VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

ANALYZÁTOR SÍŤOVÝCH PROTOKOLŮ

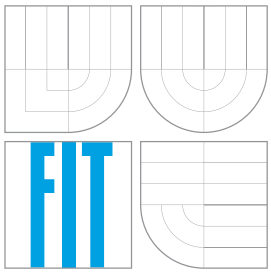
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

MIROSLAV SOBOTKA

BRNO 2010

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

ANALYZÁTOR SÍŤOVÝCH PROTOKOLŮ

NETWORK PROTOCOL ANALYZER

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — MIROSLAV SOBOTKA
AUTHOR

VEDOUCÍ PRÁCE — Ing. JIŘÍ TOBOLA
SUPERVISOR

BRNO 2010

Zadaní bakalářské práce

1. Seznamte se s architekturou TCP/IP a se síťovými protokoly aplikačních vrstev.
2. Prostudujte možnosti detekce typu protokolu na základě obsahu paketů. Zaměřte se zejména na aplikační vrstvu, dynamické internetové protokoly a algoritmy jejich detekce pro vysokorychlostní sítě.
3. Navrhněte systém pro automatickou detekci vybraných protokolů.
4. Implementujte prototyp navrženého systému.
5. Funkčnost prototypu demonstrujte na vhodně zvoleném vzorku dat.
6. Zhodnoťte dosažené výsledky a diskutujte možnosti dalšího rozšíření systému.

Abstrakt

Tato práce se zabývá problematikou identifikace síťových protokolů postavených nad protokoly rodiny TCP/IP. První část popisuje možné techniky a postupy identifikace, které jsou následně využity při návrhu analyzátoru síťových protokolů. Další část je věnována implementaci a testování navrženého analyzátoru. Tento analyzátor je určen pro operační systém GNU/Linux a využívá především knihoven libpcap a libpcre. V závěru práce jsou řešeny výsledky rychlosti a přesnosti identifikace, problémy a možná vylepšení.

Abstract

The thesis deals with the identification of network protocols, which are superior to the family of TCP/IP protocols. The first part of the work describes possible techniques and identification procedures, which are afterwards applied in design of the network protocols analyzer. Next part of the work is engaged in the implementation and testing of the analyzer. The abovementioned analyzer is intended for GNU/Linux and for its work makes use of libpcap and libpcre libraries. In the conclusion of the thesis there are discussed results of speed and precision, occurred problems and possible improving of the program.

Klíčová slova

Sítě, Identifikace, Identifikace obsahu, Statistiky, TCP/IP, Analyzátor protokolů

Keywords

Networks, Identification, Pattern matching, Statistics, TCP/IP, Protocol Analyser

Citace

Miroslav Sobotka: Analyzátor síťových protokolů, bakalářská práce, Brno, FIT VUT v Brně, 2010

Analyzátor síťových protokolů

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Jiřího Toboly

. . . . . .
Miroslav Sobotka
19. 5. 2010

Poděkování

Rád bych poděkoval svému vedoucímu bakalářské práce Ing. Jiřímu Tobolovi, který mi vnikl spoustu zajímavých myšlenek, které jsem následně ve své práci rozvíjel.

Obsah

Kapitola 1

Úvod

V dnešní době, kdy internet dorazil do téměř každé domácnosti, narůstá celosvětová síť do mamutích rozměrů. Úměrně s počtem přípojení se však zvedá počet uživatelů, kteří generují v celosvětové síti internet provoz. Komunikují přes instant messengery, surfují po internetu, stahují hudbu a videa, hrají online hry, posílají e-maily a mnoho dalšího.

Všechny tyto druhy komunikace či programy komunikují mezi sebou pomocí různých protokolů. Tato bakalářská práce se zabývá návrhem a realizací analyzátoru síťových rozhraní. Aplikace je určena na zachytávání paketů na síťovém rozhraní a jejich následnou analýzu.

Inspirací pro můj analyzátor se stal velmi oblíbený a využívaný program Wireshark [1], který se používá na komplexní analýzu toku dat na síťových rozhraních. Silnou stránkou programu Wireshark je podpora velké řady protokolů, pěkné grafické znázornění a velké možnosti filtrování již zachycených dat. Mezi jeho nevýhody však patří to, že nedokáže zpracovávat data zachycená na síťových rozhraních při velkých rychlostech. Nestíhá díky tomu zachytávat všechna data a velmi vytěžuje procesor. Další nevýhodou je pouze jeden typ identifikace protokolů. Program Wireshark totiž identifikuje protokoly pouze pomocí čísel portů.

Nejčastěji a hlavně nejsnáze se totiž provádí identifikace toku dat právě podle portů. Tento způsob identifikace protokolů však není vždy funkční, protože služby lze spouštět i na jiných než standardních portech. Z toho důvodu bude v analyzátoru zabudován identifikační automat, který rozpoznává toky nejen podle portů, ale i podle obsahu toku. Čím více má uživatel či program o toku infomací, tím přesněji může určit, o co se jedná.

Často o tocích na síti nepotřebujeme mít velmi rozsáhlý protokol dění, jaké nám poskytuje Wireshark, ale stačí nám pouze rámcově vědět, co se na síti děje. Proto analyzátor vypisuje jen ty nejpodstatnější informace o tocích, identifikovaných protokolech a završuje to statistikou o provozu.

Práce je dále členěna do několika kapitol, které jsou organizovány následovně. Kapitola 2 shrnuje teorii z oboru počítačových sítí důležitou pro návrhu analyzátoru síťových protokolů a rozebírá možnosti identifikace protokolů. V kapitole 3 je popsán vlastní návrh analyzátoru a řešení detekce protokolů. Následující kapitola se zabývá implementací a popisem řešení, jak analyzátor identifikuje síťové protokoly. Předposlední kapitola se věnuje testování analyzátoru, vyhodnocení jeho výkonnosti a přesnosti identifikace protokolů. Poslední kapitolou je shrnutí dosažených výsledků a zhodnocení vlastností a možností analyzátoru z hlediska dalšího vývoje.

Kapitola 2

Teorie

2.1 Počítačové sítě a internet

Počítačová síť je označení pro technické prostředky umožňující komunikaci mezi počítači podle určitých, předem daných pravidel za účelem výměny dat nebo zasílání zpráv. Internet je největší počítačová síť na světě, občas nazývána *síť sítí*. Jako pravidla pro výměnu dat se u internetu používají protokoly z rodiny TCP/IP.

2.1.1 Architektura sítě protokolu TCP/IP

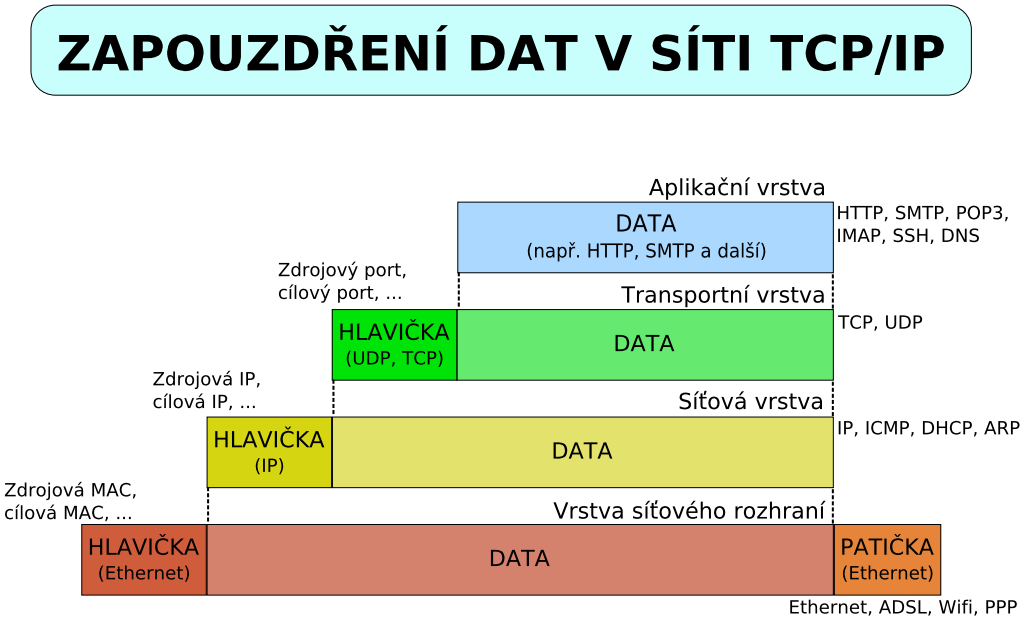
Aby nebyla komunikace mezi dvěma uzly na síti tak složitá, je rozdělena do vrstev, které znázorňují hierarchii vlastní činnosti protokolu. Každá vrstva je základ pro tu, co je jí nadřazena a zároveň využívá služby vrstvy nižší. Vrstvy mezi sebou komunikují nezávisle podle svého komunikačního protokolu. Architektura TCP/IP je rozdělena do těchto 4 vrstev:

- vrstva síťového rozhraní (network interface);
- síťová vrstva (network layer);
- transportní vrstva (transport layer);
- aplikační vrstva (application layer).

Znázornění této architektury je dobře vidět na obrázku 2.1 [18], kde je vidět, jak jsou jednotlivé vrstvy do sebe zapouzdřeny.

Vrstva síťového rozhraní

Nejnižší vrstva síťového rozhraní, která přistupuje k vlastnímu fyzickému rozhraní sítě. Identifikace v této vrstvě probíhá pomocí MAC adresy, což je adresa zařízení. Komunikaci zajišťují například technologie Ethernet, Wi-Fi (IEEE 802.11), token ring apod. V současné době je nejpoužívanější síťovou technologií Ethernet či Wi-Fi. Jejich hlavičky si nyní krátce rozebereme.



Obrázek 2.1: Zapouzdření dat v síti TCP/IP

Rámec Ethernetu dle normy IEEE 802.3 [11] je vidět na obrázku 2.2 a skládá se z následujících částí:

Preambule (7 bajtů) – vzorek dat, který zajišťuje časovou synchronizaci. Je tvořen střídajícími se hodnotami 0 a 1.

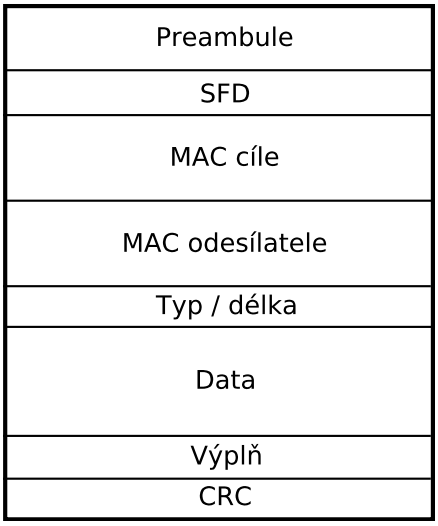
SFD (Starting Frame Delimiter) (1 bajt) – identifikace začátku rámce. Je podobně jako Preambule tvořen posloupností střídajících se nul a jedniček, jen poslední nula je nahrazena hodnotou jedna. Obsah je tedy konstantní: *10101011*.

MAC cíle (6 bajtů) – adresa cílového uzlu. Jedná se o buď individuální adresu konkrétního uzlu (unicastová adresa), adresu skupiny více uzlů (multicastová adresa) nebo všeobecnou adresu (broadcastová) adresa.

MAC zdroje (6 bajtů) – adresa zdrojového uzlu. Vždy individuální adresa.

Typ/délka (2 bajty) – význam této položky je dán její hodnotou. Pokud je hodnota větší, než 600 hexadecimálně, obsah určuje typ dat vyšší vrstvy. Pokud je hodnota menší, udává velikost dat.

Data (46 až 1500 bajtů) – obsahuje vlastní data vyšší vrstvy. Pokud je položka menší, než požadovaných 46 bajtů, je vyplněna položka Výplň právě 46 bajty o libovolném obsahu. Důvodem tohoto kroku je detekce kolizí rámců v rámci segmentu sítě, bez něhož by nemusela správně fungovat.



Obrázek 2.2: Hlavička Ethernetového rámce

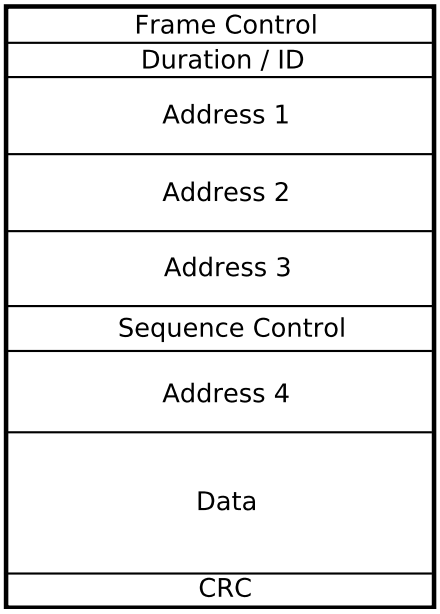
Výplň (až 46 bajtů) – viz výše.

CRC (Frame Check Sequence) [15] (4 bajty) – dvaatřicetibitový kontrolní kód, který se počítá ze všech polí s výjimkou Preambule a SFD. Slouží ke kontrole správnosti dat – příjemce si jej vypočítá z obdrženého rámce a pokud výsledek nesouhlasí s hodnotou pole, rámec zahodí jako vadný.

Oproti Ethernetu je rámec Wireless LAN (Local Area Network) dle normy IEEE 802.11 [6] výrazně složitější, jak je patrné z obrázku 2.3. Význam jednotlivých položek je následující: [4]

Frame Control (2 bajty) – bitové pole, obsahující informace o rámci. Obsahuje tyto položky:

- Protocol version (2 bity) – verze protokolu rámce. Pro IEEE 802.11 je to hodnota 0, všechny ostatní jsou rezervovány.
- Type (2 bity) – určuje vlastní typ rámce. Může nabývat těchto hodnot: *00* management, *01* control a *10* data. Hodnota *11* je rezervována.
- Subtype (4 bity) – upřesnění typu rámce. Všechny možnosti jsou uvedeny v tabulce normy IEEE 802.11 z roku 1999, strana 36.
- To DS (Distribution system) (1 bit) – je-li tento bit nastaven, paket je určen přístupovému bodu, který jej přepošle do distribučního systému a to včetně případu, kdy cílová stanice leží ve stejné buňce tj. ve stejném BSS (Basic Service Set) jako odesílatel paketu.
- From DS (1 bit) – bit je nastaven, přichází-li paket z distribučního systému.
- More Fragments (1 bit) – je-li nastaven tento bit, znamená to, že rámec je rozdělen na více fragmentů.
- Retry (1 bit) – nastavený bit říká, že tento fragment je znovu poslán. Je to proto, aby přijíjemce poznal duplikovaný paket v případě, že byl potvrzovací paket ztracen.



Obrázek 2.3: Hlavička IEEE 802.11 rámce

- Power Management (1 bit) – tento bit indikuje, že stanice, která rámec vyslala, bude po odeslání ve stavu PowerSave, nebo naopak přechází do stavu Active.
- More Data (1 bit) – značí, že jsou v bufferu uloženy další rámce pro tuto stanici.
- WEP (1 bit) – je-li nastaven, tělo je zašifrováno pomocí algoritmu WEP (Wired Equivalent Privacy).

Duration / ID (2 bajty) – význam jeho hodnoty záleží na typu rámce. Má hodnotu buď ID stanice nebo hodnotu používanou pro účely výpočtu NAV (Network Allocation Vector).

Address 1 (6 bajtů) – adresa přijímejemce. Je-li nastaven bit *To DS*, je to adresa přístupového bodu, jinak je to adresa cílové stanice.

Address 2 (6 bajtů) – adresa odesílatele (tedy aktuálního odesílatele paketu). Je-li *From DS* nastaven, je to adresa AP, jinak adresa odesílací stanice.

Address 3 (6 bajtů) – zbývající adresa, která v rámci chybí. Je-li *From DS* nastaven, je to adresa původního odesílatele paketu, je-li nastaven *To DS*, je to adresa cílové stanice.

Sequence control (2 bajty) – reprezentuje pořadí fragmentů rámce.
Obsahuje dvě pole – *Frame number*: číslo rámce a *Sequence number*: značí pořadí fragmentu v rámci.

Address 4 (6 bajtů) – tato adresa je použita ve speciálním případě, kdy je použit bezdrátový *DS* a rámec je posílán mezi dvěma AP (Access Point). V tomto případě je nastaven bit *From DS* i *To DS* a adresa značí adresu původní odesílací stanice. V adrese 3 je pak adresa příjemce.

Tělo rámce – obsahuje vlastní přenášená data. Tělo rámce nemá pevně danou délku.

CRC – dvaatřicetibitový kontrolní součet. Vypočítá se z dat v MAC hlavičce a z těla rámce před odesláním paketu. Z přijatého paketu se stejným algoritmem spočítá další kontrolní součet. Pokud nově vypočítaný součet nesouhlasí se součtem v přijatém paketu, je paket porušen a je vyžádán znovu.

Síťová vrstva

Tato vrstva zajišťuje pomocí protokolů vlastní směrování, adresaci a posílání datagramů s daty. Implementována je ve všech prvcích sítě, tj. jak směrovačích, tak koncových zařízeních. Protokoly, které tato vrstva používá, jsou pak IP, ARP, ICMP, IGMP, RARP, IGRP a IP Sec. Díky IP protokolu zde vzniká pojem IP adresa (adresa koncového bodu).

Transportní vrstva

„Transportní vrstva poskytuje transparentní, spolehlivý přenos dat s požadovanou kvalitou. Vyrovnává různé vlastnosti a kvalitu přenosových sítí. Provádí převod transportních adres na síťové, ale nestará se o směrování.“ [19]

Její implementace se nachází až v koncových zařízeních a poskytuje protokoly TCP (Transmission Control Protocol) a UDP (User Datagram Protocol), které komunikují pomocí paketů či datagramů, obsahující data. V rámci protokolu TCP se používá pro datagram pojem paket. Spojovaný TCP protokol znamená, že komunikace vytváří virtuální okruh spojení, čímž zaručuje spolehlivý přenos dat. Musí tedy proběhnout nejdříve ustanovení spojení a až poté se přenášejí data. Opakem toho je nespojovaný UDP protokol, který nemá takovouto režii (nevytváří virtuální okruh) a hned v prvním datagramu přenáší data. Jak TCP, tak UDP obsahují ve své hlavičce informace o portu, který identifikuje službu z aplikační vrstvy.

Aplikační vrstva

Aplikace mezi sebou komunikují pomocí aplikačních protokolů. Aplikační protokol je formální popis komunikace mezi aplikacemi. Na přenos mezi nimi se využívají protokoly z transportní vrstvy. Tam, kde nezáleží na tom, jesti byla data doručena, se využívá protokolu UDP. V případě, že je třeba potvrdit doručení dat, se využije protokol TCP. Existují však výjimky, které využívají obou protokolů zároveň (například služba DNS [10]).
Identifikace cílového bodu se skládá z IP adresy a portu.

2.1.2 Síťová vrstva – IP

IP (Internet Protocol) je datový protokol používaný pro přenos dat přes paketové sítě. Tvoří základní protokol dnešního Internetu.

Data se v IP síti posílají po blocích nazývaných datagramy. Jednotlivé datagramy putují sítí zcela nezávisle, na začátku komunikace není potřeba navazovat spojení či jinak „připravovat cestu“ datům, přestože spolu třeba příslušné stroje nikdy předtím nekomunikovaly.

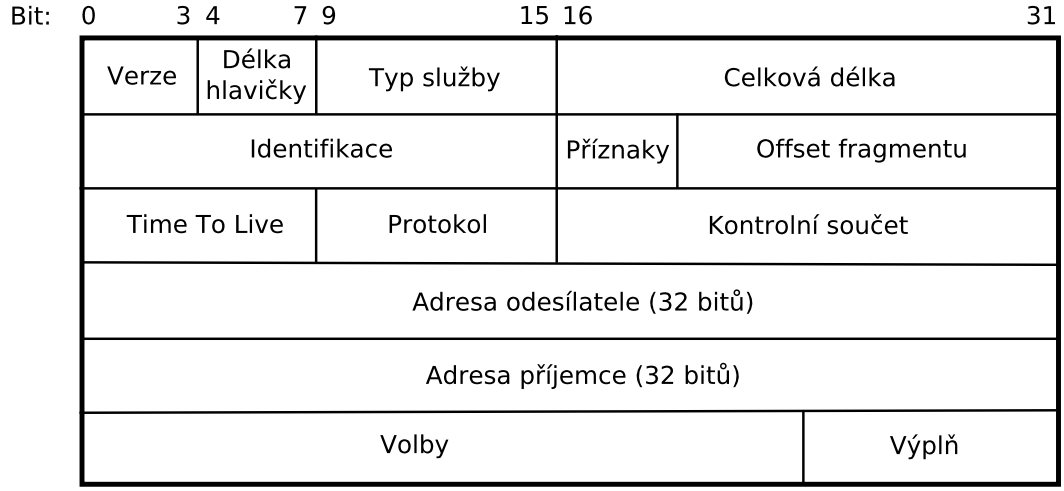
IP v doručování datagramů poskytuje nespolehlivou službu, označuje se také jako „nejlepší úsilí“ (best effort). Všechny spoje na trase se snaží podle svých možností datagram poslat blíže k cíli, ale nezaručují prakticky nic. Datagram vůbec nemusí dorazit, může být naopak doručen několikrát a neručí se ani za pořadí doručených datagramů. Pokud aplikace potřebuje spolehlivost, je potřeba ji implementovat v jiné vrstvě síťové architektury, typicky protokol bezprostředně nad IP tj. TCP.

Pokud by síť často ztrácela datagramy, měnila jejich pořadí nebo je poškozovala, výkon sítě pozorovaný uživatelem by byl malý. Na druhou stranu příležitostná chyba nemívá pozorovatelný efekt. Navíc se obvykle používá vyšší vrstva, která ji automaticky opraví. [16]

Dnes vládne internetu IPv4 (čtvrtá revize IP), která je detailně zdokumentována v RFC 791 [7]. Počátkem devadesátých let 20. století se stalo zřejmé, že technologie CIDR (Classless Inter-domain Routing) [13], která zlepšuje efektivitu rozdělení adresního prostoru, nestačí k odvrácení vyčerpání adresního prostoru IPv4. Bylo nutné dále upravit protokol IPv4. Začalo se pracovat na IPng – IP nové generace, dnes známé jako IPv6 (šestá revize IP) [2].

IPv4

IPv4 je čtvrtá revize IP, která se jako první masově rozšířila. Datagram obsahuje hlavičku, za kterou následují data. Hlavička je zarovnána násobkem čtveřice bajtů. Na níže uvedeném obrázku 2.4 je znázorněna hlavička IPv4 datagramu. Hned za hlavičkou datagramu následují data.



Obrázek 2.4: IPv4 hlavička

Verze (4 bity) – číslo verze IP, v tomto případě 4.

Délka hlavičky (4 bity) – udává velikost hlavičky v bajtech.

Typ služby (8 bitů) – původním záměrem bylo, umožnit odesílateli zvolit si parametry pro přenos. V praxi na toto nedošlo a dnes je pole používáno pro QoS (Quality of Service) [17].

Celková délka (16 bitů) – udává velikost celého datagramu včetně hlavičky v bajtech.

Identifikace (16 bitů) – při odeslání se nastaví datagramu unikátní náhodné číslo. Při fragmentaci se pak lehce určí, které fragmenty patří k sobě.

Příznaky (3 bity) – pole může nabývat dvou hodnot: *More fragments* a *Don't fragment*. První je nastavena v momentě, kdy je datagram fragmentován a není to první fragment. Druhá možnost výslovně fragmentaci zakazuje.

Offset fragmentu (13 bitů) – určuje, kde začíná v původním datagramu fragment. Číslo je celistvý násobek čísla 8.

TTL (Time To Live) (8 bitů) – každý směrovač sníží hodnotu o 1 a pokud nabude hodnoty 0, datagram směrovač zahodí. Je to jednoduchá a účinná ochrana proti zacyklení.

Protokol (8 bitů) – určuje, kterému protokolu z vyšší vrstvy se mají předat data. Jednotlivé hodnoty jsou popsány v RFC 1700 [14]. Příklad: TCP se označuje číslem 6, UDP číslem 17.

Kontrolní součet (16 bitů) – Je vypočten pouze z hlavičky a v případě neshody je datagram zahozen. Zabezpečení dat IP nijak neošetřuje.

Adresa odesílatele (32 bitů) – adresa rozhraní, odkud byl datagram vyslán.

Adresa cíle (32 bitů) – Adresa síťového rozhraní, kam má být datagram dopraven.

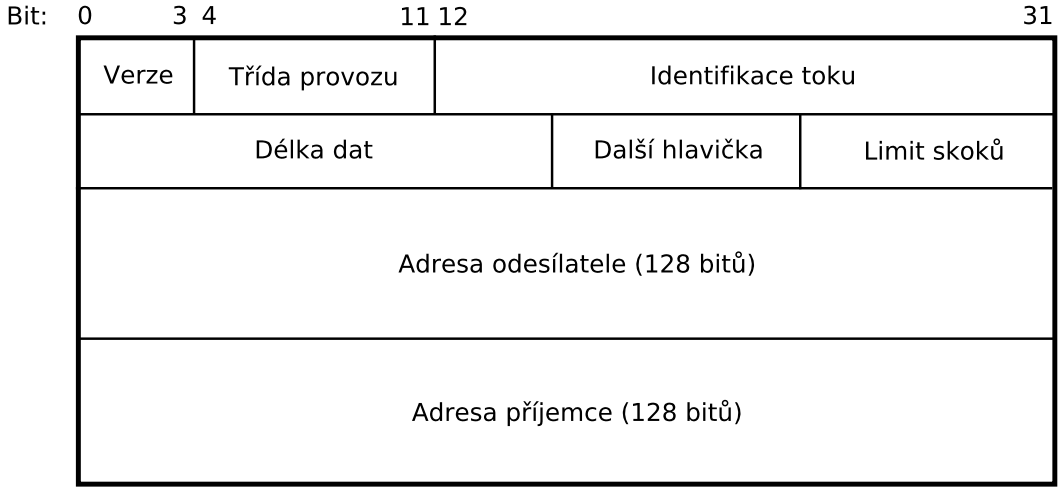
Volby – různé rozšiřující informace či požadavky. Prakticky se nevyužívá.

Výplň – nenese informaci (je vyplněna 0), slouží pouze na zvětšení hlavičky na násobek 32 bitů.

Data – vlastní data, které se přepravují. Velikost dat je maximálně (65535 – délka záhlaví) bajtů.

IPv6

IPv6 je šestá revize IP, která je nástupcem IPv4. Na níže uvedeném obrázku 2.5 je znázorněna hlavní hlavička IPv6 datagramu. Počet položek byl zredukován na pouhých 8 polí o fixní délce 40 bajtů. Za povinným záhlavím mohou ale místo dat následovat další volitelná záhlaví proměnné délky, určená buď pro zpracování v koncových uzlech nebo směrovačích. Jestli bude následovat nějaký vyšší protokol, nebo budou specifikovány volitelné hlavičky, určuje pole *Další hlavička*.



Obrázek 2.5: IPv6 hlavička

Verze (4 bity) – číslo verze IP, v tomto případě 6.

Třída provozu (8 bitů) – položka umožňuje identifikovat prioritu datagramu. Ovlivňuje zpracování datagramu ve směrovačích i koncovém uzlu.

Identifikace toku (20 bitů) – označuje datagramy vyžadující extra zpracování při zpracování ve směrovačích. Označení toku a předchozí pole *Třída provozu* umožňují přímo řešit priority provozu, QoS a řízení využití šířky pásma.

Délka dat (16 bitů) – udává celkovou velikost datagramu kromě velikosti hlavní hlavičky.

Další hlavička (8 bitů) – identifikace typu záhlaví, které následuje za povinným záhlavím IPv6 datagramu. Pokud není další záhlaví použito, identifikuje protokol vyšší vrstvy.

Limit skoků (8 bitů) – je to číslo, které udává maximální počet routerů na cestě datagramu od zdroje k cíli. Každý router po cestě zmenší toto číslo o jedničku. Pokud toto číslo dosáhne nuly, je datagram považován za neplatný. Routery po cestě mohou toto číslo měnit, protože IPv6 hlavička neobsahuje kontrolní součet. Tím pádem se zmenšuje režie pro zpracování datagramu na routeru. Maximální počet skoků v IPv6 může být 255, což daleko překračuje potřeby všech v současnosti navrhovaných sítí.

Adresa odesílatele (128 bitů) – adresa zdrojového uzlu, odkud byl datagram vyslán.

Adresa příjemce (128 bitů) – adresa cílového uzlu, pokud není použita rozšiřující směrovací hlavička.

Za povinnou hlavičkou IPv6 mohou následovat další volitelné hlavičky (Extension Headers). Ukazují na sebe stejným způsobem, jako povinná hlavička. Obsahují pole *Další hlavička*, které idenfikuje hlavičku, která bude následovat. Pokud nebude specifikována další volitelná hlavička, bude specifikován protokol z vyšší vrstvy. Čísla protokolů jsou povětšinou shodná s čísly z IPv4. Pořadí hlaviček, ani jejich počet (může se tedy vyskytnout někerá hlavička vícekrát), není normou určen, pouze doporučen.

IPv6 definuje následující volitelné hlavičky:

Hop-by-Hop options header – obsah tohoto záhlaví vyžaduje zpracování každým směrovačem na cestě.

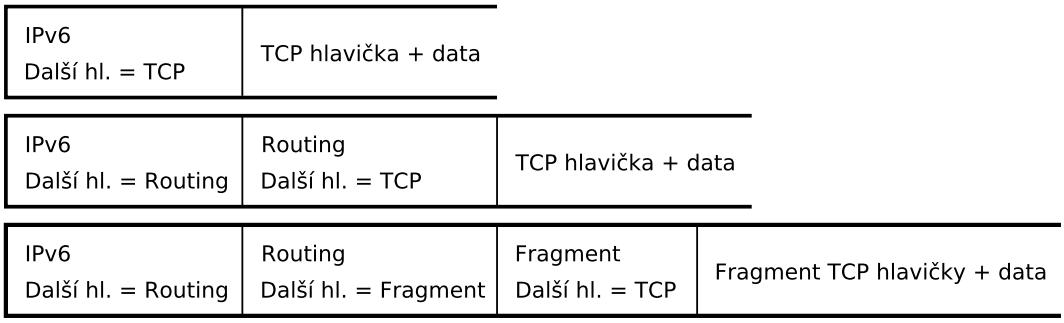
Destination options header – volitelné informace pro příjemce podle směrovacího záhlaví.

Routing header – hlavička obsahuje adresy směrovačů, kterými musí datagram projít.

Fragment header – obsahuje informace určené pro fragmentaci a znovusestavení datagramů. Fragmentace může být provedena pouze zdrojovou stanicí, nikoliv směrovači po cestě.

Encapsulating security payload header – ochrana přenášených dat v datagramu.

Authentification header – zabezpečuje integritu dat a věrohodnost datagramu.

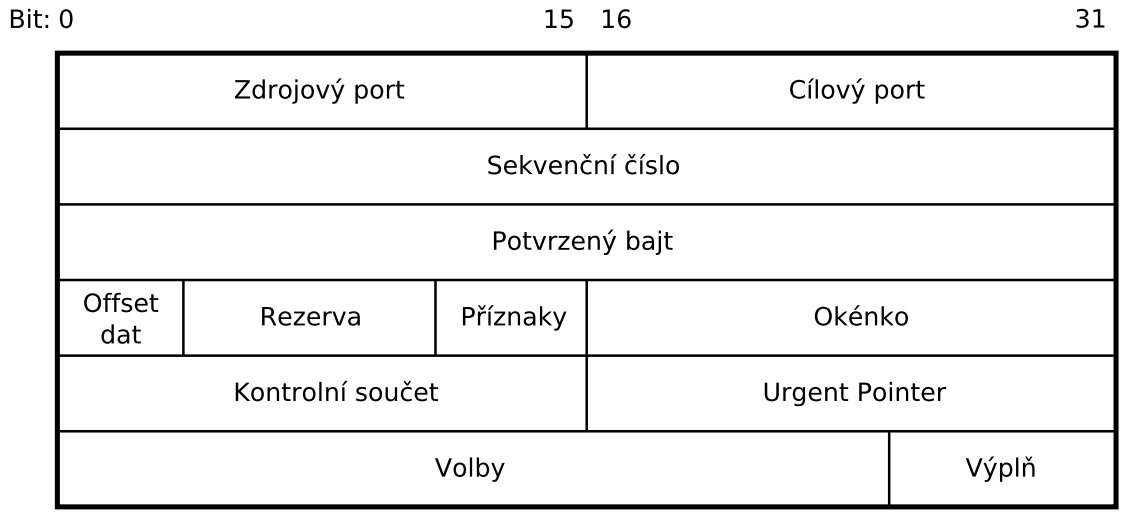


Obrázek 2.6: Příklad propojení hlaviček v IPv6 datagramu.

2.1.3 Transportní vrstva – TCP, UDP

TCP

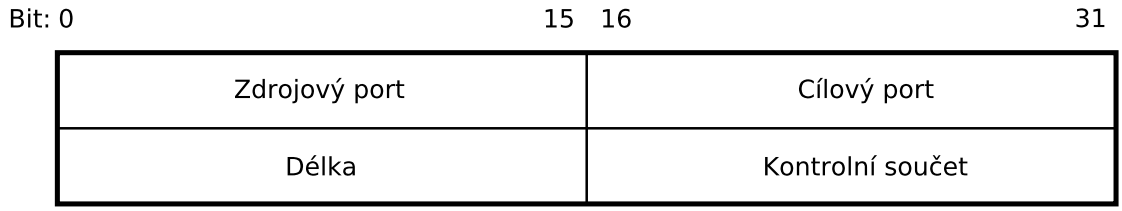
TCP protokol zaručuje spolehlivý přenos dat. Než je možno přenášet data, musí dojít k navázání spojení (anglicky nazýváno jako *handshake*), čímž se vytvoří virtuální okruh. Cílová aplikace začne naslouchat na určitém portu, kde čeká na paket s příznakem SYN. Když přijde, vyšle paket s dvěma příznaky SYN a ACK (potvrzení přijetí SYN). To přijme odesílatel a vyšle paket s příznakem ACK. Spojení je navázáno, můžou se začít přenášet data. Cíl posílá prázdný paket s potvrzením. Pokud by se odesílateli potvrzení nevrátilo do rozumné doby (round-trip time, RTT), vypršel by odesílatelův časovač a (pravděpodobně ztracená) data by vyslal znovu. Po odeslání všech dat vyšle odesílatel paket s příznakem FIN. Příjemce odpoví paketem s příznaky FIN a ACK (potvrzení přijetí FIN). Posledním paketem komunikace je pak paket pro příjemce s příznakem ACK.
Spojení může být ukončeno také nestandardním způsobem bez čekání na druhou stranu, a to paketem s příznakem RST. Bližší popis protokolu je v RFC 793 [8]. Hlavička TCP paketu je znázorněna na obrázku 2.7.



Obrázek 2.7: Hlavička TCP paketu

UDP

Jak již bylo zmíněno, UDP nezaručuje spolehlivé doručení datagramu do cíle. To ale nutně neznamená, že je to nepoužitelný protokol. Této vlastnosti se s výhodou využívá v aplikacích, kde se se ztrátami počítá a není vhodné, aby se ztrácel čas novým odesíláním (starých) nedoručených zpráv. Příkladem jsou například VoIP (Voice over IP) nebo streaming videa. Další výhodou je celková jednoduchost protokolu. Jak je vidět na obrázku 2.8 hlavička neobsahuje příliš položek. Neobsahuje především identifikátor, takže není zaručeno, v jakém pořadí datagram příjemci přijde. Více informací lze nalézt v RFC 768 [12].



Obrázek 2.8: Hlavička UDP paketu

2.2 Identifikace protokolů

Existují různé techniky, jak identifikovat protokoly. Nejjednodušší technika identifikace protokolu je podle čísla portu, přes který komunikuje. Seznam čísel portů a jim přiřazených protokolů udržuje organizace IANA (Internet Assigned Numbers Authority) [5]. Obsahuje dohodnuté pojmenování čísel portů rozdělených do tří skupin:

- velmi známé porty – port číslo 0 až 1023. Čísla portů by neměly být používány bez registrace u organizace IANA.
- registrované porty – port číslo 1024 až 49151. Čísla portů by neměly být používány bez registrace u organizace IANA.
- dynamické a/nebo soukromé porty – port číslo 49152 až 65535.

Ovšem tato technika může velmi snadno selhat, pokud je služba spuštěna na jiném portu, než na kterém by běžet měla. V tomto případě už tedy nemůžeme spoléhat na údaje, co nám poskytují protokoly transportní vrstvy a je nutné zařídit identifikaci jinak. Pokud protokol má nějakou svoji unikátní signaturu tj. například hlavičku, identifikace se stává snadnou. Problémem je, že hlavička se nachází typicky na začátku zasílaných dat. Pro tento případ se tedy vyžaduje především zachycení počátku spojení. Na vyhledávání signatur v datech se nejčastěji používají *regulární výrazy* (viz kapitola 2.2.1).

Co je však pro rozpoznávání obsahu nepřekonatelný problém, je stav, kdy obsah paketu je šifrován. Jediné, co v tomto případě lze, je pokusit se zjistit kterým nástrojem či knihovnou je paket či celý stream šifrován. Podrobnější identifikace protokolu tedy zde selhává.

Selže-li i tato technika, jsou další možností různé heurestiky či na statistikách založené odhady. Tyto metody se často aplikují na protokoly, kde není přesně určitelná signatura.

2.2.1 Regulární výrazy

Regulární výraz (anglicky regular expression), často označovaný jako regexp či regex je speciální řetězec znaků, který představuje určitý vzor pro textové řetězce. Regulární výrazy se proto nejčastěji používají ke kontrole dat zadávaných ve formulářích (například e-mailová adresa, rodné číslo či telefonní číslo) nebo „parsování“ kódu (např. HTML, XML či CSV).

Existují různé odnože regulárních výrazů. Nejznámější jsou Perl-compatible regulární výrazy (PCRE) a POSIX regulární výrazy. Ačkoliv většina novějších programovacích jazyků používá regulární výrazy odvozené od jazyka Perl, drobné rozdíly v implementaci tu stále přetrvávají. Hlavní výhodou pro programátora je výrazné zredukování a zpřehlednění kódu.

Kapitola 3

Návrh aplikace

Analyzátor bude zachytávat pakety, které přijdou na zvolený síťový adaptér nebo bude číst ze souboru, který mu bude předán jako parametr. Ze zachycených paketů vybere pouze TCP/IP pakety. V zachycených paketech poté začne hledat pakety s příznaky vytvoření virtuálního okruhu – TCP streamu. Pokud je nalezne, vytvoří si o něm záznam. Datové pakety, které patří do známého TCP streamu budou zaznamenány a zpracovány, všechny ostatní datové pakety budou zahozeny. V momentě, kdy ve virtuálním okruhu začnou proudit vlastní data, analyzátor se pokusí rozpoznat, který druh dat proudí okruhem. Veškeré dění se bude vypisovat na obrazovce ve formě statistik. Dále se bude vypisovat poslední úspěšně navázané virtuální okruhy spolu s rozpoznanými údaji.

3.1 Knihovny

Analyzátor ke svému běhu bude využívat hotové knihovny, které budou nyní stručně popsány.

3.1.1 libpcap

Základní kámen celého analyzátoru je knihovna pcap (Packet Capture Library, zkráceně libpcap). Poskytuje API (Application Program Interface) pro čtení dat na síťovém rozhraní podle předem daného filtru. Jméno rozhraní se buď knihovně předáno analyzátorem, případně si knihovna umí nějaké zařízení schopné čtení sama najít. Knihovnu lze využít pro na zápis dat – dokáže vyslat do sítě nový paket.

Knihovna taktéž podporuje zachytávání v tzv. „promiskuitním módu“. To znamená, že síťové rozhraní přijímá všechny pakety, pokud síť není přepínaná (na paketových sítích jako je Ethernet) nebo v případě, že je bezdrátový adaptér přepnut do „monitor módu“ (na Wi-Fi adaptérech). Z toho plyne, že na síťové rozhraní může dorazit unicast nebo multicast paket, který mu nepatří, ale adaptér jej akceptuje. Toho využívá především u pasivního sledování sítě.

3.1.2 libpcre

Knihovna PCRE poskytuje API, které implementuje regulární výrazy pomocí stejné syntaxe a sémantiky jako Perl jen s několika rozdíly. Podporuje rozšiřující syntax uvedenou spolu s .NET jazyky a je kompatibilní s jazykem Javascript.

3.1.3 libncurses

Knihovna ncurses se využívá pro práci s terminálem. Poskytuje API pro psaní rozhraní nezávisle na typu zvoleného terminálu. Tato knihovna je sada nástrojů, které běží pod emulátorem terminálu. Umožňuje optimalizovat změny na obrazovce s cílem snížit zpoždění překreslování terminálu, které je nejvíce patrné při připojení na vzdálený terminál.

3.1.4 libpthread

Pro práci s vlákny na operačních systémech, které splňují standard POSIX (Portable Operating System Interface), se používá standard IEEE 1003.1c (Institute of Electrical and Electronics Engineers). IEEE 1003.1c je standard pro práci s více vlákny v rámci jedné aplikace. Knihovna libpthread toto rozhraní implementuje. Obsahuje funkce pro vytváření a zánik vláken, rozhraní pro plánování práce vláken a mutex jako pomůcku na zamykání sdílených zdrojů aplikace, aby nedocházelo k nechtěnému přepisování dat v paměti.

3.2 Rozpoznávaní

Rozpoznávaní paketů bude probíhat dvěma způsoby. První bude rozpoznání podle čísel podle portů v TCP hlavičce. Pro rozpoznávání bude využita tabulka, kterou udržuje organizace IANA.

Analyzátor se pokusí v tabulce nalézt číslo zdrojového a cílového portu. Díky tomuto kroku se získají až dva identifikační údaje. Není zaručeno, že budou nalezeny právě dva. Protože pokud se aplikace připojuje na server, zdrojový port je náhodné vysoké číslo teoreticky v rozsahu 49152 až 65535, které nevypovídá o tom, kam se aplikace připojuje. Z tohoto důvodu byla zavedena jednoduchá technika. Dokáže-li přeložit oba porty na název, vezme jako název protokolu pouze ten s nižším číslem portu.

Druhý použitý způsob rozpoznání paketu je podle obsahu TCP streamu. Režijní pakety protokolu TCP neobsahují data. Je tedy nutné počkat, až v rámci virtuálního okruhu resp. TCP streamu, začnou přicházet pakety s daty. Každý protokol lze identifikovat pomocí unikátní signatury. Nejjednodušší způsob na hledání signatur, je použít regulární výrazy, které se budou aplikovat na data v paketu. Pro ulehčení bude využita již existující sada regulárních výrazů z L7 filtru [9].

3.3 Souhrnné statistiky a výstupy

Analyzátor bude průběžně vypisovat informace o své úspěšnosti. V krátkém intervalu bude uživatele informovat o tom, kolik se mu podařilo zachytit paketů. Další výpisy se odvíjejí od zachycení až po idenfitikaci. Jako první to je statistika verze IP, další je počet úspěšně syntézou složených TCP/IP paketů do TCP streamů. Následuje stastistika počtu identifikovaných streamů podle portů, kterou doplňuje statistika podle obsahu. Výpis završuje posledních 6 navázaných a identifikovaných streamů.

Kromě pravidelného výstupu na monitor umí analyzátor zapisovat vše do souboru, kde lze tedy nalézt kompletní záznam práce.

Kapitola 4

Implementace aplikace

Tato kapitola popisuje implementaci analyzátoru síťových protokolů. Kapitola je rozdělena do tří částí:

1. část: Příprava na zachytávání – spuštění analyzátoru, načtení potřebných dat, příprava síťových zařízení.
2. část: Zachytávání, zpracování a zobrazení výsledků – analýza a syntéza paketů, výpočet statistiky
3. část: Ukončení zpracování – vyhodnocení výsledků

4.1 Příprava na zachytávání

4.1.1 Zařízení

Analyzátor nejprve načte předané parametry, se kterými byl spuštěn. Jedním z důležitých parametrů analyzátoru je, zda-li má skenovat v reálném čase na síťovém rozhraní, nebo jestli má otevřít soubor se záznamem a ten zpracovat (offline mód). Tímto se zabývá funkce `prepare_capture()`. Pokud se bude pracovat v offline módu, zavolá se funkce knihovny pcap `pcap_open_offline()`, které se předá soubor s uloženým záznamem. Pokud se bude skenovat v reálném čase, příprava probíhá ve volání dvou funkcí: ověření funkčnosti rozhraní pomocí funkce `pcap_lookupnet()` a otevření rozhraní pro skenování `pcap_open_live()`. Aby nám knihovna pcap předávala jen pakety, které nás zajímají, aplikuje se na její výstup filtr. Protože nás zajímají jen TCP pakety, bude to pravidlo *tcp*.

4.1.2 Rozpoznávací prvky

Tabulka s překladem na jména portů je uložena v souboru, který je nutno načíst. Na každém řádku v souboru je uvedeno pojmenování portu následované číslem portu. Tabulka je následně uložena ve formě vektoru v paměti současně s druhým vektorem, ve kterém se počítá výskyt daného portu. Regulární výrazy na rozpoznávání obsahu prvku jsou uloženy v dalším adresáři v jednotlivých souborech. Tyto soubory mají velmi jednoduchý formát, který je ovšem nutné důsledně dodržovat. Řádky začínající znakem # jsou vyhodnoceny jako komentáře a jsou ignorovány, stejně tak jsou ignorovány řádky, které nic neobsahují. Z prvního platného řádku se pak udělá název regulárního výrazu. Následující platný řádek je označen jako vlastní regulární výraz. Všechny další řádky jsou při načítání ignorovány.

V paměti analyzátoru je pak vytvořen záznam, který obsahuje název, vlastní regulární výraz a zkompilovaný regulární výraz. Poslední jmenovaná položka je velmi důležitá především z důvodu rychlosti celého programu. Protože se na každý paket, který má být identifikován, musí vyzkoušet všechny regulární výrazy, je zbytečné v každém kroku kompilovat regulární výraz a tak už je předkompilovaný v paměti připraven na okamžitou aplikaci na data.

4.1.3 Statistika

Aby mohlo vypisování statistik o provozu probíhat nezávisle na zpracování dat, analyzátor vytvoří speciální vlákno pouze na výpočet a výpis statistik na obrazovku. Před spuštěním vlákna se vytvoří prostředí ncurses voláním funkce `initscr()` a následně se nastaví jeho parametry. Poté se zavolá funkce `run_print_stat()`, která spustí vlákno a začně vypisovat na obrazovku statistiku.

4.2 Zachytávání a zpracování

Pokud proběhly předešlé kroky bez chyby, může se začít skenovat. Knihovna pcap na toto používá funkci `pcap_loop()`. Této funkci se předá inicializované rozhraní a speciální funkce, která se zavolá, když přijde paket, který se má zpracovat. Tato speciální funkce se v mé aplikaci jmenuje `parse_packet()` a tu si nyní popíšeme.

Funkce `parse_packet()` je nejdůležitější funkce celého analyzátoru. Rozpoznává, třídí a organizuje příchozí pakety. Funkce obdrží paket v surové podobě tj. v podobě v jaké přišel na rozhraní. Musí tedy nejprve nalézt IP datagram. Ten začíná na určitém offsetu od počátku rámce, který se musí předem spočítat. Výpočet offsetu se provádí při nastavování síťového rozhraní před spuštěním zachytávání. Určí se typ síťového rozhraní a jeho mód. Podle toho se určí offset. Často se posouváme o konstantní číslo, jsou však případy, kdy se musí offset dopočítávat až při příchodu vlastního paketu. Potom se prochází hlavička rámce a vypočítává se offset za běhu. Nejčastěji se s tímto dopočítáváním setkáme při zachytávání paketů v režimu *monitor* u Wi-Fi sítí.

Po úspěšném nalezení začátku IP datagramu analyzátor testuje, zda-li se jedná o IPv4 či IPv6 datagram a podle toho počítá velikost hlavičky. V IP hlavičce se nachází pro nás velmi podstatné příznaky. Pro aplikaci jsou důležité především příznaky SYN (navázání spojení), FIN (ukončení spojení), RST (neplánované ukončení spojení bez čekání na potvrzení) a ACK (potvrzení). IP datagram, který obsahuje SYN příznak je pak brán jako první paket TCP streamu a je mu založen v paměti záznam. Zkopírují se do něj IP adresy a analyzátor se posune na offset TCP protokolu v paketu. Analyzátor načte TCP hlavičku a v ní vyhledá dvě velmi důležité položky pro identifikaci – zdrojový a cílový port. Oba se uloží následně do záznamu TCP streamu.

Každý další paket, pokud není s příznakem SYN, se pak podle IP adres a portů vyhledává v záznamech, čímž se zjišťuje, jestli nepatří do již zaznamenaného TCP streamu. Neexistuje-li v žádném záznamu je zahozen bez jakéhokoliv dalšího zpracování. V opačném případě je paket započítán do statistik v záznamu TCP streamu a analyzátor zjistí, zda kromě příznaků paket obsahuje i nějaká data. Pakety, které vytváří virtuální okruh, totiž nenesou data, nesou pouze příznaky. Až po úspěšném navázání spojení začnou proudit pakety s daty.

V tomto momentě se pak na data v paketu začnou aplikovat regulární výrazy, které mají za cíl zjistit, o jaký druh dat se jedná. Může se stát, že na data v paketu lze aplikovat více regulárních výrazů. Taková situace je řešena tak, že jako identifikátor podle obsahu

není jeden název protokolu, ale čárkami oddělené možnosti typu paketu. Může se stát, že nelze TCP stream nelze identifikovat podle prvního paketu obsahující data. Vnitřní interpretace tohoto stavu je pak jako `unknown` či prázdný řetězec. Při příchodu dalšího paketu se spustí identifikace znova. V záznamu o TCP streamu máme ještě uložená čísla portů. Ty vyhledáme v tabulce portů a inkrementujeme počet podle čísla portu.

Identifikovaný TCP stream se následně vypíše na obrazovku včetně statistik o došlých paketech jak ilustruje obrázek 4.1.

```
Analyzator sitovych protokolu
[@  ] Skenuji na zarizeni eth0
Pakety
      celkem vsech 1814
        za sekundu 2
  nahodnych paketu 314 (17.31 %)
  prijatych s daty 836 (46.09 %)
   pouze rezijnich 978 (53.91 %)
Verze IP
              IPv4 1814 (100.00 %)
              IPv6 0 (0.00 %)
TCP streamu
             pocet 36
Identifikovano podle portu: Top 8 z 2. Celkove identifikovano 36 podle portu.
        34 http (94.44 %)
         2 smtp (5.56 %)
Identifikovano podle obsahu: Top 8 z 2. Celkove identifikovano 36 podle portu.
        34 http (94.44 %)
         2 smtp (5.56 %)
Poslednich 6 identifikovanych streamu
[22:54:53.060300] { 192.168.0.102: 35455  --> 77.75.72.52: 80 http } http
[22:54:53.060338] { 192.168.0.102: 35456  --> 77.75.72.52: 80 http } http
[22:54:53.810493] { 192.168.0.102: 55049  --> 74.125.43.102: 80 http } http
[22:54:53.944790] { 192.168.0.102: 57776  --> 77.78.99.22: 80 http } http
[22:54:58.150113] { 192.168.0.102: 46890  --> 80.48.15.22: 80 http } http
[22:54:58.370066] { 192.168.0.102: 57168  --> 77.75.73.86: 80 http } http
[22:55:14.822476] { 213.226.208.2: 25 smtp --> 192.168.0.10: 49540  } smtp
[22:55:15.831379] { 213.226.208.2: 25 smtp --> 192.168.0.10: 49541  } smtp
```

Obrázek 4.1: Ukázka běhu analyzátoru

Jako poslední druh důležitých paketů jsou pakety s příznaky FIN či RST oznamující ukončení spojení. Zpracování RST je velmi prosté, protože tento příznak znamená, že se již nečeká na žádné potvrzení a virtuální okruh bude okamžitě přerušen. Po přijetí tohoto paketu tedy analyzátor okamžitě zruší záznam v paměti o TCP streamu.

Příznak FIN se zpracovává odlišně. V klasickém třícestném navazování či rušení virtuálního okruhu si musí obě strany navzájem potvrdit konec spojení. Všechny tyto stavy jsou zaznamenávány. V momentě, kdy obě strany potvrdily konec spojení, analyzátor ruší záznam v paměti.

Při povoleném logování analyzátor veškeré informace zapisuje do předem zvoleného souboru. Na začátku logování jsou zapsány souhrnné informace o zařízení, času a parametrech analyzátoru, se kterými byl spuštěn. Každý další řádek pak reprezentuje jeden navázaný, identifikovaný TCP stream.

4.3 Ukončení zpracování

Zpracování (zachytávání) je ukončeno ve třech případech:

- Analyzátor čte záznam ze souboru až dojde na konec.
- Analyzátor zachytává na fyzickém síťovém rozhraní. Zde zachytávání končí interakcí uživatele, které je reprezentováno zachycením ukončovacích systémových signálů SIGINT a SIGTERM.
- Nastane chyba čtení, kdy už nelze dále pokračovat.

Ve všech případech analyzátor zastaví svoji činnost, zapíše při povoleném logování statistiky na konec záznamu o činnosti, odstraní všechny záznamy zachycených TCP streamů z paměti a ukončí činnost.

Kapitola 5

Testování

Aby se ověřila funkčnost analyzátoru, je zapotřebí jej otestovat. Testování při vývoji probíhalo na notebooku s nainstalovaným operačním systémem GNU/Linux.

Každé testování probíhalo následujícím způsobem: proběhla kompilace programu, byly nastaveny parametry a síťové rozhraní, na kterém byly zachytávány pakety. Nejlépe se testuje na síťovém rozhraní, které je připojeno k internetu. Příkladem může být e-mailový klient. Ten si uživatel nechá spuštěn po celý čas, co sedí u počítače. E-mailový klient naváže spojení a v periodických intervalech komunikuje se serverem při kontrole pošty.

Tato data jsou však velmi náhodná. Hlavně na začátku vývoje je potřeba stejný vzorek dat, který je předem znám. Lze pak snadno určit, zda analyzátor funguje korektně.

Vzorek dat se nejsnáze získá přes aplikace *Wireshark* či *tcpdump*. Výsledný záznam se ukládá do univerzálního souboru *tcpdump capture file* s příponou `.cap`. Program Wireshark umí názorně zobrazit zachycená data, takže lze snadno ověřit, jestli analyzátor zachycuje správně.

Testování bude probíhat pomocí syntetických testů. Jsou jednoduše opakovatelné testy, dokazující funkčnost celého analyzátoru.

Protože některé testy mohou mít různé výsledky v závislosti na počítači, na kterém běžely, musel být zvolen jeden referenční počítač, na kterém proběhnou všechny testy.

Referenční počítač má tuto konfiguraci hardwaru:

- Procesor: Intel Core 2 Duo T5500 na frekvenci 1,66 GHz
- Paměť: 2 GiB
- Síťové rozhraní: Realtek 8169SC Gigabit Ethernet, Intel 4965ABGN Wireless

Neméně důležitý je použitý software, protože i ten může výsledky citelně ovlivnit. Na referenčním počítači je tedy nainstalován tento software:

- linux ve verzi 2.6.31
- tcpreplay ve verzi 3.4.1
- tcpdump ve verzi 4.0.0
- libpcap ve verzi 1.0.0

5.1 Test 1

První test se zabývá porováváním identifikace podle portů a podle obsahu. Jako příklad pro toto porovnání byl vybrán protokol *HTTP* (Hyper Text Transfer Protocol) a protokol *FTP* (File Transfer Protocol). HTTP server nejčastěji běží na portu číslo 80 a právě na něm se spustí FTP server, který jinak pracuje na portu 21. Záměrně byl vybrán FTP, protože má jednoduchý regulární výraz:

```
^220[ -~]*ftp
```

a hlavně je spolehlivý. Tím se rozumí, že TCP stream se zaručeně identifikuje jako FTP. Vzhledem k faktu, že po navázání spojení se okamžitě FTP server ohlásí, nebylo nutné použít plnohodnotný FTP server, ale pouze miniaplikaci. Ta se po navázání spojení ohlásí jako:

```
220 fake-ftp-server-on-port-80 ftp server
```

a ukončí spojení. To nám pro identifikaci stačí.

První testovací skript tedy spustí náš analyzátor, FTP server na portu číslo 80 a pomocí aplikace *telnet* naváže spojení. Analyzátor zachytí spojení, identifikuje jej a zapíše do logu. Skript pak analyzátor ukončí.

Podíváme-li se do logu, zjistíme, že analyzátor idenfikoval TCP stream na portu číslo 80 (HTTP), ale po prozkoumání obsahu určil, že se jedná o FTP.

5.2 Test 2

Další test se zabývá čtením záznamu provozu na síťovém rozhraní. Analyzátoru se pomocí parametrů předá záznam a on ho začne zpracovávat. Poté, co přečte a zpracuje celý soubor se analyzátor sám ukončí. Tento test demonstruje použití analýzy externě získaných dat. Příkladem může být zachytávání pomocí programu `tcpdump` do souboru. Výsledek práce je uložen do logu a uživatel se může podívat, co obsahoval záznam.

5.3 Test 3

Tento test ověřuje funkčnost zachytávání na reálném síťovém rozhraní. Analyzátor je spuštěn tak, aby zachytával na síťovém zařízení `lo`, což je rozhraní místní smyčky (localhost). Důvodem použití místní smyčky je, že na tomto rozhraní běžně není téměř žádný provoz. Analyzátor tedy bude snímat jen data, která mu přes toto rozhraní pošleme. To se dá velmi snadno zařídit pomocí programu `tcpreplay`. Tento program umí zasílat data na zvolené síťové rozhraní. Data čte ze souboru se záznamem provozu a lze například zvolit rychlost s jakou má data posílat. Záznam vnikal náhodným procházením webových stránek.

Měření probíhá pětkrát na referenčním počítači, aby se vyloučily odchylky. Tabulka 5.1 zobrazuje všechny iterace a naměřené výsledky. V prvním sloupci je číslo měření, druhý obsahuje maximální naměřený počet paketů za sekundu, kdy analyzátor nevynechal při analýze ani jeden paket. Další sloupec obsahuje přepočet na Mbps (megabity za sekundu) a v posledním sloupci je průměrné vytížení procesoru analyzátorem uvedené v procentech

Spočtením aritmetického průměru z naměřených výsledků dojdeme k následujícímu: Analyzátor stíhá vyhodnotit **14187,4** paketů za sekundu, což odpovídá zhruba **34,8 Mbps** při vytížení **33,14 %**.

měření	pakety	Mbps	% vytížení
1	16488	40	33,2 %
2	14705	36	33,8 %
3	13951	34	33,8 %
4	10113	25	30,5 %
5	15680	39	34,4 %

Tabulka 5.1: Hledání maximální rychlosti, při které analyzátor stíhá analyzovat všechny pakety

5.4 Test 4

Z velké části vychází z Testu 3. Rozdíl je v tom, že není dán fixní počet paketu za sekundu, co jsou poslány na síťové rozhraní, ale program `tcpreplay` vysílá na síťové rozhraní maximum, které zvládne. Na testovacím počítači rychlost kolísá kolem 250 Mbps, což už je poměrně vysoká rychlost. Test odhalil, že analyzátor tak obrovskou rychlost nezvládá, protože režie zpracování jednoho paketu trvá déle, než časová prodleva mezi příchodem jednotlivých paketů. To zapříčinilo, že analyzátor začne vynechávat pakety. Pokud bude část paketů chybět, analyzátor nemusí spolehlivě určit všechny TCP streamy. Téměř nemožné je identifikovat obsah TCP streamu s chybějícím začátkem, protože unikátní signatury jsou nejčastěji právě na začátku dat ve streamu. Nejhorší případ nastane, když se ztratí začátek TCP streamu. Analyzátor pak vůbec nepozná, že nějaký TCP stream existuje

5.5 Test 5

Jak může vypadat provoz na síťovém rozhraní, když uživatel pouze surfuje? Přeloženo do řeči analyzátoru: mnoho krátkých spojení. To je předmětem Testu 5. Pomocí aplikace `wget` se nechá rekurzivně z webu FIT VUT BRNO [1] stahovat obsah, čímž se částečně simuluje surfování po webu. Ze zjištěných statistik lze zjistit, že se průměrně naváže 190 spojení, které jsou složené z zhruba 9500 paketů. Zajímavé je rovněž, že poměr dat vůči režii spojení TCP je zhruba 50 %.

5.6 Test 6

Opakem mnoha spojení je jedno souvislé spojení. Nejčastěji se s ním setkáme při stahování archivů, hudby nebo videa. Opět byl využit program `wget` a stahoval se instalátor prohlížeče Opera [2]. Analyzátor zachytil následující: navázalo se jedno spojení a přeneslo se hruba 15000 paketů. Z udávaných 15000 bylo téměř 5500 paketů režijních. To znamená, že téměř 37 % provozu je režie. Vzniká především posílání paketů s ACK, které potvrzují doručení dat.

5.7 Test 7

Poslední test se zabývá mírou úspěšnosti identifikace protokolů. Bylo testováno, jak jsou metody schopné rozpoznat streamy a určit přesný typ protokolu. Jako vzorové protokoly

[1] *http://www.fit.vutbr.cz/*
[2] *http://www.opera.com/download/*

protokol	podle portu	podle obsahu	výsledek
ftp	ftp	ftp	rozpoznán
bittorrent	-	bittorrent	rozpoznán podle obsahu
http	http	http	rozpoznán
http-ssl	https	ssl	rozpoznán, šifrovaný
imap	imap	imap	rozpoznán
imap-ssl	imaps	ssl	rozpoznán, šifrovaný
smb	microsoft-ds	microsoft-ds	rozpoznán
smtp	smtp	smtp	rozpoznán
svn	-	-	nerozpoznán
telnet	telnet	telnet	rozpoznán
ventrilo	-	-	nerozpoznán
pop3	pop3	pop3	rozpoznán
smpp	-	-	nerozpoznán
ssh-ssl	ssh	ssh	rozpoznán
tor (nové spojení)	https	ssl	rozpoznán jako https
tor (data)	etlservicemgr	ssl	rozpoznán podle portu
x11	x11	x11	rozpoznán

Tabulka 5.2: Úspešnost identifikace protokolů

byly vybrány: ftp, http, imap, smb, smtp, svn, telnet, bittorent, ventrilo, pop3, smpp, ssh, tor a x11.

Výsledky testu pro tyto protokoly shrnuje tabulka 5.2. Většinu protokolů je analyzátor schopen rozpoznat jak podle čísel portů, tak i obsahu a dokáže tak uživateli říci, o který protokol se jedná. Očekávané problémy nastávají, pokud je protokol šifrovaný. Zde napoví už jen číslo portu. Šifrovaný obsah podle *TLS* (Transport Layer Security) [3] je nazýván jako *SSL* (Secure Sockets Layer), což je původní označení (prvních verzí).

Naopak dopadl protokol BitTorrent, zástupce P2P (Peer To Peer) sítí, který podle čísel portů nebyl identifikován, ale podle obsahu byl rozeznám. Spojení tohoto protokolu má náhodné čísla portů na obou počítačích, mezi kterými probíhá komunikace.

Kapitola 6

Závěr

Cílem této práce bylo analyzovat datový tok na síťovém rozhraní. Na toto byl naprogramován jednoduchý, přesto poměrně účinný analyzátor. Základem bylo seznámit se s možnostmi knihovny pcap, která tvoří základ analyzátoru. Ta zachytí a předá analyzátoru paket na zpracování.

IP verze 6 sice ještě není majoritní protokol sítě internet, ale bylo rozhodně výzvou do budoucna jej implementovat do analyzátoru, aby byl připraven jej zpracovávat. Díky tomu vznikly též zajímavé výsledky v poměru toků IP verze 4 a IP verze 6. V drtivé většině času převládá IP veze 4, ale najde se už spousta webů, které podporují IP verze 6.

Vhodným rozšířením analyzátoru byla podpora kromě Ethernetového rámce i pro rámec IEEE 802.11. To se projevilo možností analyzovat provoz na bezdrátovém síťovém rozhraní v řežimu monitor – tedy analyzovat provoz ve vzduchu, aniž by byl počítač kamkoliv připojen.

Aby se nemusely testovat všechny pakety, bylo velmi přínosným krokem shlukovat je do streamů. Většina regulárních výrazů testuje začátek dat, která se nachází v prvním datovém paketu. Když se idenfitikace nezdařila na poprvé (první paket), s následujícím paketem se pravděpodobnost úspěšné identifikace výrazně zvýšila. Nabízí se otázka, bude analyzátor spustěný až po navázaní spojení pracovat, když nemá k dispozici pakety se začátkem TCP streamu? Protože je tento analyzátor připraven běžet dlouhodobě, bylo rozhodnuto striktně ignorovat pakety nepatřící do známých TCP streamů a analyzátor bude čekat až na vytvoření nových TCP streamů.

Během testování bylo třeba vynechat některé regulární výrazy z originální sady L7 filtrů, protože byly příliš obecné. Téměř každý TCP stream se jevil jako identický. Jednalo se především o *skypeout*.

Pomocí nástroje *tcpreplay* jsem při testování rychlosti a s tím související ztrátovosti paketů došel k závěru, že na referenčním počítači, kde se analyzátor vyvíjel, nelze zpracovávat více než 40 Mb dat za sekundu. Analyzátoru nejdéle trvá vyhodnocení regulárních výrazů na obsah paketu. Spouštět paralelně regulární výrazy na obsah paketu a tak snížit čas potřebný na vyhodnocení se zdá jako možné rozšíření analyzátoru do budoucna.

Dalším rozšířením by mohla být podpora protokolu UDP. V této verzi analyzátoru není tento protokol podporován, protože jsem došel k závěru, že se používá především na multimediální protokoly (VoIP či streaming videa). Tyto protokoly se však hůře identifikují, vyžadují pokročilejší techniky. Více zajímavých protokolů se však staví nad protokolem TCP a byl tedy zvolen jen ten.

Literatura

[1] Combs, G.: Wireshark, Network Protocol Analyzer [online]. http://www.wireshark.org/.

[2] Deering, S.; Hinden, R.: Internet Protocol Version 6 (IPv6) Specification [online]. http://tools.ietf.org/html/rfc2460.

[3] Dierks, T.; Rescorla, E.: The Transport Layer Security (TLS) Protocol Version 1.2 [online]. http://tools.ietf.org/html/rfc5246.

[4] Hýsek, J.: IEEE 802.11 (Wireless LAN) [online]. http://trace.dump.cz/papers/802.11.pdf.

[5] IANA: Port Numbers [online]. http://www.iana.org/assignments/port-numbers.

[6] IEEE: ANSI/IEEE Std 802.11, 1999 Edition [online]. http://pdos.csail.mit.edu/decouto/papers/802.11.pdf.

[7] Information Sciences Institute, U. o. S. C.: Internet Protocol [online]. http://tools.ietf.org/html/rfc791.

[8] Information Sciences Institute, U. o. S. C.: Transmission Control Protocol [online]. http://tools.ietf.org/html/rfc793.

[9] Levandoski, J.; Sommer, E.; Strait, M.; aj.: L7 Filter - Application Layer Packet Classifier for Linux [online]. http://l7-filter.sourceforge.net/.

[10] Mockapetris, P.: DOMAIN NAMES - CONCEPTS and FACILITIES [online]. http://tools.ietf.org/html/rfc882.

[11] Peterka, J.: Ethernet II vs. IEEE 802.3 [online]. http://www.earchiv.cz/anovinky/ai2058.php3.

[12] Postel, J.: User Datagram Protocol [online]. http://tools.ietf.org/html/rfc768.

[13] Rekhter, Y.; Li, T.: An Architecture for IP Address Allocation with CIDR [online]. http://tools.ietf.org/html/rfc1518.

[14] Reynolds, J.; Postel, J.: Assigned Numbers [online]. http://tools.ietf.org/html/rfc1700.

[15] Wikipedia: Cyclic redundancy check [online]. http://en.wikipedia.org/wiki/Cyclic_redundancy_check.

[16] Wikipedia: Internet Protocol [online].
http://cs.wikipedia.org/wiki/Internet_Protocol.

[17] Wikipedia: Quality of service [online].
http://en.wikipedia.org/wiki/Quality_of_service.

[18] Wikipedia: TCP/IP [online]. http://cs.wikipedia.org/wiki/TCP/IP.

[19] Wikipedia: Transportní vrstva [online].
http://cs.wikipedia.org/wiki/Transportní_vrstva.

Dodatek A

Obsah CD

Přiložené CD obsahuje:

- adresář **src** se zdrojovými soubory analyzátoru
- adresář **protocols_pattern**, který obsahuje sadu regulárních výrazů
- soubor **protocols_ports**, který obsahuje pojmenování portů
- adresář **fake-ftp-server** s miniaturním ftp serverem pro testovací účely
- adresář **tests** s testovacími skripty a vzorky dat
- adresář **doc** obsahující dokumentaci zdrojového kódu
- soubor **README** s obecnými radami a postupy rozjetí analyzátoru
- soubory **projekt.pdf** (elektronická verze této práce) a **bp-xsobot14.tar.gz** obsahující zdrojové texty této práce (LaTeX)

Dodatek B

Manual

Aplikaci `scanner` lze spustit těmito parametry:

- `-h`
  Pouze vypíše nápovědu a skončí.
- `-v`
  Vypisuje podrobnější informace o činnosti analyzátoru.
- `-e`
  Nastaví cestu k souboru s pojmenováním portů. Výchozí hodnota je *protocols_ports*.
- `-d`
  Nastaví cestu k adresáři se sadou regulárních výrazů. Výchozí hodnota je *protocols_pattern*.
- `-i` nebo `-f`
  Analyzátor bude číst z síťového rozhraní *(-i)* nebo ze záznamu síťového provozu *(-f)*. Pokud není zadán ani jeden z těchto přepínačů, analyzátor jako výchozí zařízení bere *eth0*.

Příklady spuštění analyzátoru:

- `scanner`
  Analyzátor spustil sledování v reálném čase na síťovém rozhraní *eth0*.
- `scanner -i wlan0`
  Analyzátor spustil sledování v reálném čase na síťovém rozhraní *wlan0*.
- `scanner -i wlan0 -o zaznam.log`
  Analyzátor spustil sledování v reálném čase na síťovém rozhraní *wlan0* a bude zapisovat průběh zachytávání do souboru *zaznam.log*.
- `scanner -f zaznam.cap -o zaznam.log`
  Analyzátor provede identifikaci vzorku dat uloženém v souboru *zaznam.cap* a zapíše výsledek do souboru souboru *zaznam.log*.

Dodatek C

Seznam použitých zkratek

Zkratka	Vysvětlivka
AP	Access Point
API	Aplication Program Interface
ARP	Address Resolution Protocol
CIDR	Classless Inter-Domain Routing
CRC	Cyclic Redundancy Check
CSV	Comma-Separated Values
DNS	Domain Name Server
FCS	Frame Check Sequence
FTP	File Transfer Protocol
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
IANA	Internet Assigned Numbers Authority
ICMP	Internet Control Message Protocol
IEEE	Institute of Electrical and Electronics Engineers
IGMP	Internet Group Management Protocol
IGRP	Interior Gateway Routing Protocol
IP	Internet Protocol
IP Sec	IP Security
IPng	IP new generation
L7	Level 7 (Application Layer of ISO/OSI model)
MAC	Media Access Control
NAV	Network Allocation Vector
P2P	Peer To Peer
PCRE	Perl Compatible Regular Expressions
PEAP	Protected Extensible Authentication Protocol
POSIX	Portable Operating System Interface
QoS	Quality of Service
RARP	Reverse Address Resolution Protocol
RFC	Request For Comments
SIGINT	Signal Interrupt
SIGTERM	Signal Terminate

Zkratka	Vysvětlivka
SSL	Secure Socket Layer
TCP	Transmission Control Protocol
TLS	Transport Layer Security
TTL	Time To Live
UDP	User Data Protocol
VoIP	Voice over Internet Protocol
Wi-Fi	Wireless Fidelity
XML	Extensible Markup Language