VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

DETEKCE PEER-TO-PEER KOMUNIKACE

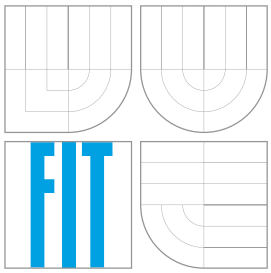
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

PAVEL LETÝ

BRNO 2014

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

DETEKCE PEER-TO-PEER KOMUNIKACE

DETECTION OF PEER-TO-PEER COMMUNICATION

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

PAVEL LETÝ

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. VÁCLAV BARTOŠ

BRNO 2014

Abstrakt

Bakalářská práce se věnuje problematice detekce P2P sítí ze síťových toků NetFlow. V teoretické části této práce jsou představeny aktuální techniky pro detekci této komunikace v síti společně s jejich výhodami a nevýhodami. Největší pozornost je věnována klasifikačnímu schématu pana Bashira, které se věnuje detekci protokolu BitTorrent a aplikace Skype z NetFlow záznamů v síti. Na základě tohoto schématu je navržen detekční modul pro modulární systém analýzy síťových dat Nemea vyvíjený organizací Cesnet. V praktické části práce je pak představena implementace tohoto modulu a jsou prezentovány výsledky experimentů nad reálnými daty.

Abstract

This thesis is focused on issues in detection of P2P network from NetFlow. In the theoretical part of this work are introduced actual techniques in detection of this communication in network. There are presented their advantages and disadvantages too. The biggest attention is focused on the classification scheme of Mr. Bashir which deals with a detection of a protocol BitTorrent and a Skype application from Netflow. Following this scheme is designed a detection module for a modular system of a traffic analysis Nemea, developed by Cesnet organization. In the practical part of this work is introduced the implementation of this module. There are also presented results of experiments with real data.

Klíčová slova

NetFlow, p2p, Nemea, detekce, modul, BitTorrent, Skype

Keywords

NetFlow, p2p, Nemea, detection, module, BitTorrent, Skype

Citace

Pavel Letý: Detekce peer-to-peer komunikace, bakalářská práce, Brno, FIT VUT v Brně, 2014

Detekce peer-to-peer komunikace

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Václava Bartoše

.....
Pavel Letý
20. května 2014

Poděkování

Rád bych poděkoval vedoucímu mé práce panu Ing. Václavu Bartošovi za poskytnutou pomoc, konzultace a trpělivost při tvorbě této bakalářské práce.

Obsah

Kapitola 1

Úvod

S narůstajícím rozvojem a popularitou P2P sítí vzniká i potřeba jejich detekce. P2P sítě dnes nacházejí využití hlavně ve sdílení souborů, ale můžeme se s nimi setkat i při sledování televize, či při využívání hlasových služeb. Jedním z hlavních důvodů, proč je nutné tuto komunikaci v síti detekovat je právě již zmiňované sdílení souborů, které může zabírat značnou část přenosového pásma a tím výrazně zpomalovat ostatní komunikaci v síti. Zvláště v menších sítích s malou šířkou přenosového pásma může být toto zpomalení velkým problémem zejména pokud jsou v sítí využívány hlasové služby.

Tato práce se zabývá detekcí vybraných P2P aplikací v síti na základě záznamů o síťových tocích NetFlow bez zásahu do datové části paketů. Tento přístup přináší výhodu v tom, že je nekonfliktní se zákonem a analýza může být poměrně rychlá. Pro detekci P2P komunikace byl vytvořen rozšiřující modul kompatibilní s modulárním systémem analýzy síťových dat Nemea, který je vyvíjen na Cesnetu. Vybrané aplikace reprezentují dvě nejznámější odvětví využití P2P sítí tj. sdílení souborů, které představuje detekce protokolu BitTorrent a detekce hlasových služeb, které představuje program Skype.

V 2. kapitole jsou popsány obecné principy P2P sítí, jejich architektura, dělení a příklady použití. Dále jsou detailně představeny protokoly BitTorrent a Skype, které jsou předmětem detekce v této práci. V 3. kapitole je popsána architektura modulárního systém pro analýzu síťových dat Nemea, pro který je tato práce navrhována. V 4. kapitole jsou představeny různé techniky pro detekci P2P komunikace v síti a také technologie NetFlow. V kapitole číslo 5 je uveden návrh modulu pro systém Nemea na základě zvolené techniky. Kapitola 6 pak pojednává o implementaci navrženého modulu. V 7. kapitole jsou uvedeny výsledky provedených experimentů nad reálnými daty ve dvou prostředích. Kapitola 8 pak zhodnocuje dosažené výsledky experimentů a navrhuje další postup vývoje a rozšíření modulu.

Kapitola 2

P2P obecně

V této kapitole bude představena architektura peer-to-peer sítí (P2P), její rozdíly oproti architektuře klient-server a proč je důležité detekovat tuto komunikaci v síti. Dále bude prezentována klasifikace z pohledu centralizace a umístění uzlů v síti. Ke konci kapitoly budou uvedeny příklady dvou nejpoužívanějších P2P aplikací BitTorrent a Skype.

2.1 Architektura P2P sítí

P2P sítě jsou typem distribuovaných systémů, kde různé uzly (anglicky peers) tvoří dynamicky překrývanou síť. Spolupracující uzly mají schopnost shromažďovat svoje zdroje tak, aby byly splněny hlavní podmínky distribuovaných systému, jako jsou škálovatelnost[1], sdílení zdrojů, tolerance chyb [3].

Architektura P2P je považovaná za opak architektury klient-server. V architektuře klient-server existuje centralizovaný prvek, kterému se říká server a klient, který komunikuje s tímto serverem. Server je zodpovědný za kontrolu přístupu ke sdíleným zdrojům, tím že udává jednotlivým klientům práva k přístupu. K serveru se připojuje mnoho klientů s požadavky na různé typy služeb (přehrávání videa, stahování souborů, webové služby). Následně jsou jednotlivé požadavky serverem obslouženy [3, 20].

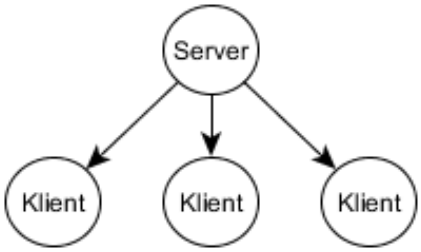
V případě čisté architektury P2P neexistuje centralizovaný prvek. Uzel plní funkci klienta i serveru zároveň. To znamená, že uzel zajišťuje serverové i klientské služby. Jedná se tedy o rovnocenné spojení mezi uzly [18]. Tento přístup s sebou přináší problémy ve správě a zabezpečení dat v rámci sítě. Spolupracující uzly se stávají náchylnými vůči různým bezpečnostním hrozbám [3]. Grafické znázornění obou architektur je vidět na obrázcích 2.1 a 2.2

Proces komunikace v sítích P2P lze rozlišit do dvou fází:

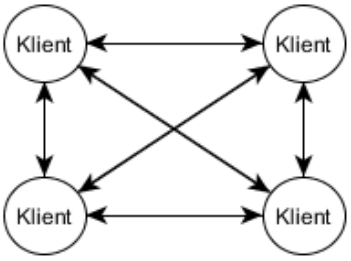
- Signalizace - během signalizační fáze klient hledá požadovaný obsah (data) a je schopen vymezit uzly, které jsou mu ochotné tyto data poskytnout. Je zde většinou použit transportní protokol UDP [14]. Obecně lze říci, že proces signalizace zahrnuje spojení a vyhledávání požadovaného obsahu [18].

- Datový přenos - fáze, kdy se kontaktuje příslušný uzel, nebo uzly, které obsahují požadované data a začne stahování. Při samotném přenosu dat je využíván transportní protokol TCP [18].

[1] Škálovatelnost je schopnost systémů, nebo sítě zvládnout narůstající množství práce schopným způsobem.



Obrázek 2.1: Architektura klient-server



Obrázek 2.2: Architektura peer-to-peer

2.2 Příklady využití P2P sítí

V dnešní době existuje nepřeberné množství P2P aplikací (uTorrent, Transmission, BitComet, BitLord). Všechny tyto aplikace jsou klienti určené primárně ke stahování souborů všech druhů (hudba, filmy, programy).

Dnes však P2P sítě najdou využití i v jiných oblastech, než je stahování souborů a to například v hlasových službách jako je služba Skype, která umožňuje zdarma volat mezi jednotlivými uživateli, nebo například sledování televize prostřednictvím programu SopCast.

2.3 Dělení P2P sítí

Z pohledu úrovně centralizace lze P2P sítě klasifikovat následujícím způsobem:

- Decentralizované architektury - v těchto sítích neexistuje žádný centralizovaný prvek, který by koordinoval činnost jednotlivých uzlů. Všechny uzly jsou rovnocenné [7].
- Částečně centralizované architektury - v sítích tohoto typu není přítomen centralizovaný prvek. Namísto centralizovaného prvku zde máme super uzly [2], které směrují požadavky mezi jednotlivými klienty. Jednotlivé uzly se většinou do P2P sítě tohoto typu hlásí skrze tyto super uzly [18].
- Hybridní decentralizované architektury - tato architektura je podobná částečně centralizované architektuře. V částečně centralizované architektuře super uzly plní funkci

[2]Super uzel (anglicky super node) je klasický uzel (např. skype klient), který má k dispozici větší systémové zdroje (výkon CPU, kapacita disku, lepší internetové připojení) oproti ostatním uzlům. Tyto uzly jsou dynamicky voleny [2].

klienta i serveru a jsou dynamicky voleny. Naproti tomu v hybridní architektuře jsou super uzly nahrazeny soustavou serverů, které jsou navzájem propojené. Tyto servery koordinují komunikaci mezi uzly a neplní funkce klienta [18].

Z pohledu lokalizace uzlů v síti:

- Tracker - v případě použití tohoto mechanismu klient kontaktuje centrálně udržovaný tracker, který obsahuje informace o uzlech majících k dispozici požadovaný soubor. Uzel je kontaktován pomocí signalizačního mechanismu uvedeného v 2.1 a to buď pomocí protokolu TCP, nebo UDP [3].
- DHT (Distributed hash Table) - jedná se o metodu ukládání kontaktních informací uzlů bez použití trackeru. Každý DHT uzel si udržuje seznam nejbližších uzlů spolu se soubory, které mohou poskytnout a má přidělené unikátní ID [11]. Nespornou výhodou DHT je fakt, že není závislé na trackeru, který může být vypnut, nebo může z nějakého důvodu zkolabovat.

2.4 BitTorrent

Bittorrent je sada protokolů a služeb, které jsou zejména používány pro distribuci/sdílení souborů. Obsah je identifikován pomocí URL. Oproti běžnému HTTP má BitTorrent výhodu v konkurentním stahování stejného souboru. Klienti si mohou poskytovat soubor mezi sebou navzájem, což umožňuje zdroji dat pokrýt relativně velký počet klientů pouze při malém zvýšení zatížení [5].

2.4.1 Popis komunikace protokolu BitTorrent

Pro zahájení stahování je nutné mít nainstalovaný BitTorentový klient, což je počítačový program. Tento klient je schopný na základě metadat uvedených v souboru `.torrent` vytvořit spojení mezi jednotlivými uzly. Soubor `.torrent` lze najít na různých webových stránkách po celém internetu. Jediné, co musí uživatel udělat je tento soubor stáhnout a spustit o vše ostatní se stará samotný klient.

Struktura `.torrent` souboru je následující:

- `announce` - obsahuje URL adresu trackeru.
- `info` - dodatečné informace pro identifikaci příslušného torrentu.
  - `name` - jméno souboru/cesta, kde je soubor uložen
  - `piece length` - velikost přenášených bloků (chunks) souborů v bytech. Zpravidla se jedná o fixní velikost. Nejčastěji 258 KiB.
  - `pieces` - obsahuje SHA-1 hash tj. kontrolní součet jednotlivých bloků.
  - `length` - velikost celého souboru v bytech

Pokud se jedná o sdílení více souborů, obsahuje tento soubor ještě navíc položku `files`, která obsahuje cesty k jednotlivým souborům a jejich délku [5].

Po otevření `.torrent` souboru začne klient komunikovat s trackerem. Obecně je tracker serverem, který asistuje (koordinuje) při komunikaci mezi jednotlivými uzly. Komunikace mezi klientem a trackerem je zajišťována prostřednictvím HTTP (port 80). Typicky klient

zašle zprávu GET, kde uvede svoji IP adresu a číslo portu na kterém bude čekat odpověď. Následně klient čeká na odpověď od trackeru, který mu pošle list s IP adresami a čísly portů, kde naslouchají ostatní uzly, které vlastní požadovaný soubor [5].

Tento poznatek je velice důležitý z hlediska detekce BitTorrentu z toho důvodu, protože nyní víme, že klient při využití trackeru musí alespoň jednou komunikovat přes HTTP na vzdálený port 80.

Krom periodické komunikace s trackerem (hlášení stavu stahování), uzly komunikují mezi sebou pomocí peer protokolu, kde si navzájem vyměňují bloky souboru, které jim chybí ke kompletaci souboru a zároveň nabízejí svoje [5].

Jiná situace nastává při využití DHT lokalizace uzlů. Postup vyhledávání bude prezentován na příkladu. Předpokládejme, že uživatel chce stáhnout nejnovější linuxovou distribuci operačního systému Fedora. BitTorentový klient se nejdříve zeptá nejbližšího sousedního uzlu (nejbližší soused je dán porovnáním unikátních ID, které má přiděleno každý DHT uzel, tento uzel se nazývá zaváděcím), který nabízí operační systémy, zda u sebe tuto distribuci nemá k dispozici. Tento uzel odpoví, že tuto konkretní distribuci nevlastní, ale že má 3 sousedy. První nabízí operační systém Windows, druhý linuxové distribuce, třetí operační systém MAC OS. BitTorrent klient tedy zvolí souseda, který vlastní linuxové distribuce. Ten mu odpoví, že danou distribuci vlastní a následně je vytvořeno TCP spojení a začne přenos dat.

Dnešní BitTorrent klienti převážně využívají pro komunikaci tracker. DHT je v těchto klientech bráno jako doplňková služba [3]. Ve většině klientů ji lze i vypnout.

2.5 Skype

Program Skype se stal jedním z nejznámějších poskytovatelů VoIP[3] na internetu. Zejména asi proto, že své služby nabízí zdarma. Poměrně jednoduše lze pořádat hovory/videohovory, případně jen chatovat s lidmi se kterými nemůžete být zrovna v kontaktu osobně [16].

Na rozdíl od klient-server modelu, který využívá signalizační protokol SIP[4], nebo H.323[5], Skype využívá pro svoji komunikaci architekturu P2P, která obsahuje tyto základní komponenty:

- Login server - každý uživatel před přístupem do sítě (po spuštění programu) se musí přihlásit vůči tomuto serveru [6].
- Skype klient - je počítač s nainstalovaným a spuštěným Skype klientem [6].
- Super uzel - super uzlem se může stát jakýkoli Skype klient, který však musí splňovat určité podmínky (rychlé připojení, neomezený přistup do internetu - mají veřejnou IP adresu). Super uzly mezi sebou provádí směrování jednotlivých požadavků klientů[2].

2.5.1 Popis komunikace protokolu Skype

Nejprve je nutné mít nainstalovaného Skype klienta. Jakmile je klient spuštěn hned se pokouší připojit k Super uzlu. Pro toto spojení využívá předdefinovaný UDP port, který ale lze změnit ve Skype klientovi. Seznam všech dostupných super uzlů společně s IP adresami

[3] VoIP(Voice over IP) umožňuje realizovat digitalizované telefonní hovory přes IP počítačovou síť [15].

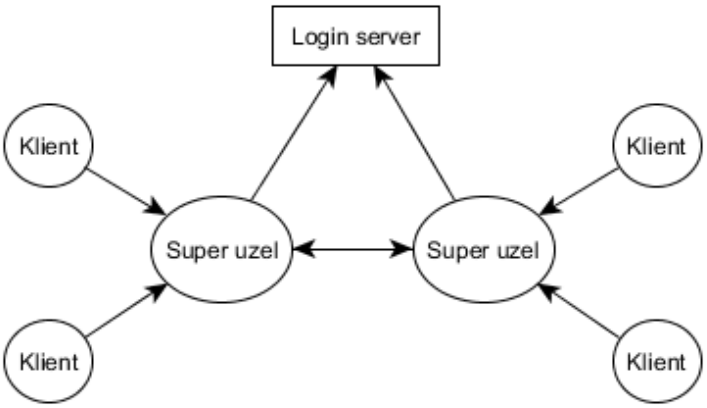
[4] Jedná se o signalizační protokol, který mimo jiné vytváří a udržuje relaci. Protokol je textový a adresování probíhá na základě SIP URI [12]

[5] Jedná se o standard ITU-T, který zahrnuje více protokolů pro směrování hovorů [12]

a portem je uložen v `host's cache`, která se nachází v instalačním adresáři klienta Skype [2].

Důležitým poznatkem pro detekci Skype komunikace je, že super uzly komunikují přes port 33033. Dále je nutné zachytit periodické dotazy (zpravidla po několika hodinách) klienta na aktuálnost jeho verze programu. Tato kontrola probíhá komunikací na vzdálený port 80 nebo 443. U klienta bude tedy muset být detekována komunikace na tyto vzdálené porty.

Po připojení k super uzlu se klient musí registrovat proti centrálnímu ověřovacímu serveru, který obsahuje databázi všech registrovaných uživatelů služby Skype a samozřejmě i jejich přihlašovací údaje. Ověřování probíhá pomocí TCP. [2].



Obrázek 2.3: Komunikace protokolu Skype

2.6 Důvody detekce BitTorrentu a Skype

Jak již bylo zmíněno stahování a zároveň distribuce dat zabírá velkou šířku přenosového pásma, což zpomaluje

Důvodem detekce Skype může být požadavek, aby komunikaci byla přidělena odpovídající priorita v sítí a byla tak zajištěna odpovídající kvalita služeb (jedná se o hlasový přenos, u kterého může docházet k ozvěně, zpoždění, rozptylu paketu, ztrátovosti) [12].

Kapitola 3

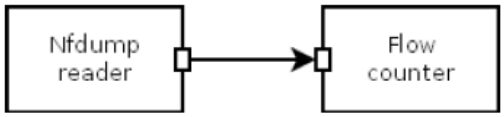
Nemea

Tato kapitola bude pojednávat o systému pro analýzu síťových dat Nemea, pro který je v rámci této práce vyvíjen rozšiřující modul na detekci P2P komunikace. Nejdříve bude prezentována architektura tohoto systému a knihovny TRAP. V další části této kapitoly budou představeny záznamy UniRec, kterými jsou reprezentovány NetFlow záznamy.

3.1 Architektura

Nemea (Network Measurements Analysis) je framework[1], pomocí kterého lze sestavit automatizovaný systém pro analyzování flow[2] záznamů v reálném čase. Tento framework je vyvíjen organizací Cesnet[3] skupinou liberouter[4] [10].

Tento systém je vybudován pomocí modulů, které jsou vůči sobě navzájem nezávislé. Každý modul je reprezentován samostatným procesem a jednotlivé moduly jsou propojeny pomocí rozhraní [1].



Obrázek 3.1: Ukázka jednoduchého Nemea systému. Převzato z [1]

Na obrázku 3.1 je vidět jednoduchý příklad Nemea systému. Systém se skládá ze dvou modulů, které jsou propojeny pomocí jednoho rozhraní. Modul `nfdump reader` čte flow záznamy ze souboru a následně je posílá modulu `flow counter`, který sečte celkový počet záznamů, packetů, bytů, které od modulu `nfdump reader` dostal.

3.2 Knihovna TRAP

Knihovna Trap implementuje komunikační rozhraní, které používají všechny Nemea moduly k výměně dat. Knihovna je reprezentována pomocí sdíleného objektu, který se nazývá `libtrap` [10].

[1] Framework je platforma, která slouží jako podpora při vyvíjení softwarových aplikací. Více viz [19]
[2] Jako flow záznamy uvažujeme cisco standard NetFlow
[3] Více viz http://www.cesnet.cz/
[4] Více viz https://www.liberouter.org/

Hlavní vlastnosti knihovny TRAP jsou [1]:

- Možnost nastavení blokujícího/neblokujícího módu přenosu pomocí těchto parametrů:
  - `TRAP_WAIT` - nastavení blokujícího přenosu. V případě odesílání zpráv z modulu systém čeká dokud není zpráva odeslána[5]. V případě příchozí komunikace systém čeká dokud data nebudou kompletně přijata[6].
  - `TRAP_NO_WAIT` - jedná se o nastavení neblokujícího přenosu. Vrací speciální návratový kód v případě, kdy nebyly přeneseny žádné zprávy.
  - `TRAP_HALF_WAIT` - jedná se o kompromis mezi předešlými možnostmi. Chová se stejně jako `TRAP_WAIT` s tím rozdílem, že neblokuje modul v případě, kdy na druhé straně žádný modul připojený není.
  - Poslední možností je zadat číselnou hodnotu doby, po kterou se má čekat v mikrosekundách. Toto nastavení zajišťuje funkce `trap_ifcctl`.
- Schopnost bufferovat záznamy pro zvýšení propustnosti
- Možnost automatického připojení po chybě.
- Možnost lokální, nebo vzdálené komunikace.

3.3 Unirec záznamy

Unirec (Unified Record) záznamy specifikuji formát zpráv, které jsou posílány přes TRAP rozhraní. Na tyto záznamy jsou kladeny požadavky v podobě flexibility, paměťové efektivity a rychlé manipulace s elementy záznamu. Tento záznam je podobný klasické struktuře, kterou známe z jazyku C - položky jsou uložené jedna za druhou.

Unirec záznam se sestává ze dvou hlavních částí:

- statická část - všechny položky mají statickou velikost
- dynamická část - položky mají dynamickou velikost a jsou uloženy na konci Unirec záznamu za statickou částí.

Moduly pracují s flow záznamy, které definovala společnost Cisco ve svém protokolu Netflow[7]. Unirec tedy obsahují elementy odpovídající položkám Netflow záznamu + spoustu dalších elementů, které jsou specifické pro jednotlivé moduly.

Při vytváření modulu je nezbytné definovat, které položky Unirec záznamu chceme v modulu používat. Soubor položek tvoří šablonu, která je specifikovaná jmény položek.

Každé TRAP rozhraní používá právě jednu šablonu (všechny zprávy přes dané rozhraní jsou posílány ve stejném formátu).

3.4 Moduly

Modul typicky funguje tak, že mu na vstup přijde posloupnost flow záznamů. Tyto záznamy zpracuje a vyhodnotí. Po vyhodnocení záznamů posílá na výstup jen záznamy, které udává funkčnost modulu. Výstupní rozhraní nemusí být definováno [1].

[5] Při odesílání zpráv je v tomto módu blokováno output rozhraní
[6] Je blokováno input rozhraní
[7] Protokol firmy Cisco pro účely monitorování provozu[4]

Například v této práci je vyvíjen modul pro detekci P2P komunikace tj. výstupem modulu jsou flow záznamy hostů, kteří vykazují P2P komunikaci. Tyto záznamy mohou být předány dalšímu modulu, který z nich např. může vytvořit statistiky.

Kapitola 4

Možnosti detekce P2P komunikace

V této kapitole budou vysvětleny současné mechanismy pro detekci P2P komunikace v síti. U každé metody budou uvedeny výhody a nevýhody tohoto přístupu.

Na konci kapitoly bude představeno klasifikační schéma podle p. Bashira, které bylo z větší části použito k implementaci detekčního modulu P2P komunikace pro systém Nemea.

4.1 Analýza potů (port based analysis)

Jedná se o nejjednodušší metodu pro detekci P2P uživatelů v síti. Metoda je založena na principu detekce předdefinovaných čísel portů, které jednotlivé P2P aplikace používají jako výchozí. Tato čísla portů jsou následně porovnávána s čísly portů, které se vyskytují na síti. Pokud se čísla portů shodují je daný uživatel podezřelý z P2P komunikace. Příklad čísel portů a jim přiřazených aplikací je znázorněn v tabulce 4.1.

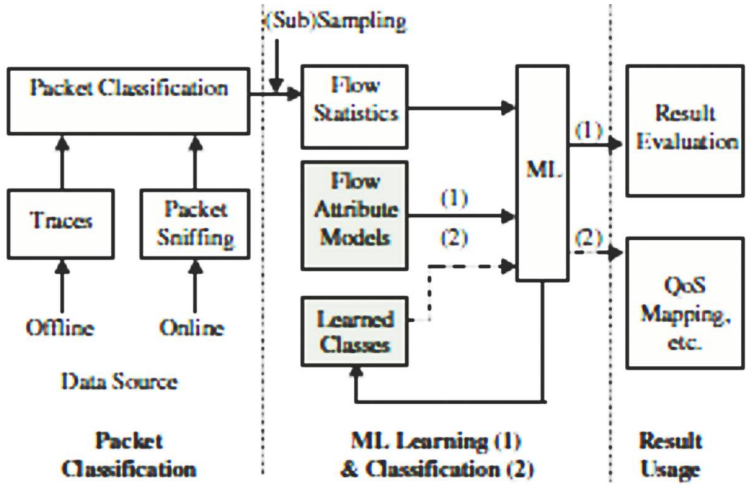
Tabulka 4.1: Seznam aplikací s předdefinovanými porty

Aplikace	Port	
	TCP	UDP
Limewire	6346	6347
Morpheus	6346	6347
BearShare	6346	6346
Edonkey	4662	-
Emule	4662	4672
Bittorrent	6881-6889	6881-6889
WinMx	6699	6257

Výhodou metody je, že se dá jednoduše použít v praxi a administrátor nemusí mít žádné speciální hardwarové vybavení. Bohužel však dnešní P2P aplikace dovolují uživateli dynamicky měnit čísla portů, nebo si je jednotlivé aplikace dokonce volí náhodně po zapnutí (čísla portů jsou nepředvídatelná). Díky tomu dnes již nelze předpokládat, že aplikace bude využívat výchozí čísla portů a to činí tuto metodu neefektivní. Na druhou stranu tato metoda může být použita pro detekci protokolů, které mají známý a neměnný port (well-known) jako je např. HTTP, FTP, ICMP, DNS [7].

4.2 Strojové učení (Machine learning)

Klasifikace probíhá na základě analýzy klíčových vlastností typických pro danou sledovanou komunikaci. Jeden z možných postupů klasifikace je znázorněn na obrázku 4.1.



Obrázek 4.1: Ukázka klasifikace toků pomocí strojového učení. Převzato z [8]

Nejdříve jsou klasifikovány pakety do toků a je spočítána jejich charakteristika. Charakteristika a vlastnosti toků jsou použity pro učební proces, kterým jsou jednotlivé toky rozřazeny do tříd (fáze (1) na obrázku č. 4.1). Jakmile je proces učení dokončen nové toky mohou být klasifikovány (fáze (2) na obrázku č. 4.1)[8].

Nevýhodou této metody je že, přesnost je závislá na datech, které algoritmus používá pro klasifikaci resp. učební proces. Další nevýhodou je existence více tříd, které mají podobné klíčové znaky. V tomto případě je obtížné přesně klasifikovat dané toky. Výhodou je rychlost klasifikace, pokud je již učební proces u konce [3].

4.3 Inspekce paketu na aplikační vrstvě (Deep packet inspection)

Metoda založená na analýze paketu na aplikační vrstvě pomocí aplikace regulárních výrazů na datovou část paketu. Tato metoda je schopná identifikovat odesílatele, příjemce, speciální klíčová slova a jiné. Nevýhodou této metody je, že vyžaduje velký výpočetní výkon a hodně času, protože každý paket musí být analyzován zvlášť a nelze ji použít, když jsou protokoly (datová část paketu) šifrované, nebo neznámé. Výhodou může být vysoká přesnost detekce v případě správného stanovení regulárních výrazů [3].

S rostoucím vývojem P2P je také nutné udržovat signatury pro regulární výrazy stále aktuální. Dalším problémem této metody je, že se snadno může dostat do konfliktu se zákonem, jelikož DPI poskytuje možnost číst, dešifrovat, filtrovat a zpožďovat internetovou komunikaci. Tím dává poskytovateli internetu (ISP) možnost kontrolovat komunikaci uživatele na internetu [17].

4.4 Analýza na základě chování uživatele (Host behaviour detection)

Podstatnou této metody je analyzovat chování jednotlivých uživatelů v síti na úrovni síťových toků (flow záznamy) a jejich statistik. Tyto záznamy lze jednoduše logovat například pomocí sond, speciálních programů, nebo směrovačů. Podobně jako u strojového učení je zde nutné pro každou aplikaci, kterou chceme detekovat vytvořit vlastní heuristiku. Tato heuristika je vytvářena na základě společných vlastností detekované aplikace. Přesnost této metody je silně závislá na stanovení této heuristiky. [3].

Tato metoda je schopna odhalit i kódové přenosy (šifrovaná datová část paketu), které používají nejnovější P2P klienti. Do datové části paketu není zasahováno, tudíž nehrozí vznik konfliktu se zákonem v souvislosti s narušením soukromí uživatelů. Nevýhodou této metody může být obtížnost stanovení správné heuristiky [9].

4.5 NetFlow

NetFlow je protokol firmy Cisco[1], který poskytuje správcům sítě možnost monitorovat provoz sítě na základě síťových toků[2]. Architekturu NetFlow tvoří [13]:

- Exportér
- Kolektor
- Komunikační protokol

4.5.1 Exportér

Jedná se o zařízení (typicky router, nebo L3 switch), sondu, či speciální software. Zařízení je připojeno na linku a monitoruje procházející provoz. Na základě tohoto provozu vytváří záznamy o tocích (Flow Records). Pokud záznam existuje tj. shoduje se ve všech položkách, které definují síťový tok, tak se záznam v Netflow cache aktualizuje. V případě, že záznam neexistuje, tak se vytváří v Netflow cache záznam nový. Toky jsou exportovány pomocí transportního protokolu UDP. Expirace (odeslání záznamu na kolektor) se děje za následujících podmínek:

- Detekce konce toku (např. TCP příznak RST, nebo FIN)
- Neaktivita toku - neaktivní timeout
- Příliš dlouhý tok - aktivní timeout
- Zaplnění paměti NetFlow cache

4.5.2 Kolektor

Hlavním úkolem kolektoru je přijímat pakety Netflow z jednoho, či více exportérů a následně tyto záznamy zpracovat. Záznamy jsou většinou ukládány do databáze, kde mohou být dále zpracovávány (grafická reprezentace v podobě grafů, tabulek, statistik), lze také provádět různé dotazy nad daty (např. kolik který klient přenesl bytů).

[1] Podrobnější informace o firmě lze najít zde: http://www.cisco.com/

[2] Síťový tok je definován jako posloupnost paketu mající společnou vlastnost a procházející bodem pozorování za určitý časový interval[4]

4.5.3 Komunikační protokol NetFlow

Protokol NetFlow tvoří hlavička a záznamy o tocích. Hlavička obsahuje obecné informace mj. verzi protokolu, počet záznamů, které se přenáší, časové razítko. Dále jsou řazeny jednotlivé záznamy o tocích. Způsob ukládání záznamů je dán verzí NetFlow protokolu. Dnes asi nejpoužívanější verzí protokolů Netflow je verze 5 a 9. V NetFlow verzi 5 najdeme fixní formát záznamů o tocích tzn. všechny flow záznamy mají stejný formát. Ve verzi 9 už lze mít variabilní formát toků většinou založený na základě šablony. V případě použití šablony je potřeba kolektoru oznámit tuto šablonu, aby věděl, jak budou záznamy vypadat.

Alternativní možnost je využití IPFIX, který vytvořila pracovní skupina IETF[3]. Standard IPFIX definuje, jaký formát mají jednotlivé toky a jak jsou přenášeny z exportéru na kolektor. Jde o univerzální standard na rozdíl od NetFlow, kde je formát záznamů uzavřený [22].

4.6 Klasifikační schéma dle Bashira

Klasifikační schéma, které představil pan Bashir ve své disertační práci, je založeno na technice analyzování chování uživatelů na sítí popsaného v kapitole 4.4 a je cíleně zaměřeno na detekci BitTorrent protokolu popsaného v 2.4 a Skype komunikace popsané v 2.5.

Protože je tato práce z větší části založena na tomto klasifikačním schématu bude v následujícím textu toto schéma podrobněji přiblíženo. Následující text je převzat z [3].

Analýza komunikace probíhá na základě síťových toků NetFlow představených v kapitole 4.5. Klasifikace je realizována do dvou fázích a to:

- Extrakce IP adres a portů uživatelů podezřelých z komunikace P2P (anglicky probing heuristic).
- Extrakce všech uživatelů P2P aplikací s využitím údajů získaných v první fázi detekce.

4.6.1 Extrakce uživatelů podezřelých z P2P komunikace

První fáze klasifikačního schématu hledá v síti uživatele, kteří ve svém počítači spustili Bittorentový klient, nebo program Skype. Schéma předpokládá práci s offline daty (tzn. nejdříve jsou nasbírána data, poté jsou analyzována). Identifikace probíhá na základě typických rysů chování těchto aplikací při jejich spouštění(tzn. v době spouštění aplikace musí již běžet sběr flow dat).

Nejdříve jsou shromážděny všechny IP adresy uživatelů z daného bloku nasbíraných dat. Poté jsou vybrány adresy se zdrojovými porty, které využili UDP jako transportní protokol a následně se skrze tento port komunikovalo nejméně do 100 různých míst. Tento krok algoritmu detekuje snahu P2P aplikací spojit se s ostatními uzly pomocí krátkých UDP zpráv (DHT probing heuristic), kterými oznamuje, že je přítomen v síti. Z toho plyne, že aby byla detekce úspěšná musí mít uživatel ve svém BitTorrentovém klientovi aktivované vyhledávání ostatních uzlů pomocí DHT. Posléze jsou vybrány adresy, jejichž procento jednopaketových toků dané velikosti dosahuje požadované velikost. Toto rozložení bylo naměřeno experimentálně na kolekcích dat a je naznačeno v tabulce 4.2.

Na základě těchto výsledků je provedeno ohodnocení, zda uživatel používá Skype, nebo BitTorrent aplikaci.

[3] IETF (The Internet Engineering Task Force) se zabývá vývojem internetových standardů. Více informací lze nalézt na domovské stránce http://www.ietf.org

Tabulka 4.2: Poměr jednopacketových toků dané velikosti k celkovému počtu jednopacketových toků [3]

Aplikace	Délka paketu	Zjištěný poměr
BitTorrent	100 - 400 bytů	85 - 88 %
Skype	0 - 100 bytů	71 - 73 %

V případě, že uživatelovo ohodnocení je BitTorrent je dále zjišťováno, zda komunikoval alespoň jednou přes daný UDP port na vzdálený port 80. Tento krok detekuje pokus o komunikaci uživatelského klienta s trackerem a pokud jsou v komunikaci zachyceny i heavy hitters[4], tak je adresa s daným portem prohlášena za podezřelou.

Pokud adresa spadá do Skype ohodnocení je zjišťováno jestli proběhla komunikace z daného UDP portu na vzdálený port 33033. Tento port používá Skype pro komunikaci se super uzly, jak je uvedeno 2.5.1. Pokud proběhla i komunikace na vzdálený port 80, nebo 443 (kontrola verze programu - update check), tak je adresa spolu s portem podezřelá, že využívá Skype.

Grafické znázornění algoritmu lze nalézt v příloze B.

4.6.2 Extrakce všech příbuzných uživatelů

Ve druhé fázi jsou na základě uživatelů nalezených v první fázi detekce hledáni příbuzní uživatelé.

Extrakce BitTorrentu probíhá třemi způsoby a to na základě způsobu lokalizace uzlů uvedeného v sekci 2.3.

- DHT - analyzovány jsou příchozí TCP toky, které mají zdrojový a cílový port větší jak 1024 a uživatel byl kontaktován pomocí IP adresy a podezřelého portu získaného v první fázi detekce. Dále musí být v rámci toku přeneseno alespoň 1500 bytů. Důvodem je vyloučení malých TCP toků, které nepřenáší data.

- UDP tracker - při detekci komunikaci s trackerem přes UDP jsou sledovány příchozí toky, které mají transportní protokol UDP, jsou směrovány na podezřelý BitTorrent port identifikovaný v první fázi klasifikace, vzdálený port je větší jak 1024 a počet přenesených bytů je opět větší jak 1500B. Důvodem je, aby tyto toky nebyly zaměňovány s DHT toky, kdy jsou pomocí malých UDP paketů hledány zaváděcí uzly viz 2.4.

- TCP tracker - v případě komunikace s využitím TCP trackeru je mechanismus následující. Vybírány jsou toky s protokolem TCP, kde zdrojový i vzdálený port je větší než 1024 a uživatel nebyl kontaktován pomocí UDP tzn.vzdálený port, IP adresa nebyly identifikovány v 1. fázi klasifikace. Další podmínkou je, aby toky obsahovaly tzv. heavy hitters. Tok je označen za heavy hitter, pokud počet přenesených bytů je větší jak 0,5MB, průměrný počet bytů v paketu je větší jak 800B, je aktivovaný TCP příznak přenosu a doba mezi jednotlivými toky je větší jak 5 vteřin. Další podmínkou je, že počet přenesených bytů musí být větší jak 1500B. Důvod je opět stejný jako v případě UDP trackeru vyloučit DHT.

[4] Jedná se o toky, které přenášejí velké množství dat a konzumují značnou část přenosového pásma

- Skype - daný síťový tok je označen za Skype tok, pokud:
  - Protokol je TCP, nebo UDP
  - Zdrojový, nebo cílový port odpovídá Skype portu detekovanému v první fázi klasifikace.
  - Zdrojová IP adresa a port byly kontaktovány pomocí IP adresy a podezřelého portu získaného v první fázi detekce.

4.7 Shrnutí

V této kapitole byly představeny mechanismy pro detekci P2P komunikace v síti. Byly představeny výhody a nevýhody jednotlivých metod.

Tyto mechanismy lze spolehlivě využít nejen pro detekci P2P, ale i jiných komunikací na síti například analýzu portů lze spolehlivě uplatnit na detekci HTTP protokolu.

Na základě požadavků na tuto práci byla zvolena metoda založená na chování uživatelů. Hlavním důvodem pro volbu této metody je:

- Pracuje se síťovými toky
- Dokáže odhalit šifrovanou komunikaci
- Není závislá na předdefinovaných číslech portů
- Není konfliktní se zákonem - nepřistupuje do datové části paketu

V další části práce bude navrhnut modul pro detekci P2P komunikace založený na klasifikačním schématu pana Bashira uvedeného v sekci 4.6, které reprezentuje jednu z metod detekce uživatelů na základě jejich chování. Důvodem je jeho vysoká přesnost (91,3%-95,4% pro BitTorrent, 100% pro Skype [3]). Další výhodou této metody je, že provádí detekci nad síťovými toky Netflow, což je i jeden z primárních požadavků na výslednou aplikaci této práce. Využití NetFlow sebou přináší řadu výhod například v rychlosti analýzy (není potřeba každý paket analyzovat zvlášť) a díky tomu, že je nám datová část paketu skryta není tato metoda v konfliktu se zákonem.

Lze tedy prohlásit, že klasifikační schéma podle pana Bashira odpovídá požadavkům na tuto práci.

Kapitola 5

Návrh modulu

V této kapitole bude představen návrh modulu pro detekci protokolu BitTorrent a aplikace Skype pro modulární systém analýzy síťových dat Nemea.

V dalším textu budou představeny jednotlivé význačné části návrhu modulu a na konci kapitoly bude navrhnuto kompletní klasifikační schéma přizpůsobené prostředí Nemea.

Jak již bylo zmíněno v 4.6 detekce probíhá ve dvou fázích. Nyní si představíme návrh modulu pro každou fázi.

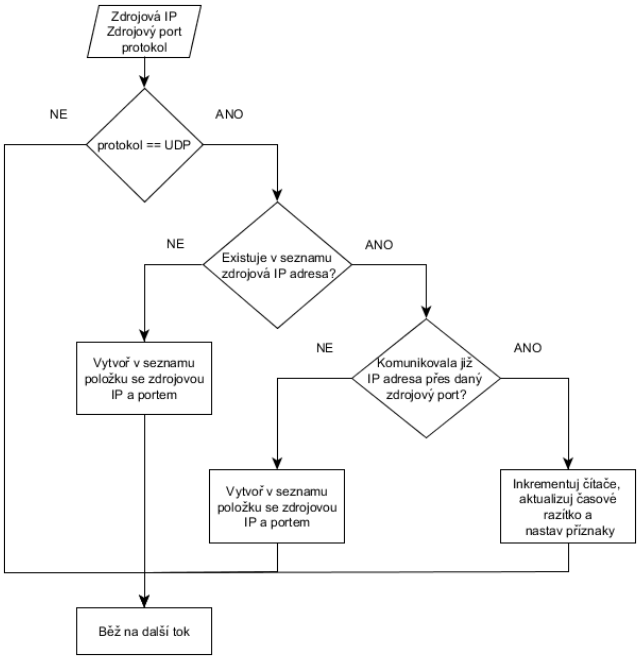
5.1 Fáze 1 - Sbírání záznamů a filtrace uživatelů

Detekce v této fázi je zaměřená na uživatele, kteří mají zapnutý BitTorrent klient a vyhledávají pomocí krátkých DHT UDP zpráv další uživatele, kteří vlastní požadovaný soubor. V případě Skype je detekce zaměřená na odhalení inicializačního procesu Skype klienta vůči super uzlu a login serveru. Sbírání záznamů je ohraničené časovým oknem více 5.5. Dobu trvání tohoto okna lze ovlivnit parametrem. V rámci této doby jsou sbírány a ukládány všechny záznamy využívající transportní protokol UDP a mající zdrojový/cílový port>1024. Výjimku tvoří porty 80 a 443. Záznamy jsou jednoznačně identifikovatelné pomocí zdrojové IP adresy a zdrojového portu a jsou ukládány do speciální datové struktury navržené v sekci 5.3. Pokud záznam se zdrojovou IP adresou a portem již existuje, tak je pouze inkrementován čítač výskytů tohoto záznamu a je provedena aktualizace časového razítka[1]. V případě že IP adresa neexistuje, nebo IP adresa nekomunikovala přes již zaznamenaný zdrojový port je vytvořen nový záznam. Algoritmus sběru záznamů je vidět na obrázku 5.1.

Při sběru záznamů jsou sledovány ještě další parametry toků. Nemají však vliv na to, jestli bude záznam přidán, ale jsou využity při filtraci záznamů po uplynutí časového okna viz funkce `filters()`, která bude představena v sekci 5.4. Jedná se o tyto parametry toků (tyto údaje jsou stanovovány pro každý záznam zvlášť:

- Celkový počet komunikací tzn. počet toků, které mají stejnou zdrojovou IP adresu/-port, ale rozdílnou cílovou IP adresu/port.
- Počet 1 paketových toků větších jak 100B.
- Počet 1 paketových toků menších jak 100B.

[1] Aktualizace časového razítka je prováděna proto, aby bylo možné zjistit v jakém čase proběhla poslední komunikace daného záznamu.



Obrázek 5.1: Algoritmus sběru dat

- Detekce zda tok komunikoval se vzdáleným portem 80, 443, 33033.
- Počet přenesených bytů v rámci toku.
- Počet paketů v toku.
- Detekce, zda se k příslušnému toku vztahuje odpovídající TCP heavy hitter.

5.2 Fáze 2 - Hledání příbuzných toků

V této fázi se vyhledávají uživatelé na základě podobností s uživateli detekovanými v první fázi algoritmu. Pokud je výsledek detekce kladný je uživatel přidán do datové struktury za předpokladu, že v ní ještě neexistuje.

V případě BitTorrentu jsou hledáni příbuzní uživatelé na základě mechanismu jejich lokalizace v P2P síti (DHT, tracker). Tato problematika byla probírána v sekci 2.3. Výjimku tvoří implementace detekce TCP trackeru. Pro úspěšnou detekci uzlů kontaktovaných TCP trackerem by bylo nutné zachovávat pro každý záznam všechny cílové adresy, na které v rámci časového okna směřovala komunikace. Tímto by velice vzrostla paměťová i časová náročnost modulu. Na základě toho bylo rozhodnuto, že tento mechanismus nebude implementován zejména i proto, že dnes se většinou setkáváme s UDP trackery.

U Skype je za příbuzného uživatele považován uživatel, který přijímá komunikaci z IP adresy a portu odhalených v první fázi klasifikačního schématu.

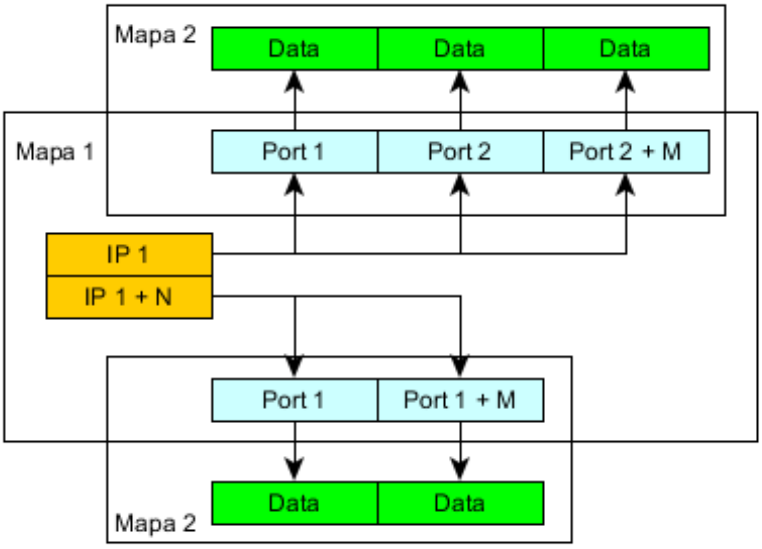
5.3 Návrh datových struktur

Před zahájením detekce je nutné nasbírat vzorek dat a vhodně jej uložit. V tomto případě pracujeme s NetFlow záznamy, které jsou v Nemea systému reprezentovány pomocí Uni-

rec záznamů. Jelikož nepotřebujeme všechny informace z Unirec záznamů je vhodné si ze záznamů vyjmout pouze ty potřebné a ty si pak pro účely další detekce uložit.

Požadavkem na aplikaci je, aby identifikovala IP adresy uživatelů v síti, kteří využívají Bittorrent, nebo Skype. Tito uživatelé mohou komunikovat přes vícero portů. Ke každému uživateli je dále zapotřebí si uchovávat informace pro účely další detekce. Shrnutí požadavků na datovou strukturu je uvedeno v následujících bodech:

- Hlavní klíč pro identifikaci uživatele je jeho IP adresa.
- Ke každé IP adrese jsou přiřazeny porty, přes, které uživatel komunikoval.
- Ke každé IP adrese a portu je nezbytné uchovávat další údaje (počet komunikací přes daný port, počet přenesených paketů, časová značka poslední komunikace atd.)



Obrázek 5.2: Navržená datová struktura.

Na základě těchto požadavků byla navrhnuta datová struktura, která je realizována pomocí dvou C++ map. Podrobná implementace bude představena v sekci 6.1. Na obrázku 5.2 je tato struktura graficky znázorněna, kde:

- IP je zdrojová IP adresa uživatele v síti (klíč pro vyhledávání v Mapě 1).
- Port je zdrojový port uživatele v síti (klíč pro vyhledávání v Mapě 2).
- Data - dodatečné informace pro potřeby detekce.

5.4 Filtrace

Po uplynutí časového okna je nad všemi záznamy přidanými do speciální datové struktury ve fázi 1 uvedené v 5.3 prováděna filtrace. Tato filtrace má za cíl vymazat z datové struktury záznamy, které nesplňují podmínky dané klasifikačním schématem popsaným v sekci 4.6.

V rámci filtrace jsou také mazány neaktivní záznamy[2]. Po ukončení filtrace jsou záznamy, které nebyly smazány odeslány na výstupní rozhraní modulu, kde si je následně může vyžádat jiný modul pro další zpracovávání. Požadavky na filtrační funkci jsou shrnuty v následujících bodech:

- Každý záznam musí komunikovat minimálně do 100 různých míst.
- Funkce musí být schopná přiřadit každému záznamu typ komunikace. V případě, že danému záznamu nelze přiřadit typ je tento záznam smazán.
- Na základě ohodnocení jsou prováděny následující úkony:
  - Pro BitTorrent je kontrolována odchozí komunikace na vzdálený port 80 a výskyt TCP heavy hitters.
  - Pro Skype je kontrolována odchozí komunikace na vzdálený port 33033 a také na vzdálený port 80 nebo 443.
- Funkce musí umět rozlišit příbuzné toky (toky získané v druhé fázi klasifikačního schématu) od toků získaných v první fázi. Tato vlastnost je důležitá v tom, že příbuzné toky musí být vyňaty z detekce uvedené ve výše zmíněných bodech.
- Dalším požadavkem je nutnost mazání neaktivních záznamů a v neposlední řadě odesílání záznamů na výstupní rozhraní modulu.

Tento postup je graficky znázorněn na obrázku 5.3, kde je uvedena posloupnost podmínek aplikovaných na každý záznam v datové struktuře.

5.5 Časové okno

Časové okno udává dobu ve vteřinách, po kterou mají být sbírána data ze zdroje (soubor, kolektor). Po uplynutí stanovené doby se nasbíraná data analyzují, viz 5.4. Po skončení filtrace je spuštěno nové časové okno. Tento proces se opakuje do doby, než jsou zpracovány všechna data, nebo je modul přerušen klávesami CTRL + C.

Délku časového okna lze ovlivnit parametrem. Výchozí nastavení délky okna je 5 minut.

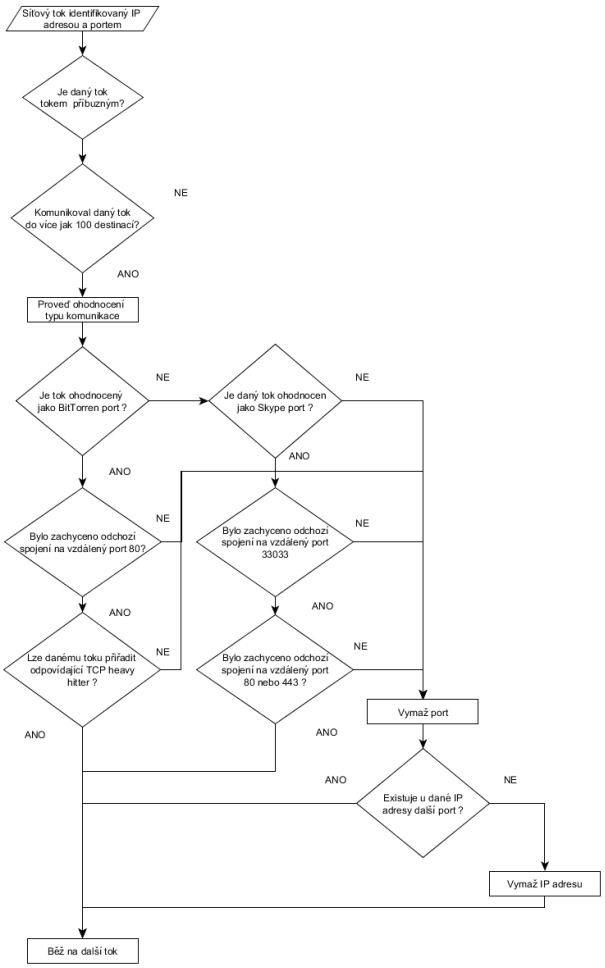
5.6 Vyřazovací mechanismus

Vyřazovací mechanismus má za úkol odebrat z datové struktury záznamy o uživatelích, kteří již nevykazují žádnou komunikaci. Nutnost návrhu tohoto mechanismu je v tom, že v datové struktuře jsou udržovány všechny záznamy, které prošly filtrací i po jejich odeslání na výstupní rozhraní. Důvodem je uchování těchto záznamů a jejich potřeba pro odhalení záznamů příbuzných tj. druhá fáze navrhovaného klasifikačního schématu uvedeného v sekci 5.2.

Doba vyřazení záznamů se liší na základě ohodnocení komunikace. V případě BitTorrentu je tato doba stanovena literaturou na 10 minut [21]. V případě Skype nebyla nalezena žádná hodnota v literatuře, proto byl čas vyřazení určen na 5 minut. Tento čas je stejný jako výchozí délka časového okna hlavně proto, že u Skype nelze nijak předem určit,

[2] Za neaktivní záznamy protokolu BitTorrent jsou označeny záznamy, které nevykazují žádnou komunikaci po dobu 10 minut [21].

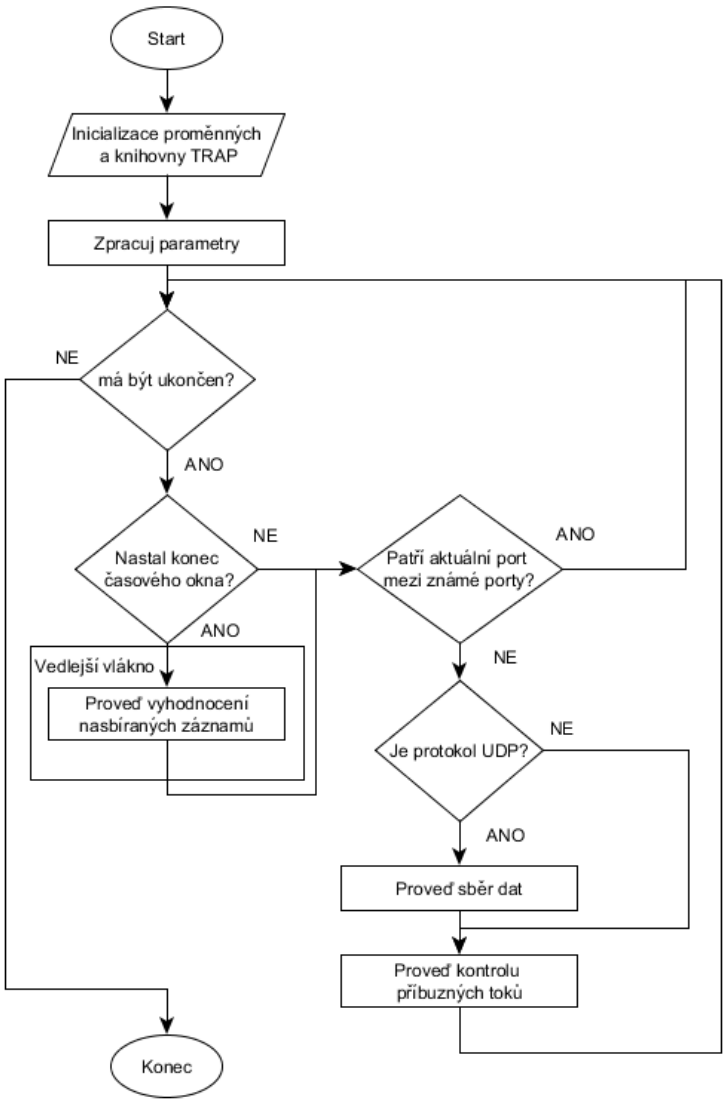
jak dlouho jej bude mít uživatel spuštěn (u BitTorrentu existuje předpoklad, že klient bude aktivní minimálně po dobu, než bude dokončeno stahování), proto je aktivita uživatele kontrolována na konci každého časového okna. Toto nastavení má předejít i falešným detekcím.



Obrázek 5.3: Schéma filtrace pro každý záznam.

5.7 Shrnutí

Na základě všech výše uvedených částí bylo vytvořeno klasifikační schéma založené na principech schématu, které představil p. Bashir 4.6. Toto schéma lze vidět v kompletní podobě na následujícím obrázku 5.4.



Obrázek 5.4: Navržené klasifikační schéma pro detekci dané P2P komunikace

Kapitola 6

Implementace modulu

Na základě návrhu uvedeného v kapitole 5 byl vytvořen modul pro detekci P2P komunikace. Jako implementační jazyk byl zvolen Jazyk C++, který patří v dnešní době k jednomu z nejvíce rozšířených a používaných jazyků. Důvodem pro zvolení tohoto jazyka je hlavně podpora map, třída string, funkce pro práci s časem, knihovna pthreads a v neposlední řadě dostupná, podrobná dokumentace a literatura. Dalším důvodem je, že celý framework Nemea je napsán v jazyce C/C++ a protože se jedná o rozšiřující modul měla by být volba jazyka zachována.

Implementace modulu je vícevláknová[1], kde hlavní vlákno vykonává sběr dat. Vedlejší vlákno provádí filtraci nasbíraných záznamů a jejich odeslání na výstupní rozhraní modulu. Důvodem vícevláknového řešení je fakt, že bez použití vedlejšího vlákna pro vyhodnocování dat by docházelo v době, kdy program vyhodnocuje nasbíraná data ke ztrátě nově příchozích dat (program by se zabýval zpracováváním dat a další by nesbíral).

Stěžejní část implementace modulu tvoří nekonečná smyčka, která po dobu časového okna sbírá data ze zdroje[2]. Data jsou ukládaná do speciální datové struktury navrhnuté v sekci 5.3. Její implementace bude probírána v sekci 6.1. Po uplynutí časového okna jsou nasbírané záznamy vyhodnoceny. Vyhodnocení provádí volání funkce `filters()`. Po dokončení vyhodnocení jsou záznamy odeslány na výstupní rozhraní.

Pro implementaci byla využita šablona Nemea modulu[3] Tato šablona obsahuje základní implementaci a inicializaci všech potřebných komponent tak, aby bylo možné do modulu posílat data a následně je po jednoduchém zpracování opět odeslat.

Nyní si jednotlivé základní prvky implementace modulu představíme. Nejdříve je modul pomocí struktury `trap_module_info_t` popsán. Tato struktura obsahuje mj. název modulu, popis modulu, počet vstupních a výstupních rozhraní.

Data budou do modulu přicházet přes jedno rozhraní (jeden zdroj). Formát vstupu je dán šablonou `<COLLECTOR_FLOW>` viz dále. Výstup je opět jeden a formát výstupních záznamů je dán skupinou záznamů `<P2P_FLOW>`, které obsahují zdrojovou IP adresa a port uživatele komunikujícího pomocí P2P a také typ komunikace (SKYPE, BITTORENT).

Dalším důležitým prvkem modulu je jeho ukončení. Ukončení práce modulu je v Nemea modulech realizováno přerušením hlavní smyčky a následným uvolněním zdrojů. Přerušení hlavní smyčky probíhá buď pomocí příkazu `break`, nebo pomocí signálů. Odchytáván je

[1] Pro implementaci vláken byla zvolena knihovna pthread. Více informací o této knihovně lze nalézt na adrese http://pubs.opengroup.org/onlinepubs/7908799/xsh/pthread.h.html

[2] Zdroj může reprezentovat soubor s nasbíranými daty, nebo lze mít modul připojen na sondu a sbírat data v reálném čase

[3] Jedná se o volně dostupnou šablonu `ExampleModule`

signál `SIGINT`, kterým je modul vynucené ukončen pomocí klávesové zkratky CTRL + C, nebo signál `SIGTERM`, který slouží při žádosti modulu o ukončení pomocí `kill`. Při zachycení některého z uvedených signálů, které je zapotřebí nejdříve registrovat, je pomocí speciální funkce nastavena statická globální proměnná `stop` z nuly na 1. To způsobí přerušení hlavní smyčky programu. Následuje uvolnění všech alokovaných zdrojů pomocí makra `TRAP_DEFAULT_FINALIZATION` a funkcí `ur_free_template, ur_free`. Poté je modul korektně ukončen voláním `return 0`.

Po zavolání hlavní funkce programu `main()` je inicializována knihovna TRAP voláním makra `TRAP_DEFAULT_INITIALIZATION` a funkce `trap_init`. Poté jsou zpracovány argumenty příkazové řádky pomocí funkce `getopt()`, která nabízí jednoduché nastavení zpracování argumentů. Tato funkce je standardně používána ve všech Nemea modulech, které vyžadují komunikaci s uživateli prostřednictvím argumentů příkazové řádky. V tomto modulu lze nastavit pomocí parametrů tyto vlastnosti:

- Délku časového okna(parametr `-t`).
- Tisk výsledků na standardní výstup(parametr `-o`)..
- Požadavky na detekci heavy hitters tj. počet přenesených bytů v toku(parametr `-b`) a průměrný počet přenesených bytů v každém paketu(parametr `-p`).

V další části jsou vytvořeny šablony formátu záznamů. Ty říkají jaké položky (a na kterých offsetech) záznam obsahuje. Každý TRAP interface může používat jinou šablonu

Aby bylo možné s UniRec šablonami pracovat, je zapotřebí vytvořit speciální strukturu `ur_template_t`, která UniReC šablonu reprezentuje. Tato struktura obsahuje informace o tom, kde v UniRec záznamu najít kterou položku a umožňuje záznamy číst a vytvářet. Šablona se vytvoří voláním funkce `ur_create_template`, které se předají názvy řetězců reprezentujících dané položky UniRec šablony. V případě našeho modulu je tato funkce volána s parametrem `<COLLECTOR_FLOW>`. To znamená, že na vstupním rozhraní budou očekávány toky, které budou obsahovat `<BASIC_FLOW>, LINK_BIT_FIELD, DIR_BIT_FIELD`, kde skupina položek `<BASIC_FLOW>` obsahuje všechny standardní položky, které tvoří síťový tok.

Nyní následuje začátek nekonečné smyčky, která tvoří základ programu (jeden průchod tvoří jeden načtený tok). V rámci této smyčky jsou čtena data ze vstupního rozhraní pomocí funkce `trap_recv`. Pokud čtení proběhlo v pořádku je zkontrolována velikost načteného záznamu. Pokud i tato kontrola proběhne v pořádku lze začít se samotným zpracováváním záznamu.

6.1 Implementace datové struktury

Pro ukládání UniRec záznamů byla vytvořena speciální datová struktura. Primárně se tato struktura skládá z C++ mapy, která zapouzdřuje další mapu s klasickou C strukturou. Primární mapa je vytvořena pomocí následujícího příkazu:

```
map<string, item> Record
```

Klíčem mapy je C++ string, který reprezentuje zdrojovou IP adresu. V UniRec záznamech je IP adresa reprezentována pomocí struktury `ip_addr_t`. Třída string byla zvolena, protože nabízí jednoduchou a bezstarostnou práci s řetězci (IP adresy lze jednoduše tisknout na výstup ve srozumitelném formátu). Asociovanou hodnotou je nově definovaný datový typ `item`, který reprezentuje druhou mapu. Definice tohoto datového typu je znázorněna zde:

```
typedef map<uint16_t, DATA> item
```

Klíčem této mapy je zdrojový port a asociovanou hodnotou je struktura `DATA`, která obsahuje podrobnější informace o daném záznamu potřebné pro účely detekce. Jejich popis naleznete níže.

Protože záznamy v mapách jsou unikátní, musí před přidáním záznamu proběhnout kontrola, zda daný záznam v mapě již neexistuje. Kdyby tato kontrola neproběhla, došlo by k přepsání informací udržovaných v záznamu, což je nežádoucí chování. Pro kontrolu existence záznamu je využitá standardní funkce z balíků map - `std::map::count`. Algoritmus přidávání záznamů byl představen v 5.1. Struktura `DATA` obsahuje tyto položky:

- `unsigned long int index1` - udává celkový počet výskytů záznamu.
- `unsigned long int index2` - udává počet 1 packetových záznamů, které přenáší více jak 100B.
- `unsigned long int index3` - udává počet 1 packetových záznamů, které přenáší méně jak 100B.
- `unsigned long int index4` - udává celkový počet 1 packetových záznamů.
- `bool skypePort` - nabývá hodnoty true, pokud proběhla komunikace na vzdálený port 33033. Jinak false.
- `bool port80` - nabývá hodnoty true, pokud proběhla komunikace na vzdálený port 80. Jinak false.
- `bool port443` - nabývá hodnoty true, pokud proběhla komunikace na vzdálený port 443. Jinak false.
- `int type` - určuje typ komunikace. Nabývá 1 - pro Skype, 2 pro BitTorrent. Hodnoty jsou reprezentovány výčtovým typem `enum`.
- `bytes` - součet bytů přenesených v rámci jednotlivých toků přes TCP.
- `packet` - součet paketů přenesených v rámci TCP toků.
- `time_t lastActive` - časové razítko posledního výskytu záznamu
- `bool flag` - nabývá hodnoty true u záznamů přidaných ve druhé fázi detekce. Důvodem je vyloučení těchto záznamů z filtrování.
- `bool heavy` - nabývá true, pokud byl zachycen odpovídající TCP heavy hitter tok.

V implementaci modulu najdeme tuto strukturu celkem třikrát. Jednotlivé struktury mají následující význam:

- `Record` - do této struktury se ukládají záznamy sbírané ze zdroje.
- `Temp` - dočasné úložiště pro záznamy uložené v `Record` v době vyhodnocování. Protože vyhodnocování probíhá ve vedlejším vlákně je nutné tomuto vláknu předat datovou strukturu obsahující nasbírané záznamy. Jelikož do struktury `Record` jsou v hlavním vlákně neustále sbírána data, tak nelze tuto strukturu předat pro vyhodnocení. K tomuto účelu slouží právě struktura `Temp`. Nicméně přímé kopírování jedné struktury

do druhé by bylo příliš časové náročné, proto jsou na tyto dvě struktury vytvořeny ukazatelé (`RecordPTR, TempPTR`), které jsou na konci každého okna zaměňovány tzn. že v prvním časovém okně je vyhodnocován obsah struktury `Record` a do struktury `Temp` jsou sbírány záznamy a ve druhém časovém okně je to přesně naopak.

- `Flows` - tato struktura obsahuje konečné výsledky detekce, které jsou pak odesílány na výstupní rozhraní.

6.2 Sběr dat

Záznamy jsou sbírány v hlavním vlákně programu. Samotný sběr dat probíhá v nekonečné smyčce, kde jeden průchod znamená jeden načtený záznam.

Po načtení záznamu je kontrolováno, zda zdrojový/cílový port neodpovídá tzv. známým portům[4]. Pokud odpovídá je tento záznam přeskočen. V opačném případě je záznam přidán do datové struktury `Record` za předpokladu, že splňuje následující podmínky:

- Transportní protokol je UDP
- Záznam do datové struktury nebyl zatím přidán

V rámci sběru dat jsou pro každý tok nastavovány příznaky, které jsou následně využívány pro potřebu detekce. Jedná se o příznaky `port80`, `port443`, `skypePort`, `lastActive`, `heavy`, `bytes`, `packet` a počítadla `index1`, `index2`, `index3`. Význam těchto příznaků a počítadel byl představen v sekci 6.1.

6.3 Extrakce příbuzných toků

Extrakce příbuzných toků probíhá na základě nasbíraných dat. V rámci nekonečné smyčky, která je součástí hlavního vlákna programu, jsou pro aktuální toky hledány závislosti na tocích, které byly identifikovány v první fázi detekce (bližší popis lze nalézt v sekci 5.1).

Extrakce je realizována pomocí třech dotazů nad datovou strukturou `Flows`. Dotazy se liší v závislosti na použitém mechanismu pro vyhledání uzlů v sítí (Tracker, DHT), jak bylo popsáno v sekci 4.6.2. Pokud je odhalen příbuzný tok, je přidán do struktury `Record`.

6.4 Funkce pro zpracování dat

Po dokončení sběru dat přichází na řadu jejich vyhodnocení. Vyhodnocení je implementováno do samostatného vlákna, kde je realizováno pomocí funkce `filters()`, která se sestává ze dvou průchodů datovou strukturou `Record` a jedním průchodem stukturou `Flows`. V těchto průchodech je každý tok testován řadou podmínek. Pokud daný tok neodpovídá těmto podmínkám je vymazán. Nyní bude představena implementace všech podmínek v jednotlivých průchodech, které tvoří funkci `filters()`.

První průchod strukturou `Record` realizuje podmínky dané klasifikačním schématem uvedeným v sekci 5.4.

První podmínkou je kontrola položky `flag` v datové struktuře. Tento `flag` je součástí datové struktury více lze nalézt v sekci 6.1. Pokud je nastaven na `true` znamená to, že

[4] Jedná se o seznam služeb a jim pevně vyhrazených čísel portů, na základě nichž se dá rozlišovat o jakou službu se jedná. Více informací lze najít na stránkách http://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xhtml?&page=1

záznam byl přidán v druhé fázi detekce tj. na základě příbuzných toků. V tomto případě je zbytek filtrace přeskočen.

Druhá podmínka maže záznamy, které nesplňují podmínku v počtu odchozích toků do různých destinací. Dle návrhu má být tento počet větší jak 100. Tento údaj nám implementuje položka v datové struktuře s názvem `index1`. Pokud je tato položka menší, tak je záznam vymazán. Záznam je mazán i v případě, kdy položka `index4` je nulová tzn. že nebyl nalezen žádný jednopacketový tok.

Třetí podmínka je jednou z nejdůležitějších. Dochází v ní k ohodnocení jednotlivých záznamů. Záznam je ohodnocen výčtem `enum` buď jako SKYPE, nebo jako BITTORENT. Pro toto ohodnocení je nezbytné vypočítat dílčí hodnotu poměru jednopaketových toků větších (pro BitTorrent), nebo menších jak 100B (pro Skype) vůči všem jednopaketovým tokům. Zda se budou počítat pakety menší, nebo větší jak 100B záleží na tom, který index (`index2`, `index3`) je větší. Výpočet probíhá podle následujícího vztahu:

$$V = \frac{P}{C} * 100$$

kde V je výsledek výpočtu, P je počet jednopaketových toků menších/větších 100B, C je celkový počet jednopaketových toků dané IP adresy a portu. Výsledek je pak porovnán s hodnotami uvedenými v tabulce 4.2 a následně je přiřazen odpovídající typ komunikace.

Pátá podmínka maže z datové struktury záznamy, které nejsou ohodnoceny v předchozím kroku.

Další podmínky jsou specifické pro daný typ komunikace:

- komunikace ohodnocená jako BitTorrent - je kontrolováno, zda proběhla komunikace na vzdálený port 80. Tato operace je implementována pomocí testování příznaku `port80` v datové struktuře. Pokud je příznak vyhodnocen jako false je záznam smazán. Dále je testována přítomnost heavy hitters toků, které signalizuje příznak `heavy`. Spolu s testováním příznaku `heavy` je zde také počítán průměrný počet bytů v paketu, který musí dosahovat hodnoty dané parametrem `-p`.

- komunikace ohodnocená jako Skype - je kontrolováno, zda jsou nastaveny příznaky `skypePort`, `port80`, `port443`. Aby záznam nebyl smazán, musí být `skypePort` nastaven na true a minimálně jeden ze dvojice příznaků `port80, port443` též. V opačném případě je záznam mazán.

Druhý průchod strukturou `Record` překopírovává záznamy ze struktury `Record` do finální struktury `Flows`. Při kopírování jsou procházeny všechny záznamy ve struktuře `Record` a pro každý kopírovaný záznam nastává jeden z následujících stavů:

- Pokud IP adresa i port již existují ve struktuře `Flows` je provedena pouze aktualizace časového razítka

- Pokud struktura `Flows` obsahuje IP adresu, ale nikoli daný port, tak je daný port přidán pod příslušnou adresu do struktury `Flows`.

- Pokud ve struktuře `Flows` neexistuje kopírovaná IP adresa, tak je spolu s příslušným portem vytvořena.

Nakonec je procházena struktura `Flows`, kde se kontroluje aktuálnost záznamu v datové struktuře. Pokud daný uživatel nebyl aktivní po stanovenou dobu, tak je odebrán. Toto

chování je realizované voláním funkce `comparetime()`, která má jako parametry časové razítko poslední komunikace daného toku a aktuální časové razítko. Tyto časová razítka jsou porovnána. Pokud je rozdíl mezi těmito dvěma razítky větší jak daný čas, tak je záznam odebrán.

6.5 Časové okno

Jak bylo uvedeno v návrhu délka časového okna, při kterém jsou sbírány a ukládány záznamy je ovlivnitelná parametrem výchozí hodnota je 5 minut. Tato funkčnost je opět zajištěna pomocí porovnání časových razítek. Při prvním průchodu nekonečné smyčky je zaznamenáno koncové časové razítko prvního toku. Jedná se o položku UniRec záznamu `UR_TIME_LAST`. Následně je na základě tohoto časového razítka zaokrouhlena hranice okna na nejbližší nižší číslo dělitelné délkou okna. To v našem případě znamená, že pokud máme délku okna 5 minut a koncová značka je například 12:03:00, tak bude první okno začínat v čase 12:00:00. Zaokrouhlení je realizováno pomocí následujícího výrazu `start_time = win_start - win_start % (int)windowsLenght`, kde `windowsLenght` je délka časového okna ve vteřinách zaokrouhlená na celé vteřiny. Protože by v prvním okně bylo méně záznamů, je z detekce vyloučeno a přeskočeno.

Podobné řešení je již používáno v ostatních modulech a je velmi užitečné v případě, že je nasazeno více detektorů pracujících s časovými okny za sebou. Pokud chceme porovnávat jejich výsledky, je nezbytné, aby měly všechny detektory nastavená stejná časová okna.

Samotná implementace je pak následující. Po stanovení času začátku prvního okna v programu, který je reprezentován proměnnou `start_time` je tento údaj porovnáván s aktuální koncovou časovou značkou `next_time` toku pomocí již zmiňované funkce `comparetime()`. Pokud je rozdíl větší, než 5 minut je hodnota proměnné `start_time` navýšena o hodnotu délky časového okna a tím je nastaven začátek nového okna. Poté je volána funkce `filters()`, která provádí vyhodnocení a odeslání nasbíraných záznamů.

6.6 Odeslání

Odesílání záznamů je implementováno ve funkci `filters()`, která pracuje ve vedlejším vlákně programu. Aby došlo k odeslání dat na výstupní rozhraní musí být toto rozhraní připojeno k modulu. Samotné odeslání je realizováno pomocí funkce `trap_send()`, která je ve výchozím stavu nastavena jako blokující (parametr `TRAP_WAIT` viz 3.2). Vzhledem k tomu, že implementace je umístěna ve vedlejším vlákně je vhodné zajistit, aby vlákno bylo ukončeno před koncem dalšího časového okna. Proto je funkci `trap_send()` nastavena doba čekání pomocí makra `TRAP_HALFWAIT`, které může blokovat modul jen v případě, kdy je na výstupním rozhraní připojen další modul pro odebírání dat. Toto nastavení se v Nemea systému provádí pomocí funkce `trap_ifcctl()` volané po inicializaci TRAP knihovny.

Pokud je výstupní rozhraní připojeno k modulu jsou postupně odesílány všechny záznamy z datové struktury `Flows`. Stejně jako u vstupních záznamů je nutné definovat formát výstupních záznamů pomocí šablony. Pro účely tohoto modulu byla vytvořena nová skupina polí UniRec záznamů `<P2P_FLOW>`, která je použita pro vytvoření šablony výstupních záznamů.

Formát výstupních záznamů je následující:

- Zdrojová IP adresa uživatele (položka `SRC_IP`)

- Zdrojový port uživatel (položka `SRC_PORT`)
- Typ P2P komunikace(položka `P2P_TYPE`) ve formátu `uint8_t`:
  - 1 pro Skype
  - 2 pro BitTorrent

Kapitola 7

Testování

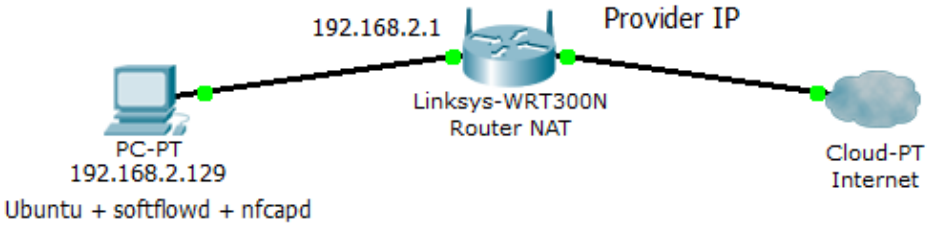
V této kapitole bude představena metodika testování navrženého modulu. Testování modulu probíhalo na základě série experimentů s různými datovými sadami. Tyto datové sady lze pomyslně rozdělit do dvou typů na základě prostředí ve kterém byly nasbírány. Záznamy první datové sady byly nasbírány v rámci lokální sítě, kde na počítači byla spuštěna softwarová NetFlow sonda, která v pravidelných intervalech exportovala záznamy na kolektor, který se nacházel na stejném počítači. Zdrojem dat pro sondu byl buď provoz na síťovém rozhraní počítače v reálném čase, nebo záznamy uložené v souboru s koncovkou `*.pcap` zachycené programem Wireshark. Exportované záznamy byly čteny kolektorem a uloženy do souboru. Druhým typem dat byly záznamy sbírané v reálném čase z přístupných kolektorů systému Nemea. Tyto záznamy ve většině případů nebyly ukládány do souborů, ale přímo posílány na vstupní rozhraní modulu. V následujícím textu budou podrobně představeny experimenty s jednotlivými datovými sadami.

7.1 Data nasbíraná v lokální síti

Sběr dat probíhal na lokálním počítači s nainstalovaným linuxovým operačním systémem Ubuntu v jeho nejnovější verzi 14.04 LTS[1]. Na tomto počítači byla nainstalována softwarová NetFlow sonda `softflowd`. Tato sonda je schopná shromažďovat informace o síťové komunikaci na daném síťovém rozhraní do toků standardu Cisco NetFlow. Následně lze tyto záznamy exportovat na kolektor, kde můžou být zpracovány. V našem případě jako kolektor posloužil nastroj `nfcapd`, který je součástí balíku `nfdump`. `Nfcapd` umí číst a posléze uložit exportované NetFlow záznamy zasílané sondou. Data jsou pak do modulu posílána prostřednictvím Nemea modulu `nfdump_reader`, který dokáže číst `*.nfcapd` soubory a zaslat je na vstupní rozhraní modulu.Všechny použitě nástroje jsou zdarma dostupné. Popis a grafické znázornění celé topologie je uvedeno na obrázku 7.1. Důvodem sběru dat v lokální sítí je fakt, že lze na daném počítači generovat vlastní síťový provoz na rozdíl od dat, která jsou sbírána v reálném čase z kolektoru. Lze tedy poměrně snadno ověřit, zda modul dokáže odhalit a správně určit příslušnou komunikaci. Nevýhodou je, že v této malé lokální síti nelze generovat takový síťový provoz, který lze očekávat v prostředí, kde bude modul reálně používán tj. páteřní linky, kde je zátěž a množství zdrojů falešné detekce několikanásobně větší. Nyní budou popsány jednotlivé experimenty.

První experiment byl zaměřen na detekci protokolu BitTorrent. Na testovacím počítači byly spuštěny celkem tři programy podporující stahování přes tento protokol (v tomto

[1] Více informací o tomto operačním systému lze najít na domovské stránce http://www.ubuntu.cz/



Obrázek 7.1: Topologie sestavená pro účely testování

případě byla data nasbírána programem Wireshark na počítači s OS Windows 7 s IP adresou 147.229.198.60). Jednalo se o programy BitComet, BitTorrent, utorrent. V každém programu bylo spuštěno stahování torrentu, konkrétně linuxové distribuce Ubuntu 14.04 LTS a to po dobu 5 minut. Pro vytvoření zdroje falešných detekcí byl využit program, který umí přehrávat P2P TV. Konkrétně se jedná o program SopCast, který umožňuje sledovat zahraniční televezní stanice. Po uplynutí stanoveného času byla data vyhodnocena navrhovaným modulem. Očekávaný výsledek detekce byl 3 záznamy se stejnou zdrojovou IP adresou a rozdílnými zdrojovými porty. Výsledky experimentu potvrdily očekávané výsledky tj. detekovány byly tři záznamy, které odpovídají komunikaci výše uvedených klientů BitTorrent a ani jedna pozitivní detekce P2P TV - program SopCast. Výsledky prvního experimentu jsou zaneseny do tabulky 7.1.

Tabulka 7.1: Výsledky detekce prvního experimentu

Spuštěná aplikace	Počet kladných detekcí
BitTorrent	1
BitComet	1
utorrent	1
SopCast	0

Druhý experiment byl zaměřený na detekci Skype. Pro účely testování byl vytvořen hovor mezi Skype klientem nainstalovaným na testovacím počítači a dalším uživatelem umístěným mimo lokální síť testovaného počítače. První pokusy s těmito daty dopadly negativní detekcí. Následně bylo zjištěno, že negativní detekce je způsobena tím, že počet odchozích toků je < 100 (což je experimentální hranice zjištěná p. Bashirem v [3]). Ostatní podmínky byly splněny. Tento stav se opakoval i při pár dalších experimentech. Počet odchozích toků se pohyboval okolo hodnoty 60. Při experimentálním snížení hranice odchozích toků pod tuto hodnotu byly detekce kladné. Řešením by bylo umožnit nastavení této hranice uživateli jako parametr příkazové řádky. Nicméně jelikož se tento problém na kolektorových datech neobjevoval bylo usouzeno, že se tato hranice nebude měnit. Zvláště proto, že je brána jako jedna z klíčových znaků pro určení typu komunikace a hrozí zde velké nebezpečí špatné klasifikace.

7.2 Data nasbíraná z kolektorů

Pro účely testovaní bylo zpřístupněno několik kolektorů, které poskytují reálná data ze sond, které jsou umístěny v síti organizace CESNET. Pro příjem dat je nutné pouze připojit na vstupní rozhraní modulu příslušný kolektor. Jakmile se modul spojí s kolektorem je zahájen přenos záznamů.

IP adresy záznamů přijatých z kolektoru jsou z důvodu zachování soukromí anonymizovány. To však z hlediska testování nezpůsobuje žádný problém, protože čísla portů jsou zachována.

Problém v testování s těmito daty nastává v tom, že nelze přesně zjistit jaký síťový provoz (jaké aplikace) byly na daných počítačích v síti používány. Tudíž vzniká riziko výskytů falešných detekcí. Z tohoto důvodu byl modul testován i na lokální síti, kde byla generována vlastní síťová komunikace a tím pádem bylo možné jednoznačně potvrdit korektní detekce. Více informací lze najít v sekci 7.1.

Aby bylo možné detekované záznamy z kolektoru co nejpřesněji testovat je nutné provést analýzu zachycených čísel portů, které nejsou anonymizovány. Analýza spočívá v porovnání zachycených čísel portů s výchozími čísly portů, které používají dnešní P2P aplikace a Skype. Výchozí čísla portů byla získána experimentálně na základě informací, které poskytli klienti dané komunikace.

U protokolu BitTorrent byly na lokálním počítači nainstalovány nejznámější klienti této komunikace a následně byl zaznamenán port, který je nastaven ve výchozí instalaci daných klientů. Bohužel každý klient nabízí možnost tento port změnit, případně dokonce volit po každém startu aplikace port jiný (aplikace utorrent). Tento fakt bohužel nelze nijak předem ovlivnit, tudíž je zde předpoklad, že uživatel využívá výchozí nastavení klienta. Seznam testovaných klientů a zjištěných portů je znázorněn v tabulce 7.2

Tabulka 7.2: Tabulka BitTorrent aplikací a jím přiřazených výchozích portů

Aplikace	Číslo portu
BitTorrent	33645
BitSpirit	23956
Vuze	37865
BitComet	15544
utorrent	6881(30589)
Transsmision	51413

Dále bylo rozhodnuto, že budou sledována čísla portů v rozsahu nad 49152. Jedná se o rozsah dynamických/soukromých čísel portů. Důvodem sledování těchto portů je fakt, že při použití funkce náhodné volby portu v programu uTorrent, který se ve výsledcích testů ukázal jako jeden z nejpoužívanějších, volí porty právě z tohoto rozsahu.

U Skype jsou zdrojem dat k porovnání informace obsažené v `Host's cache` umístěné v souboru `shared.xml` v instalačním adresáři aplikace. Položka `Host's cache` obsahuje zakódované informace o IP adresách a portech, které určují adresy super uzlů. V rámci testování bylo shromážděno pět souborů `shared.xml` od různých uživatelů. Tyto soubory byly dekódovány[2]. Bylo zjištěno, že ve všech souborech je 200 záznamů určijících adresy

[2]Pro dekódování adres super uzlů byl využit skript dostupný na stránkách http://ryandoyle.net/

super uzlů. Z toho vždy alespoň jeden záznam obsahuje port 33033, což potvrzuje závěry pana Bashira uvedené v [3]. Zbyle záznamy jsou až na pár výjimek v rozsahu čísel portů 40001 - 40050. Port na kterém naslouchá běžný klient Skype je uveden také v již zmíněném souboru. Tento port je uživateli přiřazen a i když se za dobu experimentování nezměnil je nemožné jej určit předem. Tyto porty prezentuje tabulka 7.3. Z hlediska testování lze využít fakt, že Skype přiděluje tyto čísla portů poměrně vysoká.

Tabulka 7.3: Zjištěná čísla portů na kterých naslouchají jednotlivý klienti

Uživatel	Číslo portu
Uživatel 1	42859
Uživatel 2	13659
Uživatel 3	50848
Uživatel 4	28385
Uživatel 5	40086

V následující textu budou představeny výsledky některých experimentů, které proběhly s daty nasbíranými z kolektorů. Záznamy byly sbírány po určitý čas s výchozím nastavením délky časového okna (5 minut) a s různým nastavením klasifikace těžkých toků. Výsledky byly posílány na standardní výstup a tam následně přesměrovány do souboru. Tento soubor byl posléze analyzován.

Z kolektorů byly nasbírány tři vzorky dat o délce 5, 10, 15 minut. S každým vzorkem byly provedeny tři měření na základě nastavení parametrů klasifikace těžkých toků[3]. Jednotlivé parametry byly nastaveny takto:

- Výchozí nastavení(1 řádek tabulky u daného vzorku)
  - b = 524288
  - p = 400
- Vypnutá klasifikace těžkých toků(2 řádek tabulky u daného vzorku)
  - b = 0
  - p = 0
- Snížená hranice klasifikace těžkých toků(3 řádek tabulky u daného vzorku)
  - b = 30000
  - p = 70

V tabulce 7.4 jsou uvedeny výsledky experimentů s daty nasbíranými na kolektoru `collector-nemea:9901`. Oproti lokální sítí je zde nesrovnatelně vetší provoz. Přibližné vteřinové měření počtu toků na tomto kolektoru s pomocí modulu `flowcounter` dosáhlo hodnoty 22067 toků. Výsledky experimentů potvrdili předpoklad, že detekci BitTorrentu bude hodně ovlivňovat nastavení parametrů heavy hitters. To je způsobeno hlavně tím, že někteří klienti protokolu BitTorrent jako např. uTorrent podporují dynamické snižování

`posts/decoding-the-skype-host-cache/`.

[3]Popis a výčet parametrů byl představen v kapitole 6

velikosti paketů v závislosti na stavu linky [3]. To znamená, že se mění průměrná velikost paketů a je zde možnost, že zvláště při výchozím přísném nastavení (b = 0,5MB, p = 400) nebudou některé záznamy detekovány. Na druhou stranu hrozí zde menší nebezpečí vzniku falešné detekce, protože takto velké toky jsou typickým rysem přenosu dat přes protokol BitTorrent. Proto padlo rozhodnutí umožnit uživateli/správci sítě nastavit hranici klasifikace těžkých toků. Tuto hranici může správce sítě stanovit například analýzou zachycených TCP toků pomocí programu Wireshark, tím by mělo být dosaženo větší efektivity detekce. Ukázkou zde může být porovnání prvního a třetího řádku u každého vzorku v tabulce 7.4, kde lze vidět zvýšení počtu detekcí při zjemnění hranice detekce heavy hitters. Jinak lze z výsledků vyčíst, že většina detekovaných portů je buď dynamických, nebo přiřaditelných, což naznačuje kladnost detekce.

V případě Skype se zdá být detekce poměrně přesná, což dokazuje velký počet porovnatelných čísel portů hlavně v 15 minutové datové sadě. Zbylá čísla portů jsou poměrně vysoká, což opět naznačuje kladnou detekci Skype uživatelů. Obecně zde platí pravidlo, že pokud je detekován uživatel, tak se po chvíli objeví velké množství příbuzných toků, kterými se Skype uživatel snaží kontaktovat super uzly mj. i proto aby si ověřil, že jsou aktivní.

Nakonec je nutné říci, že detekce je závislá na zachycení úvodních mechanismů jednotlivých aplikací tzn. sběr dat musí být aktivní v době, kdy se Skype přihlašuje a v případě BitTorrentu při vyhledávání sousedních uzlů pomocí DHT.

Tabulka 7.4: Výsledky experimentů s daty získanými z kolektoru

	Skype		BitTorrent		
	Nalezených	Rozsah portů 40001 - 40050	Nalezených	Známé porty	Dynamické porty
5 minut	1	0	4	0	1
	1	0	28	4	5
	1	0	8	1	3
10 minut	116	2	18	3	45
	116	2	71	6	64
	116	2	30	4	50
15 minut	26	21	33	0	22
	26	21	149	9	82
	26	21	58	1	36

Další výsledky testů lze najít na přiloženém CD,

7.3 Falešné detekce

Falešné detekce jsou problémem, který zatěžuje snad každý detekční algoritmus. V případě navrhovaného modulu je falešná detekce problémem dvojnásobným, protože takový tok je přidán do datové struktury a následně je používán pro detekci toků příbuzných. Tím počet falešných detekcí narůstá. Pan Bashir uvádí ve své práci velmi vysokou hodnotu přesnosti algoritmu při zachování minimálního počtu falešných detekcí [3].

Během testování byly falešné detekce odhalovány především na základě čísel portů. V rámci testování v lokální síti nebyla zjištěna žádná falešná detekce i v případě úmyslného generování této komunikace (P2P TV). Jak již bylo řečené nelze porovnávat podmínky lokální sítě v počtu generované komunikace s daty nasbíranými na kolektoru. Tam lze očekávat daleko více zdrojů falešných detekcí. Z tohoto důvodu byl kladen důraz na analýzu zejména nízkých čísel portů detekovaných modulem. Bylo provedeno jejich srovnání s registrovanými porty[4]. I přesto, že byly splněny podmínky dané navrhovaným schématem ve výsledcích se nacházely záznamy s registrovanými čísly portů. Pokud je port registrovaný je nepravděpodobné, že by byl využíván P2P klientem. Nicméně to nelze tvrdit s jistotou, protože i když je port registrovaný, lze ho v klientu P2P aplikace jednoduše nastavit a takové případy nalezeny byly. Dokonce to lze brát i jako záměrné chování pro průchod přes firewall.

Dalším spíše opačným problémem může být to, že některé záznamy nemusí být odhaleny. To je dáno tím, že nesplní nějakou podmínku danou klasifikačním schématem.

Nejčastěji takové chování způsobují BitTorrent klienti založené na protokolu uTorrent. Tento protokol dokáže dynamicky měnit velikost paketů na základě stavu linky a tím měnit průměrnou délku paketů [3]. K této změně dochází u paketů, které přenáší data tzn. ovlivňuje to klasifikaci těžkých toků (heavy hitters). K prevenci tohoto chování je možné nastavit pomocí parametrů klasifikaci těchto těžkých toků, jak bylo uvedeno v popisu tabulky 7.4.

7.4 Kontrola paměti a časová náročnost

Vzhledem k tomu, že v modulu je implementováno paralelní zpracování dat pomocí vláken je nutné ověřit jestli nedochází ke vzniku paměťových chyb. Pro tyto účely byl použit zdarma dostupný program `valgrind`, který kontroluje výskyt paměťových chyb[5]. Z provedených měření vyplynulo, že všechny zdroje se korektně uvolňují.

Z pohledu časové náročnosti je nejdůležitější, aby nedocházelo k příliš velkým zpožděním při sběru dat. Kritickým místem pak zde je nastavení klasifikace těžkých toků, kdy jsou v rámci hlavní smyčky kontrolovány příslušné TCP toky a v případě kladné detekce jsou ukládány informace o počtu přenesených bytů a paketů. V případě výchozího nastavení jsou tyto parametry nastaveny velmi přísně hlavně proto, aby při tak velkém množství zpracovávaných dat nedocházelo k falešným detekcím i za cenu možnosti nedetekování malých záznamů, které nepřenáší velké množství dat.

[4] Porovnání proběhlo s registrem organizace IANA dostupným na adrese https://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.txt

[5] Více podrobností o programu lze nalézt na domovských stránkách programu http://valgrind.org/

Kapitola 8

Závěr

Cílem této práce bylo vytvoření modulu pro modulární systém analýzy síťových dat Nemea na detekci vybraných P2P aplikací (Skype, BitTorrent) v sítí. Detekce protokolu BitTorrent, jakožto nejpoužívanějšího protokolu pro sdílení souborů je důležitá proto, že konzumuje značnou část přenosového pásma. Na vysokorychlostních linkách to nemusí působit problém, ale v pomalejších sítích s omezenou šířkou přenosového pásma už ano. V těchto sítích může tedy tento modul najít velké uplatnění. Výhodou pro správce menších sítí může být i fakt, že systém Nemea lze zdarma stáhnout. V případě Skype může modul najít své uplatnění v nastavení priority komunikace tohoto hlasového přenosu za účelem potlačení negativních vlivů, kterým je hlasový přenos v síti vystaven.

V úvodu této práce byla obecně popsána architektura P2P sítí a vybraných aplikací pro detekci (Skype, BitTorrent). Tomuto tématu se věnovala celá 2. kapitola. V 3. kapitole byl představen systém Nemea, pro který byl v této práci vyvíjen rozšiřující modul. V kapitole 4 byly uvedeny dnešní techniky pro detekci P2P komunikace v síti, založené na různých přístupech k analýze paketů. Tato kapitola obsahovala i popis technologie NetFlow. Obsahem kapitol 5 a 6 byl návrh a implementace zvolené metody pro detekci vybraných P2P aplikací v síti ze záznamů o síťových tocích NetFlow. V závěru práce byly uvedeny výsledky experimentů na základě dat získaných z různých prostředí.

Z výsledků experimentů bylo zjištěno, že zvolená metoda detekce podává dobré výsledky. Toto tvrzení se zejména zakládá na výsledcích testů provedených v lokální síti, kde byla generována vlastní síťová komunikace a bylo možné detekované výsledky s určitostí potvrdit. V případě skutečného nasazení modulu do sítě by bylo nutné zvolit správné nastavení těžkých toků (heavy hitters), které do jisté míry ovlivňují výsledky detekce. V případě Skype pak bylo zjištěno, že v lokální síti hranice odchozích spojení v některých případech nepřesáhla schématem požadovanou hodnotu sta spojení.

Modul by bylo možné do budoucna rozšířit o detekci dalších protokolů založených zejména na analýze známých portů jako je například HTTP, DNS. Dále by bylo možné modul rozšířit o možnost grafické reprezentace detekovaných výsledků např. nástrojem `gnuplot`, který dokáže spolupracovat s jazykem C++.

Literatura

[1] BARTOŠ, V.; ŽÁDNÍK, M.; ČEJKA, T.: Nemea: Framework for stream-wise analysis of network traffic. Technická zpráva, Brno: CESNET, 2013.

[2] BASET, S. A.; SCHULZRINNE, H.: An Analysis of the Skype Peer-to-Peer Internet Telephony Protocol. Technická zpráva, New York: Department of Computer Science Columbia University, 2004.

[3] BASHIR, A.: *Classifying P2P Activities in Netflow Records: A Case Study (BitTorrnet & Skype)*. Dizertační práce, Ontario: Carleton University Ottawa, 2012.

[4] CLAISE, B.: *Cisco Systems NetFlow Services Export Version 9*, [online]. 2004 [cit. 2014-04-28].
URL http://tools.ietf.org/html/rfc3954.html

[5] COHEN, B.: *The BitTorrent Protocol Specification*, [online]. 2008 [cit. 2014-04-25].
URL http://www.bittorrent.org/beps/bep_0003.html

[6] EHLERT, S.; PETGANG, S.: Analysis and Signature of Skype VoIP Session Traffic. Technická zpráva, Berlin: Fraunhofer FOKUS in Germany, 2006.

[7] GONG, Y.: *Identifying P2P users using traffic analysis*, [online]. 2005 [cit. 2014-04-25].
URL http://www.symantec.com/connect/articles/identifying-p2p-users-using-traffic-analysis

[8] IEEE: *Automated Traffic Classification and Application Identification using Machine Learning*, 5 2005, ISBN 0-7695-2421-4.

[9] JALSOVSZKY, Z.: *Rozpoznání uživatelů P2P sítí na základě analýzy síťového provozu*. Diplomová práce, Brno: FIT VUT v Brně, 2009.

[10] liberouter: *Nemea*, [online]. [cit. 2014-04-27].
URL https://www.liberouter.org/technologies/nemea

[11] LOEWENSTERN, A.; NORBERG, A.: *DHT Protocol*, [online]. 2008 [cit. 2014-05-05].
URL http://www.bittorrent.org/beps/bep_0005.html

[12] MATOUŠEK, P.: *Síťové aplikace a správa sítí: 7. Hlasové služby*, [přednáška]. 2013 [cit. 2014-04-26].

[13] MATOUŠEK, P.: *Síťové aplikace a správa sítí: 12. Monitorování toků NetFlow*, [přednáška]. 2013 [cit. 2014-04-28].

[14] SEN, S.; SPATSCHECK, O.; WANG, D.: *Accurate, Scalable In-Network Identification of P2P Traffic Using Application Signatures*, [online]. [cit. 2014-04-26]. URL http://ra.ethz.ch/CDstore/www2004/docs/1p512.pdf

[15] Skype; Microsoft: *What is VoIP (Voice-over-IP)*, [online]. [cit. 2014-04-26]. URL http://www.cisco.com/c/en/us/products/unified-communications/networking_solutions_products_genericcontent0900aecd804f00ce.html

[16] Skype(Microsoft): *O programu Skype*, [online]. 2014 [cit. 2014-04-26]. URL http://www.skype.com/cs/about

[17] SOLDANI, C.: *Peer-to-Peer Behaviour Detection by TCP Flows Analysis*. Dizertační práce, University of Liege, Computer Sciences, 2004.

[18] STARIGAZDA, M.: *Detekce sítí P2P pomocí NetFlow*. Diplomová práce, Brno: FIT VUT v Brně, 2013.

[19] TechTerms.com: *Framework*, [online]. 2013 [cit. 2014-04-26]. URL http://www.techterms.com/definition/framework

[20] VRBA, J.: *Detekce P2P sítí*. Diplomová práce, Brno: FIT VUT v Brně, 2009.

[21] WAGNER, A.; et al.: *Identifying P2P Heavy-Hitters from Network-Flow Data*, [online]. [cit. 2014-05-08]. URL http://www.cert.org/flocon/2005/presentations/Wagner-HeavyHitters-FloCon2005.pdf

[22] ZSEBY, T.: *IP Flow Information Export (IPFIX) Applicability*, [online]. 2009 [cit. 2014-05-17]. URL http://tools.ietf.org/html/rfc5472

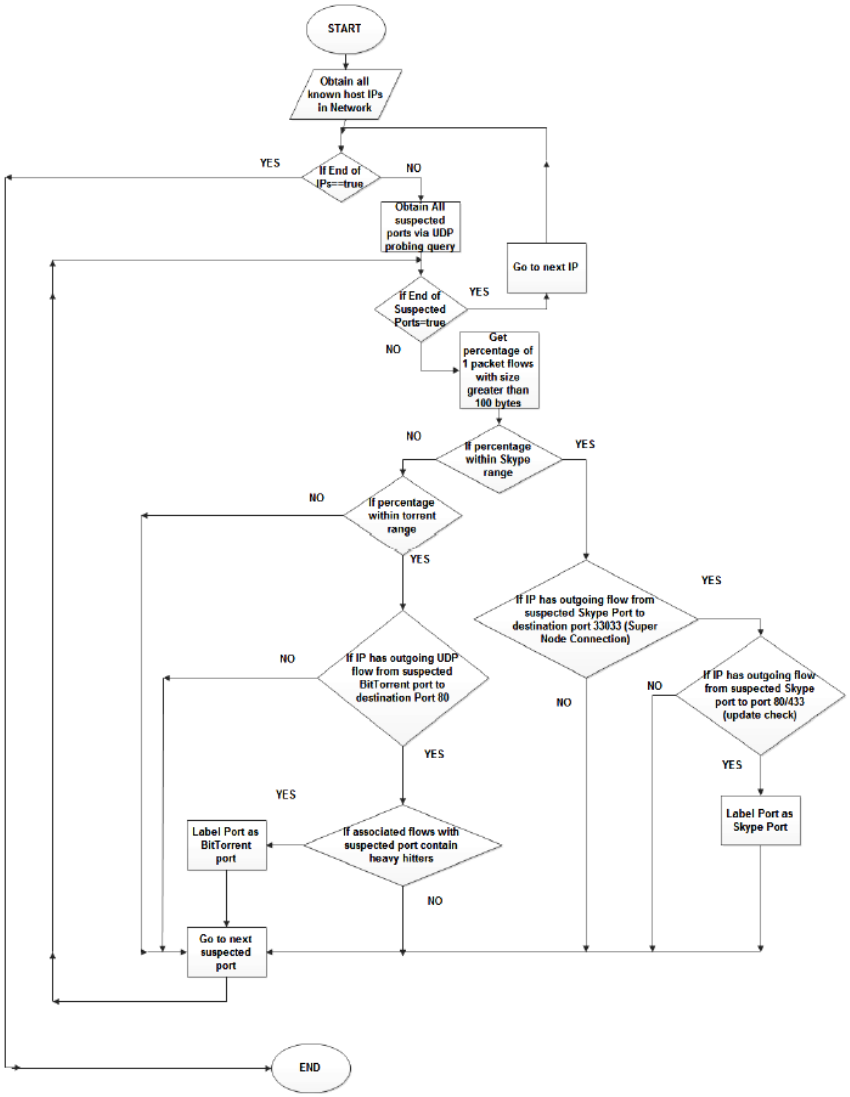
Dodatek A

Obsah CD

- Text práce ve formátu PDF včetně zdrojových kódu této práce.
- Zdrojové soubory modulu a systému Nemea + `Makefile` pro překlad.
- Testovací sady dat použité v kapitole 7 včetně výsledků.
- soubor README s návodem na instalaci.

Dodatek B

Klasifikační schéma



Obrázek B.1: Extrakce uživatelů podezřelých z P2P komunikace. Převzato z [3].