VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

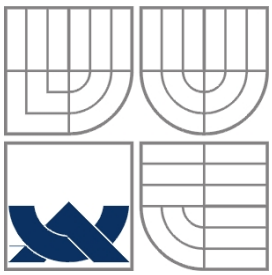
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

DETEKCE P2P SÍTÍ

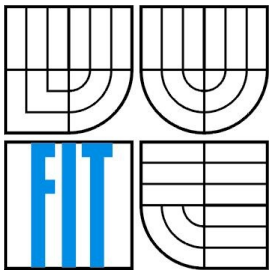
DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE Bc. Matej Březina
AUTHOR

BRNO 2008



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

DETEKCE P2P SÍTÍ

DETECTION OF P2P NETWORKS

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. Matej Březina

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. Jiří Tobola

BRNO 2008

Abstrakt

Táto práca sa zaoberá návrhom, implementáciou a testovaním softwarového detekčného systému p2p (peer-to-peer) sietí založeného na kombinácii predfiltrovania pomocou BPF a porovnávania obsahu paketov so vzormi obsahu známych p2p komunikácií POSIX regulárnymi výrazmi. Súčasťou systému je aj databáza pravidiel niektorých rozšírených p2p protokolov vo formáte veľmi podobnom definíciám pre klasifikátor L7-filter. Aplikácia je implementovaná v C, beží v userspace a je cielená na všetky POSIX kompatibilné platformy. Kombinácia detektora s užívateľsky pripojeným riadením QoS je kompletným riešením pre obmedzenie premávky známych p2p protokolov.

Kľúčové slová

distribúcia obsahu, p2p, peer-to-peer, detektor p2p premávky, inšpekcia paketov, porovnávanie vzorov, BPF, NetFlow, L7-filter, pcap

Abstract

This document deals with design, implementation and testing of software system for detecting p2p (peer-to-peer) networks based on combination of BPF prefiltering and POSIX regular expressions packet payload matching with known p2p protocol communications. The proposed detection system includes a database with some rules of most effuse p2p protocols in format resembling to definitions for L7-filter classifier. The application is implemented in C, runs in userspace and is targeted to all POSIX compatible platforms. The combination of detector with user attached QoS controlling is complete solution for traffic reduction of common p2p protocols.

Keywords

content distribution, p2p, peer-to-peer, p2p traffic detection, packet inspection, pattern matching, BPF, NetFlow, L7-filter, pcap

Citácia

Březina Matej: Detekce p2p sítí. Brno, 2008, diplomová práce, FIT VUT v Brně.

Detekce p2p sítí

Prehlásenie

Prehlasujem, že som túto diplomovú prácu vypracoval samostatne pod vedením Ing. Jiřího Toboly.
Ďalšie informácie som získaval prevažne z množstva zdrojov na internete.
Uviedol som všetky literárne pramene a publikácie, z ktorých som čerpal.

.....
Matej Březina
19.5.2008

Poďakovanie

Na tomto mieste by som sa chcel poďakovať vedúcemu práce, Ing. Jiřímu Tobolovi, za cenné rady i poskytnuté anonymizované vzorky dát z reálnej vysokorýchlostnej p2p premávky.

Obsah

1 Úvod

Príklon od tradičného klient-server modelu k distribuovaným peer-to-peer (ďalej len „p2p“) sieťam je v architektúre internetu logickým krokom pre zlepšenie priepustnosti a zvýšenie spoľahlivosti (dostupnosti) prenosu dát. Tieto siete sú primárne navrhnuté z dôvodu efektívneho zdieľania zdrojov (výpočetný výkon, úložný priestor) lepším využitím prenosovej sústavy, teda priamym spôsobom výmeny dát. Ďalšou výhodou je možnosť fungovania prenosov aj bez centrálneho servera / nódu, ktorá je základným predpokladom schopnosti úspešne prekonávať výpadky ktoréhokoľvek účastníka – a aby bolo toto možné, účastníci v sieti p2p plnia zároveň funkciu servera aj klienta.

Masové rozšírenie p2p architektúry okrem nesporných výhod prináša rad určitých problémov, ktoré môže čitateľ nájsť v druhej kapitole. Z nich vyplýva potreba rozlišovať toky p2p a ostatné toky, s tým, že p2p prenosy majú obvykle (a najmä v prípade p2p sietí pre zdieľanie súborov) nižšiu prioritu. Úplné blokovanie p2p premávky je nevhodným riešením keďže je nemožné označiť so 100 % istotou ľubovoľný paket, tok či dokonca klientskú adresu ako participanta p2p prenosu. Implementácie p2p sietí totiž vždy budú o krok napred pred ich detektormi.

Detekcia sa však stáva čoraz problematickejšou – a to nielen kvôli rastúcemu množstvu dát prenášaných p2p ale aj vďaka anonymizáciii a šifrovaniu dát p2p klientov.

Zlepšovanie algoritmov detekcie postupuje v zásade dvoma smermi. Jedna skupina detektorov stavia na jednoznačnú detekciu (regulárne výrazy či stromové štruktúry), druhá využíva prvky umelej inteligencie pri analýze dát a metainformácií. Časť tejto problematiky možno nájsť popísanú v tretej kapitole.

Aj implementácie detektorov sú v neustálom vývoji. Pre úspešné nasadenie detektora na väčšiu sieť je nevyhnutná značná paralelizácia, dosiahnutá ASIC (Application-Specific Integrated Circuit) či FPGA (Field Programmable Gate Array) obvodmi. Menšie siete vyžadujú väčšiu flexibilitu, no postačuje im nižší celkový výpočetný výkon univerzálnych procesorov – a tu sa uplatní aj bežná PC platforma. Práve návrh a testovanie takéhoto systému s jednoznačným modelom detekcie (ktorý bol rozobraný v semestrálnej práci) a s dôrazom na výkon je predmetom ďalších kapitol.

Na siete typu peer-to-peer sa bude autor v práci odkazovať zjednodušene ako na „siete p2p“. „Účastníkmi", „klientmi", „peermi“ prípadne „nódmi" bude podľa vhodného kontextu ďalej nazývať uzly či stanice, participujúce na p2p prenosoch ako ich riadni členovia.

1.1 Motivácia

Autor by chcel vyjadriť presvedčenie, že detekcia p2p sietí a ich shaping povedie, paradoxne, práve k ich rozšíreniu, keďže identifikáciou a QoS / prioritizáciou tokov umožní prevádzku p2p sietí aj v prostrediach, kde doteraz bolo p2p nežiaduce a práve za týmto účelom je toto dielo vytvorené.

2 Technológie p2p sietí

2.1 Rozdelenie p2p sietí

Rozdeľme si siete podľa typu dát, ktoré prenášajú. Od tohto sa bude odvíjať ich dôležitosť z pohľadu sieťového administrátora ako aj parametre pre ich úspešnú detekciu.

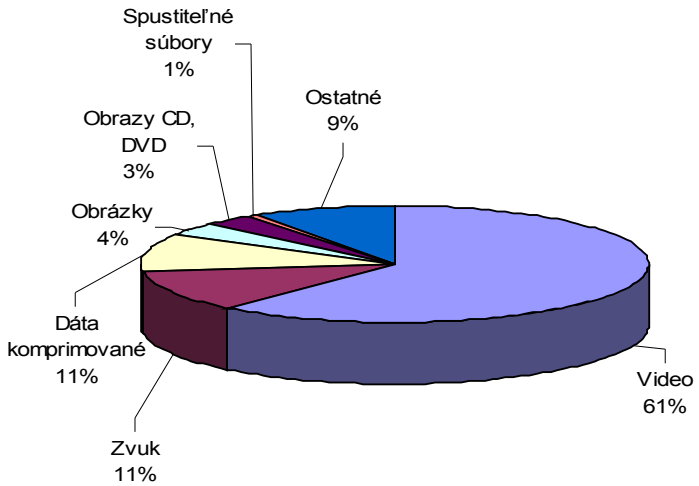
Mechanizmy detekcie sú však pre všetky p2p siete jednej triedy v zásade rovnaké, keďže princíp ich funkcie je tiež veľmi podobný. Existuje veľa rôznych variánt, niektoré siete by sa dali zaradiť aj do viacerých kategórií a tak ich zadeľovanie nie je presné.

2.1.1 Rozdelenie podľa účelu

2.1.1.1 Siete pre distribuovanie obsahu

Sú nazývané aj p2p sieťami pre zdieľanie súborov. Sú veľmi rozšírené, využívajú obvykle trvale maximálnu šírku pásma, v prípade nelegálnej ponuky obsahu môžu porušovať autorské zákony a z týchto dôvodov sú spomedzi ostatných typov sietí sú asi najkontroverznejšie, preto bývajú aj najčastejším terčom detekcie. Existuje však len niekoľko úspešných implementácií, ako napríklad BitTorrent, Gnutella, eDonkey, Kazaa a Direct Connect. Detekcii niektorých z nich, u nás najviac rozšíreným (BitTorrent, Direct Connect), sa autor venuje ďalej.

Pri konštrukcii detektora p2p môže byť vhodné vedieť približné rozloženie premávky p2p podľa formátu prenášaných súborov:



Obrázok 2.1: Rozloženie premávky p2p podľa formátu súborov.

Jednoznačne, najväčší podiel má multimediálny obsah. To znamená, že väčšina p2p sietí je optimalizovaných pre veľké súbory a je to dosť dôležitým poznatkom pri ich detekcii.

2.1.1.2 Komunikačné siete

Do tejto kategórie patria systémy, ktoré poskytujú infraštruktúru pre priamu komunikáciu a spoluprácu účastníkov, obvykle v reálnom čase. Patria sem rôzne textové komunikátory (instant-messaging) a VoIP aplikácie, typickým príslušníkom je Skype. Požiadavky na prenosovú kapacitu rovnako ako aj celkový objem prenesených dát je v tomto prípade nižší ako majú napríklad systémy pre zdieľanie súborov, maximálna šírka pásma sa využíva len zriedka a aj z právneho hľadiska sú tieto systémy bezproblémové, obvykle nenastáva dôvod na detekciu takýchto sietí.

2.1.1.3 Siete distribuovaných výpočtov a databázových systémov

Rozľahlé distribuované siete, obvykle s jedným či viacerými centrálnymi uzlami (hierarchické) využívajú zdroje klienta (obvykle procesorový čas). Podobne ako komunikačné siete, generujú obvykle nižšiu a veľmi sporadickú premávku a ich pôsobenie v globále môžeme označiť ako malé. Príkladom môže byť Seti@home.

Podobne pracujú i databázové p2p siete, ktoré rozkladajú záťaž dotazov na jednotlivých klientov. Napríklad systém Piazza môže tvoriť infraštruktúru pre veľa aplikácií pre sémantický web.

2.1.1.4 Ďalšie podporné siete

Existujú siete pre podporu rôznych, na prenosovú kapacitu náročných aplikácií, napríklad p2p multicastové systémy, distribuované systémy na ochranu proti DoS (Denial of Service) útokom alebo šíreniu vírusov; ich pôsobenie je však prevádzkovateľom obvykle známe a žiaduce, preto ich detekcia nemá zmysel.

2.1.2 Rozdelenie podľa sieťovej štruktúry

Z predchádzajúceho rozdelenia vyplýva, že najproblematickejšie a teda aj najviac sledované p2p siete sú siete pre distribúciu obsahu a autor sa ďalej bude primárne venovať práve im.

Ak by sme chceli navrhnúť dobrý detektor p2p sietí, potrebujeme vedieť ako a ktoré uzly medzi sebou komunikujú, teda aká je vlastne štruktúra siete, ďalej čo je predmetom komunikácie (typický sled paketov a ich štruktúru pri rôznych typoch komunikácie) a informáciu o približnom množstve prenášaných informácií.

SIETE	CENTRALIZÁCIA ARCHITEKTÚRY		
	Hybridná	**Čiastočná**	**Bez centralizácie**
Neštruktúrované	*Napster, Direct Connect, Audiogalaxy, WinMX*	*BitTorrent, Gnutella, Morpheus, Kazaa*	*Gnutella*
Štruktúrované infraštruktúry			DHT implementácie *Chord, CAN, Tapestry, Pastry, BitTorrent* v móde bez trackerov
Štruktúrované systémy			na Tapestry bežiaci *Mnemosyne,* na Pastry bežiaci *PAST*

Tabuľka 2.2: Rozdelenie vybraných p2p sietí podľa úrovne centralizácie a štruktúrovanosti [1].

Hybridne centralizované siete využívajú jeden centrálny server (prípadne cluster), ktorý tvorí zoznamy a uchováva indexy obsahu všetkých p2p klientov. Táto architektúra je značne obmedzená spoľahlivosťou servera a kapacitou jeho konektivity.

Čiastočne centralizované siete sú postavené na schopnosti niektorých nódov (nazývaných aj *supernódy*) pracovať nad informáciami z okolitých nódov, riadiť tok dát, a tvoriť miestne indexy (*trackery* v prípade BitTorrent, *ultrapeers* v sieti Gnutella). Tieto supernódy môžu byť staticky priradené, ale môžu vznikať aj dynamicky - záleží od konkrétnej implementácie siete).

Siete bez centralizácie sa skladajú z rovnocenných nódov majúcich rovnaké úlohy, ktoré slúžia zároveň ako servery aj klienti a všetka komunikácia prebieha len so susednými, hierarchicky ekvivalentnými nódmi (napríklad záplavový systém vyhľadávania v sieti Gnutella).

Štruktúrované siete sa od **neštruktúrovaných sietí** líšia vo fakte, že umiestnenie ľubovoľného obsahu (súboru) záleží na jeho identifikátore a teda mapujú obsah (presnejšie jeho ID) na konkrétne miesto v topológii siete.

2.2 Rozvoj p2p sietí a možnosti ich detekcie

Popularita p2p sietí sa od ich počiatku stále zvyšuje, najväčšie p2p siete majú dnes bežne rádovo milióny užívateľov a spolu s ňou rastie aj komplexnosť - prechádza sa od jednoduchých neštruktúrovaných modelov typu Gnutella k zložitejším modelom s využitím DHT (distribuovanej hash tabuľky) až po anonymizované siete, nódy môžu plniť niekoľko rôznych funkcií a detekcia je z tohto pohľadu problematickejšia.

2.2.1 Prvá generácia p2p sietí – hybridne centralizované a čiastočne centralizované

Napster, označovaný ako prvá úspešná p2p sieť pre zdieľanie súborov, začal síce ponúkať svoje služby až v roku 1999; aj keď prvé kroky k distribuovanej výmene dát sledujeme už z čias Usenetových diskusných skupín, keď klienti si navzájom vedeli preposielať nové príspevky.

Podobne ako klony jeho funkcionality (Audiogalaxy, Soulseek, Direct Connect, Scour Exchange, WinMX) pracuje s asistenciou centralizovaného uzla. Detekcia jeho prítomnosti je jednoduchá, stačí detekovať komunikáciu klienta so známou adresou - aktualizované zoznamy adries (hublisty v prípade DC) sú pri tom vhodnou pomôckou. Vzhľadom na to že ide o staršie protokoly s jednoduchou štruktúrou, možno v komunikácii skoro vždy rozpoznať aj známe reťazce.

Z tejto generácie sa používa už len Soulseek - a aj ten skôr „zo zvyku“, pre uzavreté, aj keď voľné algoritmy vyhľadávania a nepodporu swarmingu (sťahovanie jedného súboru z viacerých zdrojov) - vlastnosti ktorá sa bežne objavuje aj v tejto generácii sietí.

Direct Connect (DC) má stále vysoké zastúpenie, prevažne v menších komunitách, pre veľké množstvo dostupných implementácií serverov (tzv. hubov) - VerliHub, PtokaX, openDCHub a i. a rozbehnúť DC server vyžaduje len minimálne odborné schopnosti. Huby sú medzi sebou väčšinou nezávislé nerátajúc failover a load balancing – pri preťažení či poruche jedného hubu sú užívatelia presmerovaní na iný. Táto vlastnosť býva občas zneužívaná pre DDoS útoky, dnešné klientské programy túto činnosť dokážu potlačiť obmedzovaním presmerovaní ako aj napríklad podozrievavosťou k neznámym IP nefigurujúcim v hublistoch atď. – teda je možno uvedomiť si často skrytý potenciál už tejto generácie p2p sietí.

2.2.2 Druhá generácia p2p sietí – decentralizované

Vývoj týchto sietí bol urýchlený rastúcim záujmom o p2p siete, ktoré narážali na kapacitné problémy indexovacích serverov aj právnymi postihmi Napsteru. Priekopníkom pravej decentralizácie bola Gnutella, ktorá sa paradoxne tiež potýkala s kapacitnými problémami vlastných klientov, keďže tento jednoduchý model považoval všetky uzly za rovnocenné. Ukázalo sa, že uzly treba v čase váhovať podľa ich kapacitných možností a včasné implementovanie zmeny umožnilo jej masové rozšírenie. Podobné modely využíva aj FastTrack a eMule.

Rastúce množstvo dát a ich indexov bolo vyriešené distribuovanou hash tabuľkou (Distributed Hash Table, DHT) čo znemožnilo detekciu komunikácie na základe známych adries, avšak tým, že táto premávka bola presunutá medzi jednotlivé uzly, typické dotazy announce/request zase umožnili jednoduchšiu detekciu p2p premávky medzi participujúcimi uzlami.

Snáď najrozšírenejšie (ťažko merateľná kategória) p2p siete BitTorrent, eDonkey spolu s Gnutellou sú tiež z tejto generácie. eDonkey (niekedy nazývané aj eDonkey2000 alebo eD2k), pôvodne staršia sieť prvej generácie s centralizovanými servermi, stavilo na zväčšovanie počtu serverov (nová sieť Overnet ako následník starej eDonkey) a voľné servery sú k dispozícii (aj keď s uzavretými zdrojovými kódmi). Pohltením FastTracku, a spolu s vlastnou implementáciou Kademlie je táto veľká (variáciami i rozšírením) množina pôvodom príbuzných protokolov nazývaná spoločne eDonkey.

Aj BitTorrent je predstaviteľ tejto generácie. Zbytočnosť centralizovaných uzlov bola dosiahnutá swarmingom (prepojením záujemcov – tzv. leecherov a čistých poskytovateľov – tzv. seederov) okolo súboru s metainformáciami o najvyššej prenosovej jednotke (torrentu). Aj keď BitTorrent rátal s asistenciou trackerov (serverov, ktorý uchováva a poskytuje aktuálne informácie o ďalších klientoch swarmu okolo torrentu), dnes využíva viacero implementovaných DHT sietí (napríklad známy klient Azureus má vlastnú), čo sa nazýva trackerless mode.

Z hľadiska detekcie však ide stále o ten istý princíp – komunikácia s rovnocennými, prípadne lokálnymi, dočasnými nadradenými uzlami a výmena zoznamu zdrojov so susedmi.

Niektorí klienti týchto sietí, napr. Azureus ako klient BitTorrentu, majú aj schopnosti ako Protocol header encrypt (PHE) a Message stream encryption / Protocol encryption (MSE/PE) čím posúvajú tieto siete do úrovne dvaajpóltej generácie.

2.2.3 Tretia generácia p2p sietí – anonymné a šifrované

Rozvoj rýchlych detekovacích implementácií a najmä obava o nedostatočné utajenie prenášaných informácií (najmä hashov) viedli k vývoju p2p protokolov snažiacich sa o anonymizáciu a šifrovanie prenášaných informácií. Prenos medzi dvomi uzlami je v skutočnosti vedený cez iný, tretí uzol ktorý má v tejto situácii funkciu virtuálneho uzla siete. Klientský software užívateľa pozná len adresu partnera s ktorým komunikuje dočasne stanovenú v rámci siete., avšak skutočná adresa partnera nie je nikdy známa, čo umožňuje anonymizované prenosy.

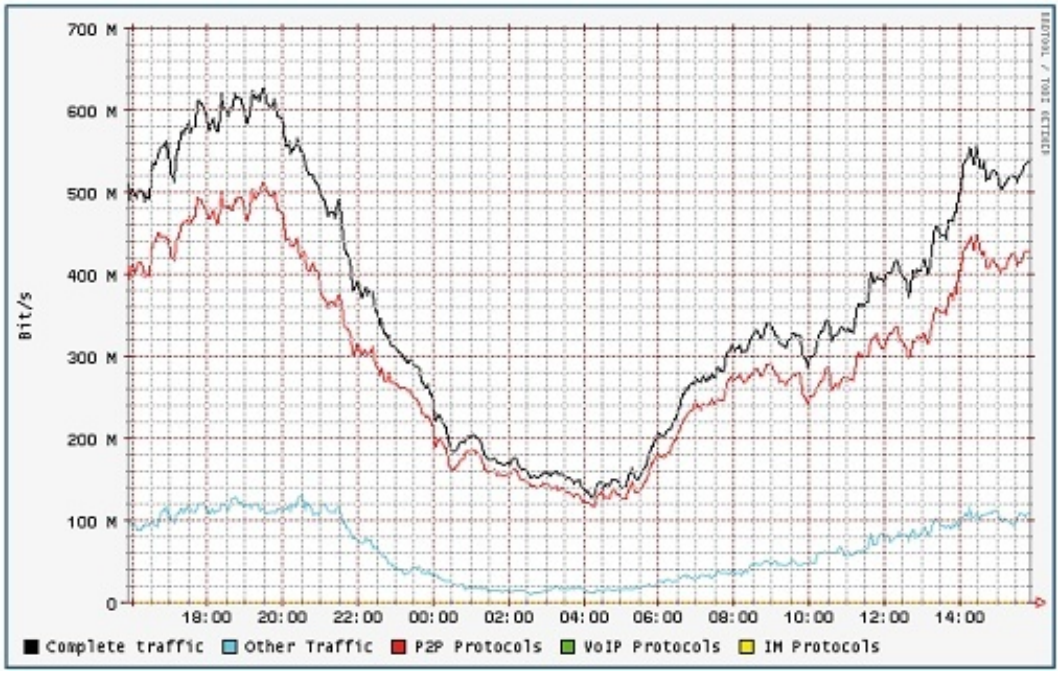
Variantom tejto architektúry sú f2f (friend-to-friend) siete medzi známymi a overenými uzlami (napríklad aj pomocou certifikátov), čo vylučuje možnosť priamej komunikácie s neznámymi účastníkmi.

Problémom všetkých týchto implementácií je neúmerné množstvo prenosov a prenášaných dát; napriek tomu, klasická detekcia na základe obsahu paketov vďaka šifrovacím algoritmom zlyháva. Výnimku tvoria snáď len triviálne substitučné šifry so známym kľúčom (napríklad rotačné šifry ako Caesar), ktoré sa, samozrejme, v týchto aplikáciách pre ich vlastnosti vôbec nepoužívajú. Aj prípadná hlavička, resp. malá, nešifrovaná časť dát je obvykle príliš krátka na to aby boli rozpoznateľné.

2.3 Rozšírenie a nároky na prenosovú kapacitu

Rýchly rozvoj p2p sietí nastal hlavne po roku 1999, v čase, keď začal ponúkať svoje služby Napster (kedysi populárna p2p sieť na zdieľanie súborov), ktorý viedol k zvýšeným nárokom na prenosové kapacity poskytovateľov internetu.

Dnes sa odhaduje, že p2p prenos predstavuje 49 až 83 percent (pri tomto odhade záleží aj od konkrétnych sledovaných ISP, aj od času sledovania) celkovej internetovej premávky a v noci dosahuje až 95 percent [2].

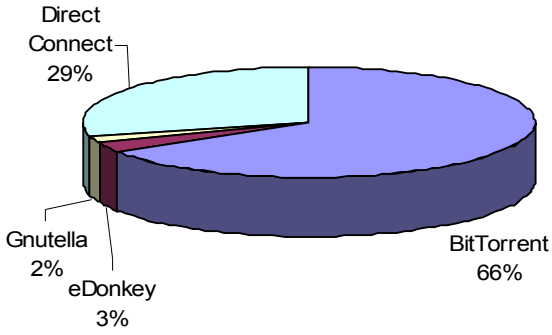


Obrázok 2.3: Štruktúra typickej IP premávky v čase [2].

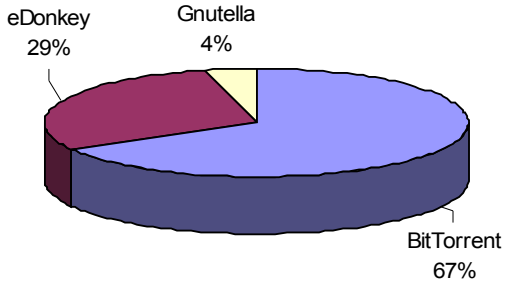
Takéto množstvo prenášaných dát okrem spomínanej vyššej nákladovosti poskytovateľov pôsobí aj ďalšie problémy, najmä zahlcuje prenosovú sústavu, keďže prenosy prebiehajú obvykle cez štandardné TCP spojenie, bez nejakého QoS (Quality of Service). Najväčšiu časť z toho tvorí premávka v p2p sieťach pre zdieľanie súborov, v poradí najmä BitTorrent, nasledovaná eDonkey a Gnutellou.

Nie všetky p2p prenosy a prenosy s nimi súvisiace (komunikácia s trackerom, alebo inou súčasťou p2p siete za účelom iným ako samotná výmena dát) sú však detekované. V čase písania tejto práce je asi 20% všetkých p2p prenosov šifrovaných [2], čo robí ich detekciu problematickou a nielen ďalšie, neznáme a menej rozšírené prenosy, ale aj niektoré toky známych sietí môžu tiež unikať detekcii. Tento pomer sa len pomaly zvyšuje pre dôvody ktoré sme si spomínali (vďaka réžii prenosov a samotnej architektúre 3. generácie p2p sietí) sú tieto siete rozšírenejšie v Japonsku alebo Kórei, kde sú bežné rýchle optické spoje až k zákazníkovi FTTH (fiber-to-the-home) / FTTP (fiber-to-the-premises).

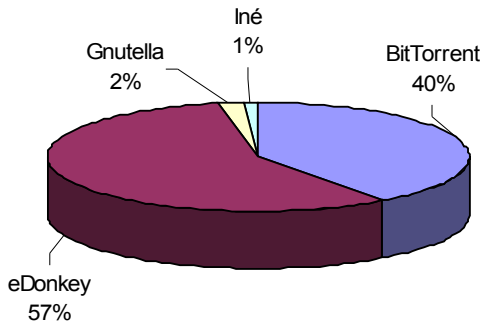
Detektory p2p sietí (najmä ich konkrétnych typov) sú obvykle optimalizované pre miestne podmienky – určité p2p siete sú totiž lokalitne vysoko špecifické – napríklad sieť Direct Connect je populárna vo východnej Európe a na Balkáne, no v západnej Európe, Amerike ani Ázii nie je používaná vôbec alebo len zanedbateľne. Naopak, BitTorrent je veľmi populárny na celom svete.



Obrázok 2.4: Objem prenosu podľa typu p2p siete v strednej a východnej Európe.



Obrázok 2.5: Používanosť p2p siete podľa jej typu v Nemecku (podobné pre celú západnú Európu).



Obrázok 2.6: Používanosť p2p siete podľa jej typu v južnej Európe.

Tieto odhady vychádzajú zo štúdie spoločnosti Ipoque [3] v auguste a septembri 2007, kedy bolo analyzovaných asi 3 Petabajty dát na všetkých kontinentoch sveta, najmä v Európe – a sú prevažne len orientačného charakteru. Na základe preštudovaných množstva online dokumentov si dovolím odhadnúť, že vzájomné pomerné zastúpenie popisovaných protokolov sa doteraz, v čase písania tejto práce príliš nezmenilo, len mierne vzrástlo zastúpenie nových protokolov poslednej generácie, aj tak však tvorí len rádovo jednotky percent celkovej premávky, najmä v závislosti na lokalite.

Čo sa týka miestnych podmienok – tu tvorí podľa údajov Slovenských poskytovateľov[4] p2p premávka 51,3% objemu všetkých sťahovaných dát, plus 1,4 % Skype ako zvlášť kategória. Situácia je v prípade odosielaných dát ešte zaujímavejšia – až 87,4 % ich objemu tvorí p2p a 3,0 % Skype.

3 Možnosti detekcie

3.1 Úroveň detekcie

Pred tým, než sa čitateľ pustí do štúdia spôsobov detekcie, treba upozorniť že detekcia p2p sietí sa dá chápať z viacerých hľadísk a aj samotná detekcia sa môže týkať rôznych úrovní.

Systém môže detekovať p2p premávku, keď:

1) zachytí typické sledy a frekvenciu komunikácie príznačné pre prítomnosť p2p sietí
2) určí paket, alebo dátový tok ako náležiaci konkrétnej p2p sieti
3) zistí, čo je obsahom p2p komunikácie (ID či názvy prenášaných súborov, alebo konkrétne dáta či, najčastejšie, len ich hash)

Detektory varianty č.2 sú hojné u poskytovateľov internetu, majúce za úlohu chrániť sieť pred preťažením. Sú kompromisom medzi komplexnosťou rozlišovania premávky v p2p sieťach a prevažne len štatistickou užitočnosťou prvej varianty.

Systém môže buď rozlišovať toky a vhodne označovať ich pakety pre radenie paketov (packet scheduling), prípadne označiť klienta s konkrétnou IP adresou ako užívateľa p2p sietí a podľa toho reagovať. Oba prístupy majú svoje výhody aj nevýhody – záleží najmä od mechanizmu detekcie.

Implementácie podrobných detektorov konkrétnych p2p sietí sú menej obvyklé, a najčastejšie používané autoritami hľadajúcimi z právneho hľadiska problematický obsah.

Existuje však množstvo podnikových a vedeckých p2p sietí (a objem ich premávky sa rapídne zvyšuje), ktorých detekcia nie je síce potrebná, ale už len premávka viacerých, rôznych p2p sietí cez jeden detekovaný uzol, ich samotnú detekciu komplikuje.

3.2 Ďalšie akcie obvykle prepojené s detekciou

Na základe výstupu detektora p2p môže pripojený systém regulovať (obmedzovať) premávku, zahadzovať niektoré pakety, vkladať pakety resetujúce spojenia či inak sťažovať p2p premávku.

Traffic shaping (obmedzovanie rýchlosti premávky) konkrétnych rozpoznaných tokov je veľmi častý jav u poskytovateľov internetu. Využívanou technikou je Class Based Queuing (CBQ), prípadne Hierarchical Token Bucket (HTB).

Komerční poskytovatelia internetového pripojenia využívajúci tieto systémy sa nazývajú Bad ISPs. Ich dobrý a často aktualizovaný zoznam, spolu aj s princípmi detekcie ktorú využívajú a možnými obranami nájde záujemca v literatúre [5].

Na druhej strane, zaujímavosťou je, že sa objavil aj softvér pre koncových používateľov (od Electronic Frontier Foundation), ktorý dokáže detekovať takéto zmeny tokov zo strany

poskytovateľa internetu, ako sú blokovanie, shaping, packet injection (RST pakety) a iné techniky boja proti p2p. Oveľa bežnejšie sú však detektory len shapingu (napríklad plugin do klienta Azureus)

Zjednodušenou obranou poskytovateľov internetu proti nadmernej p2p premávke je napríklad aj FUP (Fair Use Policy) implementované obyčajným počítaním objemu sťahovaných a/alebo odosielaných dát, alebo limitovaním počtu TCP spojení. Pre bežné surfovanie, klientskú prácu s e-mailami stačí 50-100 súčasne otvorených spojení, preto vhodným limitom bežných užívateľov je hodnota okolo 150 simultánne otvorených TCP spojení. Tieto spôsoby nie sú zavedené výhradne kvôli limitovaniu p2p (i keď je to najvýznamnejší dôvod), podmienky sú poskytovateľmi väčšinou zverejnené a preto sa títo ISP nenazývajú Bad ISPs.

3.3 Primitívne spôsoby detekcie

Najjednoduchším a najrýchlejším spôsobom detekcie konkrétnej p2p siete je sledovanie portov a / alebo IP adries p2p aplikáciami bežne používanými.

p2p sieť	**p2p klient**	**predvolené porty**	**zoznamy rozšírenejších kontaktovaných adries (servery, trackery)**
BitTorrent	*(všetky)*	6881-6889 TCP/UDP 6969 TCP (tracker port)	*http://trackerlist.net/*
eDonkey	eDonkey2000	4662/TCP	*http://www.peerates.net/*
	eMule	4661-4665/TCP, 4672/UDP	
Gnutella	Limewire, Morpheus	6346/6347 TCP/UDP	(bez významných adries)
Gnutella, Imesh	BearShare	6346 TCP/UDP	
Napster	*(všetky)*	6699, 8888, 7777/TCP 8875/TCP (redirector)	*http://gotnap.com/index.php?loc=list*
WinMX	WinMX	6699/TCP, 6257/UDP	(primárne servery boli zrušené nátlakom RIAA, avšak existuje sieť WPN)
DC++	Direct Connect	411 TCP/UDP, 412 TCP	*http://www.freeweb.hu/pankeey/dc-hubz/pankeey-dchubz.config.bz2*

Tabuľka 3.1: Predvolené porty a bežne kontaktované servery klientov niektorých rozšírených p2p sietí.

Metóda je však značne nespoľahlivá a produkuje falošne pozitívne výsledky v prípade, že daný port je práve používaný inou aplikáciu - keďže obvyklý rozsah portov je 2^{16}, existuje dosť vysoké riziko kolízie. Napríklad sieť Kazaa používa port 80, pôvodne vyhradený pre surfovanie po webe,

pre vlastnú komunikáciu. Falošne negatívne výsledky sú zase výsledkom zmeny štandardných portov p2p aplikácií či už implicitne alebo z dôvodu obsadenia portu inou aplikáciou, prekladu portov počas transportu a podobne.

Aj IP adresy serverov, či ich DNS menné ekvivalenty sa v čase značne menia, okrem toho ich existuje veľké množstvo – a ani ich zoznamy väčšinou neobsahujú všetky servery siete. Komunikácia so známym serverom však priamo neidentifikuje p2p tok a tvorí len zanedbateľnú časť objemu prenosu. Z týchto dôvodov sa pre vysokú nespoľahlivosť takéto metódy používajú minimálne, prípadne len ako doplnkové.

3.4 Detekcia na základe NetFlow

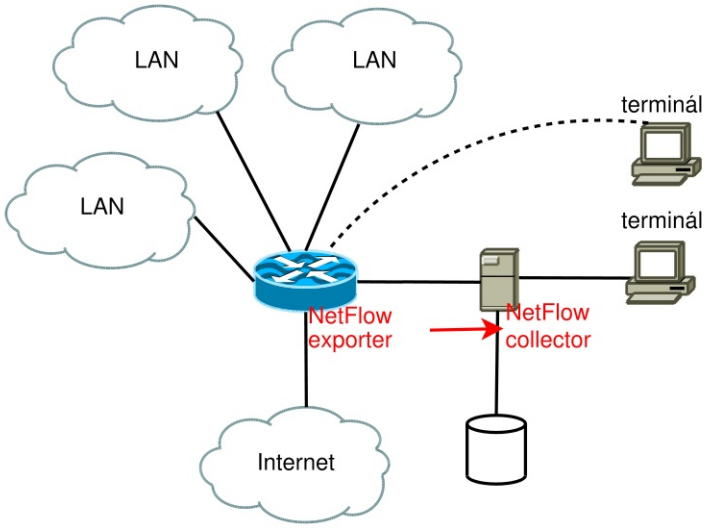
3.4.1 Stručná charakteristika

Sieťový tok je abstrakcia, definovaná ako jednosmerný prenos paketov, ktoré majú určité charakteristické (rovnaké) vlastnosti:

- IP protokol
- Zdrojová IP adresa
- Zdrojový port
- Cieľová IP adresa
- Cieľový port

Je dôležité poznamenať, že rovnako ako TCP, aj nespojovaná služba UDP môže byť tokom.

IP toky sú popisované mnohými spôsobmi a NetFlow, jedna z nich, je technológiou spoločnosti Cisco Systems - ide o otvorený protokol implementovaný Cisco IOS zariadeniami. Slúži pre zbieranie informácií a štatistík o IP tokoch. Tieto informácie sú odosielané cez UDP (User Datagram Protocol) alebo SCTP (Stream Control Transmission Protocol) na preddefinovanú adresu zariadenia NetFlow Collector, ktorý tieto informácie zbiera, spracúva a ukladá. Klienti, ktorí majú záujem o štatistiky k nemu môžu pristupovať a analyzovať zaujímavé údaje. Alternatívne, NetFlow dáta môžu byť priamo zasielané klientovi na analýzu.



Obrázok 3.2: NetFlow zber a analýza dát o IP tokoch [6].

Aj keď NetFlow bolo spočiatku implementované Ciscom, stalo sa štandardom IPFIX (Internet Protocol Flow Information eXport), čím umožnilo prevzatie veľkou časťou hlavných výrobcov zariadení sieťovej infraštruktúry, ktorí podporu pridali do svojich routrov a switchov. Táto sa nazýva aj NetFlow v10, čo je rozšírená NetFlow v9 - a práve NetFlow v9 je v čase písania tejto práce štandardom a na ňu sa bude autor ďalej odkazovať, i keď výrobcovia len pomaly prechádzajú z dlho používanej a veľmi rozšírenej NetFlow v5.

Informácia o toku je záznam, obsahujúci veľké množstvo informácií, avšak len niektoré z nich sú zaujímavé z hľadiska detekcie tokov:

- poradové číslo toku
- verzia NetFlow
- vstupné a výstupné sieťové rozhranie
- časová pečiatka začiatku a konca toku
- počet pozorovaných paketov a množstvo prenesených bajtov
- zdrojové a cieľové adresy
- zdrojové a cieľové porty
- IP protokol
- hodnota ToS (typ služby - hodnota určujúca požadované vlastnosti toku)

Posledné 4 hodnoty sú z hlavičiek protokolov z tretej vrstvy ISO/OSI modelu. Všetky tieto údaje nemusia byť odosielané nepravidelne ale až po nazbieraní určitého množstva informácií o tokoch - typicky 30 až 50 tokoch.

V praxi však, zvlášť pri intenzívnej p2p prevádzke a veľkých sieťach, tabuľky o tokoch v routroch sa môžu zaplniť, ale častejším problémom pri veľkej prevádzke je, že procesor systému si nestíha robiť prehľad o tokoch. Cisco ponúka riešenie – „Sampled NetFlow" spočívajúce v spracovaní len každého n-tého, prípadne každého m-tého (kde *m* je meniace sa náhodné číslo -

vtedy ide o „Random Sampled NetFlow") paketu. Výsledky sú potom len približné, avšak odrážajú rozloženie prenosovej kapacity.

Archivácia NetFlow údajov je tiež problematická - zaberajú veľa miesta a je vhodné, keď ich Collector rovno priebežne spracúva.

Treba povedať, že existujú alternatívy NetFlow - príkladom je sFlow, zakotvený dokonca v RFC3176, ktorý implementujú napríklad niektoré HP Procurve switche.

3.4.2 Vlastnosti p2p tokov v NetFlow dátach

Samotná detekcia je pasívna, teda až na základe prijatých NetFlow dát je možno s určitou spoľahlivosťou povedať, že ide o p2p. Ide teda o detekciu prvého stupňa. Prejavujú sa tu typické vlastnosti p2p sietí, ako napríklad veľké množstvo nadväzovaných a zanikajúcich spojení za krátky čas, najrôznejšie dosiahnuteľné IP adresy (ak je sieť pripojená k internetu, nie len LAN), vysoké množstvá prenášaných dát v tokoch, skôr spojitejšia než nárazová charakteristika zmien rýchlostí prenosu, opakovane nadväzované spojenia s tými istými cieľmi s približne rovnako objemnými prenosmi a v prípade adresovateľnosti zvonka (aktívny mód) aj približne vyrovnané množstvo prichádzajúcich a odchádzajúcich spojení.

Tieto fakty je vhodné spracovať heuristickými metódami, napríklad neurónovými sieťami. Poskytujú dobrý výsledok pre veľa druhov podobných p2p sietí a ich implementácia je univerzálnejšia. Prednosťou sú aj relatívne nízke výpočetné nároky (oproti inšpekcii paketov), veľkou nevýhodou je však práca až nad zozbieranými štatistikami.

Výsledkom analýzy NetFlow dát je len určitá miera pravdepodobnosti, že dané toky tvoria p2p premávku, takže identifikácia je, na rozdiel od detekcie na základe obsahu paketov, nejednoznačná.

3.5 Detekcia na základe obsahu paketov

Inšpekcia paketov - (deep) packet inspection je jedinou presnou možnosťou detekovať online p2p prevádzku. Siete p2p komunikujú vlastnými, aplikačnými protokolmi na najvyššej, siedmej vrstve ISO/OSI modelu a na svoj prenos môžu používať veľa rôznych protokolov nižších vrstiev bežiacich na sieťovej vrstve. Detektor p2p sa musí „prehrýzť" od sieťovej až po aplikačnú vrstvu, čo je náročné nielen na výpočetný výkon, ale aj na vybavenie detektora, ktorý musí podporovať všetky v bežnej p2p prevádzke používané transportné, relačné a prezentačné vrstvy (i keď mnohokrát sú súčasťou už p2p protokolu).

Spoľahlivý postup algoritmu analýzy je v ideálnom prípade nasledujúci:

1) Každý paket je „rozbalený" až po najvyššiu, aplikačnú úroveň.

Postup k najvyšším úrovniam môže byť buď plynulý, alebo je možno spojiť niekoľko za sebou nasledujúcich protokolov do jedného priechodu rozbalenia.

2) Konkrétne znaky konkrétneho p2p protokolu sú vyhľadávané.

Každý protokol má obvykle svoje charakteristické črty, podľa ktorých je možno ho identifikovať. Tieto hodnoty sú buď preddefinované dopredu, alebo existuje funkcia sem mapujúca hodnoty na iných miestach v pakete (prípadne aj v iných, predchádzajúcich paketoch). Keďže existuje veľa rôznych p2p protokolov, je potrebné nájsť spoločné vlastnosti usvedčujúce čo najviac rôznych protokolov.

Tento postup nám spoľahlivo detekuje známe protokoly, avšak nerozpozná neznáme a rýchlosťou tiež obvykle nevyniká. Tabuľky hodnôt pre konkrétne protokoly sú však mimo rozsah tohto dokumentu – je to rozsiahla aplikovaná problematika týkajúca sa spätného inžinierstva.

Miesto detekcie konkrétnych hodnôt na konkrétnych (preddefinovaných) pozíciách v protokole sa môže využiť **pattern matching**. Ide o rozpoznávanie pomocou známeho a preddefinovaného vzoru (pattern), ktorý môže mať podobu buď sekvencie (regulárny výraz), alebo stromovej štruktúry.

Pattern recognition je rozpoznávanie vzorov umožňujúce detekciu známych i neznámych protokolov, prípadne nových verzií už existujúcich na základe štatistických informácií o obsahoch paketov. Vzory ktoré majú byť rozpoznané sú obvykle skupiny pozorovaní umiestnené vo viacrozmernom priestore. Preddefinovaným vzorom na začiatku procesu sa hovorí trénovacia množina a klasifikátor určujúci príslušnosť skúmaného vzoru do niektorej triedy podľa množín už existujúcich sa týmto spôsobom učí.

3.5.1 Regulárny výraz

Regulárny výraz (vzor) je reťazcom popisujúcim určitú množinu reťazcov. Táto množina môže byť vymedzená nekonečným počtom regulárnych výrazov keďže regulárne výrazy (RV) je možné expandovať a niekedy aj redukovať bez zmeny ich popisovacích vlastností. Informácie o regulárnych výrazoch a ich implementáciách POSIX RE (Portable Operating System Interface Regular Expressions) či PCRE (Perl Compatible Regular Expresions) možno nájsť na mnohých webstránkach.

Automat (parser), sa typicky pokúša prijať reťazec metódou backtrackingu – zisťuje, či reťazec vyhovuje predlohe – na tento proces sa bude autor odkazovať na proces porovnávania. Je dôležité si uvedomiť, že niektoré špeciálne znaky značne zjednodušujú prácu parsera (napríklad znaky označujúce začiatok a koniec reťazca), avšak iné (znaky „iterácia“ či „pozitívna iterácia“) ju robia komplikovanejšiu (v zmysle procesorovo náročnejšiu). Podľa toho je vhodné voliť regulárne výrazy tak, aby boli pre automat čo najjednoduchšie spracovateľné, aj keď väčšina implementácií si vie výraz optimalizovať.

3.6 Hardwarové implementácie

Najčastejšie využívanou triedou detekcie p2p sietí sú zariadenia implementujúce online detekciu p2p sietí inšpekciou paketov, keďže, ako je rozobrané v práci nižšie, NetFlow analýza je použiteľná iba offline. Keďže je toto riešenie náročné na výpočetný výkon, využívajú sa samostatné akcelerátory, prípadne aplikačne špecifické obvody.

Existujú najmä proprietárne riešenia so špeciálnym HW založeným napríklad na FPGA montované v štandardných krytoch do bežných rackov, napríklad balík *Packeteer* [7]. Maximálna kapacita produktu Packeteer 10000 je až 400 000 TCP spojení, 200 000 IP hostov a 5 000 pravidiel pre pattern matching. Pracuje s 1Gbps NIC portami ktoré sú zároveň aj limitom jeho rýchlosti.

Obrázok 3.3: PacketShaper 10000 spoločnosti Packeteer.

Ďalej možno spomenúť riešenie *ARA Networks P2P Management Solution* od spoločnosti ARA Networks. Aj Cisco ponúka riešenie detekcie a limitovania p2p prevádzky nazvané Cisco Service Control Technology.

Existujú aj špeciálne riešenia pre vysokorýchlostné siete. Platforma Ellacoya e100 (dnes už je Ellacoya odkúpená spoločnosťou Arbor) ponúka priepustnosť až 20 Gbps a kontrolu až 500 000 IP hostov na jedno zariadenie tejto platformy [8]. Kombinuje vyššie popísanú hĺbkovú inšpekciu paketov so systémom prevencie proti narušeniu (intrusion prevention system, IPS).

Obrázok 3.4: Ellacoya e100.

NetFlow analýza s HW podporou / urýchlením je vzácna a poskytovatelia internetu ju využívajú primárne ako monitoring (odhad) množstva p2p prevádzky, alebo v situáciách, keď by právne aspekty súkromia pri inšpekcii paketov boli objektom príliš veľkého záujmu.

3.7 Softwarové implementácie

Typické riešenie sa skladá zo súčastí, ktoré by mali tvoriť komplexné softwarové prostredie pre detekciu a správu detekovanej p2p premávky, ako napríklad [9]:

- Komponenty identifikácie paketov, a to:
  - Jednoduché numerické identifikátory ako napríklad porovnávanie na základe portu, IP, veľkosti paketu či payloadu atd (napríklad štandardné moduly *iptables*).
  - Identifikátory na základe regulárnych výrazov (napríklad *L7-filter*)
  - Identifikátory na základe stromových vzorov či iných štruktúr popísateľných funkcionálnym programovaním (napr. *ipp2p*)
- Databázu s pravidlami [10] ako najlepšie identifikovať p2p sieť a jej vhodné pripojenie, prípadne čo najvyšší stupeň automatizácie a zjednodušenia aktualizácie
- Komponentu obmedzujúcu rýchlosti detekovaných paketov či tokov obsahujúcich detekované pakety (napr. *Linux QoS*)
- Komponentu zahadzujúcu detekované pakety (napr. *Netfilter*)
- Rozhranie (front-end) pre užívateľa, najlepšie v grafickej podobe alebo vo forme sady skriptov, aby administrátori neboli zaťažovaní konfigurovaním a vzájomným ladením vyššie spomenutých komponentov

Vzhľadom na to, že práca sa zaoberá detekciou p2p sietí, v nasledujúcich kapitolách sa pozrieme sa na niektoré používané identifikátory bližšie.

3.7.1 L7 – filter

Veľmi všestranný nástroj [9] – klasifikátor naprogramovaný v C++ pre rozpoznávanie veľkého množstva protokolov (p2p protokoly sú len malou časťou) na najvyššej vrstve. Predpokladá spoluprácu s iptables / netfilter (ktoré si viac rozoberieme neskôr). Prehľadávanie reťazcov sa deje pomocou regulárnych výrazov (ktorých definícia sa veľmi podobá základným GNU regulárnym výrazom) v prvých maximálne 8 paketoch toku po zostavení spojenia, o ktorom má informácie z Netfilter modulu Conntrack. L7-filter vie prehľadávať aj naprieč viacerými paketmi, čo nám umožňuje porovnávať vzory komunikácie.

Súčasťou L7-filtra je aj súborová databáza definícií veľkého množstva L7 protokolov, včítane približne tucta rozšírených p2p protokolov (BitTorrent, Direct Connect, eDonkey, Kazaa atď.)

Keďže regulárne výrazy sú „pohodlným“ nástrojom pre definovanie detekcie podľa obsahu paketov, databáza tohto nástroja sa pomerne ľahko upravuje, čo robí aktualizáciu veľmi jednoduchou.

I keď detektor môže byť aj v userspace, dokáže aj využívať kernel s iptables pre efektívne čítanie, značenie prípadne ďalšie preposielanie paketov, samotný engine využíva POSIXové Regular

Expressions (v userspace verzii ERE - Extended Regular Expressions) vďaka čomu je pomerne silný a univerzálny, aj keď kernel verzia je čo sa týka použiteľnosti regulárnych výrazov dosť biedna, kernel máva občas problémy so stabilitou a podporuje len slabé BRE - Basic Regular Expressions a teda je tendencia vo vývoji časom prejsť čisto len na userspace verziu - je výborne vyladená a podporuje aj multithreading.

Obsah všetkých paketov na vstupe teda „preháňaný“ POSIXivými ERE. Definícia syntaxe samotných regulárnych výrazov pre L7-filter ako vzorov p2p premávky sa od definície samotných POSIXových výrazov líší len minimálne. Zjednodušene, úloha L7-filtra je vlastne len načítať tieto filtre (s pár nutnými konverziami ako napríklad zapisovanie nezobraziteľných znakov), zoradiť do dostatočne dlhých reťazcov (payloady aj viacerých paketov) podľa požiadaviek ďalších, doplňujúcich atribútov, ktoré sprevádzajú definície regulárnych výrazov a detekciu je možné nimi dobre škálovať (rýchlostný kompromis).

Popis týchto regulárnych výrazov, ktorými sa „preháňa“ obsah všetkých paketov na vstupe, nájde čitateľ v literatúre [11].

Výhodou L7-filtra sú široké možnosti využitia nielen pre rozpoznávanie protokolov a typu prenášaných dát, ale aj veľmi spoľahlivé rozoznávanie typov prenášaných súborov, niektorých malware aplikácií apod.

Vhodnou kombináciou je využitie schopností filtrovania netfiltra a L7-filtra – týmto spôsobom dostaneme slušný výkon, avšak náročné je rozloženie pravidiel pre netfilter a pravidiel pre L7-filter tak, aby bol výkon čo najvyšší. Obvykle nastáva situácia, že administrátor nastaví pravidlá netfiltra tak, že všetky pravdepodobné (a v niektorých prípadoch zbytočne úplne všetky!) dáta „tečú“ cez L7-filter čo má zlé dôsledky na priepustnosť.

Porovnanie rýchlostí v rôznych podmienkach je mimo rozsahu tohto dokumentu, záujemca si môže prezrieť rýchlosti, zaťaženie a maximálne kapacity pri rozličných HW konfiguráciách a sieťových podmienkach na stránkach projektu v sekcii Performance.

3.7.2 IPP2P

Minimalistický a špecializovaný detektor p2p sietí [12]. Vyhľadávanie jednotlivých protokolov je nakódované funkcionálne v C „napevno“ a čo možno najefektívnejšie. Tvorí fakticky komplement vlastností (spôsobu detekcie) L7-filtra – pridávanie nových protokolov, dokonca len drobné menenie vlastností a aktualizácia je problematická. Detektor je tiež pomerne rýchly (no s inou rýchlostnou charakteristikou), má minimálne množstvo nastavovacích prvkov – v praxi je možno len určiť, ktoré p2p protokoly sa majú vyhľadávať, čo je obvykle dosť málo – všetko ostatné je „hardwired“.

Tento detektor bol kedysi približne rovnako úspešný ako L7-filter, a v niektorých špecifických prípadoch starších p2p protokolov ho dokonca predčil – je v ňom implementované rozpoznávanie skoro všetkých riadiacich príkazov detekovaných p2p sietí, dokonca aj takých pre ktoré by boli aj regulárne výrazy v L7-filtri komplikované - predsa len, výrazová schopnosť bežných programovacích

jazykov je vyššia než regulárnych výrazov, avšak pre novšie p2p protokoly, respektíve novšie verzie p2p protokolov dosahuje slabé výsledky a to najmä kvôli tomu, že koncom roka 2006 vyšla posledná verzia (0.8.2) a odvtedy sa tento detektor nevyvíja. Komunita okolo tohoto detektora je výrazne menšia ako v predchádzajúcom prípade, autor vyvíjal detektor prevažne sám a projekt dopláca na nevýhody plynúce z jeho dizajnu – najmä na obtiažnu aktualizáciu.

Podobne ako L7-filter, aj tento detektor je postavený na kombinácii iptables / netfilter, takže zapojenie detektora je triviálne, napríklad pravidlom

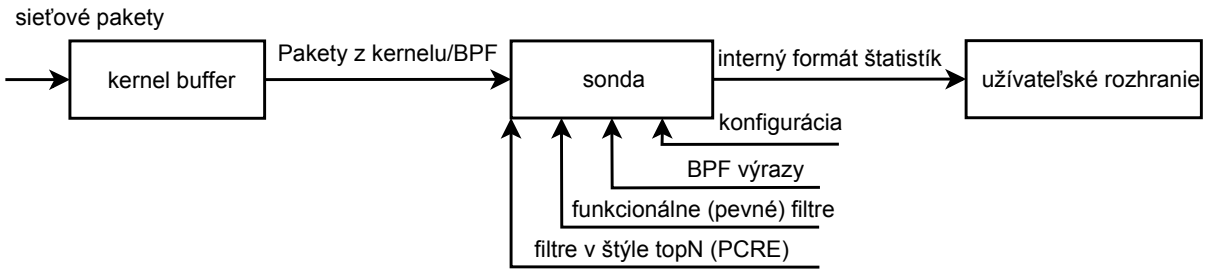
```
iptables -A FORWARD -m ipp2p --ipp2p -j DROP
```

zapojíme detektor p2p konfigurovaný pre všetky známe a otestované p2p siete tak, aby úspešne rozpoznané pakety boli vypustené.

Zbežne sa pozrieme aj na niektoré ďalšie, menej rozšírené nástroje.

3.7.3 Ourmon

Ourmon je kompletný systém pre detekciu a monitorovanie siete [13]. Tvoria ho súčasti pre zachytávanie paketov na Ethernetovom rozhraní, filter, štatistický modul a užívateľské rozhranie. Hodí sa aj pre detekciu sieťových anomálií a neštandardných spojovaných aj nespojovaných IP prenosov, keďže sa riadi vlastnou definíciou tokov. Filter využíva ako primárny filter klasický BPF a najnovšia verzia podporuje aj PCRE (Perl Compatible Regular Expressions – regulárne výrazy kompatibilné s Perlom – majú výhodnejšiu formu zápisu) vytvárajúce zotriedené zoznamy najväčších tokov, zaujímavých IP atď. v štýle topN.



Obrázok 3.5: Schéma monitorovacieho systému Ourmon.

3.7.4 HiPPIE – Hi-Performance Protocol Identification Engine

Je pasívnym identifikátorom protokolov čítajúcim dáta buď priamo zo sieťového rozhrania, ale môže spolupracovať aj s netfiltrom [14]. Druhá možnosť je už len doplnková ďalej nepodporovaná, keďže ako autor na stránke projektu uvádza, možnosti detektora sú vyššie než len robenie rozhodnutí pre netfilter. Pre svoju efektivitu vyžaduje prísun všetkých paketov bez akéhokoľvek predfiltrovania a domnievam sa, že ďalším dôvodom môže byť aj snaha o maximalizovanie priepustnosti. Projekt je cielený pre rozpoznávanie najrozšírenejších protokolov vyšších vrstiev, počnúc IP.

Implementovaný detektor má niekoľko užitečných vlastností ako session tracking, session prediction, tunnel tracking, no ale aj pár neduhov – na systémoch s podporou SMP občas mrzne a čo je dôležité, sada podporovaných protokolov je malá a rovnako ako množina vyhľadávaných reťazcov pre jednotlivé protokoly, takže detektor má v porovnaní s L7-filtrom či Ourmonom nižšiu citlivosť. Detektor je momentálne vo verzii 0.90 a ešte ho čaká dlhá cesta vývoja. Samotný princíp detekcie nie je dostatočne zdokumentovaný, no zdá sa že využíva pevne definovanú sadu pravidiel.

HiPPIE sa dá zapojiť aj ako identifikačný engine L7 protokolov do existujúcich aplikácií.

3.8 Platformy, komponenty a knižnice softwarových implementácií

Pre návrh detektora je vhodné sa oboznámiť s niektorými existujúcimi súčasťami reálnych p2p detektorov.

3.8.1 Netfilter / iptables

Asi najpopulárnejšia kombinácia netfilter / iptables vo forme loadable kernel modules umožňuje klasifikovať pakety podľa portov či IP pomocou reťazí pravidiel, preposielať takéto pakety do ďalších pripojených súčastí na podrobnú klasifikáciu a detekované, prípadne podozrivé pakety priamo označovať, spracúvať, zahadzovať či posúvať do frônt v userspace pre ďalšie spracovanie. To však platí pre kernely Linux 2.4, 2.6., príp. vyššie. Staršie kernely využívali ipchains (Linux 2.2) a ipfwadm (Linux 2.0), ktoré však neoddeľovali filtrovacie pravidlá od pravidiel modifikujúcich pakety (napr. NAT).

Iptables z pohľadu správcov sietí je nástroj ktorým môžu pomocou príkazovej riadky či shellových skriptov vytvárať pravidlá pre filtrovanie paketov (prichádzajúcich aj odchádzajúcich), teda firewall a pravidlá prekladu adries NAT.

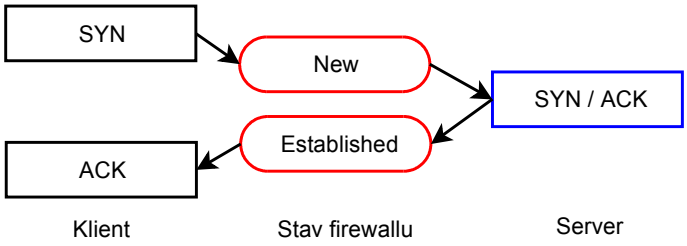
Súčasné iptables využíva 3 tabuľky s pravidlami priradenými k týmto vstavaným reťaziam:

- **Filter** – občas je samotným iptables myslená len táto tabuľka, aj keď to nie je pravda. Tu sú uložené filtrovacie pravidlá priradené vstavaným reťaziam INPUT, OUTPUT a FORWARD
- **NAT** – tabuľka prekladu adries (Network Address Translation). Používajú sa reťaze PREROUTING, POSTROUTING a INPUT
- **Mangle** – tabuľka popisujúce pravidlá zmien paketov ktoré nie sú typu NAT, napríklad zmeny QoS v TCP hlavičke paketov. Menšie siete ju často ani nevyužívajú. Umožňuje využiť vstavané reťaze PREROUTING a OUTPUT.

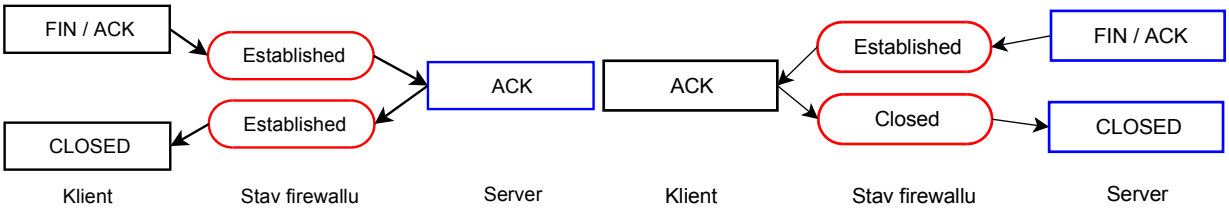
Najčastejšie pri zapájaní detektora do systému využijeme tabuľku Filter s položkou priradenou k reťazi FORWARD.

Takýto firewall používajúci Iptables sa nazýva stateful firewall, pretože iptables dokáže sledovať toky dát. Každému toku je priradený atribút popisujúci jeho aktuálny stav. Stavový stroj je riadený posledným čítaným paketom ako aj vnútorným stavom (kontextom). Takisto je prihliadané na internú sadu timeoutov.

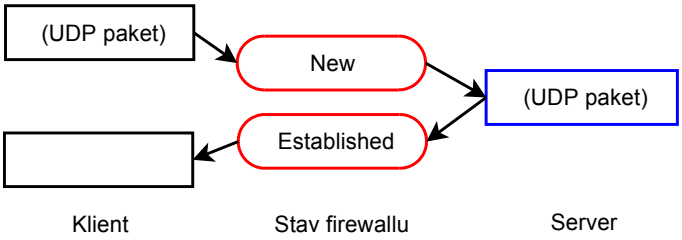
Pre TCP spojenia máme v zásade k dispozícii 3 základné stavy: New, Established a Closed.



Obrázok 3.6: Nadväzovanie TCP spojenia.



Obrázok 3.7: Ukončovanie TCP spojenia klientom a serverom.



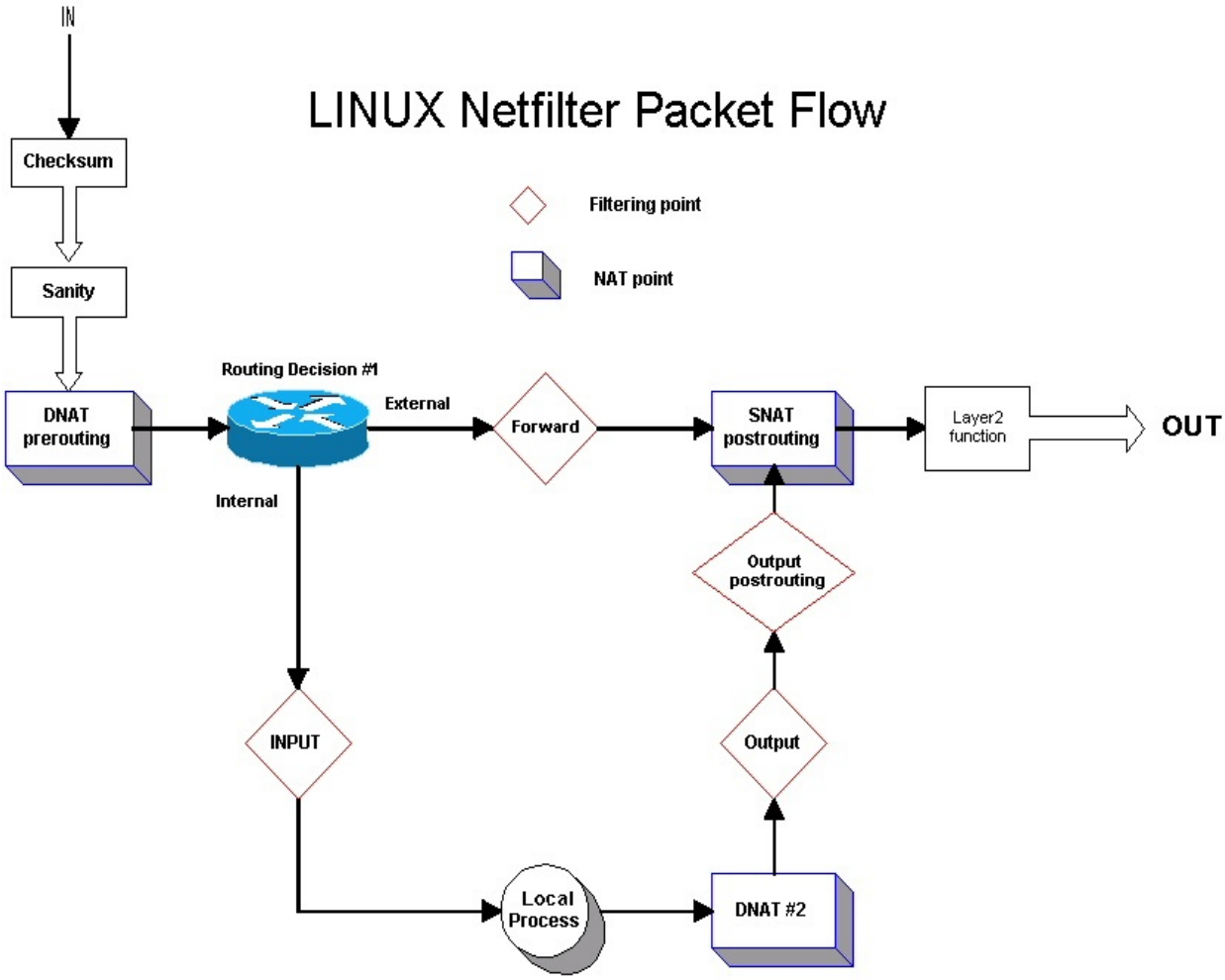
Obrázok 3.8: Nadväzovanie UDP spojenia.

V skutočnosti má iptables 10 stavov (ďalšie stavy totiž pribúdajú najmä kombináciou s timeoutmi), pre ilustráciu stavov, ktoré môžeme využiť detektorom by to však malo stačiť.

Dostávame takto nástroj pre sledovanie celých konverzácií v p2p sieťach. Predchodcovia iptables (ipchains, ipfwadm) boli stateless, teda takúto možnosť nemali a rozhodovali sa len na základe atribútov aktuálneho paketu.

Dôležitou súčasťou nasadenia iptables je netfilter – je to vlastne sada obslúh udalostí (event handlers) tvoriaca základ (framework) pre implementáciu najmä filtrovania. Aj keď mávajú často formu loadable kernel modules, rovnako existuje aj sada knižníc pre userspace použitie.

Netfilter umožňuje rozdeliť prácu s paketmi na tri nezávislé časti a to fitrovanie, sledovanie spojení a preklad adries. Treba poznamenať, že pre plné využitie schopností iptables / netfiltra je treba nainštalovať aj sadu knižníc libnetfilter_queue a libnetfilter_conntrack – vďaka čomu náš detektor získa schopnosť manipulovať s frontami netfiltra a sledovať spojenia – takto pracuje napríklad L7-filter. Práve pre tieto vlastnosti si môže dovoliť porovnávať vzory aj medzi paketmi. Viac je zrejmé z obrázka:



Obrázok 3.9: Packet flow v Netfiltri [15].

Pre základné filtrovanie budeme posielať podozrivé pakety do pripojenej aplikácie (napr. L7-filtra) len jeden tok – konkrétne zapojíme ho do „Forward“. Ako na to, a ako na zložitejšie schémy, čitateľ nájde v príslušných man stránkach iptables.

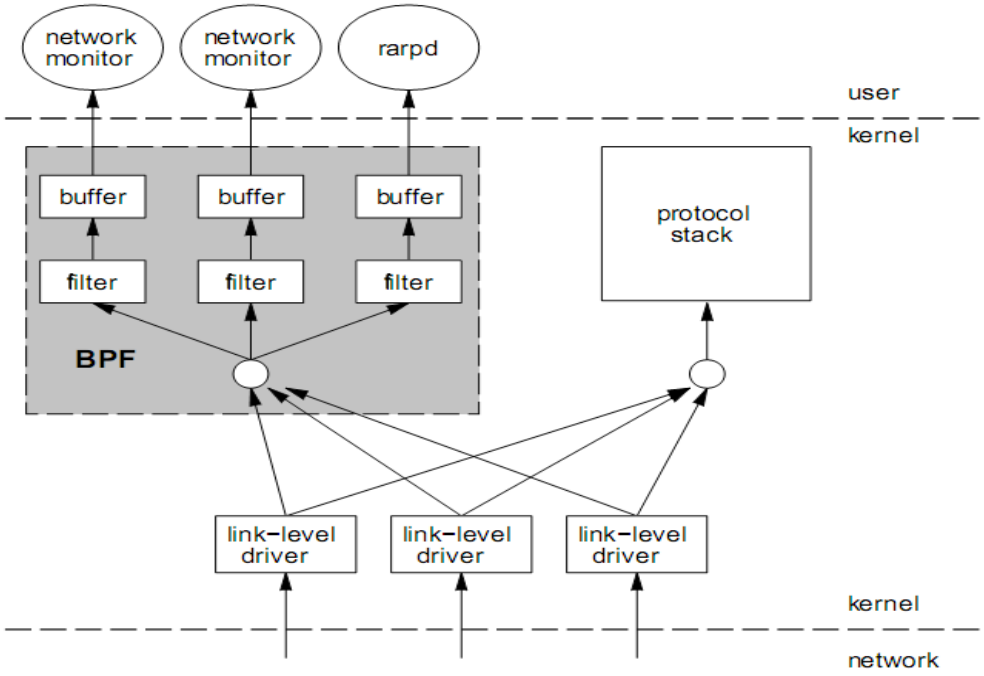
Napriek tomu že ide o populárnu voľbu, prezradím, že popísané Netfilter / iptables nie sú využité v nasledujúcej navrhnutej implementácii avšak je vhodné o nich vedieť.

3.8.2 BPF – Berkeley Packet Filter

Návrh [16] je už dosť starý, ale princíp filtra sa doteraz používa, najmä pre jeho rýchlosť. BPF býva kombinované s link-level čítaním paketov zo sieťového rozhrania a prečítané pakety sú filtrované špeciálnym strojom – automatom. Takýto optimalizovaný automat – interpret BPF kódu - môže voľne prechádzať obsahom paketu kdekoľvek od linkovej hlavičky až po payload najvyššej protokolovej vrstvy a vykonávať nad ním aritmetické operácie. Automat si môže medzivýsledky ukladať a takýmto spôsobom môžeme trebárs dynamicky dopočítať ofsety payloadov (napríklad pri neštandardnej dĺžke IP paketu) pre porovnávanie.

Jeho výstupom je booleovská hodnota a na základe nej sú prepustené sú len pakety filtru vyhovujúce; ostatné pakety sú ignorované. BPF je teda rýchlym filtrom podľa obsahu.

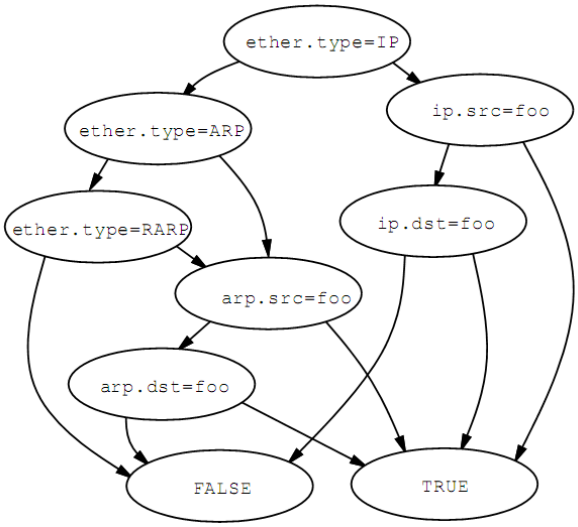
Samotné umiestnenie filtra v systéme je nasledujúce:



Obrázok 3.10: Zapojenie BPF filtra v systéme.

Ako vidíme na príklade z obrázka 3.11, „strojový kód“ tohto automatu je vystavaný na bezkontextovej gramatike (CFG) miesto obyčajného stromu, čo má výhody – niektoré čiastkové výrazy sa tak nemusia vyčíslovať viackrát a získavame značnú úsporu procesorového času.

Pseudokód reprezentujúci daný automat je interného formátu tak, aby jeho interpretácia v jazyku C bola maximálne rýchla – a bude sa odvíjať od priemernej výšky stromu. Veľa pseudoinštrukcií má približne rovnakú dobu vykonávania, teda celková rýchlosť filtra záleží od dĺžky a efektivity programu, ktorý môže byť nahraný do BPF enginu manuálne, alebo kompilovaním.



Obrázok 3.11: CFG model funkcie implementujúcej napríklad filter "host foo".

3.8.3 Pcap - sniffer s filtrom s BPF syntaxou

Libpcap je knižnica projektu pcap s API pre pohodlné zachytávanie (sniffing) sieťovej premávky. Primárne vznikol na unixových systémoch ako súčasť projektu Tcpdump, no rozšíril sa a existuje aj port pre Windows (winpcap), obsahujúci navyše medzivrstvu pre zachytávanie paketov z Windows ovládačov. Pod Linuxom zase libpcap využíva Linux Packet Filter – vlastnosť kernelu, keďže dáta zo socketov sú už vybalené zo sieťovej, prípadne transportnej vrstvy a to by detektoru nestačilo. Keďže je písaný v C, okrem čo najvyššej úrovne „multiplatformnosti" existuje veľké množstvo wrapperov pre vyššie jazyky napr. pre C++, Python, Ruby, Perl a Javu.

Obsahuje aj zabudovaný BPF engine a to aj s kompilátorom booleovského logického výrazu filtra vlastného formátu na BPF pseudokód. Tento Pcap filter má podporou množstva makier - preddefinovaných logických výrazov, porovnávajúcich obsah na konkrétnych či vypočítaných pozíciach s konkrétnymi hodnotami.

Sonda implementovaná libpcapom je veľmi častým riešením rôznych snifferov, protokol identifikátorov a IDS/IPS systémov, lebo zaobstaráva všetky nízkoúrovňové záležitosti a programátor len nastaví čo, kam a ako sa má zachytávať. Spomeňme aspoň Snort, Wireshark (predtým Ethereal), Tcpdump a Nmap ako príklady.

Možnosti pcapu nie sú len zachytávanie paketov - sniffovanie, ale obsahuje aj API pre vkladanie paketov na linkovej vrstve.

Pcap podporuje aj ukladanie do súborov vo svojom vlastnom formáte (a je dobrým zvykom, že aplikácie využívajúce libpcap podporujú pcap formát uložených dát).

Syntax pcap BPF filtra je uvedená v implementácii programu v kapitole 4.3.1.

4 Návrh a implementácia detektora p2p sietí

4.1 Analýza

Pre systém detekcie p2p sietí bol zvolený pasívny detektor p2p na základe inšpekcie paketov vzhľadom na to, že oproti NetFlow analýze poskytuje jednoznačné výsledky.

Detektory p2p využívajú pre získanie IP paketu, prípadne nižších vrstiev ISO/OSI modelu existujúce, efektívne a sieťovými aplikáciami (napríklad smerovanie) mnohokrát preverené služby systému.

Rozhodujúce pre rýchlosť aj úspešnosť je stanovenie vzorov, podľa ktorých je možno protokol identifikovať. Tento proces je časovo náročný, keďže obsahy paketov p2p sietí sú nielen čiastočne závislé od samotného protokolu a jeho verzie, ale aj od implementácií klientov – a práve pakety pre nich špecifické sú nezanedbateľnou položkou pre databázy p2p detektorov podľa obsahu.

Čisto softwarové riešenia detektorov typu packet inspection sú značne limitované priepustnosťou, v čase písania tejto práce rádovo v stovkách Mbit/s pre rozumne veľkú databázu najpoužívanejších p2p protokolov.

4.1.1 Špecifikácia

Mojim cieľom je spojiť prednosti viacerých SW implementácií popísaných v kapitole 3.7, niektoré vlastnosti doplniť, iné, podľa môjho názoru zbytočné, vypustiť a to tak, aby nevýhody vyplývajúce z kompromisu závažným spôsobom neznehodnocovali možnosti nasadenia. Zhrniem požadované vlastnosti spolu s možnosťami ich dosiahnutia do niekoľkých bodov:

- **Vysoká rýchlosť** – je jedným z hlavných cieľov. Systém by mal byť schopný spracúvať dáta prechádzajúce zariadením tak, aby aj pri pripojení na gigabitový Ethernet a bežnej prevádzke nenastávali straty packetov.
- **Senzitivita detekcie –** dôležitá, ale paradoxne, nie najdôležitejšia vlastnosť. Stačí ak bude detekovaná veľká väčšina prebiehajúcich detekovateľných prenosov (ideálne cez 90%). Je treba si uvedomiť, že je lepšie radšej prepustiť niečo málo p2p premávky, než blokovať nejednoznačne identifikovanú premávku čo by malo negatívny dopad na ďalšie aplikácie využívajúce sieť, prípadne zdržovať detekciu resp. úplne blokovať dôležitý uzol neúmerne vysokou výpočetnou zložitosťou.

- **Špecificita detekcie** – požadovaná veľmi vysoká z horeuvedených dôvodov. Vhodná špecificita je odhadom okolo 0,99999 čo tvorí pravdepodobnosť 10^{-5} FPR (False Positive Rate), no veľmi záleží od prenášaného protokolu, či akciu spojenú s falošnou detekciou p2p znesie. Ojedinelé, vypustené TCP pakety na základe detekcie nie sú síce problémom, keďže sa prenesú znova, avšak rovnaký prenesený paket pri rovnakom filtri a stave detektora vedie opäť k zamietnutiu. Rozhodnutie o celých tokoch, miesto paketoch je ešte radikálnejšie a musí existovať dostatočná istota (najlepšie na základe viacerých indícií) že sa jedná o tok p2p.
- **Rozšíriteľná databáza detekovaných protokolov** – plaintext súbor s definíciami popisov známych p2p protokolov.
- **Vstup (konfigurácia)** – obvyklé nastavenia pomocou prepínačov príkazovej riadky, ostatné, menej obvyklé a konštanty v klauzuliach `#define` v jednom, spoločnom hlavičkovom súbore v zdrojovom kóde programu.
- **Výstup –** štandardne `stdout`, avšak možnosť presmerovať do iného zariadenia, či do súbora.

Ďalšími vlastnosťami detektora by mali byť:

- **Jednoduchosť nasadenia** – program by nemal byť závislý na ďalších komponentách tak, aby bolo potrebné tieto závislosti nastavovať, ako musí byť napríklad kombinácia iptables / netfilter s L7-filtrom. Aj proces kompilácie a inštalácie by mal byť jednoduchý a priamy.
- **Možnosť pripojenia externej knižnice implementujúcej samotnú detekciu** s jasne definovaným rozhraním (viď. dokumentáciu kódu a rozhraní na priloženom CD).
- **Prostredie a programovací jazyk –** ich voľba by mala umožňovať rozbehnúť detektor na všetkých rozšírených platformách, kde je možná inštalácia libpcapu, prípadne uľahčiť portovanie do vstavaných (embedded) zariadení atď.

Systém bude cielený na nasadenie na router či bránu menšej, no aktívnej siete a koncipovaný tak, aby umožňoval detekovať niekoľko málo protokolov, ktoré sa v sieti značne rozšírili a ich podiel na premávke je vysoký a to v prostredí, kde je vysoké množstvo prenášaných dát.

Na navrhovaný systém sa možno pozerať aj ako na návrh možnosti optimalizácie maximálnych rýchlostných možností rodiny štandardných p2p detekovacích systémov založených na detekcii pomocou regulárnych výrazov spolu BPF predfiltrovaním.

4.1.2 Porovnanie s existujúcimi projektmi

Pri konštrukcii programu bude autor okrem vlastnej implementácie detekcie vychádzať z niektorých princípov, odsledovaných v iných projektoch. Niektoré sú len inšpiráciou, niektoré však boli prevzaté a čiastočne vylepšené (napríklad databáza p2p protokolov).

Názov projektu	Inšpirácia	Neprevzaté vlastnosti
L7-filter	◆ POSIX reg. výrazy ◆ Databáza p2p protokolov	○ Iptables / netfilter
IPP2P	◆ Jednoduchý, prehľadný kód	○ „Hardwired“ detekcia
Ourmon	◆ Vlastná (procesorovo efektívna) definícia tokov	○ Ďalšie spracovanie, GUI ○ PCRE (pomalé)
HiPPIE	◆ Možnosť pripojenia ďalšej aplikácie spracúvajúcej výstup	○ Bez dostatočnej dokumentácie
Snort, Hogwatch	◆ libpcap	○ IDS / IPS

Tabuľka 4.1: Porovnanie s existujúcimi projektmi.

4.2 Návrh programu

Návrh detekčného systému bol spracovaný do tabuľky:

Vlastnosť	Voľba	Dôvod
Programovací jazyk a prekladač	ISO C99 Prekladač gcc verzie 4.2.3.	◆ ľahká prenositeľnosť kódu ◆ knižnica pcap je písaná tiež v C a nepotrebuje žiadne ďalšie wrappery.
Komentovací systém	Doxygen verzie 1.5.3 s generátorom schém závislostí Komentáre v štýle QT	◆ kvalitný dokumentovací systém
Zachytávanie paketov	Knižnica libpcap	◆ BPF filtre aj s kompilátorom ◆ dobre zdokumentované API ◆ podpora offline čítania zo súbora ◆ rýchlosť

Spracovanie offline zachytených paketov	Čistý pcap formát.	◆ kompatibilita s Tcpdumpom, Wiresharkom atď.
Primárne filtrovanie paketov	Jeden, hlavný filter BPF. Preddefinované subvýrazy spolu so subvýrazmi generovanými zo súbora pravidiel a užívateľsky definovanými subvýrazmi.	◆ využitie vlastností libpcapu
Rozlišovanie podľa transportnej vrstvy a portov	Testovanie vstupujúcich paketov na port, podľa zoznamu pravidiel. Podľa výsledku sa (ne) rozpoznáva daný vzor komunikácie. Indexácia podľa transportnej vrstvy.	◆ zníženie počtu vyhodnotení reg. výrazov
Identifikácia protokolov	POSIXové ERE pomocou knižnice regex pre každú množinu typov komunikácií p2p protokolu zvlášť	◆ dostupné, relatívne rýchle ◆ vláknovo bezpečné pre budúce možné rozšírenia
Sledovanie tokov	Primitívna, ale rýchla forma, podobná obvyklému sledovaniu UDP tokov. Toky sú obojsmerné.	◆ nie je treba prezerať spojenie ustanovujúce / rušiace pakety (SYN, FIN, RST atď.) ◆ expirácie (timeouty)
Výstup	Pravidelný výstup za konštantný čas ak za tú dobu prejde filtrom aspoň minimálne (preddefinované) množstvo paketov (inak by boli výstupy prázdne). Formát výstupu viď. príloha B.	◆ pripojenie spracúvajúcej časti na výstup detektora, napríklad jednoduchý shell skript parsujúci výstup, nastavujúc napríklad QoS tokom s detekovanou p2p premávkou.

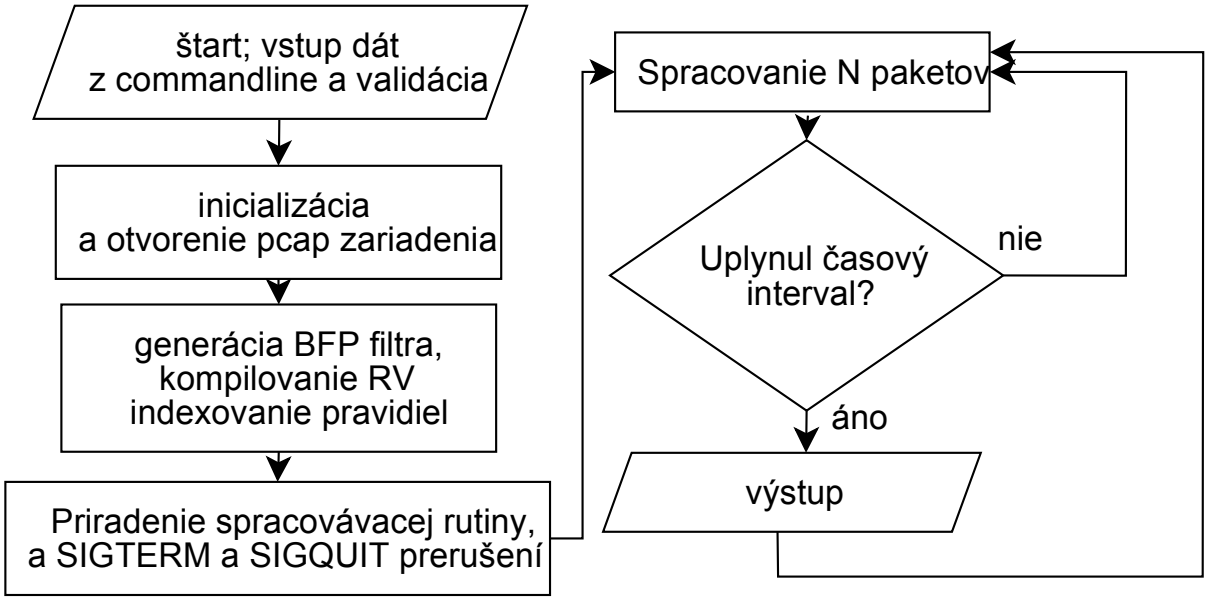
Tabuľka 4.2: Návrh systému.

Snahou je využiť výhody platformy PC – plnohodnotné regulárne výrazy – (z hľadiska budúceho možného rozšírenia sú výhodnejšie než napevno naprogramované funkcie) a vysoký výkon procesora oproti prepracovanej paralelizácii HW riešení. Celý program sa bude skladať z inicializácie a nekonečnej slučky (plus povinné „upratovanie“ pri požiadavke o skončenie).

Pozor! Aj keď je syntax regulárnych výrazov identifikujúcich protokol podobná L7-filtru, je tu rozdiel v tom, že **navrhovaný systém je citlivý na veľkosť písmen (case sensitive)**, preto prebrané výrazy z L7-filtra nemusia fungovať, ak toto nerešpektujú. Túto vlastnosť autor navrhol pre jej oveľa nižšie množstvo falošných pozitív vzhľadom na to, že sa kontrolujú všetky vyhovujúce pakety, nielen prvých *n*.

4.3 Implementácia

Implementovaný program je spustiteľný ako akýkoľvek iný program, však vyžaduje zvýšené práva (root). Bez tohto práva totiž knižnica libpcap nedokáže pristupovať k ovládačom sieťových rozhraní, ani žiadne nedokáže detekovať a program skončí s hlásením o tejto chybe.



Obrázok 4.3: Hrubá schéma systému, bez ukončovacích procedúr.

4.3.1 Pcap BPF filter

Syntax základných výrazov pcap BPF filtra je nasledujúca:

Syntax	Popis	Syntax	Popis
`host` *host*	*host* je buď IP alebo hostname	`ip`	odpovedá protokolu, ktorý chceme filtrom prepúšťať
`dst host` *host*		`arp`	
`src host` *host*		`rarp`	
`net` *net*	*net* je IP siete	`tcp`	
`src net` *net*		`udp`	
`dst net` *net*		`icmp`	

Tabuľka 4.4: Syntax pcap BPF filtra.

Kombináciou základných výrazov host filteringu so slovom `ether` pred výrazom dosiahneme filtrovanie na L2 vrstve. Základné výrazy môžeme kombinovať operátormi `and`, `or`, `not` a je možné použiť zátvorkovanie.

Zápis výrazov porovnávajúcich dáta je nasledujúci:

protokol [*ofset v bajtoch od začiatku hlavičky protokolu* : *počet bajtov pre porovnanie*]

Za týmto zápisom nasleduje jeden z porovnávacích operátorov >, <, >=, <=, =, != a porovnávaná hodnota (ktorá môže byť aj v hexadecimálnom formáte – v tom prípade má prefix `0x`), napríklad `tcp[20:2] = 0x2020` značí filter, ktorý prepúšťa tcp pakety také, že prvé dva znaky payloadu sú medzery (ak je TCP hlavička dlhá 20 znakov).

Je možné použiť základné binárne (&, |, !) aj aritmetické (+, =, *, /) operácie. Napríklad `(tcp[13] & 0x02) = 2` prepustí pakety, ktoré majú nastavený SYN flag.

Poznámka: Výraz nie je citlivý na veľkosť písmen. Návrh filtrov pre pcap je dobre zdokumentovaný v literatúre [17].

Hlavný filter

Pcap BPF filter je prvým filtrom detektora, vygenerovaným po spustení programu na základe pevne definovanej súčasti, užívateľského vstupu a BPF reťazcov v definíciách. Slúži na základné prefiltrovanie prenášaných dát, ktoré by mohli byť p2p premávkou.

Ešte pred zachytávaním paketov je výraz skompilovaný a nahraný do BPF jednotky knižnice pcap a ďalej ho už program-detektor nemusí používať. Jeho základná syntax pre naše potreby bude nasledovná:

<pevne definovaný filter> AND (<užívateľsky definovaný výraz> OR <BPF reťazec pre prvý protokol> … **OR <BPF reťazec pre posledný protokol>)**

Kde `<názov>` značí subvýrazy:

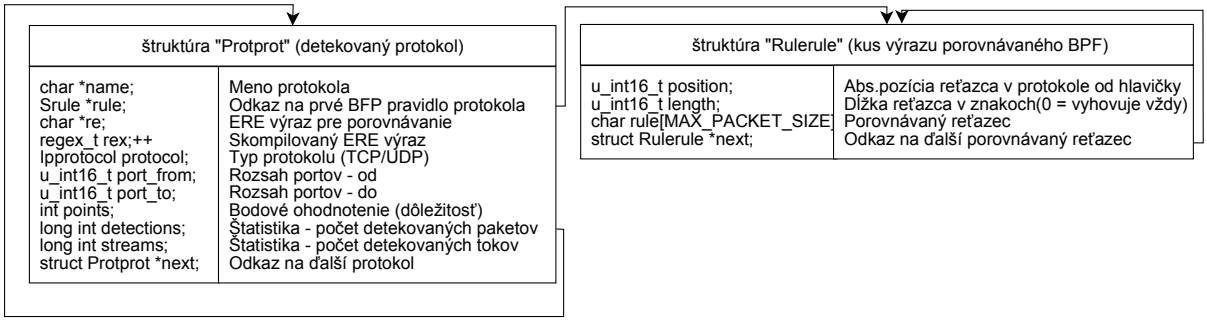
`<pevne definovaný filter>` - je preddefinovaný filter `BASIC_FILTERSTRING`. Filtruje premávku, ktorú nepotrebuje detektor spracúvať ako všeobecné adresy (broadcasty), skupinové adresy (multicasty), iná než IP premávka a tak ďalej.

`<užívateľsky definovaný výraz>` - možno nepovinne zadať ako vstup programu. Môžeme ho dodať, ak chceme rozšíriť množinu spracovávaných paketov.

`<BPF reťazec pre protokol>` - definovaný vo vstupnom súbore pre každý protokol resp. typ komunikácie zvlášť. Vymaskuje samotnú komunikáciu (payload) príznačnú pre tento protokol a prepúšta istý rozsah portov).

4.3.2 Pravidlá a protokoly

Načítané pravidlá zo súbora budeme uchovávať v nasledujúcej štruktúre:



Obrázok 4.5: Štruktúra uložených pravidiel.

Štruktúra `Protprot` popisuje jeden typ komunikácie – a to môžu byť buď všetky reg. výrazy s filtrami portov, ktorými možno protokol detekovať, alebo ich len časť. Štruktúra zaberá len málo miesta v pamäti (obvykle niekoľko stoviek bajtov, vrátane alokovaných reťazcov a skompilovaného regulárneho výrazu) a nie je problém spravovať aj niekoľko tisíc pravidiel, no v praxi nie je potrebné zapojiť viac ako 10 – 15 pravidiel. Tento zoznam zaindexujeme podľa typu transportnej vrstvy čo vzhľadom na počet pravidiel bohato stačí.

Porovnávaný reťazec `char rule` v štruktúre `Rulerule` je definovaný staticky, keďže maximálna dĺžka paketu je pretože sa táto štruktúra môže využiť nielen pri tvorbe BPF výrazu pre libpcap, ale aj pri samotnom procese detekcie.

Upozornenie: Názov štruktúry `Protprot` reprezentujúcej detekovaný protokol môže byť na prvý pohľad zavádzajúci. Pôvodne bolo počítané s myšlienkou: Niekoľko BPF reťazcov + ERE výraz = 1 detekovaný protokol. Avšak môžeme si totiž nadefinovať viac takýchto „protokolov“ pre detekovanie skutočného p2p protokolu – a každý z týchto „protokolov“ môže detekovať jeden typ komunikácie toho istého protokolu. Tento pôvodný návrh nakoniec nebol prepracovaný, pretože keď sa nad tým zamyslíme, detektoru je jedno či ide o ten istý protokol, alebo ide o iný, keď pravidlá, porty aj výrazy sú úplne iné. Nie je teda potrebné nadefinovať ďalšiu štruktúru, ktorá by uzatvárala takéto pravidlá (definície typov komunikácií) do protokolov – spoločný by totiž mali len názov – a aj ten dokonca môžeme pre každý typ komunikácie (štruktúru `Protprot`) definovať iný.

4.3.3 Regulárne výrazy

Regulárne výrazy sú posúvané knižnici regexp bez úprav (okrem nutných zmien zaisťujúcich kompatibilitu s výrazmi L7-filtra) a očakáva sa len booleovský výsledok. Filter je case sensitive a začiatok a koniec obsahu paketu je považovaný za začiatok a koniec výrazu, nad ktorým prebieha porovnávanie pomocou predkompilovaného RV.

4.3.4 Toky

V systéme je implementovaná interná, primitívna správa tokov (na ktoré sa bude autor ďalej odkazovať), nekorešpondujúca so skutočnými TCP tokmi. Dôvodom je jednak maximalizácia rýchlosti a fakt, že niektoré siete si využívajú aj UDP na prenos dát.

Každý paket je preskúmaný - aj keď už možno patrí do niektorého identifikovaného toku. Zvyšuje to síce nároky na výpočetný výkon (napriek tomu zostáva stále pomerne vysoký), ale prístup má výhodu v tom, že potlačuje dôsledky možnej chybnej identifikácie (omylu). Umožňuje to lepšiu identifikáciu protokolu (mnoho sietí prenáša detekovateľné reťazce viackrát v jednom toku, napríklad Direct Connect) alebo preklasifikovanie toku na tok s viacerými vzormi komunikácie (viacprotokolový). Tu sa uplatní bodovací systém (score) tokov – podľa detekovaného paketa, náležiaceho určitému vzoru komunikácie (protokolu) sa inkrementuje obodovanie toku.

Pre každý pozitívne identifikovaný paket prejdeme náš zoznam identifikovaných tokov p2p sietí, či sa tam tok s rovnakou s cieľovou, zdrojovou IP a portom už nenachádza.

Možno čitateľa napadne – čo ak sú takýchto identifikovaných tokov tisíce – bude sa pre každý takto rozpoznaný paket prechádzať celý zoznam? Odpoveď je: áno bude. Ale nie je to také hrozné. Identifikácia prebieha obvykle na začiatku prenosu pri zostavovaní prenosu (handshake), alebo pri špeciálnych akciách ako sú výmeny metainformácií medzi účastníkmi či medzi účastníkom a serverom, takže len jeden z niekoľkých tisíc packetov vyvolá takúto akciu, a aj to je významné len vtedy, ak je tokov naozaj veľa.

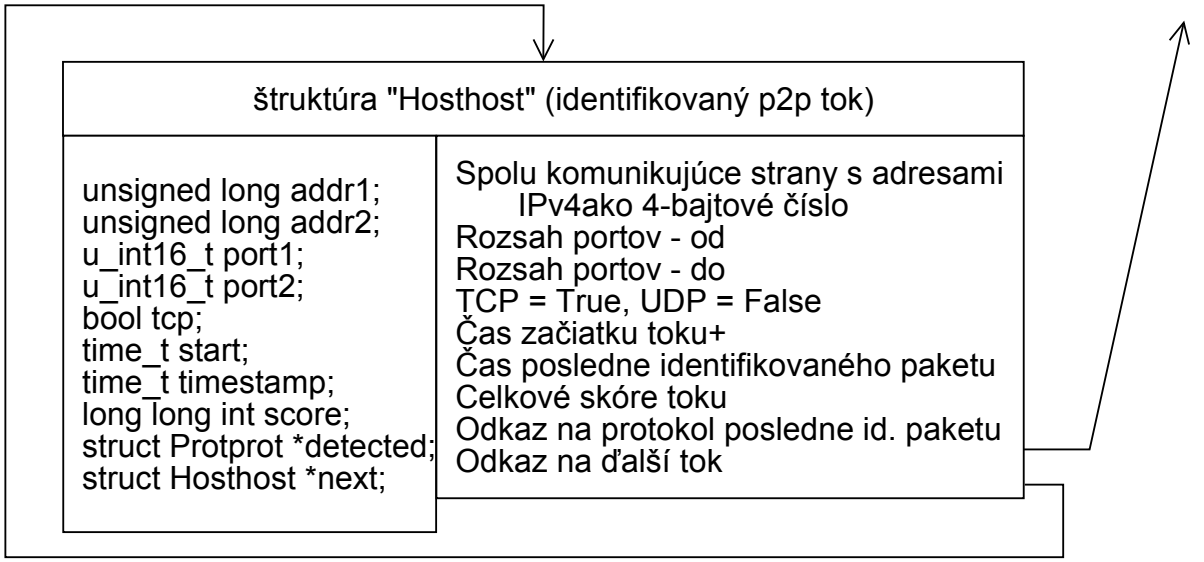
A keď už tým zoznamom tokov prechádzame, vyhadzujeme aspoň expirované záznamy (timeout) z tohto lineárneho zoznamu. Nevadí, že ak narazíme na tok, do ktorého identifikovaný paket náleží, nepokračujeme ďalej. Zoznam je teraz do tohto miesta prečistený a nasledujúce miesta nás netrápia, ledaže by sme sa k nim prepracovali pri inej príležitosti - a to by sme ho vtedy aj tak vyčistili. Keďže najnovšie toky sú pridávané na koniec zoznamu, týmto spôsobom sa nám zároveň hromadia najstaršie toky na začiatku zoznamu, kde sú oproti ostatným relatívne častejšie kontrolované na expiráciu.

Prístup má výhodu aj v tom, že periodicky nemusíme kontrolovať expiráciu (stačí zoznam skontrolovať pri výstupe).

To, že sa pri každom rozpoznanom pakete (aj keď napríklad chodia často, čo je dosť netypické) zoznam prechádza, nevadí. Zároveň to obvykle znamená, že je tokov veľa a teda ich môže byť aj veľké množstvo expirovaných.

Napriek tomu by bolo ľahké zaviesť indexáciu zoznamu tokov, napríklad podľa niektorého bajtu ľubovoľnej IP alebo portu (a zase naopak, táto indexácia môže proces trošku zdržať v prípade že je tokov málo) - najvhodnejšie bude zaviesť index na posledný bajt IP adresy, čo zaručí pomerne rovnomerné rozloženie v blokoch nezhlukovaného (non-clustered) indexu. Zhlukovaný index by vyžadoval časovo náročné triedenie často sa meniaceho zoznamu tokov.

Tok uložíme do nasledujúcej štruktúry:



Obrázok 4.6: Štruktúra informácií o identifikovanom p2p toku.

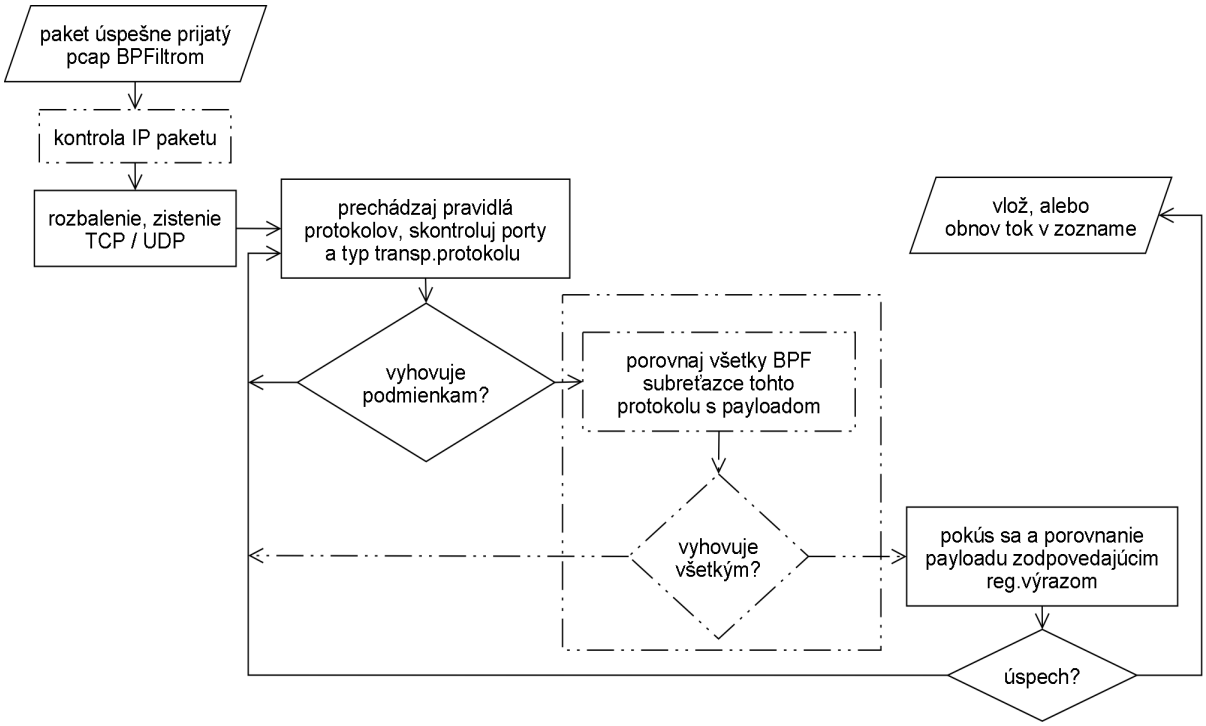
4.3.5 Princíp detekcie

Samotná detekcia je viacstupňová:

1. Najprv musí prejsť paket BPF filtrom (dodaný detektorom pri inicializácii), ktorý implementuje rovno libpcap na nízkej úrovni. Toto odfiltruje nezaujímavé pakety, napríklad také, ktoré sú určené viacerým účastníkom (broadcasty, multicasty) na sieťovej či linkovej vrstve a všetky pakety ktoré nie sú typu IP. Filter ďalej zabezpečí požadovaný smer odchytávanej komunikácie, prípadne bránu, cez ktorú musí komunikácia viesť.
2. Detektor dostane pcap hlavičku obsahujúcu dĺžku a časový údaj paketu a odkaz naň. Teraz prebehne kontrola ako napr. či ide skutočne o IP paket, prípadne by sa mohla skontrolovať checksum atd... avšak takéto kontroly sú nepovinné a zdržujú, preto je vhodné ich vypnúť (dodať do kódu #define ULTRAFAST). V každom prípade, určíme aspoň typ protokolu transportnej vrstvy (TCP / UDP).
3. Funkcia `getpacket`, ktorá paket zachytila prechádza lineárnym zoznamom vzorov komunikácií (štruktúra `Protprot`) indexovaným podľa transportnej vrstvy a ďalej spracúva len tie, kde rozsah portov a transportná vrstva odpovedá. Táto funkcia z výkonnostných dôvodov zabezpečuje úplne celý proces rozpoznávania (okrem externej funkcie `regexec` implementujúcej POSIX ERE).
4. V nasledujúcom kroku, voliteľne, by mohla prebehnúť rýchla kontrola známych fixných reťazcov (ktoré boli zoskladané do BPF filtra pri štarte) a takto pomôcť rozlíšiť protokol (keďže pcap filter prepustil všetky pakety ktoré filtru vyhovujú, avšak detektor nevie, ku

ktorému protokolu sa viažu), aby pomalší parser (prechádzací automat) s regulárnym výrazom nedostal pakety, ktoré evidentne nie sú skúmaným p2p protokolom. Túto koncepciu som ďalej testoval len málo, a do záverečnej implementácie som ju nezahrnul, pretože pohľadom do databázy p2p protokolov napr. klasifikátora L7-filter zistíme, že bežné detekčné regulárne výrazy nebývajú príliš zložité a množina vzorov komunikácie s fixnými reťazcami je často podmnožinu možností danú inými pohyblivými reťazcami, takže je v drtivej väčšine prípadov zbytočné porovnávať reťazce viackrát. Tento člen je v diagrame postupu znázornený bodkočiarkovanými čiarami.

5. Nasleduje kontrola POSIX ERE, predkompilovaným pre predpokladaný (práve testovaný) protokol.
6. V prípade že bol test úspešný, je založený nový tok v internom formáte, alebo je obnovený tok už existujúci. Inšpekcia paketa tým končí.



Obrázok 4.7: Postup pri identifikácii toku.

4.4 Prehľad vstupných parametrov

Program je použiteľný s nasledujúcimi parametrami:

```
p2pdetect [-d <device name or filename>] [-f
    <protocols_definition_file>] [-r <results_rate>] [-a <out_file>]
    [-u <user_BPF_expression_extension>] [-g <gateway_IP or
    gateway_hostname> -G <gateway_MAC or gateway_ethername>] [-i |
    -o] [-v]
```

Prepínač	Význam
-d <*názov zariadenia alebo súbora*>	Použije dané sieťové zariadenie alebo súbor vo formáte pcap (automaticky rozoznáva). Možno použiť aj **any** pre čítanie zo všetkých dostupných sieťových zariadení, alebo **lo** pre loopback.
-g <*gateway IP alebo gateway hostname*> -G <*gateway MAC alebo gateway ethername*>	Bude zachytávať len premávku prechádzajúcu cez zadaný uzol. Treba špecifikovať oboje.
-f <*súbor s definíciami pravidiel*>	Vstupný súbor s pravidlami. V prípade nezadania sa hľadá súbor **rules.dat**
-r <*frekvencia výstupu v sekundách*>	Špecifikuje, ako často sa má posielať tabuľka detekovaných tokov na výstup.
-a <*výstupný súbor*>	Špecifikuje súbor pre výstup, kam sa budú posielať len výsledky detekcie, očistené o hlásenia programu. Štandartne sa jedná o stdin.
-u <*užívateľský, rozširujúci BPF výraz*>	Rozširuje množinu paketov vyhovujúcich filtru dodaním užívateľského BPF výrazu v pcap syntaxi
-i alebo -o	Špecifikuje smer zachytávania packetov. Je možné použiť len jeden z prepínačov. V prípade nezadania sa zachytáva v oboch smeroch. Ak je čítané zo súbora, prepínač bude ignorovaný.
-v	Zapína možnosť rozšíreného hlásenia a debug informácií programu.

Tabuľka 4.8: Prehľad vstupných parametrov.

4.5 Poznatky a problémy implementácie

Nasleduje opis niektorých problémov či zaujímavostí, s ktorými sa autor stretol, avšak žiaden z nich závažným spôsobom neznehodnocuje funkcionalitu systému.

1. Bolo treba vyjadriť v pcap syntaxi filtra ofset porovnávaného reťazca payloadu TCP paketov, keďže pcap v súčasnej implementácii má podporu dynamického vypočtu ofsetu pre linkové a sieťové protokoly, avšak nie pre transportné. UDP pakety boli jednoduchšie, keďže dĺžka ich hlavičky je 8 bajtov, tak stačilo pripočítať túto hodnotu k ofsetu IP payloadu. Avšak dĺžka hlavičky TCP paketu môže byť 20 až 60 bajtov v násobkoch 4, ju vypočítať, takže filter dostal podobu `tcp[((tcp[12] & 0xf0) / 4) + ofset : dĺžka ]`. Obával som sa, že tento zápis zdrží filtrovanie, čo sa čiastočne (avšak nie výrazne) potvrdilo, no inej cesty niet a spoliehanie sa na obvyklé 20-bajtové hlavičky znamená stratu časti paketov.
2. Funkcia `pcap_dispatch` knižnice pcap sa správa zvláštne na niektorých systémoch. Jej funkcia je spracovať n paketov alebo skončiť v stanovenom čase, no časovač skončenia však nie vždy funguje. Presnú príčinu sa mi nepodarilo vypátrať, súvisí pravdepodobne s implementáciou hodín v systéme. To sa v programe môže prejaviť tak, že ak do detektora prichádza veľmi málo paketov, zobrazenie výstupu môže meškať. Našťastie, tento prípad nastáva vzácne.
3. Pri detekcii zo súboru beží reálny čas pre expirovanie tokov. To znamená, že záznamy z časovo dlhého snímaného obdobia expirujú až keď prejde skutočná doba expirácie, nie keď „by vtedy mali“. Toto môže skresliť testy rýchlostí veľmi dlhých záznamov s intenzívnou p2p prevádzkou (pribúdajúce množstvo tokov), na druhej strane výsledný zoznam je dobrým prehľadom všetkých detekovaných tokov (nie len aktuálnych v čase skončenia) čo je obvykle žiadúce a preto nebola zmena spočívajúca v zámene sledovania systémového času za čas posledného skúmaného paketu prevedená.

4.6 Tvorba sady pravidiel

Pri tvorbe pravidiel autor vychádzal z definícií protokolov L7-filtra zo dňa 20.2.2008. Výrazy boli pre BitTorrent a Direct Connect na základe skúseností z pozorovaní upravené. Ostatné výrazy pre p2p protokoly neboli testované – sú bez zmien prevzaté s tým, že žiadne predfiltrovanie pre ne nebolo stanovené. Formát vstupného súbora navrhovaného systému najde čitateľ v prílohe A.

V prípade BitTorrentu nevyužijeme BPF porovnávanie reťazcov, prípad Direct Connectu však ukáže jeho výhody.

4.6.1 BitTorrent

Tento výraz pre L7-filter bol rozšírený o ďalšie vzory a rozdelený na 2 časti (vzory komunikácie).

^(\x13bittorrent protocol|azver\x01$|get /scrape\?info_hash=)|
d1:ad2:id20:|\x08'7P\)[RP]

Prvý sa týka samotných prenosov, ponúkania a požadovania častí súborov po ostatných účastníkoch. Tieto prenosy, aj keď v majorite prípadov sú cielené na port 6881-6889, môžu využívať teoreticky ľubovoľný port na TCP spojení. Pravidlo pre navrhnutý systém bude vyzerať takto:

BitTorrent Communication

```
BitTorrent;0
BitTorrent;Protocol=TCP
BitTorrent;exp=^(\x13BitTorrent protocol|azver\x01$|GET /scrape
\?info_hash=|GET /announce\?peer_id=)|\x08'7P\)
```

Druhá časť sa týka len UDP prenosov – ide o požiadavku na odozvu a samotnú odozvu, ktorá sa uskutočňuje cez UDP – aby sa nemusela sledovať všetka premávka:

BitTorrent Ping

```
BitTorrent Ping;0
BitTorrent Ping;Protocol=UDP
BitTorrent Ping;exp=d1:ad2:id20:|d1:rd2:id20:
```

4.6.2 Direct Connect

Na tomto príklade si ukážeme užitočnosť porovnávania BPF filtrom. Výraz pre L7-filter

```
^(\$mynick |\$lock |\$key )
```

Bol upravený a prepísaný do tvaru vhodného pre navrhovaný systém

```
DirectConnect HubComm;0
DirectConnect HubComm;exp=^\\x24(Supports |Key |Mynick |Lock )
```

Je možné ho prepísať aj tak, že presunieme porovnávanie reťazcov na konštantných pozíciách z regulárneho výrazu do BPF filtra, čím celý proces urýchlime, ako uvidíme v nasledujúcej kapitole.

```
DirectConnect1;0;$Supports[medzera]
DirectConnect2;0;$Key[medzera]
DirectConnect3;0;$Mynick[medzera]
DirectConnect4;0;$Lock[medzera]
```

V prípade že nedetekujeme už žiadne iné protokoly, nie je treba definovať regulárny výraz. Avšak ak hodláme detekovať aj protokoly, ktorých definície spôsobia prijatie všetkých TCP / UDP paketov, tento prípad nie je možný.

5 Testovanie implementovaného detektora

5.1 Testovacia množina

Testovanie detektora založeného na inšpekcii paketov je chúlostivá záležitosť, vzhľadom na dôvernosť prenášaných dát. Prezentované výsledky sa týkajú „offline“ testovania zo súborov, aby bolo možné porovnať rôzne prístupy na rovnakej sade dát a otestovať maximálnu priepustnosť. Množstvo dát bolo stanovené tak, aby bolo možné uložiť všetko do vyrovnávacej pamäte, bez nutnosti použiť pomalé úložné zariadenia. Testovanie sa sústredilo najmä na p2p sieť BitTorrent, premávka v sieti Direct Connect je zahrnutá hlavne z porovnávacích dôvodov.

Dáta boli nazbierané na Brnenskom KolejNete na koleji PPV z jednej pripojenej konečnej stanice pomocou programu Wireshark a uložené v pcap formáte. Cez 99.9% premávky bolo úmyselne vygenerovaných vďaka aplikáciám KTorrent (BitTorrent) a Valknut (Direct Connect), no vďaka promiscuous módu (čítanie všetkých prichádzajúcich paketov, i tých čo majú iný cieľ) bolo identifikovaných aj niekoľko prichádzajúcich „zatúlaných“ paketov, neurčených pripojenej stanici, ktoré dokonca boli s vysokou pravdepobnosťou súčasťou premávky p2p sietí (BitTorrentu).

	Testovacia množina 1	**Testovacia množina 2**	**Testovacia množina 3**
Množstvo dát	350,8 MB (202205 pak.)	372,1 MB (315487 pak.)	652, 1 MB (414772 pak.)
BitTorrent premávka (nešifrovaná)	stredná (cca 45% objemu) 2 torrenty leeching 1 torrent seeding	veľmi intenzívna (cca 95% objemu dát) 2 torrenty leeching 1 torrent seeding	žiadna
Direct Connect premávka	žiadna	ľahká (cca 1 % objemu dát), väčšina je komunikácia s hubom	žiadna
Dĺžka záznamu	30 min	15 min	15 min intenzívnej (ne-p2p) http premávky

Tabuľka 5.1: Testovacie množiny dát.

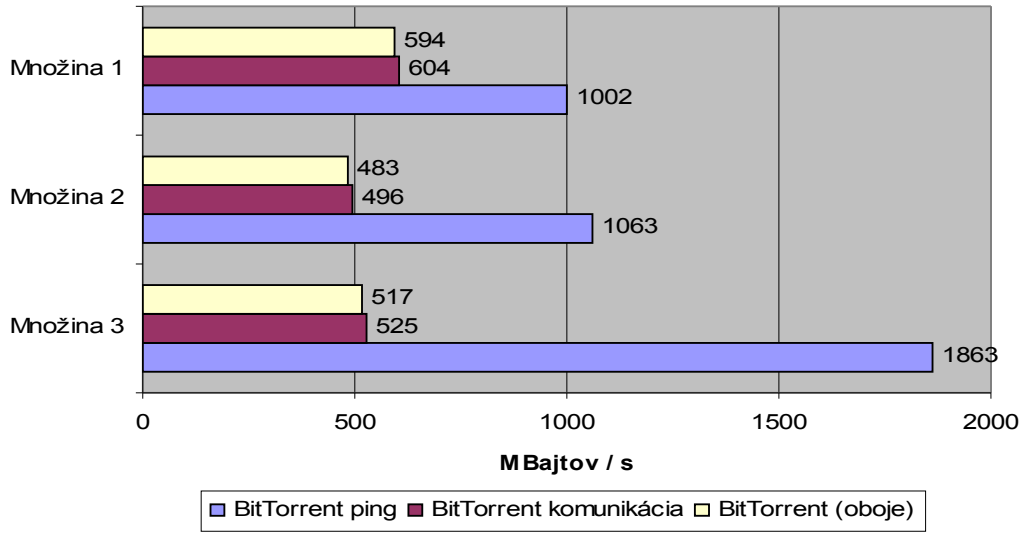
Zvyšnú časť objemu vo všetkých prípadoch tvorí podľa názoru autora typická premávka generovaná bežným koncovým užívateľom, a to najmä HTTP prenosy a malé množstvo streamovaného zvuku a instant-messaging premávky. Súčasťou je aj relatívne veľké množstvo ne-IP premávky (ARP, DNS dotazy a pod.) odpovedajúce väčšej podsieti (adresný priestor 2^{10} účastníkov).

5.2 Testovanie výkonu

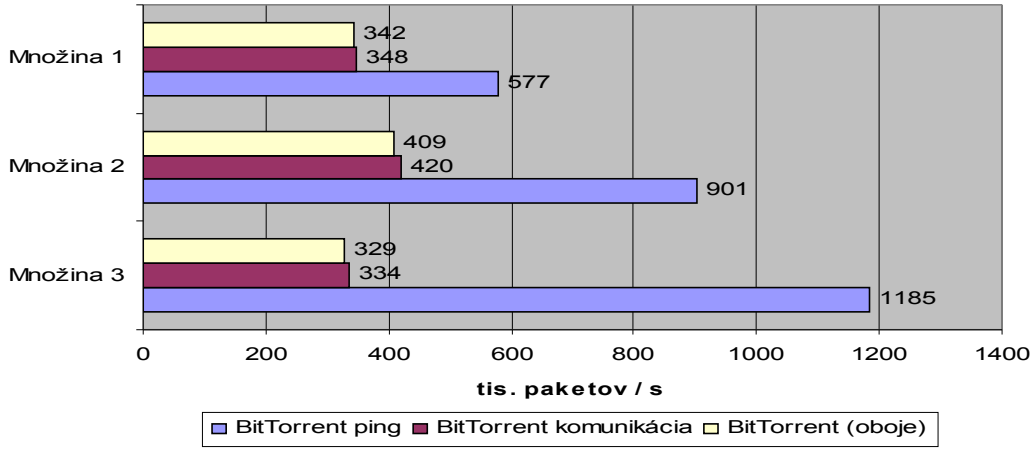
5.2.1 BitTorrent

Pri prvom pohľade na obrázky 5.3 a 5.4 je zrejmé, že detekovanie niektorých reťazcov je oveľa rýchlejšie (napriek tomu že regulárny výraz pre BitTorrent Ping nie je oveľa časovo náročnejší ako výraz pre všeobecnú komunikáciu BitTorrentu, i keď sa tak na prvý pohľad nemusí zdať – v oboch prípadoch sa musí prechádzať celý paket a v druhom prípade pevné reťazce od začiatku nijak výrazne nespomalia parsing – viď kapitola 4.6.1).

Odpoveď hľadajme najmä v tom, že vo všetkých troch množinách je UDP premávky oveľa menej ako TCP premávky (a tento pomer je sa v každej nasledujúcej množine zvyšuje).



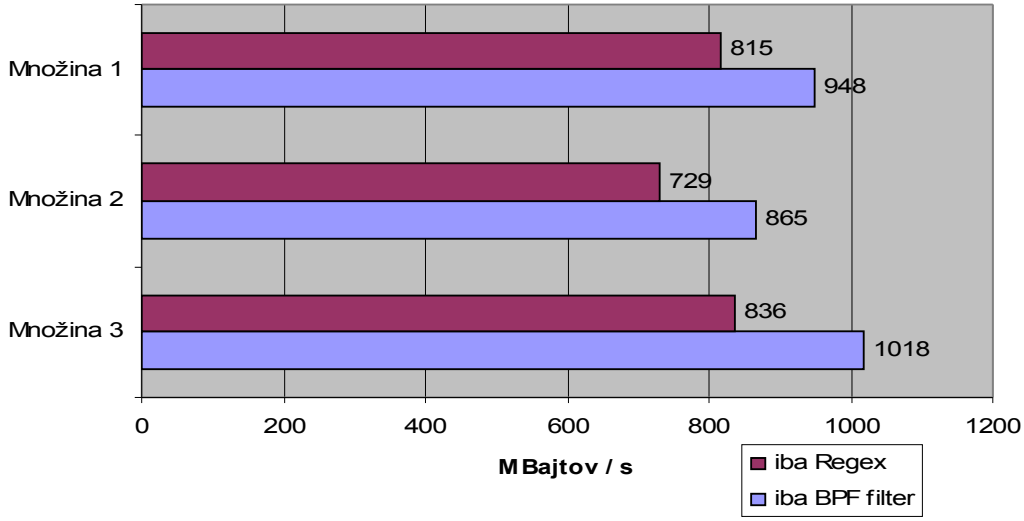
Obrázok 5.2: Hrubý výkon systému pre protokol BitTorrent v MB / s.



Obrázok 5.3: Hrubý výkon systému pre protokol BitTorrent v paketoch / s.

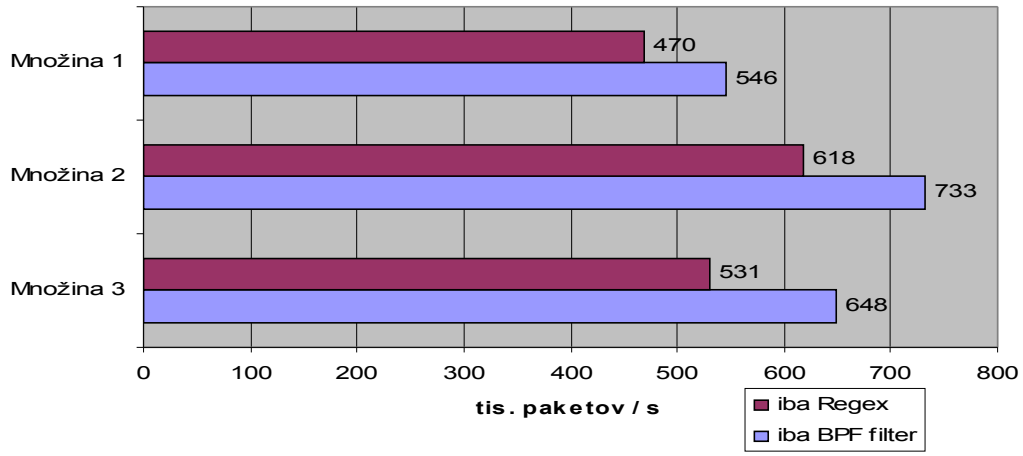
5.2.2 Direct Connect

Na príklade Direct Connectu je dobre vidieť výhody porovnávania pomocou BPF filtra (ak je to možné) oproti obvyklému regulárnemu výrazu. Detekuje sa len p2p sieť Direct Connect a celú prácu vykonáva buď len BPF filter alebo len parser regulárneho výrazu.



Obrázok 5.4: Hrubý výkon systému pre protokol Direct Connect v MB / s.

Keďže v žiadnej množine nebola vysoká Direct Connect premávka (v druhej množine len ľahká a v ostatných žiadna), preto je v meraniach mizivý vplyv množstva tokov a lepšie vynikne výkon detektora. V prípade množiny č.3, kde sú, ako vyplýva z tabuľky 5.1, prevažne väčšie pakety, bolo spracovanie dát mierne rýchlejšie – naopak množina č.2 obsahuje relatívne väčšie množstvo malých paketov, čo sa odrazilo na menšej rýchlosti dát, no väčšom množstve spracovaných paketov.

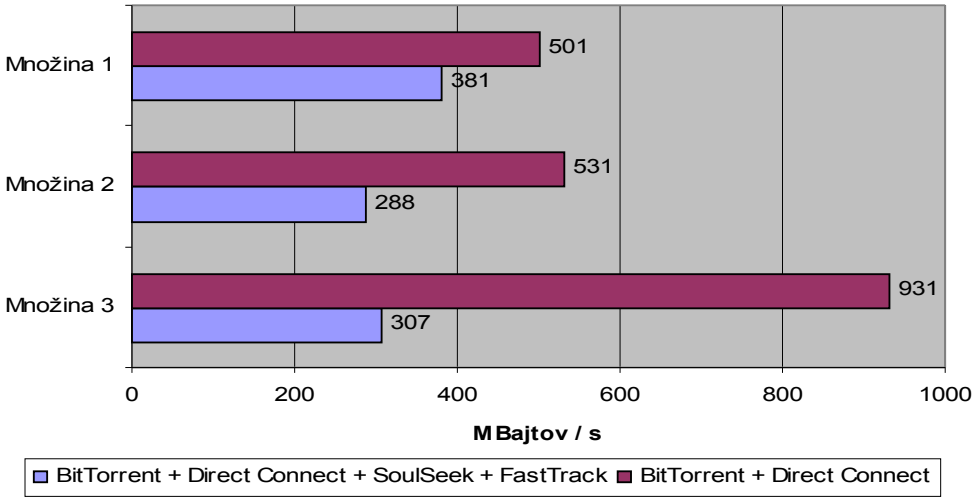


Obrázok 5.5: Hrubý výkon systému pre protokol Direct Connect v paketoch / s.

Samozrejme, v skutočnosti ak je detekovaných viacej protokolov, regulárne výrazy sa musia skoro vždy použiť na rozlíšenie protokolov. Dobrý dizajn teda presunie porovnávanie čo najväčšieho počtu spoločných vlastností do BPF filtra.

5.2.3 Kombinované testy

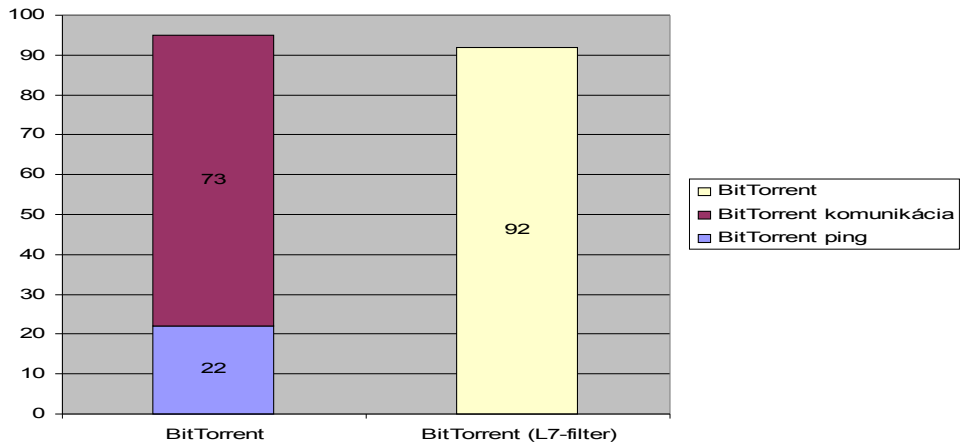
Otestované boli aj sady pre rôzne kombinácie p2p protokolov. Na obrázku 5.7 vidíme porovnanie sady vyššie popisovaných filtrov pre BitTorrent a Direct Connect (nie BPF, len reg. výraz) s rovnakou detekčnou množinou rozšírenou o výrazy pre SoulSeek a FastTrack z L7-filtra. Tieto výrazy ako príklady identifikácie p2p boli testované len málo, no bolo z nich zrejmé, že nie sú ideálne – napríklad reg. výraz pre SoulSeek falošne identifikoval 9 paketov v tretej množine ako pozitívne.



Obrázok 5.6: Hrubý výkon systému pre protokol Direct Connect v MB / s.

5.3 Testovanie spoľahlivosti detekcie

Neposlednou otázkou po úspešnej implementácii bola otázka citlivosti upravených pravidiel (koľko percent tokov sa podarilo detekovať v testovacom prostredí). Pretože sa venujeme najmä BitTorrentu, všimnime si grafické porovnanie (testované dáta pochádzali zo všetkých troch testovacích množín):



Obrázok 5.7: Porovnanie spoľahlivosti v % pre dáta zo všetkých testovacích množín.

Rozšírenie výrazu pochádzajúceho z L7-filtra sa premietlo v náraste množstva detekovaných BitTorrent p2p tokov o 3 percentá.

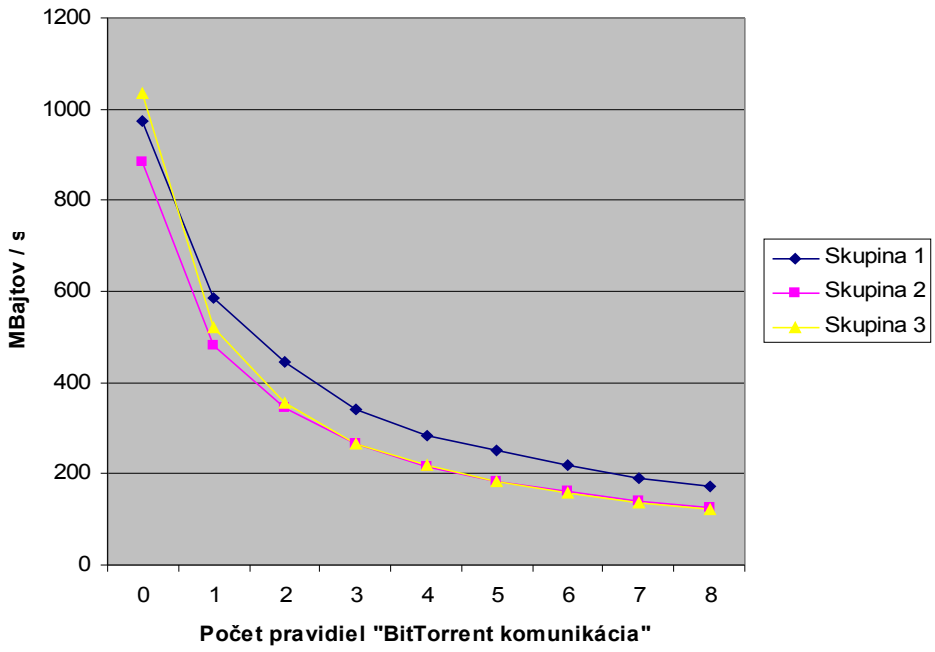
Špecificita bola v prípade BitTorrentu vo všetkých prípadoch 100 %, čo znamená že žiaden iný tok nebol chybne označený ako náležiaci BitTorrentu. Vidíme, že pravidlá sú dobre nastavené a takýto filter by bol bez problémov nasaditeľný v praxi.

Treba zdôrazniť, že šifrovaná BitTorrent premávka bola zakázaná – žiaden z diskutovaných prístupov nemá schopnosti takúto premávku detekovať.

5.4 Zhodnotenie testovania

Výsledkom implementácie je algoritmus detekčnou silou porovnateľný s L7-filtrom, či vlastne akýmkoľvek iným založeným na regulárnych výrazoch. Veľmi dôležitý je najmä vhodný návrh filtrov, ktorý je v tejto práci popísaný len zbežne.

Porovnanie rýchlosti navrhnutého systému s inými je ťažké, ba priam nemožné. Každý detektor je navrhnutý trochu ináč – jeden pracuje nad paketmi, iný nad tokmi, ďalší je funkcionálny a iný zase využíva regulárne výrazy či stromy. Môžu sa líšiť systémom zachytávania paketov aj vrstvami na ktorých pracujú.



Obrázok 5.8: Rýchlosť systému pre rôzny počet pravidiel (kópií).

Napriek tomu, výkon v rádoch stoviek megabajtov za sekundu pri použití rozumne veľkej sady (3-6) používaných protokolov autor hodnotí ako pomerne solídny. V praxi to iste záleží aj od sieťovej karty a ovládačov nízkej úrovne – predsalen, popisované testy sú syntetické. Informatívne bola otestovaná aj sada sto protokolov a výkon aj vtedy kopíruje trend naznačený grafom.

6 Záver

Na základe znalostí získaných preštudovaním problematiky p2p sietí a analýzy niektorých už existujúcich špecializovaných p2p detektorov bol navrhnutý detekčný systém založený na bežne dostupných softwarových komponentoch so zameraním pozornosti prevažne na optimalizáciu využitia jednotlivých blokov – na BPF filter a POSIX parser regulárneho výrazu.

Okrem výbornej rýchlosti, ktorá sa preukázala v testoch, je navrhnutý systém v rámci svojej kategórie detektorov založených na regulárnych výrazoch dosť flexibilný – možno jednoducho pridať nové pravidlá, existujúce rozhranie pre bezproblémové pridanie externej funkcie implementujúcej detekciu na základe obsahu paketov a výstup detekovaných tokov v definovanom formáte pre zapojenie reakčného modulu taktiež zvyšujú použiteľnosť.

Systém sa dá skompilovať a použiť na ľubovoľnom POSIX systéme – je navrhnutý tak, že stačia len základné knižnice s kompilátorom gcc a knižnica libpcap. Pre reálne nasadenie a udržovanie systému v budúcnosti je tvorba vhodných sád pravidiel nevyhnutnosťou, preto vhodným doplnkom systému by bola aplikácia extrahujúca rysy z viacerých, užívateľom označených tokov.

S minimálnymi modifikáciami možno upraviť systém tak, že bude možné detekovať protokoly na iných transportných protokoloch, dokonca aj sieťových či linkových protokoloch.

Možností rozšírenia systému je mnoho. Čo sa týka implementácie, je napríklad možné jednoduchým spôsobom zaviesť aj indexáciu tokov pre zvýšenie rýchlosti na sieťach s vyšším množstvom tokov, vymeniť POSIX regulárne výrazy za PCRE pre pohodlnejší zápis RV, či umožniť multithreadové porovnávanie pomocou poolu pracovných vlákien s vlastnými RV, keďže knižnice pcap aj regexp sú vláknovo bezpečné. Zložitejšia, avšak prínosná alternatíva je navrhnúť kompilátor programu pre BPF filter z dostupných informácií o p2p protokoloch. Predsalen, pcap syntax je univerzálna a pre naše účely vhodná len s obmedzeniami. Implementácia v C taktiež priamo ponúka možnosť využita najrôznejších platforiem - dedikovaných HW akcelerátorov, alebo, zaujímavou a v súčasnej dobe autorom intenzívne skúmanou možnosťou je implementovať detektor na GPGPU (General-Purpose computation on GPUs). Dnešné grafické akcelerátory môžu PCI Express rozhraním prenášať niekoľko GB/s, majú univerzálnu sadu inštrukcií a existuje viacero dobrých rozhraní pre podporu výpočtu na GPU (napríklad CUDA - Compute Unified Device Architecture). Je však možné že všetky výpočetné jednotky GPU (stovky) sa nepodarí využiť a fakt, že jeden program je vykonávaný všetkými výpočetnými jednotkami súčasne (jediný inštrukčný ukazateľ) pravdepodobne zapríčiní zmenu architektúry tak, že program bude spracúvať všetky porovnávané výrazy naraz periodicky nad dostatočne veľkou množinou zozbieraných dát – a tvorba takého vhodného prekladača bude k tomu kľúčom.

Literatúra

[1] Androutsellis-Theotokis, S., Spinellis, D.: *A survery of peer-to-peer content distribution technologies.* ACM Computing Surveys, 36(4):335–371, december 2004. Dokument dostupný na URL http://www.spinellis.gr/pubs/jrnl/2004-ACMCS-p2p/html/AS04.html (január 2008).

[2] Anderson, N.: *Nocturnal P2P transmissions account for 95 percent of Internet traffic.* 28. novembra 2007. Článok dostupný na URL http://arstechnica.com/news.ars/post/20071128-nocturnal-p2p-transmissions-account-for-95-percent-of-internet-bandwidth.html (marec 2008).

[3] Štúdia spoločnosti Ipoque: *Internet study 2007.* Dokument dostupný na URL: http://www.ipoque.com/news_&_events/internet_studies/internet_study_2007 (apríl 2008).

[4] Hanker, Filip: *Provideri proti ťahačom: FUP a shaping.* Portál Živé.sk, 1. decembra 2008. Dokument dostupný na URL http://www.zive.sk/default.aspx?article=274110 (apríl 2008).

[5] Wiki stránky projektu Azureus: *Bad ISPs.* Naposledy modifikované dňa 15. marca 2008. Dokument dostupný na URL http://www.azureuswiki.com/index.php/Bad_ISPs (apríl 2008).

[6] Wikipedia, The Free Encyclopedia: *Netflow*. Naposledy modifikované dňa 7. decembra 2007. Dokument dostupný na URL http://en.wikipedia.org/wiki/NetFlow (december 2007).

[7] Packeteer, Inc., 10201 N. De Anza Boulevard Cupertino, CA 95014, USA. *PacketShaper.* Dokument dostupný na URL http://packeteer.com/products/packetshaper (máj 2008).

[8] Ellacoya Networks, Inc., 7 Henry Clay Drive, Merrimack, NH 03054, USA. *Ellacoya e100 Platform.* Dokument dostupný na URL http://ellacoya.com/products/e100.shtml (marec 2007).

[9] WWW stránky projektu *L7-filter,* http://L7-filter.sourceforge.net (apríl 2008).

[10] Wiki stránky Protocolinfo: *List of P2P protocols*. Naposledy modifikované dňa 26. novembra 2007.Stránky dostupné na URL http://protocolinfo.org/wiki/List_of_P2P_protocols (máj 2008).

[11] The IEEE and The Open Group: *The Open Group Base Specifications Issue 6.* IEEE Std 1003.1, 2004 Edition. Chapter 9: Regular Expressions. Dokument dostupný na URL http://www.opengroup.org/onlinepubs/009695399/basedefs/xbd_chap09.html (apríl 2008).

[12] WWW stránky projektu *PP2P*, http://www.ipp2p.org (apríl 2008).

[13] WWW stránky projektu *Ourmon*, http://ourmon.sourceforge.net (apríl 2008).

[14] WWW stránky projektu *HiPPIE*, http://hippie.oofle.com (apríl 2008).

[15] Wronkowski, M.: *Linux Netfilter*. c2008. Dokument dostupný na URL http://www.csh.rit.edu/~mattw/proj/nf (máj 2008).

[16] McCanne, S., Jacobson, V.: *The BSD packet filter: A New Architecture for User-level Packet Capture.* Lawrence Berkeley Laboratory, 19. decembra 1992. Dokument dostupný na URL http://www.tcpdump.org/papers/bpf-usenix93.pdf (december 2007).

[17] Horn, M.: *Designing Capture Filters for Ethereal/Wireshark.* Dokument dostupný na URL http://home.insight.rr.com/procana (máj 2008).

Zoznam príloh

Poznámka: Obsah nosiča CD je v súbore README

Príloha A – Formát vstupného súbora

Vstupný súbor má formát bežného textového súbora. Prázdne riadky a riadky začínajúce znakom # sú ignorované. Posledný riadok tohto súbora definícií vzorov protokolov musí byť prázdny.

Riadky definujúce jeden vzor komunikácie (protokolu) musia nasledovať za sebou, tzn. nie je možné definovať viac protokolov preskakovane.

Definícia protokolu začína jedným alebo viacerými riadkami, ktoré definujú súčasti jeho BPF filtra. Ak je riadkov viac, platí medzi nimi vzťah „a“, čo značí že všetky musia byť splnené, aby paket prešiel filtrom.

<názov protokolu>;*<štartovná pozícia pre porovnávanie ekvivalencie reťazcov od počiatku obsahu paketu>*;*<samotný reťazec>*\n

napríklad

```
BitTorrent;1;BitTorrent protocol
```

Ak je štartovná pozícia 0 a toto číslo je zároveň posledným znakom v riadku, BPF filter bude prepúšťať všetky pakety.

Za tým môžu voliteľne nasledovať niektoré z riadkov

<názov protokolu>;from=*<číslo>*\n

<názov protokolu>;to=*<číslo>*\n

<názov protokolu>;protocol=*<TCP alebo UDP>*\n

kde „číslo“ znamenajá rozsah od (resp. do) portov prepúštaných paketov a „TCP alebo UDP“ typ transportnej vrstvy. Ak niektorý riadok chýba, odpovedajúce „from“ bude štandartne nastavené na 0, odpovedajúce „do“ nastavené na 65535, prípadne filter bude prepúšťať oba typy protokolov transportných vrstiev.

Záverečný riadok definície bude typu

<názov protokolu>;exp=*<POSIX regulárny výraz>*\n

Reťazec za značkou „rovná sa“ je POSIX regulárnym výrazom, ktorým sa protokol rozpoznáva.

Príkladom nech je ukážka súbora s komunikačnými vzormi pre BitTorrent:

```
BitTorrent Ping;0
BitTorrent Ping;Protocol=UDP
BitTorrent Ping;exp=d1:ad2:id20:|d1:rd2:id20:
BitTorrent;0
BitTorrent;Protocol=TCP
BitTorrent;exp=^(\x13BitTorrent protocol|azver\x01$|GET /scrape\?
```

(pokračovanie)

```
info_hash=|GET /announce\?peer_id=)|\x08'7P\)
```

Príloha B – Formát výstupu

Výstup je vo formáte tabuľky:

IP	From IP	To IP	From Port	To Port	Score	Durat	Last detected	P2P protocol
Typ transp. protokolu na IP vrstve.	Zdrojová a cieľová adresa prvého paketu detekovaného toku.		Zdrojový a cieľový port prvého paketu detekovaného toku.		Celkové bodové ohodnotenie toku.	Dĺžka toku v sek.	Čas v sek. od posledného detekovaného paketu	Názov protokolu (vzoru komunikácie)

Tabuľka B.1: Formát výstupu analýzy.

Pozn. Hlavička sa dá odkomentovaním riadka #define OUTPUT_PRINTHEADER v súbore datatypes.h odstrániť. Ak bolo v toku detekovaných viac vzorov komunikácie, položka v stĺpci „P2P protocol“ bude mať hodnotu (multiple).