PRÁCE
AUTHOR

BC. TOMÁŠ ŠKARECKÝ

BRNO 2010



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

VYUŽITÍ PROTOKOLU RTP PRO DISTRIBUCI PROCESNÍCH DAT V REÁLNÉM ČASE

USING OF RTP PROTOCOL FOR REAL-TIME PROCESS DATA DISTRIBUTION

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

BC TOMÁŠ ŠKARECKÝ

VEDOUCÍ PRÁCE
SUPERVISOR

DOC. DR. ING. PETR HANÁČEK

BRNO 2010

Zadání diplomové práce/10684/2009/xskare01

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav inteligentních systémů Akademický rok 2009/2010

Zadání diplomové práce

Řešitel: **Škarecký Tomáš, Bc.**

Obor: Informační systémy

Téma: **Využití protokolu RTP pro distribuci procesních dat v reálném čase**
RTP Protocol for Real-Time Data Distribution

Kategorie: Počítačové sítě

Pokyny:

1. Nastudujte vlastnosti rodiny protokolů RTP, možné způsoby jejich rozšíření a vlastnosti volně dostupných implementací protokolu.
2. Vytvořte návrh rozšíření protokolu o tok procesních dat.
3. Implementujte prototyp serveru a klienta využívající navržené rozšíření.
4. Srovnejte vlastnosti navrženého protokolu s protokoly specializovanými na sběr a distribuci procesních dat.
5. Zhodnoťte celkový přínos a možnosti začlenění navrženého rozšíření do stávajících produktů.

Literatura:

- Dle pokynů vedoucího

Při obhajobě semestrální části diplomového projektu je požadováno:

- Body 1 - 3

Podrobné závazné pokyny pro vypracování diplomové práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva diplomové práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap, které byly vyřešeny v rámci ročníkového a semestrálního projektu (30 až 40% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Hanáček Petr, doc. Dr. Ing.**, UITS FIT VUT

Datum zadání: 21. září 2009

Datum odevzdání: 26. května 2010

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav inteligentních systémů
612 66 Brno, Božetěchova 2

L.S.

doc. Dr. Ing. Petr Hanáček
vedoucí ústavu

Abstrakt

Tato práce zkoumá možnost přenosu procesních dat v reálném čase pomocí protokolu RTP. Nejdříve jsou zhodnoceny existující protokoly, které jsou k tomuto účelu používané nebo teoreticky použitelné. Dále je uveden popis trojice protokolů RTP, RTCP a RTSP, vysvětlena jejich funkce a zkoumány možnosti jejich rozšiřování. Na základě takto získaných znalostí jsou identifikovány problémy, které se při přenosu procesních dat pomocí RTP mohou vyskytnout a jsou navrženy možnosti jejich řešení.
Pro demonstraci distribuce dat pomocí RTP je navržena a implementována aplikace, která shromažďuje GPS pozice ze zařízení PDA a následně je poskytuje klientům, kteří je zobrazují na mapovém podkladu.

Abstract

This paper explores the potential of process data distribution in real time using the RTP protocol.
Firstly, there is an evaluation of existing protocols that are used for this purpose. Secondly, there is a description of the trio of protocols RTP, RTCP, and RTSP, explained their functions and explored possibilities for their expansion. On the basis of this acquired knowledge, problems which may occur with the distribution of process data by means of RTP are identified and their possible solutions are proposed.
To demonstrate the data distribution via RTP, the application, that collects GPS position from the PDA and then provides clients with them, has been designed and implemented. Clients can display these data on the map.

Klíčová slova

RTP protokol, procesní data, přenos procesních dat, proudový přenos dat, GPS.

Keywords

RTP protocol, process data, process data transfer, data streaming, GPS.

Citace

Škarecký Tomáš: Využití protokolu RTP pro distribuci procesních dat v reálném čase, diplomová práce, Brno, FIT VUT v Brně, 2010

Využití protokolu RTP pro distribuci procesních dat v reálném čase

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením Doc. Dr. Ing. Petra Hanáčka.
Další informace mi poskytli Ing. Miroslav Fitz a Ing. Rostislav Fitz.
Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Tomáš Škarecký
18.5.2010

Poděkování

Na tomto místě bych rád poděkoval Doc. Dr. Ing. Petru Hanáčkovi za vedení diplomové práce a cenné připomínky. Dále pak Ing. Miroslavu Fitzovi a Ing. Rostislavu Fitzovi za poskytnuté podnětné konzultace.

Obsah

1 Úvod

Sběr a distribuce procesních dat jsou součástí velkého množství systémů nejrůznějšího zaměření. Slouží k automatizovanému ovládání systémů, ke generování statistik, ke kontrole správného chodu systému a k mnoha jiným činnostem.

Pro přenos těchto dat je již navrženo velké množství specializovaných protokolů, jak otevřených standardů, které jsou podporované mnoha výrobci, tak uzavřených protokolů jednotlivých výrobců, které slouží k přenosu jen mezi jeho zařízeními. I přes existenci těchto specializovaných protokolů je možné na procesní data pohlížet jako na proud dat poskytovaný svým zdrojem a tak využít protokoly, které jsou určeny pro přenos takových proudů. Protokol RTP je dnes využíván především pro distribuci multimediálních dat například ve video konferencích. Tato práce zkoumá využití a možné přínosy tohoto protokolu při distribuci procesních dat.

Kapitola číslo dvě poskytuje přehled dvou skupin protokolů. První skupina obsahuje protokoly, které jsou pro přenos procesních dat používány. Druhou skupinu pak tvoří protokoly, které by bylo možné pro přenos procesních dat teoreticky použít. Jsou zde nastíněny jejich výhody a nevýhody. Na začátku kapitoly číslo 3 je zdůvodněna volba protokolu RTP, RTCP a RTSP a vymezeny oblasti jejich použití při přenosu. Dále jsou pak detailně popsány vlastnosti těchto protokolů a možnosti jejich rozšíření. Na konci kapitoly jsou pak popsány další, s tématem související, technologie.

Kapitola číslo 4 se již zabývá návrhem samotné distribuce. Jsou zde rozebrány její problémy a možnosti jejich řešení. U každého takového řešení jsou udány podmínky, kdy je vhodné ho použít. Jsou tak vytvořeny kousky pomyslné skládačky, ze kterých je možné vybrat právě ty nejvhodnější části.

Kapitola číslo 5 popisuje návrh demonstrační aplikace pro distribuci a zobrazení GPS pozic na mapě. A na ní navazující kapitola číslo 6 popisuje realizaci této aplikace. V posledních dvou kapitolách se pak nachází zhodnocení distribuce přenosu procesních dat pomocí RTP a závěr.

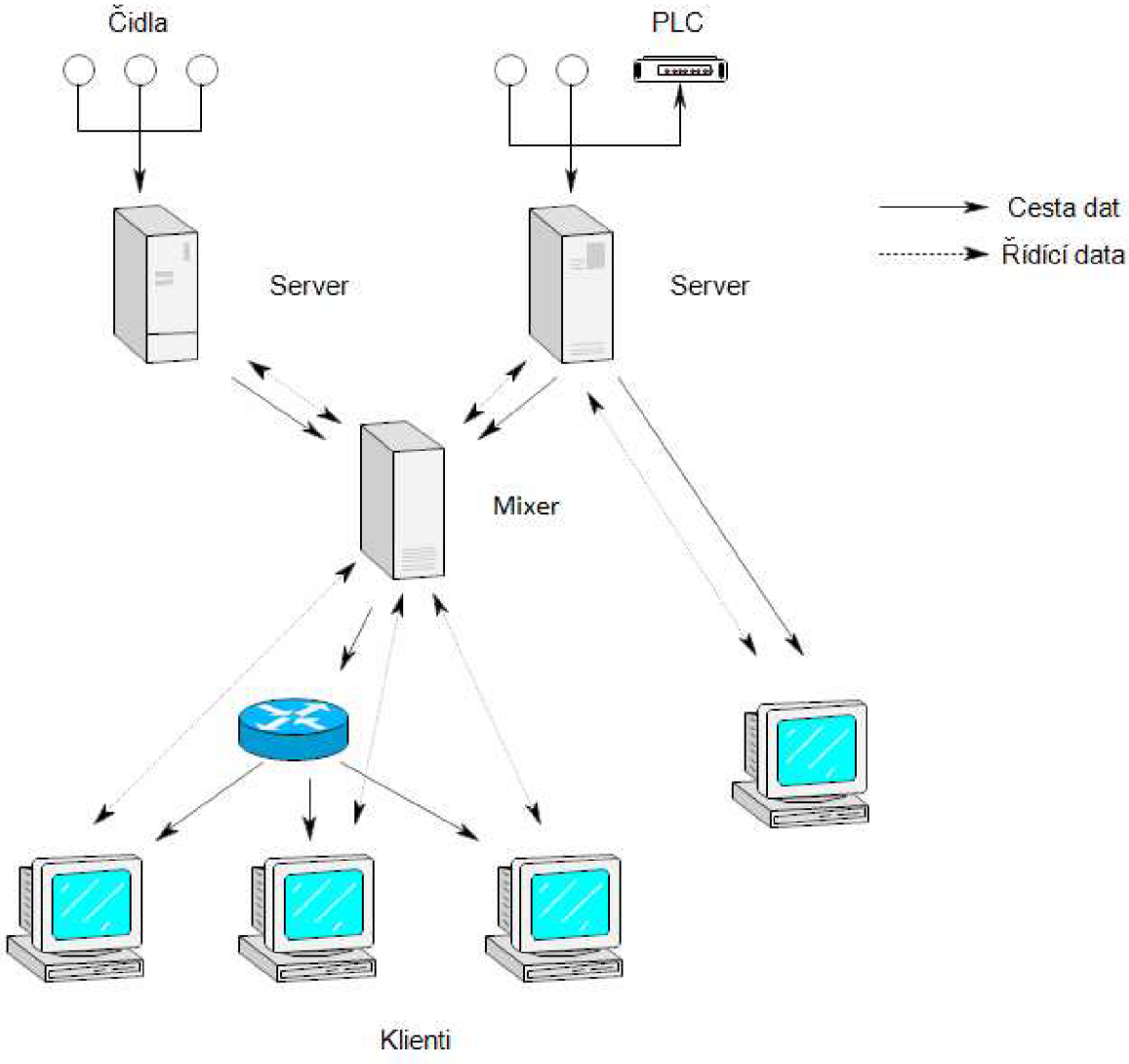
Ze semestrálního projektu je převzata a nepatrně rozšířena především kapitola číslo 3, popisující protokol RTP a možnosti jeho rozšíření. Základ kapitoly číslo 2 je také tvořen prací ze semestrálního projektu. Tato kapitola byla ale oproti předchozí práci přepracována a doplněna.

2 Současný stav problematiky

Pojmem procesní data označujeme data, která jsou většinou získána jako výsledek měření nejrůznějších čidel, popřípadě hodnoty reprezentující stavy v automatizačních systémech. Často se jedná o fyzikální veličiny. Pro přenos procesních dat se v dnešní době používá velké množství specializovaných protokolů, které jsou k tomuto účelu přímo navrženy. I přesto existují druhy aplikací, kdy tyto protokoly neposkytují příliš dobré výsledky. V této kapitole je nejprve popsán typ aplikace, na který se práce zaměřuje. Poté jsou z tohoto pohledu představeny a zhodnoceny vybrané existující protokoly pro přenos procesních dat.

2.1 Cílový typ aplikace a požadavky na přenos

Práce se zaměřuje na problém, kdy se na jediný, nebo velmi malé množství zdrojů dat připojuje velké množství klientů, kteří požadují stejná data, nejčastěji v reálném čase. U klientů je navíc předpokládána žádná, nebo velmi malá potřeba interakce se serverem v podobě odesílání příkazů.
Servery sdružují data z průmyslových sítí, které leží za nimi. Může se jednat o přímo připojená zařízení, nebo o rozsáhlejší sítě komunikující pomocí vlastních protokolů. Tato data jsou poté zpřístupněna pro klienty na síti typu Ethernet. Následující obrázek zobrazuje typickou strukturu sítě, která je zde popsána.



Obrázek 2.1: Struktura typické aplikace

Zdrojem dat je v uvažovaném případě server, který pro klienty zastupuje celou síť, ležící za ním. Cílem je poskytovat velké množství stejných dat ze serveru klientům. Z tohoto pohledu jde tedy o velmi podobný problém, který je řešen při poskytování multimediálních dat. Naopak protokoly, které jsou používány pro distribuci procesních dat v průmyslových sítích, jsou postaveny spíše na komunikaci bod-bod, což s rostoucím počtem klientů přináší nárůst množství přenášených dat. Je tedy nutné najít protokol, který by se vlastnostmi blížil spíše protokolům pro přenos proudů dat, než klasickým průmyslovým protokolům.

Přenos dat bude probíhat v síti postavené na TCP/IP. Protokol pro tuto aplikaci musí být jednoduchý na implementaci, otevřený, dobře specifikovaný a umožňující přenos dat pomocí přenosu typu multicast.

2.1.1 IP Multicast a unicast

Pro přenášení dat více klientům je velmi důležité vysvětlit pojmy unicast a multicast. V případě komunikace typu unicast jsou data přenášena mezi dvěma konkrétními účastníky. Pokud jsou data

jednoho účastníka požadována větším počtem jiných účastníků, je nutné je přenést každému účastníkovi zvlášť. Roste tedy množství přenášených dat.
U IP multicastu jsou data posílána na speciální rozsah ip adres, jedná se o adresy skupiny D, tedy adresy v rozsahu 224.0.0.0 - 239.255.255.255. Data odeslaná na tuto adresu jsou doručována všem účastníkům, kteří se přihlásí o odběr dat z této adresy.

Použití protokolu UDP s využitím přenosu typu multicast umožní jednak snížení celkové zátěže sítě oproti klasickému TCP spojení každého klienta se serverem a také snížení zátěže serveru díky jednoduššímu zpracování UDP paketů.

2.2 Protokoly použitelné pro distribuci procesních dat

V následující části jsou popsány protokoly, které by bylo možné použít pro distribuci procesních dat a jejich výhody a nevýhody.

2.2.1 Průmyslové protokoly pro sériový přenos

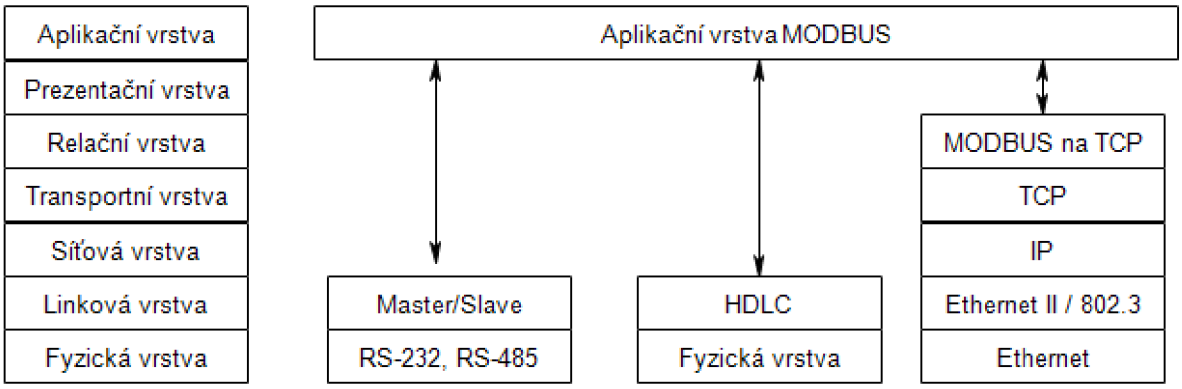
I přes fakt, že komunikace probíhá na síti Ethernet, by bylo možné pro přenos dat mezi serverem a klienty použít některý z průmyslových protokolů a ten nad IP tunelovat. Server by ovšem musel řešit synchronizaci komunikujících klientů. Pro komunikaci typu multicast tyto protokoly neposkytují žádnou podporu a musel by se navrhnout mechanismus znovu zasílání dat v případě jejich ztráty. Další důvod hovořící proti tomuto řešení je fakt, že server provádí archivování dat a je tedy nutné vyřešit i přístup k nim. Použití těchto protokolů je tedy velmi komplikované.

2.2.2 Průmyslové protokoly pro Ethernet

Síť typu Ethernet přináší oproti klasickým průmyslovým sítím několik výhod. Mezi největší patří například možnost propojení do jiných sítí LAN a sítě internet, možnost jednoduchého napojení na PC oproti sítím jako Profibus či Device Net. Je rychlý a díky masovému rozšíření je výroba síťových prvků velmi levná. Mezi nevýhody pak patří například nedeterministický přístup k médiu. Pro Ethernet vzniklo několik protokolů specializovaných na přenos procesních dat, některé jsou postaveny nad protokoly TCP a UDP, jiné pomocí nich přenáší pouze část dat a kritická data jsou přenášena pomocí vlastních protokolů. Velké množství protokolů nemá otevřenou specifikaci.

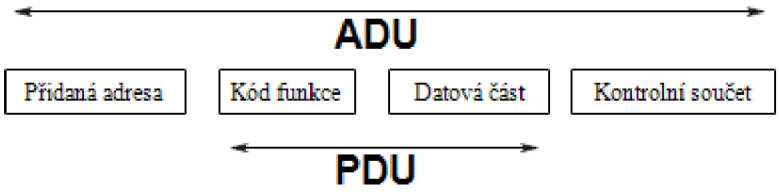
Modbus TCP

Prvním zástupcem průmyslových protokolů pro Ethernet je Modbus TCP. Vychází z původního protokolu Modbus, který byl vytvořen v roce 1979 firmou Modicon pro jejich programovatelné kontroléry [1]. Jedná se o velmi jednoduchý aplikační protokol, který je podporován velkým množstvím průmyslových zařízení. Na následujícím obrázku je vidět vztah protokolu Modbus a přenosových sítí.



Obrázek 2.2: Modbus a ISO/OSI model. Zdroj [1].

Protokol definuje strukturu zprávy na úrovni protokolu (PDU – Protocol Data Unit), která je nezávislá na typu sítě. K této zprávě poté přidává v závislosti na typu sítě další části a vytváří tak celkovou zprávu na aplikační úrovni (ADU – Application Data Unit). Díky tomu je snadné vytvořit most mezi sítěmi různých typů. Na následujícím obrázku je struktura zprávy.



Obrázek 2.3: Schéma zprávy Modbus. Zdroj [1].

Jedná se o protokol typu požadavek/odpověď. Na požadavek odpovídá dotazované zařízení zprávou obsahující buď indikaci úspěšného vykonání a případná data, nebo kód chyby.

Výhody:

- Jednoduchost a z toho vyplývající snadná implementace.
- Rozšířený a odzkoušený.
- Jednoduché propojení sítí více typů.
- Otevřená specifikace.

Nevýhody:

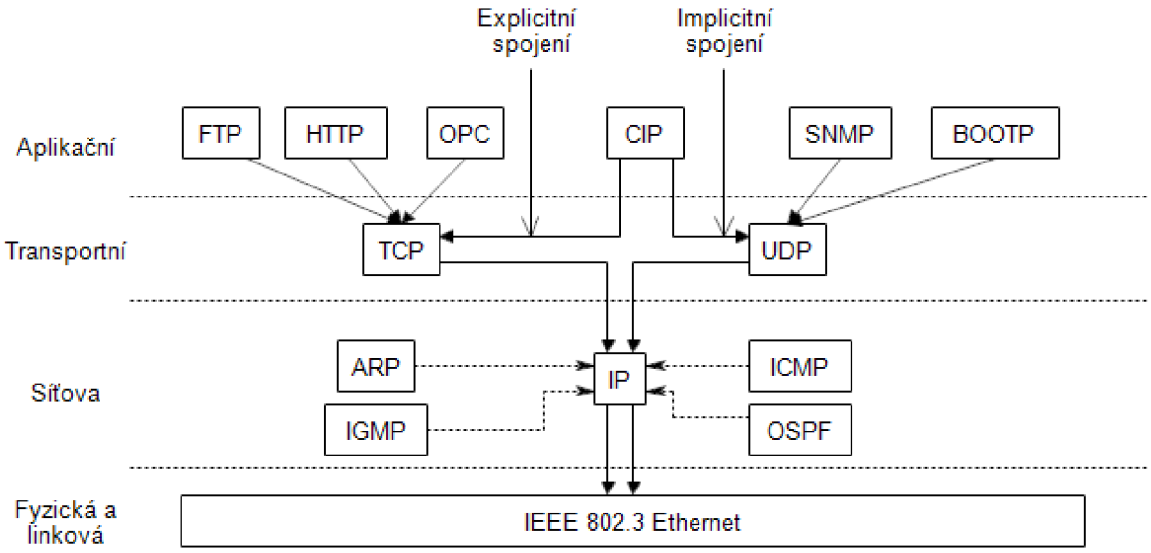
- Jedná se o zastaralý protokol.
- Je definován pro TCP. Z toho vyplývají problémy s přenosy typu multicast.
- Komunikace typu požadavek/odpověď není příliš vhodná pro navrhovaný typ distribuce dat.

Existuje i Modbus pro UDP, ale ten není definován jako standard.

Ethernet/IP

Ethernet/IP byl představen v roce 2001 s cílem poskytnout komplexní řešení pro automatizaci a síť Ethernet. V současnosti je spravován organizací ODVA. Patří do rodiny protokolů, které jako vyšší vrstvu implementují protokol CIP (Common Industrial Protocol). CIP představuje každé síťové zařízení jako množinu objektů. Definuje přístup, jejich chování a způsoby rozšíření, což umožňuje

přístup k mnoha různým zařízením pomocí jednotného obecného přístupu. Díky této abstrakci je možné používat pro správu a konfiguraci zařízení jednotné nástroje a nebýt tak vázán na výrobce zařízení. Také to umožňuje komunikaci mezi zařízeními v různých sítích, například DeviceNet a ControleNet [2].



Obrázek 2.4: Použití protokolů u Ethernet/IP.

Protokol definuje dva typy spojení. Explicitní, která jsou ustanovena mezi dvěma uzly a komunikace probíhá metodou požadavek/odpověď. Toto spojení slouží pro přenos konfigurací, diagnostických informací a událostí. Toto spojení je postaveno na spolehlivém TCP. Implicitní spojení slouží pro vysílání živých dat pomocí UDP s možným využitím přenosu typu multicast.

Výhody:

- Velmi komplexní protokol, umožňuje konfiguraci a ovládání zařízení a sběr dat.
- Je kompatibilní s protokoly pro přístup a výměnu dat, například OPC.
- Rozdělení implicitních a explicitních spojení mezi TCP a UDP optimalizuje využití síťových prostředků.
- Možnost komunikace s jinými sítěmi, které obsahují protokol CIP.

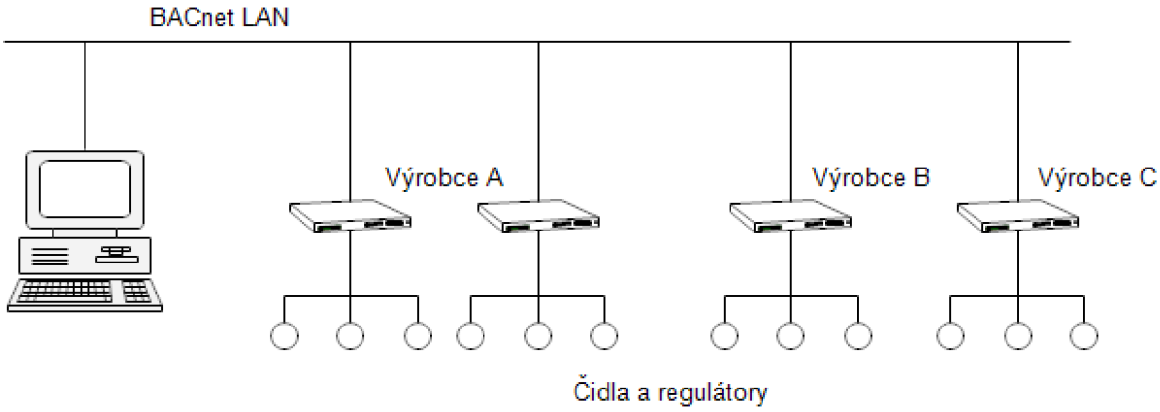
Nevýhody:

- Velmi složitý.
- Specifikace pro vývojáře není zdarma dostupná.
- Pro použití je nutné získat licenci.
- Šíření dat pomocí přenosu typu multicast z mnoha zdrojů vyžaduje řízení, aby nedošlo k zahlcení sítě.

BACnet/IP

BACnet je komunikační protokol pro automatizaci a ovládání budovních systému [3]. Vývoj protokolu začal v roce 1987 a v roce 2003 se stal ISO standardem. Cílem při návrhu bylo vyvinout protokol, který by umožnil spolupracovat budovním systémům od různých výrobců. Pro dosažení tohoto cíle jsou v protokolu BACnet definovány tři části. První popisuje standardní metodu pro reprezentaci jakéhokoli typu automatizovaného budovního zařízení. Druhá část definuje zprávy, které

mohou být posílány pro kontrolu a monitorování těchto zařízení. Poslední část definuje, které sítě mohou být použity pro přenos protokolu BACnet. Jsou to například ARCNET, Ethernet, LonTalk.



Obrázek 2.5: Příklad sítě s protokolem BACnet

BACnet/IP definuje mechanismy pro komunikaci pomocí protokolu BACnet přes síť Ethernet pomocí protokolu IP a UDP.

Pro šíření dat mezi více sítěmi používá BACnet/IP dva přístupy. První možností je klasický IP multicast, jehož výhodou je již existující podpora ze strany síťových zařízení. Druhou možností je použít zařízení nazývané „BACnet/IP Broadcast Management Device“ (BBMD), které přepošle zprávy typu broadcast z vlastní sítě dalšímu zařízení BBMD v jiné síti, které poté zajistí rozšíření zprávy ve vlastní síti.

Výhody:

- Otevřená specifikace.
- Síť BACnet se může skládat z více IP podsítí.
- Více možných přístupů pro šíření zpráv.
- Umožňuje připojení zařízení, které nejsou součástí žádné podsítě.

Nevýhody:

- Zaměřený především na budovní systémy.
- Šíření dat pomocí přenosu typu multicast z mnoha zdrojů vyžaduje řízení, aby nedošlo k zahlcení sítě.

2.2.3 Protokoly používané na síti internet

V rodině aplikačních protokolů používaných v síti internet je několik protokolů, které jsou specializované na přenášení dat, například FTP, HTTP, RTSP. Tyto protokoly se dají použít jako podpůrné, například pro výběr obsahu nebo ovládání, ale pro přenos živých dat nejsou vhodné. Problematický je především přenos přes multicast, absence prostředků pro časovou synchronizaci a šíření informace o kvalitě služby.

Pro získávání procesních dat se někdy využívá Simple Network Management Protocol (SNMP). Jedná se o aplikační protokol, který je původně určen pro monitorování a řízení sítě[4]. U běžného použití existuje jedna, nebo více administrátorských stanic, které mají za úkol monitorovat a spravovat skupinu síťových zařízení. Každé řízené zařízení má v sobě softwarovou komponentu nazvanou agent, která přes SNMP zasílá zprávy řídícímu zařízení. Standardní SNMP je založen na principu jeden dotaz, jedna odpověď. Živá procesní data však musí být vysílána automaticky s co

nejkratším zpožděním. Toto řešení je tak použitelné pro sběr procesních dat, ovšem pro jejich distribuci více klientům z jednoho serveru je nevhodné.

UDT

UDT je aplikační protokol postavený nad UDP. Je určen především pro přenos velkého množství dat přes vysokorychlostní síť. UDT doplňuje UDP o mechanismy spolehlivého přenosu dat a kontroly zahlcení linky. Spolehlivosti přenosu dat je zde dosaženo pomocí metody znovuzasílání ztracených paketů. Na rozdíl od TCP je její použití volitelné, případně lze nastavit dobu, po jejíž uplynutí jsou chybějící data zahozena. Následující tabulka obsahuje srovnání vlastností protokolů TCP, UDP a UDT. Přestože se jedná o aplikační protokol svým určením ho lze zařadit mezi protokoly transportní, proto je takovéto srovnání možné.

	TCP	**UDP**	**UDT**
Spojově orientovaný	ANO	NE	ANO
Spolehlivý	ANO	NE	ANO
Řízení zahlcení	ANO	NE	ANO
Multicast	NE	ANO	NE

Tabulka 2.1: Srovnání transportních protokolů

Pro využití protokolu UDT pro distribuci procesních dat by bylo potřeba, stejně jako u RTP, definovat mechanismy pro řízení spojení a výběr přenášených dat. Nevýhodou tohoto protokolu je fakt, že je spojově orientovaný, tudíž nelze použít pro přenos typu multicast.

2.2.4 Protokoly pro proudový přenos dat

MMS

MMS protokol slouží k přenosu multimediálních dat v reálném čase. Navíc obsahuje mechanismy pro řízení přehrávání - odesílání požadavků typu start, stop atd. na server. Jedná se o protokol navržený firmou Microsoft, který byl původně uzavřený, ale v roce 2008 byla uvolněna plná specifikace protokolu. V současné době již není tento protokol dále oficiálně podporován a přehrávač Windows Media Player 11 dokonce neobsahuje žádnou podporu pro protokol MMS. Místo MMS je nyní využívána kombinace protokolu RTP a RTSP.

Windows Media HTTP Streaming Protocol

Jedná se o protokol navržený firmou Microsoft pro přenos proudu dat v reálném čase [5]. K přenosu všech dat využívá protokol HTTP, kdy klient vysílá požadavky na server a server v odpovědích odesílá data. Protože je k přenosu použit HTTP protokol, je jako transportní protokol k dispozici pouze TCP. Problémem je také nutná aktivní účast klienta, který musí odesílat požadavky na data. Použití pouze protokolu TCP zase znemožňuje přenos typu multicast, při odesílání dat více klientům tak dochází k zatěžování sítě.

RTMP

Real Time Messaging Protocol je protokol navržený firmou Adobe Systems, který je využíván pro doručování multimediálních dat mezi přehrávačem a serverem [6]. RTMP protokol slouží jako kontejner pro přenášená data. Samotný přenos může probíhat třemi způsoby.

1. Pomocí čistého RTMP protokolu, který je přenášen přes binární TCP spojení.
2. Protokol RTMP lze tunelovat přes HTTP spojení tak, že klient periodicky vysílá POST požadavky na server, aby zjistil, zda jsou k dispozici nějaké nové události. Pro tento účel je definován protokol RTMPT, který slouží jako obálka pro protokol RTMP.
3. Pomocí stejného principu jako v předchozím případě, ale je tunelován protokol HTTPS a jako obálka je definován protokol RTMPS.

Tunelování HTTP popřípadě HTTPS protokolu umožňuje poskytování obsahu i klientům, kterým firewall neumožňuje jinou komunikaci, než právě přes porty využívané HTTP protokolem. Na rozdíl od protokolu RTP obsahuje RTMP funkce pro navazování a řízení spojení. V rámci RTMP je definováno několik na sobě nezávislých kanálů. Například samostatný kanál pro audio, kanál pro video a kanál pro vzdálené volání procedur. Data mohou být přijímána a odesílána souběžně na více kanálech.

Stejně jako v případě Windows Media HTTP Streaming Protocol je díky použití TCP protokolu problematické přenášení dat k více klientům.

2.2.5 Protokol RTP

Protokol RTP slouží pro distribuci multimediálních dat. Jeho detailnímu popisu je věnována samostatná kapitola, kde jsou uvedeny příklady typického použití. Pro účely distribuce procesních dat je nejzajímavější možnost vysílání z jednoho zdroje mnoha klientům. RTP nedefinuje způsob uložení dat, která přenáší a umožňuje částečně modifikovat své chování pomocí tzv. profilů a je tak snadno rozšiřitelný o další typy přenášených dat.

Jedná se o poměrně jednoduchý otevřený protokol, pro který existuje velké množství implementací. Je přímo určen pro přenosy typu multicast což ho předurčuje pro přenos dat k mnoha klientům. Komplementární protokol RTCP dále poskytuje informace o kvalitě poskytované služby.
Naopak velkou nevýhodou je neexistence kanálu, pomocí kterého lze přenášet data jako například povely ve směru klient – server a nutnost definovat způsob jakým bude přenášen obsah ve formě živých procesních dat.

3 Volba technologií pro distribuci procesních dat a jejich popis

V první části této kapitoly je zdůvodněn výběr protokolů, které jsou vhodné pro dříve popsanou aplikaci. Tyto protokoly jsou poté v následujících částech kapitoly podrobně popsány. Tyto znalosti jsou uplatněné v pozdějších kapitolách při návrhu distribuce procesních dat a při implementaci demonstrační aplikace.

3.1 Volba protokolů

V předchozí kapitole bylo popsáno několik existujících protokolů, které se pro přenos procesních dat používají nebo by je bylo možné použít. U každého protokolu byla zhodnocena vhodnost použití pro popisovanou aplikaci. Některé protokoly nelze použít vůbec u jiných by se vyskytly problémy zejména s přenosem typu multicast, nebo s přílišnou složitostí. Protokol RTP je poměrně jednoduchý a je přímo navržen pro přenos typu multicast. Díky své snadné rozšiřitelnosti je tak pro popsanou aplikaci vhodným řešením.

RTP řeší jen samotný přenos, nikoli však navazování a řízení sezení a získávání informací o kvalitě poskytované služby. Pro řízení sezení je nutné vybrat jiný protokol. Pro tento účel je standardně používán protokol RTSP [7]. RTSP také poskytuje kanál ve směru klient – server, pomocí něhož je možné přenášet řídící informace. Dalším protokolem, který je při přenosu dat pomocí RTP použit je protokol RTCP. Jedná se o komplementární protokol k protokolu RTP a je spolu s ním specifikován. Umožňuje získávání informací o kvalitě spojení, vzájemnou synchronizaci více zdrojů a je také rozšiřitelný dle aplikačních požadavků. Všechny tři zde zmíněné protokoly jsou otevřené a existuje několik jejich volně dostupných implementací.

3.2 Popis protokolu RTP

Real time transport protocol (RTP) je protokol, který definuje standard pro paketový přenos především multimediálních dat. Byl vyvinut Audio-Video Transport Working Group při IETF a publikován v roce 1996 jako standard RFC 1889. V roce 2003 byla specifikace aktualizována v RFC 3550 [8].

3.2.1 Základní charakteristika

Protokol RTP poskytuje službu pro přenos dat, která jsou poskytována v reálném čase. Jedná se především o multimediální obsah a je využíván například u videokonferencí pro přenos audia a videa, nebo pro vysílání videa po internetu. Tento typ dat je velmi náročný na přenosovou kapacitu sítě. Ke snížení nároků je využíváno přenosu pomocí multicast skupin. Aplikace je typicky postavena na přenosu pomocí protokolu UDP, ale je možné využít i jiných protokolů. Protokol žádným způsobem nespecifikuje data, která přenáší. Způsob uložení konkrétního typu dat v paketu, popřípadě speciální chování protokolu, je vždy definováno zvlášť. V anglicky psané literatuře se obsah RTP paketu označuje jako RTP payload. V této práci bude označován jako typ obsahu, popřípadě pouze obsah.

Protokol RTP měl umožnit přenos dat nad nespolehlivou transportní vrstvou [7]. K dosažení tohoto cíle byly použity dva koncepty, které je možné vidět i u jiných síťových protokolů. Jsou jimi rámcování na aplikační úrovni a princip konec-konec.

Rámcování na aplikační úrovni

Koncept rámcování na aplikační úrovni byl poprvé publikován v roce 1990 [9]. Ústřední myšlenkou tohoto přístupu je, že jedině samotná aplikace má dostatek informací o datech, aby bylo možno rozhodnout, jakým způsobem by měla být přenášena. Důvodem pro vznik této myšlenky byl poznatek, že při ztrátě dat existuje několik způsobů, jak se s touto ztrátou vyrovnat. Jsou případy, kdy je nutné ztracená data znovu odeslat, v jiných případech je možné data nahradit jinými, takže data získaná po přenosu jsou rozdílná od dat, která byla odeslána.

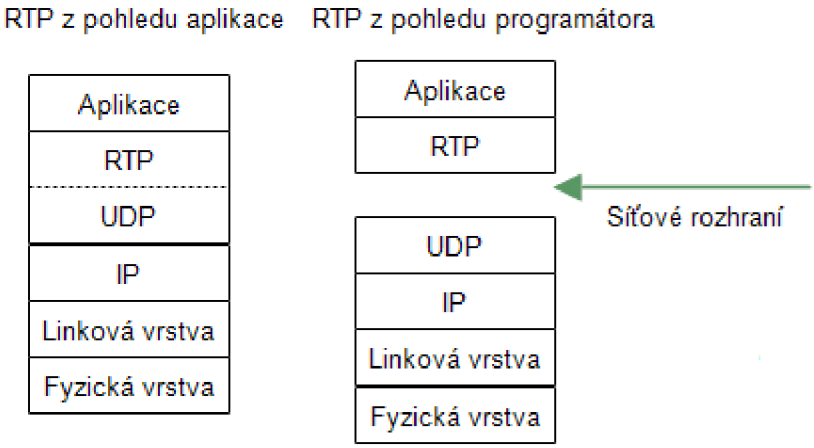
Tato myšlenka je v rozporu s protokolem TCP, který poskytuje spolehlivý přenos dat na úkor časového zpoždění, avšak je dobře aplikovatelná s využitím přenosu na bázi UDP a charakteristickými vlastnostmi multimediálních dat. Multimediální data jsou tolerantní ke ztrátám, takže je možné ztráty akceptovat. Pokud je i přesto nutné chybějící data doplnit, tak jsou k dispozici prostředky pro jejich znovuzískání. Například znovuzasílání dat, nebo dopředná korekce chyb. O jejich volbě a vhodnosti rozhoduje sama aplikace.

Princip konec-konec

Princip konec-konec představuje dvě možnosti, které je možné použít při návrhu systému, které musí spolehlivě komunikovat přes nespolehlivou síť. Jedním přístupem je, že systém předává odpovědnost za přenos dat spolu s daty, takže zajištění spolehlivosti je na každém jednotlivém prvku, který se přenosu účastní. Druhou možností je přenechání odpovědnosti za spolehlivý přenos na koncových prvcích. Druhého přístupu bylo využito při návrhu protokolu TCP i RTP.

3.2.1.1 Protokol RTP v modelu TCP/IP

Při zařazení protokolu RTP do příslušné vrstvy se nabízí dva pohledy. Z pohledu aplikace je možné ho zařadit jako mezivrstvu transportní vrstvy, protože obsahuje mechanismy jako časové známky, sekvenční čísla atd., které poskytuje aplikační vrstvě. Z pohledu aplikačního vývojáře se však jedná o protokol aplikační vrstvy, protože musí být přímo integrován do aplikace. Na následujícím obrázku jsou znázorněny tyto dva rozdílné pohledy.



Obrázek 3.1: Umístění v modelu TCP/IP

Jelikož protokol RTP využívá služeb transportní vrstvy, budeme na něj v této práci nahlížet jako na protokol vrstvy aplikační.

3.2.2 Dostupné implementace protokolu

3.2.2.1 ccRTP

ccRTP poskytuje C++ základ pro vývoj aplikací využívajících protokol RTP. ccRTP je postavený na GNU Common C++ framework, ze kterého využívá služby jako vlákna a práci se síťovou vrstvou. Podporována je verze protokolu v RFC 3550 a od verze 1.5 je navíc podporován i protokol SRTP (secure RTP). ccRTP podporuje všechny typy komunikace unicast, multicast – unicast a multicast. ccRTP je volně dostupné pod licencí GNU General Public License.

3.2.2.2 JRTPLIB

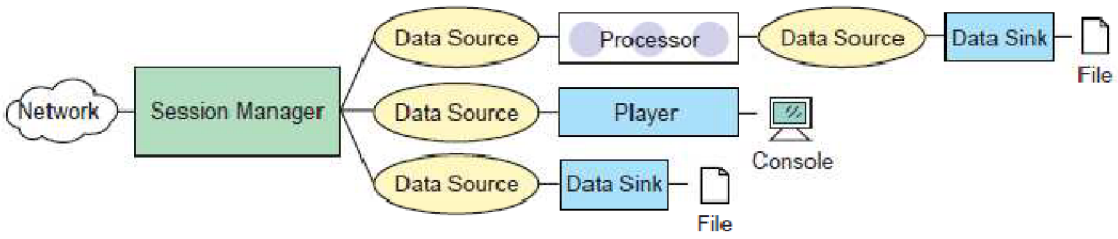
JRTLIB je objektově orientovaná knihovna napsaná v jazyce C++, která implementuje RTP a RTCP protokol, jak je popsaný v RFC 3550. Její zdrojové kódy jsou vydány pod licencí, která v sobě neobsahuje žádné restrikce na způsob, jakým je knihovna použita. Navíc obsahuje velmi podrobnou dokumentaci.

3.2.2.3 RTP.NET

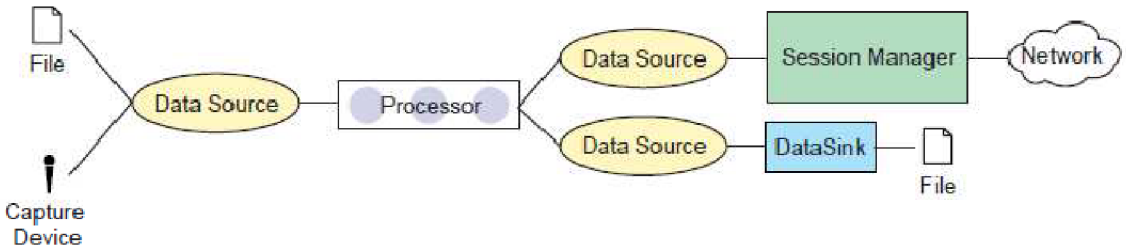
RTP.NET je implementace RTP v prostředí .NET. Je plně rozšiřitelné a navržená tak, aby ji bylo možné propojit s dalšími komponentami, které poskytuje prostředí .NET. Implementace je zdarma dostupná v podobě .NET assembly. Jedinou podmínkou je uvedení informace o jejím použití ve vydaném software. Zdrojové kódy jsou zpoplatněny. Bohužel dokumentace, která je k dispozici, není příliš podrobná.

3.2.2.4 JMF RTP

JMF (Java Media Framework) poskytuje základ pro vývoj multimediálních aplikací v jazyce Java[10]. Byla vyvinuta firmami Sun Microsystems, Silicon Graphics a Intel a první verze byla uvolněna v roce 1997. Verze 2.0 byla vyvinuta firmou SUN a IBM a vydána v roce 1999. V současné době je aktuální verze 2.1.1.
Kromě podpory multimedií JMF také umožňuje příjem a vysílání dat přes RTP protokol. Rozhraní pro práci s protokolem RTP je definováno v balíčcích javax.media.rtp, javax.media.rtp.event, javax.media.rtp.rtcp. JMF poskytuje standardní mechanismus pro rozšíření již existujících RTP formátů o formáty nové.



Obrázek 3.2: Příjem RTP dat. Zdroj [10].



Obrázek 3.3: Odesílání RTP dat. Zdroj [10].

3.2.3 Popis funkce protokolu RTP

Základní principy funkce protokolu RTP jsou definovány v příslušném RFC [8]. Protokol RTP je ale navržen tak, aby bylo možné ho použít v co možná nejširším spektru aplikací pro přenos dat, které mají rozdílné požadavky. Proto je část vlastností protokolu definována až na základě aplikace, která ho využívá. V této kapitole jsou probrány základní principy, které jsou v RTP protokolu použity.

3.2.3.1 Struktura paketu

Každý RTP paket se skládá z hlavičky, která má pevnou část, za níž následují možná rozšíření a poté samotná data. Hlavička má následující strukturu:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
V		P	X	CC				M	PT							Sekvenční číslo															
Časová známka																															
Identifikátor synchronizovaného zdroje (SSRC)																															
Identifikátor přispívajících zdrojů (CSRC)																															

Obrázek 3.4: Struktura RTP paketu

V – Verze. První pole udává verzi protokolu. V současné době je nejnověji definovaná verze 2 v RFC 3550. Hodnota 1 byla používána pro verzi definovanou v RFC 1890.
P – Vycpávka. Pokud je bit v tomto poli nastaven, na konci paketu jsou umístěny další byty jako vycpávka, která může být vyžadována některými šifrovacími algoritmy, které pracují s bloky data s pevnou délkou.
X – Rozšíření. Pokud je bit nastaven, musí za hlavičkou následovat přesně jedna rozšiřující hlavička.
CC – Udává počet identifikátorů přispívajících zdrojů, které následují za hlavičkou.
M – Interpretace tohoto bitu je ponechána na konkrétním RTP profilu.
PT – Typ přenášeného obsahu. Pole definuje formát přenášeného obsahu a jeho interpretaci aplikací. Toto pole je podrobněji popsáno v kapitole zabývající se rozšiřováním protokolu RTP.
Sekvenční číslo – Sekvenční číslo je zvětšeno o jedna s každým odeslaným paketem. Může být použito příjemcem pro detekci ztrát paketů a znovu sestavení sekvence paketů. Počáteční hodnota by měla být zvolena náhodně, aby byly obtížněji realizovatelné útoky na zašifrovaná data s pomocí známých částí textu. Je nutné si uvědomit, že i v případě, že samotný zdroj odesílaná data nešifruje, může k jejich zašifrování dojít během jejich cesty k příjemci.

Časová známka – Časová známka souvisí s prvním bajtem v datové části paketu. Časová známka je zvyšována v závislosti na typu přenášených dat. Jakmile dosáhne maximální hodnoty, začne číslování opět od nuly.
SSRC – Identifikátor synchronizovaného zdroje. Tento identifikátor by měl být zvolen náhodně a žádné dva zdroje v RTP sezení ho nesmí mít stejný. Všechny implementace protokolu musí umožňovat detekci a vyřešení případných kolizí.
CSRC – List zdrojů, které jsou obsaženy v obsahu paketu. Počet těchto zdrojů je dán v poli CC. Jako identifikátor každého přispěvatele je použit jeho SSRC a jsou vkládány při mixování více zdrojů do jednoho paketu.

3.2.3.2 Správa spojení

V reálných aplikacích je kromě RTP protokolu pro přenos dat použito mnoho dalších, které zajišťují funkce nutné pro poskytovanou službu a které nejsou jeho součástí [7]. Jde například o sestavování spojení, popis sezení, výběr obsahu atd. V této podkapitole je uvedeno několik příkladů protokolů, které se používají pro navazování a správu sezení. Jejich detailní popis je však mimo rozsah této práce. Volba vhodného protokolu záleží na typu aplikace a spojení. V zásadě mohou nastat následující případy:

- Interaktivní sezení, například telefonní hovor nebo video konference. Pro tyto účely jsou definovány dva standardy. Původní protokol pro tuto oblast byl H.323. Později byl definován nový protokol SIP.
- Pro neinteraktivní aplikace, jako je poskytování videa na vyžádání, je běžně používán protokol RTSP.
- V případě využívání IP multicastu je typicky použit protokol SAP.

Protože výše zmíněné případy vyžadují protokoly s velmi odlišnými možnostmi a vlastnostmim, nebyla do protokolu RTP začleněna žádná funkcionalita spojená se správou sezení. Ta byla přenechána na jiné specializované protokoly.

3.2.3.3 Translátory a mixery

Kromě koncových účastníků sezení jsou definovány další dva prvky, které stojí mezi koncovými účastníky a modifikují proud dat, který jimi prochází. Jedná se o translátory a mixery.

Translátory

Translátor je prvek, který operuje nad RTP daty a pro koncové účastníky je neviditelný. Činnost translátoru je dána typem aplikace a může mít mnoho podob. Zde jsou uvedeny příklady typických funkcí poskytovaných translátorem [7].

- Přemostění mezi dvěma různými transportními protokoly.
- Změna kódování multimediálních dat.
- Rozšíření jednoho paketu mezi více translátorů.
- Sjednocení paketů z více translátorů do jednoho paketu.

Translátory nejsou účastníky sezení a neposílají žádné RTCP pakety.

Mixery

Mixer je prvek, nacházející se mezi koncovými účastníky, který má za úkol přijímat data z více zdrojů a spojovat je do jednoho výsledného proudu dat. Každý mixer má své vlastní SSRC a to je použito pro odchozí proud dat. SSRC příchozích proudů jsou zkopírována do pole CSRC. Při

skládání příchozích proudů musí mixer provést jejich vzájemnou synchronizace a může také změnit kódování.

3.2.3.4 Ztráty paketů

Díky nespolehlivé transportní vrstvě, kterou protokol RTP využívá, může během přenosu docházet ke ztrátám paketů. Protokol RTP neposkytuje žádnou přímou podporu, která by umožnila tyto situace řešit [7]. V případě, že přenášená data nejsou ke ztrátám paketů tolerantní, je na aplikaci aby případné výpadky vyřešila.

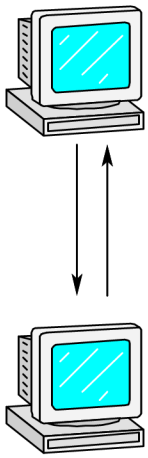
K tomuto účelu se využívají dva postupy, dopředná korekce chyb a znovuzasílání chybějících dat.

3.2.3.5 Modely komunikace

Při používání protokolu RTP může být využito několik různých modelů komunikace, záleží na počtu a typu jednotlivých účastníků sezení. V této podkapitole si představíme jednotlivé možnosti.

Unicast

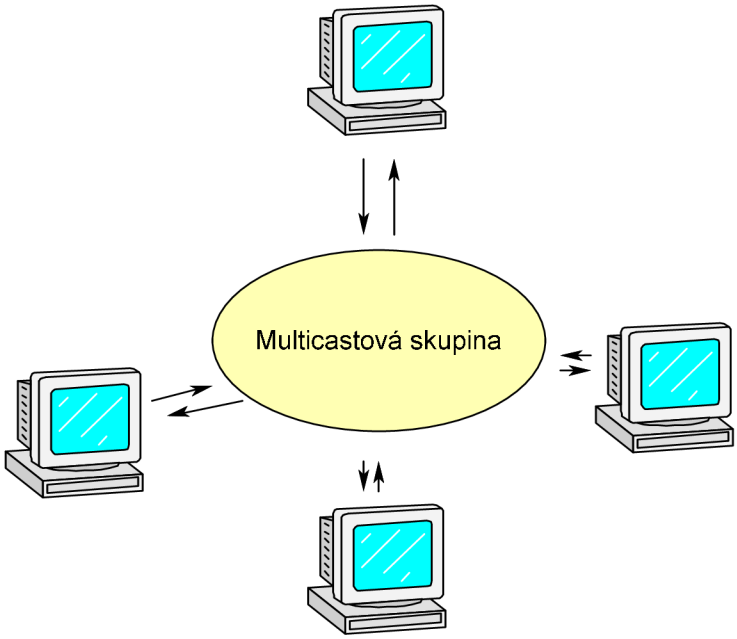
Nejjednodušší možností je vysílání dat z jednoho zdroje jednomu příjemci, tzv. unicast.



Obrázek 3.5: Příklad komunikace typu unicast

Multicast

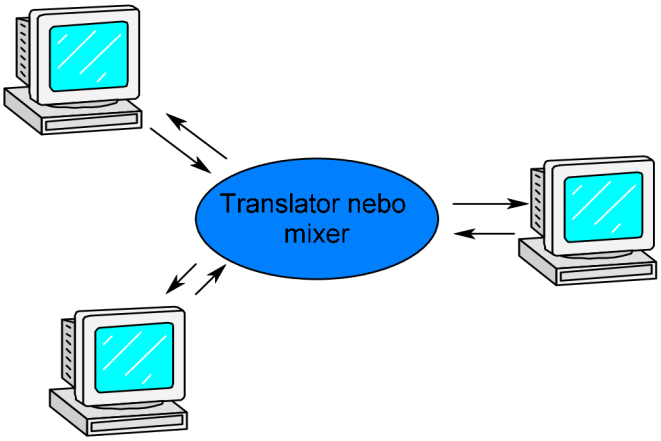
Při větším množství účastníků je vhodné, aby zbytečně nedocházelo k zatěžování vícenásobným přenosem stejných dat různým účastníkům. K tomu slouží multicast, jehož princip byl podrobněji popsán v samostatné kapitole.



Obrázek 3.6: Příklad komunikace typu multicast

Replikovaný unicast

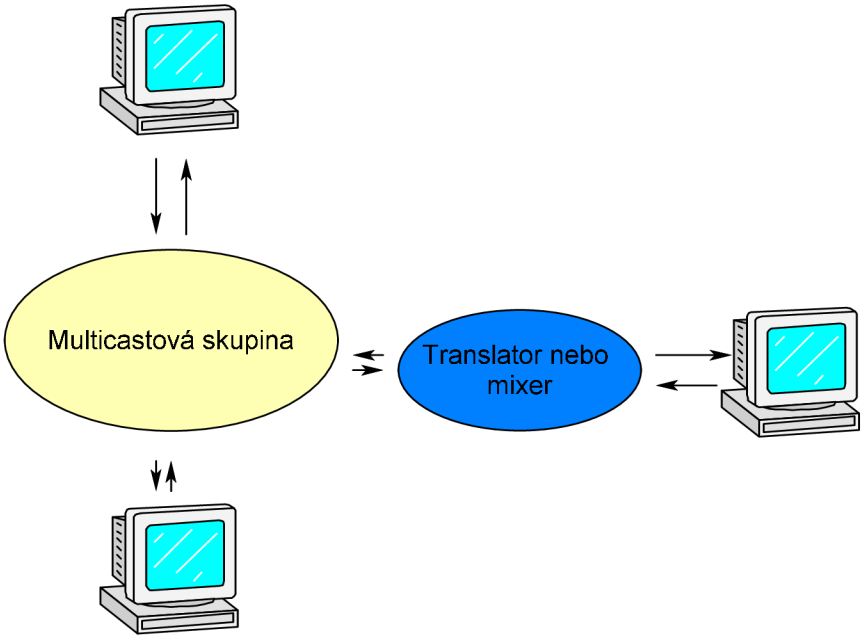
Mezi komunikující členy sezení je vložen RTP translátor, nebo mixer, který zajišťuje přenos dat typu unicast mezi účastníky.



Obrázek 3.7: Příklad komunikace typu replikovaný unicast

Kombinace přenosu typu multicast a unicast

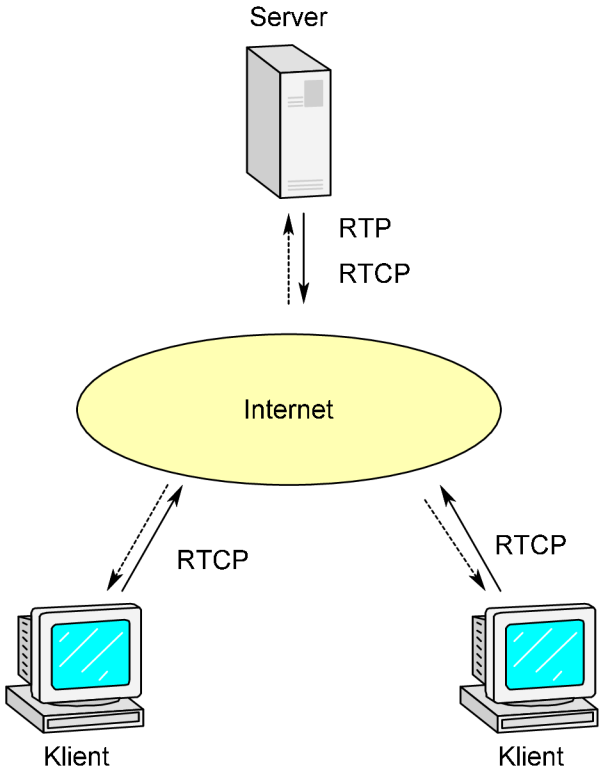
S využitím RTP translátoru nebo mixeru je možné, aby část účastníků sezení komunikovala pomocí multicastové skupiny a část byla připojena do této skupiny přes mixer nebo translátor. Toto zapojení znázorňuje následující obrázek.



Obr. 3.8: Příklad komunikace typu multicast a unicast

3.2.4 Protokol RTCP

Real-time transport control protocol (RTCP) je řídící protokol, který je těsně spjat s protokolem RTP [8]. Poskytuje prostředek pro získávání statistických dat, která mohou být použita pro řízení toku dat v samotném RTP protokolu.



Obrázek 3.9: Komunikace pomocí RTCP

3.2.4.1 Funkce protokolu RTCP

Protokol RTCP je založen na periodickém vysílání kontrolních paketů všem účastníkům sezení. Kontrolní pakety využívají stejný způsob distribuce jako data, ale musí být od nich odděleny. Například u přenosu pomocí protokolu UDP pomocí samostatného portu pro data a pro kontrolní pakety. RTCP protokol poskytuje čtyři typy funkčnosti:

1. Základní funkcí je poskytování statistických informací o kvalitě distribuce dat. Tyto informace mohou být využity v případě přenosu dat s přizpůsobivým kódováním, nebo pro detekci chyb přenosu.
2. Protokol RTCP obsahuje trvalý identifikátor zdroje známý pod zkratkou CNAME. Jelikož může v případě kolizí nebo restartu programu dojít ke změně SSRC identifikátoru zdroje, je CNAME vyžadován příjemcem k přiřazení příslušného poskytovatele.
3. Jelikož jsou RTCP pakety vysílány mezi všemi příjemci, je možné pomocí nich vypočítat počet příjemců. Ten je použit pro výpočet četnosti vysílání RTCP paketů.
4. Poslední funkcí je možnost přenosu kontrolních informací sezení. Tato funkce je volitelná a nemusí ji poskytovat všechna prostředí.

3.2.4.2 Typy zpráv

Specifikace RTPC definuje několik základních typů zpráv. Protokol je však rozšiřitelný, takže další typy zpráv mohou být definovány na základě potřeb konkrétní aplikace. Každý RTCP paket obsahuje pevnou část, která je podobná té v RTP, za níž následuje část proměnné délky, která je závislá na typu zprávy. Následující typy zpráv jsou definovány:

Zpráva odesílatele (SR)

Stanice, které odesílají data, pravidelně vysílají zprávy tohoto typu. Jsou v nich obsaženy informace o probíhající komunikaci a také přijímací statistiky pro všechny posílané RTP pakety. To umožňuje příjemcům odhadnout přenosovou rychlost a kvalitu.

Zpráva příjemce (RR)

Zprávy odesílají příjemci dat a obsahují informace o kvalitě služeb, problémech příjemce a mohou obsahovat čísla ztracených paketů. Tyto zprávy jsou odesílány všem účastníkům sezení.

Zpráva popisu zdroje (SDES)

Odesílatel dat pravidelně posílá zprávy SDES, které v sobě obsahují informace o odesílateli.

Zpráva bye (BYE)

Touto zprávou oznamuje odesílatel dat ukončení vysílání proudu dat.

Aplikačně specifické zprávy (APP)

Aplikačně specifické zprávy umožňují rozšíření o aplikačně specifické zprávy, které nejsou definovány ve standardu.

3.2.4.3 Interval odesílání

Protokol RTP byl navržen tak aby byl škálovatelný od několika málo poskytovatelů až po tisíce. Trafik protokolu RTCP by však bez omezujících podmínek rostl lineárně s každým novým

účastníkem. Proto musí být interval odesílání počítán dynamicky na základě nastavených podmínek a počtu účastníků. Pro každé sezení je definován parametr šířka pásma sezení, který udává maximální možné využití kapacity sítě. Jedná se o agregovaný limit společný pro všechny poskytovatele. Tento parametr je využíván při výpočtu intervalu odesílání a pro všechny účastníky musí mít hodnotu. Hodnota může být zvolena na základě znalosti šířky pásma sítě, nebo pomocí nějakého druhu ocenění.

Trafik RTCP paketů by měl být limitován na malou a známou část celkové šířky pásma sezení. Malou tak, aby nebyla narušena primární funkce RTP protokolu přenášet data a známou proto, aby mohla být tato informace využita pro plánování rezervace zdrojů. Ve specifikaci RTP protokolu je doporučeno, aby byla hodnota trafiku RTCP paketů fixně nastavena na 5% z celkové šířky pásma sezení. Druhým doporučením je, aby 25% z trafiku RTCP bylo přiděleno účastníkům, kteří odesílají data. Tím je zaručeno, že v sezení ve kterém je málo odesílatelů, ale velké množství příjemců, což je typický stav pro většinu sezení využívající RTP protokol, dostane nový účastník rychleji CNAME od vysílajících účastníků. Doporučený minimální interval pro odesílání je 5s. Algoritmus pro výpočet intervalu musí mít jako vstupní proměnnou počet účastníků. RTCP paket by měl účastník odeslat mezi náhodně zvoleným 0.5 – 1.5 násobkem vypočteného intervalu, aby nedošlo k současnému odesílání všech účastníků.

RTP profil může definovat jiné hodnoty pro interval odesílání, než ty které jsou doporučené ve specifikaci. Například může dojít k rozdělení parametru šířky pásma sezení na šířku pásma pro odesílající účastníky a šířku pásma pro příjemce. Nebo může dojít k drastickému snížení časových limitů v případě potvrzování RTP paketů pomocí RTCP spojení.

3.3 Možnosti rozšíření protokolu RTP

Protokol RTP byl navržen tak, aby jej mohlo využívat co nejširší spektrum aplikací. S tím je spojena velká odlišnost vlastností, které jednotlivé aplikace od protokolu vyžadují. Protokol tedy umožňuje modifikaci některých svých vlastností a rozšíření pro další typy dat, která může přenášet, bez toho aby bylo nutné měnit specifikaci samotného RTP. V této kapitole jsou vysvětleny principy, pomocí kterých jsou tato rozšíření vytvářena a nachází se zde souhrn již existujících standardizovaných rozšíření.

3.3.1 Princip rozšiřování protokolu

Pro použití protokolu RTP v aplikaci jsou potřeba další dvě definice. První z nich je RTP profil, který upravuje chování protokolu RTP. Druhou je definice obsahu paketu, který popisuje jakým způsobem je konkrétní typ dat pomocí RTP přenášen.

3.3.1.1 RTP profily

Jak již bylo řečeno RTP profil upravuje chování RTP protokolu pro konkrétní aplikaci. Typická aplikace využívá v rámci jednoho sezení pouze jeden RTP profil. Proto RTP protokol neobsahuje žádnou explicitní identifikaci, který profil je právě využíván. Specifikace protokolu RTP [8] určuje následující položky, které je možné v rámci profilu modifikovat. Mohou ale být modifikovány i jiné, které se v tomto seznamu nenachází.

RTP datová hlavička

Bajt v datové hlavičce, který obsahuje bit značky a typ obsahu může být modifikován, tak aby vyhovoval různým požadavkům, například může být zvětšen prostor pro značku na úkor typu obsahu.

Typ obsahu

Pokud pole typ obsahu existuje, profil může definovat několik typů obsahu, které jsou tímto profilem podporovány. Definice může být dána i odkazem na již existující typ obsahu definovaný v příslušném RFC. Dále ke každému definovanému formátu obsahu definuje defaultní statické mapování na hodnotu typu obsahu, která tento formát identifikuje v datové hlavičce. U každého typu obsahu musí být definován a rychlost čítače hodiny pro časovou známku.

Přídavky RTP datové hlavičky

Za fixní datovou hlavičku mohou následovat další rozšiřující pole, pokud jsou vyžadována aplikací v rámci celého profilu a jsou nezávislé na konkrétním typu obsahu.

Rozšíření RTP datové hlavičky

Pokud aplikace využívá mechanismus rozšíření, musí profil definovat prvních 16 bitů fixní rozšiřující hlavičky.

Typ RTCP paketu

Profil může definovat nové typy RTCP paketů, které jsou specifické pro danou aplikaci.

RTCP interval odesílání

Profil by měl specifikovat hodnoty, které budou použity pro výpočet intervalu odesílání RTCP zpráv. Jedná se o podíl protokolu RTCP na šířce pásma sezení, minimální interval odesílání a rozdělení šířky pásma mezi odesílateli a příjemci.

SR/RR rozšíření

Mohou být definována rozšíření pro zprávy SR/RR v případě, že aplikace vyžaduje pravidelné odesílání informací o odesílateli nebo příjemci.

Použití SDES

Profil může specifikovat priority pro odesílání RTCP SDES položek, nebo je dokonce úplně vyloučit z komunikace. Dále může změnit syntaxi a sémantiku jednotlivých položek, nebo dokonce definovat nové.

Bezpečnost

Profil může specifikovat které bezpečnostní služby a algoritmy by měly být poskytovány aplikacemi a měl by také poskytnout vodítko pro jejich správné použití.

Přetížení

Profil by měl specifikovat chování kontroly přetížení.

Protokol nižší vrstvy

Profil může vyžadovat použití konkrétní sítě, nebo přenosového protokolu pro přenos RTP paketů.

Mapování transportní vrstvy

Profil může definovat mapování RTP a RTCP na adresy transportní vrstvy, pokud je vyžadováno jiné než standardní. Například porty v případě použití protokolu UDP.

3.3.1.2 Typ obsahu

Typ obsahu musí především definovat, jaká data budou přenášena a jak budou uložena v RTP paketu. Jeden typ obsahu může být využíván ve více RTP profilech, proto může být výhodné ho definovat nezávisle na konkrétním profilu. Profil musí definovat mapování typu obsahu na hodnotu typu obsahu v hlavičce RTP paketu.

3.3.2 Existující rozšíření

Existuje velké množství rozšíření pro protokol RTP. Některé jsou ve formě definice samostatného typu obsahu, jiné definují celý profil, který obsahuje definici množství typů obsahu. V této podkapitole je krátký popis multimediálního profilu pro video konference spolu s výčtem typů obsahu, které jsou s ním spojeny. Dále se podkapitola zabývá existujícími rozšířeními pro nemultimediální data.

3.3.2.1 RTP profil pro audio a video konference

Profil je určen pro přenos dat v audio video konferencích s minimální kontrolou sezení. Je definován v RFC 3551 [11]. Na úrovni RTP není k dispozici žádná funkcionalita pro kontrolu členství, ale může být poskytována nějakým protokolem na vyšší vrstvě. Tento druh aplikací je velmi blízký základní definici RTP protokolu, proto je v profilu minimální počet modifikací. Následující tabulka zobrazuje typy obsahu, které jsou využívány v tomto profilu.

PT	Jméno	Typ	Reference
0	PCMU	Audio	RFC 3551

1	1016	Audio	RFC 3551
2	G721	Audio	RFC 3551
3	GSM	Audio	RFC 3551
4	G723	Audio	
5	DVI4	Audio	RFC 3551
6	DVI4	Audio	RFC 3551
7	LPC	Audio	RFC 3551
8	PCMA	Audio	RFC 3551
9	G722	Audio	RFC 3551
10	L16	Audio	RFC 3551
11	L16	Audio	RFC 3551
12	QCELP	Audio	RFC 2658, RFC 3551
13	CN	Audio	RFC 3389
14	MPA	Audio	RFC 2250, RFC 3551
15	G728	Audio	RFC 3551
16	DVI4	Audio	RFC 3551
17	DVI4	Audio	RFC 3551
18	G729	Audio	RFC 3551
25	CellB	Video	RFC 2029
26	JPEG	Video	RFC 2435
28	nv	Video	RFC 3551
31	H261	Video	RFC 4587
32	MPV	Video	RFC 2250
33	MP2T	Audio/Video	RFC 2250
34	H263	Video	RFC 3551, RFC 2190

Tabulka 3.1: Přehled typů obsahu pro multimediální formáty

3.3.2.2 Nemultimediální rozšíření

Přestože je protokol RTP primárně určen pro přenos multimediálních dat existuje několik rozšíření pro přenos dat, na které lze pohlížet jako by byly vysílány v proudu a lze je tak pomocí RTP přenášet.

Typ obsahu pro textovou konverzaci

Typ obsahu pro textovou konverzaci je definován v RFC 2793 [12] a umožňuje přenášet textovou konverzaci v reálném čase. Text může být přenášen buď samostatně, nebo spolu s videem a zvukem,

které jsou přenášeny ve vlastních proudech. Je předpokládané, že je text zadáván ručně pomocí nějakého vstupního zařízení, takže počet přenášených znaků je poměrně malý.

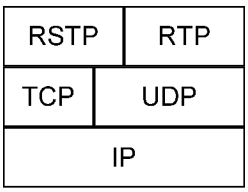
Typ obsahu pro ukazatele v reálném čase

Tento typ je definován v RFC 2862 [13] a slouží k přenosu ukazovátka například ve video konferencích.

3.4 Popis protokolu RTSP

RTSP (Real Time Streaming Protocol) je síťový protokol určený pro řízení serverů, které vysílají multimediální obsah. Protokol byl vyvinut uskupením Multiparty Multimedia Session Control Working Group v roce 1998 a publikován v RFC 2326 [14]. Umožňuje obousměrnou komunikaci mezi serverem a klientem a jeho hlavním úkolem je ustanovení spojení, nastavení parametrů a poté ovládání přehrávání. Samotný přenos multimediálních dat však ve většině případů není v režii protokolu RTSP, ale některého specializovaného protokolu. K přenosu dat je velmi často používán právě protokol RTP, ale někteří výrobci implementují své vlastní proprietární protokoly.

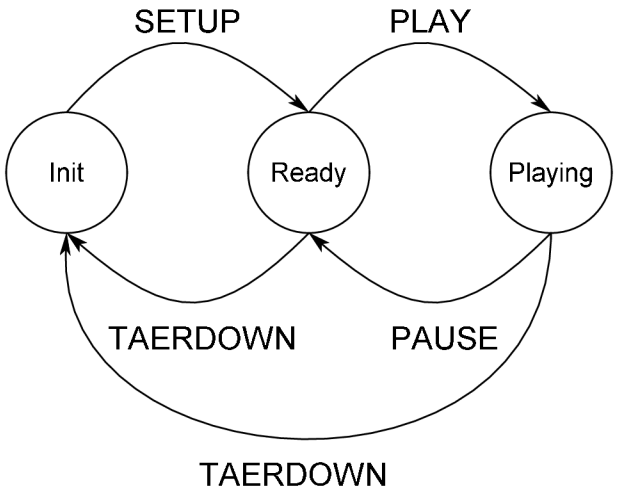
RTSP je založen na protokolu HTTP. Od něj převzal syntaxi zpráv i chování a přidal několik typů požadavků, které mohou být odeslány a které jsou potřebné pro řízení multimediální relace. Na následujícím obrázku je znázorněno v jakém vztahu je protokol RTSP k ostatním protokolům v TCP/IP modelu.



Obrázek 3.10: Protokoly v modelu TCP/IP

3.4.1 Rozdíly oproti protokolu HTTP

Hlavním rozdílem mezi RTSP a HTTP je ten, že každá relace udržuje stav, zatímco protokol HTTP je typickým zástupcem bezstavových protokolů. Na dalším obrázku je stavový diagram typického klienta využívajícího protokolu RTSP.



Obrázek 3.11: Stavový diagram RTSP protokolu

Další rozdíly jsou níže shrnuty pouze v bodech, jejich detailnější popis se v případě potřeby vyskytuje v kapitolách zabývajících se využitím protokolu RTSP při řízení relace přenosu procesních dat.

- Oproti HTTP obsahuje protokol několik nových metod a má jiný identifikátor protokolu v hlavičce zprávy.
- Poskytuje oboustranné spojení, tudíž může požadavky zasílat i server klientovi.
- Data jsou většinou přenášena jiným protokolem.
- Definice RTSP doporučuje používání kódování UTF-8 (ISO 10646).

3.4.2 Popis struktury zprávy

RTSP zpráva se skládá z hlavičky a z těla zprávy odděleného znaky CRLF, které však nemusí být ve všech zprávách obsaženo. Existují dva druhy zpráv, které mají mírně odlišný tvar stavové (první) řádky a to požadavek a odpověď. Obecný tvar zprávy požadavku je následující:

{Jméno metody}{URL}{Verze protokolu}CRLF
{parametry}
CRLF

Zpráva typu odpověď má tvar:

{Verze protokolu}{Stavový kód}{Fráze}CRLF
{parametry}
CRLF

Každá zpráva musí v poli parametry obsahovat parametr CSeq, což je sekvenční číslo požadavku v dané relaci. Pomocí něj jsou spárovány požadavky a odpovědi na ně. Stavové kódy u odpovědi odpovídají protokolu HTTP s tím, že protokol RTSP přidává některé své vlastní kódy. Příjemce kódu není povinen rozumět konkrétnímu kódu, ale musí rozumět skupině, do které přijatý kód patří. Skupiny kódů jsou shrnuty v následující tabulce.

Skupina	**Jméno**	**Popis**
1xx	Informační	Požadavek byl přijat a pokračuje jeho zpracování.
2xx	Úspěch	Požadavek byl přijat, rozpoznán a akceptován.
3xx	Přesměrování	Pro splnění požadavku musí být provedeny další akce.
4xx	Chyba klienta	Chyba v syntaxi požadavku.
5xx	Chyba serveru	Došlo k chybě během zpracování požadavku.

Tabulka 3.2: Chybové kódy RTSP

Pokud je ve zprávě spolu s hlavičkou obsaženo i tělo, jeho velikost je uložena v parametru Content-Length. Zpráva typu odpověď nesmí obsahovat žádné tělo a je ukončena za první prázdnou řádkou.

3.4.3 Metody

Protokol RTSP definuje několik nových metod, které mohou být použity pro řízení relace. Následující tabulka obsahuje jejich názvy, směr, ve kterém jsou zasílány a nutnost implementace.

Jméno	Směr	Požadováno
DESCRIBE	$C \rightarrow S$	Doporučeno
ANNOUNCE	$C \rightarrow S \quad S \rightarrow C$	Volitelné
OPTIONS	$C \rightarrow S \quad S \rightarrow C$	Vyžadováno
PLAY	$C \rightarrow S$	Vyžadováno
PAUSE	$C \rightarrow S$	Doporučeno
RECORD	$C \rightarrow S$	Volitelné
REDIRECT	$S \rightarrow C$	Volitelné
SETUP	$C \rightarrow S$	Vyžadováno
TEARDOWN	$C \rightarrow S$	Vyžadováno
GET_PARAMETER	$C \rightarrow S$	Volitelné
SET_PARAMETER	$C \rightarrow S \quad S \rightarrow C$	Volitelné

Tabulka 3.3: Metody protokolu RTSP

Jak vyplývá z tabulky, ne všechny servery musí implementovat všechny metody. V případě, že klient požaduje metodu, která není implementována, měl by server vracet hodnotu „501 Not Implemented“ a klient by se již neměl znovu pokoušet tuto metodu vyžadovat.

DESCRIBE

Metoda DESCRIBE získá ze serveru popis objektu na zadané URL. Popis je ve formátu protokolu SDP, kterému je věnována samostatná podkapitola. Tyto informace jsou pro funkčnost podstatné, nicméně specifikaci nepřikazuje jejich získání pomocí metody DESCRIBE. Je možné je získat například stažením z webu pomocí protokolu HTTP, nebo klientovi tyto informace předat jako jeho konfiguraci.

ANNOUNCE

Umožňuje serveru zasílat popis objektu klientovi v reálném čase a reagovat tak na případné změny v objektu.

OPTIONS

Server na požadavek OPTIONS vrací seznam metod, které jsou na serveru dostupné.

PLAY

Spustí přehrávání dat ze serveru. Klient nesmí odesílat požadavek PLAY, dokud má nějaké nepotvrzené požadavky SETUP. Jako parametr může být vložen časový rozsah, který má být přehrán.

REDIRECT

Požadavek informuje klienta, že se musí přepojit na jiný server. Pokud chce klient nadále dostávat data od tohoto serveru, musí vyslat požadavek TEARDOWN, ukončit relaci a zahájit nové.

SETUP

Metoda slouží k nastavení parametrů přenosu dat. Jednak před zahájením přenosu, ale klient může vyslat požadavek na změnu těchto parametrů i u přenosu, který byl již zahájen.

TEARDOWN

Metoda ukončí přehrávání ze zadané URI. Jsou uvolněny všechny zdroje serveru alokované pro příslušnou relaci.

GET_PARAMETER

Metoda slouží k získávání parametrů ze serveru. Jako odpověď je vrácena hodnota parametru objektu, specifikovaném v URI požadavku.

SET_PARAMETER

Metoda umožňuje nastavit parametr objektu, který je specifikován v URI. Každý požadavek SET_PARAMETER by měl obsahovat nastavení pouze jednoho parametru, aby bylo možné v případě selhání identifikovat, který parametr se nepodařilo nastavit.
Metoda by neměla sloužit k nastavování vlastností samotného přenosu dat. K tomuto účelu je vyhrazena metoda SETUP.

3.4.4 Typy spojení

RTSP může být přenášen několika způsoby. V případě požadavku na obousměrné spojení je vytvořeno persistentní spojení, které je poté využíváno pro všechny požadavky a odpovědi, které jsou mezi klientem a serverem poslány. Další možností je vytvoření spojení pro každý pár požadavek - odpověď. Poslední možností je přenos požadavků bez vytváření spojení.
Typ spojení je definován v RTSP URI, v případě, že je jako schéma použito „rtsp“, je předpokládáno persistentní spojení. V případě použití „rtspu“, je použita komunikace bez vytváření spojení.

3.4.5 RTSP URL

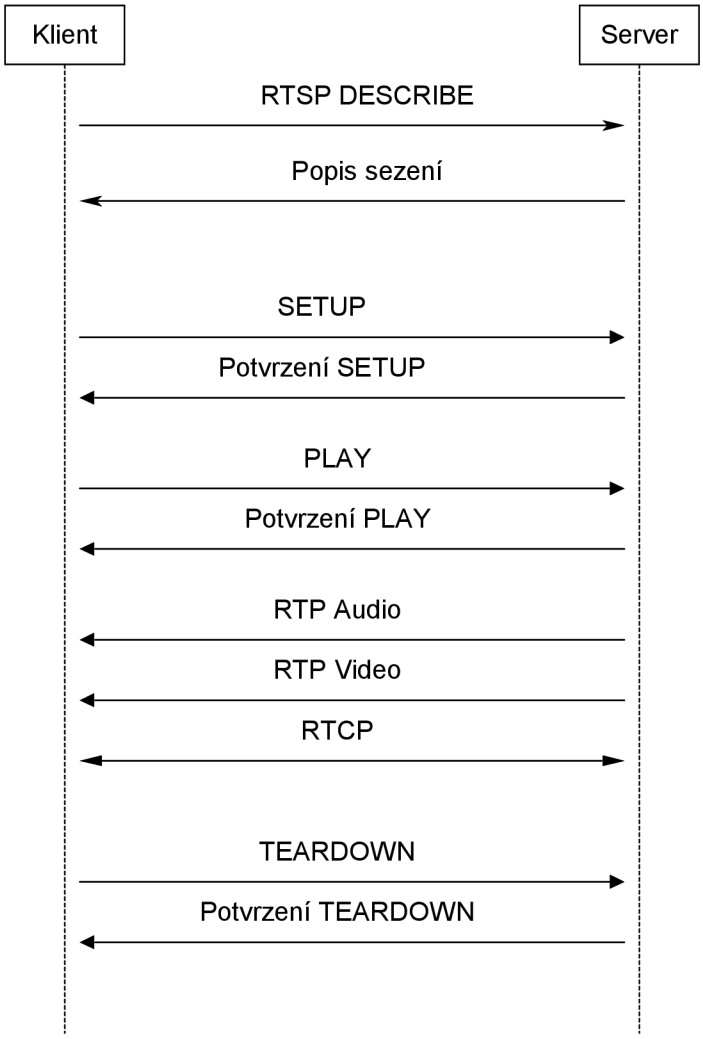
Pro adresaci zdrojů na síti jsou v RTSP použity schémata „rtsp“ a „rtspu“. Adresa může mít jako cíl konkrétní multimediální zdroj nebo agregaci více takových zdrojů. Interpretace je ponechána na serveru. URL má následující tvar:

(rtsp: | rtspu:)//{ jméno hosta }[: port][absolutní cesta]

Pod jménem hosta se skrývá hostitelské jméno cílového serveru. Je preferována textová reprezentace, používání ip adresy se nedoporučuje. V případě, že není specifikováno číslo portu, je použit port 554. Absolutní cesta je textová identifikace požadovaného objektu. Nemusí se jednat o konkrétní objekt, ale jak již bylo výše uvedeno o skupinu více objektů. Tento fakt dovoluje velmi snadné vytváření hierarchie objektů a jejich přesnou adresaci.

3.4.6 Příklad spojení RTSP

Na obrázku níže je uveden typický průběh RTSP relace. Nejdříve je získán popis dostupného média ze serveru pomocí metody DESCRIBE. Poté je nastaveno spojení a spuštěno přehrávání. Přenos dat je již plně v režii protokolu RTP a RTCP. Po ukončení přenosu je odeslán požadavek TEARDOWN, který na serveru uvolní zdroje a ukončí relaci.



Obrázek 3.12: Příklad komunikace RTSP

3.4.7 Rozšiřitelnost RTSP

RTSP je navržen tak, aby jej bylo možné snadno rozšířit. První možností, jak upravit chování konkrétní implementace, je vynechání některých metod. V případě, že není umožněn záznam, tak server neimplementuje metodu RECORD. Stejně tak nemusí každý server podporovat nastavování pomocí metody SET_PARAMETER, nebo pozastavení vysílání metodou PAUSE.

V těchto případech se však nejedná o rozšíření, ale pouze o úpravu chování serveru vynecháním části funkčnosti. Skutečného rozšíření je možné docílit třemi způsoby.

1. Existující metody mohou být rozšířeny o nové parametry. Tyto parametry však musí být možno bezpečně ignorovat na straně příjemce.
2. Mohou být přidány nové metody. V případě, že příjemce metodu nezná, odpoví kódem 501. Klient může získat seznam podporovaných metod pomocí metody OPTIONS.
3. Může být definována nová verze protokolu. To umožní změnit téměř všechny aspekty chování. Kromě pozice čísla protokolu v hlavičce.

3.5 Další technologie

V této kapitole se nachází stručný popis technologií, které se netýkají přímo přenosu dat, ale slouží například k jejich popisu, nebo získávání.

3.5.1 SDP

SDP (session description protocol) slouží k popisu multimediálních relací v textové formě. Původní verze byla vydána organizací Internet Engineering Task Force v roce 1998. Revize protokolu, která je aktuální až do dnešních dnů, byla vydána v roce 2006 jako RFC 4566 [15]. V popisu relace se nachází množina vlastností a parametrů, které představují profil relace. Původně byl SDP použit jako součást SAP protokolu, ale je využíván v mnoha dalších případech. Pro naše účely je nejzajímavější možnost použití spolu s RTSP protokolem pro inicializaci relace a pro předání požadovaných informací mezi RTSP a RTP serverem.

Na každém řádku SDP popisu se nachází dvojice atribut - hodnota. U nepovinných parametrů následuje za znakem '=' ještě znak '*'. Protokol SDP byl navržen tak, aby bylo možné přidávat další atributy, a tak rozšiřovat jeho popisné schopnosti na další nově vzniklé typy multimediálních formátů. Pořadí atributů v popisu je pevně dané a je popsáno níže.

```
      Popis relace
         v=  (verze protokolu)
         o=  (původce a identifikátor relace)
         s=  (jméno relace)
         i=* (informace o relaci)
         u=* (URI popisu)
         e=* (emailová adresa)
         p=* (telefonní číslo)
         c=* (informace o spojení)
         b=* (informace o šířce pásma)
         z=* (časová zóna)
         k=* (klíč kódování)
         a=* (nula až n řádek popisujících dostupné medium)
         t=  (čas popisující začátek a konec relace)
         r=* (čas opakování relace)
         m=  (jméno a adresa vztažená k médiu)
         i=* (Titulek)
         c=* (informace o spojení (volitelné, pokud je dostupné v sekci
popisu relace))
         b=* (nula, nebo více řádku s popisem šířky pásma)
         k=* (klíč kódování)
         a=* (nula až n řádek popisujících dostupné multimedium)
```

3.5.2 XML

Extensible Markup Language (XML) je obecný značkovací jazyk, který byl vyvinut a standardizován konsorciem W3C. Umožňuje snadné vytváření konkrétních značkovacích jazyků pro různé účely a různé typy dat. Jazyk je určen především pro výměnu dat mezi aplikacemi [16]. XML slouží k popisu informací, které jsou v dokumentu obsaženy. Příklad krátkého XML dokumentu je uveden níže. Jedná se o konfiguraci klientů z demonstrační aplikace této práce.

```
<?xml version="1.0" encoding='UTF-8'?>
<users>
      <user login="tomas" password="heslo" id="100"/>
</users>
```

3.5.3 OPC

OLE for Process Control (OPC) je skupina standardů popisující OLE/COM protokoly a rozhraní, jejichž účelem je zvýšení interoperability mezi automatizačními/řídícími aplikacemi při řízení průmyslových procesů [17]. Pro přístup k živým datům slouží skupina rozhraní definovaných v OPC Data Access. Klient požadující data se připojí k tzv. OPC serveru, který pomocí definovaných rozhraní poskytuje data, aniž by klient musel znát detaily zařízení, které existují za OPC serverem a jejichž data server poskytuje.

4 Návrh distribuce procesních dat pomocí RTP

V této kapitole je navržen způsob, jakým je možné distribuovat procesní data pomocí proudového přenosu dat protokolem RTP a dalších protokolů, které jsou k tomu zapotřebí. Protože stejně jako v multimediálních aplikacích nestojí protokol RTP samostatně a k realizaci přenosu jsou použity i jiné prostředky, tak i při tomto druhu přenosu je nutné využít služeb dalších protokolů. Postupně jsou zde rozebrány jednotlivé problémy takového přenosu a navrženo jejich řešení, postavené na protokolu RTP.

4.1 Správa relace

Jelikož protokol RTP neobsahuje žádný mechanismus, který by klientovi umožnil výběr dat o které má zájem a ovládání samotného přehrávání, je nutné najít cestu, která by to umožňovala. Pro řízení interaktivních relací lze využít například protokoly H.323, nebo SIP. Při vysílání neinteraktivních multimediálních dat se k tomuto účelu používá dříve popsaný protokol RTSP [7]. Ten obsahuje metody jak pro výběr dat k přehrání tak k řízení relace. Protokol jako takový není nutné nijak rozšiřovat, protože obsahuje veškerou potřebnou funkcionalitu. Pouze nebude implementována metoda RECORD, která v případě vysílání nedává smysl.

Způsob komunikace a předávání informací od klienta mezi RTSP serverem a RTP serverem je záležitostí implementace a bude popsán v kapitole zabývající se návrhem aplikace. V praxi se pro předání informací o klientem vyžádaném proudu dat využívá již zmíněný protokol SDP[7]. Ten by měl předat dostatek informací pro zahájení vysílání.

4.2 Meta informace

Mimo samotných procesních dat existuje v aplikaci skupina informací, které jsou převážně statické a popisují distribuovaná data. Jedná se o takzvané meta informace, které mohou obsahovat například datový typ, textový popis přenášených dat a jiné aplikačně závislé položky. Informace mohou být klientovi dostupné lokálně ve formě jeho konfigurace, nebo musí existovat mechanismus, který mu je umožní získat.

Zásadní nevýhodou lokálního uložení je obtížná distribuce změny v těchto datech. Proto je vhodná pouze pro aplikace, u kterých je pravděpodobné, že ke změně nebude docházet. Nebo při kombinaci s mechanismem, který o změnách klienty informuje a umožní získání verze nové. Protože se data příliš často nemění, většinou se jedná o jednorázový přenos směrem ke klientovi. Tento fakt umožňuje mít meta informace uložené na serveru a klientovi je při připojení poskytnout.

Použití stejného kanálu, který je použit pro procesní data, není praktické. Meta informace jsou poskytnuty každému klientovi zvlášť a musí být doručeny spolehlivě. Pro tento přenos je možné využít protokolu RTSP a SDP. SDP poskytuje informace o umístění meta informací a metoda GET protokolu RTSP zase umožní klientovi o informace server požádat.

Způsob uložení a formát poskytnutých meta informací je opět záležitostí konkrétní aplikace. Vhodnou technologií pro tento účel je například XML. Pro přenos navržený v této práci je minimální obsah meta informací následující.

```
<?xml version="1.0" encoding='UTF-8'?>
<data>
      <item id="" type=""/>
</data>
```

Atribut id obsahuje adresu (identifikátor) hodnoty a atribut type její datový typ.

4.3 Řešení ztráty paketů

UDP narozdíl od TCP, poskytuje nespolehlivý přenos dat. To znamená, že v průběhu přenosu dat mezi dvěma uzly může dojít ke ztrátě celého paketu a je na aplikaci, která měla data obdržet, aby se se ztrátou dat vypořádala. V případě, že není možné ztrátu zanedbat, je nutné použít nějakou metodu pro řešení ztrát paketů. Tyto metody můžeme rozdělit do dvou kategorií. První je nějaká forma znovuzasílání ztracených dat, tato metoda je použita v protokolu TCP. Druhá metoda je dopředná korekce chyby. V této podkapitole jsou obě kategorie rozebrány a jsou diskutovány jejich výhody a nevýhody pro použití při přenosu živých dat.

4.3.1 Důvody vzniku

Ke ztrátám paketů přispívá velké množství faktorů. Může dojít k poškození dat v paketu vlivem nekvalitní přenosové linky a ten je pak nepoužitelný, k zahození paketu díky zahlcení linky, chybě v hardwaru na některém ze síťových zařízení atd. Ztráty lze rozdělit na dvě kategorie, podle toho jakým způsobem je lze řešit. První kategorií jsou ztráty, které vznikají v důsledku zahlcení linky. Ty jsou z hlediska odesílatele řešitelné pouze přizpůsobením datového toku možnosti přenosové sítě. Druhou kategorií jsou chyby vzniklé nekvalitní linkou, chybným směrováním apod. Ty je možné řešit některou z metod pro opravu ztrát paketů, které jsou popsány níže.

4.3.2 Znovuzasílání dat

Nejspolehlivějším způsobem pro dosažení spolehlivého přenosu je znovuzasílání ztracených dat. Aby to bylo možné, je nutné, aby byl k dispozici kanál ve směru klient- server, pomocí něhož jsou zasílány informace o ztrátách paketů. Existují dva způsoby, jakým je o znovuzaslání žádáno. První možností je potvrzování každého přijatého paketu. Druhou možností je zasílat jen žádosti o pakety, které nedorazily. Druhá možnost dává příjemci výhodu v možnosti rozhodnout se, zda jsou pro něj chybějící data nutná, nebo je může postrádat a o znovuzaslání tedy nepožádá. V aplikacích, které přenáší data s tolerancí vůči ztrátám, může být druhá možnost vhodnějším přístupem, který pak vede k nižším nárokům na přenosovou kapacitu. Nevýhodou u obou přístupů je nutnost uchovávat po nějaký čas zaslané pakety na serveru pro případ že by o ně klient požádal, respektive případ kdy by nebyl příjem klientem potvrzen.

Zpětný kanál pro přenos žádosti, nebo potvrzovacích informací není v RTP protokolu obsažen. Je však možné využít zpětného kanálu, který poskytuje protokol RTCP k žádosti o znovuzaslání ztracených dat. Pro využití protokolu RTCP je nutné definovat formát paketů, který pro tento účel bude využit a dále upravit časovací pravidla tak, aby byla možná okamžitá zpětná vazba.

4.3.2.1 Znovu zasílání v přenosu typu multicast

IP multicast je velmi efektivní cesta, jak dostat velké množství dat mnoha klientům z jednoho zdroje. Díky použití protokolu UDP je však přenos nespolehlivý a použití techniky znovu zasílání ztracených paketů obtížnější než v přenosu typu unicast. Spolehlivému přenosu přes multicast se věnuje mnoho

prací, například [18] nebo Pragmatic General Multicast definovaný v RFC 3208 [19]. Na tomto místě jsou jen nastíněny problémy, se kterými se takový přenos potýká a některé způsoby jejich řešení.
U přenosu typu unicast klient sám inicializuje opravu případné ztráty dat. U přenosu typu multicast je však velké množství klientů a každý může postrádat jiné pakety. Každý takový klient vysílá požadavek na opravu a server poté zašle chybějící data, která dorazí ke všem klientům, tedy i k těm, kteří je nepotřebují. Při rostoucím množství klientů může dojít k zahlcení sítě datovým tokem určeným pouze k opravě chybějících paketů. K řešení tohoto problému existuje několik různých přístupů. Žádný z nich však neposkytuje ideální metodu, která je použitelná ve všech případech. Některé z přístupů lze i vzájemně kombinovat.

Potlačení NACK zprávy
Zprávu NACK zasílá klient serveru a oznamuje mu, že nedorazil některý paket. Jedná se o takzvané negativní potvrzení. Normálně každý klient, v případě, že nedorazil některý paket, pošle serveru zprávu NACK. Při tomto přístupu však klient neposílá zprávu serveru, ale do multicastové skupiny. Ostatní klienti tak nemusí, v případě že tuto zprávu obdrželi, již zasílat svoje žádosti a čekají náhodný čas, zda chybějící paket dorazí. V případě velkého množství klientů však stejně může dojít k zahlcení.

Hierarchické uspořádání klientů
Klienti jsou seřazeni do stromu se serverem v kořenu. Každý uzel je zodpovědný za spolehlivý přenos pouze svým potomkům, což umožňuje snížení počtu NACK zpráv. Bohužel vytvoření a udržování takovéhoto stromu je velmi problematické. V případě statického stromu může mít ztráta některého uzlu katastrofální následky. Dynamický strom je zase nestabilní v případě, že se množina klientů hodně mění. Dalším problémem je výběr vhodných uzlových klientů. Některé klienty nelze použít, například z důvodu nízkého výkonu nebo kapacity linky.

Dotazování
U tohoto přístupu je vysílání žádosti o znovuzaslání řízeno serverem. Každý klient vygeneruje svůj vlastní náhodný klíč. Server poté vyšle dotaz, který obsahuje klíč a hodnotu, která značí, kolik bitů z klíče klienta musí souhlasit s klíčem v dotazu, aby mohl klient odeslat požadavek. V případě velkého množství klientů však může i tak dojít k zahlcení nebo k velmi velkému zpoždění opravy.

4.3.2.2 Použitelnost pro přenos procesních dat

Znovu zasílání ztracených dat poskytuje spolehlivý způsob přenosu dat. Metoda je výhodná při komunikaci dvou účastníku, zejména pak, pokud je vyžadován skutečně spolehlivý přenos, při kterém se netolerují žádné ztráty. Pokud se však jedná o komunikaci typu multicast, je použití této metody problematické a pro živá data jsou použitelné jen některé metody. Další nevýhodou je, že znovuzaslání dat vnáší do přenosu dat časové zpoždění, které může u přenosu živých dat zapříčinit to, že znovu zaslaná data jsou již zastaralá. Proto je u přenosu živých dat výhodnější použít metodu dopředné korekce chyb, která umožní rekonstrukci ztracených dat s menší časovou ztrátou.

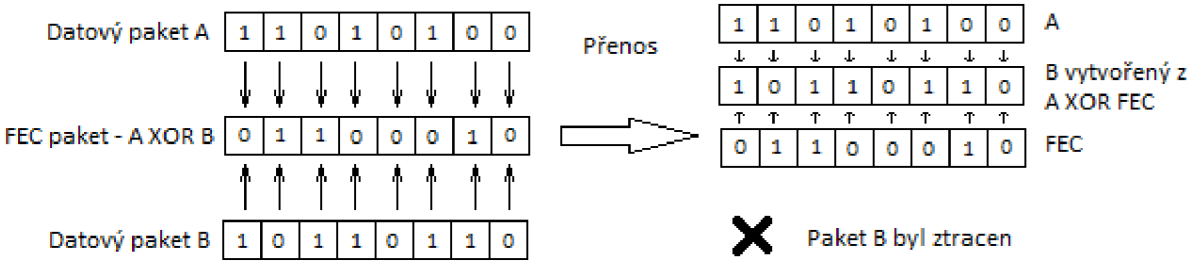
4.3.3 Dopředná korekce chyb

Metoda je založena na přidávání redundantních dat do proudu odesílaných dat, pomocí kterých je možné v případě potřeby chybějící data rekonstruovat. Výhodou tohoto postupu je, že množství přidávaných dat je škálovatelné. Je tak možné v případě sítě, ve které dochází k velkým ztrátám, přidávat větší množství opravných dat, nebo naopak v případě malých ztrát tyto nadbytečná data omezit a nezatěžovat tak přenosové médium. Informaci o množství ztrát může odesílatel získat od příjemce, například pomocí protokolu RTCP.

Rekonstrukce dat je možná pouze v případě, že má příjemce k dispozici dostatek dat, tedy pokud nedochází k větším ztrátám, než pro jaké byla vložena opravná informace. Nevýhodou je, že v případě většího množství příjemců s různě ztrátovým připojením může některý příjemce dostávat nadbytečné množství opravných informací a tak zbytečně zatěžovat přenosové médium. Další nevýhodou je, že v případě poškozených dat je nutné čekat, až dorazí data potřebná k jejich opravě. Tím může vznikat nežádoucí zpoždění, které je problematické především u přenášení živých dat.
U multimediálních dat jsou tyto metody často postavené na vlastnostech přenášených dat, ale některé principy lze přenést i pro přenos živých dat. V této části jsou popsány základní přístupy pro využití dopředné korekce chyb (dále jen FEC – Forward Error Correction) u přenosu živých dat.

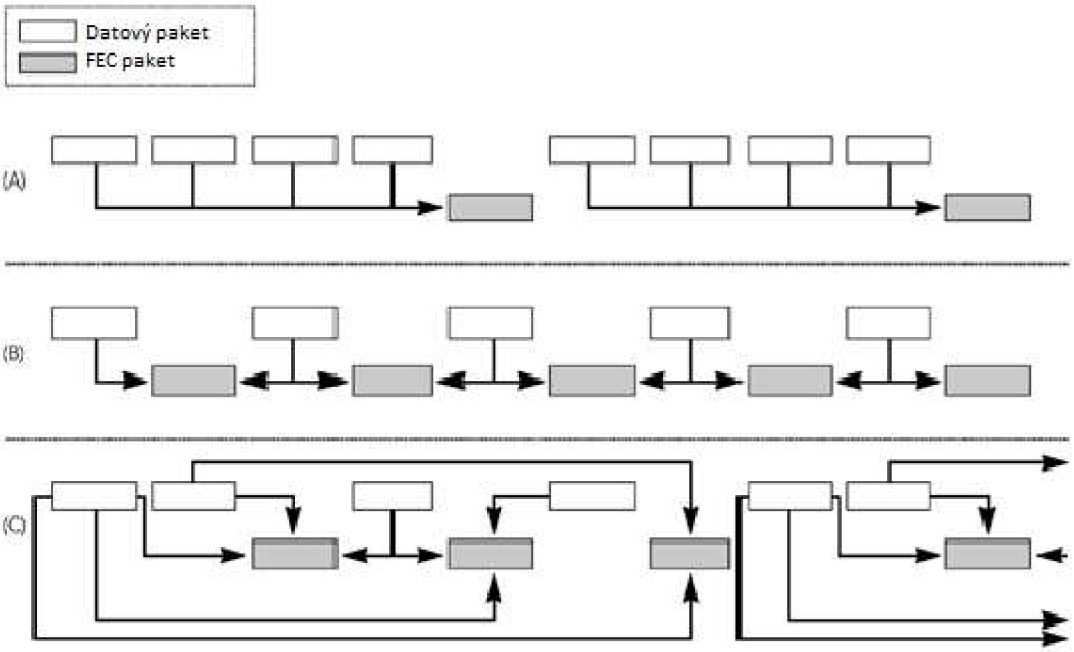
Paritní FEC

Nejjednodušší způsob zavedení redundantního kódu je použití operace XOR na dva, nebo více paketů [7]. Takto vzniklý paket je odeslán spolu s původními daty. V případě ztráty některého z datových paketů je možné jej rekonstruovat s použitím paritního paketu. Tento postup prezentuje následující obrázek.



Obrázek 4.1: Příklad opravy ztraceného paketu. Zdroj [7].

Pro posílání redundantních paketů je definován speciální typ obsahu v RFC 2733 [20]. Ten umožňuje spárování datových paketů a k nim příslušejících paketů redundantních. Schopnost klienta rekonstruovat ztracená data závisí na množství ztracených paketů a na množství obdržených redundantních dat. Na následujícím obrázku jsou některá možná schémata odesílání paketů, jak jsou uvedena v knize [7]



Obrázek 4.2: Příklad možných schémat odesílání FEC paketů. Zdroj [7].

Vrstvený FEC
Možností jak se vypořádat s klienty, kteří mají výrazně odlišné nároky na množství redundantních dat je vytvořit více samostatných FEC proudů tak, aby si klienti mohli vybrat, které budou přijímat. Různé proudy budou obsahovat různé množství redundantních dat. Výhodou je, že klienti, kteří nepodporují FEC, mohou přijímat pouze zdrojová data a ostatní proudy ignorovat. Tento princip je detailněji popsán v podkapitole zabývající se klienty s různou rychlostí připojení. Velmi podrobně je metoda popsána pro přenos videa pomocí přenosu typu multicast v [21].

Adaptivní FEC
Velikost datového toku lze optimalizovat pomocí tzv. Adaptivního FEC. Ten přizpůsobuje množství redundantních informací aktuálnímu stavu linky – množství ztracených paketů. Informace o něm musí serveru poskytovat klient. Při přenosu typu multicast je však jeho použití velmi problematické, obzvláště pokud se jedná o neheterogenní síť, kde jsou někteří klienti připojeni přes výrazně horší připojení. Ti zvyšují množství redundantních informací i těm klientům, kteří mají připojení s minimálními ztrátami paketů. Rozbor tohoto tématu výrazně překračuje rozsah této práce.

Výběr metody
Popsané metody umožňují snížit pravděpodobnost ztráty paketu, ale nikdy ji nemůžou úplně vyloučit. Proto jsou vhodné pouze pro aplikace, které jsou ke ztrátám tolerantní, nebo v kombinaci s metodou potvrzování, kde slouží k omezení žádostí o znovuzaslání.

Rozhodnutí, která metoda bude použita, závisí na typu aplikace. Pro aplikaci, ke které jsou připojení klienti s různými typy připojení, může být výhodné použití vrstveného modelu. Pro aplikaci využívající přenos typu unicast zase adaptivní model.

4.3.4 Perioda zasílání dat

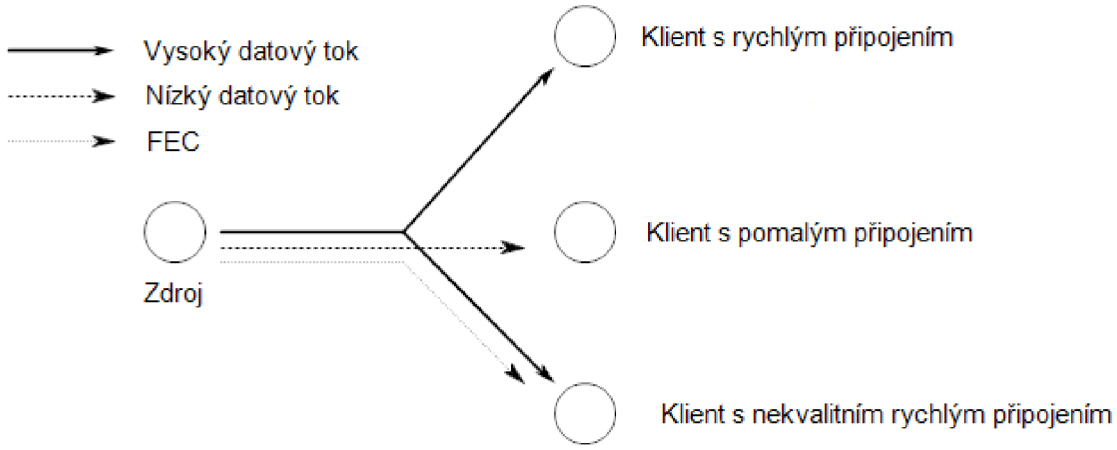
Pokud není použita skutečně spolehlivá metoda přenosu pomocí potvrzování, je nutné zajistit, aby se v případě, že dojde ke ztrátě paketu obsahujícího hodnotu položky, která se mění v dlouhém

intervalu, ke klientovi tato změna dostala v akceptovatelném čase. Server tedy musí hodnotu zasílat periodicky znovu. Interval v jakém musí dojít k znovuzaslání hodnoty je však odvislý od velkého množství protikladných požadavků, například dostupná přenosová kapacita, charakter dat, časové požadavky a je nutné ho pro každý typ aplikace stanovit zvlášť.

Pro optimalizaci množství přenášených dat je možné stanovit interval pro odeslání hodnot individuálně. U méně důležitých hodnot nastavit interval vyšší a u hodnot důležitých nižší.

4.3.5 Klienti s různou rychlostí připojení

Pokud je klient připojený přes připojení schopné přenést celý poskytovaný datový tok dochází ke ztrátám. Pokud to charakter aplikace a dat umožňuje, je možné vysílat data s nižším datovým tokem do jiné multicastové skupiny a klienti si sami vyberou nejvhodnější kombinaci datového toku a případné FEC. Následující obrázek ilustruje klienty s různými požadavky na datový tok a FEC.



Obrázek 4.3: Poskytování různých proudů dat různým klientům

Dostupné kanály musí být nějakým způsobem oznámeny. Jednou z možností je využít SDP protokol, ve kterém bude u popisu přenášeného média i seznam dostupných kanálů.

4.4 RTP profil

RTP profil definuje chování protokolu RTP pro daný typ použití. V této podkapitole je navrhnut RTP profil pro přenos živých procesních dat.

Je použita standardní hlavička RTP a nejsou potřebné žádné externí hlavičky. Typ obsahu je definován v samostatné podkapitole. Dále není potřeba definovat žádné další typy zpráv protokolu RTCP. V případě, že není použita žádná metoda pro úpravu datového toku na základě ztrát paketů, není nutné, aby klienti informace o ztrátách odesílali. Pokud použita je, jsou časové intervaly odesílání definované v RFC vyhovující. Také není potřeba, aby byly protokolem RTCP přenášeny informace o zdroji dat. Ty budou klientům buď známé, nebo budou přeneseny pomocí RTSP. Protokol RTCP bude tedy sloužit pouze k časové synchronizaci více zdrojů vůči společným hodinám, identifikaci vysílaných proudů a distribuci informací o případných výpadcích některého ze zdrojů. V případě použití nějaké formy dopředné korekce chyb je pro její distribuci použito standardního typu obsahu definovaného v RFC 2733 [20].

Odezvy běžných průmyslových sítí se pohybují v řádech milisekund [22], proto je doporučena hodnota pro interval RTP časové známky 1 ms. V případě nutnosti může aplikace definovat vlastní typ obsahu a tuto hodnotu upravit dle svých požadavků.

4.5 Typ obsahu pro procesní data

4.5.1 Struktura dat

Distribuovaná data mohou být velmi rozmanitá a jejich typ a struktura se bude lišit v závislosti na typu aplikace. Jak již bylo na začátku práce řečeno, hlavním cílem jsou nejrůznější měřené a stavové hodnoty. U těchto hodnot je jejich datová velikost poměrně malá, jedná se řádově o jednotky bytů. Následující tabulka shrnuje základní datové typy, které se mohou u aplikací pracujících s procesními daty vyskytovat.

Typ	Velikost (B)	Popis
INT32	4	Celočíselný typ
INT64	8	
FLOAT	4	Reálné číslo
DOUBLE	8	Reálné číslo s dvojitou přesností
BOOL	1	Pravdivostní hodnota
CHAR	1	Znak
ENUM	4	Výčtový typ reprezentující hodnotu 1 z N
MASK	4	Maska reprezentující hodnotu M z N
STRING	-	Řetězec
DATA	-	Obecná datová struktura

Tabulka 4.1: Datové typy

Pro datový typ STRING bylo zvoleno kódování UTF8. Datové typy STRING a DATA obsahují ještě před samotnými daty dva byty, které v sobě nesou údaj o velikosti dat. Problémem těchto datových typů je, že mohou nabývat velikostí, které zcela znemožňují přenos v rámci jednoho paketu, a bylo by nutné je rozdělit do více paketů. V kombinaci s možností ztráty některého z paketů to představuje závažný problém. Tento návrh není pro tento typ dat vhodný.

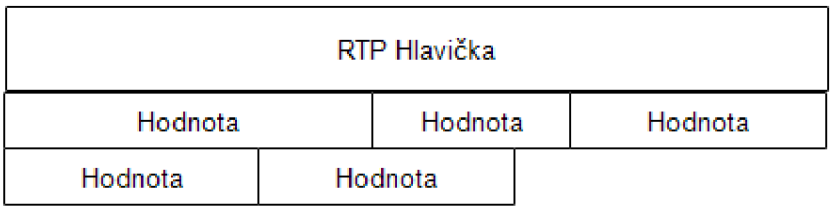
Ke každé přenášené hodnotě je připojen identifikátor, což je jakási adresa určující příslušnost hodnoty k místu jejího vzniku, popřípadě určení. Obecně se může jednat o libovolnou hodnotu, textový řetězec, popřípadě složitější datovou strukturu. Pro přenos po síti je vhodnější jako identifikátor použit celočíselný datový typ, oproti složitějším datovým typům, jehož přenosy by zbytečně zvyšovaly zatížení sítě. Posledním přenášeným údajem je informace o kvalitě hodnoty. Interpretace této hodnoty je záležitostí aplikace. Následující tabulka ukazuje definované hodnoty a jejich předpokládaný význam.

Typ kvality	Význam
OK	Hodnota je v pořádku
INIT	Iniciální hodnota, nejedná se o naměřenou hodnotu apod.
LAST	Poslední známá hodnota. Nejedná se o aktuální hodnotu, ale o hodnotu, která je získána z dočasného úložiště apod.

BAD	Hodnota je nesprávná, například díky výpadku komunikace apod.
UNKNOWN	Není známá kvalita hodnoty.

Tabulka 4.2: Kvalita hodnoty

Na následujícím obrázku je znázorněno uložení dat v RTP paketu.



Obrázek 4.3: Uložení dat v paměti a v RTP paketu

S existencí datových typů je spjat problém s jejich rozpoznáním příjemcem. V zásadě existují dvě možnosti řešení. První je přenos datového typu v hlavičce přenášené hodnoty, tak jak je to znázorněno na obrázku 4.4. Druhou možností je informování o datovém typu jinou cestou. Dá se předpokládat, že konkrétní zdroj hodnot bude poskytovat hodnoty stále stejného datového typu. Je tedy možné každé adrese (identifikátoru) přiřadit datový typ a tuto informaci přenést při inicializaci spojení klientovi.

Obrázek 4.4: Uložení dat v paměti spolu s datovým typem.

Nevýhodou prvního řešení je nutnost s každou hodnotou posílat datový typ, což je vede ke zbytečnému navyšování datového toku. Výhodou je, že případná změna datového typu na straně serveru je jednoduše rozšířena mezi všechny klienty. Na straně klienta však musí existovat mechanismus, který se s takovou změnou dokáže vyrovnat.

V případě druhého řešení je možné přenos datového typu vynechat a snížit tak nároky na přenosovou kapacitu. Díky existenci perzistentního TCP spojení protokolu RTSP je možné pomocí zprávy ANNOUNCE informovat klienta o změně datového typu. Jelikož se časté změny datových typů nepředpokládají, je výhodnější druhé řešení. V meta informacích, které jsou zasílány při inicializaci spojení, je obsažena i informace o datovém typu na všech adresách.

4.5.2 Interval odesílání

Dalším parametrem pro konkrétní typ dat je interval odesílání. Volba intervalu odesílání dat má několik důsledků. S rostoucím intervalem roste zpoždění dat na serveru, které může mít v nejhorším případě až velikost délky intervalu. Roste také množství dat přenášených jedním paketem, takže jeho ztráta má horší důsledky, než ztráta paketu malého. Výhodou je, že klesá množství přenášených režijních dat ve formě hlaviček protokolů.

Při přenosu živých dat je každé zpoždění nežádoucí, proto musí být interval odesílání krátký. Z tohoto hlediska je doporučena hodnota menší než 100 ms pro aplikace které přenáší data pro

lidského pozorovatele. V případě strojového vyhodnocování na klientovi by měla být hodnota ještě nižší v řádu jednotek až desítek milisekund, ale tato doba je velmi závislá na typu aplikace.

4.6 Speciální data

Kromě živých dat je v aplikacích potřeba přenášet další data, pro která není kanál poskytovaný RTP příliš vhodný. V této podkapitole jsou dva takové příklady představeny.

4.6.1 Iniciální data

Při přenosu typu multicast se do multicastové skupiny mohou připojit klienti v kterémkoli okamžiku. Je nutné, aby nově příchozí klient získal co nejdříve všechna data, tedy i ta, která se mění s delší periodou, nebo se nemění vůbec. Pokud je nastavena krátká perioda opakovaného zasílání neměnící se hodnoty, není nutné tento problém řešit. Klient v krátkém čase obdrží všechna data. Pokud je nastavena dlouhá perioda může být čekání na data pro aplikaci nepřijatelné. V takovém případě je možné zaslat celkový aktuální stav dat pomocí jiného kanálu. Tento kanál je poskytován RTSP protokolem a jeho metodou GET_PARAMETER.

4.6.2 Kritická data

Při přenosu procesních dat existuje skupina dat, která lze označit za kritická. Mezi ně patří především řídící data, která musí být vždy doručena. Pro tato data nemůže být použit přenos spolu s ostatními daty. Ale stejně jako u iniciálních dat je možné použít protokol RTSP, tentokrát jeho metodu SET_PARAMETER. Ta umožňuje poslat požadovanou informaci k příjemci spolehlivým způsobem.

4.7 Rozšíření RTSP

Pro potřeby spolehlivého přenosu dat musí být RTSP metody SET_PARAMETER a GET_PARAMETER rozšířeny o schopnost přenést binární data. Obsah dat závisí na konkrétní aplikaci. Pro tyto účely je zaveden nový Content-type:

```
Content-Type: binary/parameters
```

U metody GET_PARAMETER je dále nutné definovat parametr pro celkový obraz dat.

```
actual_data_mirror
```

V případě požadavku na konkrétní hodnotu je metodou GET_PARAMETER zaslán její identifikátor, respektive množina identifikátoru při požadavku na více hodnot a tyto hodnoty jsou pak v odpovědi příjemcem vráceny. Obdobně funguje metoda SET_PARAMETER, která zašle aktuální hodnoty, které mají být nastaveny a příjemce je musí akceptovat dle nastavení přístupových práv.

Využití protokolu RTSP pro přenos řídících informací je postačující v případě, že k tomuto přenosu dochází spíše ojediněle. Nutnost přenosu textové hlavičky přináší oproti specializovaným binárním protokolům velké množství režijních dat. Pokud je to možné doporučuje se přenos více hodnot v jednom požadavku.

Metoda PLAY u běžného použití RTSP obsahuje parametr range, který udává rozsah, který má být RTP serverem přehrán. Při řízení sezení s živými daty tento parametr postrádá smysl a požadavek na přenos určitého rozsahu by měl být vyhodnocen jako chybný a klientovi vrácen chybový kód – 400 BAD REQUEST.

4.7.1 SDP

SDP popisuje dostupné datové zdroje na serveru. Nejdůležitějšími parametry jsou ty, které popisují spojení, přenos a adresu meta informací. Pro popis přenosu živých dat je specifikace SDP vyhovující, pouze je nutné jedno upřesnění a definice parametru pro kontrolu odesílání informací o ztrátách paketů klientem. V parametru u=, který je určen pro přenos URI dokumentu popisující daný zdroj, by měla být URI dokumentu s meta informacemi pro tento zdroj. S tímto URI může klient pomocí RTSP metody GET požádat o meta informace ze serveru.

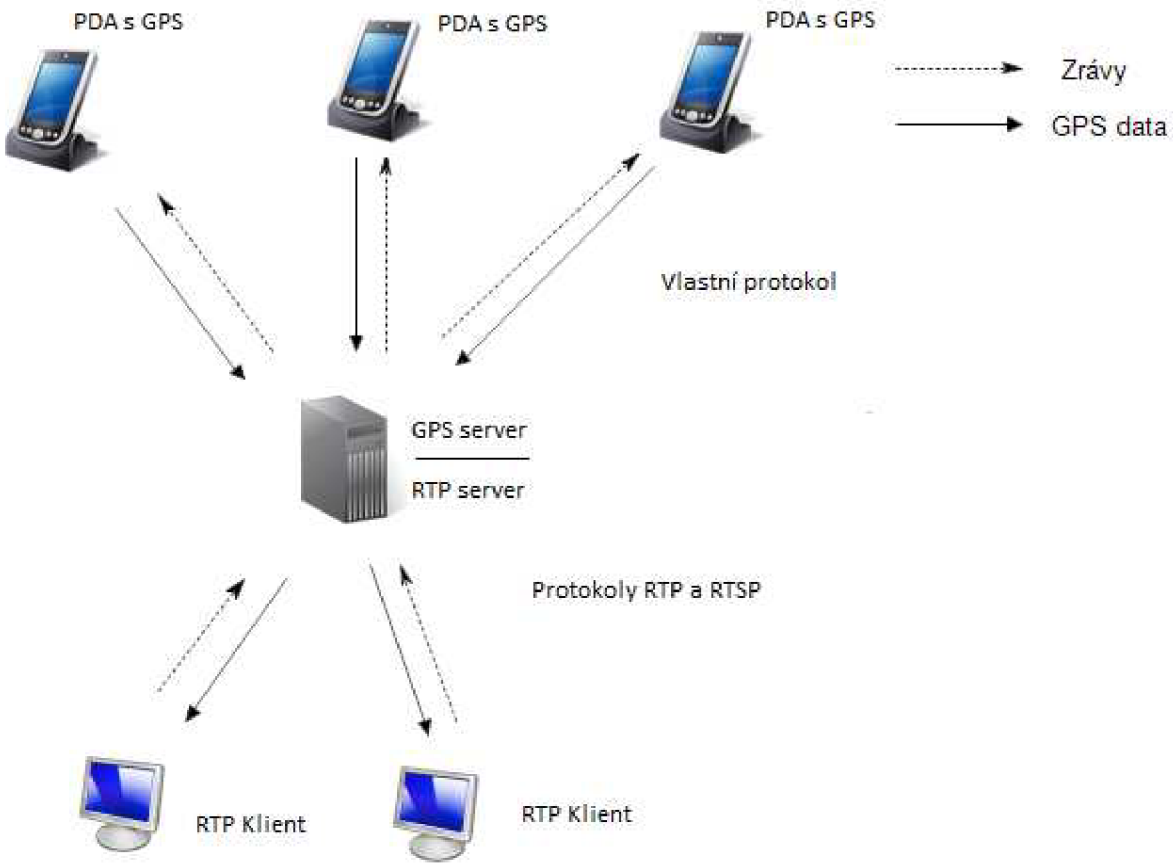
SDP umožňuje dynamické mapování typu obsahu za běhu. Takže je pomocí něj možné klientu oznámit například použití FEC a typ obsahu a kanál, který je pro něj využíván. S kontrolou chyb také souvisí možnost klientovi oznámit, zda má odesílat informace o ztracených paketech. K tomuto účelu je definován parametr:

```
a=rtcppacketlost:[yes|no]
```

Informace získané ze SDP klientovi stačí pro připojení ke zdroji dat. Klient by však měl i v případě přenosu typu multicast inicializovat sezení pomocí metod protokolu RTSP SETUP a PLAY a akceptovat parametry, které jsou vráceny jako odpovědi na tyto požadavky.
Pomocí SDP je také možné přenést statické informace o poskytovateli dat, jako jsou například adresa, email apod.
Pojmenování profilu pro přenos procesních data je definováno jako *„Process Data Profile“ – PDP*.

5 Návrh demonstrační aplikace

Pro demonstraci distribuce pomocí živých dat pomocí RTP byl zvolen přenos GPS souřadnic z mobilních zařízení typu PDA vybavených GPS modulem na server a jejich následná distribuce klientům pomocí RTP. V opačném směru bude možné odesílat krátké textové zprávy jednotlivým zařízením, které se jejich uživatelům zobrazí. Spolupráci částí tohoto systému ilustruje následující obrázek.



Obrázek 5.1: Schéma navrhované aplikace

Výsledná aplikace se bude skládat ze tří částí, které jsou vidět již na obrázku. První částí je program zajišťující zjišťování polohy pomocí GPS modulu a její odesílání na server a naopak příjem textových zpráv ze serveru. Druhou a nejsložitější částí je samotný RTP Server s modulem pro příjem GPS dat, který bude přeposílat GPS data připojeným klientům a textové zprávy od klienta k PDA. Poslední částí je RTP klient, který dokáže přijímat GPS data ze serveru a zobrazovat je na mapě.

V této kapitole je popsán návrh všech tří částí, tak aby bylo možné je implementovat. Postupy zde navržené vychází z předchozích kapitol, které popisují zde použité protokoly a možnosti jejich rozšiřování.

5.1 PDA a sběr dat

Program v PDA plní dvě funkce, první je sběr GPS dat a druhou je komunikace s GPS Serverem. Komunikace mezi serverem a PDA probíhá přes perzistentní TCP spojení pomocí vlastního, velmi jednoduchého protokolu.

5.1.1 Komunikace PDA Server

Pro komunikaci mezi PDA a serverem je navržen jednoduchý binární stavový protokol, který poskytuje velmi jednoduchou autorizaci klienta a možnost přenášet GPS data a zprávy. Následující tabulka obsahuje popis zpráv protokolu.

Zpráva	Směr	Popis
GPSDATA	C → S	Zpráva obsahuje aktuální pozici a informaci o kvalitě.
LOGIN	C → S	Pokus o přihlášení, obsahuje jméno a heslo.
LOGOUT	C → S	Odhlášení ze serveru.
LOGINOK	C ← S	Potvrzení úspěšného přihlášení.
LOGINFAIL	C ← S	Neplatné přihlášení.
MESSAGE	C ← S	Textová zpráva zasílaná klientovi.

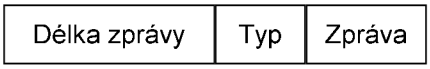
Tabulka 5.1: Zprávy protokolu

Následující obrázky popisují strukturu zpráv.

Obrázek 5.2: Obecná hlavička zprávy

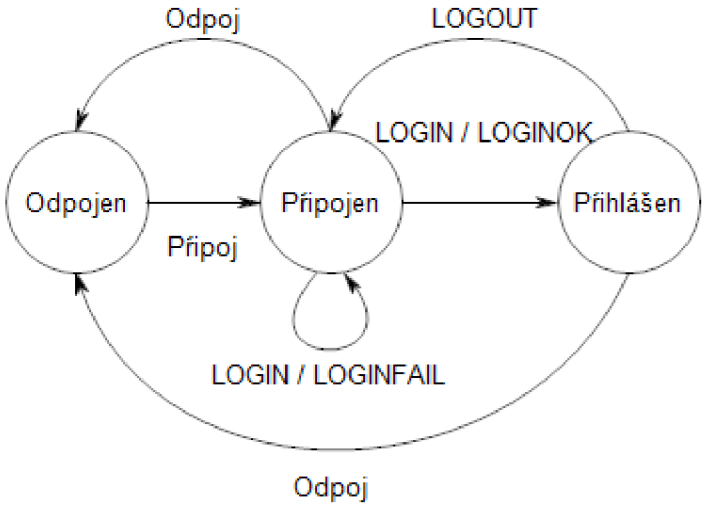
Obrázek 5.3: Struktura zprávy LOGIN

Obrázek 5.4: Struktura zprávy GPSDATA



Obrázek 5.5: Struktura zprávy MESSAGE

Existují tři stavy, ve kterých se může komunikace nacházet. Následující stavový diagram popisuje přechody mezi nimi. Posílání dat je možné pouze ve stavu přihlášen.



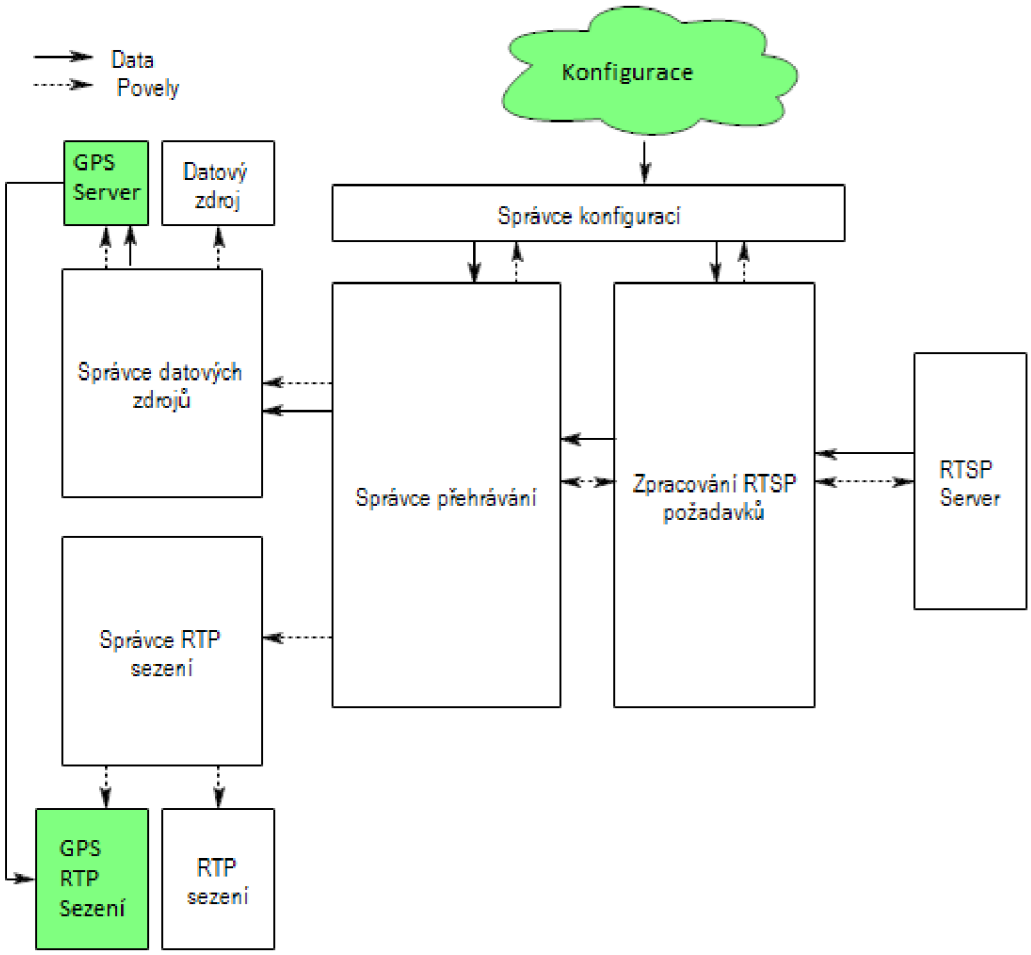
Obrázek 5.6: Stavový diagram komunikace mezi PDA a serverem.

5.1.2 GPS

GPS modul je připojen pomocí sériového rozhraní, na kterém jsou poskytována data ve formě textového protokolu NMEA [23]. Ten poskytuje kromě GPS pozice i další informace, například aktuální čas, počet dostupných satelitů atd. Spolu s GPS souřadnicemi se ve zprávě serveru přenáší i kvalita signálu. Ta je pouze dvoustavová a indikuje platnost přenášené pozice.
GPS souřadnice jsou serveru odesílány v intervalech 0,5 s. Pro pozorovatele je to dostatečně krátký čas a nedochází tak ke zbytečnému zatěžování sítě častějšími aktualizacemi. Pro aplikace s vyššími nároky na přesnou aktuální pozici nebo plynulejší pohyb je možné interval snížit.

5.2 RTP Server

RTP server je nejsložitější ze všech částí aplikace. Většina jeho částí je navržena tak, aby byl univerzální a mohl být použit pro přenos libovolných živých procesních dat. Na následujícím obrázku je architektura serveru. Barevně jsou odlišeny části, které jsou specializované pro přenos GPS dat.



Obrázek 5.7: Architektura serveru

Specializováno je především RTP sezení, zajišťující balení dat do paketu a jejich odesílání podle profilu, který je definován níže. Dále je jako zdroj dat použit GPS server, který je napojen na univerzální rozhraní RTP serveru.

Součástí RTP serveru je i RTSP server, který umožňuje klientům inicializaci získávání dat z RTP serveru. Uchovává jejich stav a poskytuje přístup k dalším datům, která nejsou odesílána pomocí RTP. Navíc jeho metody GET_PARAMETER a SET_PARAMETER slouží jako zpětný kanál, jímž může klient přenášet například povely přes spolehlivý přenos.

U všech serverů, které v rámci aplikace existují, se jedná o neblokující servery, které požadavky od příchozích spojení zpracovávají v samostatných vláknech. K synchronizaci a zajištění výlučného přístupu ke společným zdrojům je použito zámků.

5.2.1 Přenášená data

GPS souřadnice jsou u zařízení aktualizovány pravidelně v krátkých časových intervalech. Data jsou tak velmi tolerantní vůči ztrátám paketů. Vysoké ztráty se projeví pouze trhanějším pohybem na mapě, což je akceptovatelné. Z tohoto důvodu není zavedena žádná redundance dat, ani žádné opravné kódy a je zakázáno odesílání RTCP zpráv z klienta na server.

Problematická jsou však zařízení, která neaktualizují svou pozici, jsou odpojena, nebo přejdou do stavu odpojeno. U klienta, který nezaznamená změnu, by nedošlo k aktualizaci stavu a zobrazoval by tak zastaralou informaci. Proto je nutné pravidelně odesílat i takové hodnoty, které se nemění. Interval odesílání je nastaven na 60 s. Je tedy zaručeno, že každá hodnota bude přenesena v případě dostatečného dostupného datového toku minimálně jednou za 60 s.

5.2.2 Odesílání dat

Odesílání dat je komplikované především možností zavedení limitů na přenos. Takže musí být navržen mechanismus pro rozložení případné vyšší zátěže, který tento limit zohlední. Další komplikací je nutnost opakovaného odesílání hodnot.

5.2.2.1 Výpočet maximální velikosti paketu

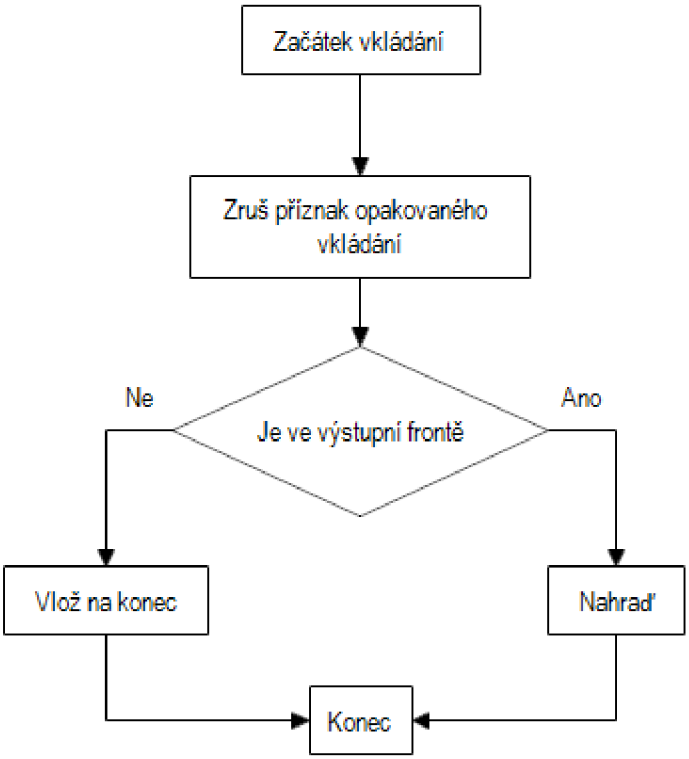
U každého zdroje je možné nastavit maximální šířku pásma, která pro něj bude k dispozici. Od tohoto údaje se odvíjí množství dat, které může být v daný okamžik odesláno. Pro výpočet maximální možné velikosti odesílaného paketu s_{max} platí následující vztahy:

$$s_{max} = B - \sum_{T=-k}^{0} s_T \qquad k = \frac{1000}{n}$$

Kde B je přenosová rychlost v B/s, s_T je velikost paketu odeslaného v čase T a n je interval odesílání paketů v ms. Pokud není nastaven žádný limit, jsou data odesílána do maximální velikosti paketu.

5.2.2.2 Výstupní fronta dat

Ve výstupní frontě typu FIFO jsou ukládána data před odesláním. Od běžné fronty typu FIFO se liší tím, že je možné v ní nahrazovat hodnoty, které zastaraly předtím, než došlo k jejich odeslání. To se může stát v případě, že během jednoho intervalu odesílání paketů dojde ke dvěma a více změnám jedné hodnoty, nebo pokud dochází ke zpoždění odesílání vlivem nedostatečné šířky pásma. Postup vkládání hodnoty do fronty je znázorněn na následujícím diagramu.



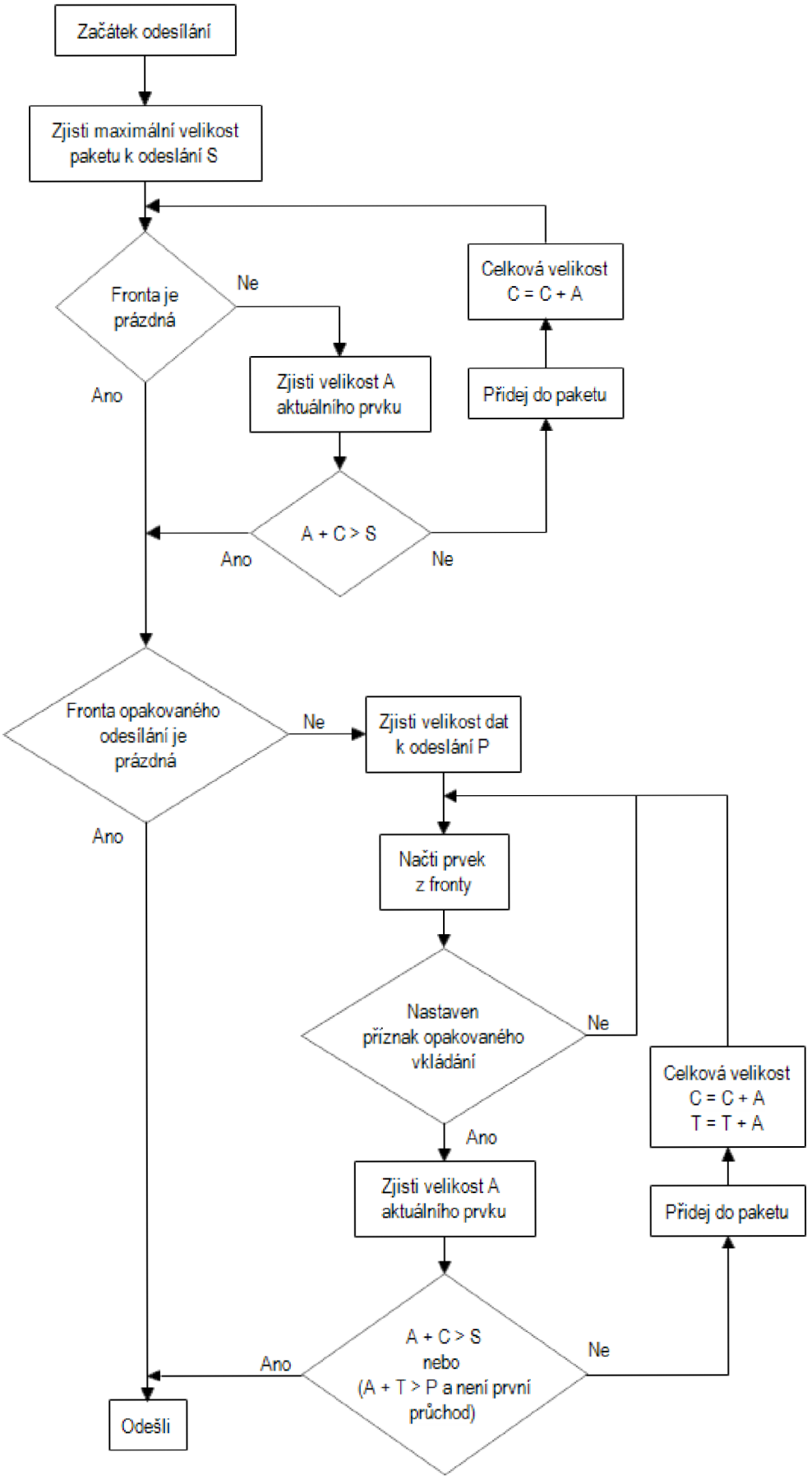
Obrázek 5.9: Diagram vkládání hodnoty do výstupní fronty

5.2.2.3 Interval opakovaného odesílání

Jak již bylo zmíněno, jednou za 60 s musí dojít k odeslání každé hodnoty. Aby nedocházelo ke skokovému zatížení sítě během odeslání těchto hodnot, je jejich odeslání rozprostřeno do celého následujícího intervalu. Fronta těchto dat má nižší prioritu, než hodnoty změn, tudíž se nemusí podařit odeslat všechna data z fronty v průběhu jednoho intervalu. V takovém případě dojde k novému naplnění fronty až v okamžiku úplného vyprázdnění fronty v průběhu některého z následujících intervalů. Každá hodnota má v sobě nastaven příznak, který udává, že je vložena do fronty opakovaného odesílání. Pokud je v průběhu čekání na odeslání tento příznak zrušen změnou této hodnoty, je z fronty odstraněna. Výpočet průměrné velikosti, která má být odeslána z fronty opakovaného odesílání v rámci jednoho odesílání je dán:

$$V_{avg} = \frac{\sum_{i=0}^{n-1} V_i}{n}$$

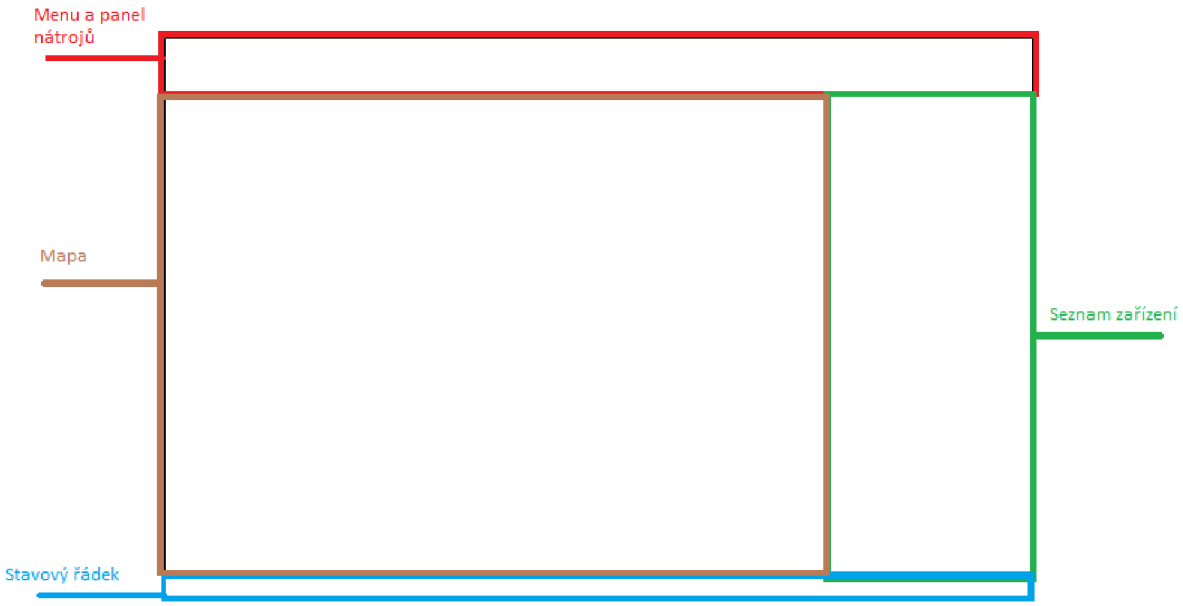
Kde n je počet hodnot ve frontě a V je velikost hodnoty. Postup odesílání paketu je popsán následujícím diagramem.



Obrázek 5.10: Diagram odesílání paketu

5.3 Klient

Klientská aplikace zobrazuje na mapě zařízení podle GPS souřadnic, které jí poskytuje server. Návrh vzhledu aplikace je na následujícím obrázku.



Obrázek 8.11 Navrhovaný vzhled aplikace

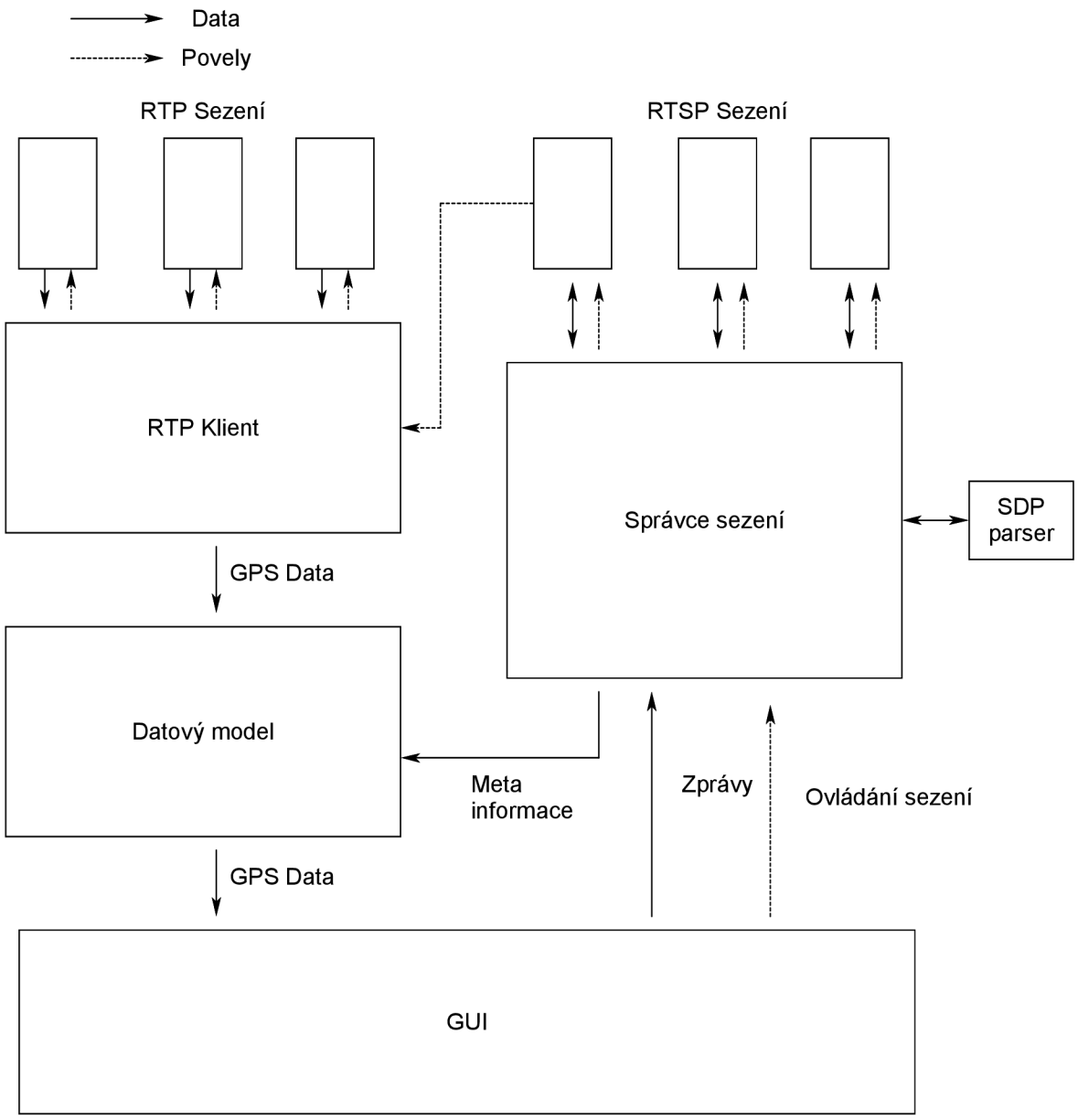
Kromě pozice na mapě bude umět aplikace zobrazit aktuální stav zařízení, který bude signalizován pomocí stavových ikon v seznamu zařízení. Jejich seznam a význam je v následující tabulce.

Stav	Popis
	Zařízení není připojeno k serveru.
	Zařízení je připojeno a zasílá platnou polohu.
	Zařízení je připojeno, ale ještě neposlalo žádnou platnou polohu.
	Zařízení je připojeno, ale posílá neplatnou polohu. Například z důvodu špatného GPS signálu.
	Stav zařízení je neznámý

Tabulka 5.1: Stavy klienta

5.3.1 Architektura

Architektura aplikace je postavena na třech vrstvách. První je grafické rozhraní, pod ním leží řídící a datová vrstva a poslední je vrstva síťová, která zprostředkovává komunikaci se serverem. Celkový pohled na architekturu poskytuje následující obrázek. Pro jednoduchost v něm není uvedena správa dostupných datových zdrojů, která uchovává informace o dostupných serverech a jimi poskytovaných datových proudů.

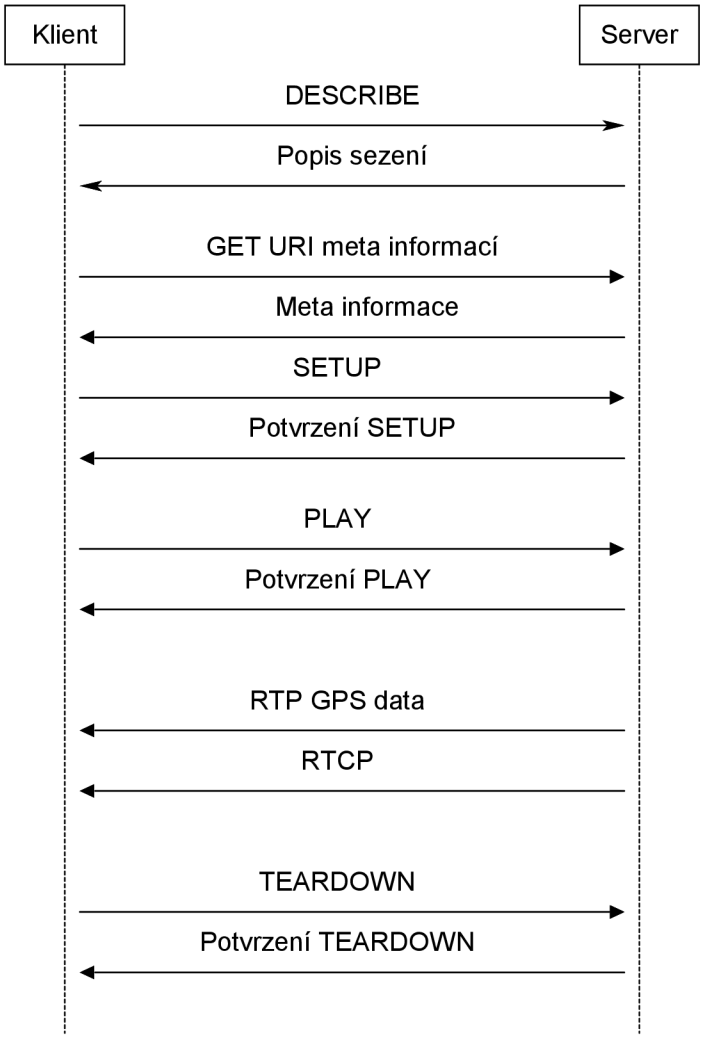


Obrázek 5.12: Architektura aplikace

Pro každý datový zdroj je vytvořeno samostatné RTSP sezení, které řídí přenos dat z tohoto zdroje. Pokud dojde k úspěšnému navázání komunikace a nastavení parametrů přenosu, je vytvořeno RTP sezení, které je ovládáno přes RTSP sezení.

5.3.2 Popis komunikace klient – server

Chování síťové vrstvy na klientovi odpovídá chování popisovanému v předchozí podkapitole o návrhu RTP serveru. Typickou výměnu zpráv při ustanovení spojení, nastavení parametrů RTP přenosu a ukončení spojení popisuje následující obrázek. Pořadí zpráv je stejné jako u obrázku z podkapitoly, která popisovala protokol RTSP. Pouze přibyla zpráva pro získání meta informací a RTP přenos je přizpůsoben navrhované aplikaci.



Obrázek 5.13: Popis komunikace mezi klientem a RTSP serverem

5.3.3 Meta informace

Klient ve své konfiguraci neobsahuje žádná data o zařízeních, která jsou připojeny k serveru a posílají data. Proto je nutné ještě před zahájením příjmu GPS dat, tyto informace získat. Jak bylo uvedeno v kapitole zabývající se návrhem přenosu dat, je k tomuto účelu použita metoda GET protokolu RTSP. Při vyjednávání spojení je serverem, jako reakce na požadavek DESCRIBE, zaslána zpráva obsahující záznam protokolu SDP, který popisuje sezení. V parametru u= je uvedena URI souboru s dostupnými meta informacemi. Meta informace jsou uloženy ve formě XML souboru s následující strukturou.

```
<data>
 <item name=" " icon="" desc="" id="" messageid="" />
</data>
```

Položky desc, name a icon slouží pouze k zobrazení v uživatelském rozhraní. Položka id je identifikátor hodnoty, která obsahuje GPS data příslušející tomuto záznamu. Položka messageid zase identifikátor hodnoty, do které je možné ukládat zprávu pro zařízení.

6 Realizace demonstrační aplikace

Aplikace byla implementována dle návrhu z předchozí kapitoly. Pro vývoj byl zvolen jazyk C++, který poskytuje velmi dobrý výkon a je pro něj dostupné velké množství knihoven. Volba jazyka C++ souvisí také s volbou knihovny pro RTP, která je popsána v následující části. V celém projektu je použit multiplatformní framework Qt, který poskytuje abstrakci nad síťovou komunikací, podporu pro vytváření uživatelského rozhraní, vláken a mnoho dalších usnadnění.
Pro ukládání konfigurací je použita technologie XML.

6.1 RTP a RTSP

Pro RTP byla zvolena objektově orientovaná knihovna JRTPLIB, která je dobře zdokumentovaná a snadno použitelná. Oproti jiným knihovnám, u kterých nejsou k dispozici zdrojové kódy, existuje možnost v případě nutnosti upravit její chování. Do knihovny byla přidána možnost posunout RTP časovač, aniž by byl odeslán paket.

Jelikož nebyla nalezena žádná vhodná knihovna pro vytvoření RTSP serveru, byl tento server implementován. Výsledný RTSP server poskytuje všechny potřebné metody, které byly v této práci identifikovány jako nutné pro přenos živých procesních dat, a jeho chování odpovídá specifikaci dané RFC 2326 [14]. Tabulka níže obsahuje seznam metod, které jsou implementovány. Požadavky na ostatní metody vracejí kód 501 NOT IMPLEMENTED. Jediný rozdíl je v metodě PAUSE, která je implementována, ale vrací kód 405 METHOD NOT ALLOWED pro všechny požadavky. V případě, že by typ aplikace umožňoval použití metody PAUSE, je její zprovoznění otázkou jednoduché úpravy.

Jméno	Směr
DESCRIBE	$C \rightarrow S$
OPTIONS	$C \rightarrow S$
PLAY	$C \rightarrow S$
SETUP	$C \rightarrow S$
TEARDOWN	$C \rightarrow S$
GET_PARAMETER	$C \rightarrow S$
SET_PARAMETER	$C \rightarrow S$

Tabulka 6.1: Přehled implementovaných metod RTSP serveru

Metodu OPTIONS aplikace nevyužívá, ale byla implementována, protože ji vyžaduje specifikace.

6.2 Klientská aplikace

Základem klientské aplikace je webový prohlížeč Webkit, u kterého poskytuje Qt jednoduchou možnost vložení do aplikace. Webkit obsahuje rozhraní, pomocí kterého je možné z programu volat javascriptové funkce v kontextu rámce webové stránky. Díky tomu je možné z uživatelského rozhraní ovládat mapu a měnit pozici značek.

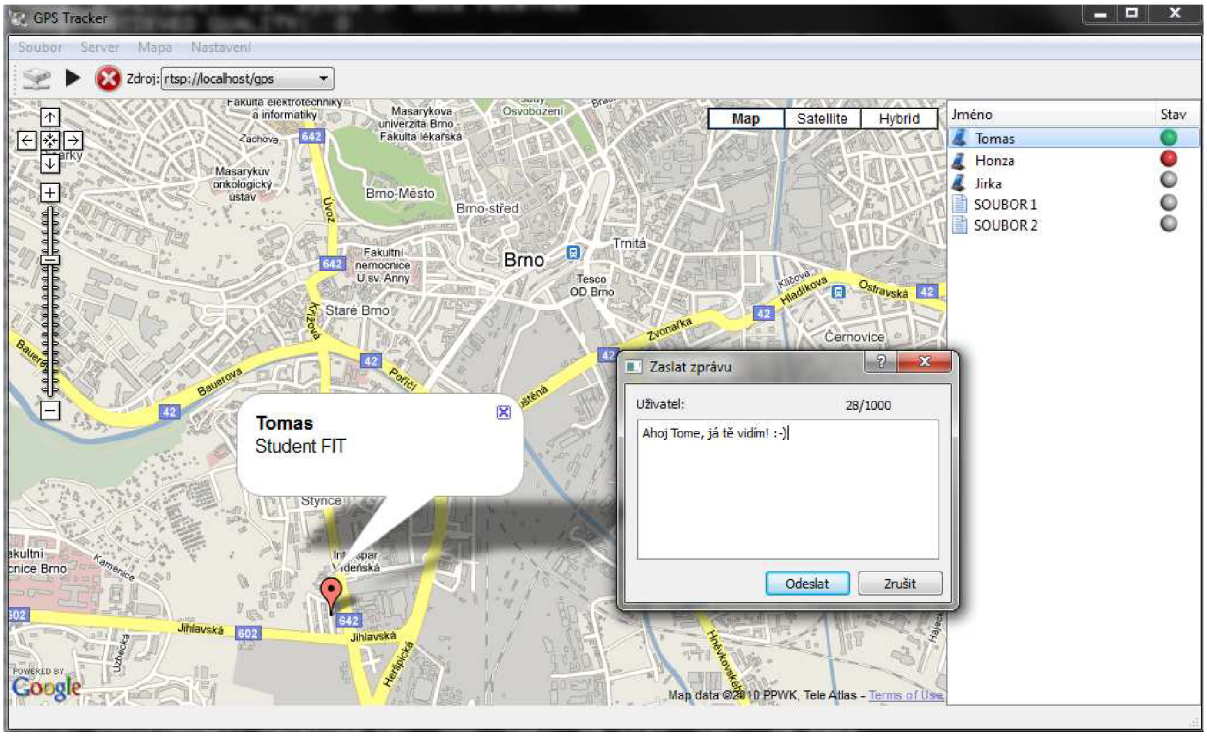
Mapa je realizována pomocí webové stránky, která je zobrazena ve vloženém webovém prohlížeči. Jako mapový podklad jsou použity mapy od společnosti Google a jejich Google Maps

API, což je javascriptové rozhraní pro možnost manipulace s mapou. Pro účely této aplikace jsou nejdůležitějšími funkcemi pozicování, vkládání značek a jejich následná správa.

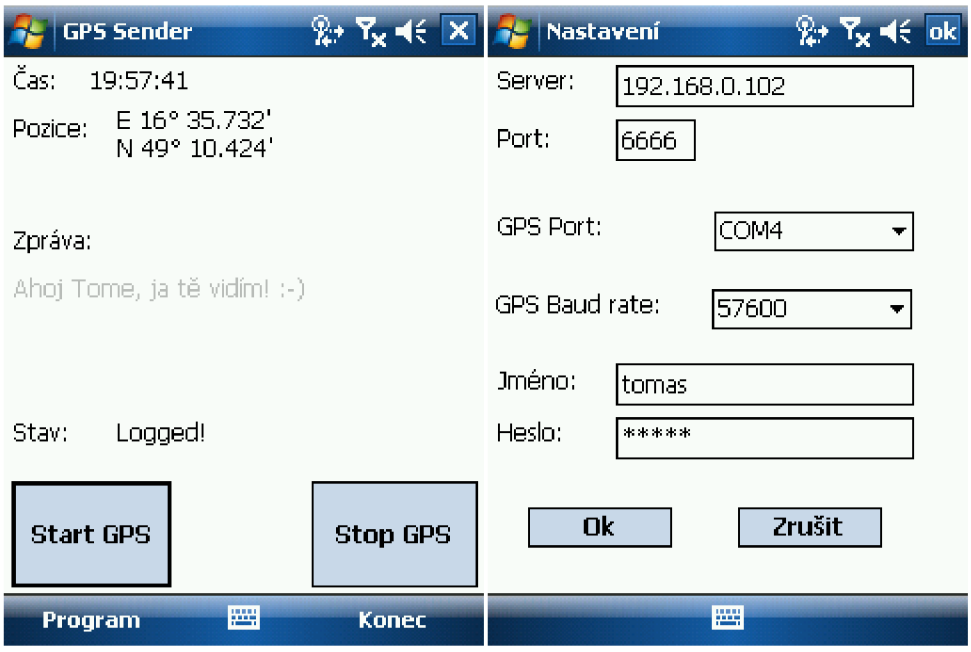
6.3 GPS Klient

GPS klient pro PDA byl implementován v programovacím jazyce C# v prostředí .NET. Hlavním důvodem pro tuto volbu byla vysoká abstrakce práce se sériovým portem, která byla pro účely aplikace dostačující. Dalším plusem pro tuto technologii je možnost velmi rychlého vývoje grafického rozhraní.

6.4 Ukázky z vytvořené aplikace



Obrázek 6.1: Ukázka vzhledu klientské aplikace



Obrázek 6.2: Ukázka vzhledu PDA aplikace

7 Zhodnocení výsledků

Jak ukázala tato práce, problémy, které se vyskytují při distribuci procesních dat mnoha klientům, jsou s využitím protokolu RTP řešitelné. Zejména využití multicastu může výrazně zredukovat velikost datového toku ze serveru. Také použití mixerů k sjednocení více proudů dat a translátorů k modifikaci dat mohou najít své uplatnění.

Jako ideální cílový typ se jeví aplikace s velkým objemem často se měnících dat, která slouží především k pasivnímu monitorování velkým množstvím klientů. V takovém případě se nejméně projeví problémy s výpadky paketů a zasílání povelů. Pro takovou aplikaci poskytuje protokol RTP dobře definovaný standard.

Existuje však i skupina aplikací, pro které je takový přenos nevhodný. Především při přenosu dat, která se často nemění, se díky nutnosti znovu zasílání výsledný datový tok skládá převážně z dat, která jsou přenášena zbytečně. Také u aplikací s velkým množstvím povelů ve směru klient -server se na velikosti datového toku negativně projeví velké množství režijních dat protokolu RTSP.

Přínosem protokolu RTP pro distribuci procesních dat je číslování paketů, identifikace typu přenášeného typu obsahu a identifikaci serveru. Protokol RTCP dále přináší především možnost získávat informace o kvalitě poskytované služby. Další výhodou je možnost škálování poskytované služby pomocí vrstveného modelu, kdy si klient vybere pro sebe vhodnou skladbu kanálů. To umožňuje obsluhu klientů s velmi rozdílným charakterem připojení.

Bohužel jedna z nejdůležitějších schopností RTP při přenosu multimediálních dat zůstává u přenosu živých dat nevyužita. Tou schopností je časová synchronizace více proudů, například zvuku a videa. U distribuce živých dat tato schopnost postrádá smysl, protože přenesená data jsou považována za aktuální obraz stavu dat.

Zajímavou možností je také spojení distribuce živých procesních dat s multimediálním proudem. Například přenos obrazu z videokamery a dat týkajících se prostředí, ve kterém je kamera umístěna. V takovém případě se dostává ke slovu i časová synchronizace mezi procesními a multimediálními daty a dochází k plnému využití protokolu RTP.

8 Závěr

V této práci byla prozkoumána možnost distribuce procesních dat pomocí protokolu RTP. Sumarizace znalostí z první části práce umožnila vytvořit návrh distribuce živých dat s využitím trojice protokolů RTP, RTCP a RTSP, který je hlavním tématem druhé poloviny práce. Vytvořený návrh je největším přínosem této práce a je použitelný především v aplikacích zaměřených na monitorování aktuálního stavu řízených procesů.

Na základě navrženého rozšíření protokolů byla vytvořena demonstrační aplikace, která umožňuje distribuci GPS souřadnic z mobilních jednotek, v tomto případě mobilních zařízení se systémem Windows Mobile, ke klientům a jejich následné zobrazení na mapě.

Pro širší možnost uplatnění by bylo potřeba prozkoumat problematiku přenosu historických procesních dat pomocí RTP. Další možné oblasti rozvoje tématu jsou například zkoumání vlivu použití různých typů dopředné korekce dat při distribuci dat v reálných sítích na kvalitu poskytované služby, nebo již zmiňované spojení datových a multimediálních proudů dat.

Literatura

[1] Ronešová, A.: *Přehled protokolu MODBUS., k*věten *2005.* Dokument dostupný na URL: http://home.zcu.cz/~ronesova/bastl/files/modbus.pdf (prosinec 2009)

[2] Open DeviceNet Vendor Association: *A Guide for EtherNet/IP™ Developer,* 2008

[3] Newman, M., H.: *BACnet - The New Standard Protocol*. září 1997. Dokument dostupný na URL: http://www.bacnet.org/Bibliography/EC-9-97/EC-9-97.html (květen 2010)

[4] Wikipedia: *Simple Network Management Protocol*, Dokument dostupný na URL: http://en.wikipedia.org/wiki/Simple_Network_Management_Protocol (květen 2010)

[5] Microsoft Corporation. *Windows Media HTTP Streaming Protocol Specification*. 2009. 153 s.

[6] Adobe Systems Incorporated. *Real-Time Messaging Protocol (RTMP) specification*. 2009. 70 s. Dokument dostupný na URL: http://www.adobe.com/devnet/rtmp (prosinec 2009)

[7] Perkins, C. *RTP: Audio and Video for the Internet*. 2003.: Addison Wesley, 2003. 432 s. ISBN 0-672-32249-8.

[8] Schulzrinne, H., Casner, S., Frederick, R., Jacobson, V.: *RTP: A Transport Protocol for Real-Time Applications. 2003.* Dokument dostupný na URL: http://www.ietf.org/rfc/rfc3550.txt (prosinec 2009)

[9] Clark, D. D., Tennenhouse, D. L. Architectural considerations for a new generation of protocols. In *ACM SIGCOMM*. ACM, 1990. s. 200-208. Dokument dostupný na URL: http://portal.acm.org/citation.cfm?id=99553. ISSN 0146-4833 (prosinec 2009)

[10] Sun Microsystems. *Java Media Framework API Guide*. 1999. 245 s. Dokument dostupný na URL: http://www.scribd.com/doc/6932630/Java-Media-Framework-API-Guide. (prosinec 2009)

[11] Schulzrinne, H., Casner, S.: *RTP Profile for Audio and Video Conferences with Minimal Control.* Dokument dostupný na URL: http://www.ietf.org/rfc/rfc3551.txt (prosinec 2009)

[12] Hellstrom, G.: *RTP Payload for Text Conversation.* 2000. Dokument dostupný na URL: http://www.ietf.org/rfc/rfc2793.txt (prosinec 2009)

[13] Civanlar, M., Cash, G.: *RTP Payload Format for Real-Time Pointers,*. 2000. Dokument dostupný na URL: http://www.faqs.org/rfcs/rfc2862.html (květen 2010)

[14] Network Working Group: *Real Time Streaming Protocol (RTSP).* 1998. Dokument dostupný na URL: http://www.ietf.org/rfc/rfc2326.txt (prosinec 2009)

[15] Handley, M., Jacobson, V., Perkins, C.: *SDP: Session Description Protocol.,* 2006. Dokument dostupný na URL: http://www.rfc-editor.org/rfc/rfc4566.txt (květen 2010)

[16] Wikipedia: *Extensible Markup Language,* Dokument dostupný na URL: http://cs.wikipedia.org/wiki/Extensible_Markup_Language (květen 2010)

[17] Opc Task Force: *OPC Overview Version 1.0,* 27. 9. 1998

[18] Schoolar, E., Gemmel, J.: *Using Multicast FEC to Solve the Midnight Madness Problem,* Microsoft Research, 30. 9. 1997. Dokument dostupný na URL: http://research.microsoft.com/apps/pubs/default.aspx?id=69542 (květen 2010)

[19] Speakman, T., et al.: *PGM Reliable Transport Protocol Specification.*, prosinec 2001. Dokument dostupný na URL: http://www.rfc-editor.org/rfc/rfc3208.txt (květen 2010)

[20] Rosenberg, J., Schulzrinne, H.: *An RTP Payload Format for Generic Forward Error Correction,* prosinec 1999. Dokument dostupný na URL: http://www.faqs.org/rfcs/rfc2733.html (květen2010)

[20] Rosenberg, J., Schulzrinne, H.: *An RTP Payload Format for Generic Forward Error Correction,* prosinec 1999. Dokument dostupný na URL: http://www.faqs.org/rfcs/rfc2733.html (květen2010)

[21] Tan, W., Zakhor, A.: *Video Multicast using Layered FEC and Scalable Compression*

[22] Zezulka, F., et al.: *Průmyslové komunikační sítě,* Skripta, ÚAMT FEI VUT, Brno 2000.

[23] SiRF Technology: *NMEA Reference Manual*, *Revision 1.3,* leden 2005 Dokument dostupný na URL: http://www.sparkfun.com/datasheets/GPS/NMEA%20Reference%20Manual1.pdf (květen2010)

Seznam příloh

Příloha č. 1: Obsah přiloženého CD
Příloha č. 2: CD se zdrojovými kódy a demonstrační aplikací

Příloha č. 1: Obsah přiloženého CD

Application – Spustitelné soubory aplikace pro systém Windows a všechny knihovny potřebné pro její spuštění.
PDA source – Zdrojové texty aplikace pro PDA spolu s projektem pro vývojové prostředí Visual Studio 2008.
PDA Appliacation – Spustitelné soubory aplikace pro PDA.
Sources - Zdrojové texty aplikace. Obsahují také projekty pro vývojové prostředí Eclipse s nainstalovaným pluginem pro Qt.

- **GPSServer** – Server pro mobilní klienty. Funguje jako modul pro server.
- **GPSTracker** – Klientská aplikace zobrazující pozice na mapě.
- **jRTPLib** – Knihovna pro podporu RTP.
- **jThread** – Knihovna pro práci s vlákny používaná v jRTPLib.
- **RTPBase** – Knihovna sdílených objektů mezi serverem a klientem.
- **RTSPLib** – Knihovna implementující RTSP server.
- **Server** – Server pro odesílání RTP proudů dat.

help.pdf – Nápověda ke klientské aplikaci.
Readme.txt – Základní popis programu. Obsahuje mimo jiné i seznam potřebných knihoven.