VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

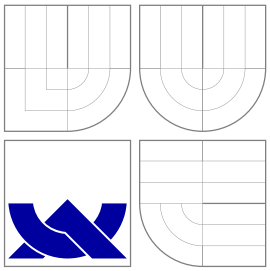
LOGOVÁNÍ PRŮCHOZÍCH DAT V ROUTERECH

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

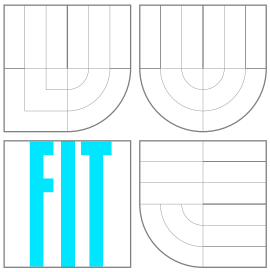
AUTOR PRÁCE
AUTHOR

Bc. PAVEL KISLINGER

BRNO 2007



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

LOGOVÁNÍ PRŮCHOZÍCH DAT V ROUTERECH

THE LOGGING OF TRANSMITTING DATA IN ROUTERS

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. PAVEL KISLINGER

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. DAVID MARTINEK

BRNO 2007

Abstrakt

Cílem této diplomové práce je analyzovat problematiku logování průchozích dat v routerech. Na základě této analýzy je vytvořen návrh systému pro zaznamenávání datových toků v routerech a zároveň vybrána nejvhodnější technologie, která je využita při vlastní implementaci systému v rámci tohoto projektu.

V práci je rozebrána problematika právní odpovědnosti provozovatele routeru za průchozí data. V další části je uveden obecný úvod do problematiky logování dat v počítačové síti, včetně základního popisu protokolů a komunikačních modelů. Následuje analýza konkrétního prostředí, pro které je systém navrhován. Dále je v dokumentu popsán návrh a implementace systému. V poslední části jsou shrnuty dosažené výsledky této práce.

Klíčová slova

log
netflow
router
poskytování internetu
bezpečnost
přenos dat
ochrana osobních údajů
směrování
paket
ISO/OSI
FreeBSD

Abstract

Transmitted data logging in routers is the main point of this semestral project. The suggestion of a system for data flows logging in routers and selection of suitable technology, that is used by implementation of the system within this thesis, is based on this analysis.

In the thesis, a law responsibility of router administrator for transmitting data is analysed. In the next part, a general introduction to issue of data logging in computer networks including basic description of protocols and fundamentals of standard communication models is presented. Analysis of real enviroment is following. Suggestion and implementation of the system is described too. In the last part a reached results of this thesis are revealed.

Keywords

log
netflow
router
internet providing
security
data transmission
personal information protection
routing
packet
ISO/OSI
FreeBSD

Logování průchozích dat v routerech

Prohlášení

Prohlašuji, že tato práce je mým původním autorským dílem, které jsem vypracoval samostatně. Všechny zdroje, prameny a literaturu, které jsem při vypracování používal nebo z nich čerpal, v práci řádně cituji s uvedením úplného odkazu na příslušný zdroj.

. . . . . .
Pavel Kislinger
22. května 2007

Poděkování

Děkuji Ing. Davidu Martinkovi za odborné vedení diplomového projektu a poskytování cenných rad při jejím zpracování, dále děkuji Petru Zahnášovi za výklad právních omezeních v oblasti informačních technologiích a Ing. Petru Hermanovi za poskytnutí informací o komerční síti provozované v prostorách kolejí VUT v Brně, bez nichž by tato práce nemohla vzniknout.

© Pavel Kislinger, 2007.
Tato práce vznikla jako školní dílo na Vysokém učení technickém v Brně, Fakultě informačních technologií. Práce je chráněna autorským zákonem a její užití bez udělení oprávnění autorem je nezákonné, s výjimkou zákonem definovaných případů.

Obsah

Kapitola 1

Úvod

Za několik posledních let došlo k obrovskému rozvoji v oblasti počítačových sítí, ať už v nárůstu rychlosti nebo dostupnosti pro uživatele. Neustále se zvětšující potřeba komunikace, sdílení dat a zdrojů vedla k vytvoření celosvětové sítě Internet. V současné době má prakticky kdokoli možnost se do této sítě připojit. Negativní stránku tohoto rozvoje představují počítačové útoky a viry, které kvůli internetu a rozsáhlým sítím obecně představují velké nebezpečí.

Připojováním k internetu se zabývá mnoho firem – takzvaných providerů. Pro připojení koncového uživatele využívá většina providerů různé technologie od běžných drátových propojů, přes rádiová pojítka až po optickou technologii. Tyto prostředky představují, společně s přenosovými médii a aktivními prvky, síťovou infrastrukturu providera. Tato infrastruktura tvoří hierarchicky uspořádaný stromový graf s jediným vrcholem. Vrchol grafu představuje router, který předává pakety od koncového uživatele do internetu a zpět, uzly grafu představují přepínače. Směrovač tedy představuje ideální místo pro měření datových přenosů do/z internetu.

Cílem této práce je navrhnout systém pro logování datových toků v routerech postavených na platformě FreeBSD. Systém bude umožňovat zpětné dohledání útočníka připojeného z lokální sítě v případech, kdy provozovateli routeru přijde oznámení o trestné činnosti některého z připojených uživatelů. Jedná se především o případy, kdy provozovatel routeru využívá překlad síťových adres (PAT), tudíž v internetu vystupují všichni uživatelé pod jednou adresou. Logovací systém bude univerzální a kromě výše popsané činnosti bude umožňovat měření objemu přenášených dat jednotlivých uživatelů. Navržený systém bude nasazen v prostředí komerční sítě na kolejích VUT. Návrh takového systému je podmíněn znalostí právních omezení s vazbou k dané problematice. Znalosti základů z oblasti počítačových sítí jsou rovněž velmi důležité.

Následující kapitola této diplomové práce se věnuje problematice právních omezení. Rozebírá zranitelná místa navrhovaného systému z právního pohledu a popisuje zákonem dané způsoby, jakými se proti hrozbám bránit. V závěru této kapitoly jsou shrnuty praktické důsledky zmíněných právních omezení, které je nutné akceptovat při návrhu systému.

Třetí kapitola je věnována síťovým technologiím. Popisuje základní komunikační modely se zaměřením na logování paketů. Na základě těchto modelů uvádí, které informace je možné na routerech zaznamenat. Dále ukazuje softwarové a hardwarové technické prostředky, které lze pro zaznamenávání informací z paketů využít.

Popisu prostředí, pro které je systém primárně navrhován, je věnována čtvrtá kapitola. Obsahuje, kromě popisu hardwarových prostředků v síti, také výčet omezení, se kterými je nutné při návrhu systému počítat. Součástí této kapitoly je rovněž popis topologie sítě.

Knihovna *libpcap*, která je využita při implementaci systému, je popsána v další kapitole. Jsou zde zmíněny hlavní principy při práci s touto knihovou a zároveň popsány detaily nejdůležitějších funkcí.

Cílem další části je návrh a implementace vlastního systému. Je zde popsána jeho logická struktura a detailně rozebrány vlastnosti jednotlivých části systému. Zvláštní pozornost je věnována zvolenému paměťovému modelu, který představuje největší přínos, komunikaci mezi vlákny v rámci jedné aplikace a síťové komunikaci mezi částmi systému.

V závěru této práce jsou shrnuty požadované parametry systému. Na základě chování navrženého systému v popsaném prostředí, jsou zjištěny jeho reálné vlastnosti. Porovnáním těchto vlastností s vlastnostmi požadovanými jsou vyvozeny cíle pro další rozšíření stávajícího systému.

Kapitola 2

Právní omezení

Tato část práce je koncipovaná jako obecný úvod do problematiky právních ustanovení týkajících se provozu routeru a logování průchozích dat. Cílem této kapitoly je uvést zákony, které mají přímý dopad na systém pro logování paketů na routerech. Z těchto zákonů je nutné vycházet při návrhu vlastního systému. Tato část práce se snaží podat srozumitelný výklad těchto zákonů. Hlavním zdrojem informací pro sepsání této kapitoly je [1]. Dalšími zdroji jsou [6] a [7].

2.1 Úvod a základní pojmy

Justice představuje v České republice samostatný výkonný orgán, který do jisté míry stojí mimo politické dění. Zákony, které české soudnictví využívá, navrhuje a schvaluje Poslanecká sněmovna Parlamentu České republiky. Vlastní proces nutný pro přijetí zákonu od prvotního podnětu k sepsání zákonu až po dobu, kdy zákon vejde v planost, sestává z několika kroků. Navrhovatel zákona musí zákon sepsat, nechat hlasovat o jeho schválení poslaneckou sněmovnu, následně senát a vše musí potvrdit svým podpisem Prezident České republiky. V případě, kdy je zákon přijat výše zmíněnými instancemi a podepsán prezidentem, uplyne od prvního podnětu do doby než zákon vejde v platnost několik měsíců (nemluvě o složitých zákonech, které senát může vrátit poslanecké sněmovně a jejich schvalování může trvat déle než rok). Tento postup má své výhody a nevýhody.

Výhodou tohoto principu je určitá stálost zákonů (když už se právníci nějaký zákon naučí vykládat, tak ho mohou určitou dobu využívat) a také kvalita výsledných zákonů, tu sice nelze nijak měřit, ale přes takto strukturovaný systém by špatný zákon projít neměl.

Hlavní nevýhodou tohoto systému je doba než dojde zákon v platnost. To se projevuje hlavně u rychle se rozvíjejících odvětví, která vyžadují rychlé vymezení pravidel. Mezi tato odvětví jednoznačně patří oblast informačních technologií, která hlavně s rozvojem internetu vyžaduje jistou právní úpravu. Tento obor se vyvíjí tak rychle, že zákony které jeho funkčnost přesně definují, přicházejí až s mnohaletým zpožděním.

Většina zákonů je psaná obecně, takže nepopisuje konkrétní případy, kdy se daným zákonem řídit. To umožňuje právníkům využít určitý zákon v mnoha situacích. Na druhou stranu to vnáší do oblasti zákonů velkou úroveň abstrakce, která má za následek, že většina lidí, kromě právníků, nedokáže z textu zákona vyčíst jeho význam. Z toho důvodu se objevují takzvané výklady zákona, ty popisují konkrétní případy, ve kterých daný zákon využít a na které se daný zákon naopak nevztahuje.

2.1.1 Zákony v oblasti informačních technologií

Popsat všechny právní problémy, které mohou vyvstat v oblasti IT je téměř nemožné. K vlastnímu uzákonění se proto dostávají pouze závažné problémy, ze kterých pramení spory při podnikání. Oblast podnikání musí být v každé prosperující zemi dobře upravena, jelikož tato oblast přináší do státní kasy ve všech evropských zemích největší sumu peněz.

České IT právo je v současnosti ve fázi rozvoje. Naše dnešní zákony definují počítačovou kriminalitu, chrání osobní údaje, upravují nákup domén, definují spam, popisují e-podpis, určují informační bezpečnost, definují vlastnosti elektronického podnikání apod. U všech těchto problémů je nápadná souvislost s internetem. Všechny tyto problémy jsou sepsány v několika zákonech. Každý zákon obvykle upravuje obvykle hned několik problémů, aby procesem schvalování nemusel procházet balík zákonů (každý zvlášť), ale jen jeden zákon. Schvalování IT zákonů není denním chlebem poslanecké sněmovny. Ke schvalování zákonu z této oblasti se poslanci dostávají jen několikrát za dobu volebního období. Většina zákonů týkající se oboru IT je z roku 2000 a 2004. Není jisté jak se k tomuto problému postaví současná vláda, která hned na začátku svého působení zrušila ministerstvo informatiky (resp. jej sloučila s ministerstvem spravedlnosti).

2.2 Zákony ve vztahu k logování paketů na routerech

Ačkoli to není na první pohled zřejmé, logování paketů na routerech skrývá hned několik právních problémů.

První problém vychází z funkce routeru, který slouží pro připojení uživatele do nějaké nadřazené sítě (obvykle internetu). Jedná se o poskytování služby zákazníkovi. Pokud zákazník spáchá škodu (například tím, že sdílí nelegální data do internetu), může být za tuto škodu podle občanského zákoníku zodpovědný i poskytovatel internetu (provider).

Dalším problémem, který je obsažen přímo v tématu diplomového projektu, je logování. Jedná se zpravidla o chronologický sběr statistických informací o procházejícím datovém toku na routeru, případně jiném veřejném datovém uzlu, jakým může být například webový server. V těchto případech jsou svým způsobem nebezpečná sbíraná data. Z povahy těchto dat vyplývá, že obsahují informace o tom, který uživatel využil danou službu. Svým způsobem se jedná o problém z oblasti ochrany osobních údajů a je nutné navrhovat systém s ohledem na tuto skutečnost.

Posledním problémem je vlastní zabezpečení routeru. Jelikož se jedná o produkční systém, který zajišťuje službu zákazníkovi, je nutné omezit výpadky takového systému na minimum. K tomu musí provozovatel takového systému využít dostupných prostředků a minimalizovat rizika spojená s počítačovým útokem na tento systém. Důležitým krokem při řešení tohoto problému je definice všech rizik spojených s provozem routeru. Z popisu možných rizik se vychází při návrhu vlastního systému a jeho zabezpečení.

Podrobnějším rozborem výše zmíněných problémů včetně návaznosti na konkrétní zákony se zabývají následující kapitoly.

2.3 Poskytování internetu

Problematika odpovědnosti poskytovatelů internetu za obsah přenášených dat byla donedávna velice aktuální. Hlavní problém spočíval v tom, že podle předchozí právní úpravy bylo možné poskytovatele internetu úspěšně žalovat dle principu solidární odpovědnosti. Z toho vycházelo najevo, že škodu nezpůsobil jen sám primární škůdce (např. tím, že někam umístil závadný obsah), ale také

škůdce sekundární (poskytovatel, a to např. tím, že předmětný obsah nezkontroloval), a že tedy dle §439 Občanského zákoníku za ni odpovídají všichni společně a nerozdílně.

Určitým řešením tohoto problému je zákon 480/2004 Sbírky, který definuje odpovědnost, práva a povinnosti osob, které poskytují služby informační společnosti a šíří obchodní sdělení. Paragraf §3 přímo reguluje odpovědnost poskytovatele služby za obsah přenášených informací. Poskytovatel internetu odpovídá za obsah přenášených dat pouze v pokud:

- přenos sám iniciuje
- zvolí uživatele přenášené informace
- zvolí nebo změní obsah přenášené informace

Dále tento zákon v paragrafu §6 říká, že poskytovatelé služeb nejsou povinni dohlížet na obsah přenášených nebo ukládaných informací, zároveň nejsou povinni aktivně vyhledávat skutečnosti a okolnosti poukazující na protiprávní obsah informace.

2.4 Logování

Logování je ze své podstaty automatická činnost. Jedná se o počítačový program, který automaticky zaznamenává události v určité oblasti do takzvaného logovacího souboru případně databáze. Tyto záznamy se obvykle využívají při diagnostice a následném řešení problémů. Dále představují zdroj statistických informací, které lze využít v mnoha odvětvích.

Příkladem takového logovacího systému může být černá skříňka v letadle. Do této skříňky se za letu zaznamenávají všechny informace o povelech pilota v čase a hodnotách jednotlivých senzorů. Údaje z černé skříňky jsou využívány hlavně při haváriích letadel pro přesné zjištění důvodu havárie. Další význam mají černé skříňky při vývoji nových součástí letadel, aby vývojáři mohli zjistit jak se projeví nová součást letadla na jeho celkovém chování.

Logování paketů na routerech slouží primárně k zaznamenávání hlaviček paketů procházejících tímto routerem. Z principu komunikace server – klient plyne, že tyto hlavičky obsahují IP adresu klienta (zákazníka), IP adresu serveru, typ požadované služby a podobné údaje. Podrobný popis komunikace a hlaviček paketů je rozebrán v kapitole 3.3. Klientská IP adresa může mít za určitých podmínek stejnou vyjadřovací schopnost jako jméno a příjmení daného uživatele. První podmínkou je, že v datové cestě mezi uživatelem a routerem není žádný PAT (někdy mylně označován jako NAT), který by způsobil změnu klientské IP adresy ještě před jejím zaznamenáním. Druhou podmínkou je existence nějakého systému, který realizuje vazbu IP adresy ke jménu, příjmení, rodnému číslu nebo jinému údaji, který přesně určuje danou fyzickou osobu. Prakticky se může jednat například o tabulku v databázi. Pokud taková vazba existuje je nutné zabezpečit systém s ohledem na ochranu osobních údajů.

2.4.1 Ochrana osobních údajů

Ochranu osobních údajů upravuje sama ústava a zákon 101/2000 Sbírky. Oba tyto dokumenty obsahují mnoho prakticky nepoužívaných položek. Cílem této kapitoly je vybrat z daných dokumentů pouze části, které mají vztah k výše popsanému problému.

Stěžejním dokumentem v oblasti ochrany osobních údajů je Ústava České republiky. Ta na jedné straně definuje svobodu projevu a volný přístup k informacím, na straně druhé mluví o nedotknutelnosti osoby, právu na ochranu před neoprávněným zasahováním do soukromého a rodinného života a právu na ochranu před neoprávněným shromažďováním, zveřejňováním nebo jiným zneužíváním

údajů o své osobě. Osobní údaje definuje zákon 101/2000 Sbírky jako jakoukoliv informaci týkající se určeného nebo určitelného subjektu údajů. Jedná se zpravidla o jméno, příjemní, datum narození, rodné číslo, adresa apod.

Logování paketů na routeru je tedy z pohledu těchto zákonů někde na pomezí volného přístupu k informacím a ochrany osobních údajů. Správce systému má právo na informace, uživatel právo na ochranu před shromažďováním údajů o své osobě. Tento právní rozpor řeší poskytovatelé internetu obvykle tak, že do smlouvy o službu (o připojení k internetu) zahrnují několik vět o tom, že si vyhrazují právo monitorovat a logovat pakety pro své účely. U takových dat je nutné jasně definovat, kdo k těmto datům bude mít přístup a jak se budou tato data využívat (typickým využitím těchto logů je monitorování zátěže sítě a detekce počítačových útoků).

Řešení výše popsaného problému představuje nová vyhláška k zákonu o elektronických komunikacích, podle které musí poskytovatelé internetu od 1. prosince 2006 uchovávat a na vyžádání policie poskytnout informace o veškeré komunikaci uživatelů (resp. konkrétních IP adres). Záznamy je nutné uchovávat po dobu šesti měsíců, u některých typů komunikace je tato lhůta s ohledem na objem zaznamenávaných dat snížena. Některé typy komunikace, jako například přenos multimediálních dat (internetová rádia či televize), není nutné zaznamenávat vůbec.

2.5 Zabezpečení systému

Pro správce systému je vždy důležité, aby router dobře a spolehlivě vykonával svoji činnost. Pro dosažení takového stavu serveru je nutné jeho zabezpečení. Zabezpečení produkčního HW serveru přímo neupravuje žádný zákon, nicméně je možné se v tomto ohledu inspirovat zákonem 56/1999 Sbírky, který definuje bezpečnost informačního systému jako komplexního celku. Tudíž obsahuje i pasáž věnovanou zabezpečení serveru z hlediska operačního systému a produkčních softwarových aplikací. Podle tohoto zákona musí správce daného systému vynaložit úsilí na to, aby nemohlo dojít k bezpečnostnímu incidentu na tomto serveru. Správce musí prokazatelně:

- udržovat server aktualizovaný
- udržovat pravidla firewallu a firewall samotný ve funkčním stavu
- zajistit, že na serveru běží pouze nezbytné služby
- zajistit, že k serveru mají přístup pouze kvalifikovaní uživatelé v omezeném počtu

Cílovým stavem je dosažení věrohodnosti serveru, aby v případě bezpečnostního incidentu (když nějaký uživatel spáchá trestný čin nebo přestupek) bylo možné využít informace ze serveru pro určení viníka nastalé situace. Prakticky se bezpečnost takového systému nastavuje pomocí firewallu, konfigurace jednotlivých serverových aplikací a promyšlené správy uživatelských účtů.

2.6 Shrnutí právních omezení

Výklad zákonů nemusí být přesný, zákony jsou totiž psány obecně, proto to co z nich šikovný právník „dostane“ nemusí vůbec odpovídat záměru navrhovatele zákona. Nejlepší by bylo, kdyby byla vytvořena soustava zákonů o počítačové komunikaci a internetu. Nevýhodou takového zákona bude vždy to, že nebude časově stálý, budou nutné jeho časté změny. V tomto oboru nelze vytvořit stálá obecná pravidla, protože potřeby současné IT se vyvíjí (kdo slyšel před 10 lety o spamu – dnes je tento problém široce medializován a je ho potřeba řešit zákonně). V poslední době jsou již

přijímané IT zákony na vysoké úrovni. Přesně vymezují pole svojí působnosti s pomocí odborných termínů typu IP adresa, paket nebo NAT a tím se stávají čitelné pro IT odborníky. Současně je ovšem důležité, aby také právní zástupci význam těchto termínů chápali a mohli tyto zákony ve své praxi využít.

Mezi právní problémy, které mají určitou souvislost s logováním paketů na routerech, patří odpovědnost providera za průchozí data a problematika logování dat. Problematiku odpovědnosti providera za průchozí data jeho infrastrukturou řeší zákon 480/2004 Sbírky, který zbavuje providera odpovědnosti, kromě výjimečných případů. Problematiku logování paketů v současné době vymezuje zákon o elektronických komunikacích 485/2005 Sbírky (resp. jeho upravující vyhláška). Ta dává poskytovatelům telekomunikačních služeb povinnost zaznamenávat většinu datové komunikace, která byla přes infrastrukturu poskytovatele provedena, a to po dobu šesti měsíců zpětně.

Záznamy datové komunikace (tzv. logy) mají z určitého hlediska povahu osobních údajů, proto je nutné zaznamenané logy zabezpečit odpovídajícím způsobem. K zabezpečení zaznamenaných údajů se prakticky využívají aktualizace operačního systému, firewall a promyšlená správa uživatelských účtů. Přitom je důležité, aby tento systém mohl v případě bezpečnostního incidentu podat věrohodný důkaz (záznam z logovacího subsystému). Nikde není napsáno co přesně je nutné udělat proto, aby byl systém považován za důvěryhodný (změnit log v souboru není pro administrátora systému nikdy moc těžké). Pokud neexistuje žádná vazba mezi administrátorem systému a výše zmíněným bezpečnostním incidentem, považuje se log na zabezpečeném systému za pravdivý.

Logovací subsystém na routeru je obzvláště důležitý na routerech využívajících NAT, protože uživatelé vystupují v internetu pod jedinou IP adresou, která je navíc shodná s IP adresou routeru, který uživatelům internet poskytuje. Pokud některý z uživatelů spáchá bezpečnostní incident v internetu a tento incident je zaznamenán cílem tohoto incidentu, je v tomto záznamu uvedená právě ona společná IP adresa. Jelikož vlastníkem této IP adresy je sám provider, je velice důležité mít důkaz o tom kdo, kdy a kam přistupoval, k tomu lze úspěšně využít logovací subsystém (logování paketu na routeru).

Kapitola 3

Agregace a logování dat

Tato část práce je koncipovaná jako obecný úvod do problematiky počítačových sítí se zaměřením na logování paketů. Jejím cílem je vysvětlit, které informace je možné z průchozích paketů získat, a představit různé technologie logování paketů. Úvod této kapitoly seznamuje se základními pojmy z oblasti počítačových sítí. Informace použité v této kapitole vycházejí z [2] a [3]. Hlavním zdrojem v oblasti sledování datových toků je [5].

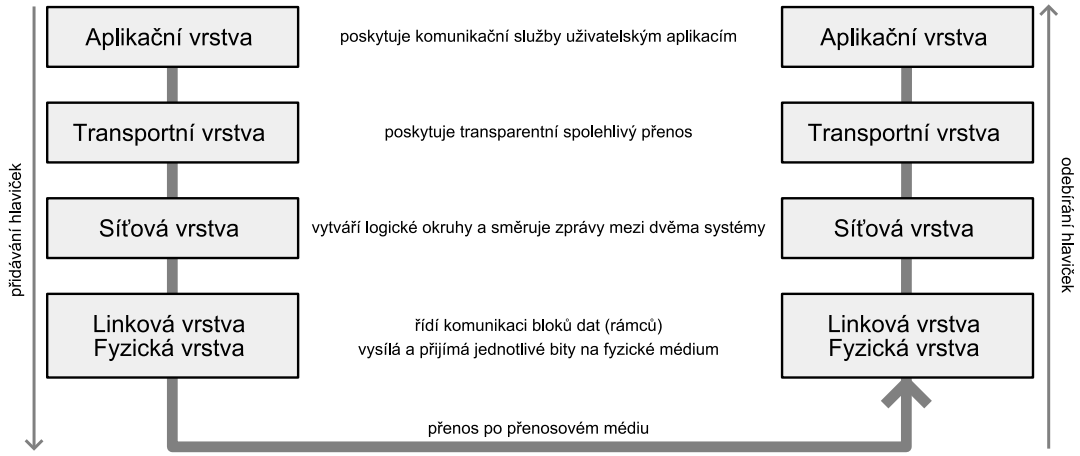
3.1 Úvod a základní pojmy

Počítačová síť je souhrnné označení pro technické prostředky, které realizují spojení a výměnu informací mezi počítači. Umožňují tedy uživatelům komunikaci podle určitých pravidel, za účelem sdílení, využití společných zdrojů nebo výměny zpráv. Historie sítí sahá až do 60. let 20. století, kdy začaly první pokusy s komunikací počítačů. V průběhu vývoje byla vyvinuta celá řada síťových technologií. Dnes jsou všechny sítě postupně spojovány do globální celosvětové sítě Internet využívající sadu protokolů TCP/IP, která se tímto stala nepsaným standardem.

Síťová architektura představuje strukturu řízení komunikace v systémech, tj. souhrn řídících činností umožňujících výměnu dat mezi komunikujícími systémy. Komunikace a její řízení je složitý problém. Proto je nutné tento problém rozdělit na dílčí části, takzvané vrstvy. Členění do vrstev odpovídá hierarchii činností, které je nutné při řízení komunikace vykonat. Každá vrstva sítě je definována službou, která je poskytována nadřazené vrstvě, a funkcemi které vykonává v rámci protokolu. Řízení komunikace slouží ke spolupráci komunikujících prvků sítě. Tato spolupráce musí být koordinována pomocí řídících informací. Koordinaci zajišťují protokoly, které definují formální stránku komunikace. Protokoly jsou tedy tvořeny souhrnem pravidel, formátů a procedur, které definují výměnu informací mezi dvěma či více komunikujícími prvky na určité vrstvě.

3.1.1 ISO/OSI model

Hierarchii vrstev popisuje ISO/OSI model. Tento model se skládá ze sedmi vrstev: aplikační, prezentační, relační, transportní, síťové, linkové a fyzické. Každá z těchto vrstev vykonává skupinu jasně definovaných funkcí potřebných při komunikaci, ke své činnosti využívá služeb nižší vrstvy a své služby poskytuje vyšší vrstvě. Praktického nasazení se tento model pro svoji komplexnost a složitost nikdy nedočkal. V dnešní době se využívá zjednodušený model, který z ISO/OSI modelu vychází (viz obrázek 3.1).



Obrázek 3.1: Zjednodušený model síťové komunikace.

3.1.2 Zjednodušený model

Popisuje, stejně jako ISO/OSI model, hierarchii vrstev. Zjednodušený model se skládá ze čtyř vrstev. Rozdíl oproti ISO/OSI modelu představuje aplikační vrstva, která má význam aplikační, prezentační a relační vrstvy ISO/OSI modelu. Zároveň linková a fyzická vrstva je sloučena do jedné[1]. Ostatní vrstvy si v obou modelech vzájemně odpovídají. Význam jednotlivých vrstev je uveden níže:

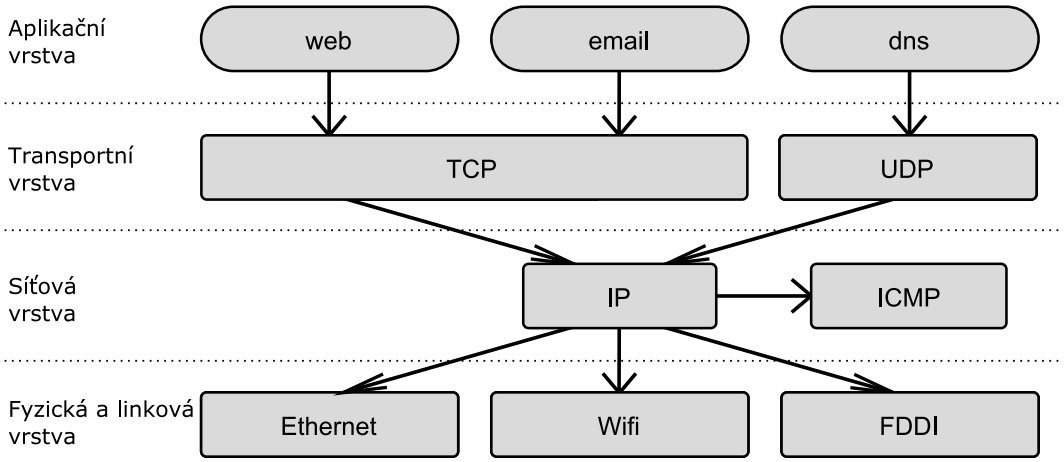
- **Fyzická a linková vrstva** – *Physical and Link Layer*. Jak již jméno této vrstvy napovídá, jedná se v podstatě o dvě vrstvy, které jsou na sobě úzce závislé, a proto jsou ve zjednodušeném modelu uvedeny jako jediná vrstva. Fyzická vrstva definuje všechny elektrické a fyzikální vlastnosti zařízení. Obsahuje rozložení pinů, napěťové úrovně a specifikuje vlastnosti přenosových médií. Linková vrstva poskytuje spojení mezi dvěma sousedními systémy, řadí přenášené rámce do fronty, zabezpečuje nastavení parametrů přenosové linky, oznamuje neopravitelné chyby a opatřuje fyzické rámce fyzickou adresou. Hlavní funkce poskytované fyzickou vrstvou jsou: navazování a ukončování spojení s komunikačním médiem, efektivní rozložení přenosového pásma pro všechny zájemce, modulace a demodulace digitálních dat na signály používané přenosovým médiem. Linková vrstva poskytuje funkce potřebné k přenosu dat mezi jednotlivými síťovými jednotkami a detekuje (případně opravuje) chyby vzniklé na fyzické vrstvě. Na většině lokálních sítích je tato vrstva obvykle rozdělena na dvě podvrstvy: na vrstvu řízení přístupu k médiu (Medium Access Control MAC) a vrstvu pro řízení logických linek (Logical Link Control LLC). Funkce této vrstvy spočívá v uspořádávání dat z fyzické vrstvy do logických celků označovaných jako rámce. Na této vrstvě pracují opakovače, rozbočovače, mosty, přepínače a síťové adaptéry. Mezi protokoly fyzické a linkové vrstvy patří Ethernet, Token Ring, ATM, ISDN nebo FDDI.
- **Síťová vrstva** – *Network Layer*. Tato vrstva se stará o směrování a adresování v síti. Poskytuje spojení mezi systémy, které spolu přímo nesousedí. Síťová vrstva poskytuje funkce k zajištění přenosu dat od zdroje k příjemci přes několik vzájemně propojených sítí při zachování kvality služby. Síťová vrstva poskytuje směrovací funkce a také reportuje o problémech při doručování dat. Na této vrstvě pracují routery, které zabezpečují správné směrování paketů mezi sítěmi.

[1] V některé literatuře se fyzická a linková vrstva uvádějí odděleně.

Síťová vrstva využívá hierarchickou strukturou adres. Nejznámější protokol této vrstvy je IP (Internet Protocol).

- **Transportní vrstva** – *Transport Layer*. Síťová vrstva poskytuje transportní vrstvě služby, které zajišťují přenos paketů mezi libovolnými dvěma uzly sítě. Transportní vrstvu proto zcela odstiňuje od skutečné topologie sítě a vytváří iluzi, že každý uzel sítě má přímé spojení s kterýmkoli jiným uzlem sítě. Transportní vrstva díky tomu může řešit pouze komunikaci koncových účastníků (tzv. end-to-end komunikaci), tedy komunikaci mezi původním odesílatelem a konečným příjemcem. Transportní vrstva zajišťuje sestavování jednotlivých paketů. Při odesílání rozděluje přenášená data do paketů, při příjmu je z paketů vybírá a skládá do původního tvaru. Dokáže tak zajistit přenos libovolně velkých zpráv, přestože jednotlivé pakety mají omezenou velikost. Mezi nejznámější protokoly této vrstvy patří TCP a UDP.
- **Aplikační vrstva** – *Application Layer*. Aplikační vrstva je nejvyšší vrstvou modelu. Jejím účelem je zajistit uživatelským aplikacím síťovou komunikaci a umožnit tak jejich spolupráci. Protokoly aplikační vrstvy představují konkrétní skupinu aplikací, jako například dns, smtp, www nebo ssh.

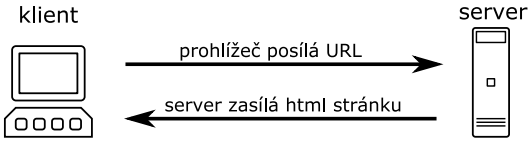
Tento zjednodušený model vrstev je někdy označován jako internetový, protože právě podle tohoto modelu je postaven současný internet. Mezi charakteristické protokoly, které tvoří strukturu internetu, patří IP, TCP a UDP (viz obrázek 3.2). Na celém systému vrstev je důležité si uvědomit, jaké informace jsou ve výsledku přenášeny po síti. Jsou to kromě vlastních dat také řídicí informace jednotlivých vrstev, takzvané hlavičky.



Obrázek 3.2: Komunikační model, na kterém je postavený současný internet. Bloky zobrazené v aplikační vrstvě představují aplikace charakteristické pro internet. Tyto aplikace předávají data transportní vrstvě, která přidává k datům svoje řídící hlavičky. Takto upraveným datům se říká paket. Další hlavičky přidává k paketu síťová vrstva. Fyzická a linková vrstva definují přenosové médium. Linková vrstva přidá k paketu svoji hlavičku (typickou pro dané přenosové médium), čímž vzniká takzvaný rámec. Rámce jsou přenášeny pomocí fyzického média, která je definováno fyzickou vrstvou.

3.1.3 Komunikace klient-server

K tomu, aby bylo možné pochopit princip síťové komunikace, nestačí znát pouze vrstvy. Je nutné pochopit základní komunikační modely. Z nich nejpoužívanější je v sítích TCP/IP model klient-server. V těchto modelech představuje server službu, která je poskytována klientovi. Pokud klient chce službu využít, požádá server o její vykonání, server danou činnost vykoná a vrátí výsledek. Princip modelu klient-server dokresluje následující příklad webového serveru. Server je realizován aplikací, která běží na daném serveru a čeká na požadavky klientů. Jakmile některý z klientů zašle serveru svůj požadavek na konkrétní webovou stránku, server jeho žádost zpracuje, zjistí jestli je možné požadavek zpracovat, v případech, kdy požadavek zpracovat lze, vrací server požadovanou webovou stránku, jinak vrací chybové hlášení (např: Nenalezeno. Požadovaná URL nabyla na tomto serveru nalezena). Princip komunikace klient-server ukazuje obrázek 3.3.



Obrázek 3.3: Příklad komunikace klient server.

3.2 Obecný úvod do logování

Pojem „logování“ lze popsat jako zaznamenávání sekvenčních dat, která bývají obvykle zaznamenávána chronologicky, tedy podle pořadí, v jakém se děj v časové ose odehrál.

Počátky zaznamenávání dat je možné vidět ve středověkých letopisech a kronikách. Úlohou těchto literárních útvarů je chronologicky popsat jednotlivé historické události, což v přeneseném významu představuje konkrétní aplikaci logování dat s tím rozdílem, že při logování bývá záznamovým médiem místo papíru nějaké elektromagnetické médium.

Oblastí, kde se logování dat využívá, je celá řada. Logování nachází uplatnění například v ekonomice, účetnictví, zabezpečovacích systémech, informačních systémech a sítích. Získané informace představují ideální zdroj dat pro statistické zpracování.

Praktickým příkladem nasazení logovacího systému může být obchod, kde logovací systém zaznamenává všechno zboží, které bylo prodáno. Na základě zaznamenaných údajů je možné zjistit, které zboží se prodává dobře a které špatně. Toto rozdělení umožní prodávajícímu přizpůsobit objednávku zboží u dodavatelů. Z toho vyplývá, že logování v podstatě realizuje zpětnou vazbu prodeje zboží (odráží vůli zákazníka), proto jsou získaná data tak důležitá.

3.2.1 Logování s využitím počítačových systémů

Logování dat je v dnešní době realizováno převážně pomocí počítačových systémů a to hned z několika důvodů:

- Zaznamenávanou informací jsou obvykle elektronická data. Čtení takových dat v počítači je proto snadné. Ve výše uvedeném příkladu poskytuje data o prodeji elektronická pokladna.
- Získaná data je nutné někam uložit. Obvyklým úložištěm bývá soubor na pevném disku počítače a to buď přímo nebo zprostředkovaně pomocí databáze. Rozhodnutí zda využít databázi

nebo soubor záleží na objemu ukládaných dat, na způsobu jejich následného zpracování a na výkonu systému.

- Při ukládání získaných dat je někdy žádoucí uložit, kromě samotných dat, také čas, kdy byla data získána. Dnešní počítače obsahují hodiny reálného času, ze kterých operační systém čte čas.

- Počítačové systémy poskytují při nízké pořizovací ceně ve většině případů dostatečný výkon pro logování.

Z výše uvedených důvodů jsou počítačové systémy tou nejvýhodnější technologií pro logování, která v určitých aplikacích poskytuje několikanásobně vyšší výkon, než požaduje logovací subsystém. Proto se logovací systémy v dnešní době objevují i v zařízeních, která možnosti logování nabízí jako svoji nadstandardní funkci.

Příkladem mohou být různá vestavěná zařízení, jakými jsou například přístupové terminály. Hlavní funkcí těchto terminálů je povolit přístup vybraným osobám, zároveň však logují jednotlivé přístupy.

3.2.2 Architektura logovacího systému

Architektura logovacího systému se skládá ze dvou hlavních částí. První část *exporter* data načítá, druhá část *kolektor* data ukládá. Přitom každá z těchto částí může běžet odděleně na různých počítačových systémech. To umožňuje určitou distribuci zátěže, na druhou stranu zatěžuje komunikační kanál. Tato nevýhoda se projevuje zvláště u výkonově náročných aplikací, ve kterých je nutné sbírat mnoho informací v krátkém čase. Vyřešit tento problém je možné buď zvětšením přenosové rychlosti komunikačního kanálu a nebo zmenšením objemu dat přenášených mezi jednotlivými částmi systému. První řešení může být v určitých případech přínosné, bohužel šířku komunikačního kanálu nelze zvětšovat do nekonečna. Z toho důvodu je výhodnější řešit výše uvedený problém redukcí objemu přenášených dat. K tomu se obvykle využívají takzvané agregační funkce. Ty zavádí určitou nepřesnost do měření, protože do databáze už nejsou ukládány všechny zaznamenané informace, ale pouze jejich sumy a průměry. Taková omezení nehrají ze statistického hlediska velkou roli, proto je výhodné tato omezení přijmout a agregační funkce využít. Bližší popis agregační funkce je popsán v kapitole 3.3.2.

3.2.3 Objem zaznamenávaných dat

Problémem při logování je objem zaznamenaných dat. V určitých případech se přírůstek zaznamenaných informací může pohybovat v řádu několika megabytů za minutu. Ovšem otázku objemu ukládaných dat je nutné řešit ve všech logovacích systémech. Při řešení tohoto problémů je nutné vhodně stanovit dobu, po kterou budou data uložena přímo v databázi. Po vypršení této doby jsou informace z databáze přesunuty do záložního souboru, který je z důvodu úspory kapacity pevného disku obvykle komprimovaný. Aby bylo možné z databáze odebírat staré záznamy, je nutné každý záznam opatřit časovým razítkem. Časová razítka sice zvětšují velikost databáze, na druhou stranu umožňují správci systému, kromě mazání starých záznamů, tvořit statistiky s časovou osou. Tyto statistiky umožňují vytvářet časové závislosti pozorovaných veličin. Na příkladu prodeje zboží je možné s pomocí logovacího systému s využitím časových razítek odpověďet na otázky: „Ve kterou hodinu během dne chodí zákazníci nejčastěji nakupovat?“ nebo „Ve kterém ročním období se nejvíce prodává určitý typ zboží?“, a tím umožnit prodejci nabízet požadované služby a zboží ve správný čas.

3.3 Logování paketů

Logování paketů představuje na první pohled stejný problém jako logování čehokoli jiného, má však svá specifika. Zdrojem dat pro logovací systém je v tomto případě síťové rozhraní (například síťová karta). Asi největším rozdílem oproti standardnímu logovacímu systému jsou požadavky na výkon. Při logování paketů na plně vytíženém sto-megabitovém síťovém rozhraní může být teoreticky nutné zpracovat až dvě sta megabitů za sekundu (pro oba směry komunikace). Z toho vyplývá, že výkon systému může hrát klíčovou roli při logování paketů, obzvláště při nasazení gigabitové technologie.

3.3.1 Jaké informace je možné z paketů získat?

V úvodní části kapitoly jsou uvedeny základní informace z oblasti počítačové komunikace. Je zde vysvětlen princip vrstev a právě z vrstvového modelu vyplývá, jaké informace paket obsahuje. V sítích TCP/IP se pod pojmem paket rozumí hlavička[2] TCP protokolu, hlavička IP protokolu a vlastní přenášená data. A právě z hlaviček těchto protokolů lze zjistit mnoho informací (viz obrázek 3.4).

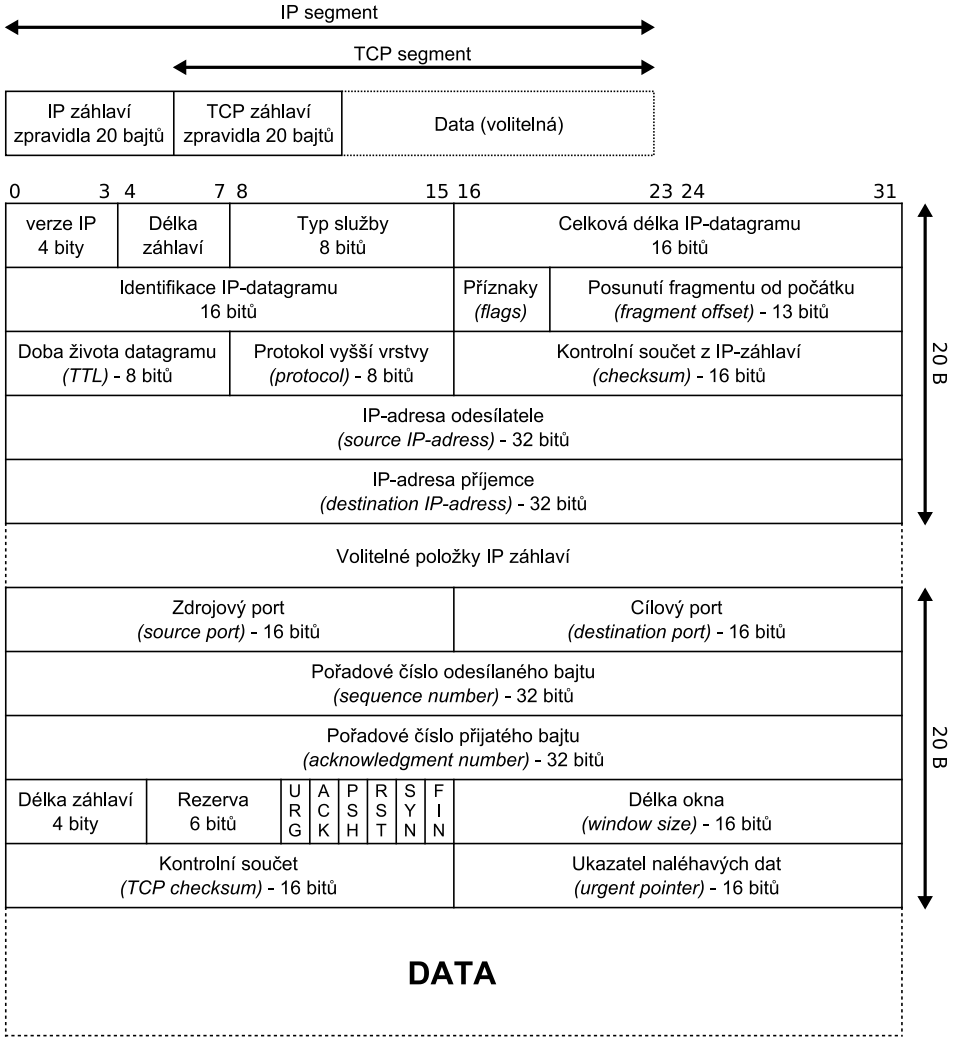
V hlavičkách TCP a IP protokolu se nachází mnoho informací, mezi nejdůležitější patří IP adresa cíle a odesílatele. Tyto adresy jednoznačně definují oba komunikující body v síti, klienta i server. Další důležitou informací, která se dá vyčíst z hlaviček, je zdrojový a cílový port. Port jednoznačně určuje komunikující aplikaci. Zdrojový port určuje aplikaci na počítači s IP adresou odesílatele. Cílový port určuje aplikaci na počítači s IP adresou cíle. Z toho vyplývá, že tyto čtyři položky přesně definují aplikace na konkrétních strojích, které spolu komunikují. Logování těchto položek tedy umožňuje zpětně dohledat veškerou komunikaci, kterou sledovaný systém vykonal.

3.3.2 Ukázka aplikace agregační funkce

Aby bylo možné ukázat jakým způsobem mohou být pakety agregovány, je nutné získat nějaká vzorová data. V tomto příkladu byla vzorová data získána pomocí programu *Ethereal* [10] a představují přenos textového souboru o velikost 6 kB z webového serveru do okna prohlížeče na klientském počítači. Níže uvedená data reprezentují pakety, které jsou přeneseny po síti mezi serverem a klientem při přenosu souboru.

	čas	zdroj	cíl	protokol
1	2.891808	147.229.202.76	147.229.191.135	DNS
2	2.893813	147.229.191.135	147.229.202.76	DNS
3	2.894876	147.229.202.76	147.229.203.200	TCP
4	2.895285	147.229.203.200	147.229.202.76	TCP
5	2.895314	147.229.202.76	147.229.203.200	TCP
6	2.898058	147.229.202.76	147.229.203.200	TCP
7	2.898550	147.229.203.200	147.229.202.76	TCP
8	2.895602	147.229.203.200	147.229.202.76	TCP
9	2.898630	147.229.202.76	147.229.203.200	TCP
10	2.895667	147.229.202.76	147.229.203.200	TCP
11	2.898980	147.229.203.200	147.229.202.76	TCP
12	2.899016	147.229.203.200	147.229.202.76	TCP
13	2.899033	147.229.203.200	147.229.202.76	TCP
14	2.899060	147.229.202.76	147.229.203.200	TCP
15	2.899269	147.229.202.76	147.229.203.200	TCP

[2] Z anglického „*Header*“. Bývá překládáno buď jako hlavička, nebo jako záhlaví.



Obrázek 3.4: Rozložení hlaviček v paketu a popis hlaviček TCP a IP.

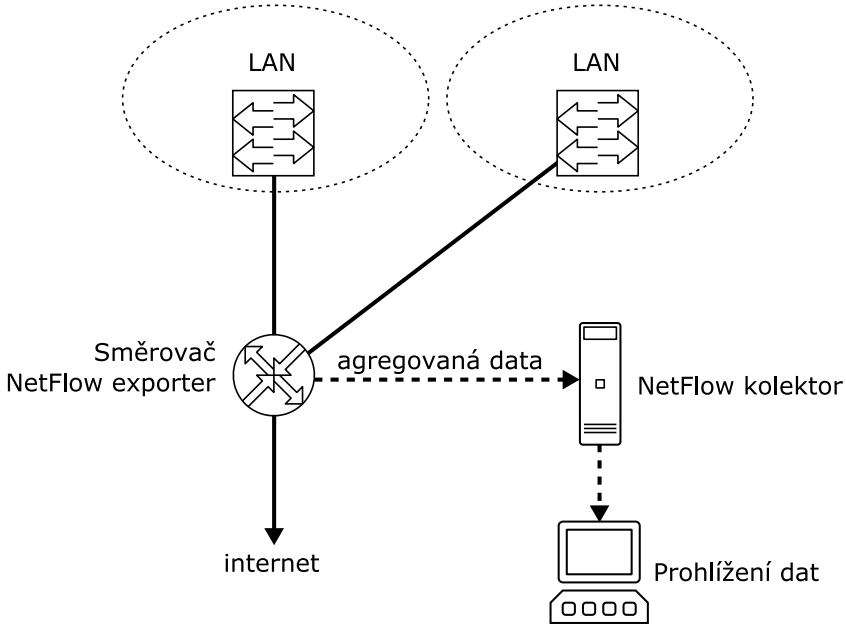
První dva pakety představují překlad DNS jména počítače na IP adresu. Další pakety realizují přenos souboru. Při logování je nutné všechny tyto hlavičky paketů přenést do úložiště logovacího systému, tudíž každou IP adresu je potřeba přenést několikrát. Agregace spočívá ve shromažďování informací o paketech na základě shodné zdrojové a cílové adresy. Výsledek aplikací agregace na výše uvedené hlavičky paketů je uveden níže. Úspora při přenosu dat v rámci logovacího systému je zřejmá.

```
od         do         zdroj            cíl              protokol   počet
2.891808   2.891808   147.229.202.76   147.229.191.135  DNS        1
2.893913   2.893813   147.229.191.135  147.229.202.76   DNS        1
2.894876   2.899263   147.229.202.76   147.229.203.200  TCP        7
2.895285   2.899033   147.229.203.200  147.229.202.76   TCP        6
```

V tomto vzorovém případě tedy byly v původních 15 paketech rozpoznány 4 datové toky. Přitom nejvyšší míru agregace vykázal třetí rozpoznaný datový tok (uvedený na třetím řádku v druhé tabulce), který agregoval 7 paketů do jednoho datového toku. Příklad neodráží reálnou úsporu, ke které dochází při běžném provozu, kde je možné agregovat v jednom datovém toku tisíce paketů, snaží se pouze o hrubý nástin principu.

3.4 Cisco NetFlow protokol

Firma Cisco je jedním z průkopníků technologie NetFlow. Cisco NetFlow protokol byl původně navržen jako otevřený protokol pro monitorování zdroje a cíle datových přenosů na síťové vrstvě ISO/OSI modelu. Hlavní význam tohoto protokolu spočívá v předávání informací mezi exporterem a kolektorem. Přestože se tento protokol hojně využívá v produktech různých výrobců, je i dnes považován za proprietární technologii firmy Cisco. Základní princip NetFlow protokolu ukazuje obrázek 3.5.



Obrázek 3.5: Základní architektura NetFlow. Směrovač, který kromě své hlavní činnosti – routování, rovněž zajišťuje funkci NetFlow exporteru. Zaznamenává datový provoz do své paměti a zároveň na sebraná data aplikuje agregační funkci. Agregovaná data jsou v určitých časových intervalech odeslána pomocí UDP paketu do NetFlow kolektoru. Jeho úlohou je přijatá data zpracovat a zároveň poskytnout správci sítě požadované informace.

Komunikaci mezi exporterem a kolektorem popisuje NetFlow protokol. Přesný formát této komunikace závisí na verzi NetFlow protokolu. V současné době existují čtyři verze tohoto protokolu: verze 1, verze 5, verze 7 a verze 8. Jednotlivé verze přenáší důležité charakteristiky datového toku (zdrojová a cílová IP adresa, porty, protokol IP vrstvy, časy od a do datového toku) a zároveň mnoho rozšiřujících informací, specifických pro jednotlivé verze.

Ve verzích 1, 5 a 7 obsahuje UDP datagram hlavičku a jeden nebo více záznamů o datových tocích. První pole v hlavičce udává číslo verze použitého protokolu. Druhé pole udává počet záznamů datových toků, které datagram obsahuje. Verze 8 navíc udává množinu rozšířených agregačních funkcí, které se využívají převážně na routerech, čímž je možné dále snižovat objem dat přenášených po síti. Nejpoužívanější je v současné době NetFlow protokol ve verzi 5 (zkráceně NetFlow v5). Jeho popisu je věnována následující kapitola.

Bytes	Contents	Description
2	version	NetFlow export format version number
2	count	Number of flows exported in this packet (1-30)
4	SysUptime	Current time in milliseconds since the export device booted
4	unix_secs	Current count of seconds since 0000 UTC 1970
4	unix_nsecs	Residual nanoseconds since 0000 UTC 1970
4	flow_sequence	Sequence counter of total flows seen
1	engine_type	Type of flow-switching engine
1	engine_id	Slot number of the flow-switching engine
2	reserved	Unused (zero) bytes

Tabulka 3.1: Formát hlavičky NetFlow v5

3.4.1 Formát NetFlow v5

Informace o datových tocích jsou mezi exporterem a kolektorem přenášeny pomocí UDP datagramů. Každý datagram se skládá z hlavičky a záznamů, které obsahují informace o jednotlivých datových tocích. Počet vložených datových toků je dán položkou v hlavičce. Hlavičku popisuje tabulka 3.1. Záznam je popsán v tabulce 3.2. Z tabulek vyplývá, že hlavička je dlouhá 24 bytů a záznam 48 bytů. Jeden UDP datagram se obvykle skládá z 24 bytů hlavičky, za kterou následuje několik 48bytových záznamů.

3.5 Logování paketů na routerech

Router neboli směrovač je síťové zařízení, které procesem zvaným směrování přeposílá pakety směrem k jejich cíli. Základům směrování se věnuje kapitola 3.5.2. Hlavní úlohou routeru je spojování sítí, tj. zajištění komunikace počítače z jedné sítě s počítačem ze sítě druhé. Současný internet v podstatě tedy stojí na routerech (síť sítí). Podle hierarchického umístění routeru v internetu jsou rozlišovány dva typy routerů: okrajový router (*edge router*) – router, který připojuje klienty k nadřazené síti (internetu) a vnitřní router (*core router*) – router přenášející data mezi jinými routery. Požadavky na tyto dva typy routerů jsou různé. Okrajové routery potřebují především dostatek síťových rozhraní. Vnitřní routery vyžadují velkou propustnost. Co však oba typy routerů spojuje je architektura.

3.5.1 Architektura routeru

Každý router se skládá z několika síťových rozhraní a vnitřní logiky. Tato logika rozhoduje o tom, na které rozhraní přepošle router přijatý paket. Obecně se routery dělí do dvou skupin. První skupinu představují routery postavené na běžných počítačích. Druhou skupinou jsou specializovaná zařízení navržená výhradně pro účely routování.

Routery postavené na běžných počítačích využívají pro routování softwarové implementace routovacích algoritmů. Tyto implementace přistupují k jednotlivým síťovým rozhraním počítače pomocí služeb operačního systému (existují implementace pro operační systémy Unix i Windows). Z principu routování je zřejmé, že takový software musí běžet na počítači neustále, jako démon. Nevýhodou těchto systémů je omezený počet fyzických síťových rozhraní a pro jisté aplikace nízký výpočetní výkon. Přitom výkonová omezení mohou být způsobena jednak nedostatečným výkonem

Bytes	Contents	Description
4	srcaddr	Source IP address
4	dstaddr	Destination IP address
4	nexthop	IP address of next hop router
2	input	SNMP index of input interface
2	output	SNMP index of output interface
4	dPkts	Packets in the flow
4	dOctets	Total number of Layer 3 bytes in the packets of the flow
4	First	SysUptime at start of flow
4	Last	SysUptime at the time the last packet of the flow was received
2	srcport	TCP/UDP source port number or equivalent
2	dstport	TCP/UDP destination port number or equivalent
1	pad1	Unused (zero) bytes
1	tcp_flags	Cumulative OR of TCP flags
1	prot	IP protocol type (for example, TCP = 6; UDP = 17)
1	tos	IP type of service (ToS)
2	src_as	Autonomous system number of the source, either origin or peer
2	dst_as	Autonomous system number of the destination, either origin or peer
1	src_mask	Source address prefix mask bits
1	dst_mask	Destination address prefix mask bits
2	pad2	Unused (zero) bytes

Tabulka 3.2: Formát záznamu jednoho datového toku NetFlow v5

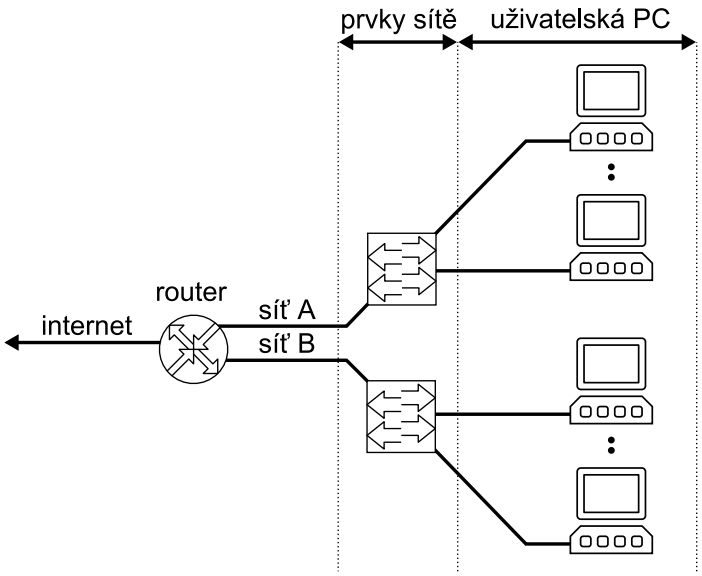
procesoru, pomalou pamětí, nebo omezenou propustností sběrnic. Výhodou systémů postavených na běžných počítačích je flexibilita systému (na daném systému může běžet mnoho dalších aplikací) a především cena.

Oproti tomu routery postavené na specializovaném hardwaru využívají pro routování speciální obvody. Tyto obvody poskytují dostatečný výkon, umožňují obsloužit několik desítek portů a pracují i na rychlostech vyšších než 1Gb za sekundu. Dnešní nejvýkonnější routery nabízejí i funkce NetFlow exporteru. Zapnutí této funkce sice přináší velkou zátěž samotného routeru, ovšem za cenu kompletního dohlednu nad lokálními sítěmi. Nevýhodou tohoto řešení je malá flexibilita systému, zajištěná pouze na úrovni upgradu firmware.

Další část práce se zaměřuje pouze na okrajové routery postavené na běžných počítačích (viz obrázek 3.6) a to hlavně kvůli flexibilitě. Osobní počítač, který realizuje funkci routeru, totiž může zajišťovat nejen funkci routeru a paketového filtru, ale zároveň může zajišťovat funkci NetFlow exporteru a dalších serverových služeb.

3.5.2 Základy směrování

Základní službou, kterou musí zajišťovat každý router, je směrování (routování). Pod pojmem směrování se v podstatě rozumí proces hledání cest v počítačových sítích. Jeho úkolem je vyslat datový paket směrem k jeho adresátovi co nejefektivnější cestou. Tuto cestu tvoří kromě odesílatele a adresáta soustava routerů, každý z těchto routerů rozhoduje o tom, kterým směrem bude paket vyslán dál tak, aby se co nejvíce přiblížil ke svému cíli. Směrování probíhá na síťové vrstvě ISO/OSI modelu a využívá protokoly této vrstvy (nejčastěji IP protokol).



Obrázek 3.6: Připojení dvou sítí do internetu. Na obrázku je znázorněno schéma zapojení sítě A a sítě B do internetu pomocí routeru. Prvky sítě představují přepínače, které tvoří hierarchicky uspořádanou topologii sítě. Tato síť slouží k připojení uživatelských počítačů v pravé části obrázku do internetu. Z obrázku vyplývá, že router (respektive jeho vnější síťové rozhraní) představuje ideální místo pro sledování datového provozu z lokálních sítí do internetu.

Statické a dynamické směrování

Základní rozdělení směrovacích algoritmů je na statické a dynamické. Statické směrování využívá statickou routovací tabulku, která se sestavuje na základě konfigurace síťových rozhraní daného routeru. Dynamické směrování vychází ze statického směrování, rovněž využívá routovací tabulku, ale oproti statickému směrování přizpůsobuje routovací tabulku aktuální topologii sítě. Aby bylo možné rozpoznávat určité stavy v topologii sítě (například přerušenou linku), musí mezi sebou routery komunikovat. K jejich vzájemné komunikaci slouží například protokoly OSPF nebo RIP.

Směrování na okrajovém routeru

Tyto routery mají jen jedno připojení do internetu, které v případě výpadku není možné nahradit žádnou záložní linkou, tudíž by dynamické směrování nemělo smysl. Proto se u okrajových routerů obvykle využívá statické směrování.

Ve většině dnešních operačních systémů, využívajících sadu protokolů TCP/IP, je statická routovací tabulka přítomna automaticky. Jejím úkolem je zajistit směrování odchozích paketů na výchozí bránu (Default Gateway) a také zaručit korektní směrování v případech, kdy má hostitelský systém více síťových rozhraní. Tato routovací tabulka může rovněž sloužit pro směrování paketů z lokální sítě přes hostitelský systém do nadřazené sítě (internetu). První podmínkou takového chování hostitelského systému je existence síťového rozhraní jak v lokální, tak v nadřazené síti. Druhou podmínkou je schopnost jádra předávat pakety z jednoho síťového rozhraní na druhé. Tato schopnost se v různých operačních systémech nastavuje různě. Ve většině *UNIXových* operačních systémů (FreeBSD, Linux) se obvykle využívá příkazu *sysctl*, jehož pomocí je možné nastavit systémovou proměnnou `net.inet.ip.forwarding` na hodnotu „1“. Tím se dosáhne požadovaného chování jádra operačního systému.

Pokud by mělo být směrování jedinou technologií využívanou pro připojení počítačů z lokální sítě do internetu, museli by mít všechny počítače z lokální sítě veřejnou IP adresu. Těch je v dnešní době citelný nedostatek, proto je výhodné využít na routeru, z důvodu úspory veřejných IP adres, technologii překladu síťových adres. Té je věnována následující kapitola.

3.5.3 Překlad síťových adres

V dnešní době existuje několik služeb, které lze zařadit mezi technologie pro překlad síťových adres: transparentní proxy, *NAT* (Network Adress Translation) nebo *PAT* (Port Adress Transaltion).

- Transparentní proxy přesměrovává pakety, které přichází na určitý port hostitelského systému, na vzdálenou IP adresu a port. Označení „transparentní“ vychází z její funkce, kdy uživatelé využívají službu na určitém portu hostitelského systému a přitom nevědí, že tato služba fyzicky běží na nějakém vzdáleném počítači. Pro účely připojení uživatelů do internetu se tato služba moc dobře využít nedá.
- *NAT* zajišťuje překlad lokálních adres na adresy veřejné, čímž v podstatě přiděluje počítačům z lokální sítě další IP adresu. Tímto způsobem nedochází ke změně čísla portu, ale také nedochází k úspoře veřejných IP adres, protože pro současnou komunikaci *n* počítačů z lokální sítě je potřeba *n* veřejných IP adres. Pro překlad adres se využívá jednoduchá tabulka, která obsahuje IP adresy počítačů z lokální sítě a přeložené (veřejné) IP adresy.
- *PAT* zajišťuje překlad adres počítačů z lokální sítě na jednu veřejnou adresu a port. Často bývá mylně označován jako *NAT*. Ke svojí činnosti využívá překladovou tabulku, která obsahuje původní port, IP adresu počítače z lokální sítě, vzdálený port a vzdálenou IP adresu. Výhodou této technologie je úspora veřejných IP adres, protože pro připojení několika počítačů do internetu stačí jedna veřejná IP adresa. Další výhodou je ochrana lokální sítě před útoky z internetu.

Vztah mezi technologiemi pro překlad adres a logováním paketů není zcela zřejmý. Vysvětlení tohoto vztahu podává následující text.

Vztah logování a překladu adres

V kapitole 3.5.1 bylo vysvětleno, proč okrajové routery představují ideální místo pro logování paketů (viz obrázek 3.6). Přesněji řečeno, ideálním místem pro nasazení NetFlow exporteru je to síťové rozhraní routeru, které zajišťuje spojení s nadřazenou sítí (internetem). Přes toto rozhraní musí projít veškerá komunikace mezi lokálním počítačem a jeho protějškem v internetu, a proto je možné na něm jednoduše číst z hlaviček paketů informace o datových tocích. Pokud ale router realizuje překlad síťových adres, jsou hlavičky paketů na vnějším rozhraní routeru modifikované následujícím způsobem:

- V případě transparentního proxy serveru se pakety, adresované na určitý port síťového rozhraní routeru změní na IP adresu a port cílového počítače podle konfigurace proxy serveru. Takové chování je z pohledu logovacího systému v pořádku, neboť na vnějším rozhraní routeru figuruje v hlavičce přeloženého paketu, na pozicích zdrojové adresy a portu, IP adresa a port lokálního počítače, který komunikaci vyvolal.
- V případě překladu technologií *NAT* se na vnějším rozhraní routeru již neobjevují lokální IP adresy počítačů, ale pouze adresy přeložené podle překladové tabulky, která vytváří vztah 1:1

k lokálním adresám. Tím vytváří určitou bariéru snadnému logování paketů z vnějšího rozhraní, neboť je třeba přeložené IP adresy zpětně přeložit na lokální a to především v případech, kdy se překladová tabulka mění dynamicky za běhu systému.

- Při překladu pomocí technologie *PAT* se na vnějším rozhraní routeru objevují pakety, které obsahují, místo IP adresy lokálního počítače a čísla portu, vnější IP adresu routeru a změněné číslo portu. Číslo portu se vypočítává na základě překladové tabulky, která je složitější než v případě technologie *NAT* (viz předchozí kapitola). Proto logovací systém na vnějším rozhraní routeru, využívajícího technologii PAT, postrádá svůj význam.

Popsaným problémům se dá vyhnout několika způsoby. Řešení, které přichází v úvahu jako první, je aktivní zpětný překlad IP adresy dle překladové tabulky na lokální adresu a v případě *PAT* také port. Hlavní nevýhodou tohoto řešení je složitost přístupu k překladové tabulce a případné vyhledávání v ní. Dalším řešením může být přesunutí logovacího systému na vnitřní rozhraní lokální sítě, kde se ještě nachází hlavičky paketů v nezměněné podobě. Protože má router obvykle více vnitřních síťových rozhraní, je nutné logovací systém provozovat na všech těchto rozhraních, z čehož vyplývají větší paměťová nároky na hostitelský systém.

3.6 Technologie logování paketů

Logování paketů nepředstavuje z technického pohledu složitou činnost. V podstatě se jedná pouze o čtení určitých položek ze síťového rozhraní, zpracování a jejich následné uložení do databáze. Jediné úskalí představuje čtení dat ze síťového rozhraní, neboť většina operačních systémů neumožňuje přímý přístup k hardwaru. Řešení tohoto problému spočívá ve využití příslušné knihovní funkce operačního systému, přitom obvykle platí, že tyto funkce může využívat pouze administrátor operačního systému. Existuje také specializovaný hardware, který nabízí mnohem vyšší výkon oproti systémům postaveným na bázi PC. V následujícím textu je uvedeno několik systémů, které lze považovat za významné zdroje inspirace pro následující práci.

3.6.1 Program tcpdump

Výše popsaný způsob čtení paketů, je velice snadný. Jeho praktickou implementaci představuje program *tcpdump* a knihovna *libpcap* [8]. Má ovšem svá omezení. Nejužším místem je sběrnice mezi síťovou kartou a základní deskou (resp. její částí zvanou Southbridge), protože každý načítaný paket musí být přes tuto sběrnici přenesen. Sběrnice PCI umožňuje spolehlivé čtení dat ze síťového rozhraní do rychlosti 100MBit. Při vyšších rychlostech síťové karty (1Gbit) začne docházet ke ztrátám paketů (*packet loss*). Z praktického hlediska nemá výše uvedené omezení velký význam, neboť při vhodném návrhu logovacího systému, jsou zaznamenávány pouze pakety, které směřují z/do internetu. Navíc šířka pásma přenosové linky, která realizuje propojení mezi routerem a internetem, se ve většině případů pohybuje v v řádu jednotek megabitů za sekundu. Šířka této přenosové linky je přímo úměrná maximálnímu počtu přenesených paketů touto linkou. Z toho vyplývá, že omezená šířka pásma značně snižuje pravděpodobnost zahlcení sběrnice PCI.

3.6.2 Program fprobe

Na knihovně *libpcap* staví celá řada mnohem komplexnějších nástrojů než je program *tcpdump*. Jedním z programů, který slouží pro měření datových toků, je *fprobe* [9].

Jedná se o NetFlow exporter, který naslouchá na síťovém rozhraní, agreguje informace z hlaviček přenášených paketů a posílá NetFlow v5 datagramy do vzdáleného kolektoru na zpracování. Datový tok identifikuje pomocí zdrojové a cílové IP adresy, zdrojového a cílového portu a protokolu IP vrstvy. V současné době podporuje pouze ethernetová rozhraní a je dimenzován na 100MBit datové linky, na kterém pracuje spolehlivě. Při větší zátěži má tendenci zabrat veškeré přidělené systémové prostředky (procesorový čas, paměť). Při vyčerpání dostupných prostředků začne program zahazovat velké množství paketů (až 90 procent), takže se stává zcela nepoužitelným.

Program nabízí velké množství funkcí, některé velmi promyšlené. Velmi zajímavou je myšlenka active a idle timeru (aktivního a neaktivního časovače). Tyto časovače udávají čas, po jehož vypršení dojde k exportu agregovaného toku. Aktivní časovač je závislý na přijetí prvního paketu v datovém toku. Neaktivní časovač je závislý na přijetí posledního paketu v datovém toku. Pokud dojde k vypršení času jednoho z časovačů, dochází k exportu. Program *fprobe* tuto funkčnost zajišťuje průchodem všech aktivních datových toků v pevně daných intervalech. Z toho plyne výše zmíněný problém s vyčerpáváním dostupných systémových prostředků při velkém počtu aktivních datových toků.

Další zajímavou funkcí je možnost akceptovat pouze určité datové toky. K tomu slouží filtr poskytovaný knihovnou *libpcap*. Pravidla tohoto filtru jsou velmi dobře navržená, umožňují filtrování na základě adres, portů, typu komunikace a zároveň nabízí několik operátorů pro jejich snadnou aplikaci.

3.6.3 Program flowd

Jedná se rychlý NetFlow kolektor, který umí zpracovávat datagramy z jednotlivých NetFlow exporterů. Získaná data ukládá v binárním formátu do souborů po jednotlivých dnech. Nevýhodou tohoto systému je, že na získaná data aplikuje příliš silnou agregační funkci, která sdružuje datové toky za dobu celého dne. Zajímavou vlastností tohoto programu je, že umožňuje volat uživatelsky definovanou funkci, napsanou v jazyce Perl nebo Python, na každý přijatý paket. Tato struktura umožňuje definovat téměř libovolné chování, ovšem za cenu velkého zatížení procesoru při zpracování interpretovaného jazyka.

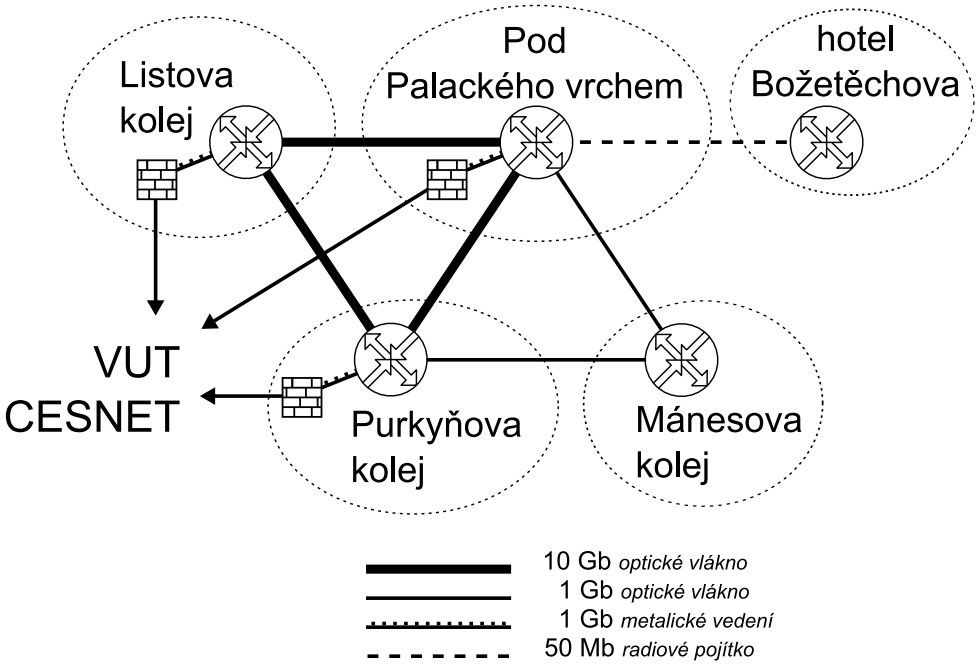
3.6.4 HW implementace

Případů, kdy je potřeba zaznamenávat větší objemy dat, není mnoho, nicméně existují (např zaznamenávání paketů na páteřních linkách nadnárodních providerů). V těchto případech nelze model s využitím síťové karty počítačového systému využít. Proto přichází na řadu specializovaný hardware. Příkladem takového hardware může být *NetFlow* sonda [4], realizovaná jako karta do několika typů sběrnic (PCI, PCI-Express). Hlavní myšlenkou této sondy je zpracovat velký objem dat přímo hardwarem na kartě, do počítače už jsou pak přenášena pouze agregovaná data v binární formě tak, aby malá rychlost sběrnice nebyla významná. Toto řešení nachází uplatnění především na linkách o přenosových rychlostech větších než 1Gb za sekundu, protože při takových rychlostech již není možné využít jinou technologii.

Kapitola 4

Popis prostředí

Studentská kolejní síť VUT (KolejNet) je gigabitová metropolitní síť čítající několik tisíc klientských počítačů, mnoho serverů a vyspělou síťovou infrastrukturu. V současnosti se skládá z pěti areálů v lokalitě Brno – Královo pole. Jedná se o areál koleje Pod Palackého vrchem, Purkyňovy koleje, Listovy koleje, Mánesovy koleje a o hotel Božetěchova. Propojení těchto kolejí ukazuje následující schéma.



Obrázek 4.1: Schématické propojení jednotlivých areálů kolejí VUT.

V každém areálu kolejí se nachází nezávislý směrovač, který odděluje provoz v rámci areálu od ostatního provozu. Jednotlivé areály kolejí se obvykle skládají z několika výškových budov. Pro každou budovu je vytvořena jedna síť, která je pomocí optického vlákna propojena s areálovým směrovačem. Topologie sítě v rámci budovy je obvykle postavena na gigabitovém přepínači, do kterého jsou připojeny koncové sto-megabitové přepínače. Uživatelé jsou připojování do těchto koncových přepínačů.

4.1 Uživatelé sítě

Jelikož je síť KolejNet připojena k internetu prostřednictvím sítě Cesnet, je nutné dodržovat podmínky dané provozovatelem této nadřazené sítě. Podmínky sítě Cesnet dovolují připojit pouze takového uživatele, který se zabývá vědou, výzkumem, vývojem nebo šířením vzdělanosti a kultury.

Většinu uživatelů sítě KolejNet tvoří studenti VUT v Brně. Jsou zde ovšem i studenti jiných vysokých škol nebo jejich zaměstnanci. Všichni tito uživatelé mohou být z podstaty vysoké školy jako instituce zabývající se vědou, výzkumem a vývojem, připojeni do sítě Cesnet.

Postupem času se v jednotlivých areálech kolejí, objevilo několik subjektů, které projevily zájem o připojení, ale výše uvedená pravidla jejich připojení nedovolovala. Šlo o provoz hotelu a několik komerčních firem. Na základě této skutečnosti byla vybudována druhá síť v rámci areálů kolejí, která je do internetu připojena prostřednictvím komerčního providera, tudíž umožňuje připojení komerčních subjektů.

4.2 Komerční síť v prostorách kolejí

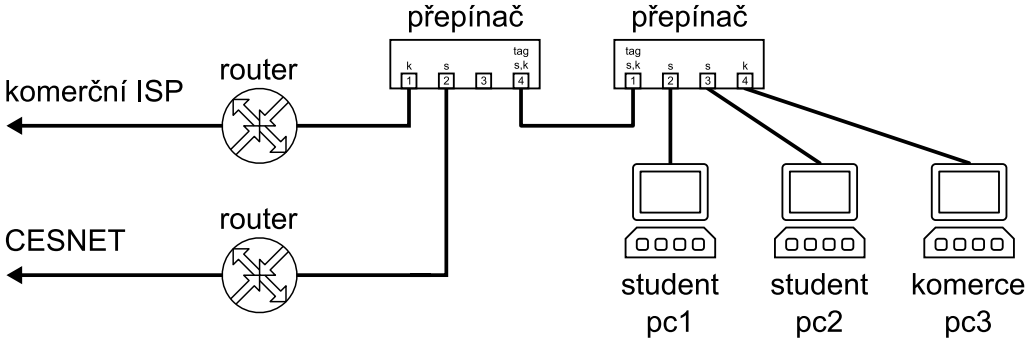
Komerční síť v prostorách kolejí vznikla teprve nedávno (v roce 2005). Tato síť využívá stejnou síťovou infrastrukturu jako studentská síť KolejNet (tj. stejné aktivní prvky, stejnou kabeláž). Obě sítě se liší především tím, že pakety směrované do internetu z komerční sítě jsou předávány do sítě komerčního providera, zatímco pakety ze studentské sítě jsou předávány do sítě Cesnetu.

Jelikož jsou obě sítě provozovány na stejné síťové infrastruktuře, je nutné provoz jednotlivých sítí v rámci této infrastruktury rozlišit. Pro rozlišení se využívá technologie virtuálních sítí (*VLAN*). Tato technologie umožňuje virtuálně rozdělit přepínač jakoby na dva přepínače, jeden přepíná pakety ve studentské síti, druhý v komerční. Zároveň umožňuje přenášet pakety z obou sítí přes jedno síťové rozhraní pomocí tagů příslušného síťového rozhraní (*VLAN tagging*). Tag představuje další hlavičku přenášeného paketu. Tato hlavička obsahuje informace o tom, do jaké virtuální sítě daný paket patří (zdali náleží do komerční nebo studentské sítě). Technologii virtuálních sítí demonstruje obrázek 4.2.

Na výše zmíněného obrázku je technologie *VLAN* využita pouze na úrovni komunikace jednotlivých páteřních přepínačů. Ovšem využití virtuálních sítí je možné i na úrovni směrovačů. Směrovač může být připojen pouze jedním síťovým rozhraním. Přes toto rozhraní přijímá pakety z různých virtuálních sítí a zpětně je směruje dle směrovacích pravidel k dalšímu aktivnímu prvku. Pro tuto funkčnost je však nutná podpora ze strany aktivních prvků sítě. Využití této technologie je přínosné zvláště na routerech, které postrádají dostatek síťových rozhraní, na druhou stranu klade vyšší výkonové požadavky na router, který musí u každého paketu kromě standardních hlaviček zpracovávat navíc tag virtuální sítě.

4.3 Routery v kolejní síti

Oblastí zájmu této práce jsou routery. Těchto zařízení je v prostředí kolejní sítě šest. Pět z nich realizuje vzájemné propojení areálů v rámci studentské sítě, jeden zajišťuje komunikaci mezi areály v rámci komerční sítě. Tento nepoměr je dán rozdílným zatížením těchto sítí. Studentská část sítě čítá několik tisíc uživatelů a poskytuje uživatelům prakticky neomezenou šířku pásma, zatímco komerční síť využívá asi třicet uživatelů, kteří vzájemně sdílejí linku o přenosové rychlosti 2Mb za sekundu.



Obrázek 4.2: Princip využití technologie virtuálních sítí (viz IEEE 802.1Q [14]) při rozdělení jedné síťové infrastruktury na dvě logické sítě. Obrázek ukazuje dva počítače, jejichž pakety musí být směrovány do sítě Cesnetu a jeden počítač, jehož pakety jsou směrovány do sítě komerčního providera. Pro zajištění správného směrování paketů je nutné označit porty uživatelského přepínače (tj. vybrat, které porty patří do jaké logické sítě). Porty 2 a 3 koncového přepínače náleží do sítě KolejNet, port 4 do komerční sítě, port 1 využívá tagování virtuální sítě (*VLAN tagging*) a jsou jeho pomocí přenášeny informace z obou logických sítí do nadřazeného přepínače. Tento přepínač rozlišuje na základě tagu virtuální sítě, kterému směrovači má paket přeposlat. Vlastní router zajistí směrování paketu mimo areál kolejí. Pokud by v tomto případě chtěl počítač pc2 komunikovat s pc3, není tato komunikace realizována přímo v nadřazeném přepínači, ale přes soustavu nadřazených routerů (tj. přes internet).

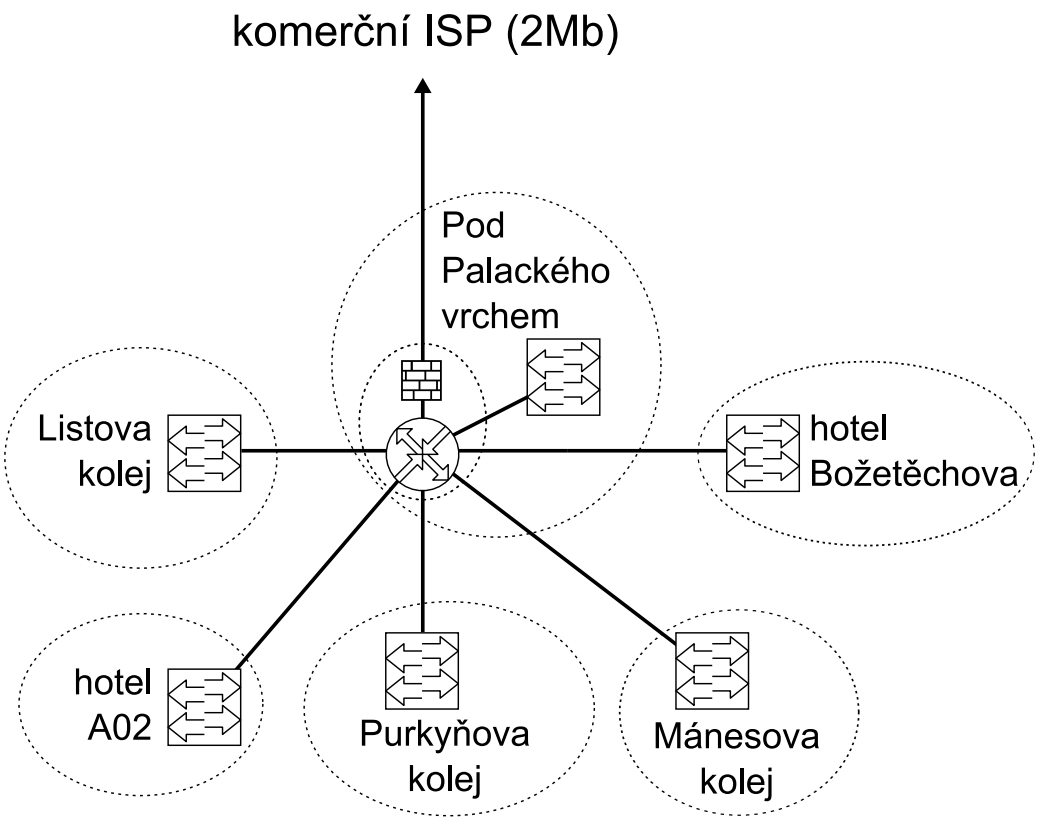
4.3.1 Routery ve studentské části sítě

Pro směrování paketů mezi studentskou částí sítě KolejNet a sítí Cesnet jsou v současné době využívány velmi výkonné hardwarové routery od firmy Extreme Networks. V areálech Pod Palackého vrchem, Listova kolej a Purkyňova kolej je nasazen SummitX450. Tento router má k dispozici 24 gigabitových portů (optických nebo metalických) a zároveň umožňuje připojení speciálního modulu. Tento modul obsahuje dvě 10Gb optická rozhraní, která zajišťují vzájemnou komunikaci mezi výše uvedenými areály (viz obrázek 4.1).

Zbývající dva areály připojují menší počet uživatelů, z toho důvodu stačí pro směrování jejich provozu méně výkonná řešení. V případě Mánesových kolejí je využit prvek Summit1i, který má k dispozici 8 gigabitových portů (6 metalických a 2 optické). Směrování provozu na hotelu Božetěchova obstarává Dell PowerConnect 5224.

4.3.2 Router v komerční části sítě

Poslední router zajišťuje komunikaci uživatelů komerční sítě s uživateli v jiných areálech a také zabezpečuje připojení do internetu. Router je realizován na platformě PC od firmy SuperMicro. Tento počítač se skládá z procesoru Intel Pentium 4, 1GB paměti a dvou gigabitových síťových rozhraní. Na tomto systému je nasazen operační systém *FreeBSD* ve verzi 6.2. Tento operační systém má podporu pro směrování přímo v jádře operačního systému, zároveň umožňuje využít technologii virtuálních sítí. Operační systém rovněž zajišťuje funkci firewallu a realizuje překlad síťových adres (*NAT*). Topologii komerční sítě popisuje obrázek 4.3. Výše popsaný hardware počítače společně s operačním systémem definují prostředí, pro které je určen navrhovaný logovací systém. Konfiguraci routeru v komerční síti se věnuje následující kapitola.



Obrázek 4.3: Logická topologie komerční sítě na kolejích VUT.

4.4 Konfigurace routeru v komerční síti

Popis celé konfigurace PC routeru není cílem této práce. Pro bližší pochopení principu logování na reálném systému je důležité znát především konfiguraci síťových rozhraní a služeb, které mohou mít vliv na chování navrhovaného logovacího systému. Jedná se především o služby pro překlad síťových adres. V následující části se od čtenáře očekává základní znalost *UNIXových* operačních systémů a sítí. Hlavním zdrojem informací pro sepsání této kapitoly je [11] a [12].

4.4.1 Síťová rozhraní

Výše popsaný počítačový systém obsahuje dvě gigabitové síťové karty. Jelikož tento router propojuje několik podsítí a zároveň je připojuje do internetu, je nutné na těchto dvou fyzických rozhraních provozovat více síťových rozhraní. Pro tento účel je nejvýhodnější využít technologii *VLAN* (viz obrázek 4.2). Konfigurace jednoho síťového rozhraní tak, aby pracovalo s určitou *VLANou* na daném fyzickém síťovém rozhraní, je v operačním systému *FreeBSD* velice jednoduché. Pro definici virtuálního rozhraní je třeba zapsat do souboru `/etc/rc.conf` následující volby:

```
ifconfig vlan0 create
ifconfig_vlan0=”inet 10.10.0.1 netmask 255.255.255.0 vlan 119 vlandev em0”
```

Volba zapsaná na prvním řádku zajistí vytvoření nového virtuálního rozhraní s názvem *vlan0* (případná další rozhraní se rozlišují číslicí na konci). Volba uvedená na druhém řádku zajišťuje propojení vytvořeného rozhraní s konkrétní *VLANou* na fyzickém rozhraní *em0*. Virtuální síť je

určená číslem, v tomto případě 119, což je číslo, které se je součástí tagu virtuální sítě. Rozhraní *em0* značí fyzické rozhraní gigabitové síťové karty. Druhý řádek konfiguračního souboru zároveň přidělí vytvořenému rozhraní *vlan0* IP adresu a masku podsítě. Podobným způsobem jsou vytvořené virtuální sítě pro jednotlivé areály tak, že je možné se ke komerční síti připojit ze všech lokalit kolejí VUT (viz nástin topologie na obrázku 4.3). Skutečná topologie je ještě o něco složitější, neboť každý areál se dále dělí na jednotlivé bloky kolejí, které odpovídají jednotlivým lokálním sítím.

Aby byla výše zmíněná konfigurace síťových rozhraní funkční, je nutné dovést *VLANy* z jednotlivých lokalit až k tomu aktivnímu prvku, který je fyzicky propojen UTP kabelem s PC routerem komerční sítě (respektive jeho síťovou kartou). Rovněž je nutné nastavit aktivní prvek tak, aby dané *VLANy* na příslušném portu „otagoval“. Tímto postupem je dosaženo stavu, kdy je na jedné síťové kartě (*em0*) funkční komunikace z jednotlivých lokalit kolejí VUT s příslušným virtuálním rozhraním (*vlan0, vlan1* . . .) na routeru komerční sítě.

Druhá síťová karta (*em1*) zajišťuje komunikaci s poskytovatelem internetu.

4.4.2 Překlad síťových adres

V komerční síti na kolejích VUT je nasazen *PAT*. Jeho správnou funkci zajišťuje PF firewall [13]. Problém spjatý se současným nasazením technologie *PAT* a logováním paketů, který rozebírá kapitola 3.5.3, je řešen přesunem logovacího systému na vnitřní rozhraní routeru tak, aby měl logovací systém přístup k nezměněným hlavičkám paketů. Jelikož router komerční sítě připojuje k internetu několik lokálních sítí, musí logovací systém sledovat provoz na příslušných vnitřních rozhraních routeru *vlan0, vlan1* . . .). Pro snazší návrh a implementaci logovacího systému (těm se věnuje zbývající část této práce) je veškerá komunikace na vnitřních rozhraních routeru zrcadlena do virtuálního síťového rozhraní *pflog0*. Jedná se o síťové rozhraní poskytované logovacím démonem PF firewallu *pflogd*, na které je možné, díky filtrovacím pravidlům PF firewallu, zaznamenat požadovanou komunikaci. Pravidla PF firewallu potřebná k zajištění funkce *PAT* a zaznamenání komunikace na vnitřních rozhraních routeru do virtuálního rozhraní *pflog0* jsou následující:

```
int_if="vlan1"
klist_if="vlan8"
int_net="10.10.0.0/24"
klist_net="10.229.200.0/22"

...
nat on $ext_if from $int_net to any -> $ext_addr
nat on $ext_if from $klist_net to any -> $ext_addr
...
pass out log on $int_if from any to $int_net
pass in  log on $int_if from $int_net to any
pass out log on $klist_if from any to $klist_net
pass in  log on $klist_if from $klist_net to any
...
```

Na této části konfiguračního souboru PF firewallu `/etc/pf.conf` jsou důležitá klíčová slova „nat“ a „log“. Klíčové slovo „nat“ definuje paradoxně službu *PAT*. (O tom, že se tyto dvě služby často zaměňují, již bylo řečeno.) Paket, na který se aplikovalo pravidlo obsahující klíčové slovo „log“, je zpracován démonem *pflogd* a tím pádem se objeví na virtuálním síťovém rozhraní *pflog0*.

Tato architektura umožňuje logovacímu systému běh pouze nad jediným virtuálním síťovým rozhraním (*pflog0*) namísto běhu nad všemi výše zmíněnými rozhraními (*vlan0, vlan1* . . .). Tímto je router v komerční síti na kolejích VUT připraven pro snadné nasazení NetFlow exporteru.

Kapitola 5

Knihovna libpcap

Na základě volně dostupných implementací programů pro monitorování sítě (viz kapitola 3.6) byla jako nejvhodnější technologie pro návrh a implementaci logovacího systému vybrána knihovna *libpcap*.

Jedná se o knihovnu, která umožňuje přistupovat k síťovým rozhraním hostitelského systému a číst z nich jednotlivé rámce linkové vrstvy ISO/ISO modelu. Tímto mechanismem jsou dostupné nejen rámce určené pro hostitelský systém, ale také všechny rámce procházející tímto rozhraním.

Knihovna je součástí většiny *UNIXových* operačních systémů, zároveň existuje její implementace *winpcap* pro operační systém *Windows*. Jedná se tedy o multiplatformní nástroj. Základem této knihovny je BPF (BSD Packet Filter), jež je součástí operačních systémů jako *FreeBSD, OpenBSD a jiných*. Nespornou výhodou této knihovny je její *BSD* open-source licence, která umožňuje její použití ve většině produktů.

Aplikací, ve kterých je možné tuto knihovnu nasadit, je mnoho. Jedná se o různé protokolové analyzátory, síťové monitory, systémy IDS/IPS, síťové skenery a také systémy pro logování provozu.

5.1 Základní funkce knihovny

Jak již bylo řečeno v úvodu, tato knihovna umožňuje programátorovi přistupovat přímo k syrovým datům (*raw data*) načítaným ze síťového rozhraní. Načítání dat, rámců linkové vrstvy, je možné v obou směrech komunikace (*in/out*), komunikaci je navíc možné filtrovat pomocí souboru vlastních filtrovacích pravidel. Další užitečnou funkcí knihovny je automatické zaznamenávání statistických informací o síťovém provozu.

Rozhraní, poskytované touto knihovnou, je na velmi vysoké úrovni. Představuje systémově nezávislé API, které řeší odlišnosti v přístupu k síťovým rozhraní v jednotlivých operačních systémech. Nejdůležitější funkce tohoto API jsou popsány v následující kapitole.

5.2 Popis funkcí

Při návrhu a implementaci aplikace s využitím knihovny *libpcap* je důležité vykonat určitou posloupnost logických kroků. Vykonáním této posloupnosti kroků ve správném pořadí je navrhované aplikaci zajištěna plná funkčnost jednotlivých částí knihovny, především pak funkčnost integrovaného paketového filtru. Označení „paketový“ filtr má spíše zjednodušující charakter, správnější by bylo, mluvit o filtru rámců. Struktura aplikace využívající knihovnu *libpcap* je následující:

I: Volba požadovaného rozhraní. Pro operační systém *Linux* je typickým síťovým rozhraním např. *eth0*, v *BSD* systémech např. em0, fxp0 a jiná.

II: Inicializace knihovny. Při této operaci je knihovně předán název rozhraní. Zároveň je vytvořena struktura podobná souborovému ukazateli (*File Handle*), jejíž pomocí se dále s knihovnou komunikuje.

III: Filtrace paketů. Pokud je třeba přijímanou komunikaci filtrovat (např. vybrat pouze ty pakety, které směřují na webový server) je nutné příslušné pravidlo vytvořit, zkompilovat a následně aplikovat. Všechny tyto operace jsou realizovány voláním určité funkce z knihovny (není potřeba žádná externí aplikace). Pravidlo je nutné nejprve vytvořit, k tomu slouží klíčová slova a operátory uvedené v kapitole 5.2.3. Vytvořené pravidlo je představováno řetězcem, který se procesem kompilace převede na *libpcap* formát. Posledním krokem je vlastní aplikace již zkompilovaného pravidla.

IV: Hlavní cyklus. V tomto stavu *libpcap* čeká, dokud nepřijme požadovaný počet paketů. Poté na každý přijatý paket aplikuje uživatelsky definovanou funkci.

V: Ukončení práce s knihovnou. Uvolnění paměti.

Těchto pět kroků představuje jádro aplikačního rozhraní knihovny *libpcap*.

5.2.1 Volba rozhraní

Existují dva způsoby výběru správného rozhraní. Buď jej programátor zadá přímo, nebo je možné využít funkci *pcap_lookupdev()*. Přímým zadáním názvu rozhraní je myšleno využití konstanty, načtení názvu rozhraní z příkazové řádky, případně načtení z konfiguračního souboru. Záleží tedy jedině na programátorovi jakým způsobem název rozhraní získá. Druhou možností je využít funkci *pcap_lookupdev()* definovanou prototypem:

```
char *pcap_lookupdev(char *errbuf);
```

Tato funkce vrací ukazatel na řetězec obsahující název prvního síťového rozhraní operačního systému. Pořadí síťových rozhraní je dáno operačním systémem. V případě chyby vrací funkce hodnotu *NULL* a do řetězce *errbuf* vloží chybové hlášení.

Funkce nachází uplatnění v případech, kde nebyl název rozhraní uživatelem specifikován. Tím dává aplikaci ještě určitou šanci vybrat správné rozhraní pro svou činnost. Tato funkce je zvláště užitečná na uživatelských stanicích, které mají obvykle pouze jedno síťové rozhraní, a proto tato funkce vrátí požadovaný název rozhraní.

5.2.2 Inicializace

Inicializace zajišťuje vytvoření tzv. *sniffing session* na základě volání funkce *pcap_open_live()*. Prototyp této je definován následovně:

```
pcap_t *pcap_open_live(char *device,int snaplen,int p,int to_ms,char *ebuf);
```

Prvním parametrem je název síťového rozhraní zvoleného na základě předchozí kapitoly. Druhý parametr *snaplen* definuje maximální počet bytů, které budou z jednoho paketu načteny. Je-li třetí parametr nastaven na nenulovou hodnotu, bude vybrané síťové rozhraní přepnuto do promiskuitního

módu. V tomto módu přijímá síťová karta i ty pakety, které nejsou přímo adresovány některému síťovému rozhraní této karty. Vypnutí promiskuitního módu tímto parametrem nemusí fungovat, neboť promiskuitní mód může být vynucen jinou aplikací, případně jádrem operačního systému. Parametr *to_ms* představuje zpoždění mezi načtením dat ze síťového rozhraní a jejich následným zpracováním. Poslední parametry má stejný význam jako u funkce *pcap_lookupdev()*. Funkce vrací tzv. *session handle*, což je struktura, se kterou se pracuje podobně jako se souborem.

5.2.3 Filtrace paketů

Jenom výjimečně je potřeba sledovat veškerou komunikaci na síťovém rozhraní. V takovém případě je velmi výhodné využít služeb filtru, jež je součástí knihovny *libpcap*. Práce s tímto filtrem se skládá ze tří kroků: vytvoření pravidla, jeho kompilace a následná aplikace pravidla na otevřenou *sniffing session*.

Pravidlo je určeno logickým výrazem. Pro jeho správné sestavení je nutné znát jeho syntaxi a sémantiku. Výraz se skládá jednoho nebo více primitiv. Primitiva obsahují identifikátor (jméno nebo číslo), jemuž předchází jeden nebo více kvalifikátorů. Kvalifikátory se dělí do třech skupin:

- **type** udává subjekt, ke kterému se vztahuje identifikátor. Povolené typy jsou *host*, *net* a *port*. Např.: 'host merlin', 'net 147.229', 'port 53' atd. Jestliže není vybrán žádný typ, použije se výchozí typ *host*.
- **dir** specifikuje směr komunikace. Povolené směry jsou: *src*, *dst*, *src or dst* a *src and dst*. Např.: 'src merlin', 'dst net 147.229' atd. Výchozí hodnotou je *src and dst*.
- **proto** udává typ komunikačního protokolu. Tímto protokolem mohou být: *ether*, *fddi*, *wlan*, *ip*, *ip6*, *arp*, *icmp*, *icmp6*, *tcp*, *udp* a další.

Existují ještě další speciální primitiva: *broadcast*, *multicast* a *gateway*. Jejich význam je zřejmý. Komplexnější filtry využívají kromě výše popsaných primitiv navíc operátory. Jejich pomocí je možné kombinovat významy jednotlivých primitiv. Operátory, které je možno využít ve filtrovacích pravidlech jsou: *and*, *or* a *not*. Výsledné pravidlo se tedy skládá z primitiv a operátorů. Jejich pořadí je dáno následujícím vzorem:

```
<primitivum> <operátor> <primitivum> <operátor> <primitivum> ...
```

Filtrovací pravidlo, vytvořené na základě výše zmíněné syntaxe, je nutné dále zkompilovat. K tomu slouží funkce: *pcap_compile()*. Prototyp jí definuje jako:

```
int pcap_compile(pcap_t *p, struct prog *fp, char *str, int o,  u_int32 mask);
```

První parametr představuje ukazatel, na konkrétní instanci knihovny (*libpcap)* (*session handle*). Parametr *prog* představuje ukazatel na volné paměťové místo, na které bude uložen zkompilované pravidlo filtru. Třetím parametrem je řetězec, představující pravidlo filtru. Optimalizaci zkompilovaného pravidla zajišťuje parametr *o*. Posledním parametrem je síťová maska, jejíž pomocí je možné vybrat komunikaci, na kterou bude pravidlo aplikováno. Funkce vrací hodnotu *-1* při chybě.

Aplikaci pravidla zajišťuje funkce *pcap_setfilter()* daná prototypem:

```
int pcap_setfilter(pcap_t *p, struct prog *fp);
```

První parametr představuje ukazatel, na konkrétní instanci knihovny (*libpcap)* (*session handle*). Parametr *prog* představuje ukazatel na již zkompilované pravidlo. Funkce vrací hodnotu *-1* při chybě.

5.2.4 Načítání paketů

Existují dva způsoby načítání paketů, které se odlišují počtem načítaných paketů a způsobem jejich zpracováním. Prvním způsobem je načtení jednoho paketu a jeho následné zpracování. Tento způsob není pro praktické nasazení příliš výhodný. Druhým způsobem je načtení několika paketů, na které je aplikována callback funkce. Tento způsob je využíván ve většině aplikací hlavně díky své robustnosti. Funkce realizující nekonečný cyklus pro načítání paketů se jmenuje *pcap_loop()*. Je definovaná jako:

```
int pcap_loop(pcap_t *p, int cnt, pcap_handler callback, u_char *user);
```

První parametr představuje ukazatel, na konkrétní instanci knihovny (*libpcap)* (*session handle*). Následující parametr *cnt* říká kolik paketů má být načteno před jejich vlastním zpracováním. Třetím parametrem je jméno callback funkce (bez závorek). Posledním parametrem je možné specifikovat uživatele, s jehož oprávněními budou pakety načítány ze síťového rozhraní.

Callback funkce, předávaná jako parametr funkci *pcap_loop()*, musí mít tři parametry, jejichž pomocí jsou předávány informace o zpracovávaném paketu. Rozhraní callback funkce popisuje její prototyp:

```
void got_packet(u_char *args, const struct pcap_pkthdr *hdr, const u_char *pkt);
```

Parametr *args* odpovídá prvnímu parametru funkce *pcap_loop()*. Druhý parametr *hdr* je ukazatelem na strukturu *pcap_pkthdr*. Tato struktura obsahuje informace o načteném paketu (např. jeho velikost). Definice této struktury následuje pod tímto odstavcem. Poslední parametr představuje ukazatel na paměťový blok, ve kterém je uložen načtený paket (respektive tolik bytů od začátku paketu, kolik bylo předáno funkci *pcap_open_live()* v parametru *snaplen*).

```
struct pcap_pkthdr {
  struct timeval ts; /* time stamp */
  bpf_u_int32 caplen; /* length of portion present */
  bpf_u_int32 len; /* length this packet (off wire) */
};
```

5.2.5 Ukončení práce

Ukončením práce se rozumí uzavření *session handle* a uvolnění paměti po zkompilovaném pravidle. Funkce, které zabezpečují tyto činnosti jsou:

```
void pcap_close(pcap_t *p);
void pcap_freecode(struct bpf_program *);;
```

Tímto byly vysvětleny funkce knihovny *libpcap*, které jsou potřebné při implementaci systému pro logování paketů. Návrhu tohoto systému se věnuje následující kapitola.

Kapitola 6

Návrh a implementace systému

Návrh systému je důležitým krokem při vývoji jakékoli nové aplikace, který by se nikdy neměl podcenit. Kvalitním a promyšleným návrhem je možné se vyhnout různým komplikacím při implementaci. Naopak na základě špatného návrhu mohou vyvstat různé neočekávané problémy, vinou kterých se může doba vývoje aplikace několikanásobně prodloužit. Proto je návrhu systému věnována podstatná část této práce.

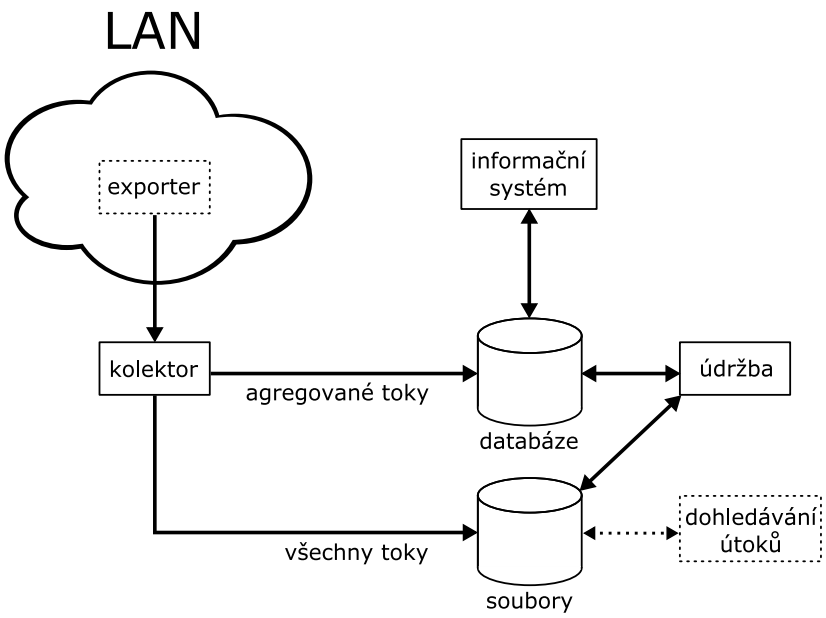
Při návrhu systému pro logování paketů je důležité uvážit všechny aspekty tohoto systému. Především pak jeho paměťovou náročnost, požadavky na výkonnost procesoru, potřebný diskový prostor apod. Návrh systému musí být rovněž podřízen prostředí, pro které je navrhován tak, aby bylo minimalizováno riziko neočekávaných komplikací při nasazení systému v cílovém prostředí. Toto prostředí je popsáno v kapitole 4. Na základě znalosti výkonnostních požadavků na systém a znalosti podobných systémů byl jako implementační jazyk pro celý logovací systém zvolen jazyk *c*. Ten poskytuje potřebný výkon a knihovny, které značně usnadňují vlastní implementaci. Znalost tohoto programovacího jazyka je pro pochopení následujícího textu nezbytná.

6.1 Základní architektura systému

Základní architektura systému se skládá z několika nezávislých částí. První část realizuje funkci NetFlow exporteru. Druhou částí je démon, zajišťující funkci NetFlow kolektoru. Třetí částí je databáze. Další částí jsou komprimované logy uložené na pevném disku. Poslední část realizuje úklid starých záznamů v databázi a také smazání starých souborů s logy na pevném disku. Vztah jednotlivých částí ukazuje obrázek 6.1.

Význam prvních dvou částí, exporteru a kolektoru, je popsán v kapitole 3.2.2. Exporter načítá informace o datových tocích ze síťového rozhraní a na získaná data aplikuje agregační funkci (viz kapitole 3.3.2). Agregované informace o datových tocích zasílá pomocí UDP paketu do exporteru, který tyto pakety přijímá a zpracovává. Načtené informace o datových tocích ukládá do soubor na pevném disku. Do databáze vkládá výrazně agregované datové toky tak, aby nebyla databáze příliš zatěžována, a zároveň aby poskytovala dostatečně přesné informace o objemu přenesených dat.

Hlavním účelem databáze je poskytovat informace o objemu přenesených dat informačnímu systému KolejNetu. Soubory ukládané na pevný disk obsahují přesné informace o tom kdo, kdy, s kým a jak komunikoval. Na základě těchto informací je možné dohledávat původce počítačových útoků. Tyto informace zároveň musí každý poskytovatel internetu uchovávat po dobu šesti měsíců, tak jak to ukládá zákon o elektronických komunikacích 485/2005 Sbírky (viz kapitola 2).



Obrázek 6.1: Základní architektura systému.

6.2 NetFlow exporter

První a zároveň nejsložitější částí systému je zcela jistě exporter. Ten načítá jednotlivé pakety ze síťového rozhraní. Z hlaviček paketu rozpozná datový tok, který představuje index (klíč) asociativní paměti. Klíč ukazuje na konkrétní položku v paměti, ve které jsou uloženy čítače, monitorující datový tok. Jeli datový tok sledován dostatečně dlouho, dochází k jeho přesunutí z asociativní paměti do fronty. V této frontě čeká dokud nedojde k ukončení více datových toků. Jestliže je ve frontě dostatečný počet ukončených datových toků, jsou jednotlivé záznamy o datových tocích vloženy do struktury NetFlow v5 (viz kapitola 3.4) a dále je tato struktura vyslána pomocí UDP datagramu do kolektoru.

V dokumentaci technologie NetFlow se odeslání toku z exporteru do kolektoru děje v případě: ukončení TCP spojení (FIN, RST), vypršení času od přijetí prvního paketu z daného toku, vypršení času od přijetí posledního paketu z daného toku, z důvodu uvolnění paměti nebo při hrozbě přetečení čítačů. Pro účely měření objemu přenesených dat a zaznamenávání datové komunikace ve smyslu zákona 485/2005 Sbírky, je postačující odesílat toky z exporteru do kolektoru pouze v případě vypršení času od přijetí posledního paketu z daného toku, z důvodu uvolnění paměti nebo při hrozbě přetečení čítačů.

Tím byla vysvětlena zjednodušená funkce exportu. Následující kapitoly se věnují detailům jednotlivých částí této aplikace.

6.2.1 Čtení paketu

Pro načítání paketů je využito aplikační rozhraní knihovny *libpcap* (viz kapitola 5). Její pomocí se z každého paketu postupně extrahují hlavičky linkové, síťové a transportní vrstvy ISO/OSI modelu. Z těchto hlaviček jsou nejdůležitější ty položky, které definují datový tok. Jedná se o zdrojovou IP adresu, cílovou IP adresu, zdrojový port, cílový port a protokol zabalený do IP paketu (obvykle TCP nebo UDP). Těchto pět položek definuje datový tok v rámci tohoto systému. Oproti standardní

definici datového toku dle firmy *Cisco*, která datový tok definuje mnohem komplexněji, nabízí pětiprvková definice datového toku určité výhody. Hlavní výhodou je menší paměťová náročnost a tím pádem i rychlejší porovnávání a vyhledávání. Nevýhodou je menší diferenciace datových toků. Implementace této části exporteru využívá funkce knihovny *libpcap*. Především pak funkci *pcap_loop()*, která na každý přijatý paket volá callback funkci *parse_link_layer()*. Ta je definovaná prototypem jako:

```
void parse_link_layer(u_char *args,
                      const struct pcap_pkthdr *header,
                      const u_char *packet);
```

Jednotlivé parametry této funkce odpovídají definici callback funkce (viz kapitola 5.2.4). Callback funkce je optimalizována tak, aby strojový čas, potřebný na vykonání této funkce, byl minimální. Při příliš dlouhé obsluze jednoho paketu, by došlo k zaplnění čtecího bufferu aplikačního rozhraní knihovny *libpcap* a tím pádem i k zahazování příchozích paketů, což je zjevně nežádoucí stav. Bližšímu popisu callback funkce se věnuje následující text.

Funkce *parse_link_layer()* nejprve „přeskočí“ hlavičku linkového rámce. Aby bylo možné tuto hlavičku přeskočit, je nutné znát její velikost v bytech. Velikost této hlavičky závisí na typu síťového rozhraní.

Nejčastějším typem síťového rozhraní je ethernetového rozhraní. Linkovou hlavičku ethernetového rozhraní popisuje tabulka 6.1. V tomto případě má hlavička velikost 14 bytů. Existují ovšem i jiné typy rozhraní, které mají odlišnou velikost hlavičky linkového rámce. Speciálním případem je rozhraní *pflog* (viz kapitola 4.4.2). Jedná se o virtuální rozhraní poskytované PF firewallem, na které je zaznamenávána vybraná komunikace. Výběr této komunikace zajišťují pravidla firewallu. Linkovou hlavičku virtuálního rozraní *pflog* popisuje tabulka 6.2. Tato hlavička má velikost 48 bytů.

Bytů	**Popis**
6	Fyzická adresa cíle
6	Fyzická adresa zdroje
2	Protokol síťové vrstvy (IP, ARP, RARP a jiné)

Tabulka 6.1: Položky obsažené v hlavičce ethernetového rámce.

Bytů	**Popis**
1	Délka hlavičky
1	Rodina adres
1	Akce firewallu
1	Důvod akce
16	Název rozhraní, na kterém byla komunice odchycena
16	Množina pravidel postupně aplikovaných na paket
4	Číslo posledního aplikovaného pravidla
4	Číslo posledního aplikovaného subpravidla
1	Směr komunikace
3	Zarovnání

Tabulka 6.2: Položky obsažené v hlavičce pflog rámce.

Když funkce *parse_link_layer()* přeskočí hlavičku linkového rámce, načte hlavičku IP protokolu, ze které zjistí především tři položky: zdrojovou IP adresu, cílovou IP adresu a protokol IP vrstvy. Pokud je tímto protokolem TCP nebo UDP, je dále načten zdrojový a cílový port. V opačném případě jsou čísla portů nastavená na nulu. Těchto pět položek definuje datový tok. Na konci této funkce je datový tok přidán do paměti datových toků. Tuto činnost realizuje funkce flow_add(). Před popisem činnosti této funkce, je nutné seznámit se s paměťovým modelem.

6.2.2 Hlavní paměťový model

Získanou informaci o probíhajícím datovém toku je nutné uložit do paměti. Přitom je výhodné na tuto informaci zároveň aplikovat agregační funkci. Pro zajištění takové funkčnosti je nutné využít asociativní paměť. Hodnota v asociativní paměti není určena adresou, jako v případě „normální“ paměti, ale klíčem. Klíčem této asociativní paměti jsou informace o datovém toku (výše zmíněná pětice údajů). Klíč asociativní paměti popisuje struktura *item_key*, která je definovaná jako:

```
struct item_key  {
  u_long    srcaddr;      // Zdrojová IP adresa
  u_long    dstaddr;      // Cílová IP adresa
  u_short   srcport;      // Zdrojový port TCP/UDP nebo ekvivalentní hodnota
  u_short   dstport;      // Cílový port TCP/UDP nebo ekvivalentní hodnota
  u_char    prot;         // IP protokol (např.: TCP = 6; UDP = 17)
};
```

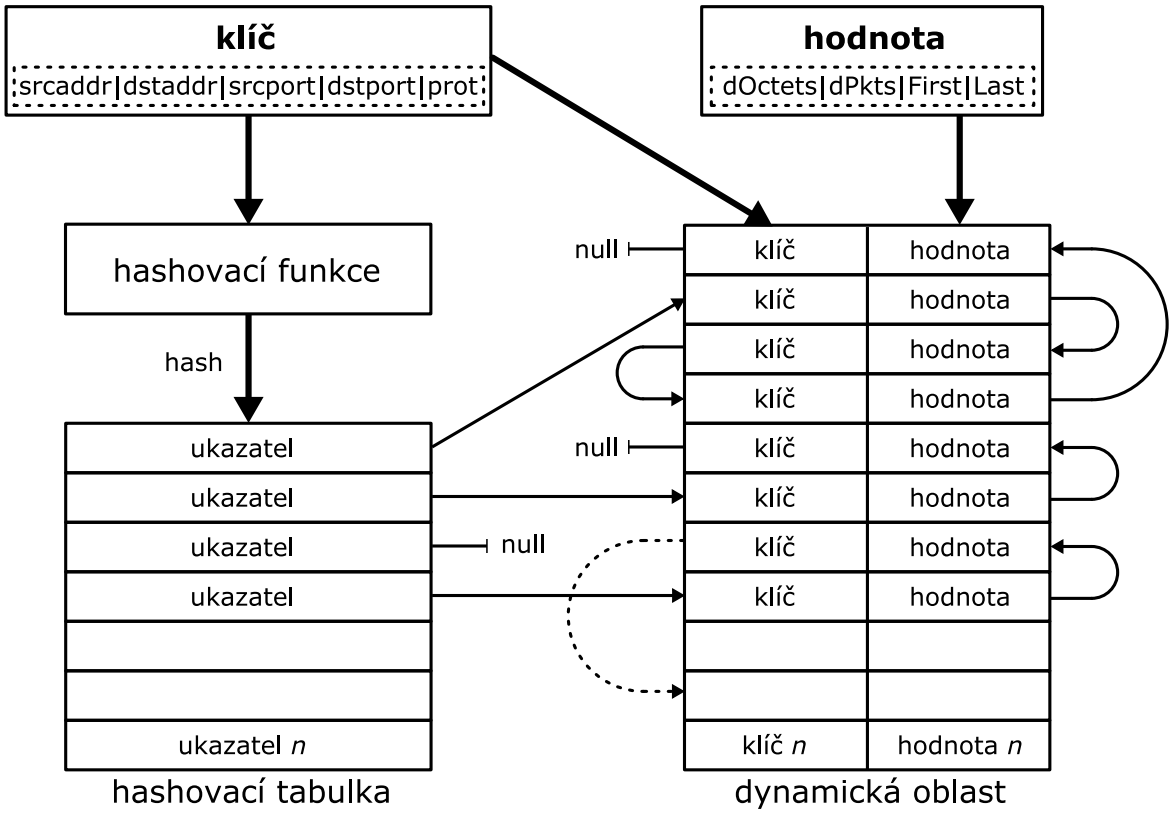
Hodnotu představují dvě počítadla a dva časové údaje. První počítadlo sleduje celkový počet bytů přenesených během datového toku. Druhé zaznamenává celkový počet přenesených paketů. Časové údaje vymezují interval od kdy a do kdy datový tok trval. Hodnotu popisuje struktura definovaná jako:

```
struct item_val  {
  u_long    dPkts;      // Počet paketů v˜datovém toku
  u_long    dOctets;    // Celkový počet bytů přenesených během datového toku
  u_long    First;      // Doba začátku datového toku
  u_long    Last;       // Doba konce datového toku
};
```

Nevýhodou asociativní paměti je nutnost její implementace, neboť standardní programovací prostředky (například kontejner *map* implementovaný v jazyce *c++*) nepředstavují dostatečně výkonná řešení. Řešením, které se nabízí, je hashovací tabulka (zkráceně *hashtable*) v kombinaci s lineárním seznamem. Ta představuje dostatečně výkonné řešení za cenu složitější implementace.

Hlavní paměť pro uložení datových toků je rozdělena na statické pole a dynamickou oblast. Hashovací tabulka je realizována statickým polem, které je indexováno hodnotou hashovací funkce. V tomto poli je pak vytvořen odkaz do dynamické oblasti, kde jsou udržovány informace o aktivních datových tocích. Struktura tohoto paměťového modelu je znázorněna na obrázku 6.2. Návrh vhodné hashovací funkce je z výkonnostního hlediska velmi důležitý, neboť právě hashovací funkce určuje efektivitu hashovací tabulky. Tato funkce musí být zároveň dostatečně rychlá, neboť její běh navyšuje strojový čas potřebný pro vykonání callback funkce.

Navržená hashovací funkce využívá bitového operátoru *xor*, jehož časová náročnost je, na většině současných aritmeticko-logických jednotkách, zanedbatelné. Hashovací funkci nejlépe popisuje vlastní zdrojový kód:



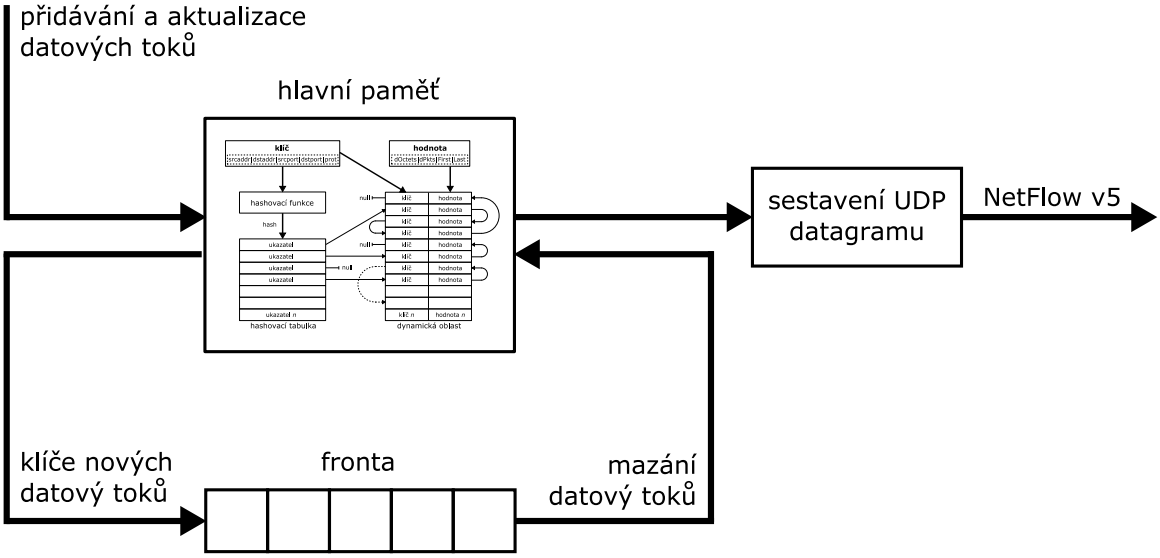
Obrázek 6.2: Struktura paměťového modelu.

```
inline u_long get_hash(struct item_key *k) {
  u_long tmp;
  tmp = k->srcaddr;
  tmp ^= k->dstaddr;
  tmp ^= k->srcport<<16;
  tmp ^= k->dstport;
  tmp ^= k->prot;
  tmp ^= tmp >> 8;
  tmp ^= tmp >> 16;
  tmp &= HASH_SIZE-1;
  return tmp;
}
```

Hashovací funkce využívá pouze několik triviálních bitových operací (xor, and a logické posuvy) a jedno odečítání. Na základě testování různých mutací této funkce vznikla tato implementace, která v reálném provozu vykazuje až 98 procentní využití hashovací tabulky. Tato hashovací funkce tedy vykazuje dostatečnou rychlost a zároveň umožňuje efektivně využít hashovací tabulku.

6.2.3 Chronologicky uspořádaná fronta

Součástí hlavního paměťového modelu je chronologicky uspořádaná fronta, která zajišťuje funkčnost aktivního časovače. Nástin funkce tohoto časovače je uveden v kapitole 3.6.2. Zjednodušeně jde o to, že každý datový tok je sledován po dobu určitého časového intervalu. Tato doba udává míru agregace. Po vypršení této doby jsou agregované informace o datovém toku kompletní a jsou



Obrázek 6.3: Blokové schéma znázorňující význam aktivního časovače.

předány další části aplikace, která zajišťuje jeho odeslání pomocí UDP datagramu do kolektoru.

K tomu, aby nemuselo být testováno stáří všech datových toků, je využita výše zmíněná chronologicky uspořádaná fronta. V této frontě stačí testovat pouze prvek na jejím vrcholu. Když je datový tok agregován požadovanou dobu (tj. dobu danou aktivním časovačem), tak je klíč z vrcholu fronty vyjmut. Vyjmutý klíč určuje záznam v hlavní paměti, který je následně předán k dalšímu zpracování.

Jednotlivé záznamy v chronologicky uspořádané frontě obsahují kromě klíče také čas začátku datového toku, který se využívá při testování jeho stáří. Chronologické uspořádání je zajištěno tím, že do fronty jsou vkládány pouze nově vznikající datové toky, tedy takové toky, které v danou chvíli nejsou uloženy v seznamu aktivních toků (tj. v dynamické oblasti hlavní paměťové struktury). Strukturu aktivního časovače znázorňuje obrázek 6.3.

6.2.4 Paměťové operace

Při implementaci exporteru se jako nejsložitější ukázala práce s paměťovým modelem. Tu zajišťují čtyři funkce: *flow_add()*, *flow_del()*, *activ_add()* a *activ_del()*. První dvě funkce pracují s hlavní pamětí, druhé dvě s chronologicky uspořádanou frontou.

Přidávání záznamů

Ze struktury paměťového modelu uvedeného na obrázku 6.2 vyplývá, že při přidávání záznamů do paměti se nejprve vypočítá hodnota hashe pro daný klíč. Tento hash slouží jako index do hashovací tabulky. Dále je nutné rozpoznat tři stavy paměťového modelu:

- Pokud je položka hashovací tabulky na pozici dané hashem rovna hodnotě *null*, je vytvořen nový záznam v dynamické oblasti. Dále je na odpovídající pozici v hashovací tabulce vložen ukazatel na nově vytvořený záznam. Klíč určující datový tok je zaznamenán do chronologicky uspořádané fronty aktivního časovače. Čítač určující počet přenesených paketů v jednom toku je nastaven na hodnotu jedna. Čítač určující objem přenesených dat je nastaven na velikost

datové části paketu. Oba časové údaje jsou nastaveny shodně na čas přijetí tohoto paketu systémem.

- Pokud je položka hashovací tabulky, na pozici dané hashem, definovaná (různá od *null*) a zároveň není nalezen odpovídající klíč mezi položkami lineárního seznamu, je vytvořen nový záznam v dynamické oblasti. Dále je na odpovídající pozici v hashovací tabulce vložen ukazatel na nově vytvořený záznam. Klíč určující datový tok je zaznamenán do chronologicky uspořádané fronty aktivního časovače. Čítač určující počet přenesených paketů v jednom toku je nastaven na hodnotu jedna. Čítač určující objem přenesených dat je nastaven na velikost datové části paketu. Oba časové údaje jsou nastaveny shodně na čas přijetí tohoto paketu systémem.
- Pokud je položka hashovací tabulky, na pozici dané hashem, definovaná (různá od *null*) a zároveň je nalezen odpovídající klíč mezi položkami lineárního seznamu, jsou aktualizovány hodnoty čítačů a čas přijetí posledního paketu.

Tímto je popsána kompletní činnost callback funkce a tím i činnost hlavního vlákna exporteru. Z výkonnostního hlediska je důležité, že na každý přijatý paket jsou v rámci callback funkce aplikovány pouze výše zmíněné kroky.

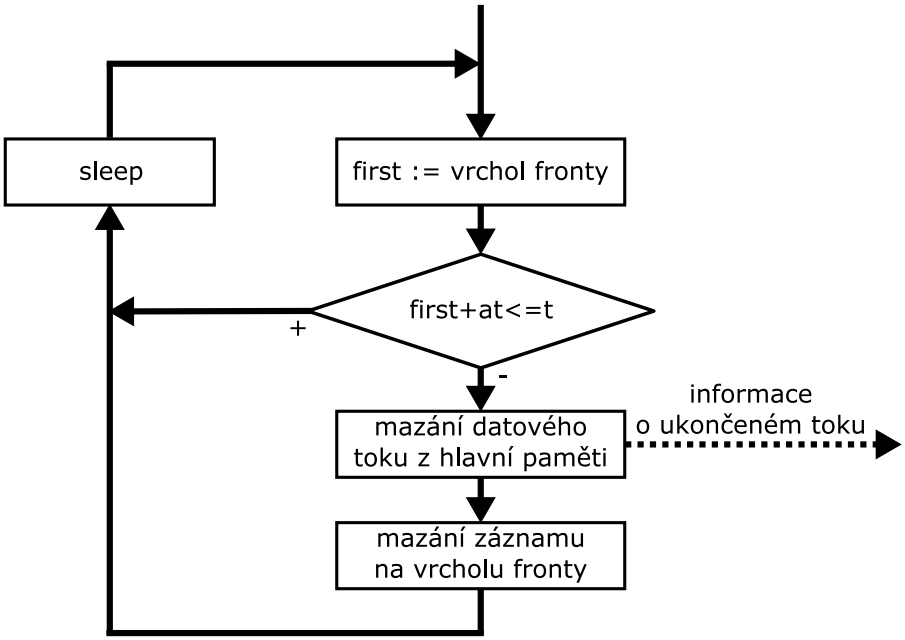
Mazání záznamů

Z výše uvedeného textu vyplývá, že mazání záznamu v hlavní paměti je vyvoláno na základě určitého stáří datového toku. To je dáno rozdílem mezi časem začátku datového toku a aktuálním časem. Čas začátku každého aktivního toku je uložen v chronologicky uspořádané frontě, tím pádem je, na vrcholu této fronty, nejstarší datový tok. Testování stáří tohoto toku zajišťuje vlákno *thread_activ*. Funkci tohoto vlákna popisuje vývojový diagram na obrázku 6.4. Jelikož vkládání položek do fronty zajišťuje hlavní vlákno aplikace a mazání položek z fronty zajišťuje vlákno *thread_activ*, je nutné zajistit správnou synchronizaci těchto vláken při přístupu ke sdílené frontě. K tomu je využit jednoduchý zámek, implementovaný pomocí funkcí *pthread_mutex_lock()* a *pthread_mutex_unlock()* z knihovny *lpthread*.

Mazání záznamů z hlavní paměti není úplně snadné, vyžaduje zajištění konzistence hashovací tabulky, zachování logických celků v dynamické oblasti a správu nevyužitého prostoru dynamické oblasti. Při mazání záznamů z hlavní paměti se nejprve vypočítá hodnota hashe pro klíč načtený z vrcholu chronologicky uspořádané fronty (klíč nejstaršího datového toku). Tento hash slouží jako index do hashovací tabulky. Hodnota načtená z hashovací tabulky představuje ukazatel na začátek lineárního seznamu, ve kterém je hledán záznam se shodným klíčem v dynamické oblasti. Při shodě klíčů je nutné dále ověřit hodnotu začátku datového toku, protože se může jednat o nový datový tok mezi stejnými koncovými uzly (původní datový tok již mohl být smazán jiným mechanismem). Nepodaří-li se požadovaný záznam nalézt, nic se dál neprovádí. Jeli záznam nalezen, následuje operace pro logické vymazání příslušného záznamu v lineárním seznamu a operace pro zachování konzistentního stavu paměti i po vymazání záznamu. Pokud se jednalo o první položku v lineárním seznamu, musí být navíc změněna hodnota v hashovací tabulce.

Operace potřebná pro vymazání záznamu z lineárního seznamu je jednoduchá. Spočívá v přesměrování jednoho ukazatele.

Správa dynamické oblasti hlavní paměti využívá pro zajištění konzistentního stavu myšlenku, kdy je právě vymazávaný záznam nahrazen posledním záznamem v paměti. Tím je dosaženo stavu, kdy jsou na začátku dynamické oblasti souvisle uloženy všechny záznamy o aktivních tocích, za kterými následuje souvislé prázdné místo. Operace zajišťující tuto funkčnost se skládá z přesměrování



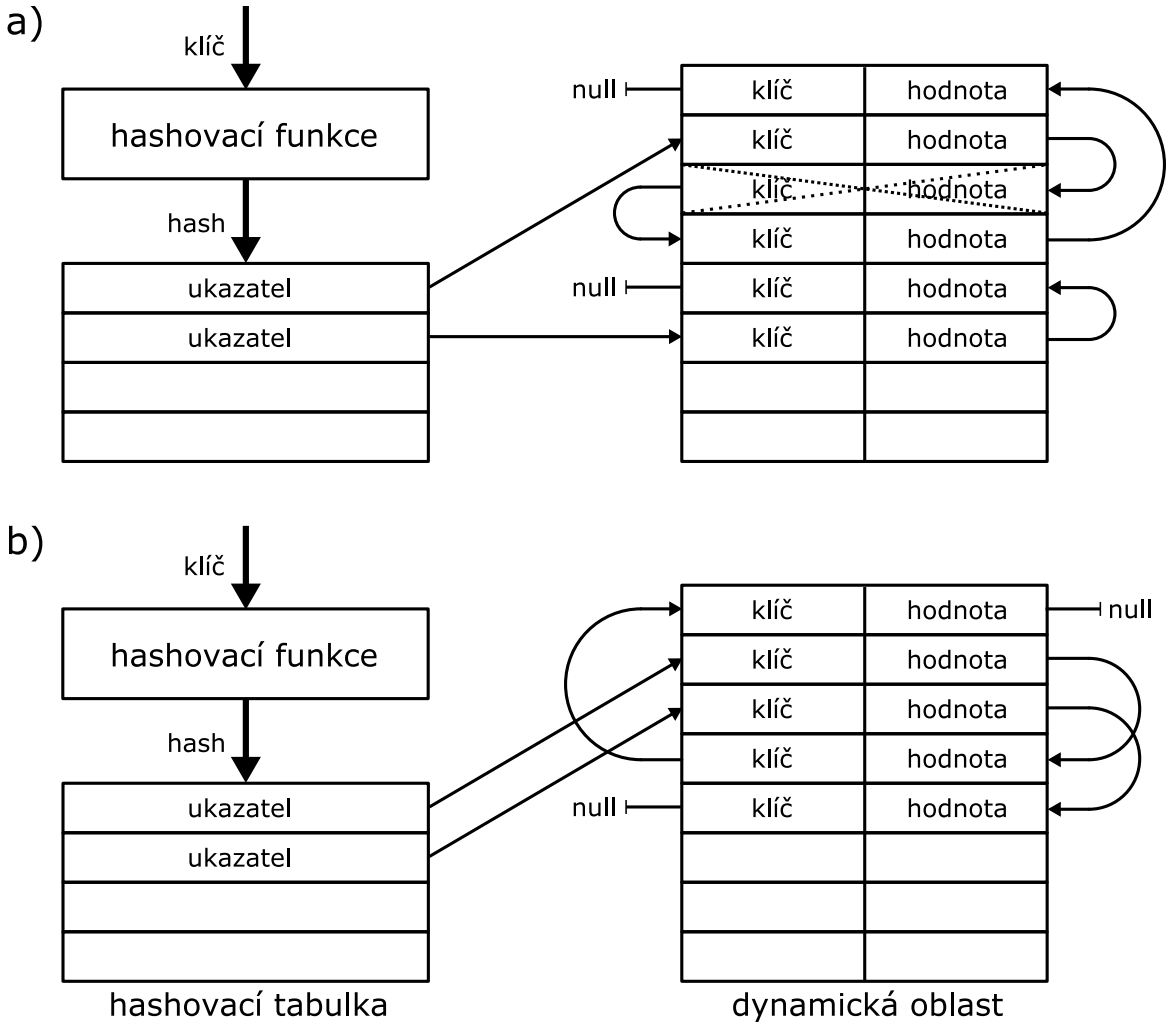
Obrázek 6.4: Vývojový diagram popisující činnost vlákna *thread_activ*. Proměnná *at* představuje hodnotu aktivního časovače (z angl. activ timer). Proměnná *t* představuje aktuální čas (z angl. time).

čtyř ukazatelů a přesunu datové části záznamu. Demonstraci této myšlenky ukazuje obrázek 6.5. Při implementaci těchto operací je důležité uvážit všechny výjimky, které se od standardního schématu odlišují. Jedná se o: mazání prvního záznamu lineárního seznamu, mazání záznamu, uloženého na posledním místě v dynamické oblasti a přesunování prvního záznamu lineárního seznamu na místo smazaného záznamu. Při implementaci těchto výjimek je zachována konzistence paměťového modelu při mazání záznamů.

Je významné, že před vlastním smazáním záznamu jsou informace o datovém toku ukládány do fronty. Přesněji řečeno, jde o dvě fronty o délce třiceti záznamů, protože přesně tolik záznamů je možné vložit do jednoho NetFlow v5 paketu. Když je aktuálně využívaná fronta plná, začnou se další ukončené datové toky ukládat do druhé fronty. Mezi tím se z informacích uložených v prvním bufferu sestaví NetFlow v5 paket, který je následně odeslán do kolektoru. Sestavení a odeslání NetFlow v5 paketu zabezpečuje třetí vlákno aplikace *thread_export*. Synchronizaci mezi tímto vláknem a vláknem *thread_activ* zajišťuje semafor, realizovaný pomocí funkcí *await()* a *advance()*. První funkce umožní přístup do kritické sekce pouze pouze vláknu s lístkem. Druhá funkce funkce předá lístek druhému vláknu, ukončí kritickou sekci a vzbudí pozastavená vlákna. Činnost vlákna *thread_export* popisuje následující kapitola.

6.2.5 Komunikace s kolektorem

Komunikaci exporteru s kolektorem zajišťuje vlákno *thread_export*. Činností, které toto vlákno vykonává je několik. Na začátku se ověří správnost adresy kolektoru, když adresa obsahuje doménové jméno, dojde k jeho převodu na IP adresu. Následně dojde k ověření čísla portu, které může být zadáno buď číslem nebo jménem služby, které je dále přeloženo na odpovídající číslo portu. IP adresa a port určují konkrétní počítač a službu na něm běžící, v tomto případě je tou službou kolektor. Pro připojení ke kolektoru jsou využita systémová volání *socket()* a *connect()*. První z nich vytvoří



Obrázek 6.5: Příklad smazání záznamu z hlavní paměti. *a)* ukazuje původní stav paměti. *b)* ukazuje stav paměti po smazání záznamu.

koncový uzel pro komunikaci. Druhé z nich otevře komunikační kanál. Dále v programu následuje nekonečný cyklus, která ve svém těle čeká, dokud jí nebude předán dostatečný počet ukončených paketů. Když jsou tyto toky připravená v jedné z výše zmíněných front, tak je vytvořena hlavička NetFlow v5 datagramu, do kterého jsou zároveň vloženy ukončené datové toky z fronty. Tím je UDP paket připraven k odeslání do kolektoru. Odeslání paketu zajišťuje systémové volání *send()*.

6.2.6 Shrnutí

Navržená implementace exporteru využívá mnoho technologií z oblastí operačních systémů a počítačových sítí. Jedná se o vícevláknovou aplikaci, která pro jejich synchronizaci využívá jednoduché zámky i složitější semafor. Zároveň zakládá na promyšleném paměťovém modelu, který se skládá z hashovací tabulky, dynamické oblasti pro ukládání záznamů o aktivních tocích, chronologicky uspořádané fronty a dvou front, které na principu double bufferingu akumulují dostatečný počet ukončených datových toků před vlastním sestavením a odesláním NetFlow v5 datagramu.

Navržená aplikace se spouští příkazem `netflow` a komunikuje s uživatelem pomocí parametrů zadaných z příkazové řádky. Nápověda aplikace vypadá takto:

```
Usage: netflow [options] remote:port ...

-h              Display this help
-p              Don't put the interface into promiscuous mode
-D              Demonize netflow process
-i <interface>  Network interface name
-f <expression> Filter expression (see tcpdump manual for details)
-d <id>         Engine identification [1]
-e <seconds>    Activ flow lifetime (activ timer) [300]
-n <number>     Flows stored in memory [16384]
-S <bytes>      Snaplen [1514]
-K <bytes>      Link layer header size [14]
remote:port     Address of the NetFlow collector [127.0.0.1:2222]
```

Z této nápovědy je patrná syntaxe jednotlivých argumentů, jejich význam a u některých položek je rovněž uvedená výchozí hodnota. Jediným povinným argumentem aplikace je „-i“ pro zadání jména síťového rozhraní.

6.3 NetFlow kolektor

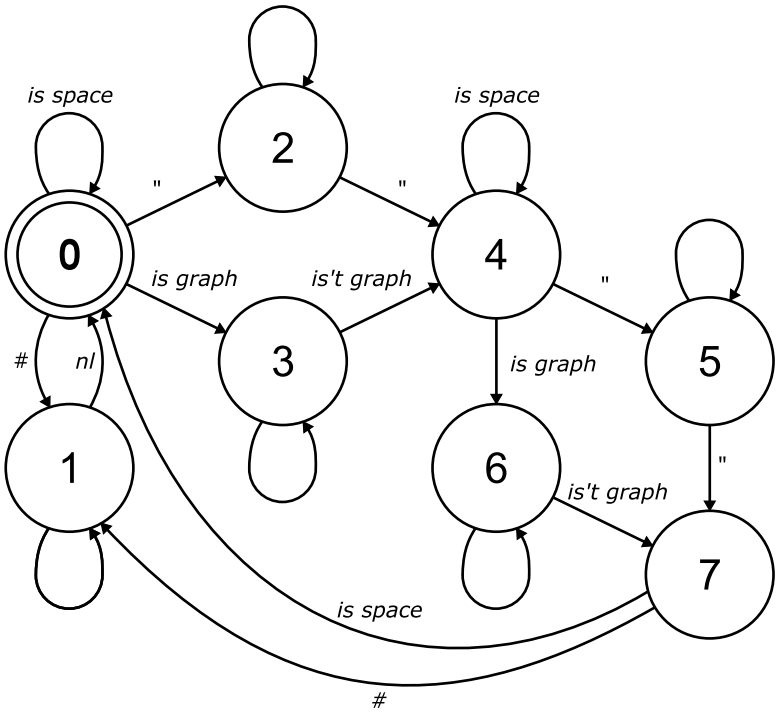
Oproti exporteru je kolektor znatelně jednodušší. Jde o síťového démona, který po svém startu načte konfigurační soubor. Na základě informací z tohoto souboru začne naslouchat na zadaném portu určitého síťového rozraní. Je-li přijat UDP datagram z některého privilegovaného exporteru, ověří se verze datagramu (NetFlow v5), a následně jsou z datagramu vybrány jednotlivé záznamy datových toků. Tyto záznamy jsou jednak komprimovány do souborů na pevném disku, jednak ukládány s další agregací do databáze. Informace uložené v souborech na pevném disku slouží k případnému dohledávání útočníka a obsahují všechny datové toky přijaté kolektorem. Databáze poskytuje informace o objemu přenesených dat z jednotlivých IP adres v lokální síti.

Tím byla vysvětlena zjednodušená funkce kolektoru. Následující kapitoly se věnují detailům jednotlivých částí této aplikace.

6.3.1 Čtení konfiguračního souboru

Všechna nastavení, která kolektor vyžaduje, se jako argumenty z příkavé řádky zadávají komplikovaně. Výhodnější řešení představuje konfigurační soubor. Jedná se obvykle o textový soubor, ve kterém jsou jednotlivé řídicí proměnné aplikace uvedené na řádcích. Jedinným úskalím využití konfiguračního souboru je načítání informací z tohoto souboru do proměnných. Tuto činnost zajišťuje lexikální analyzátor jehož činnost popisuje obrázek 6.6.

Symboly `'#'` a `'"'` v tomto obrázku představují načtení odpovídajícího ascii znaku. Symbol *nl* odpovídá načtení konce řádku (z angl. new line). Symbol *is space* představuje načtení „bílého“ znaku (např. mezery, tabelátoru nebo nového řádku). Symbol *is graph* představuje načtení tisknutelného znaku (např. písmena, číslice nebo speciálního znaku). Načtení netisknutelného znaku je představováno symbolem *is't graph*. Počátečním i koncovým stavem konečného stavového automatu je stav 0. Stav 1 zajišťuje přeskakování komentářů v souboru. Stavy 2 a 3 zajišťují načítání jména proměnné. Stav 4 přeskakuje bílé znaky mezi jménem proměnné a hodnotou, která je načítaná ve stavech 5 a 6. Stav 7 přečte zbytek řádku a přechází do stavu 0 tak, aby umožnil načtení dalšího řádku. Na závěr



Obrázek 6.6: Konečný stavový automat pro načítání konfiguračního souboru.

jsou načtené hodnoty uloženy do globální struktury, se kterou komunikují další části kolektoru. Jednotlivé konfigurační volby kolektoru nejlépe popisuje demonstrační konfigurační soubor:

```
# Specify where collector should store its flow records [1]
logdir   ”/upl/flows”

# Specify what addresses/ports collector should listen on. [0-1]
listen ”127.0.0.1:2222”
# listen ”any”
# listen ”ANY:2222”

# Accept flows from specific agents [0-n]
# eg. IPs of routers, that netflow probe use for sending UDP datagrams
agent ”127.0.0.1”
agent ”172.20.0.100”

# Nets and addresses of computers in local network [0-n]
# localnet ”10.0.0.0/8”
# localnet ”172.16.0.0/12”
# localnet ”192.168.0.0/16”

# Database information
dbhost ”127.0.0.1”  # Database host
dbname ”flows”      # Database name
dbuser ”flows”      # Database user
dbpass ”netflow”    # Database password

# Command to compress output log file
compresscmd ”bzip2 -z %s”
```

Položka *logdir* udává adresář, do kterého budou ukládány jednotlivé komprimované soubory s datovými toky (jedná se o povinnou položku). Položka *listen* určuje IP adresu a port, na kterých bude naslouchat serverová část kolektoru. Zároveň je možné využít místo IP adresy klíčové slovo „any“. Výchozí hodnotou je `127.0.0.1:2222`. Položka *agent* udává IP adresy exporterů, od kterých jsou akceptovány příchozí pakety. Těch je možné specifikovat několik. Aby bylo možné zaznamenávat pouze datové toky od vybraných uživatelů, je součástí konfiguračního souboru volba *localnet*. Ta určuje množinu IP adres a adres sítě. Datové toky, v nichž nefiguruje adresa počítače z této množiny, jsou automaticky zahozeny. Zároveň tento systém zahazuje ty datové toky, které monitorují lokální komunikaci (tj. komunikaci v rámci LAN). Položky začínající písmeny *db* slouží ke konfiguraci připojení k databázovému serveru, jejich význam je zřejmý. Nejzajímavějším parametrem je položka *compresscmd*, která udává externí příkaz, jehož pomocí je zkomprimován soubor s datovými toky. Důležité je aby, na místě názvu komprimovaného souboru, stál řetězec %s, ten je v programu nahrazen skutečným názvem souboru. Tím jsou volby, v konfiguračním souboru kolektoru, vysvětleny. Hlavní částí kolektoru je neblokující UDP server. Jeho popisu je věnována kapitola 6.3.2.

6.3.2 Neblokující UDP server

Využití neblokujícího serveru je pro potřeby kolektoru velmi důležité, protože pakety přijímané tímto serverem mohou přicházet velmi rychle a tím pádem by je blokující UDP server s velkou pravděpodobností nestačil zpracovat.

UDP server vyžaduje ke své činnosti především IP adresu a port. Tyto údaje získává z konfiguračního souboru (viz předchozí kapitola). Když adresa obsahuje doménové jméno, dojde k jeho převodu na IP adresu. Následně dojde k ověření čísla portu, které může být zadáno buď číslem nebo jménem služby, které je dále přeloženo na odpovídající číslo portu. Následuje vytvoření soketu pomocí systémové funkce *socket()*, ke kterému je posléze připojena získaná IP adresa a port. Připojení těchto údajů k soketu zajišťuje funkce *bind()*. Tím je kolektor připravený k přijímaní paketů. To je v případě UDP komunikace realizováno pomocí blokující funkci *recvfrom()*. Bezprostředně za touto funkcí následuje systémové volání *fork()*, které zajistí požadovanou funkčnost neblokujícího serveru. Význam této funkce spočívá ve vytvoření nového procesu (tzv. potomka), který následně zajišťuje obsluhu přijatého paketu. Rodičovský proces se přitom vrací zpět k funkci *recvfrom()*, kde čeká na přijetí dalšího paketu.

6.3.3 Zpracování přijatých paketů

Přijaté UDP datagramy mohou být odesílány z libovolné IP adresy, proto není možné zaručit, že přijatá data byla vygenerována konkrétním NetFlow exporterem. Tento problém je částečně odstraněn díky volbě *agent* z konfiguračního souboru. Ta umožňuje určit množinu IP adres exporterů, ze kterých jsou přijaté pakety akceptovány. Toto omezení ale uvedený problém příliš neřeší, neboť případný útočník může v odchozím paketu lehce změnit zdrojovou IP adresu. S tímto problémem je nutné počítat a snažit se jej omezit pomocí firewallu případně jiných technik, to však není cílem této diplomové práce.

Zpracování obsahu přijatého NetFlow v5 datagramu zajišťuje funkce *process_pkt()*. Ta nejprve rozkóduje hlavičku přijatého paketu, ze které zjistí počet záznamů o datových tocích uvnitř paketu (v jeho datové části). Následně prochází těmito záznamy a ukládá je ve specifickém interním formátu do binárního souboru.

Ukládání do souboru

Interní formát se skládá ze záznamů fixní délky. Každý záznam se skládá z několika položek. Význam většiny těchto položek odpovídá položkám záznamu NetFlow v5 (viz kapitola 3.4) s tím rozdílem, že položek je v tomto případě méně a každý záznam obsahuje úplné informace o datovém toku bez potřeby nahlížet do nějaké další hlavičky, jako v případě NetFlow v5. Největší výhodou tohoto binárního formátu je malá velikost výsledného souboru a jeho dobrý kompresní poměr. Záznam popisuje nejlépe jeho struktura:

```
struct mflow_record {
  u_long    srcaddr;
  u_long    dstaddr;
  u_long    dPkts;
  u_long    dOctets;
  u_long    First;
  u_long    Last;
  u_long    SysStart_secs;
  u_short   srcport;
  u_short   dstport;
  u_short   SysStart_msecs;
  u_char    engine_id;
  u_char    prot;
};
```

Sběr těchto záznamů probíhá po dobu jednoho dne (tedy 0:00:00 až 23:59:59) do souboru v adresáři, který je dán hodnotou položky *logdir* z konfiguračního souboru. Vytvořený soubor je pojmenován podle aktuálního datumu. Přitom je využit mezinárodní formát pro datum, tedy RRRR-MM-DD. Na konci dne (resp. s přijetím prvního paketu během následujícího dne) je soubor zkomprimován pomocí externího příkazu, který je zadán volbou *compresscmd* z konfiguračního souboru. Ukázkový konfigurační soubor je uveden v kapitole 6.3.1. V něm je zvolen komprimační algoritmus *bzip2*. Ten vykazuje ve většině případů nejlepší kompresní poměr. V další části systému je prozatím implementována podpora pouze pro tento algoritmus. Tímto postupem jsou záznamy o datových tocích ukládány do jednotlivých komprimovaných souborů v požadovaném adresáři.

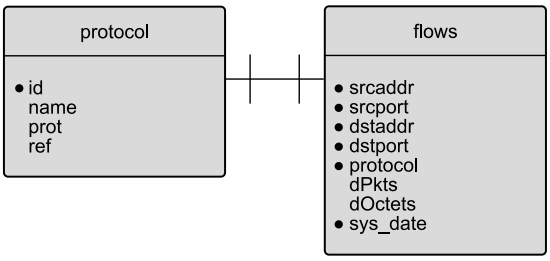
Ukládání do databáze

Ukládání záznamů do databáze má jiný účel než ukládání záznamů do souborů. Úkolem databáze je poskytovat informace o objemu přenesených dat dalším systémům (například informačnímu systému nebo systému zajišťujícímu *Fair Use Policy*). Z toho vyplývá, že do databáze nemusejí být vkládány veškeré informace o datových tocích, ale pouze taková data, která budou poskytovat informace o objemu přenesených dat za jednotlivé dny. Je tedy možné využít agregační funkci podobně jako v případě exporteru. Tuto agregaci je možné velmi efektivně implementovat přímo pomocí jazyka SQL nad databázovým systémem MySQL. Databázový subsystém navrženého systému je triviální, skládá se ze dvou tabulek tak, jak to ukazuje ER diagram na obrázku 6.7.

Tabulka *flows* obsahuje záznamy o jednotlivých datových tocích a tabulka *protocol* obsahuje informace o protokolech IP vrstvy. Kolektor využívá pouze tabulku *flows*, do které vkládá záznamy o datových tocích. Vložení jednoho záznamu zajišťuje SQL příkaz:

```
INSERT INTO flows VALUES(... ) ON DUPLICATE KEY UPDATE(... );
```

Z jeho konstrukce je patrné, že při shodě primárního klíče nedochází ke vložení nové položky, ale aktualizaci položky s tímto primárním klíčem. Při aktualizaci se inkrementuje čítač počtu přenesených paketů a objemu přenesených dat. Tímto je činnost kolektoru kompletní.



Obrázek 6.7: ER diagram databázového modelu.

6.3.4 Shrnutí

Navržená implementace kolektoru využívá, stejně jako exporter, několik technologií. Jedná se v první řadě o neblokující UDP server, který načítá a zpracovává NetFlow v5 datagramy. Dále o stavový konečný automat pro načítání konfiguračního souboru. Důležitou částí je zápis informací o datových tocích do binárního souboru a jeho následná komprese. Poslední činností kolektoru je sestavení SQL dotazu, který zajistí vložení záznamu do databáze nebo jeho agregaci.

Navržená aplikace se spouští příkazem `colector` a komunikuje s uživatelem pomocí parametrů zadaných z příkazové řádky. Nápověda aplikace vypadá takto:

```
Usage: collector [options] ...

-h              Display this help
-D              Demonize collector process
-f <file>       Specify config file [collector.conf].
```

Z této nápovědy je patrná syntaxe jednotlivých argumentů, jejich význam a u položky pro výběr konfiguračního souboru, je rovněž uveden název výchozího konfiguračního souboru.

6.4 Čtení záznamů z binárních souborů

Posledním programem, který je součástí navrhovaného systému, je program pro čtení záznamů z binárních souborů. Jedná se o jednoduchou aplikaci, která převádí záznamy o datových tocích z binárních souborů (viz kapitola 6.3.3) do podoby textových řetězců. Navržená aplikace se spouští příkazem `read-flows` a umožňuje uživateli vybrat formát, do kterého bude příslušný binární soubor převeden. Nápověda aplikace vypadá takto:

```
Usage: read-flows [options] <log_file>

-h              Display this help
-t <type>       Specify type of output
                0 - human readable format [default]
                1 - csv
```

Z této nápovědy je zřejmé, že formáty pro zobrazení binárního souboru jsou dva. První formát je podobný běžnému výpisu aplikace *tcpdump*. V případě druhého formátu se jedná o standardní CSV formát souboru, kdy jsou jednotlivé záznamy na samostatných řádcích a jednotlivé položky záznamu jsou odděleny pomocí středníků. Když je programu jako parametr předán komprimovaný soubor s příponou **.bz2*, tak je tento soubor automaticky dekomprimován a vypsán jeho obsah v požadovaném formátu.

Tímto jsou popsány všechny implementované části systému pro logování paketů na routerech.

Kapitola 7

Nasazení systému v cílovém prostředí

Během vývoje systému byly některé jeho části navrženy s ohledem na cílové prostředí, které je popsáno v kapitole 4. Proto je zajímavé pozorovat reálné chování navrženého systému v tomto prostředí. Zjištěné charakteristiky a vlastnosti systému jsou shrnuty v této kapitole.

Cílové prostředí představuje PC router s operačním systémem FreeBSD. Prvním krokem, který je nutné učinit před kompilací, je ověření existence potřebných nadstandardních knihoven. Těmi jsou knihovny *pcap* a *pthread*. Kompilaci jednotlivých částí systému zajišťuje program *make*. Po úspěšné kompilaci je nutné upravit konfigurační soubor kolektoru (výchozím konfiguračním souborem je *collector.conf*), otestovat funkčnost komprimačního programu (program *bzip2*), otestovat funkční připojení k MySQL databázi a ověřit oprávnění zápisu do adresáře, kam se budou ukládat binární soubory s datovými toky. Tím je systém připraven ke spuštění.

Při spuštění systému je důležité zachovat správné pořadí při spouštění jednotlivých částí systému: nejprve spustit kolektor (síťového démona), poté exporter. Při spouštění v opačném pořadí by NetFlow v5 datagramy z exporteru mohli dorazit dříve, než je spuštěn kolektor.

7.1 Výsledky měření

Na základě dvouměsíčního testování systému v cílovém prostředí se podařilo nejen ověřit modelové předpoklady, ale zároveň bylo zjištěno několik zajímavých vlastností cílového prostředí.

Při návrhu systému byl kladen důraz na objem zaznamenaných dat. Ten lze jednoduše ověřit pomocí příkazu *ls* pro binární soubory uložené na pevném disku a pomocí SQL dotazu pro záznamy uložené v databázi. Výpisu příkazu `ls -al` aplikovaného na adresář, kde jsou uložené záznamy o datových tocích:

```
...
-rw-r--r--  1 root  wheel   387253  8 kvě 23:59 2007-05-08.bz2
-rw-r--r--  1 root  wheel  2424938  9 kvě 23:55 2007-05-09.bz2
-rw-r--r--  1 root  wheel  1599284 10 kvě 23:57 2007-05-10.bz2
-rw-r--r--  1 root  wheel  1034731 11 kvě 23:59 2007-05-11.bz2
-rw-r--r--  1 root  wheel   358175 12 kvě 23:59 2007-05-12.bz2
-rw-r--r--  1 root  wheel   568984 13 kvě 23:59 2007-05-13.bz2
-rw-r--r--  1 root  wheel  1437123 14 kvě 23:58 2007-05-14.bz2
-rw-r--r--  1 root  wheel  1951457 15 kvě 23:58 2007-05-15.bz2
-rw-r--r--  1 root  wheel  2031480 16 kvě 11:32 2007-05-16
```

Velikost komprimovaných souborů se pohybuje v rozmezí 300 kB až 3 MB. Soubor, který náleží aktuálnímu dni ještě zkomprimovaný není a jeho velikost před zkomprimováním nikdy nepřesáhla

10 MB. Minimální disková kapacita pro uložení záznamů po dobu šesti měsíců zpětně, tak jak to vyžadují zákony uvedené v kapitole 2, se blíží hodnotě 550 MB. Při uvážení předpokládaného zvětšení šířky přenosového pásma pro připojení PC routeru komerční sítě do internetu ze 2 Mbsp na 8 Mbsp a současný nárůst komunikace uživatelů na čtyřnásobek, je zřejmé, že pevný disk o kapacitě 80 GB bude pro tuto činnost dostačovat.

Počet záznamů, které přibývají do tabulky flows, se pohybuje v rozmezí 2 000 až 20 000 za den. Přitom počet zaznamenaných datových toků je mnohem vyšší hlavně díky provedené agregaci pomocí SQL konstrukce *INSERT OR UPDATE*. Celkový počet paketů a objem přenesených dat ukazuje tabulka 7.1.

datum	dPkts	dOctets
2007-05-08	1207314	958143860
2007-05-09	10493185	8848999125
2007-05-10	11978999	10151793954
2007-05-11	4876836	3660717065
2007-05-12	849463	537280941
2007-05-13	2859817	2155298412
2007-05-14	5786964	4422983166
2007-05-15	7323913	5821729996
2007-05-16	4146896	3152202853

Tabulka 7.1: Počet paketů a objem přenesených dat v jednotlivých dnech.

Informace uložené v databázi představují velmi dobrý zdroj statistických dat. Tato data určitým způsobem popisují chování uživatelů v lokální síti. Na základě chování uživatelů je možné tyto uživatele zařadit do určitých typových skupin, které představují určité riziko pro danou síť. Chování uživatelů komerční sítě na kolejích VUT v Brně popisují následující tabulky. Řetězec *xxx* v těchto tabulkách skrývá poslední část IP adresy koncové stanice z důvodu ochrany osobních údajů.

IP adresa	dns	dPkts	dOctets
195.250.146.*xxx*	*xxx*.vol.cz	3330118	302192135
147.229.10.31	video1.fit.vutbr.cz	2105021	188182388
147.229.10.32	video2.fit.vutbr.cz	1730296	154249518
195.113.116.81	amp1.cesnet.cz	1306438	115181993

Tabulka 7.2: Nejčastěji adresované počítače podle počtu odeslaných paketů.

IP adresa	dns	dPkts	dOctets
195.178.73.170	aladin.mendelu.cz	626785	558168160
147.229.18.12	kazi.fit.vutbr.cz	694319	468457207
68.194.112.*xxx*	*xxx*.optonline.net	702090	342127737
195.250.146.*xxx*	*xxx*.vol.cz	3330118	302192135

Tabulka 7.3: Nejčastěji adresované počítače podle objemu dat.

Z těchto tabulek vyplývá, že komerční síť využívá početná skupina osob se vztahem k FIT VUT v Brně a MZLU v Brně. Zároveň je z nich možné vyčíst, že uživatelé komerční sítě ve větší míře přijímají vysílání rádiových stanic ze serveru *amp1.cesnet.cz*.

Nejdůležitější funkcí databáze je poskytovat přesné údaje o objemu dat, který byl přenesen jednotlivými uživateli lokální sítě. Následující tabulka ukazuje ty IP adresy z lokální sítě, které přijali nejvíce dat. V této tabulce mohou být uvedeny IP adresy jednotlivých uživatelů, neboť se jedná o lokální IP adresy jejichž vypsáním není nijak porušen Zákon o ochraně osobních údajů.

IP adresa	dPkts	dOctets
10.229.221.33	7549527	10816235833
10.229.193.33	8181246	9160282798
10.229.197.17	6462150	8324063697
10.229.197.16	5550651	6044621116
10.229.206.32	3724710	4531197410

Tabulka 7.4: Lokální počítače podle objemu přijatých dat.

Jak vyplývá z výše uvedených tabulek, databáze může poskytovat mnoho zajímavých statistických informací, záleží jen na vhodně položeném SQL dotazu. Příkladem může být informace, že 98,7 % všech paketů využívá protokol TCP, pouhé 1 % UDP, ostatním protokolům náleží 0,3 %, nejsou tedy téměr využívány.

7.2 Výkonnost systému

Z pohledu PC routeru, na kterém je navržený systém provozován, je důležité kolik systémových prostředků běh systému spotřebovává. Exporter vyžaduje ke svému běhu 3588 KB paměti. Jelikož nedochází k žádné dynamické alokaci paměti, je tato hodnota neměnná. Kolektor využívá 1952 KB paměti, ale nutno podoknout, že při zpracovávání většího počtu NetFlow v5 datagramů dochází k nárůstu této hodnoty. Z pohledu strojového času procesoru nepředstavuje žádná z aplikací navrženého systému větší zátěž, což je dáno především pomalým připojením routeru k internetu a malým počtem uživatelů v síti. Testování výkonnosti systému v extrémních případech není v současné chvíli možné na PC routeru provést, neboť na něm dočasně běží dohledový systém nad aktivními prvky, který využívá téměř 50 procent výkonu procesoru. Po jeho migraci na nový server, budou otestovány i tyto vlastnosti navrženého systému.

Z výsledků měření vyplynulo, že největším probémem při provozu logovacího systému je tzv. skenování portů, neboť tato činnost v sobě skrývá vyslání několika tisíc paketů na různé porty cílového systému. Tím pádem logovací systém zaznamená několik tisíc nových datových toků, které musí určitou dobu sledovat a poté vyexportovat do kolektoru. To představuje velkou zátěž jak pro komunikační subsystém, tak pro samotný kolektor. Nadruhou stranu tato činnost představuje dobrý způsob pro testování výkonnosti navrženého systému.

Kapitola 8

Závěr

Zadání diplomové práce v sobě spojuje několik oblastí informačních technologií. Zahrnuje problematiku teorie lokálních sítí, operačních systémů, databází a počítačových systémů. Z tohoto pohledu vytváří prostor pro široké působení a tvůrčí činnost. Na druhou stranu klade velké nároky na srozumitelnost textu, který musí být, i při velkém objemu zpracovaných informací, dobře čtivý.

Úvodní část dokumentu shrnuje podstatné aspekty současných právních omezení v oblasti informačních technologií. Následně popisuje možné právní problémy, které mohou vyvstat při využívání logovacího systému na routerech. Bylo zjištěno, že provozovatel routeru neodpovídá za obsah přenášených dat. Dále bylo zjištěno, že informace zaznamenávané logovacím systémem nemají povahu osobních dat v případech, kdy neexistuje přímá vazba mezi určitou osobou a zaznamenanou informací. Nejdůležitějším zjištěním je fakt, že každý poskytovatel komunikační služby musí zaznamenávat informace o veškeré komunikaci připojených uživatelů a to po dobu šesti měsíců zpětně. V případě počítačové komunikace zákon přímo definuje, které informace z hlaviček paketů je nutné uchovávat, čímž nepřímo vyžaduje nasazení systému pro logování paketů. Posledním důležitým zjištěním je fakt, že zaznamenaný log nemá při soudním procesu povahu důkazu na jehož základě by mohl být někdo obviněn, ale jistou vypovídací hodnotu má. To je způsobeno hlavně tím, že každý log lze lehce podvrhnout nebo zfalšovat.

Hlavním cílem diplomové práce bylo prostudovat problematiku logování údajů o průchozích paketech v routerech a navrhnou systém, který by tuto činnost zajišťoval. Důležité informace z hlediska logování paketů jsou shrnuty ve třetí kapitole. Jsou zde vysvětleny základní principy počítačové komunikace a logování. Byly definovány zdroje problémů při logování, které jsou důležité především pro návrh systému.

Prostředí, pro které je systém primárně navrhován, není jednoduché. Proto je jeho rozboru věnována čtvrtá část dokumentu. Z tohoto rozboru vyplývá, že PC router, pro který je systém navrhován, využívá technologii překladu síťových adres, a proto není možné logovat pakety na vnějším síťovém rozhraní tohoto routeru (v hlavičkách paketů je na jeho vnějším rozhraní již přeložená IP adresa). Další komplikaci představují virtuální sítě. Jelikož má PC router pouze dvě síťové karty, jsou všechna vnitřní rozhraní routeru přenášena přes jednoho fyzické rozhraní (jednu síťovou kartu) pomocí technologie virtuálních sítí (*VLAN*). Rozlišení, do které sítě přenášený paket náleží, zajišťuje operační systém automaticky na základě tagu virtuální sítě.

Při vlastním návrhu systému byl kladen důraz na minimalizaci objemu zaznamenaných dat při zachování vysoké výkonnosti systému a odpovídající paměťové náročnosti. Z těchto důvodů byl systém při návrhu rozdělen na několik nezávislých částí: exporter, kolektor a databázi. První částí je kolektor. Jedná se o aplikaci, která sbírá pakety ze síťového rozhraní. Z těchto paketů získává informace o datových tocích a o objemu přenášených dat. Každý datový tok je sledován určitou dobu,

během které jsou inkrementována počítadla objemu přenesených dat a počtu paketů. Po vypršení této doby jsou nasbírané informace o datových tocích předávány pomocí NetFlow v5 datagramů do kolektoru. Kolektor představuje druhou část systému. Jedná se o síťového démona, který přijímá UDP datagramy neblokujícím způsobem. Každý přijatý datagram obsahuje informace o několika datových tocích, které po zpracování ukládá jednak do databáze, jednak do binárních souborů na pevném disku, které následně komprimuje.

Navržený systém byl implementován a nasazen v cílovém prostředí. Na základě dvouměsíčního testování v tomto prostředí, kdy nedošlo k žádným neočekávaným potížím systému, byl systém pro logování paketů na routerech shledán stabilním. Jelikož cílové prostředí nepředstavuje významnou zátěž pro systém, je dalším cílem otestování tohoto systému v prostřední páteřní sítě KolejNet. Mezi případná rozšíření, na která se v rámci této práce nedostalo je implementace části systému, která zajišťuje konzistenci záznamů v databázi a mazání již nepotřebných souborů s datovými toky. Tuto část systému je sice možné nahradit občasným ručním promazáním starých souborů a záznamů v databázi, nicméně z dlouhodobého hlediska se automatizace v této oblasti stává nepostradatelnou.

Literatura

[1] Spáčilová Danuše: *Právo v informačních systémech*.
Skripta; Brno, katedra právní teorie Prf MU Brno, 2005.

[2] Alena Kabelová, Libor Dostálek: *Velký průvodce protokoly TCP/IP a systémem DNS*.
ISBN: 80-7226-849-X; Brno, nakladatelství Computer Press, 2002.

[3] Rita Pužmanová, Pavel Šmrha: *Propojování sítí s TCP/IP*.
ISBN: 80-7232-080-7; České Budějovice, nakladatelství Protisk, 1999.

[4] FlowMon: *Web věnovaný specializované HW kartě pro měření datových toků*.
`http://www.flowmon.org` (prosinec 2006)

[5] Cisco: *Projektová dokumentace firmy Cisco Systems*.
`http://www.cisco.com/univercd` (prosinec 2006)

[6] Lupa: *Server o českém internetu. Série článků o legislativě v IT*.
`http://www.lupa.cz/r/legislativa` (prosinec 2006)

[7] IT právo: *Server o internetovém a počítačovém právu*.
`http://www.itpravo.cz` (prosinec 2006)

[8] tcpdump: *Stránky věnované projektu tcpdump a knihovně pcap*.
`http://www.tcpdump.org` (prosinec 2006)

[9] fprobe: *Domovská stránka projektu fprobe*.
`http://fprobe.sourceforge.net` (prosinec 2006)

[10] ethereal: *Web o paketovém analyzátoru*.
`http://www.ethereal.com` (prosinec 2006)

[11] Lucas Michael: *FreeBSD Podrobný průvodce*.
ISBN: 80-7226-795-7; Brno, nakladatelství Computer Press, 2003.

[12] man: *Manuálové stránky projektu FreeBSD*.
`http://www.freebsd.org/cgi/man.cgi` (prosinec 2006)

[13] PF firewall: *Stránky věnované paketovému filtru*.
`http://www.openbsd.org/faq/pf` (prosinec 2006)

[14] IEEE 802.1Q: *Standard definující virtuální sítě VLAN*.
`http://standards.ieee.org/getieee802/download/802.1Q-2005.pdf`
(prosinec 2006)

Seznam použitých zkratek a cizích slov

API(Application Programming Interface) – rozhraní pro programování aplikací
IDS(Intrusion Detection System) – systém zajišťující detekci a řešení bezpečnostního incidentu v síti
IP(Internet Protocol) – protokol zabezpečující adresování v internetu
IPS(Intrusion Prevention System) – systém zajišťující předcházení bezpečnostnímu incidentu v síti
ISO/OSI – vrstvový model, na kterém staví většina aplikací komunikující po síti
ISP(Internet Service Provider) – poskytovatel internetu
IT – informační technologie
IT právo – právo na poli informačních technologií
KolejNet – studentská kolejní síť VUT
LAN(Local Area Network) – síť v rámci jednoho areálu
Log – chronologický záznam o určité činnosti
Logování – chronologicky zaznamenávání sekvenčních dat
NAT(Network Adress Translation) – služba využívaná na routerech k překladu IP adres
NIC(Network Interface Controller) – hardware v PC zajišťující síťovou komunikaci
PAT(Port Adress Translation) – služba využívaná na routerech k překladu IP adres
PC – počítačový systém [(počítač)]
PCI(Peripheral Component Interconnect) – sběrnice v PC pro připojení rozšiřujících karet
RFC(Request For Comments) – dokument využívaný pro popis internetových standardů
TCP(Transmission Control Protocol) – protokol zajišťující zabezpečený přenos
UDP(User Datagram Protocol) – protokol zajišťující rychlý nezabezpečený přenos
VLAN(Virtual LAN) – technologie umožňující logické dělení LAN na menší celky