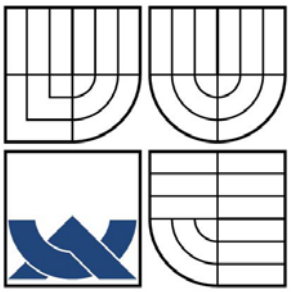
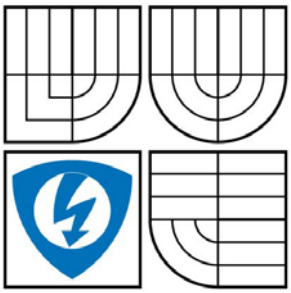


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKACNÍCH TECHNOLOGIÍ
ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF TELECOMMUNICATIONS

VIZUALIZACE SPOJENÍ MEZI POČÍTAČI

VISUALIZATION OF CONNECTION BETWEEN COMPUTERS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — JAN KOVÁŘ
AUTHOR

VEDOUCÍ PRÁCE — Ing. DAN KOMOSNÝ, Ph.D.
SUPERVISOR

BRNO 2008

LICENČNÍ SMLOUVA
POSKYTOVANÁ K VÝKONU PRÁVA UŽÍT ŠKOLNÍ DÍLO

uzavřená mezi smluvními stranami:

1. Pan/paní

Jméno a příjmení: Jan Kovář

Bytem: Rozsíčka 63, 679 74

Narozen/a (datum a místo): 2.6.1984 v Boskovicích

(dále jen „autor“)

a

2. Vysoké učení technické v Brně

Fakulta elektrotechniky a komunikačních technologií

se sídlem Údolní 244/53, 602 00, Brno

jejímž jménem jedná na základě písemného pověření děkanem fakulty:

.....

(dále jen „nabyvatel“)

Čl. 1
Specifikace školního díla

1. Předmětem této smlouvy je vysokoškolská kvalifikační práce (VŠKP):
   - □ disertační práce
   - □ diplomová práce
   - □ bakalářská práce
   - □ jiná práce, jejíž druh je specifikován jako .....

   (dále jen VŠKP nebo dílo)

Název VŠKP: Vizualizace spojení mezi počítači

Vedoucí/ školitel VŠKP: Jan Kovář

Ústav: Telekomunikací

Datum obhajoby VŠKP: .....

VŠKP odevzdal autor nabyvateli v*:

- □ tištěné formě – počet exemplářů …1……..
- □ elektronické formě – počet exemplářů …1……..

* hodící se zaškrtněte

2. Autor prohlašuje, že vytvořil samostatnou vlastní tvůrčí činností dílo shora popsané a specifikované. Autor dále prohlašuje, že při zpracovávání díla se sám nedostal do rozporu s autorským zákonem a předpisy souvisejícími a že je dílo dílem původním.
3. Dílo je chráněno jako dílo dle autorského zákona v platném znění.
4. Autor potvrzuje, že listinná a elektronická verze díla je identická.

Článek 2
Udělení licenčního oprávnění

1. Autor touto smlouvou poskytuje nabyvateli oprávnění (licenci) k výkonu práva uvedené dílo nevýdělečně užít, archivovat a zpřístupnit ke studijním, výukovým a výzkumným účelům včetně pořizovaní výpisů, opisů a rozmnoženin.
2. Licence je poskytována celosvětově, pro celou dobu trvání autorských a majetkových práv k dílu.
3. Autor souhlasí se zveřejněním díla v databázi přístupné v mezinárodní síti
   - □ ihned po uzavření této smlouvy
   - □ 1 rok po uzavření této smlouvy
   - □ 3 roky po uzavření této smlouvy
   - □ 5 let po uzavření této smlouvy
   - □ 10 let po uzavření této smlouvy

   (z důvodu utajení v něm obsažených informací)
4. Nevýdělečné zveřejňování díla nabyvatelem v souladu s ustanovením § 47b zákona č. 111/ 1998 Sb., v platném znění, nevyžaduje licenci a nabyvatel je k němu povinen a oprávněn ze zákona.

Článek 3
Závěrečná ustanovení

1. Smlouva je sepsána ve třech vyhotoveních s platností originálu, přičemž po jednom vyhotovení obdrží autor a nabyvatel, další vyhotovení je vloženo do VŠKP.
2. Vztahy mezi smluvními stranami vzniklé a neupravené touto smlouvou se řídí autorským zákonem, občanským zákoníkem, vysokoškolským zákonem, zákonem o archivnictví, v platném znění a popř. dalšími právními předpisy.
3. Licenční smlouva byla uzavřena na základě svobodné a pravé vůle smluvních stran, s plným porozuměním jejímu textu i důsledkům, nikoliv v tísni a za nápadně nevýhodných podmínek.
4. Licenční smlouva nabývá platnosti a účinnosti dnem jejího podpisu oběma smluvními stranami.

V Brně dne: …….....

…….. ……
Nabyvatel Autor

Anotace

Projekt Multicast IPTV Research Group se zabývá zejména vývojem nového algoritmu, využití stávajících a vývojem nových protokolů v rámci IPTV technologie. Ta bývá spojována s VOD (Video on demand) a internetovými službami jako např. přístup na web či internetová telefonie VoIP (Voice over Internet Protocol).

Tato práce se zabývá teorií IPTV, její historií a architekturou. Dále pak multicastem jako způsobem šíření skupinového vysílání, jeho vlastnostmi, směrováním skupinového vysílání a protokoly použitými pro přenos. Rozebírá možností grafických formátů pro vykreslování, jejich výhody a nevýhody. Konečným cílem této práce, která je řešena právě v rámci projektu Multicast IPTV Research Group, je aplikace, která čte a analyzuje data vzniklá hierarchickou sumarizací. Pomocí zvoleného grafického prostředku graphviz vykresluje zdrojová data ve formátu svg. Takto vytvořené soubory se pak dají snadno prohlížet v novějších typech internetových prohlížečů.

Klíčová slova: IPTV, multicast, hierarchická sumarizace, graphviz, svg

Abstrakt

Project Multicast IPTV Research Group mainly deals with the proposing of new mathematical models and communication protocols within IPTV technology. This technology is associated with VOD (Video on demand) and internet services such as web pages or VoIP (Voice over Internet Protocol).

This bachelor thesis is focused on IPTV theory, history and achitecture. Analyse multicast, its attributes, routing and protocols used to broadcasting. Subtilize posibility of graphics formats for visualization, their advantages and disadvantages. Final objective of this project, which is solved on Multicast IPTV Research Group project, is application, which analyse input data generated by hierarchy summarization. By virtue of graphviz are generated output data in svg format. It is easy to view this svg files in later types of internet browsers.

Keywords: IPTV, multicast, hierarchy summarization, graphviz, svg

Obsah

Seznam použitých zkratek

IPTV	Internetová televize (Internet protocol television)
VOD	Video na vyžádání (Video on demand)
VoIP	Internetová telefonie (Voice over internet protocol)
RTP	"Real-time" transportní protokol (Real-time transport protocol)
IGMP	Protokol k řízení multicastových skupin (Internet group management protocol)
IP	Internetový protokol (Internet protocol)
DVMRP	Směrovací protokol s vektory vzdálenosti (Distance vector multicast routing protocol)
PIM	Směrovací protokol (Protocol independent)
MOSPF	Protokol první nejkratší cesty (Multicast open shortest path first)
LAN	Místní síť (Local area network)
HTML	Jazyk pro tvorbu hypertextových dokumentů (Hypertext markup language)
SVG	Vektorový grafický formát (Scalable Vector Graphics)
CPL	Licenční podmínky pro volné šíření softwaru (Common public license)

Seznam obrázků

Úvod do projektu

Na ústavu telekomunikací vznikl projekt, který se zabývá zejména vývojem nového algoritmu, využití stávajících a vývojem nových protokolů. Tento projekt nese název Multicast IPTV Research Group.

V rámci tohoto projektu existuje několik dílčích projektů, které jsou řešeny jako bakalářské práce. Mimo mé osoby spolupracují na projektu i tito lidé:

- **Scripty pro instalaci a konfiguraci operačního systému Fedora – Czerner Lukáš**
  Úkolem řešitele je tvorba skriptů pro instalaci a konfiguraci operačního systému Fedora, tedy systému, který má běžet na virtuálních serverech v síti Planetlab. Postupem času se ale ukázalo, že samotné uzly Planetlabu, resp. virtuální servery na nich provozované, jsou již tímto softwarem vybaveny a nevyžadují žádné další větší zásahy z naší strany. Bylo tedy dohodnuto s vedoucím projektu Ing. Danem Komosným, Ph.D., že cílem snažení bude sledování „zdraví“ používané části sítě a tvorba nástrojů (skriptů), které mají dalším účastníkům projektu usnadnit umístění, popřípadě stáhnutí, dalších potřebných souborů z či na virtuální servery.
- **Simulace zjištění polohy počítače v Internetu - Vrzal Tomáš**
  Cílem je navrhnout aplikaci, ve které si uživatel rozvrhne rozmístění počítačů a tím zajistí vstupní data. Tyto data projdou algoritmem pro sestavení tabulky potřebné pro vykreslení struktury stromu, která je jednotná pro všechny členy projektu IPTV. Dále bude program umožňovat poslat výsledné data do databáze nebo uložit je na pevný disk.

Cílem mého projektu – ***Vizualizace spojení mezi počítači*** – je vytvořit aplikaci, která bude číst data z externího souboru, vytvořené Tomášem Vrzalem, a dle dat uložených v tomto souboru, bude vykreslovat stromovou strukturu spojení jednotlivých serverů a podřízených stanic pomocí Graphvizu. Tato práce podrobněji rozebírá problematiku IPTV, multicastu, možnosti vizualizace, resp.grafických formátu a konečné řešení daného problému za využití grafického softwaru Graphiz a C# aplikace.

1. IPTV

[1] IPTV (Internet protocol television) je systém služeb digitální televize, které jsou doručovány pomocí protokolu IP. Zatímco u nás je popularita IPTV a služeb s ní spojené teprve na vzestupu, v zemích, kde je IPTV více rozšířené, se často spojuje s VOD (Video on demand) a internetovými službami jako např. přístup na web či internetová telefonie VoIP (Voice over Internet Protocol).

1.1. IPTV - Historie

[1], [3] Poprvé bylo zahájeno vysílání televize přes internet v roce 1994 společností American Broadcasting Company. O rok později byl firmou Precept Software navržen a vytvořen formát internetového videa pojmenovaný IP/TV kompatibilní s aplikacemi na bázi Windows i Unix dosahující DVD kvality a používající protokol RTP (Real-time Transport Protocol), resp. RCTP (Real-time Control Protocol).

V minulosti byla tato technologie omezena nízkou šířkou pásma propustnosti. Avšak s nadcházejícími roky a narůstajícím počtem uživatelů, který strmě vzrostl a předpokládá se, že tento trend bude nadále pokračovat, vzrostla i šířka pásma pro IPTV. Jenom za rok 2005 bylo zaznamenáno více jak 200 miliónů aktivních uživatelů, přičemž do roku 2010 by se měl tento počet zdvojnásobit.

1.2. Architektura IPTV

[1], [2] Vysíláni IPTV může být buď zpoplatněné a nebo bezplatné. Již od června roku 2006 je k dispozici více než 1300 bezplatných IPTV kanálů, přičemž tento počet neustále roste. Tyto IPTV kanály požadují pouze internetové připojení a potřebné hardwarové zázemí, kterým může být obyčejné PC nebo třeba mobilní telefon podporující sít 3G.

Služba IPTV pokrývá jak živé vysílání, tak „video na vyžádání“ neboli VOD. Přehrávání IPTV vyžaduje buď osobní počítač nebo set-top-box připojený k televizi. Video využívá kompresní formát v kodeku MPEG-2 nebo MPEG-4, přičemž MPEG-2 je poslední dobou vytlačován kodekem H.264 (MPEG-4), a je doručováno přes IP multicast v případě živého vysílání nebo IP unicast pro VOD.

Standartní protokoly pro přenos jsou definovány následovně:

- Pro živé TV vysílání se používá protokol IGMP verze 2 pro spojení s tzv. „multicastovým proudem“ (multicast kanál) a pro změnu z jednoho multicastového kanálu na jiný
- VOD používá tzv. Real Time Streaming Protocol (RTSP)

1.3. Planetlab

I když mnou řešený semestrální projekt je spíše simulací řešení problému hierarchické sumarizace, v rámci komplexního projektu Multicast IPTV Research Group se používá prostředí Planetlab, na kterém by se mé řešení aplikovalo v praxi. V jiné části projektu totiž bude vytvořena hierarchická struktura, jež bude popsaná tabulkou v databázi (viz dále). Rozhodující pro tuto strukturu bude zřejmě doba odezvy mezi jednotlivými uzly, kterou měří známá funkce "ping". A právě v této fázi je velmi těžké nasimulovat takovou síť a není zcela jisté, zda-li bude simulace odpovídat reálu. Proto bude použita síť PlanetLab, kde lze algoritmy strukturalizace i vykreslení otestovat na skutečném prostředí a chování v síti.

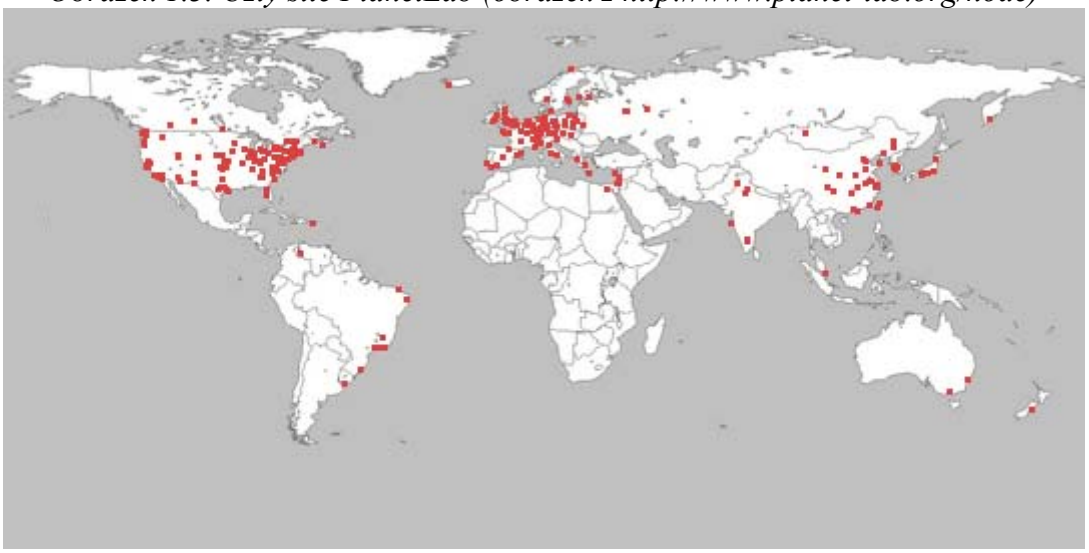
PlanetLab by se dal přirovnat k jedné velké laboratoři, sloužící pro výzkum nových síťových aplikací a technologií, složená z cca 700 nezávislých uzlů umístěných v různých koutech světa, včetně České republiky (máme v síti 3 uzly, "planetlab1.fit.vutbr.cz", "planetlab1.cesnet.cz" a "planetlab2.cesnet.cz").

Žádná firma sama o sobě nemá takové prostředky na vytvoření takovéto celosvětové sítě. Přispěním všech účastníků PlanetLabu (dnes cca 1000 institucí a firem) vznikla sít s uzly, které jsou dle určitých pravidel k dispozici všem uživatelům PlanetLabu k testování různých aplikací a protokolů tak, aby bylo zajištěno co nejvíce reálné prostředí. Na každém uzlu tedy může běžet spousta nezávislých aplikací najednou, na uzlech běží operační systém Fedora.

Tato síť je postavena na principu P2P (peer-to-peer) sítě, kdy se výměna dat neděje způsobem, jaký byl navržen před desítkami let, tedy že klient požádá o data server, nýbrž data se vyměňují mezi v podstatě decentralizovanými klienty navzájem bez potřeby serveru, ovšem s tím, že ten, kdo data žádá, musí si je nejprve vyhledat.

Flexibilita takovéto sítě spolu s dalšími výhodami vedla k tomu, že PlanetLab se stala prototypem pro tzv. Internet2, tedy "nová verze" internetu. Nebude se zřejmě jednat o úplně jiné a nové aplikace, ale spíše o takové, aby fungovali na stávajících systémech a přinesly změnu, bude se tedy jednat o síť, která bude nadstavbou a překryje stávající internet.

Obrázek 1.3. Uzly sítě PlanetLab (obrázek z http://www.planet-lab.org/node)



2. Multicast: skupinové vysílání

2.1. Základní vlastnosti skupinového vysílání

[4] Klíčovým cílem této technologie je zásadní odlehčení zátěže vysílajícího uzlu a přenosové soustavy při přenosech typu jeden zdroj - mnoho příjemců. Zdroj tedy vysílá data, určená neznámému, potenciálně velmi velkému počtu příjemců (skupině), pouze jednou a veškerá režie spojená s distribucí příjemcům je ponechána na přenosové soustavě, v prostředí Internetu tedy (v ideálním stavu) na směrovačích (routerech). Na nich také je, aby zajistily efektivní přenos dat od zdroje k příjemcům, tedy aby vysílaná data poslaly po každém spoji nejvýše jedenkrát, a to pouze tehdy, je-li daným směrem skutečně nějaký příjemce. Na rozdíl od klasického přímého vysílání (unicast), kdy přenos paketu dat od zdroje k cíli je iniciován zdrojem, je tok paketů skupinového vysílání určován příjemci.

K identifikaci skupin příjemců se používá speciální třída adres IP (třída D), zahrnující adresy z množiny *224.0.0.0 až 239.255.255.255*. Vysílající uzel odesílá pakety dat s cílovou adresou skupiny (a svou vlastní obyčejnou zdrojovou adresou). Další šíření přes směrovače by mělo (viz RFC 1112) probíhat stejnou metodou best effort (aneb dělám, co můžu) jako šíření běžných paketů přímého vysílání. V případě skupinového vysílání ovšem může směrovač provést replikaci paketu a jeho vyslání do více směrů.

2.2. Skupinové vysílání v lokální síti

[4] Protokoly na 2. vrstvě síťové hierarchie (v našich podmínkách je z nich daleko nejrozšířenější ethernet) obsahují ve svých specifikacích podporu skupinového vysílání v podobě speciálních MAC adres. Běžné síťové karty pracovních stanic (včetně PC) pak mají schopnost podle svého okamžitého nastavení (na základě požadavků programu) filtrovat pakety skupinového vysílání a nejbližším vrstvám programového vybavení již předávat jen relevantní část paketů skupinového vysílání, které se v lokální síti pohybují, tedy pouze skupiny, jež jsou předmětem momentálního zájmu dané stanice. Nedochází tedy k zatěžování stanic lokální sítě, jichž se dané skupinové vysílání netýká. Z výše řečeného vyplývá, že například k experimentu se skupinovým vysíláním v rámci lokální sítě může postačit běžné technické vybavení a příslušný aplikační program (a samozřejmě případné další technické prostředky, které jsou pro uvažovanou aplikaci potřebné, např. zvuková karta a reproduktory).

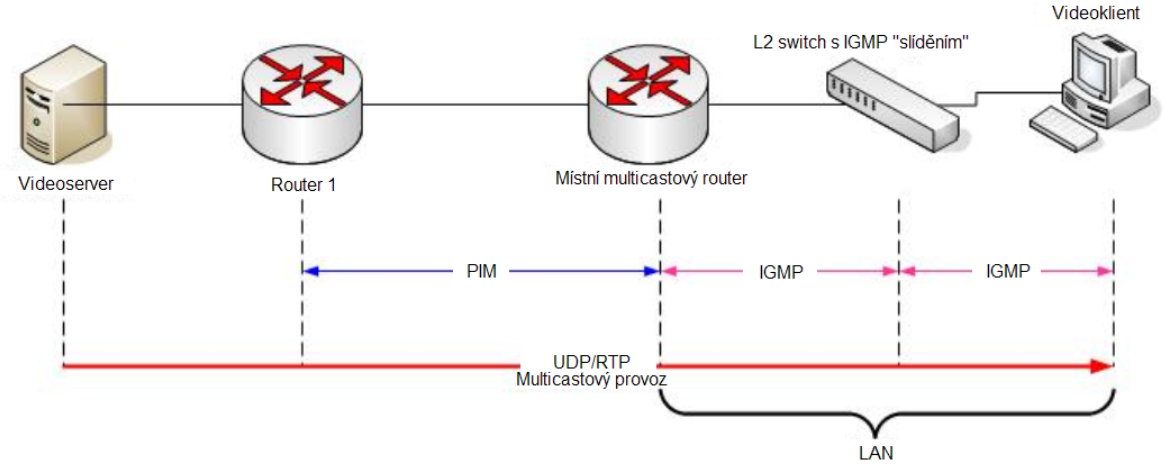
2.3. Přenos skupinového vysílání mezi sítěmi

[4], [5] Snadnost implementace skupinového vysílání v rámci lokální sítě se vytrácí, jakmile chceme dosáhnout přenosu v rámci propojených sítí. Do hry vstupují směrovače s jejich primárním úkolem získat informace o tom, které skupiny mají být vysílány do sítí, jež jsou ke směrovači bezprostředně připojeny. K tomuto účelu byl vyvinut speciální protokol IGMP, Internet Group Management Protocol. Jeho pomocí směrovač periodicky zjišťuje zájem stanic v připojených sítích o jednotlivé proudy skupinového vysílání.

Protokol IGMP neboli Internet Group Management Protocol je komunikační protokol k řízení jednotlivých IP multicastových skupin. Je používán IP hosty a přilehlými multicastovými routery k založení jednotlivých multicastových skupin.

IGMP je nedílnou součástí IP multicastové specifikace, pracující na síťové vrstvě. Je analogický k protokolu ICMP v unicastovém spojení.

Obr. 2.3. základní architektura sítě pro užívání multicastových služeb za pomocí IGMP protokolu



IGMP protokol je využíván jak klientským počítačem, tak i přilehlými switchi k připojení k místnímu multicastovému routeru. Mezi ním a vzdáleným routerem se pak používá protokol PIM neboli Protocol Indenpendent Multicast k řízení provozu mezi servery a jednotlivými klienty.

IGMP snooping neboli „slídění“, jak by se tento termín dal přeložit, je proces pro sledování IGMP provozu. Základní myšlenkou této metody je vlastnost, která povoluje přepínání tzv. „naslouchání“ v IGMP konverzaci mezi hosty a routery pomocí paketů IGMP v síťové vrstvě tzv. OSI modelu.

Pokud je IGMP snooping ve switchi aktivní, analyzují se veškeré IGMP pakety mezi hosty a multicastovymi routery v síti. Když je zachycena IGMP zpráva směrující od hosta směrem k multicastové skupině, switch přidá číslo portu účastníka do multicastového seznamu skupiny. Při zachycení „opouštecí“ IGMP zprávy účastníka pro změnu ze seznamu vyřadí.
IGMP snooping tak umí velmi efektivně redukovat provoz a tím i zatížení multicastu.

V praxi vše probíhá tak, že směrovač vyšle do připojené sítě dotaz (paket se speciální skupinovou adresou 224.0.0.1) a jednotlivé stanice odpovídají (s náhodně zvoleným zpožděním, aby nedocházelo k zahlcení sítě při současné odpovědi všech najednou) informací o adresách skupinového vysílání, o něž mají zájem. Odpovědi jsou rovněž vysílány na adresu 224.0.0.1 a odposlouchávány ostatními stanicemi. Tím se zamezí duplicitnímu vysílání požadavků na stejnou skupinu. Programové vybavení koncové stanice tedy musí navíc podporovat protokol IGMP. Směrovače tak pomocí protokolu IGMP sledují zájem o příjem konkrétních skupin ve svém bezprostředním okolí.

2.4. Směrování skupinového vysílání

Směrovače musejí - kromě trvalého mapování svého bezprostředního okolí - zajistit tok paketů skupinového vysílání i do vzdálených oblastí sítě, a to pokud možno optimálním způsobem. K tomu slouží tzv. směrovací protokoly. Jejich pomocí směrovače hledají minimální strom spojů pokrývající cestu od zdroje skupinového vysílání k momentálním zájemcům o příjem. Je zřejmé, že na rozdíl od klasického směrování přímého vysílání půjde o proces velmi dynamický. Cesta od daného zdroje k danému cíli je totiž stálá, pokud nedojde k nějaké vnější události měnící topologii sítě, např. poruše linky. Naproti tomu zájemci o příjem daného skupinového vysílání mohou vznikat a zanikat trvale a tento proces průběžných změn musejí směrovací protokoly vhodně reflektovat. V současné době se nejvíce používají protokoly DVMRP (Distance Vector Multicast Routing Protocol) a dvě varianty protokolu PIM (Protocol Independent Multicast).

2.5. Momentální dostupnost a aplikace skupinového vysílání

[5] Dosavadní výklad, kdy jsme nerozlišovali mezi běžným směrovačem a směrovačem podporujícím skupinové vysílání, mohl vést k dojmu, že podpora skupinového vysílání je organickou součástí všech směrovačů, a to od nepaměti. Skutečnost je však trochu složitější. Přestože různé normy pamatují na skupinové vysílání již delší dobu, naráželi vývojáři aplikací skupinového vysílání dlouhou dobu na tvrzení výrobců směrovačů, že skupinové vysílání není třeba podporovat, protože nejsou jeho aplikace. Cesta z této pasti vedla přes implementaci speciálních skupinových směrovačů (multicast router, mrouter) do pracovních stanic připojených do lokálních sítí experimentujících se skupinovým vysíláním. Jejich vzájemné propojení zprostředkovaly tzv. tunely, kdy pakety skupinového vysílání byly zabaleny do

přímého vysílání mezi mroutery. V současné době je situace o poznání příznivější, prakticky každý průmyslově vyráběný směrovač podporuje skupinové vysílání a některý směrovací protokol. I tak ovšem nejsou vyloučena nepříjemná překvapení, zejména při změnách verzí programového vybavení směrovačů.

IP multicast je metoda přeposílání IP datagramů z jednoho zdroje skupině více koncových stanic. Místo odesílání jednotlivých datagramů ke každému cíli je odeslán jediný datagram. IP směrování přenosu multicast bylo vyvinuto, aby doplnilo technologie unicast a broadcast, které učinně nezvládaly nové aplikace. Adresace a přenosy multicast umožnují např. více hostitelům přenést jediný datagram IP.

Broadcast je poslední adresa v daném segmentu sítě. Při poslání paketu na tuto adresu se tento rozešle všem uzlům v dotyčném segmentu sítě. Broadcast adresa se používá například pro zjištění MAC adres cílového počítače pomocí ARP.

Broadcast adresa se dá zjistit podle IP adresy a masky podsítě. Například pro síť, v níž je počítač s IP adresou a maskou:

192.168.8.4 255.255.255.0

je IP adresa broadcastu:

192.168.8.255

MAC adresa broadcastu je vždy:

FF-FF-FF-FF-FF-FF.

V rozsáhlých sítích může odeslání na broadcast adresu způsobit její zahlcení.

2.6. Protokoly

2.6.1. Internetový protokol se skupinovým adresováním IGMP

Protokol IGMP dynamicky registruje jednotlivé hostitele, patřící do skupiny adres D. Hostitel identifikuje členství ve skupině odesláním zpráv protokolu IGMP a data zasílá vždy všem členům skupiny. Směrovače používající protokol IGMP pravidelně naslouchají zprávám protokolu IGMP a systematicky odesílají dotazy s cílem zjistit, které skupiny jsou v síti LAN aktivní. Směrovače spolu komunikují pomocí dalších protokolů a pro každou skupinu připravují cesty pro spoje s přenosem typu multicast.

2.6.2. Směrovací protokoly IP multicast

[6] K identifikaci skupin náležejících k přenosu multicast a k vytváření cest pro každou skupinu se používá:

- PIM (Protocol Independent Multicast) - nezávislý prokol přenosu multicast.
- DVMRP (Distance Vector Multicast Routing Protocol) - směrovací protokol přenosu multicast s vektory vzdálenosti.
- MOSPF (Multicats Open Shortest Path First) - protokol první nejkratší cesty pro přenos multicast.

2.6.3. Nezávislý protokol PIM

Tento protokol je popsán v RFC 2362 a popisuje dva režimy chování pro:

- Hustý provoz - kde PIM využívá proces reverzního zaplavování cesty.
- Režim PIM pro řídký provoz - je navržen pro sítě s mnoha datovými toky, ale s malým počtem LAN. Definuje společné místo v síti (rendezvous point), které se používá jako registrační bod pro usnadnění správného směrovámí paketů.

2.6.4. Směrovací protokol multicast s vektorem vzdálenosti DVMPR

Protokol používá reverzní techniku zaplavování cesty. Je definován v RFC 1075 a je základem pro páteřní IP síť. Protokol má některé nedostatky, např. špatně se přizpůsobuje škálování sítě, což má za následek násobné zaplavování, hlavně u verzí bez pročistovacího algoritmu. Zpětné zaplavování cesty vyžaduje, aby směrovač odeslal kopii paketu při jeho přijetí do všech cest. Poté směrovač odešle zpět zprávu ke zdroji, aby zastavil datový tok, pokud je směrovač napojen na LAN, jenž si nepřeje přijímat data příslušné skupiny s přenosem multicast.

Další metody DVMRP protokolu:

- Opětovné zaplavování, kdy DVMRP směrovače periodicky znovu zaplavují síť, s cílem dosáhnot žádaného nového hostitele. Zaplavovací mechanismus bere v úvahu frekvenci a dobu zaplavování, nutnou pro nového člena skupiny multicast, aby mohl zahájit příjem skupinového datového toku.
- DVMRP unicast se používá ke stanovení rozhraní, které vede zpět ke zdroji datového toku.

2.6.5. Protokol první nejkratší cesty s přenosem multicast MOSPF

Používá směrovací protokol s adresací unicast a požaduje po každém směrovači v síti, aby znal všechna dostupná spojení. Směrovač MOSPF vypočítává cestu od zdroje ke všem členém skupiny. Jakmile směrovač přijme provoz, vypočtená cesta je uložena do doby, než dojde ke změně topologie a novému výpočtu.

3. Možnosti řešení zadaného projektu

Řešení mého problému spočívá ve čtení dat vzniklých hierarchickou sumarizací. Ta obnáší sumarizaci došlých dat z koncových uzlů na podřazených serverech. Tzn. z došlých paketů jsou přečteny informace, tyto informace jsou ukládány do paměti. Tyto data jsou cyklicky čtena a z ních se vypočítávají jejich průměrné hodnoty. Tyto hodnoty jsou pak jako jeden paket posílány o úroveň výše, tedy na hlavní server. Koncové uzly jsou rozděleny do dvou skupin: koncové uzly a sumarizační uzly. Koncové uzly se chovají jako běžní uživatelé. Sumarizační uzly zavádí novou funkci, sbírají data z náležících koncových uzlů, ty seskupují a z dat vytváří sumarizační zprávy, které poté posílají nadřazenému sumarizačnímu nebo již kořenovému uzlu.

3.1. Úložiště a formát dat

V rámci projektu IPTV Multicast Research group, na kterém, jak již bylo řečeno spolupracuje více lidí, byl určen formát dat pro identifikaci hierarchické sumarizace do databáze. Databáze byla zvolena hlavně pro svoji přehlednost a jednoduchost při její zpracování.

Obr. 3.1.1 Struktura databáze pro ukládání dat

Sloupec	Typ	Porovnávání	Vlastnosti	Nulový	Výchozí	Extra
id	bigint(11)			Ano	*NULL*	auto_increment
IP	bigint(12)			Ano		
IP_master	bigint(12)			Ano	*NULL*	
active	varchar(1)	utf8_czech_ci		Ano	y	
insert_time	timestamp		ON UPDATE CURRENT_TIMESTAMP	Ano	CURRENT_TIMESTAMP	

- *IP (BIGINT, délky 12)* - IP adresa uzlu. Kvůli rychlosti databáze bylo zvoleno zapisovat IP adresu v dekadikcých číslech bez teček s dolněním nul, tedy *82.168.1.100 -> 82168001100*. Pro tento účel je nutné v MySQL zvoli datový typ BIGINT, jelikož INT má rozsah pouze *< -2147483648 ; 2147483647>* nebo *<0 ; 4294967295 >*, kdežto BIGINT má rozsah *<-9223372036854775808 ; 9223372036854775807 >* resp. *<0 ; 18446744073709551615>*. Tento sloupec nemůže být nulový.
- *IP_master (BIGINT, délky 12)* - Sloupec slouží k uložení adresy nadřazeného uzlu (feedback target, viz výše ODKAZ). Opět se jedná o integer BIGINT. Takto tedy máme jednoznačně určeno, kde se uzel ve stromu nachází a jsme schopni strom vykreslit.
- *active (VARCHAR, délky 1)* - Pokud uzel ze sítě vypadne, bude na něj při jeho kontrole vysílán požadavek PING na odezvu. Nicméně odezva nebude žádná, proto se do databáze zapíše, že se již jedná o neaktivní prvek, tedy active="n". Pokud je prvek stále aktivní, jeho hodnota je "y". Takový postup zapsání, zda-li je prvek aktivní nebo neaktivní slouží k tomu, aby bylo možné odlišit při zpětném vyobrazení. Tento sloupec může být nulový.
- *insert_time (TIMESTAMP)* - Čas uložení prvku do databáze, má výchozí hodnotu on update_current_timestamp, což znamená, že MySQL sám do pole zapíše čas, kdy byl řádek vytvořen nebo změněn. Slouží opět ke zpětnému výpisu.

Z těchto hodnot je tedy možné sestavit strom a časově zpětně jej vypisovat. Reálná tabulka s konkrétními daty by teda mohla vypadat následovně:

Obr. 3.1.2 Ukázka databáze s daty

id	IP	IP_master	active	insert_time
2	192168001002	127000000001	y	2007-11-25 13:11:54
3	192168001001	127000000001	y	2007-11-25 13:11:54
4	192168002041	192168002030	y	2007-11-25 13:11:54
5	192168002040	192168002030	y	2007-11-25 13:11:54
6	192168002030	192168002001	y	2007-11-25 13:11:54
7	192168002015	192168002001	y	2007-11-25 13:11:54
8	192168002001	192168001001	y	2007-11-25 13:11:54
9	192168002002	192168001001	y	2007-11-25 13:11:54
10	192168002004	192168001002	y	2007-11-25 13:11:54
11	192168002010	192168001002	y	2007-11-25 13:11:54
12	192168002011	192168001002	y	2007-11-25 13:11:54
13	192168002042	192168002004	y	2007-11-25 13:43:50
14	192168002043	192168002042	y	2007-11-25 13:57:09
15	127000000001	*NULL*	y	2007-11-25 13:13:23
16	192168001002	127000000001	y	2007-11-26 16:13:37

V našem případě se úložiště dat zjednodušilo na ukládání do lokálního souboru s informacemi pouze o IP adresách. Formát ukládání je čistě textového formátu a to následovně:

127.0.0.1 123.456.789.0

přičemž první adresa je vždy nadřazena té druhé.

3.2. Vykreslování

3.2.1. Úvod ke grafickým formátům

S postupným rozšiřováním komunitních sítí a následně Internetu se zvyšovaly i požadavky uživatelů na přenos, zobrazování a sdílení grafických informací. Tato potřeba dále vzrostla s rostoucí komercionalizací Internetu, protože bylo zapotřebí rozumným způsobem přenášet a zobrazovat reklamu. V oblasti rastrové grafiky se používal grafický formát GIF, který byl v pozdější době částečně nahrazen formáty JPEG a PNG. Mnohem složitější je však situace okolo grafiky vektorové, která se paradoxně na WWW hodí mnohem více než grafika rastrová (například proto, že v samotných HTML stránkách je možné měnit velikost písma a stránka se automaticky znovu zalomí, ovšem rastrové obrázky mají stále stejnou velikost, nehledě na obecně menší nároky na kapacitu linek u grafiky vektorové). Adeptů na standard v oblasti webové vektorové grafiky nebylo a není málo. Mezi neúspěšné formáty patří WML, málo rozšířený je také DWF (formát navržený firmou AutoDesk pro sdílení výkresů na webu), všeobecně se neujalo ani využití DXF či PostScriptu. Částečným řešením – a nutno říci, že pro mnoho oblastí zdaleka ne špatným – je využití Flashe, jedná se však o uzavřený formát, což omezuje tvorbu grafických editorů, prohlížečů i pluginů do webových prohlížečů, které by tento formát podporovaly. Stejně tak je podobný problém s formátem SVG či s prvkem canvas, neboť tyto nové metody podporují pouze novější verze prohlížečů. Projedeme si tedy několik možností vizualizace a na konci kapitoly shrneme jejich vhodnost či nevhodnost v rámci mého projektu.

3.2.2. Canvas

[9] Canvas je nový prvek HTML, který umožňuje kreslit grafiku pomocí skriptování (obvykle JavaScriptem). Může to být například vykreslování grafů, komponování obrázku jednoduchých (i složitějších) animací.

<canvas> byl poprvé uveden firmou Apple pro Dashboard v Mac OS X a později implementován do Safari. Prohlížeče založené na jádře Gecko 1.8 a novějších, například Firefox 1.5, již také podporují tento nový element. Prvek <canvas> je obsažen ve specifikaci WhatWG Web applications 1.0, známé jako HTML 5.

Na rozdíl od SVG, canvas podporuje pouze jeden základní tvar – obdélníky. Všechny další musí být vytvořeny kombinací jedné či více cest. Naštěstí máme soubor kreslících funkcí cest, které umožní vytvářet tvary i velmi složité.

K vytvoření tvarů pomocí cest potřebujeme několik zvláštních kroků. První krok k vytvoření cesty je zavolání metody *beginPath*. Cesty jsou vnitřně uloženy jako seznam sub-cest (čáry, oblouky, atd…), které společně vytvoří tvar. Při každém volání této metody je tento seznam vymazán a my můžeme začít kreslit nové tvary.Druhý krok je volání metod, které určí vlastní

cesty k vykreslení. Třetí a nepovinný krok je volání metody *closePath*. Tato metoda zkouší uzavřít tvar vykreslením přímky z aktuálního bodu k počátku. Pokud již byl tvar uzavřen nebo existuje v seznamu jen jedna položka, tato funkce neudělá nic.

Posledním krokem je volání metody *stroke* a nebo *fill*. Voláním jedné z nich se v momentě vykreslí tvar na plátno. *stroke* je používána k vykreslení tvarů s obrysem, zatímco *fill* je používána k malování vyplněných tvarů.

Dále se používají pomocné funkce pro kreslení:

- čar - *lineTo(x, y)*
- kružnic - *arc(x, y, radius, pocUhel, konecnyUhel, protismeruHodinovychRucicek)*
- kvadratických křivek - *quadraticCurveTo(cp1x, cp1y, x, y)*
- Bézierových křivek - *bezierCurveTo(cp1x, cp1y, cp2x, cp2y, x, y)*

Jednotlivé křivky a tvary se mohou vzájemně doplňovat a vytvářet tak komplexní složité tvary.

Obr. 3.2.2 Příklad použití metody složitější kvadratické křivky a její zdrojový kód

```
// Příklad kvadratické křivky
ctx.beginPath();
ctx.moveTo(75,25);
ctx.quadraticCurveTo(25,25,25,62.5);
ctx.quadraticCurveTo(25,100,50,100);
ctx.quadraticCurveTo(50,120,30,125);
ctx.quadraticCurveTo(60,120,65,100);
ctx.quadraticCurveTo(125,100,125,62.5);
ctx.quadraticCurveTo(125,25,75,25);
ctx.stroke();
```

3.2.3. SVG

[8] Vektorový grafický formát SVG (*Scalable Vector Graphics*) byl navržen právě pro oblast webové grafiky, není to však zdaleka jediná oblast, ve které je možné se s tímto formátem setkat. Právě naopak: tento formát se postupně stává průmyslovým standardem pro přenos vektorové grafiky mezi různými platformami i aplikacemi. Dochází také k integraci knihoven pro práci se SVG do grafických uživatelských rozhraní operačních systémů. Vzhledem k tomu, že se implementace celého SVG ukázala jako poměrně problematická, vznikly dvě podmnožiny SVG (SVG Basic a SVG Tiny), přičemž se předpokládá, že aplikace určené například pro mobilní zařízení s omezeným výpočetním výkonem a menší kapacitou paměti budou používat právě tyto méně náročné, ale pro mnoho služeb více než dostačující, podmnožiny celého standardu.

Pro grafický formát SVG jsou nejdůležitější vlastnosti z uživatelského hlediska především to, že se jedná o plnohodnotný vektorový formát podporující základní geometrické tvary, cesty (obecné křivky), pokročilou práci s textem, průhlednost apod. Jedná se o vlastnosti, které můžeme v prakticky stejné podobě najít například i v PostScriptu či Portable Document Formatu (PDF). SVG však umožňuje i tvorbu animací a především interaktivitu – SVG tedy nemusí sloužit pouze k zobrazování statických či animovaných obrázků, ale může se nad ním vybudovat například interaktivní mapový portál, geografický informační systém, jednodušší hry, grafické editory integrované do HTML stránek apod. To vše bez nutnosti vázat se na

jednoho dodavatele technologie, jeden prohlížeč či platformu. SVG soubory je možné komprimovat pomocí GZIPu a tím dále snížit velikost celého souboru, což se příznivě projeví zejména při pomalejším síťovém připojení.

Po softwarové stránce SVG jsou v současnosti dostupné pluginy (zásuvné moduly) do webových prohlížečů a rozšiřují se webové prohlížeče, které SVG podporují nativně, tj. bez nutnosti instalace zásuvného modulu. Prakticky všechny významné vektorové editory dnes SVG podporují. Jedná se jak o komerční produkty (Adobe Illustrator, CorelDraw, Xara pro Linux, Sketsa...) i o programy šířené jako open source (skvělý Inkscape, Sodipodi, Scribus...). Taktéž existuje mnoho programů, které dokáží SVG importovat či exportovat (OpenOffice.org, FreeMind, animační programy...).

SVG je zajímavý i z programátorského hlediska. Jedná se totiž o XML aplikaci, tj. grafické objekty i přidružené informace (například animace) jsou uloženy v XML formátu odpovídajícímu oficiálně vydané specifikaci. Každý SVG dokument je možné před vlastním zpracováním zkontrolovat na validitu, což zjednodušuje další práci. Díky XML je možné měnit, přidávat či ubírat jednotlivé uzly stromu, ze kterého se dokument skládá a ovlivňovat tak výslednou podobu kresby.

Základním objektem pro tvorbu vektorových obrázků jsou v případě SVG takzvané cesty (paths). Cestou je myšlena polyčára složená z úseček, kruhových i eliptických oblouků a Bézierových kvadratických a kubických křivek. Cesta může být vykreslena různým způsobem, je možné měnit barvu i styl čáry, způsob zakončení lomených čar, tloušťku vykreslované stopy apod. Cestu je také možné vyplnit, přičemž je podporována průhlednost.

Vzhledem k tomu, že je SVG určené i pro síťové aplikace, ve kterých je mnohdy kritický objem dat (delší přenosové časy), jsou jednotlivé segmenty cesty zapisovány v extrémně krátké formě: typ každého segmentu (úsečka, oblouk, Bézierova křivka) je určen jedním písmenem, mezi číselnými hodnotami nemusí být v některých případech vložen žádný oddělovač (tím může být znaménko) a jsou podporovány jak absolutní, tak i relativní posuny, což vede ke zmenšení rozsahu hodnot a tím i délky řetězců reprezentujících čísla.

Teoreticky je sice možné pro veškerou kresbu použít pouze cesty, SVG však umožňuje i práci se základními geometrickými tvary, se kterými se snáze pracuje (na druhou stranu je zápis pomocí cest kratší). Mezi tyto tvary patří obdélník, kružnice, elipsa, úsečka, polyčára (lomená čára) a uzavřený polygon. Podobně jako v případě cest, i u těchto tvarů je možné nastavit mnoho způsobů (stylů) vykreslení, včetně stylu okrajů, vnitřní výplně, průhlednosti a filtrů, které jsou na vykreslované objekty aplikovány při rasterizaci. Změnu tvarů je možné provádět programově (pomocí DOM) a naopak – ke každému tvaru je možné zaregistrovat funkce, které se provedou při výskytu nějaké události, například při kliknutí tlačítkem myši nad daným tvarem

Ve formátu SVG je možné vytvářet i různé nápisy či delší texty pomocí elementu text (například nápověda k programu Inkscape je vytvořena v SVG, nejde o textový dokument). Styl textu, tj. jeho barva, použitý font, velikost atd. se specifikuje podobným způsobem, jako v CSS, což opět zjednodušuje práci vývojářům, kteří již znají technologie HTML a CSS (ve skutečnosti se v SVG používá právě CSS 2). SVG samozřejmě podporuje Unikód, ostatně jako všechny XML dokumenty. Zajímavou vlastností – alespoň ve vztahu ke grafickému formátu – je možnost přidání vysvětlujícího (normálně neviditelného) textu k mnoha

elementům, čehož je možné využít například při zpracovávání kresby indexovacími roboty. Ve své podstatě se jedná o vyspělejší variantu atributu *alt* použitého v HTML u obrázků.

Příklad zápisu zdrojového kódu ve formátu SVG:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.0//EN"
"http://www.w3.org/TR/2001/REC-SVG-20010904/DTD/svg10.dtd">
<svg    width="200"
        height="200"
        viewBox="0 0 200 200"
        xmlns="http://www.w3.org/2000/svg"
        xmlns:xlink="http://www.w3.org/1999/xlink">

<!-- různobarevné vodorovné úsečky -->
<line stroke="red" stroke-width="4" x1="50" y1="20" x2="150" y2="20" />
<line stroke="blue" stroke-width="4" x1="50" y1="40" x2="150" y2="40" />

<!-- jednoduchý obdélník bez zakulacených rohů -->
<rect x="10" y="10" width="100" height="75" />

<!-- kružnice bez nastavených stylů -->
<circle cx="140" cy="140" r="50" />
```

Mnoho atributů v SVG, například umístění geometrických tvarů, jejich barvu či průhlednost, je možné průběžně měnit v čase a vytvořit tak animaci. Animace je zapsána podobným způsobem, jako například v částečně konkurenčním formátu Flash. Popsány jsou pouze klíčové stavy objektů, mezistavy se dopočítávají automaticky při vykreslování animace (možné jsou skokové přechody, lineární přechody a přechody zadané Bézierovými křivkami). Kromě deklarativně zapsané animace (tento způsob využívají především animační programy) lze samozřejmě využít i skripty, které postupně mění parametry zobrazovaných grafických objektů.

Vzhledem k tomu, že je celá vektorová kresba v paměti počítače reprezentována stromem a obraz tohoto stromu je uložen ve formátu XML, je možné pomocí DOM (Document Object Model) manipulovat s jednotlivými uzly stromu, podobně jako je možné manipulovat s jednotlivými elementy HTML stránky ("dynamické HTML"). Ke grafickým objektům lze zaregistrovat callback funkce a vytvářet tak kresby, které reagují na akce prováděné uživatelem.

V geografických informačních systémech se například může zobrazit mapa s vykreslenými parcelami. Po kliknutí na parcelu se (bez provedení nového dotazu na server) zobrazí podrobnější informace o parcele: její vlastník, výměr apod. Zařídit tuto funkcionalitu je poměrně jednoduché, protože dané informace mohou být uloženy například ve formě pole dostupného z JavaScriptového programu (samotné pole je samozřejmě dynamicky vygenerováno serverem, ovšem pouze při prvním zobrazení stránky obsahující SVG). Oproti systému založenému na rastrových obrázcích a neustálých dotazech na server (i pomocí AJAX) se jedná o mnohem rychlejší a uživatelsky příjemnější aplikaci.

Nejčastěji je pro dynamickou manipulaci se SVG používán JavaScript a to zejména z toho důvodu, že se jedná o programovací jazyk dostupný z prakticky jakéhokoli webového prohlížeče.

3.2.4. Java applet

Java je objektově orientovaný jazyk vyvinutý firmou SUn Microsystems a byla navržena jako snadno přenositelný jazyk. Výhodou vytvořené java aplikace je vyšší dostupnost. Tzn. že pokud bychom vytvořili applet, který bude dostupný na webových stránkách, bude stejný pro všechny OS. Výhodou je, že java je plnohodnotná aplikace, tzn. může otevírat jakékoli síťové spojení na jakémkoli portu a číst data. Také samotné vykreslování není nesdolatelnou záležitostí a je možné vytvořit uživatelsky přívětivé rozhraní. Problémem je, že uživatel musí mít nainstalovaná virtuální java stroj (JVM - Java Virtual Machine), tedy Java Runtime Enviroment.

3.2.5. Flash

Flash je vektorový grafický formát, určený zeména pro tvorbu animací a potažmo i webových stránek. Jeho nespornou výhodou je i podpora ActionScript, což je jazyk na řízení animací podobný JavaScriptu. Je možné tedy tvořit pokročilé interaktivní animace. Jeho částečnou nevýhodou je, v některých prohlížečích nemusí být plně podporován, i když se pomalu, ale jistě stává standartní součástí internetových stránek.

3.2.6. PHP script a GD knihovna

PHP je skriptovací jazyk určený pro tvorbu webu (stránek a webových aplikací), který pracuje na straně serveru. Mezi výhody PHP patří jeho relativní jednoduchost, podpora velkého množství webových a a souvisejících standardů a technologií, otevřenost a také dobrá komunikace s databázemi. Je tedy vhodný pro účely vizualizace, pokud bychom dynamicky generovali webové stránky, které by data zobrazovaly. Mezi nevýhody patří zejména jeho rychlost, jelikož se jedná o interpretovaný, nikoli kompilovaný jazyk.

GD knihovna je, dalo by se říci, "nadstavba" PHP umožňující práci s grafikou. Umožňuje vytváření základních grafických obrazců a práci s nimi, jakož i práci s externími soubory (vložené obrázky GIF, JPEG apod.). Její nevýhodou je rychlost a náročnost na server a zejména to, že výstupem GD knihovny je většinou rastrový obrázek (opět JPEG GIF, PNG apod), takže se nedá mluvit o interaktivitě takové grafiky. Je vhodná spíše pro dynamické vykreslování grafů, úpravy vlastností obrázků před vlastním vykreslení na webové stránce apod.

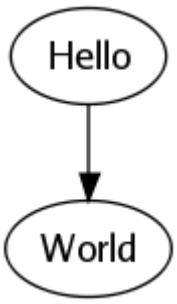
3.2.7. Graphviz

[7] Graphviz je grafický Open Source vizualizační software šířený pod licencí CPL. Obsahuje několik hlavních grafických výstupů, webové interaktivní grafické rozhraní a přídavné nástroje či knihovny. Jedná se o balík C++ knihoven a aplikací, která je možno jakkoliv editovat v rámci již uvedené CPL licence. Jako zdrojová data používá jednoduché textové příkazy v tzv. *dot* jazyce a vytváří diagramy v několika použitelných formátech a to buď jako jednoduché obrázky, SVG pro webové stránky, postcript pro použití jako PDF nebo přímo zobrazuje interaktivní diagramy v prohlížeči.

Graphviz má mnoho všestranných použití a možných prostředků pro zobrazení konkrétní hierarchické sumarizace za použití různých barev, fontů, stylů propojovacích čar či měnitelných tvarů. V praxi jsou pak jednotlivé grafické výstupy generovány z externích datových zdrojů, mohou ale být vytvořeny a editovány manuálně, například jako textové soubory typu RAW nebo přímo v grafickém editoru.

Jak již bylo řečeno, jako zdrojové příkazy pro graphviz se používají jednoduché textové

Obr. 3.2.7.1: Diagram "Hello world" a jeho zdrojový kód




```
digraph G
{
Hello->World
}
```

příkazy, které graphviz přeloží do zadaného výstupního formátu. Jak můžeme vidět na příkladu, základní stavební jednotkou pro určení „směru“ z jednoho prvku k druhému slouží sekvence znaků – a >, které tvoří tzv. kótovací šipku určující hierarchickou nadřazenost jednoho prvku nad druhým. Zdrojový soubor začíná hlavičkou *digraph* s názvem diagramu a složené závorky pak uvozují samotné příkazy pro zpracování textových příkazů.
Pro formátování konečného vzhledu pak rozlišujeme tři základní prvky, kterým můžeme přiřazovat různé vlastnosti. Jedná se o *node,* tj. uzel, *edge* neboli „kótovací šipka“, cesta od jednoho uzlu k druhému a *graph*, u kterého nastavujeme obecné vlastnosti pro celý diagram. Následující tabulky popisují vlastnosti, které se dají pro jednotlivé elementy *node, edge* a *graph* nastavit.

Tabulka 3.2.7.2: Vlastnosti pro node

Název	Defaultní hodnota	Hodnoty;popis
Color	Black	Barva uzlu
comment		Jakýkoliv text
distortion	0.0	pokřivení uzlu pro: *shape*=polygon
fillcolor	lightgrey/black	barva výplně
fixedsize	false	text uzlu nemá vliv na velikost obálky (*shape*)
fontcolor	black	barva písma
fontname	Times-Roman	název fontu
fontsize	14	velikost písma
group		název skupiny, do které patří uzel
height	.5	výška v palcích
label	node name	jakýkoliv text
layer	overlay range	*all*, *id* nebo *id:id*
orientation	0.0	úhel natočení uzlu
peripheries	shape-dependent	počet okrajů uzlu
regular	false	pravidelný tvar obálky pro *shape*=polygon
shape	ellipse	obálka uzlu; viz tabulka 3.2.7.5
shapefile		externí EPSF nebo SVG soubor pro shape
sides	4	počet stran pro *shape*=polygon
skew	0.0	zešikmení uzlu pro *shape*=polygon
style		bold, dotted, filled;
URL		URL asociované s uzlem
width	.75	Šířka v palcích
z	0.0	*z* koordináty pro VRML výstup

Tabulka 3.2.7.3: Vlastnosti pro edge

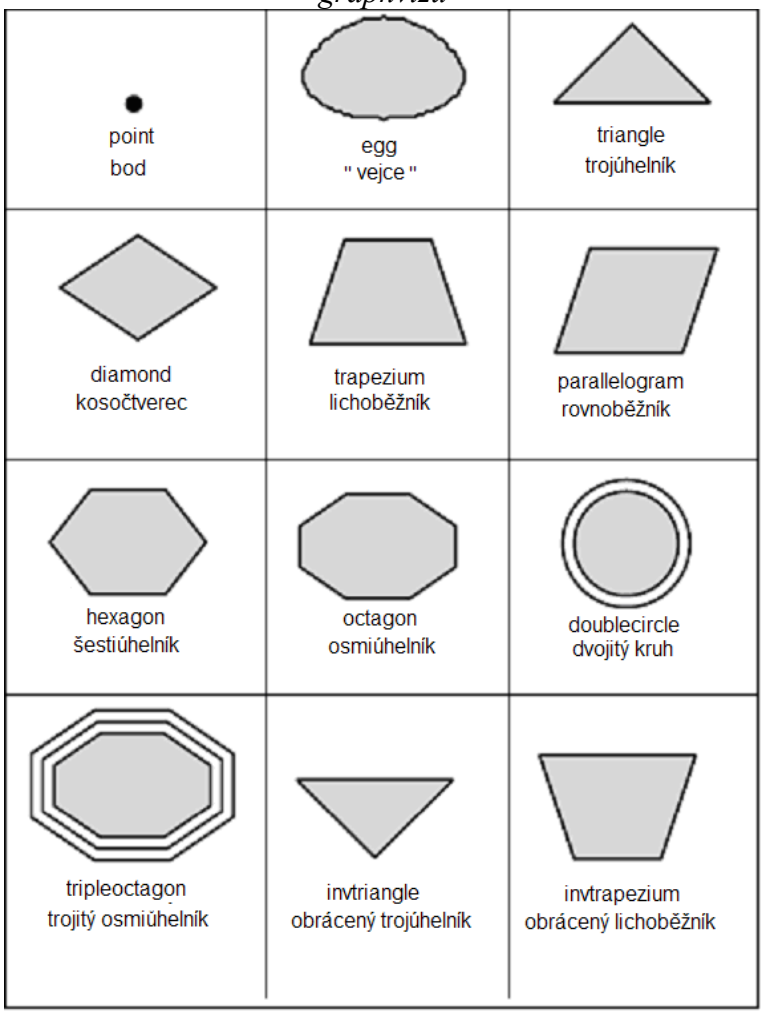
Název	Defaultní hodnota	Hodnoty;popis
arrowhead	normal	Styl kótovací šipky na začátku
arrowsize	1.0	měřítko
arrowtail	normal	Styl kótovací šipky na konci
color	black	Barva čáry
comment		Jakýkoliv text
constraint	true	Použití *edge* ovlivní pozici uzlu
decorate		Pokud je nastaveno, kreslí čáru spojující *labels* s jejich *edge*
dir	forward	forward, back, both, or none
fontcolor	black	Barva písma
fontname	Times-Roman	Název fontu
fontsize	14	Velikost písma
headlabel		label umístěn blízko začátku kótovací šipky
headport		n,ne,e,se,s,sw,w,nw
headURL		URL asociované, pokud výstupní formát je *ismap*
label		Označení *edge*
labelangle	-25.0	Úhel ve stupních, pokud začátek nebo konec *label* je pootočen
labeldistance	1.0	Meřítko pro odstup začátku nebo konce *label* od *node*
labelfloat	false	sníží omezení pro umístění *edge label*
labelfontcolor	black	Barva písma
labelfontname	Times-Roman	Název fontu
labelfontsize	14	Velikost písma
layer	overlay range	*all*, *id* nebo *id:id*
lhead		Název skupiny pro použití jako začátek *edge*
ltail		Název skupiny pro použití jako konec *edge*
minlen	1	Minimální vzdálenost mezi začátkem a koncem
samehead		tag pro *head node*; začátky *edge* se stejným tagem jsou směrovány do stejného portu
sametail		tag pro *tail node*; konce *edge* se stejným tagem jsou sloučeny do stejného portu
style		bold, dotted, filled;
taillabel		label umístěn blízko konce *edge*
tailport		n,ne,e,se,s,sw,w,nw
tailURL		URL asociované, pokud výstupní formát je *ismap*
weight	1	Celočíselná hodnota pro protažení *edge*

Tabulka 3.2.7.4: Vlastnosti pro graph

Name	Default	Values
bgcolor		Barva pozadí
center	false	Vykreslování na středu stránky
clusterrank	local	Může být *global* nebo *none*
color	black	Pro skupiny, barvy čar a výplní, pokud nejsou definovány
comment		Jakýkoliv text
compound	false	Povolí *edges* mezi skupinami
concentrate	false	povolí *edge* koncetrování
fillcolor	black	Barva výplně skupin
fontcolor	black	Barva písma
fontname	Times-Roman	Název fontu
fontpath		Cesta k externím fontům
fontsize	14	Velikost fontu
label		Jakýkoliv text
labeljust	centered	”l” a ”r” pro levé nebo pravé zarovnání návěstí skupin
labelloc	top	”t” a ”b” pro vrchní či spodní zarovnání návěstí skupin
layers		*id:id:id...*
margin	.5	Okraj stránky v palcích
mclimit	1.0	Měřítko pro minimální křížení iterakcí
nodesep	.25	Minimální vzdálenost mezi uzly v palcích
orientation	portrait	Pokud rotace není použita a hodnota je *landscape*, použije orientaci stránky naležato
page		Jednotka stránkování, např. "8.5,11"
rank		*same, min, max, source* nebo *sink*
rankdir	TB	LR "Left-To-Right" (z leva do prava) nebo TB Top-To-Bottom (z vrchu dolů)
ranksep	.75	Minimální vzdálenost mezi *ranks*, v palcích
ratio		přibližný poměr stran, *fill* nebo *auto*
rotate		Při hodnotě 90, nastaví *orientation* na *landscape*
size		maximální vykreslovací plocha, v palcích
style		Grafické nastavení, např. *filled* pro skupiny
URL		URL asociované s diagramem

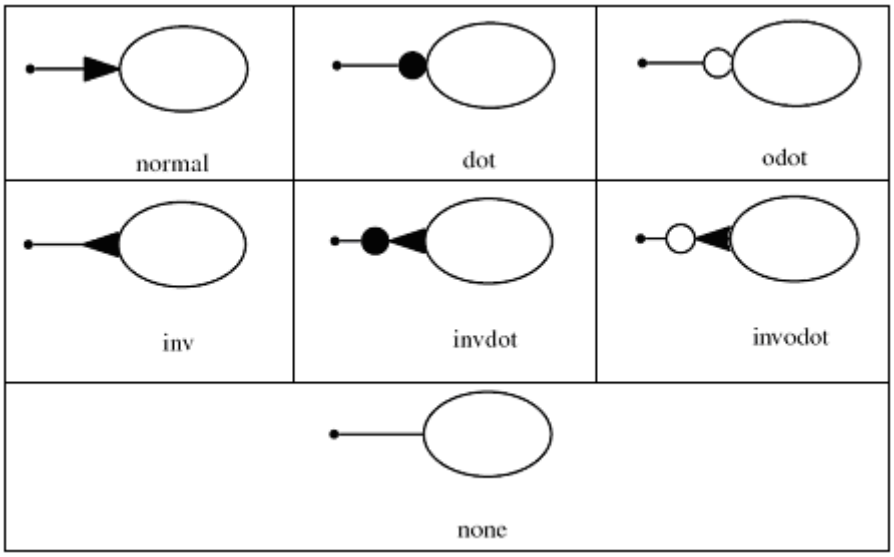
Jak vyplívá z tabulky 3.2.7.2, parametrem *shape* jde nastavit různý vzhled ohraničení pro uzel. Následuje přehled všech různých tvarů, které se dají použit.

Tabulka 3.2.7.5: vlastnoti pro parametr shape, jejich český ekvivalent a grafické vyjádření v graphvizu



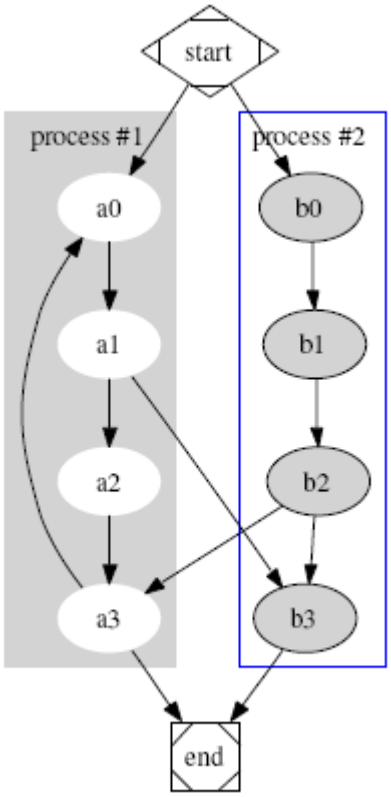
Stejně tak lze nastavovat různé zakončení pro kótovací šipky parametrem *arrowhead* pro element *edge*, jak ukazuje následující tabulka:

Tabulka 3.2.7.6: hodnoty pro parametr arrowhead a jejich grafické vyjádření v graphvizu



Možné použití různých parametrů včetně práce se skupinami demonstruje následující příklad:

Obr. 3.2.7.7: Příklad složitějšího diagramu včetně zdrojového kódu v jazyce dot




```
digraph G {
subgraph cluster0 {
node [style=filled,color=white];
style=filled;
color=lightgrey;
a0 -> a1 -> a2 -> a3;
label = "process #1";
}
subgraph cluster1 {
node [style=filled];
b0 -> b1 -> b2 -> b3;
label = "process #2";
color=blue
}
start -> a0;
start -> b0;
a1 -> b3;
b2 -> a3;
a3 -> a0;
a3 -> end;
b3 -> end;
start [shape=Mdiamond];
end [shape=Msquare];
}
```

Po té, co máme kompletní zdrojový soubor, voláme příkazovým řádkem graphviz, který nám tento soubor přeloží. Pomocí různých parametrů příkazového řádku pak můžeme ovlivnit výstupní zdroje:

-T*format* nastavuje výstupní formát. Možné hodnoty jsou následující:

canon..tzv. „prettyprint“ vstup, výsledkem není žádný layout
dot….. připsaný DOT, výstupem je opět layout ve formátu DOT s informacemi o zdroji připojenými jako atributy
fig….. FIG výstup
gd….. GD formát, používaný GD grafickými knihovnami
gif….. formát GIF
hpgl… HP/GL-2 vektorový grafický formát pro použití na HP plotterech
imap.. produkuje tzv. HTML map soubory použitelné v internetových stránkách na straně serveru, použitelné např. v kombinaci s -T*jpg*
cmap.. produkuje tzv. HTML map soubory na straně klienta (viz výše)
mif…. FrameMaker MIF formát
mp….. MetaPost
pcl….. PCL-5 výstup pro HP laserové tiskárny
pic….. PIC výstup
plain…jednoduchý ASCI formát,
png…. PNG (Portable Network Graphics) formát
ps……Postscript

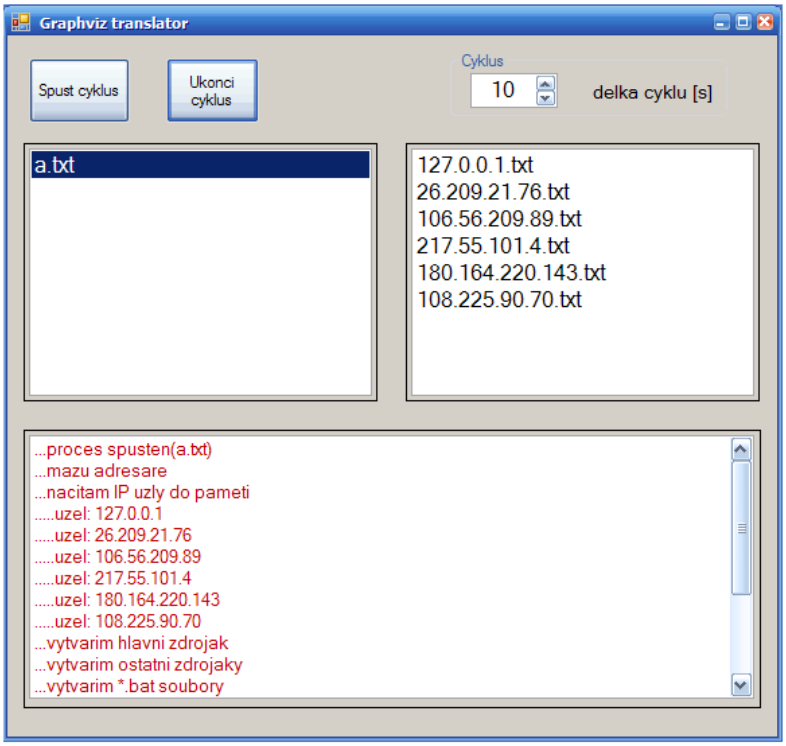
ps2…. Postscript s PDF anotací
svg…. SVG (Scalable Vector Graphics)
vrml…VRML výstup
vtx….. VTX formát
wbmp..Wireless BitMap formát

-G*name*=value nastavuje diagramu deafultní hodnoty atributů;analogicky funguje i –N a –E pro node a edge
-l*libfile* specifikuje cestu pro použití externí grafické knihovny
-o*outfile* výstup uložen do daného souboru
-V výstupem pouze číslo verze

4. Řešení zadané problematiky

Pro řešení zadaného problému jsem využil kombinace C# aplikace a Graphvizu, C# aplikace pro čtení zdrojových dat pro vizualizaci a jejich úpravu do jazyka *dot*, který pak přeložíme pomocí samotného graphvizu do formátu *svg*. Aplikace načte zdrojový soubor a z něho vygeneruje soubor(y) ve formátu *dot* pro překlad graphvizem do formátu *svg* a *htm* soubory. Po přeložení všech takto vygenerovaných souborů je výsledkem hlavní *svg* soubor *127.0.0.1.svg* a několik dílčích *svg* a *htm* souboru, na které odkazují jednotlivé IP adresy neboli uzly diagramu, jinými slovy *svg* soubory pro grafické vyjádření hierarchického postavení jednotlivých IP adres a *htm* soubory pro textové vyjádření IP adres patřící vždy pod jeden uzel, jak si ukážeme dále.
Vzhledem k tomu, že zdrojových souborů může být vícero, je program v jazyce C# koncipován na dvě části. První část načte seznam všech souborů, pak vybere první soubor, počká stanovený počet sekund, načte další zdrojový soubor a takto pokračuje stále dokola.

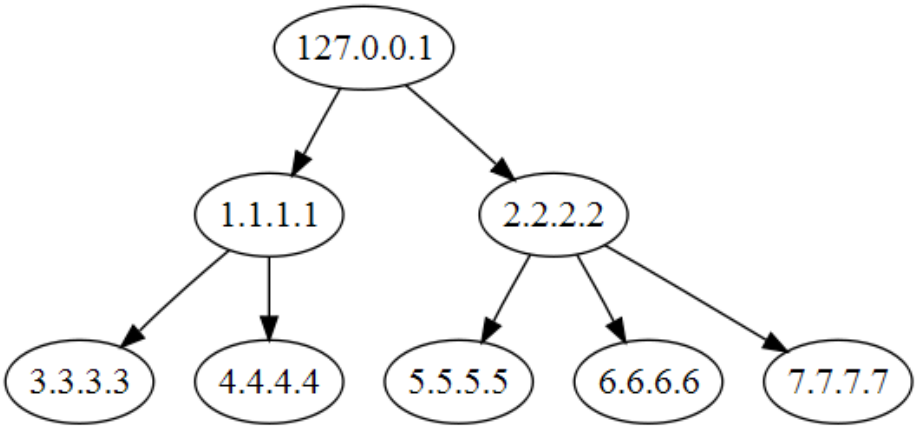
Obr. 4.1: Vzhled aplikace



Program má z vizuálního hlediska několik částí neboli komponent. Jak můžeme vidět z obrázku, celý cyklus se zapíná a vypínací pomocí dvou komponent *button* neboli tlačítek *"Spustit cyklus"* a *"Ukonči cyklus",* dále pak z komponenty *numericUpDown* pro určení času mezi cykly a ze tří komponent *listbox*. První, *listBox* zobrazuje všechny zdrojové soubory pro konverzi, druhý *listBox* zobrazuje seznam všech vytvořených zdrojových souborů pro překlad graphvizem a třetí *listBox* sloužící pro report neboli pro výpis zrovna prováděných akcí.
Celý program potom funguje tak, že načte obsah adresáře *SourceFiles*, z jednoho aktuálně vybraného souboru vytvoří několik dílčích souborů do adresáře *SourceForSVG*, které pak hromadně přeloží pomocí graphvizu. Tyto přeložené soubory **.svg* se též objeví v adresáří *SourceForSVG*.

Pro demonstraci si vysvětlíme běh programu na jednoduchém příkladu. Představme si hierarchické postavení tří úrovní IP adres dle obrázku:

Obr.4.2: Ukázkový diagram hierarchického postavení IP adres



Zdrojový soubor pro toto uspořádání by vzhledem k dříve řečenému (viz kapitola 3) vypadal následovně:

```
127.0.0.1  1.1.1.1
127.0.0.1  2.2.2.2
1.1.1.1  3.3.3.3
1.1.1.1  4.4.4.4
2.2.2.2  5.5.5.5
2.2.2.2  6.6.6.6
2.2.2.2  7.7.7.7
```

Naše aplikace vytvoří první hlavní soubor pro překlad graphvizem, který ctí jméno první IP adresy s příponou *txt*. Po překladu pak bude mít tento soubor příponu *svg*. První IP adresa bude vždy *127.0.0.1*, tudíž tento soubor se bude jmenovat *127.0.0.1.txt* a soubor přeložený pak *127.0.0.1.svg*. Do něho se vloží povinná hlavička, která obsahuje mimo základních příkazů pro vzhled diagramu také informaci o tom, s jakým souborem se zrovna pracuje:

```
digraph G {
ranksep="1";
ratio="auto";
node[shape=point];
nodesep="0.05";
edge[labeldistance=0.05];
```

edge[arrowhead=none];
node[heigth=.06,width=.06];
label="\n zdrojovy_soubor.txt";

Digraph značí, že se jedná o strukturovaný diagram, *ranksep* pak určuje minimální vzdálenost mezi jednotlivými elementy diagramu, *ratio* přibližný poměr stran a *nodesep* minimální vzdálenost mezi *node* neboli uzly. *Node[shape=point]* pak nastavuje všem uzlům vzhled tečky, parametry *height a width* předepisují šířku a výšku pro uzly. Pro kótovací šipky neboli *edge* pak parametr *labeldistance* určuje meřítko pro odstup začátku nebo konce návěstí od *node* a parametr *arrowhead=none* říká, že spojení mezi dvěma uzly bude vykresleno bez šipky, tzn. pouze čarou.

Dále program čte IP adresy a doplňuje je o povinné znaky, tzn. o uvozovky, a mezi dvě IP adresy vždy vloží sekvenci znaků "-" a ">". Konkrétně program pracuje tak, že načte IP adresu v levé části zdrojového souboru a poté prochází celý tento soubor a do nového souboru vždy přidá řádek s danou IP, se kterou se pracuje, a IP adresy, která k této přísluší. V praxi to pak znamená, že zdrojová data nemusejí být seřazena sestupně, ale i třeba náhodně. Po té, co jsou zapsána data pro první zvolenou IP, vloží se další formátovací atribut a sice *edge[style=dotted]*, který říká, že nyní budou kótovací čáry zobrazeny tečkovaně. V našem případě bude generovaný soubor pokračovat následovně:

"127.0.0.1"->"1.1.1.1"
"127.0.0.1"->"2.2.2.2"
edge[style=dotted];

Dále program pokračuje ve čtení IP adres a doplňování potřebných znaků podobným způsobem. Opět IP adresy doplní o potřebné znaky a pak specifikuje atributy pro druhou IP každého řádku a sice atribut URL s odkazem na soubor, který má název shodný s IP adresou, která je této nadřazena. V našem konkrétním případě jsou třetím řádkem zdrojového souboru IP adresy *1.1.1.1* a *3.3.3.3*. Program tedy toto vyhodnotí jako "1.1.1.1"->"3.3.3.3" a na druhý řádek pak vloží "3.3.3.3" a doplní jej o vlastnost *[URL ="1.1.1.1.htm"],* tedy *"3.3.3.3"[URL ="1.1.1.1.htm"]*. Tato vlastnost URL pak při překladu graphvizem způsobí, že při kliknutí na tuto IP dojde k přesměrování právě na zadaný soubor. Další doplněná část souboru *127.0.0.1.txt* bude tedy vypadat následovně:

"1.1.1.1"->"3.3.3.3"
"3.3.3.3"[URL ="1.1.1.1.htm"];
"1.1.1.1"->"4.4.4.4"
"4.4.4.4"[URL ="1.1.1.1.htm"];
"2.2.2.2"->"5.5.5.5"
"5.5.5.5"[URL ="2.2.2.2.htm"];
"2.2.2.2"->"6.6.6.6"
"6.6.6.6"[URL ="2.2.2.2.htm"];
"2.2.2.2"->"7.7.7.7"
"7.7.7.7"[URL ="2.2.2.2.htm"];

V této části program také vytváří ony **.htm* soubory, na které jednotlivé uzly odkazují. Při načtení IP adresy ze zdrojového souboru se totiž ihned vytvoří soubor s příponou *htm*, který má název shodný s touto IP. Do tohoto souboru se uloží povinné html tagy, tag pro vytvoření

odkazu na předchozí stránku a při každém načtení IP adresy příslušící té, se kterou se zrovna pracuje, se zapíše do tohoto *htm* souboru. V našem případě je zvolená IP *1.1.1.1* a k ní přísluší IP *3.3.3.3* a *4.4.4.4*. Vytvoří se tedy soubor *1.1.1.1.htm* a ten bude vypadat následovně:

```
<html>
<a href="127.0.0.1.svg">back</a>
<br>
3.3.3.3<br>
4.4.4.4<br>
</html>
```

Dalším krokem se vrátíme zpět do hlavního souboru *127.0.0.1.txt* a přiřadíme každé IP adrese, ze které pokračují další IP adresy, vlastnost *shape=ellipse* a *URL* s odkazem na *svg* soubor, který má stejný název jako tato IP. IP adresu, která není koncovým účastníkem poznáme snadno. Ve zdrojovém soboru se nachází vždy v levé části. Tudíž program projde všechny IP adresy v levé části zdrojového souboru a pro každou přiřadí danou vlastnost. Nakonec přidá patičku "}" a hlavní soubor je tímto hotov. V našem případě tedy:

```
"127.0.0.1"[shape="ellipse" URL="127.0.0.1.svg"];
"1.1.1.1"[shape="ellipse" URL="1.1.1.1.svg"];
"2.2.2.2"[shape="ellipse" URL="2.2.2.2.svg"];
}
```

Nyní zbývá vytvořit zdrojové soubory, ze kterých vznikne překladem zbytek výše zmiňovaných SVG souborů. Tyto zdrojové soubory pro překlad budou mít taktéž název shodný s IP adresou, kterou reprezentují, a koncovku **.txt*. Do každého souboru opět přidáme hlavičku a vypíšeme IP adresy, které právě této IP patří, konkrétně pro soubor *1.1.1.1.txt* pak tedy všechny IP, které přísluší adrese *1.1.1.1*:

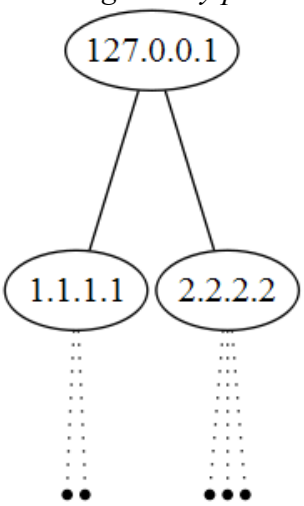
```
digraph G {
rankdir="TB";
node[heigth=.08 width=.08];
"1.1.1.1"->"3.3.3.3"
"1.1.1.1"->"4.4.4.4"
"127.0.0.1"[shape=box URL="127.0.0.1.svg"];
edge[style = dotted];
"127.0.0.1"->"1.1.1.1"
}
```

Vlastnost *rankdir="TB"* říká, že diagram bude tzv. *Top-To-Bottom* neboli ze shora dolů, *shape=box* pro IP *127.0.0.1* určuje, že "obálka" neboli tvar uzlu bude hranatého tvaru, a URL atribut říká, že po kliknutí na tuto IP dojde k přesměrování na soubor *127.0.0.1.svg*, neboli na hlavní stranu.
Nyní máme vytvořeny všechny soubory nutné pro překlad. Zavoláme tedy příkazovým řádkem graphviz a každý tento soubor přeložíme. V našem případě budeme překládat soubory *127.0.0.1.txt*, *1.1.1.1.txt* a *2.2.2.2.txt*. tedy příkazový řádek bude vypadat následovně

```
$dot -Tsvg SourceForSVG/127.0.0.1.txt -o SourceForSVG/127.0.0.1.svg
$dot -Tsvg SourceForSVG/127.0.0.1.txt -o SourceForSVG/127.0.0.1.svg
$dot -Tsvg SourceForSVG/127.0.0.1.txt -o SourceForSVG/127.0.0.1.svg
```

-*Tsvg* určuje výstupní formát SVG, po té následuje cesta ke zdrojovému souboru, -*o* říká, že výstupem bude soubor a poslední parametr přímo určuje cestu a název generovaného souboru. Výsledkem celého programu jsou tedy soubory s příponou txt se zdrojovým kódem v jazyce *dot* pro překlad graphvizem, soubory svg vzniklé přeložením těchto textových souborů a htm soubory s textovým výpisem IP adres.

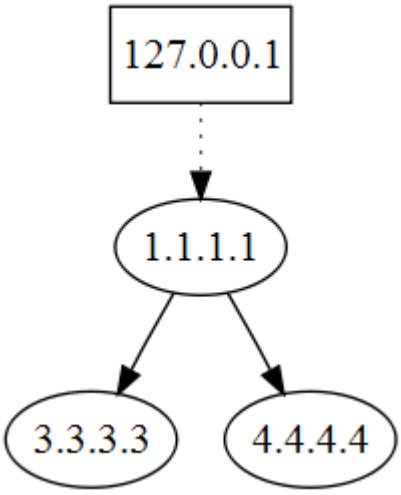
Obr. 4.3: Soubor 127.0.0.1.svg vzniklý přeložením pomocí graphvizu



zdrojovy_soubor.txt

Po kliknutí na uzel *1.1.1.1* dojde k přesměrování na soubor *1.1.1.1.svg*, při kliknutí na IP adresy znázorněné tečkou dojde k přesměrování na *htm* soubor s názvem nadřazené IP adresy, tzn. při kliknutí na tečku spadající pod IP *1.1.1.1* na soubor *1.1.1.1.htm.*

Obr. 4.4.: Soubor 1.1.1.1.svg vzniklý přeložením pomocí graphvizu



Soubor *1.1.1.1.htm* tedy obsahuje textový výpis ip adres *3.3.3.3* a *4.4.4.4*.

Po uplynutí zadaného času dojde k načtení dalšího zdrojového souboru, vymazání adresářů SourceForSVG a celý cyklus se opakuje.

5. Závěr

Výsledkem mého snažení vznikl program kombinující rychlost C# aplikace, vizualizační možnosti graphvizu a univerzálnost SVG výstupu, který se v dnešní době stává standartní součástí všech internetových prohlížečů.

Pro správný běh celého programu je nutné mít nainstalovaný graphviz, který je možno stáhnout z oficiálních stránek www.graphviz.org. Aplikace se spouští ze stejného umístění, kde máme graphviz nainstalovaný, aby ho bylo možno volat příkazovým řádkem. Z tohoto místa také pracuje s podřazenými adresáři SourceFiles a SourceForSVG. Z adresáře SourceFiles čte postupně všechny zdrojové soubory a při každém přeložení vždy jednoho zdrojového souboru uloží všechny výstupní soubory do druhého adresáře, tj. SourceForSVG. Po skončení daného cyklu, resp. při načtení dalšího zdrojového souboru tento adresář SourceForSVG vždy vymaže, aby nedošlo k "záměně" dat vytvořených překladem dalšího zdrojového souboru.

Výstupní zdroje jsou ve formátu *svg* a *htm*, které je možno prohlížet v obyčejném internetovém prohlížeči. *Svg* graficky znázorňuje hierarchický strom IP adres a *htm* soubory jsou textovým vyjádřením IP adres spadající pod jednu skupinu, uzel. Vzhledem k tomu, že *svg* je vektorový formát, může být vykreslování složitějších diagramů náročné na čas. Navíc se jedná o relativně nový prvek *html* standardu, tudíž je zapotřebí použití novější verze internetového prohlížeče.

Seznam literatury

[1] *ITU Telecommunication Standartization Sector : IPTV Focus Group* [online]. 2007 [cit. 2007-11-30]. Dostupný z WWW: <http://www.itu.int/ITU-T/IPTV/>

[2] *An introduction of IPTV* [online]. 2007 [cit. 2007-11-30]. Dostupný z WWW: <http://arstechnica.com/guides/other/iptv.ars>

[3] *IPTV vs. Internet Television : Key differences* [online]. 2005 [cit. 2007-11-30]. Dostupný z WWW: <http://www.masternewmedia.org/2005/06/04/iptv_vs_internet>

[4] *Multicast over TCP/IP : HOWTO* [online]. 1998 [cit. 2007-11-30]. Dostupný z WWW: <http://www.tldp.org/HOWTO/Multicast-HOWTO.html>

[5] *TechTalk : An Analysis of Multicast Methods* [online]. 2003 [cit. 2007-11-30]. Dostupný z WWW: <http://tech-talk.wikidot.com/an-analysis-of-multicast-methods>

[6] *Protocol Independent Multicast (pim)* [online]. 2007 , 2007-09-26 [cit. 2007-11-30]. Dostupný z WWW: <http://ietf.org/html.charters/pim-charter.html>

[7] *Graphviz : Graph Vizualization* [online]. 2004 [cit. 2007-11-30]. Dostupný z WWW: <http://www.graphviz.org/>

[8] TIŠNOVSKÝ, Pavel. *Root.cz : Vektorový grafický formát SVG* [online]. 2004 [cit. 2007-11-30]. Dostupný z WWW: <http://www.root.cz/clanky/vektorovy-graficky-format-svg/>

[9] *Canvas Tutorial* [online]. 2007 [cit. 2007-11-30]. Dostupný z WWW: <http://developer.mozilla.org/cs/docs/Canvas_tutorial>

Seznam příloh

Příloha 1: Část zdrojového kódu programu

```
        //--------------------------------------
        //  Metoda pro nacteni seznamu zdrojaku
        //--------------------------------------
        public void NactiSeznamZdrojaku(ListBox in_Seznam)
        {
           //vymaze seznam zdrojovych souboru
           in_Seznam.Items.Clear();

           //zdrojovy adresar neexistuje --> vytvor ho a vrat prazdny
seznam
           if( !Directory.Exists(SourceFilesPath) )
           {
               //Vytvoreni adresare kde budou zdrojova data
               Directory.CreateDirectory( SourceFilesPath );
           }
           else
           {
               //ziskej adresar info
               DirectoryInfo dirInfo = new DirectoryInfo(SourceFilesPath);

               //ziskej seznam souboru
               FileInfo[] fileInfo = dirInfo.GetFiles("*.txt");

               //zkopiruj seznam souboru do listboxu do formulare
               foreach (FileInfo fi in fileInfo)
               {
                   in_Seznam.Items.Add(fi.Name);
               }
           }
        }
        public void NactiSeznamNovychZdrojaku(ListBox in_Seznam)
        {
            //vymaze seznam zdrojovych souboru
            in_Seznam.Items.Clear();

            //zdrojovy adresar neexistuje --> vytvor ho a vrat prazdny
seznam
            if (!Directory.Exists(SourceForSVGFilesPath))
            {
                //Vytvoreni adresare kde budou zdrojova data
                Directory.CreateDirectory(SourceForSVGFilesPath);
            }
            else
            {
                //ziskej adresar info
                DirectoryInfo dirInfo = new
DirectoryInfo(SourceForSVGFilesPath);

                //ziskej seznam souboru
                FileInfo[] fileInfo = dirInfo.GetFiles("*.txt");

                //zkopiruj seznam souboru do listboxu do formulare
                foreach (FileInfo fi in fileInfo)
                {
                    in_Seznam.Items.Add(fi.Name);
                }
            }
```

Prohlášení

Prohlašuji, že svou bakalářskou práci na téma "Vizualizace spojení mezi počítači" 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)

PODĚKOVÁNÍ

Děkuji vedoucímu bakalářské práce Ing. Danu Komosnému, Ph.D., za velmi užitečnou metodickou pomoc a cenné rady při zpracování bakalářské práce.

V Brně dne ……..... ……....……..

(podpis autora)