VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV TELEKOMUNIKACÍ

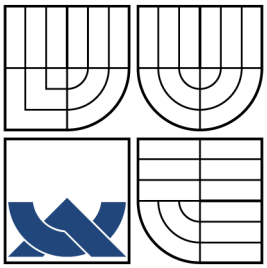
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF TELECOMMUNICATIONS

TUNELOVÁNÍ MULTICASTOVÝCH DAT

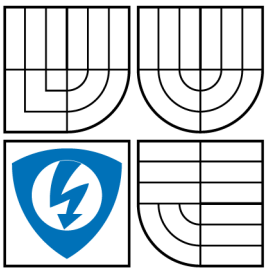
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE STANISLAV STODŮLKA
AUTHOR

BRNO 2009



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY
A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND
COMMUNICATION
DEPARTMENT OF TELECOMMUNICATIONS

TUNELOVÁNÍ MULTICASTOVÝCH DAT
MULTICAST DATA TUNNELING

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — STANISLAV STODŮLKA
AUTHOR

VEDOUCÍ PRÁCE — ING. MILAN ŠIMEK
SUPERVISOR

BRNO 2009

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav telekomunikací

Bakalářská práce

bakalářský studijní obor
Teleinformatika

Student:	Stanislav Stodůlka	***ID:***	77835
Ročník:	3	***Akademický rok:***	2008/2009

NÁZEV TÉMATU:

Tunelování multicastových dat

POKYNY PRO VYPRACOVÁNÍ:

Prostudujte principy a možnosti tunelování multicastových dat přes unicastovou síť. Seznamte se s nástrojem mrouted, sloužícím pro realizaci multicastového tunelu. Navrhněte aplikaci pro přenos multicastových dat typu klient/server. Aplikaci otestujte na experimentální síti v laboratoři 249 na ústavu Telekomunikací. Po úspěšném otestování aplikace, implementujte na koncových stanicích aplikaci mrouted, pomocí které budete schopen data nadále přenášet i bez multicastové podpory na místních směrovačích. Postup tunelování důkladě popište. Pro aplikaci navrhněte grafické uživatelské prostředí, tak aby bylo možné ověřit funkci multicastového tunelu.

DOPORUČENÁ LITERATURA:

[1] Zentralen Informatik Dienst, "mrouted Daemon", [online], 2001, dostupné na <http://www.unet.univie.ac.at/aix/cmds/aixcmds3/mrouted.htm>
[2] E. NEMETH, G. SNYDER, T, HEIN, Linux-kompletní příručka administrátora, ComputerPress, a.s.,2004, ISBN: 80-722-6919-4

Termín zadání:	9.2.2009	***Termín odevzdání:***	2.6.2009

Vedoucí práce: Ing. Milan Šimek

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

UPOZORNĚNÍ:

ABSTRAKT

Tento dokument se zabývá vyvsvětlením pojmů ve směrování multicastových dat a jejich tunelování sítí. Dále také vývojem aplikace v jazyce Java pro otestování unicastového a multicastového posílání dat sítí. Aplikace jsou na bázi klient/server. Dokument popisuje Linuxový daemon mrouted a alternativní daemony s podobnou funkcí směrování multicastových dat.

KLÍČOVÁ SLOVA

Linux, multicast, tunelování, java, mrouted

ABSTRACT

This document contains introducion of terms in routing multicast data and their tunneling in network. As well developing aplication in Java to test sending unicast and multicast data through network. Aplication are based on client/server. Document describe Linux daemon mrouted and alternativ daemons with similar function of routing multicast data.

KEYWORDS

Linux, multicast, tunneling, java, mrouted

STODŮLKA, S. *Tunelování multicastových dat.* Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2009. 25 s. Bakalářská práce. Vedoucí bakalářské práce Ing. Milan Šimek.

PROHLÁŠENÍ

Prohlašuji, že svou bakalářskou práci na téma „TUNELOVÁNÍ MULTICASTOVÝCH DAT“ jsem vypracoval samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené bakalářské práce dále prohlašuji, že v souvislosti s vytvořením této bakalářské práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení § 152 trestního zákona č. 140/1961 Sb.

V Brně dne ....

(podpis autora)

Rád bych touto cestou poděkoval svému vedoucímu bakalářské práce, Ing. Milanu Šimkovi, za velmi užitečnou metodickou pomoc a cenné rady při zpracování bakalářské práce. Dále bych chtěl ještě poděkovat panu Ing. Petru Juráskovi z firmy soLNet s.r.o. za rady ohledně OS Linux.

OBSAH

SEZNAM OBRÁZKŮ

1 ÚVOD

V první části své bakalářské práce se budu zabývat teoretickým výkladem týkajícím se multicastového přenosu dat sítí a k čemu je multicast vhodný. Dále vysvětlím na jakém principu multicast funguje a jakých metod ke svému chodu využíva. Budu se zabývat výkladem o připojování a odpojování koncových stanic od zdrojů multicastového vysílání a jak se tvoří cesty od koncové stanice ke zdroji. V poslední části teoretického výkladu popíši princip tunelování a zapouzdřování tunelovaných dat.

V druhé části práce se budu zabývat vývojem a popisem aplikací typu klient/server pro přenos multicastových dat. Dále se budu zabývat instalací a konfigurací Linuxových koncových stanic a uzlů. V části věnované Linuxu ještě popíši instalaci potřebných balíku pro realizaci projektu. Poté se budu věnovat popisu daemona mrouted a jeho nevýhodám. Dále pak alternativním směrovacím multicastovým daemonům, které mohou nahradit mrouted, a také nahradě tunelování.

V poslední části je samotná konfigurace, testování a dosažené výsledky.

2 TEORETICKÉ VYPRACOVÁNÍ

2.1 Multicast

Multicast je rozšíření IP protokolu o doručování datagramů skupině koncových stanic a to s využitím každého uzlu v cestě pouze jednou a s množením doručovaného datagramu na místě, kde je to nezbytně nutné. Historie IP multicastu se datuje od roku 1989, kdy jej Steve Deering popsal ve své disertační práci. IP multicast se dnes nejčastěji spojuje s distrubicí streamovaného videa a audia na internetu a je používán i k distribuci aktualizací rozšířených operačních systémů. [2, 3]

2.1.1 princip IP multicastu

Princip IP multicastu je založen na komunikaci one-to-many na IP infrastruktuře. Je zde (server), který založí na vhodné multicastové IP adrese skupinu, do které zasílá data. K této skupině se pomocí IGMP (Internet Group Management Protocol) protokolu připojují jednotlivé koncové stanice, které chtějí tato data přijímat. Zdroji nezáleží na tom, kolik je připojených účastníků a kde v síti jsou, protože jeho jediným účelem je založit skupinu a posílat data. O vybírání cesty od zdroje dat ke koncové stanici se stará sama síť, tedy spíše uzly a na nich vhodné multicastové směrovací protokoly. Uzly v síti se starají o směrování a replikaci datagramů. Nejčastěji používaným protokolem transportní vrstvy modelu TCP/IP pro přenos multicastových dat je UDP (User Datagram Protocol), do kterého jsou zapouzdřena multicastová data, adresa zdroje, adresa příjemce, port zdroje a port příjemce (viz RFC 768). Důvodem, pro tak hojné používání protokolu UDP v multicastové komunikaci je to, že se při něm neprovádí navázání spojení tzv. handshake a hlavně se nekontroluje přijetí dat koncovou stanicí. To může způsobit, že datagramy přijdou ve špatném pořadí nebo nedojdou vůbec. Jelikož je ale drtivá většina využívání IP multicastu zaměřena na přenos streamovaného videa a audia, kde se předpokládá množství připojených koncových stanic v multicastové skupině na stovky tisíc, je s přihlédnutím k povaze a objemu doručených dat nesmyslné kontrolu přijetí provádět. Pokud se IP multicastové vysílání používa k distribuci dat, při které potřebujeme kontrolu o přijetí, dá se využít protokol PGM (Pragmatic General Multicast). [1, 4]

2.1.2 IP multicastový adresní prostor

IP multicastový adresní prostor je z třídy adres D. Tedy adresy, jejichž oktet binárně začíná na 1110 (rozsah 224.0.0.0 - 239.255.255.255). Tyto adresy se dále dělí na několik skupin, které jsou přesně specifikovány v RFC 3171. [1]

- 224.0.0.0 - 224.0.0.255 (224.0.0/24) Local Network Control Block
- 224.0.1.0 - 224.0.1.255 (224.0.1/24) Internetwork Control Block
- 224.0.2.0 - 224.0.255.0 AD-HOC Block
- 224.1.0.0 - 224.1.255.255 (224.1/16) ST Multicast Groups
- 224.2.0.0 - 224.2.255.255 (224.2/16) SDP/SAP Block
- 224.252.0.0 - 224.255.255.255 DIS Transient Block
- 225.0.0.0 - 231.255.255.255 RESERVED
- 232.0.0.0 - 232.255.255.255 (232/8) Source Specific Multicast Block
- 233.0.0.0 - 233.255.255.255 (233/8) GLOP Block
- 234.0.0.0 - 238.255.255.255 RESERVED
- 239.0.0.0 - 239.255.255.255 (239/8) Administratively Scoped Block

2.2 Směrování v IP multicastových sítích

Směrování obecně znamená hledání cest v IP sítích. Úkolem směrování je dopravit datový paket z místa odesilatele na místo příjemce a to co nejrychleji a nejefektivněji. Jelikož síťová infrastruktura mezi odesilatelem a příjemcem může být velmi složitá, nezabívá se směrování celou cestou, ale jen jedním krokem a to komu dalšímu předat přijatý paket. [3]

2.2.1 Princip IP multicastového směrování

Pro názornost principu směrování v multicastových sítích, bude provedeno srovnání se směrováním v unicastových sítích.

U unicastového směrování přijde na jeden port směrovače paket, který má v hlavičce adresu jediného cílového příjemce. Směrovač se podívá do své směrovací tabulky a na základě toho se rozhodne, co s paketem udělá v dalším kroku. Pokud cílovou adresu ve své tabulce nemá, tak paket zahodí (vymaže ho z paměti). Pokud cílovou adresu směrovač ve své tabulce má, tak pošle paket na příslušný port, kde se cíl nachází. V případě multicastového směrování je příjemců několik a směrovače proto musí řešit, kdy mají daný paket replikovat, aby zatížení přenosového pásma nebylo vyšší, než musí být.

To znamená dvě podmínky:

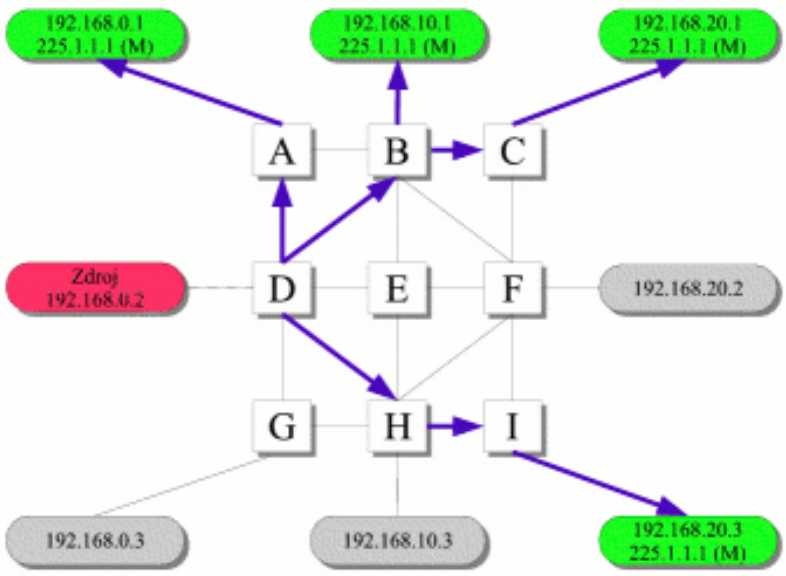
1. Aby se pakety replikovaly na místě, kde to bude nejefektivnější.
2. Aby replikovaných paketů nebylo více než je potřeba. Tedy aby jeden cíl nedostal více kopií stejného paketu.

2.2.2 Replikace multicastových paketů

K určení místa, kde se má paket replikovat, slouží tak zvané „multicastové distribuční stromy“. Kde je kořen vnímán jako zdroj multicastových dat, listy jako účastníci multicastové skupiny a větve jako cesty vedoucí od stromu k listům (dělící se v místech směrovačů). Existují dva základní typy těchto stromů: zdrojový strom (source tree) a sdílený strom (shared tree). [5]

Zdrojový strom (Obr. 2.1)

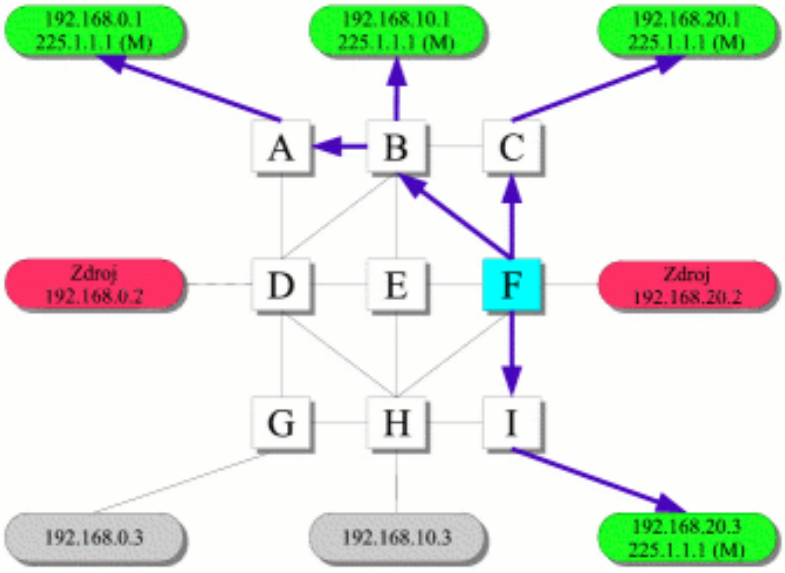
Kořeny zdrojových dat můžou být různé. Používá se pro něj označení (S, G) kde S (Source) je adresa zdroje multicastového vysílání a G (Group) je adresa skupiny, na kterou je vysíláno. Takže pro různé zdroje posílající data do stejné multicastové skupiny se budou tvořit jiné zdrojové stromy. [5]



Obr. 2.1: Zdrojový strom [5]

Sdílený strom (Obr. 2.2)

Sdílený strom má kořen pouze na jednom místě v síti, který se nazývá rendezvous point (RP), a nezáleží na tom, kdo zrovna data posílá. Může existovat pouze jeden RP v jednom autonomním systému (AS). Pro sdílený strom se používá označení (*, G), kde * (Star) značí „any source“ vysílání dat a G (Group), jako v předcházejícím případě, adresu multicastové skupiny. Tyto stromy se ještě rozdělují na jednosměrné a obousměrné. U obousměrného stromu zdroj vysílá jak ke kořenům tak po směru stromu k listům. U jednosměrného stromu pošle zdroj data pouze ke kořenu a ten je pak rozešle k listům. [5]



Obr. 2.2: Sdílený strom [5]

2.2.3 Protokoly zdrojových a sdílených stromů

Dense mode protokoly pracují na push principu a používají se na zdrojových stromech. Push princip vychází z předpokladu, že na každém subnetu (S, G) zdrojového stromu je multicastový provoz přenášen téměř všude a téměř každý účastník chce odebírat multicastová data. Jelikož jsou multicastová data tímto způsobem posílána všem účastníkům v subnetu, účastníci, kteří do dané multicastové skupiny nepatří, musí požádat o nezasílání dat. To je v dense mode protokolech zajištěno prune zprávou. Účastník, který pošle prune zprávu směrem ke kořenu bude odpojen od skupiny. Tato zpráva se musí periodicky zasílat ke kořenu, protože má omezenou

časovou platnost. Tímto způsobem se „ořeže“ strom o větve bez účastníků dané multicastové skupiny. Pokud ovšem přibude nový posluchač, pošle router zprávu graft ke kořenu pro navázání spojení. [5]

Protokoly využívající dense mode:

- *PIM-DM (Protocol Independent Multicast - Dense Mode)* používá směrovací unicastové tabulky jiných unicastových protokolů. RFC 3973 [1]
- *DVMRP (Distance Vector Multicast Routing Protocol)* je nejstarší multicastový protokol, navržený Stevem Deeringem. Staví si vlastní unicastové tabulky (princip podobný RIP). Byl využit v první multicastové experimentální síti MBone. RFC 1075 [1]

Sparse Mode protokoly. Tyto protokoly pracují na pull principu a využívají sdílené stromy. Pull pracuje na uplně obráceném principu než push. Pull předpokládá, že multicastová data v síti nejsou primárně požadována. Takže účastník, který chce přijímat multicastová data si o ně první musí říci. To se dělá tak, že router nadřazený účastníkovy pošle ke kořenu stromu zprávu join. Platnost zprávy má opět omezenou časovou platnost a musí se periodicky zasílat. Sparse mode protokoly používají také graft zprávy v případě, kdy router nemá už žádného aktivního posluchače dané multicastové skupiny a nemá multicastová data komu předávat. V tomto případě se zpráva graft neopakuje. [5]

Protokoly využívající sparse mode:

- *PIM-SM (Protocol Independent Multicast - Sparse Mode)* používá směrovací unicastové tabulky jiných unicastových protokolů. RFC 2362, 4602 [1]

Link State protokoly

- *MOSPF (Multicast Open Shortest Path First)* je modifikací OSPF (Open Shortest Path First) pro multicast. Místo zpráv využívá informace o stavu linky a metriku. Všechny routery si vytvářejí vlastní zdrojové stromy. RFC 1584 [1]

2.2.4 Přihlášení a odhlášení koncové stanice ze skupiny

Přihlášení k multicastové skupině je na uzlech realizováno protokolem IGMP. Ten má tři verze IGMPv1 (viz RFC 1112), IGMPv2 (viz RFC 2236) a IGMPv3 (viz RFC 3376).

princip fungování IGMPv1

IGMP funguje na principu Query-Report. Chce-li se nějaká koncová stanice přihlásit do skupiny, musí poslat zprávu Host Membership Report s vyplněnou adresou této skupiny v poli Group Address a poslat paket IGMP na adresu skupiny. Aby byly informace o skupině aktuální, posílá router (každých 60 sekund) zprávu Host Membership Query na adresu 224.0.0.1 (adresa pro všechna multicastová zařízení v lokální síti). Koncové stanice bývají k routerům připojeny přes switche, které když zaslechnou od routerů zprávu Query, rozešlou ji na všechny své odchozí porty a čekají na zprávy Report od účastníků skupiny. Když k účastníkovy skupiny přijde Query zpráva, vygeneruje si číslo od 0 do 10 a počká stejný počet sekund, než začne posílat zprávu Report. Do deseti sekund tedy všichni účastníci pošlou zprávu Report switchi. Switch tedy ví, kolik má k sobě připojených účastníků skupiny a může poslat jednu zprávu Report svému nadřazenému routeru směrem ke zdroji. Každý router pak řeší odesílání zprávy Report nadřazenému routeru podobně. Když odposlechne stejnou zprávu Report při vyčkávací smyčce, nastane výjimka a svou zprávu Report neposílá. V případě, že účastník chce skupinu opustit, jednoduše přestane odpovídat na zprávu Query. Pokud nedostane router odpověď Report ani po třech nezodpovězených Query (po 3 minutách), přestane na tuto adresu vysílat multicastová data. [6]

rozdíly IGMPv2 a IGMPv3 od IGMPv1

V IGMPv2 je lépe vyřešeno opouštění multicastové skupiny účastníkem. Stačí zaslat zprávu Leave Group routeru.

Hlavním přínosem IGMPv3 je, když jste členem nějaké multicastové skupiny a nechcete přijímat data od všech zdrojů skupiny (*, G), ale pouze od určitého zdroje (S, G), můžete poslat zprávu o zařazení jen k tomuto zdroji. IGMPv3 Report se posílá na adresu 224.0.0.22 (všechny multicastové routery v lokální síti s podporou IGMPv3). Ostatní zdroje potom vědí, že tomuto účastníkovi nemají data posílat.

2.2.5 Nadbytečnost multicastových paketů

Princip TTL (Time To Live, RFC 3443)

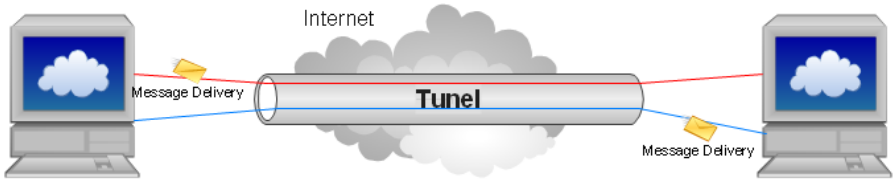
Nejjednodušším způsobem, jak zabránit v IP multicastu šíření nepotřebných či ztracených paketů, je stejně jako u unicastu parametr TTL (Time To Live) paketu. Jeho princip je jednoduchý, ale jeho účinnost obrovská. Pokud router přijme paket, tak mu sníží parametr TTL o jedna. Je-li TTL vyšší než nula, tak paket dále zpracovává. Pokud je TTL rovno nule, tak paket zahodí (vymaže z paměti). [7]

Princip RPF (Reverse Path Forwarding, RFC 5496)

Protože se data v multicastových přenosech duplikují, musely být do multicastových routovacích tabulek přidány ke každému směrovacímu údaji ještě údaje o adrese zdroje, ze kterého byl paket vyslán, a o portu, na který paket přišel. Důvod je takový, že s těmito informacemi pracuje RPF (Reverse Path Forwarding). Směrovač si udržuje unicastovou směrovací tabulku, na jejímž základě se rozhoduje, co bude s pakety dělat. Přijde-li routeru paket, podívá se odkud přišel (zdrojová adresa) a na jaký port. Pomocí unicastové tabulky provede test, jestli by přes stejný port poslal této adrese (zdrojové multicastové adrese) i unicastová data. Pokud ano, pošle směrovač podle multicastové směrovací tabulky multicastový paket dál ke svému cíli. Pokud ne, zahodí multicastový paket. Aby se test neprováděl s každým paketem, který má stejnou hodnotu cíle, zdroje a port, tak se výsledek testu uloží do cache. Pokud se změní unicastová routovací tabulka, musí test proběhnout znovu. [8]

2.3 Tunelování

Tunemování je obecně proces, při kterém jsou data jednoho typu spojení „zapouzdřena“ na jedné straně tunelu a přenesena sítí k druhému konci tunelu, kde jsou „odpouzdřena“. V praxi se tohoto využívá například při vytváření homogení sítě z geograficky oddělených poboček jedné firmy. Další využití je při propojení dvou počítačových sítí se stejným typem síťového protokolu (IPv4), kdy jsou tyto sítě propojeny odlišnou verzí IP protokolu (IPv6). Zde dochází k tunelování IPv4 paketů sítí s protokolem IPv6. [10]



Obr. 2.3: Tunel

2.3.1 Princip tunelování

Tunelování je založeno na skrytí původních dat zapouzdřením tak, aby transportní síť nevěděla jaká data jsou vlastně přenášena. Můžou se tedy zapouzdřovat jak segmenty do paketů a pakety do rámců, tak třeba rámce do rámců, či třeba rámce do transportních segmentů. To se děje přidáním záhlaví k původní datové jednotce, čímž se umožní přenos danou sítí. To umožňuje i šifrování přenášených dat.

Tunelování se provozuje na druhé a třetí vrstvě, což umožňuje podporu aktivních síťových prvku jako jsou směrovače a přepínače. Tunel je vytvořen mezi dvěma místy správcem sítě, který specifikuje typ přenášeného protokolu, typ tunelovacího protokolu, mechanizmus tunelování a adresy začátku a konce tunelu.

Tunel je původními zapouzdřenými daty chápán jako jeden skok, nezáleží tedy přes kolik směrovačů jsou zapouzdřená data směrována. Zapouzdřením se také zvýší šířka přenosového pásma o data v záhlaví, umožňující průchod původních dat transportní sítí. [10]

2.3.2 Tunelování na různých vrstvách

Tunely je možné vytvářet na spojové i síťové vrstvě dané síťové architektury.

tunelování na druhé (spojové) vrstvě: Tunelování rámců, které může být vyvoláno buď klientem, nebo přístupovým serverem bez iniciativy klienta. Při tunelování na spojovací vrstvě je tunel vytvořen, udržován a ukončen protokolem druhé vrstvy.

tunelování na třetí (síťové) vrstvě: Tunelovaná data (IP datagramy) musí být zapouzdřena do jiného datagramu (IPv6 v IPv4, IP v IP), tím se vyhneme problému špatné adresace v síti. Konfigurace tunelu se provádí obvykle manuálně.

2.3.3 Tunelovací protokoly

Tunelovací protokoly na druhé (spojové) vrstvě

Na druhé (spojové) vrstvě se používá protokolu L2TP (Layer Two Tunneling Protocol), který je popsán v RFC 2661. [1]

Ve vytvořeném L2TP tunelu je spojení PPP (Point-to-Point Protocol, RFC 1661), kterým jsou posílány L2TP pakety, obsahující L2TP hlavičku a posílaná data.

Dva koncové body L2TP tunelu jsou nazývány LAC (L2TP Access Concentrator) a LNS (L2TP Network Server). LAC je iniciátorem vytvoření tunelu s LNS serverem, který čeká na nové tunely. Po vytvoření tunelu je přenos dat mezi body obousměrný. Každá relace ve spojení je izolována protokolem L2TP, takže lze vytvořit několik virtuálních sítí jedním tunelem.

L2TP neposkytuje žádné šifrování ani zabezpečení. K zabezpečení L2TP paketů je využíváno IPsec protokolu. [10]

IPsec (RFC 2401, 2411) je bezpečnostní rozšíření IP protokolu na třetí (síťové) vrstvě OSI modelu. Vytváří logické kanály - SA (security agreements), které jsou

vždy jednosměrné. Při přijetí paketu může dojít k ověření, zda vyslaný paket odpovídá odesilateli či vůbec existuje. Před začátkem šifrování se obě strany dohodnou na formě šifrování paketu a poté dojde k zašifrování buď jen dat bez hlavičky nebo zašifrování celého paketu a přidání nové hlavičky. [9]

Tunelovací protokoly na třetí (síťové) vrstvě

Na třetí (síťové) vrstvě se používá protokol GRE (Generic Routing Encapsulation), který byl vyvinut firmou Cisco k vytváření virtuálních point-to-point spojení na síťové vrstvě. GRE je specifikován v RFC 2784. [1]

GRE tunely jsou koncipovány jako statické, což znamená, že každý konec tunelu si neuchovává informace o stavu nebo dostupnosti opačného konce tunelu. Důsledkem je, že výchozí směrovač nemá schopnost uzavřít cestu tunelu vytvořeným GRE protokolem, pokud je druhý konec tunelu nedostupný. Výchozí směrovač pouze klasifikuje tento tunel jako DOWN. [10]

2.3.4 VPN

VPN (Virtual Private Network) je privátní síť vybudovaná v rámci veřejné síťové infrastruktury, kterou může být internet. VPN nabízí možnosti tunelování, šifrování, autentizace a řízení přístupu najednou. Skládá se z bran VPN a klientů VPN. Brány VPN musí mít alespoň jedno rozhraní v privátní síti a druhé rozhraní ve veřejné síti (veřejná IP adresa). Při komunikaci dvou klientů VPN to potom vypadá následovně. První VPN klient zašle data přes svoji VPN bránu na privátní adresu druhého VPN klienta. VPN brána tato data s cílovou privátní adresou druhého VPN klienta zapouzdří do datagramu s veřejnou IP adresou VPN brány druhého VPN klienta. Druhá VPN brána přijmutý datagram odpouzdří a pošle na privátní adresu druhého klienta. VPN brána má tedy funkci řízení přístupu k privátní síti, autentizaci přístupu klienta do sítě a šifrování komunikace. VPN branou může být směrovač či firewall. [10]

Typy VPN

Pomocí VPN lze pokrýt velké množství požadavků na přístup do privátní sítě.

site-to-site připojení geograficky oddělených sítí. Zde se musí dbát na autentizaci jednotlivých sítí (ne klientů), aby nedošlo k útoku hackerů, kvůli statické adresaci.

remote access připojení z různých míst veřejné sítě, kdy brána VPN musí zajistit ještě přidělováni IP adres (DHCP server) a překlad adres (DNS). Toto klade velký důraz na autentizaci jednotlivých klientů. Výhodou je mobilita připojení.

Tunely ve VPN

Ve VPN lze vytvářet tunely jak na spojové, tak na síťové vrstvě. Při založení VPN na druhé vrstvě je získán dohled nad směrováním, zabezpečením a překladem adres. Tyto atributy jsou ztraceny při VPN na třetí vrstvě, kdy je dohled nad VPN přidělen provozovateli.
U VPN typu remote access se uplatňuje protokol L2TP a pro VPN typu site-to-site protokol IPSec nebo MPLS. [10]

MPLS (MultiProtocol Label Switching) je popsán v RFC 3031.
MPLS používá pro urychlení cesty paketů sítí princip přepínání značek, založený na důsledném oddělení procesu směrování od vlastního předávání paketů. Jedná se o protokol, který se nedá zařadit ani do spojové (vrstva 2) ani do síťové vrstvy (vrstva 3). Jeho vlastnosti totiž mají blízko k oboum z nich a proto je často nazýván jako protokol vrstvy 2,5. Důvod je, že byl navržen, aby poskytoval jednotný datový nosič pro služby založené na přepínání okruhů a přepínání paketů. Podle tohoto protokolu mají hlavičky paketů více informací než je nutné k uskutečnění dalšího hopu v síti. Proto si vytvoří FECs (Forwarding Equivalence Classes) „třídy s ekvivalentním směrováním“ a každou tuto třídu směruje na určitý směrovač. Rozhodnutí o tom, do které FEC jednotlivé pakety patří, se provede pouze jednou a paketu se přidělí label „nálepka“. Na dalším hopu potom není nutné znovu analyzovat paket, ale stačí podle labelu poslat dál. Tento protokol je využíván na páteřních sítích. Směrovače s podporou MPLS jsou označovány jako LSR (Label Switching Router). [11]

3 PRAKTICKÉ VYPRACOVÁNÍ

3.1 Aplikace pro přenos multicastových dat

Pro realizaci aplikací byla zvolena platforma JAVA. Jako vývojové prostředí bylo použito NetBeans IDE 6.1 (JDK 6 Update 13), které je dostupné ke stažení na stránkách www.netbeans.org.
Výhody:

- jednoduchost a rozšířenost mezi programátory
- objektově orientované programování
- přenositelnost
- bezpečnost
- nezávislost na architekruře operačního systému
- vícevláknové zpracování
- podporuje grafické rozhraní
- robustnost

Nevýhody:

- náročnost na hardware
- pomalý start

[12]

3.1.1 Návrh aplikací

Aplikace mají být navrženy pro přenos multicastových dat a být typu klient a server. Dalším požadavkem bylo vytvoření grafického uživatelského prostředí, aby bylo možné ověřit funkci multicastového provozu.

Proto byly navrženy dvě aplikace. Server pro posílání dat a klient, který data přijímá. Pro testovací účely aplikace vycházejí z jednoduššího principu přenášení unicastových dat s pozdější implementací příjmu stejných dat ze serveru větším množstvím klientů v multicastovém režimu. To znamenalo návrh grafického rozhraní s obsluhou jak unicastové tak multicastové komunikace.

Jako nosič dat byl zvolen datagram, poskytovaný protokolem UDP. UDP datagram je velmi jednoduchý ke zpracování a je v Javě dobře implementován. Zabývá

se jím balíček `java.net` ve kterém jsou obsaženy třídy `DatagramPacket` a `DatagramSocket`, na kterých komunikace mezi klientem a serverem funguje.

Při práci jsem vycházel z již napsaného kódu, který je volně šířitelný pod podmínkami společnosti Sun Microsystems.

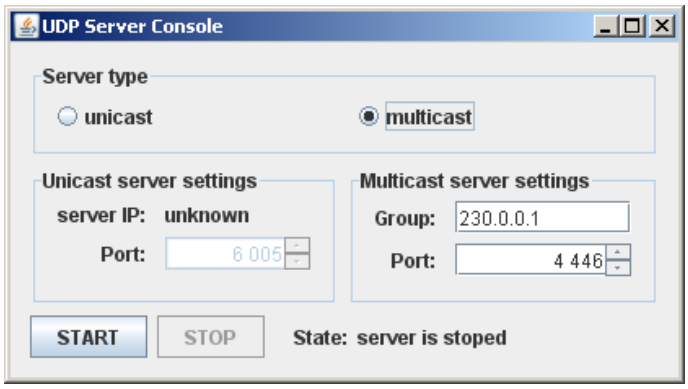
3.1.2 Java aplikace serveru

Aplikace serveru se skládá ze tří tříd.

- UDPServerStodulkaConsole - GUI formulář
- UDPServerStodulkaUnicastThread - logika unicastového vysílání dat
- UDPServerStodulkaMulticastThread - logika multicastového vysílání dat

UDPServerStodulkaConsole

Nejdříve si popíšeme formulář (Obr. 3.1) aplikace UDP Server.
Formulář je vytvářen ze swing komponent vývojového prostředí NetBeans.



Obr. 3.1: Grafické rozhraní aplikace server

Sekce Server Type: umožňuje výběr mezi unicastovým nebo multicastovým přenosem dat. Aby nedocházelo ke kolizím, je aktivní vždy jen jedna ze sekcí Unicast server settings a nebo Multicast server settings.

Sekce Unicast server settings: zde se nastavuje pouze port, na kterém bude aplikace naslouchat příchozím požadavkům. Je zde ještě pole *server IP*, které pomocí metody *InetAddress.getLocalHost()* zjisti IP adresu, kde je tento server spuštěn.

Sekce Multicast server settings: zde se provádí nastavení portu a identifikátor skupiny, do které se mají data ze serveru posílat.

Tlačítko START: vyvolá metodu *ButtonStartActionPerformed()*, která vytvoří instatnci unicastového nebo multicastového serveru (záleží na vybraném typu serveru v sekci Server type). Zároveň změní text stavu serveru na „server is running“ a zakáže všechny komponenty včetně ukončovacího křížku v horním rohu aplikace. Jediná povolená komponenta je tlačítko STOP.

Tlačítko STOP: vyvolá metodu *ButtonStopActionPerformed()*, která spustí metodu *ShutdownServer()* (zastavení serveru) z třídy `UDPServerStodulkaUnicastThread`. Pro zastavení obou typů serverů stačí volat pouze tuto jednu metodu, jelikož třída `UDPServerStodulkaMulticastThread` tuto metodu dědí z třídy `UDPServerStodulkaUnicastThread`. Metoda ButtonStopActionPerformed změní text stavu serveru na „server is stoped“ a znovu povolí všechny komponenty kromě sebe sama.

UDPServerStodulkaUnicastThread

Třída `UDPServerStodulkaUnicastThread` dědí z třídy `Thread`.
Toto je konstruktor třídy. Jako parametr volá konstruktor předka a port z formuláře. Pokud je vytvořen, vytvoří *DatagramSocket()* na daném portu. Toto je Socket, přes který server komunikuje s klientem. Konstruktor také otevře soubor *one-liners.txt*, který obsahuje seznam jednořádkových zpráv, které se budou posílat.

Je nutné mít soubor **one-liners.txt** ve stejné složce jako serverovou aplikaci!

```
public UDPServerStodulkaUnicastThread(String ThreadName,int port)
    throws IOException {
        super(ThreadName);
        socket = new DatagramSocket(port);
        socket.setSoTimeout(100);
        try {
            in = new BufferedReader(new FileReader("one-liners.txt"));
        }
    }
```

Zde je metoda *run()*, která je běhovou metodou celé třídy a přepisuje původní metodu *run()* z třídy `Thread`. Metoda *run()* obsahuje smyčku *while*, která pokračuje dokud jsou k dispozici jednořádkové zprávy nebo není ukončena tlačítkem STOP z formuláře. Během každého proběhnutí smyčky, vlákno čeká na *DatagramPacket()* přijatý přes *DatagramSocket()*. Přijatý *DatagramPacket()* indikuje požadavek od klienta. Odpovědí na požadavek je, že server vyjme jednořádkovou zprávu ze souboru, vloží ji do *DatagramPacket()* a pošle přes *DatagramSocket()* klientovi.

```
public void run() {
    while (moreQuotes && !isClosing.get()) {
        try {
            byte[] buf = new byte[256];
            // přijetí žádosti
            DatagramPacket packet =
                new DatagramPacket(buf, buf.length);
            try { socket.receive(packet);
            }
            catch (SocketTimeoutException e) {
                continue;
            }
            // vytvoření odpovědi
            String dString = null;
            if (in == null)
                dString = new Date().toString();
            else
                dString = getNextQuote();
            buf = dString.getBytes();
            // poslání odpovědi klientovi
            InetAddress address = packet.getAddress();
            int port = packet.getPort();
            packet =
                new DatagramPacket(buf, buf.length, address, port);
            socket.send(packet);
            Thread.sleep(10);
        }
        catch (InterruptedException ex) {
            Logger.getLogger
                (UDPServerStodulkaUnicastThread.class.getName())
                .log(Level.SEVERE, null, ex);
        }
        catch (IOException e) {
            e.printStackTrace();
            moreQuotes = false;
        }
    }
    socket.close();
}
```

Zde je ošetření konce jednořádkových zpráv.

```
protected String getNextQuote() {
    String returnValue = null;
    try {
        if ((returnValue = in.readLine()) == null) {
            in.close();
            moreQuotes = false;
            returnValue = "No more quotes. Goodbye.";
        }
    }
    catch (IOException e) {
        returnValue = "IOException occurred in server.";
    }
    return returnValue;
}
```

UDPServerStodulkaMulticastThread

Třída `UDPServerStodulkaMulticastThread` dědí vlastnosti z třídy `UDPServerStodulkaUnicastThread`. Toto je konstruktor třídy. Jako parametr volá konstruktor předka a identifikátor skupiny spolu s portem z formuláře.

```
public UDPServerStodulkaMulticastThread(String ThreadName, int port,
    String groupName) throws IOException {
    super(ThreadName, port);
        this.groupName = groupName;
        this.port = port;
    }
```

Běhová metoda *run()* je stejná jako u třídy `UDPServerStodulkaUnicastThread` s tím rozdílem, že multicastový server nečeká až ho požádá klient o data. Po spuštění začne sám posílat data na identifikátor skupiny. Další změnou je nastavení časovače po poslání každe zprávy a to náhodně v rozsahu 0 až 5 sekund.

```
public void run() {
    while (moreQuotes && !isClosing.get()) {
        try {
            byte[] buf = new byte[256];
            // vytvoření zprávy
            String dString = null;
            if (in == null)
                dString = new Date().toString();
            else
                dString = getNextQuote();
            buf = dString.getBytes();
            // poslání zprávy
            InetAddress group = InetAddress.getByName(groupName);
            DatagramPacket packet =
                new DatagramPacket(buf, buf.length, group, this.port);
            socket.send(packet);
            // usnutí
            try
                sleep((long)(Math.random() * FIVE_SECONDS));
            }
            catch (InterruptedException e) { }
        }
        catch (IOException e) {
            e.printStackTrace();
            moreQuotes = false;
        }
    }
    socket.close();
}
```

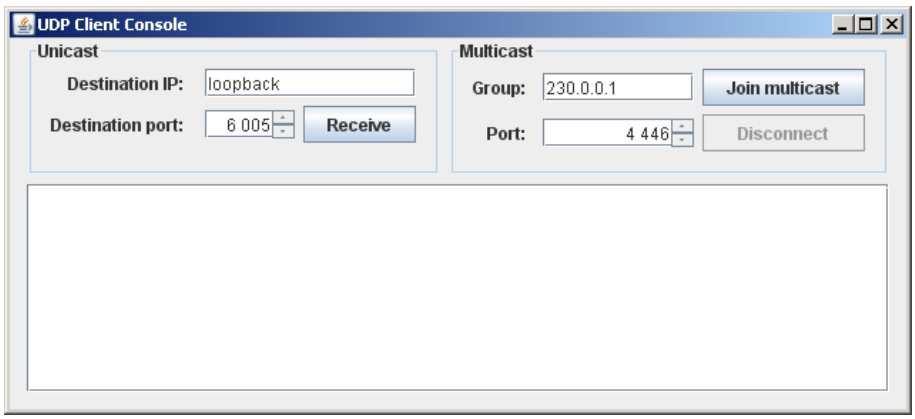
3.1.3 Java aplikace klienta

Aplikace klienta se skládá ze tří tříd.

- UDPClientStodulkaConsole - GUI formulář
- UDPClientStodulkaUnicast - logika unicastového klienta
- UDPClientStodulkaMulticast - logika multicastového klienta

UDPClientStodulkaConsole

Nejdříve si popíšeme formulář (Obr. 3.2.) aplikace UDP Client.



Obr. 3.2: Grafické rozhraní aplikace klienta

Sekce Unicast Zde se nastavuje IP adresa a port serveru, na který se má klient dotazovat. Po stlačení tlačítka *Receive* se vyvolá instance třídy `UDPClientStodulkaUnicast`, které se předává odkaz na instatnci formuláře. Na instatnci se vyvolá metoda *recieve()* s parametry vyplněnými ve formuláři. Výstup na dotaz je odchytáván a posílán metodě *showProgress()*.

```
private void ButtonReceiveUnicastActionPerformed
        (java.awt.event.ActionEvent evt) {
    try {
        new UDPClientStodulkaUnicast(this).receive(TextFieldIP.
            getText(), (Integer)SpinnerUnicastPort.getValue());
    }
    catch (Exception ex) {
        showProgress(ex.getMessage());
    }
}
```

Sekce Multicast Zde se nastavuje port a identifikátor skupiny, ze které budeme data odebírat. Tlačítko *Join multicast* vyvolá novou instatnci třídy `UDPClientStodulkaMulticast` spolu s parametry z formuláře a to do atributu *multicastClientThread*, aby bylo možné ho vypnout. Potom se samotné vlákno pustí s příznakem *start()*, zakáží se komponenty ovládání multicastu, kromě tlačítka *Disconnect*. Výstup se opět odchytává a posílá metodě *showProgress()*.

```
private void ButtonJoinMulticastActionPerformed
                (java.awt.event.ActionEvent evt) {
    try {
        multicastClientThread = new UDPClientStodulkaMulticast(this,
            "MulticastClientThread", (Integer)SpinnerMulticastPort.
            getValue(), TextFieldMulticastIP.getText());
        multicastClientThread.start();
        ButtonDisconnect.setEnabled(true);
        ButtonJoinMulticast.setEnabled(false);
        TextFieldMulticastIP.setEnabled(false);
        SpinnerMulticastPort.setEnabled(false);
        this.setDefaultCloseOperation(JFrame.DO_NOTHING_ON_CLOSE);
    }
    catch (IOException ex) {
        showProgress(ex.getMessage());
    }
}
```

V sekci multicast je ještě tlačítko *Disconnect*, které provede kontrolu, jestli je v atributu *multicastClientThread* uložena insatance vlákna. Pokud ano, tak vyvolá metodu pro ukončení vlákna *multicastClientThread.ShutdownClient()*, vyčistí atribut a znovu zapne komponenty ovládání multicastu.

```
private void ButtonDisconnectActionPerformed
    (java.awt.event.ActionEvent evt) {
    if(multicastClientThread != null) {
        multicastClientThread.ShutdownClient();
        multicastClientThread = null;
        ButtonDisconnect.setEnabled(false);
        ButtonJoinMulticast.setEnabled(true);
        TextFieldMulticastIP.setEnabled(true);
        SpinnerMulticastPort.setEnabled(true);
        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
```

```
    }
}
```

metoda showProgress() je metoda pro zobrazení přijaté jednořádkové zprávy. Jako parametr má řetězec *received*, který vytiskne textového pole aplikace klienta.

```
public synchronized void showProgress(String received) {
    String quote = received;
    System.out.println(quote);
    jTextArea1.append(quote+"\n");
    jTextArea1.setCaretPosition(jTextArea1.getText().length());
}
```

UDPClientStodulkaUnicast

nedědí z jiné třídy. Tato třída si bere odkaz na formulář, aby do něj mohla zapisovat.

```
public UDPClientStodulkaUnicast(UDPClientStodulkaConsole TheForm) {
    _theForm = TheForm;
}
```

Obsahuje pouze jednu metodu *receive()*, která má parametry *tempIP* a *port*. Vytvoří *DatagramSocket()*, přes který komunikuje se serverem a do něj vloží *Datagrampacket()*. Požadavek se vyšle pomocí *socket.send(packet)* a čeká na odpověď. Pokud odpověď neobdrží do konce vyčkávací smyčky (1 sekunda), zobrazí zprávu „Server did not respond!“. Pokud odpověď obdrží, vytiskne jí do formuláře pomocí metody *showProgress()*. V obou případech uzavře komunikační socket se serverem.

```
void receive(String tempIP, int port) throws IOException,
        InterruptedException {
    InetAddress address = InetAddress.getByName(tempIP);
    DatagramSocket socket = new DatagramSocket();
    socket.setSoTimeout(1000);
    byte[] buf = new byte[256];
    DatagramPacket packet =
        new DatagramPacket(buf, buf.length, address, port);
    try {
        socket.send(packet);
        packet = new DatagramPacket(buf, buf.length);
        socket.receive(packet);
        String received =
```

```
            new String(packet.getData(), 0, packet.getLength());
        System.out.println("Quote of the Moment: " + received);
        _theForm.showProgress(received);
    }
    catch (SocketTimeoutException e) {
        _theForm.showProgress("Server did not respond!");
    }
    socket.close();
}
```

UDPClientStodulkaMulticast

dědí vlastnosti z třídy `Thread`. Vlákno se uvádí do chodu metodou *start()*, která pak sama pouští běhovou metodu *run()*. Třída si opět bere odkaz na formulář, aby do něj mohla zapisovat.

```
public class UDPClientStodulkaMulticast extends Thread {
    private UDPClientStodulkaConsole theForm;
```

Konstruktor třídy zavolá konstruktor předka *super(ThreadName)*, uloží si odkaz na formulář, vytvoří síťové spojení (socket), připojí se do skupiny *socket.joinGroup(address)* a do formuláře napíše zprávu o připojení do skupiny.

```
public UDPClientStodulkaMulticast(UDPClientStodulkaConsole theForm,
        String ThreadName, int port, String multicastIP)
        throws IOException {
    super(ThreadName);
    this.theForm = theForm;
    socket = new MulticastSocket(port);
    socket.setSoTimeout(100);
    address = InetAddress.getByName(multicastIP);
    socket.joinGroup(address);
    theForm.showProgress("Multicast group joined.");
}
```

Přepisovaná metoda *run()* mateřské tříty `Thread` běží ve smyčce *while* dokud není zavolána metoda *Shutdown()*, která odpojí klienta od skupiny *socket.leaveGroup(address)*, uzavře komunikaci *socket.close()* a do formuláře vypíše „Multicast group disconnected.“. Jinak smyčka *while* čeká na příchozí pakety, rozbalí je a jednořádkové zprávy vytiskne do formuláře pomocí metody *showProgress()*.

```
public void run() {
    while (!IsClosing()) {
        DatagramPacket packet;
        byte[] buf = new byte[256];
        packet = new DatagramPacket(buf, buf.length);
        try {
            socket.receive(packet);
        }
        catch (SocketTimeoutException e) {
            continue;
        }
        catch (IOException e) {
            continue;
        }
        String quote = "Quote of the Moment: "
            + new String(packet.getData(), 0, packet.getLength());
        theForm.showProgress(quote);
        System.out.println(quote);
    }
    try {
        socket.leaveGroup(address);
    }
    catch (IOException e) {
        theForm.showProgress(e.getMessage());
    }
    socket.close();
    theForm.showProgress("Multicast group disconnected.");
}
```

3.1.4 Vývoj aplikací

Vývoj samotných aplikací byl prováděn na platformě Windows. Jednou z hlavních výhod jazyka JAVA je přenositelnost napsaného kódu v podobě bajtového kódu (bytecode), což je strojový jazyk JVM (Java Virtual Machine). Tento bajtový kód je možné spustit na všech operačních systémech (Windows, Linux, Solaris, Mac OS). V programovacím jazyce Java se veškerý zdrojový kód nejprve ukládá do neformátovaných textových souborů s příponou `.java`. Kompilátor je poté překládá do bajtového kódu, který má příponu `.class`. Aby mohl být bytecode interpretován, musí být spuštěn v prostředí JDK například NetBeans. Proto při kompilaci

projektu v NetBeans je vytvářen ještě soubor s příponou `.jar`, který se chová jako komprimovaný soubor a lze jej spustit, pokud je v operačním systému nainstalovaný JRE (Java Runtime Environment).

Jelikož jsou aplikace klienta i serveru vícevláknové, lze spustit unicastový i multicastový server a potom na jednom klientovy odchytávat multicastová data a přitom přijímat i unicastová data.

3.2 Linux

Jako distribuci Linuxu jsem si vybral Debian. Debian je velmi stabilní a proto je často využíván pro serverové instalace. Na desktopových stanicích se dnes již tak často nenachází a to zejména kvůli příchodu distribuce Ubuntu, která ovšem z Debianu vychází. Ze stránek www.debian.org/distrib/netinst jsem si stáhl poslední stabilní verzi s kódovým označením „lenny“ pro i386. Zvolil jsem soubor `debian-501-i386-netinst.iso`, který má velikost pouhých 180MB a některé části instalace stahuje z internetu.

3.2.1 Instalace Linuxu

Instalaci jsem prováděl v prostředí VMWare s použitím přemostění síťového spojení. Postup instalace:

- V „Installer boot menu“ zvoleno `Install`
- Language: `Czech`, Keyboard: `česká`
- Jméno počítače: `debian Stodulka`
- Doména: `Stodulka`
- Způsob rozdělení: `Asistované - použít celý disk`
- Vyberte disk pro rozdělení: `SCSI1(0,0,0) (sda)`
- Způsob rozdělení: `Samostatné oblasti pro /home, /usr, /var a /tmp`
- Ukončit rozdělování a zapsat změny na disk
- Zapsat změny na disk? `Ano`
- Heslo uživatele root: `a`
- Znovu zadejte heslo pro ověření: `a`

- Celé jméno nového uživatele: `Stanislav Stodůlka`
- Uživatelské jméno pro nový účet: `stodulka`
- Zadejte heslo pro nového uživatele: `a`
- Znovu zadejte heslo pro nového uživatele: `a`
- Země se zrcadlem archivu Debian: `Česká republika`
- Zrcadlo s archivem Debianu: `ftp.cz.debian.org`
- Informace o http proxy: `„prázdné"`
- Připojit se k průzkumu o nejpopulárnější balíky? `Ne`
- Zvolte programy k instalaci: `Standardní systém`
- Instalovat zavaděč GRUB do hlavního zavaděcího záznamu? `Ano`
- Instalace je kompletní: `Pokračovat`
- `//Restartování Debianu`

3.2.2 Instalace potřebných balíků

Vylistování balíků podle klíčového slova se provádí příkazem:

```
apt-cache search <klicove_slovo> | less
```

Instalace se provádí příkazem:

```
apt-get install <jmeno_baliku>
```

Byl nainstalován balík `openjdk-6-jre`. Obsahuje celý Java Runtime Environment, potřebný ke spouštění Java GUI a Webstart aplikací. Využívá Hotspot Zero. Balíky jsou vytvořeny s využitím podpory Iced Tea.

Druhý balík který byl nainstalován je `gdm`. GDM je zkratka pro Gnome Display Manager. Gdm je ekvivalent pluginu pro X, zobrazí přihlašovací okno a spouští X session. Funguje stejně jako „xdm“ včetně podpory XDMCP pro vzdálené X displeje. Úvodní okno je napsáno s použitím knihoven Gnome a tudíž vypadá jako aplikace Gnome včetně podpory témat. V základním nastavení běží z bezpečnostních důvodu pod neprivilegovaným uživatelem. To lze změnit po přihlášení neprivilegovaného uživatele z Systém - Správa - Přihlašovací okno - záložka Bezpečnost - zaškrtnout `Umožnit přihlašení místního správce systému`.

Gdm bohužel nenainstaluje balík terminálu pro přístup k shellu UNIX, takže byl doinstalován ještě balík `gnome-terminal`.

3.2.3 Ověření multicastové podpory rozhraním

Abychom ověřili, jestli síťové rozhraní podporuje multicast, spustíme v terminálu superuživatele (root) příkaz `ifconfig -a`. Pokud není ve čtvrtém řádku eth0 atribut MULTICAST, je pravděpodobné, že jádro nebylo zkompilováno s podporou multicastu. Před rekompilací jádra je možné zkusit povolit multicast příkazem `ifconfig <rozhraní> multicast`.

3.3 Mrouted

3.3.1 Popis daemona mrouted

Nejhojněji používaným daemonem pro IP multicastové směrování byl mrouted daemon. Mrouted byl navržen patronem multicastu Stevem Deeringem a právě na mrouted bylo postaveno směrování v nejrozsáhlejší světové multicastové síti Mbone. Mrouted patří mezi daemony postavené na dense mode protokolu DVMRP. K vytvoření lokální multicastové sítě stačí nastavit na sousedících směrovačích mrouted. Pokud ale chceme provozovat multicast mezi podsítěmi, které jsou odděleny unicastovými routery, které nepodporují multicast, mrouted obsahuje podporu pro tunely. Tunely jsou point-to-point spojení mezi páry multicastových routerů, umístěných kdekoli v internetu. IP multicastové pakety jsou zapouzdřeny pro přenos tunely, takže unicastovým routerům připadají jako unicastové pakety. Mrouted se sám nakonfiguruje pro směrování na multicastu schopná rozhraní a najde jiné DVMRP routery přímo dosažitelné pomocí těchto rozhraní. Pro potlačení defaultní konfigurace, nebo přidání tunelových spojů k jiným multicastovým routerům, jsou konfigurační příkazy umístěny v `/etc/mrouted.conf` (nebo v jiném souboru, definovaném pomocí `-c configfile`). Formát konfiguračního souboru je volné formy. Prázdná místa (i s novými řádky) nejsou důležitá.

3.3.2 Nevýhody mrouted

Problém mrouted je ten, že není implementován v nejnovějším jádře 2.6 a není dostupný ani v podobě balíků. Také nebyl velmi dlouho aktualizován. Poslední zmíňky, které byly nalezeny o stabilním používání mrouted, jsou na jádře 2.2 a několik málo informací o používání na jádře 2.4. Podle zdroje [13] byl mrouted vyjmut ze systému. Zdroj pro jinou distribuci [14] jako důvod udává DVMRP multicastový směrovací protokol, který se dnes již nepoužívá a byl nahrazen PIM multicastovým směrovacím protokolem.

3.4 Alternativy multicastového směrovacího daemona v Linuxu

Jelikož se daemon mrouted dnes pro multicastové směrování nepoužívá, byli jsme nuceni nalézt vhodné alternativy pro vyřešení problému směrování multicastových dat v Linuxu. Jako zdroj pro vhodné daemony bylo vylistování jiných balíků podporovaných OS Linux Debian. Jelikož už byl připraven počítač s fungujícím OS Linux a GUI rozhraním gnome, bylo možné spustit *Správce balíků Synaptic*, který našel po zadání klíčového slova *multicast* 67 balíku, jejichž obsah je nějakým způsobem spjat s multicastem. Po prostudování popisu všech vylistovaných balíků, bylo vybráno několik alternativ mrouted.

3.4.1 pimd

Pimd je implementací PIM daemona. Podporuje PIMv2-SM. Musí se v kernelu zapnout podpora PIM.

3.4.2 xorp

Xorp v sobě implementuje několik směrovacích protokolů pro IPv4 a IPv6 a je schopný je konfigurovat. Jsou v něm podporovány tyto síťové protokoly: BGP, OSPF, RIP/RIPng, IGMP/MLD a PIM-SM. Je schopny převádět procesy na daemony.

3.5 Alternativy tunelování v Linuxu

Linux s jádrem 2.6 v sobě obsahuje podporu statických GRE tunelů. GRE tunely jsou ale nešifrované a je tedy vhodné použít protokol IPsec. Proto byl v této práci použit balík *racoon*, který IPsec využívá.

4 TESTOVÁNÍ

Testování bylo prováděno v prostředí VMWare, kde byly použity následující typy konfigurací:

- PC.....Počítač nainstalovaný podle návodu výše spolu s balíky.
- NODE.....Uzel nainstalovaný podle návodu výše.

Použité balíky a jejich využití:

- *pimd* multicastový směrovací daemon (protokol PIM-SM, IGMP)
- *xorp* multicastový směrovací daemon (protokol PIM-SM, IGMP)
- *bird* unicastový směrovací daemon (protokol OSPF)
- *ssmping* testuje vytvořený PIM strom od klienta k serveru
- *psmics* balík utilit využívajících systém souborů PROC

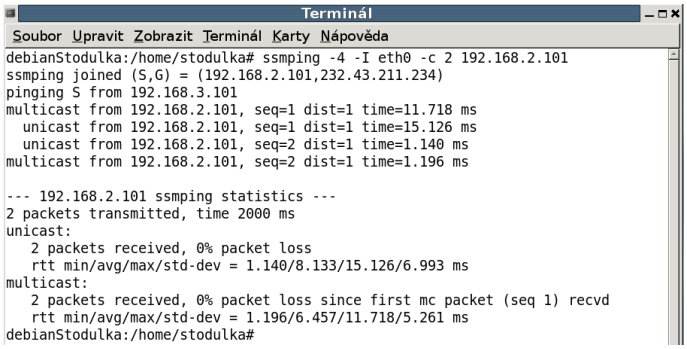
Některé použité příkazy a jejich využití:

- `apt-get install <balík>` instalace balíku
- `vi /etc/network/interfaces` konfigurace rozhraní
- `/sbin/sysctl -a` soubor systémových proměnných
- `ifconfig <rozhraní>` vylistování konfigurace rozhraní
- `route -n` vylistování dostupných sítí
- `pstree -Anp` stromová struktura běžících procesů (kontrola běhu daemonů)
- `ping <host>` prověření dostupnosti hosta
- `taceroute <host>` výpis uzlů směrem k hostu
- `cat <soubor>` vylistování souboru
- `echo <hodnota> > <soubor>` zapsání hodnoty do souboru

Popis funkce některých programů z balíků:

xorp Po nastavení hodnoty parametru „RUN“ na „yes“ v souboru */etc/default/xorp*. Proběhne po restartu automatické nastavení směrování multicastu (IGMP a PIM) na dostupných rozhraních.

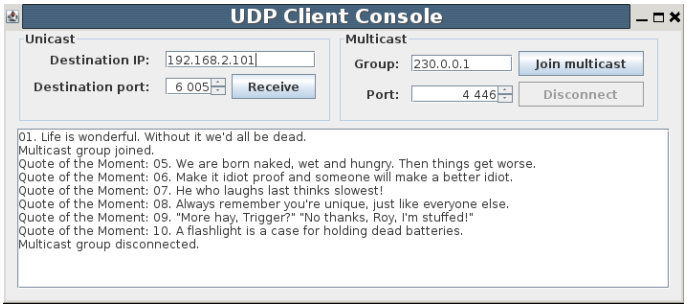
ssmping Na serverové stanici se spustí příkazem `ssmpingd` serverová část programu `ssmping`. Pokud jiný host v síti spustí klientskou část s parametry, otestuje program SSM spojení.Výstup je zobrazen na Obr. 4.1.



Obr. 4.1: Výstup ssmping dotazu na server

1. řádek je samotný dotaz z klintské stanice
2. řádek obsahuje zdroj z kterého jsou data vysílána a do jaké skupiny
3. až 7. řádek jsou přijatá data z multicastové skupiny

Popis výstupu klientské části java aplikace (Obr. 4.2)

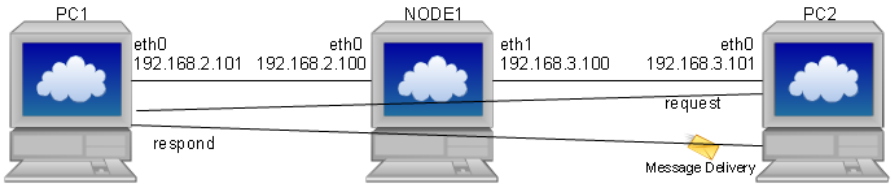


Obr. 4.2: Výstup klientské aplikace java

1. řádek unicastový dotaz
2. řádek zpráva o připojení do skupiny
3. až 8. řádek data ze serveru
9. řádek opuštění skupiny

4.1 PC1-NODE1-PC2

V této části byl testován multicastový a unicastový provoz dvou počítačů mezi nimiž byl jeden uzel a to podle Obr. 4.3.



Obr. 4.3: Topologie PC1-NODE1-PC2

PC1

Instalované balíky: ssmping
Konfigurace rozhraní:

```
auto eth0
iface eth0 inet static
        address 192.168.2.101
        netmask 255.255.255.0
        up ip ro add 192.168.3.0/24 via 192.168.2.100
        up ip ro add 224.0.0.0/4 via 192.168.2.100
```

Restartování.

NODE1

Instalované balíky: pimd, psmisc
Konfigurace rozhraní:

```
auto eth0
iface eth0 inet static
        address 192.168.2.100
        netmask 255.255.255.0
auto eth1
iface eth1 inet static
        address 192.168.3.100
        netmask 255.255.255.0
```

Restart.
Kontrola spuštěných daemonů: pimd

Kontrola parametrů systému:

- */proc/sys/net/ipv4/ip_forward=1* povolení směrování unicastových paketů
- */proc/sys/net/ipv4/conf/all/mc_forwarding=1* povolení směrování multicastových paketů

PC2

Instalované balíky: ssmping
Konfigurace rozhraní:

```
auto eth0
iface eth0 inet static
       address 192.168.3.101
       netmask 255.255.255.0
       up ip ro add 192.168.2.0/24 via 192.168.3.100
       up ip ro add 224.0.0.0/4 via 192.168.3.100
```

Restartování.

4.1.1 Test unicastového spojení

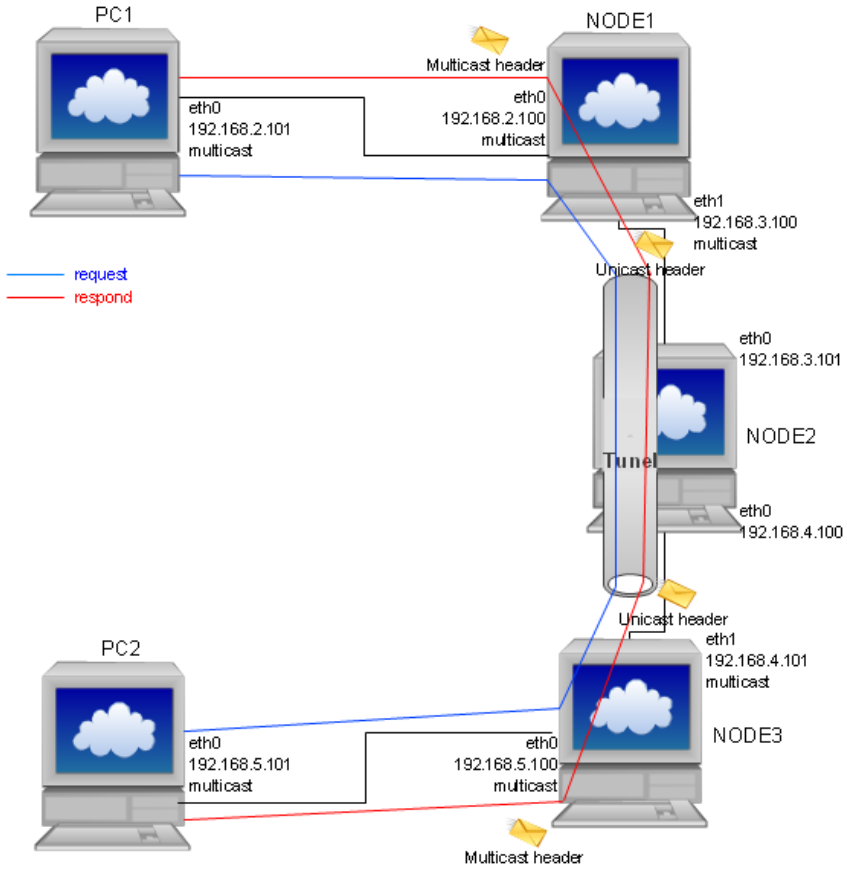
Unicastové spojení bylo otestováno pomocí příkazu `ping 192.168.2.101` z PC2. Dále byl proveden test pomocí aplikací Java. Na PC1 byl spuštěn server na který se úspěšně dotazoval klient z PC2.

4.1.2 Test multicastového spojení

Tento test byl prvně proveden pomocí programu *ssmping*. Na PC1 byla spuštěna serverová část příkazem `ssmpingd` a na PC2 klientská část příkazem `ssmping -4 -I eth0 -c 2 192.168.2.101` s podobným výstupem jako je na obrázku 4.1.
Dále byl proveden test pomocí aplikací Java. Na PC1 byl spuštěn server na který se úspěšně dotazoval klient z PC2.

4.2 PC1-NODE1-NODE2-NODE3-PC2

Zde byla provedena závěrečná simulace obsahující jak multicastový přenos tak tunelování multicastových dat sítí. Topologie zapojení byla provedena podle Obr. 4.4.



Obr. 4.4: Topologie PC1-NODE1-NODE2-NODE3-PC2

PC1

Instalované balíky: ssmping
Konfigurace rozhraní:

```
auto eth0
iface eth0 inet static
        address 192.168.2.101
        netmask 255.255.255.0
        up ip ro add 192.168.0.0/16 via 192.168.2.100
        up ip ro add 224.0.0.0/4 via 192.168.2.100
```

Restartování.

NODE1

Instalované balíky: xorp, bird, psmics
Konfigurace rozhraní:

```
auto eth0
iface eth0 inet static
       address 192.168.2.100
       netmask 255.255.255.0
auto eth1
iface eth1 inet static
       address 192.168.3.100
       netmask 255.255.255.0
auto tun0
iface tun0 inet static
       address 10.0.201.1
       netmask 255.255.255.0
       up ifconfig tun0 multicast
       pre-up iptunnel add tun0 mode gre
         remote 192.168.4.101 local 192.168.3.100
         ttl 255
       pointopoint 10.0.201.2
       up ip ro add 192.168.5.0/24
         via 10.0.201.2
       post-down iptunnel del tun0
```

Konfigurace balíků:

- xorp: hodnota paramtru RUN změněna na „yes“ v souboru */etc/default/xorp*
- bird: odkomentování sekce OSPF v souboru *etc/bird.conf*

Restart.
Kontrola spuštěných daemonů: bird, xorp
Kontrola dostupných sítí: 192.168.2.0/24 až 192.168.5.0/24
Kontrola parametrů systému:

- */proc/sys/net/ipv4/ip_forward=1* povolení směrování unicastových paketů
- */proc/sys/net/ipv4/conf/all/mc_forwarding=1* povolení směrování multicastových paketů

NODE2

Instalované balíky: bird, psmics
Konfigurace rozhraní:

```
auto eth0
iface eth0 inet static
       address 192.168.3.101
       netmask 255.255.255.0
auto eth1
iface eth1 inet static
       address 192.168.4.100
       netmask 255.255.255.0
```

Konfigurace balíků:

- bird: odkomentování sekce OSPF v souboru *etc/bird.conf*

Restart.
Kontrola spuštěných daemonů: bird
Kontrola dostupných sítí: 192.168.2.0/24 až 192.168.5.0/24
Kontrola parametrů systému:

- */proc/sys/net/ipv4/ip_forward=1* povolení směrování unicastových paketů

NODE3

Instalované balíky: xorp, bird, psmics
Konfigurace rozhraní:

```
auto eth0
iface eth0 inet static
       address 192.168.4.101
       netmask 255.255.255.0
auto eth1
iface eth1 inet static
       address 192.168.5.100
       netmask 255.255.255.0
auto tun0
iface tun0 inet static
       address 10.0.201.2
       netmask 255.255.255.0
       up ifconfig tun0 multicast
```

```
pre-up iptunnel add tun0 mode gre
  remote 192.168.3.100 local 192.168.4.101
  ttl 255
pointopoint 10.0.201.1
up ip ro add 192.168.2.0/24
  via 10.0.201.1
post-down iptunnel del tun0
```

Konfigurace balíků:

- xorp: hodnota paramtru RUN změněna na „yes“ v souboru */etc/default/xorp*
- bird: odkomentování sekce OSPF v souboru *etc/bird.conf*

Restart.
Kontrola spuštěných daemonů: bird, xorp
Kontrola dostupných sítí: 192.168.2.0/24 až 192.168.5.0/24
Kontrola parametrů systému:

- */proc/sys/net/ipv4/ip_forward=1* povolení směrování unicastových paketů
- */proc/sys/net/ipv4/conf/all/mc_forwarding=1* povolení směrování multicastových paketů

PC2

Instalované balíky: ssmping
Konfigurace rozhraní:

```
auto eth0
iface eth0 inet static
       address 192.168.5.101
       netmask 255.255.255.0
       up ip ro add 192.168.0.0/16 via 192.168.5.100
       up ip ro add 224.0.0.0/4 via 192.168.5.100
```

Restartování.

4.2.1 Test unicastového spojení

Unicastové spojení bylo otestováno pomocí příkazu `ping 192.168.2.101` z PC2. Dále byl proveden test pomocí aplikací Java. Na PC1 byl spuštěn server na který se úspěšně dotazoval klient z PC2.

4.2.2 Test multicastového spojení

Tento test byl prvně proveden pomocí programu *ssmping*. Na PC1 byla spuštěna serverová část příkazem `ssmpingd` a na PC2 klientská část příkazem `ssmping -4 -I eth0 -c 2 192.168.2.101` s podobným výstupem jako je na obrázku 4.1.
Dále byl proveden test pomocí aplikací Java. Na PC1 byl spušťen server na který se úspěšně dotazoval klient z PC2.

5 ZÁVĚR

Ve své bakalářské práci jsem vytvořil ve vývojovém prostředí Java NetBeans IDE 6.1 (JDK 6 Update 13) aplikace server a klient. Aplikace server pracuje v unicastovém a multicastovém módu a má za úkol posílat jednořádkové zprávy. Tyto se vysílají buďto na dotaz klienta v unicastovém módu nebo na identifikátor multicastové skupiny. Klientská aplikace tyto jednořádkové zprávy přijímá buďto na vyžádání v unicastovém módu nebo z identifikátoru multicastové skupiny. Pokud server vysílá v multicastovém módu, stejné jednořádkové zprávy může přijímat více klinetských aplikací na různých koncových stanicích a to i v jiných sítích než je serverová aplikace umístěna. To vše za předpokladu správné konfigurace uzlů podle mého návodu.

Dále jsem vytvořil v programu VMware Workstation 6.0.2, který slouží k virtualizaci jednoho nebo i více počítačů na jednom hostitelském počítači, vlastní testovací multicastovou síť založenou na operačním systému Linux distribuce Debian. Tato síť se skládala ze dvou koncových stanic, na nichž byly spuštěny vytvořené aplikace server a klient, a tři uzlů, na nichž bylo prováděno testování konfigurace multicastového směrování pomocí daemonů. Na prostředním uzlu byl zakázan multicastový provoz spolu s vypnutím multicstového směrovacího daemona a byl překlenut šifrovaným tunelovým spojením s podporou multicastu.

Ve své práci jsem také zdůvodnil, proč nebylo použito pro multicastové směrování daemona mrouted a jaké mohou být alternativy tohoto daemona. Tyto alternativní daemony jsem zdokumentoval a vytvořil názorný návod, jak mají být používaný.

LITERATURA

[1] IETF RFC Page *Request for Comments* [online]. 16.1.2008 [cit 24.5.2009]. Dostupné z URL: <http://www.ietf.org/rfc.html>.

[2] Wikipedie, otevřená encyklopedie *IP Multicast - Wikipedie, otevřená encyklopedie* [online]. 14.3.2009 [cit 24.5.2009]. Dostupné z URL: <http://cs.wikipedia.org/wiki/IP_Multicast>.

[3] Wikipedia, the free encyclopedia *IP multicast - Wikipedia, the free encyclopedia* [online]. 15.5.2009 [cit 24.5.2009]. Dostupné z URL: <http://en.wikipedia.org/wiki/IP_multicast>.

[4] NOSKA, M. *Technologie internetu (12): Multicast* [online]. 3.12.2007 [cit 24.5.2009]. Dostupné z URL: <http://computerworld.cz/internet-a-komunikace/technologie-internetu-12-multicast-1952>.

[5] FILIP, O. *Úvod do IP multicastu (díl třetí) - LUPA* [online]. 8.10.2004 [cit 24.5.2009]. Dostupné z URL: <http://www.lupa.cz/clanky/uvod-do-ip-multicastu-dil-treti/>.

[6] DEERING, S. *Host Extensions for IP Multicasting* [online]. August 1989 [cit 24.5.2009]. Dostupné z URL: <http://www.ietf.org/rfc/rfc1112.txt?number=1112>.

[7] AGARVAL, P.; BROCADE; AKYOL, B. *Time To Live (TTL) Processing in Multi-Protocol Label Switching (MPLS) Networks* [online]. January 2003 [cit 24.5.2009]. Dostupné z URL: <http://www.ietf.org/rfc/rfc3443.txt?number=3443>.

[8] WIJNANDS, IJ.; BOERS, A.; ROSEN, E. *The Reverse Path Forwarding (RPF) Vector TLV* [online]. March 2009 [cit 24.5.2009]. Dostupné z URL: <http://www.ietf.org/rfc/rfc5496.txt?number=5496>.

[9] THAYLOR, R.; DORASWAMY, N. ; GLENN, R. *IP Security Document Roadmap* [online]. November 1998 [cit 24.5.2009]. Dostupné z URL: <http://www.ietf.org/rfc/rfc2411.txt?number=2411>.

[10] RUŽMANOVÁ, R. *Virtuální privátní sítě pro vzdálený přístup* [online]. 7.9.2006 [cit 24.5.2009]. Dostupné z URL: <http://www.dsl.cz/clanky-dsl/clanek-511/virtualni-privatni-site-pro-vzdaleny-pristup>.

[11] ROSEN, A.; VISWANATHAN, A.; CALLON, R. *Multiprotocol Label Switching Architecture* [online]. January 2001 [cit 24.5.2009]. Dostupné z URL: <http://www.ietf.org/rfc/rfc3031.txt?number=3031>.

[12] ZAKHOUR, S.; HOMMEL, S.; ROYAL, J.; RABINOVITCH, I.; RISSER, T.; HOEBER, M. *Java 6, Výukový kurz.*Vydání první. Brno: Computer Press, a.s., 2007. 534 s.

[13] *Overview of mrouted source package* [online]. 28.3.2009 [cit 24.5.2009]. Dostupné z URL: <http://packages.qa.debian.org/m/mrouted.html>.

[14] *FreeBSD 7.0-RELEASE Release Notes* [online]. 24.2.2008 [cit 24.5.2009]. Dostupné z URL: <http://www.freebsd.org/releases/7.0R/relnotes.html>.

SEZNAM SYMBOLŮ, VELIČIN A ZKRATEK

DVMRP – Distance Vector Multicast Routing Protocol

FEC – Forwarding Equivalence Classes

GRE – Generic Routing Encapsulation

IGMP – Internet Group Management Protocol

JRE – Java Runtime Environment

JVM – Java Virtual Machine

L2TP – Layer Two Tunneling Protocol

LAC – L2TP Access Concentrator

LNS – L2TP Network Server

LSR – Label Switching Router

MOSPF – Multicast Open Shortest Path First

MPLS – MultiProtocol Label Switching)

OSPF – Open Shortest Path First

PGM – Pragmatic General Multicast

PIM-DM – Protocol Independent Multicast - Dense Mode

PIM-SM – Protocol Independent Multicast - Sparse Mode

PPP – Point-to-Point Protocol

RP – Rendezvous Point

RPF – Reverse Path Forwarding

TTL – Time To Live

UDP – User Datagram Protocol

VPN – Virtual Private Network

SEZNAM PŘÍLOH

A PRVNÍ PŘÍLOHA

A.1 Zdrojové kódy a spustitelné soubory

Ve složce Java na přiloženém médiu jsou uloženy projekty UDPClientStodulka a UDPServerStodulka. Ve stejné složce jsou i spustitelné archivy UDPClientStodulka.jar a UDPServerStodulka.jar spolu s textovým souborem one-liners.txt. Dále jsou ve složce Java spustitelné soubory UDPClientStodulka.exe a UDPServerStodulka.exe, které lze spustit jen v OS Windows.