Univerzita Pardubice

Fakulta elektrotechniky a informatiky

Zachytávání a analýza síťových paketů v jazyce java

Luboš Čábelka

Bakalářská práce

2013

Univerzita Pardubice
Fakulta elektrotechniky a informatiky
Akademický rok: **2012/2013**

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Luboš Čábelka**
Osobní číslo:	**I09080**
Studijní program:	**B2646 Informační technologie**
Studijní obor:	**Informační technologie**
Název tématu:	**Zachytávání a analýza síťových paketů v jazyce java**
Zadávající katedra:	**Katedra informačních technologií**

Zásady pro vypracování:

Cílem práce je návrh a implementace zachytávání a analýza síťových paketů v jazyce Java.
V teoretické části bude popis práce se sítí v jazyce Java, síťových modelů, paketů, datagramů, IPv4 adresy. Dále teoretická část bude obsahovat návrh zpracování síťové komunikace, její filtrace a případné dekodování přenášených dat.
Realizační část bude Java GUI aplikace, která bude zachytávat pakety. Pakety se bude označovat časem přijetí, ukládat do souboru a opětně zobrazovat. Aplikace musí být navržena tak, aby bylo možné snadno doplňovat další filtry a dekodéry datového obsahu.

Rozsah grafických prací:

Rozsah pracovní zprávy:

Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam odborné literatury:

[1] Pecinovský,R., Návrhové vzory, Computer Press, 2007 vydání první, ISBN 978-80-251-1582-4
[2] Page-Jones, Meilir , Základy objektově orientovaného návrhu v UML, Grada 2001, ISBN 80-247-0210-X

Vedoucí bakalářské práce: **Ing. Karel Šimerda**
Katedra softwarových technologií

Datum zadání bakalářské práce: **21. prosince 2012**
Termín odevzdání bakalářské práce: **10. května 2013**

Univerzita Pardubice Česká republika

L.S.

prof. Ing. Simeon Karamazov, Dr.
děkan

Ing. Lukáš Čegan, Ph.D.
vedoucí katedry

V Pardubicích dne 29. března 2013

Prohlašuji:

Tuto práci jsem vypracoval samostatně. Veškeré literární prameny a informace, které jsem v práci využil, jsou uvedeny v seznamu použité literatury.

Byl jsem seznámen s tím, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorský zákon, zejména se skutečností, že Univerzita Pardubice má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona, a s tím, že pokud dojde k užití této práce mnou nebo bude poskytnuta licence o užití jinému subjektu, je Univerzita Pardubice oprávněna ode mne požadovat přiměřený příspěvek na úhradu nákladů, které na vytvoření díla vynaložila, a to podle okolností až do jejich skutečné výše.

Souhlasím s prezenčním zpřístupněním své práce v Univerzitní knihovně.

V Kojicích dne
30. 7. 2013

Luboš Čábelka

Poděkování

Na tomto místě bych chtěl poděkovat Ing. Karlovi Šimerdovi za vedení mé bakalářské práce a Ing. Zděnkovi Šilarovi za vedení své semestrální práce, kterou jsem použil jako základ bakalářské práce.

ANOTACE

Práce je věnována zachytávání a analýze síťových paketů v programovacím jazyce java. Pojednává o síťových modelech, protokolech využívaných v dnešních sítích.

V praktické části jsou využity představené technologie pro zachytávání a následnou analýzu.

KLÍČOVÁ SLOVA

java, síť, TCP/IP, zachytávání paketů

TITLE

Capturing and analyzing network packets in java

ANNOTATION

The work deals with capturing and analysing network packets in programming language java. Deals with network models, protocols used in todays networks.

Described technologies are used in practical part for capturing and analyzing.

KEYWORDS

java, network, TCP/IP, capturing packets

Obsah

Seznam obrázků

Seznam tabulek

Seznam zkratek

ISO/OSI	International Organisation for Standardization/ Open System Interconection
FTP	File Transfer Protocol
TFTP	Trivial File Transfer Protocol
DHCP	Dynamic Host Control Protocol
DNS	Domain Name System
POP3	Post Office Protocol
SMTP	Simle Mail Transfer Protocol
IMAP	Internet Message Access Protocol
TCP	Transmition Control Protocol
UDP	User Datagram Protocol
PDU	Protocol Data Unit
MTU	Maximum Transmission Unit
SFD	Start Frame Delimiter
QoS	Quality of Service
IP	Internet Protocol
LLC	Logical Link Control
MAC	Media Access Control
Wi-Fi	Wireles Fidelity
TCP/IP	Transmition Control Protocol/Internet Protocol
RFC	Request For Comments
NFS	Network File System
HTTP	Hypertext Transport Protocol
ARP	Address Resolution Protocol
RARP	Reverse Address Resolution Protocol
ICMP	Internet Control Message Protocol
IGMP	Internet Group Management Protocol
OSPF	Open Shortest Path First
IGRP	Interior Gateway Routing Protocol
EGRP	Enhanced Interior Gateway Routing Protocol
FDDI	Fiber Distributed Data Interface
ATM	Asynchronous Transfer Mode

ACK	Acknowledgement
PING	Packet Internet Groper
TTL	Time To Live
IEEE	Institute of Electrical and Electronics Engineers
BGP	Border Gateway Protocol
RIP	Routing Information Protocol
CAT5e	Category 5e
JPCAP	Java Packet CAPtor
IPv4	Internet Protocol verze 4
IPv6	Internet Protocol verze 6

0. Úvod

Jako svou bakalářskou práci jsem zvolil zachytávání a analýzu síťových paketů v jazyce Java.

Zachytávání a analýza síťových paketů je zajímavá činnost, která může být provozována buď s pomocí již vytvořených aplikací (například Wireshark – kompletní program pro zachytávání, analýzu a tvorbu statistik, obsahuje spoustu nástrojů, naprogramován v jazyce C), nebo naprogramováním vlastní aplikace, která sice nebude tak propracovaná a široce použitelná, ale bude přesně na míru a při její tvorbě člověk pochopí principy síťové komunikace mnohem důkladněji než prostým užíváním již hotové aplikace.

Práce je rozdělena na dvě hlavní části. Teoretická část popisuje síťové modely a síťové protokoly, se kterými se setká každý při zachytávání a analýze paketů, dále nejpoužívanější zařízení v počítačových sítích. Na závěr je kapitola, která popisuje možnosti přístupu k síti v jazyce Java, především externí knihovnu JPCAP, se kterou jsem se seznámil v roce 2012 při vytváření semestrální práce pro předmět Programovací techniky v jazyce Java. Praktická část popisuje postup vývoje aplikace pro zachytávání a analýzu paketů, od návrhu, přes vývoj po dokončenou aplikaci, její vstupy, výstupy, filtry paketů a ovládání. V příloze je vzorový paket analyzovaný pomocí této aplikace.

Práce je určena pro všechny kdo chtějí pracovat s pakety v jazyce Java, seznámit se se síťovými protokoly a vytvořit si vlastní aplikaci pro zachytávání paketů.

0.1 Požadavky na software

0.1.1 Funkční požadavky

Cílem praktické části bylo vytvořit aplikaci na zachytávání a analýzu síťové komunikace.

Funkční požadavky dle zadání byly následující:

1. Zachytávání paketů na síťovém rozhraní
2. Paket bude označen časem přijetí
3. Ukládání a načítání paketů ze souboru
4. Filtrování přijatých paketů
5. Analýza obsahu paketů

6. Do aplikace musí být snadno doplnitelné další filtry a dekodéry obsahu

0.1.2 Nefunkční požadavky

Aplikace byla vyvíjena a je primárně určena pro 32bitový operační systém Windows XP s rozšířením knihovnami Jpcap a WinPcap. Grafické uživatelské rozhraní je určeno pro rozlišení 1024x600 a vyšší.

Jazykové konvence

Pro zvýšení čitelnosti textu byla nastavena následující jednotná konvence zápisu textu.

důležitý text	podstatné texty, důležité názvy, zkratky
zdrojový kód	zdrojové kódy aplikace
`název třídy`	názvy tříd, metod, proměnných atd.

Teoretická a rešeršní část

1. Referenční model ISO/OSI

Protože komunikace dvou zařízení byla složitá, bylo rozhodnuto ji rozdělit na několik dílčích úloh (vrstev). Model **ISO/OSI** byl přijat jako mezinárodní standard v roce 1984 (norma **IS 7498**). Jedná se o sedmivrstvý model, kde každá vrstva využívá služeb vrstvy nižší a každá vrstva poskytuje služby vyšší vrstvě. Komunikovat spolu mohou pouze dvě stejné vrstvy, ovšem zprostředkovaně přes nižší vrstvy.

1.1 Vrstvy modelu ISO/OSI

1.1.1 Aplikační vrstva

Poskytuje aplikacím přístup ke komunikaci a umožňuje jejich vzájemnou spolupráci. Služby poskytované aplikační vrstvou: identifikace komunikujících, přenos zpráv, ochrana zpráv, synchronizace komunikujících aplikací (klient – server), navázání a ukončení spojení. V této vrstvě mohou komunikovat zařízení, programy ale i lidé. Protokoly: **FTP**, **TFTP**, **telnet**, **DHCP**, **DNS**, **POP3**, **SMTP**, **IMAP.**

1.1.2 Prezentační vrstva

Zajišťuje přenos zpráv mezi uživateli, upravuje data do podoby vhodné k odesílání, aby zprávy pro aplikaci byly prezentovány stejným způsobem, zabývá se pouze podobou dat ne jejich obsahem. Služby poskytované prezentační vrstvou: žádost o vytvoření, zrušení relace, přenos dat, dohoda o syntaxi, formátování, komprese, dekomprese, **V OSI modelu jediná vrstva, která může modifikovat přenášená data.**

1.1.3 Relační vrstva

Vytváří, organizuje a ukončuje spojení mezi dvěma stanicemi, řídí přenos dat. Relační vrstva poskytuje služby: vytváření spojení (relace), přenos zpráv (normální, spěšný, pozdržený). Informuje prezentační vrstvu o výjimečných stavech (neopravitelná porucha

spojení). V jednom relačním spojení může vzniknout a zaniknout několik transportních spojení.

1.1.4 Transportní vrstva

Poskytuje transparentní, spolehlivý a cenově dostupný přenos dat. Vyrovnává rozdílné vlastnosti sítí. Všechny protokoly v transportní vrstvě jsou koncové, nenacházejí se po cestě v síti. Protokoly: **TCP** – spolehlivý přenos dat, navazuje a udržuje spojení, znovu posílá nedoručené pakety. **UDP** – nespolehlivý přenos dat, pouze přenáší bloky dat. Transportní vrstva určuje velikost bloků dat – **PDU** (protokolová datová jednotka určená k přenosu po síti).

1.1.5 Síťová vrstva

Pokud spolu dva uzly, které chtějí komunikovat, nesousedí, je třeba nalézt mezi nimi cestu v síti. Na základě síťové adresy je na síťové vrstvě hledána cesta přes síť (směrování), po které jsou přenášeny pakety (datagramy). Služby poskytované síťovou vrstvou: Adresace, směrování, vytváření a rušení síťových spojení, oznamování chyb, řízení datového toku (například pomocí **QoS**). Na této vrstvě se používá **IP** adresa, datová jednotka paket, zařízení směrovače, protokol **IP**.

1.1.6 Linková vrstva

Poskytuje spojení mezi dvěma sousedními uzly (v jedné části sítě). Mezi funkce linkové vrstvy patří především: zahájení, udržování a ukončení přenosu, formátování rámců, identifikace koncových bodů, detekce a oprava chyb. Na této vrstvě se používá adresa **MAC** (jedinečná fyzická adresa stanice nebo zařízení), datová jednotka rámec, zařízení: most a switch (přepínač).

Linková vrstva se rozděluje na **LLC** a **MAC** podvrstvu.

- **LLC** (Logical Link Control) – poskytuje rozhraní mezi vyššími vrstvami a přenosovým prostředkem.
- **MAC** (media acces Control) – poskytuje konkrétní funkce dané přenosovým prostředkem.

1.1.7 Fyzická vrstva

Na této vrstvě jsou data pouze tok bitů, definuje elektrické/optické signály přenosu. Fyzické spojení může být realizováno po kabelu (měděné nebo optické kabely) nebo bezdrátově (**Wi-Fi**, radiové vlny, mikrovlny, infračervené světlo, satelit).

2. Model TCP/IP

Označuje celou síťovou architekturu tvořenou více protokoly. Protokoly **TCP** a **IP** jsou základní, proto je po nich pojmenován celý model. Od počátku tvořena jako otevřená architektura, aktualizace vycházejí v **RFC** (Request for comments − žádost o komentáře). Model obsahuje 4 vrstvy. Pro uživatele se síť jeví jako jedna virtuální síť, k níž je přímo připojen každý počítač. Ve skutečnosti je však v síti mnoho propojovacích prvků rozdělených do mnoha sítí.

Dnes se používá IP verze 4, ale je snaha přejít na verzi 6, zařízení již **IPv6** podporují, ale celé sítě obvykle stále používají **IPv4**, přechod je pomalý.

2.1 Vrstvy modelu TCP/IP

2.1.1 Aplikační vrstva

Poskytuje aplikacím přístup ke komunikaci po síti. Protokoly aplikační vrstvy lze rozdělit na uživatelské a administrativní. Mezi uživatelské aplikace patří: **FTP**, **TFTP** – protokoly pro přenos souborů, **NFS** – distribuované sdílení souborů, **telnet** – virtuální terminál, **SMTP**, **IMAP**, **POP3** – emailové protokoly, **IRC** − síťová diskuse. Mezi administrativní aplikace patří: **DHCP**, **BOOTP** − automatická konfigurace, **DNS** − překlad adres a jmen, **NTP**, **SNTP** − protokoly zjišťování času.

2.1.2 Transportní vrstva

Stará se o koncový přenos dat, určování správného pořadí datagramů, opakované posílání nedoručených. Protokoly transportní vrstvy jsou **TCP** a **UDP**.

Port

Slouží k identifikaci protokolu aplikační vrstvy. Servery očekávají komunikaci na známých portech (80 − **HTTP**), klienti přidělují komunikaci libovolné číslo portu mimo porty do 1024.

Rozdělení portů

- **Známé porty** – 0 - 1023 − známá čísla protokolů, přiděluje je **IANA**

- **Registrované porty** – 1024 - 49 151 – porty pro uživatelské aplikace a procesy, registruje je IANA
- **Soukromé a dynamické porty** – 49 152 - 65 535 – náhodné porty pro klientské procesy a aplikace, nejsou nikde registrovány

Socket

Kombinace **IP** adresa a port. Jednoznačně identifikuje proces v rámci internetu. Například 77.75.72.3:80.

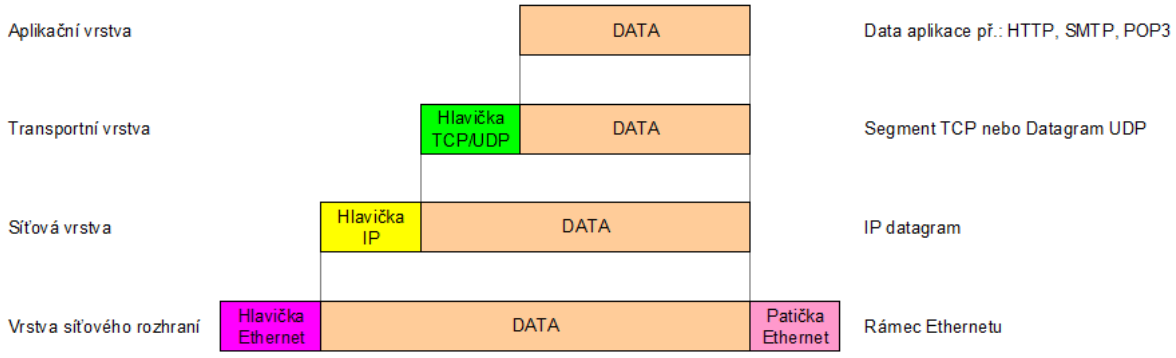
2.1.3 Síťová vrstva (Internet Layer)

Odpovídá síťové vrstvě modelu **OSI**. Funkce síťové vrstvy: adresace, směrování, segmentace, znovu sestavování datagramů, funkce pro skupinové vysílání. Protokoly síťové vrstvy: **IP** verze 4 a 6**, ARP** a **RARP** – protokoly mapování adres, **ICMP** – protokol řídících zpráv, **IGMP** – protokol členství ve skupině, **OSPF**, **IGRP**, **EIGRP** – směrovací protokoly.

2.1.4 Vrstva síťového rozhraní

Umožňuje přístup k fyzickému přenosovému médiu. Je specifická pro různé typy přenosových sítí (Ethernet, Token Ring, **FDDI**, **X.25**, **ATM)**.

2.2 Zapouzdřování



Obrázek 1 Zapouzdření komunikace v TCP/IP[1]

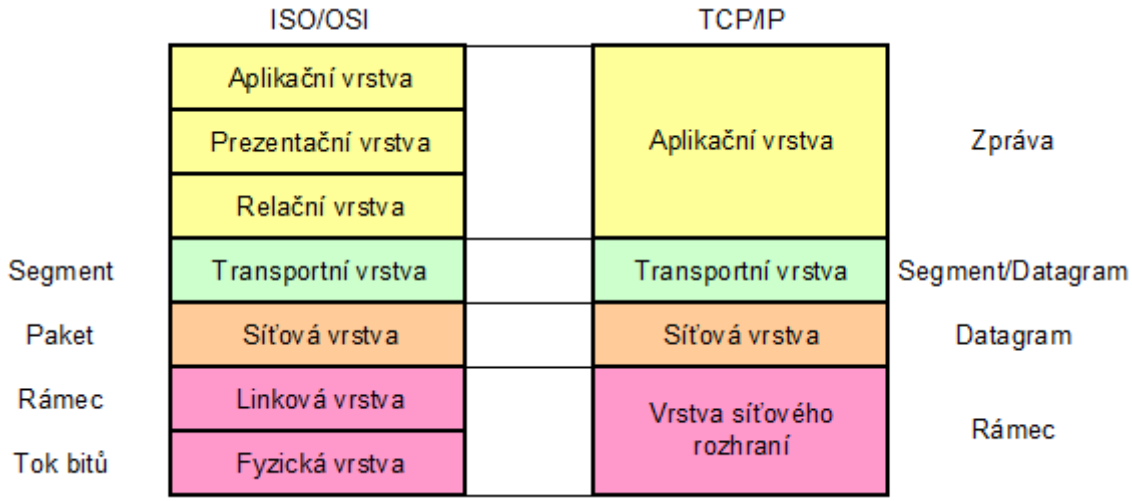
[1] Rita Pužmanová, TCP/IP v kostce, Kopp 2004, ISBN 80-7232-236-2

Data aplikační vrstvy jsou předána transportní vrstvě, jsou opatřena hlavičkou **TCP** nebo **UDP** protokolu a předána na síťovou vrstvu, na síťové vrstvě je přidána **IP** hlavička a na vrstvě fyzického rozhraní hlavička a patička Ethernetu.

2.3 Porovnání vrstev modelů ISO/OSI a TCP/IP

Model **ISO/OSI** má 7 vrstev, model **TCP/IP** má pouze 4 vrstvy. Vrstvy nejsou přesně stejné, ale z hlediska funkcí a služeb jejich účel celkem odpovídá.

- Aplikační vrstva modelu **TCP/IP** odpovídá funkcemi třem nejvyšším vrstvám modelu **OSI**, aplikační, prezentační a relační.
- Transportní vrstva odpovídá transportní vrstvě modelu **OSI.**
- Síťová vrstva modelu **TCP** odpovídá vrstvě síťové modelu **OSI**. V **TCP/IP** se však omezuje na službu bez spojení.
- Vrstva síťového rozhraní odpovídá linkové a fyzické vrstvě modelu **OSI**.



Obrázek 2 Porovnání vrstev modelů ISO/OSI a TCP/IP[2]

[2] Rita Pužmanová, TCP/IP v kostce, Kopp 2004, ISBN 80-7232-236-2

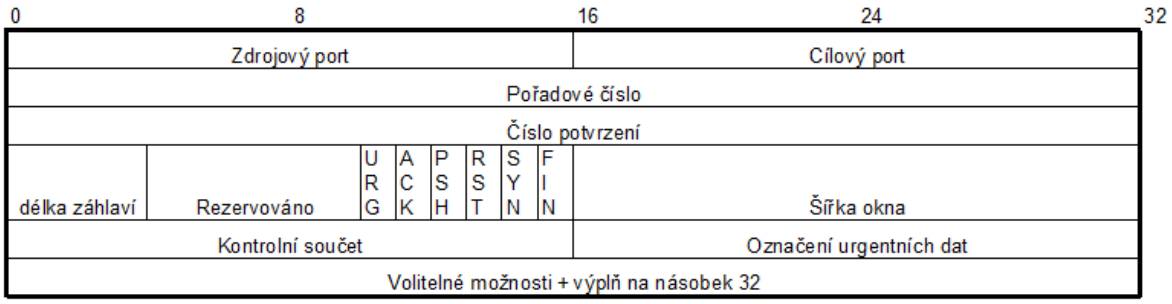
3 Protokoly v modelu TCP/IP

3.1 Protokoly Transportní vrstvy

3.1.1 Protokol TCP (Transmition Control Protocol)

Poskytuje spolehlivou službu se spojením, opakovaně odesílá nedoručená data. Protokoly využívající **TCP** jsou například: **HTTP**, **FTP**, **telnet**, **SMTP**, **DNS**. Funkce **TCP** jsou: navázání, ukončení spojení, asociace portů a spojení, segmentace dat, specifikace velikosti okna, výpočet kontrolního součtu, potvrzení příjmu dat.

TCP segment se zapouzdřuje do datagramu **IP**.



Obrázek 3 Hlavička TCP segmentu[3]

Obsah hlavičky **TCP**

- **Zdrojový port** – port zdrojového aplikačního procesu
- **Cílový port** – port cílového aplikačního procesu
- **Pořadové číslo** – číslo prvního bytu dat v segmentu, nezačíná jedničkou ale náhodným pořadovým číslem **ISN** (Initial Sequence Numer)
- **Číslo potvrzení** – **ACK** (Acknowledgment Number) pořadové číslo bytu který je očekáván jako následující
- **Délka záhlaví** – délka záhlaví v násobcích 32 bitů
- **Řídící funkce** – 6 bitových hodnot pro řízení spojení

  URG – urgentní data

  ACK – platnost pole s číslem potvrzení

[3] Rita Pužmanová, TCP/IP v kostce, Kopp 2004, ISBN 80-7232-236-2

PSH – požadavek na okamžité doručení segmentu vyšší vrstvě

RST – požadavek na okamžité ukončení spojení

SYN – žádost o navázání spojení

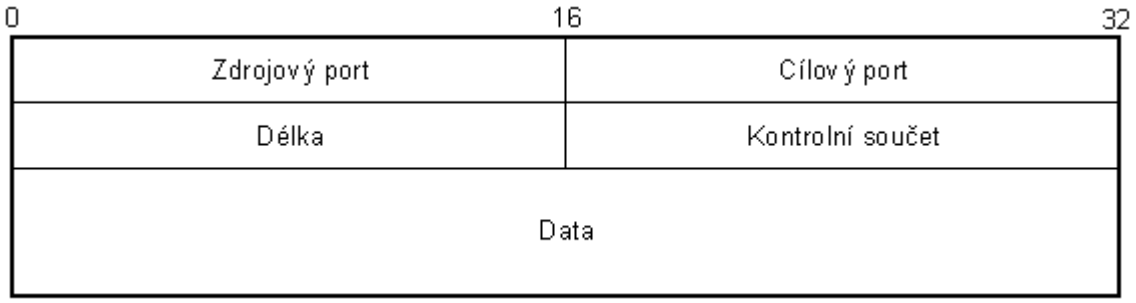
FIN – žádost o ukončení spojení

- **Šířka okna** – počet bytů, které lze odeslat bez průběžného potvrzení
- **Kontrolní součet** – zabezpečení přes celý segment včetně pseudozáhlaví
- **Označení urgentních dat** – poslední byt urgentních dat
- **Volitelné možnosti** – další informace o segmentu př.: maximální velikost segmentu. Případně doplněno na násobek 32

3.1.2 Protokol UDP (User Datagram Protocol)

Jednoduchý transportní protokol, poskytuje nespolehlivou službu bez spojení pro aplikace, které nevyžadují spolehlivost a funkce **TCP** jako navazování spojení, opakované odesílání dat jsou pro ně příliš zdlouhavé a mohlo by objemově překročit samotná data. **UDP** využívají například protokoly **TFTP**, **NTP**, **DHCP**.

UDP Datagram se stejně jako **TCP** segment zapouzdřuje do **IP** datagramu.



Obrázek 4 UDP datagram[4]

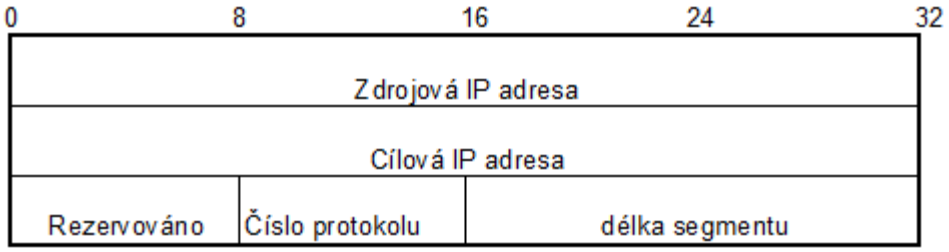
Obsah hlavičky **UDP**

- **Zdrojový port** – identifikuje zdrojový proces
- **Cílový port** – identifikuje cílový proces
- **Délka** – délka celého datagramu v násobcích 32
- **Kontrolní součet** – zabezpečení přes celý segment včetně pseudozáhlaví

[4] Rita Pužmanová, TCP/IP v kostce, Kopp 2004, ISBN 80-7232-236-2

3.1.3 Pseudozáhlaví

Záhlaví vytvořené z části **IP** hlavičky, slouží k výpočtu kontrolního součtu. Pseudozáhlaví se neposílá spolu se segmentem, pouze se vytváří ke kontrole chyb. Nechrání data proti útokům (pozměnění přenášených dat) pouze proti chybám způsobeným přenosem.



Obrázek 5 Pseudozáhlaví[5]

Obsah pseudozáhlaví

- **Zdrojová IP adresa** – **IP** adresa zdroje
- **Cílová IP adresa** – **IP** adresa cíle
- **Číslo protokolu** – číslo protokolu transportní vrstvy
- **Délka segmentu** – délka původního segmentu **TCP** nebo **UDP** bez pseudozáhlaví

3.2 Protokoly Síťové vrstvy

3.2.1 Protokol IP verze 4

Adresa **IPv4** má 32 bitů, maximálně tedy 4 294 967 296 adres. Ne všechny se však mohou použít pro koncové stanice, protože některé adresy jsou adresy sítě a adresy pro všesměrové vysílání (broadcast). Příklad **IP** adresy **192.168.1.1**

Třídy adres

- **Třída A –** 126 adres po 2^{24} - 2 uzlů. Největší sítě s nejmenším počtem uzlů. Pro adresu sítě se používá prvních 8bitů. Adresy sítí 1.0.0.0 - 126.0.0.0

[5] Rita Pužmanová, TCP/IP v kostce, Kopp 2004, ISBN 80-7232-236-2

- **Třída B –** 2^{14} adres po 2^{16} - 2 uzlů. Středně velké sítě se středním počtem uzlů, Pro adresu sítě se používá prvních 16bitů. Adresy sítí 128.1.0.0 - 191.255.0.0
- **Třída C –** 2^{21} adres po 2^{8} - 2 uzlů. Hodně malých sítí s malým počtem uzlů. Pro adresu sítě se používá prvních 24bitů. Adresy sítí 192.0.0.0 - 223.255.255.0
- **Třída D** – pro skupinové vysílání, adresy: 224.0.0.0 - 239.255.255.255
- **Třída E** – slouží pouze pro experimentální účely a jako rezerva, adresy 240.x.x.x. - 254.x.x.x

Používání adres třídy A v počátcích vedlo k velkému plýtvání adresami.

Privátní adresy

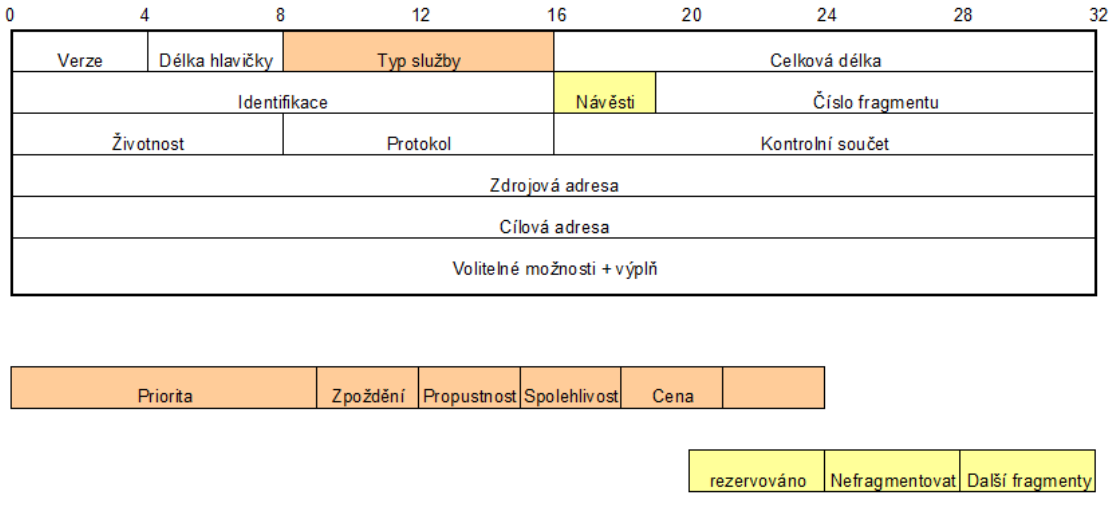
V prvních třech skupinách jsou definované privátní adresy pro stanice skryté za směrovačem, který působí jako brána do internetu. Pakety s privátní zdrojovou nebo cílovou adresou nejsou směrovány v síti, ale pokud nějaký omylem projde do internetu je zahozen.

Třída A – privátní adresa 10.0.0.0

Třída B – privátní adresy 172.16.0.0 - 172.31.0.0

Třída C – privátní adresy 192.168.0.0 - 192.168.255.0

IPv4 Datagram



Obrázek 6 Hlavička IPv4 datagramu[6]

Hlavička IP datagramu verze 4 má délku 20 bytů + volitelné možnosti.

[6] Rita Pužmanová, TCP/IP v kostce, Kopp 2004, ISBN 80-7232-236-2

Obsah hlavičky IP verze 4

- **Verze** (Version, 4bity) – verze IP protokolu v IPv4 vždy 4
- **Délka hlavičky** (Internet Leader Length, 4bity) – Délka hlavičky v násobcích 32 maximum 15*32 = 480 bitů
- **Typ služby** (Type of service, 8bitů) – Informace pro směrovače jak mají s datagramem zacházet

  Priorita – nejvyšší = 7

  Zpoždění – minimální zpoždění = 1

  Propustnost – maximální propustnost = 1

  Spolehlivost – maximální spolehlivost = 1

  Cena – minimální cena = 1
- **Celková délka** – celková velikost datagramu v násobcích 8 maximálně 65 535*8
- **Identifikace** – jedinečná identifikace datagramu, využívá se pro fragmentaci
- **Návěsti** – používají se pro fragmentaci

  Rezervováno – vždy 0

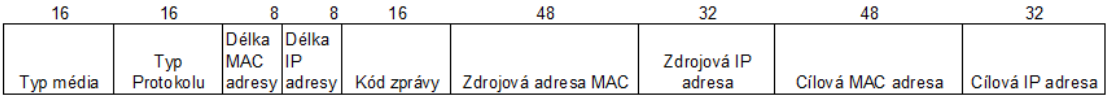
  Nefragmentovat – paket nemůže být cestou fragmentován (1 znamená nefragmentovat)

  Další fragmenty – označuje, jestli následují další fragmenty (poslední fragment =0)
- **Číslo fragmentu** – jednoznačně určuje pořadí fragmentu jako vzdálenost od začátku datagramu v násobcích 64
- **Životnost TTL** – označuje počet směrovačů, skrz které může paket projít, než je zničen
- **Protokol** – protokol vyšší, transportní vrstvy
- **Kontrolní součet** – zabezpečuje hlavičku proti chybám (pro výpočet se nastaví hodnota 0)
- **Zdrojová adresa** – 32 bitová adresa zdroje
- **Cílová adresa** – 32 bitová adresa cíle
- **Volitelné možnosti + výplň** – zabezpečení, záznam cesty sítí, dodržení předepsané cesty + výplň na násobek 32bitů

3.2.2 Protokol ARP

Protokol mapování adres, který při znalosti cílové adresy **IP** zjistí adresu **MAC**. Pokud chce jedna stanice poslat data jiné, musí znát její **IP** adresu, aby mohla vygenerovat datagram s cílovou adresou, která se nebude měnit, ten se zabalí do rámce se správnou **MAC** adresou (ta se bude při průchodu sítí měnit). Pokud stanice nezná cílovou **MAC** adresu, použije protokol **ARP**. Vyšle **ARP** dotaz s cílovou **IP** adresou adresovaný všem stanicím v cíli (všeobecná **MAC** adresa je FF:FF:FF:FF:FF:FF). Pokud je stanice v dosahu odpoví **ARP** odpovědí s příslušnou adresou. Pokud cílová stanice není ve stejné části sítě, odpoví směrovač, který zná cestu k cílové stanici.

ARP cache – obsahuje informace o **MAC** adresách, se kterými stanice komunikovala.



Obrázek 7 ARP rámec[7]

Obsah ARP rámce

- **Typ média** – přenosové médium na kterém je prováděna komunikace (Ethernet, Wi-Fi)
- **Typ protokolu** – typ použitého protokolu
- **Délka MAC adresy** – aby protokol **ARP** mohl fungovat pro více protokolů s různou délkou MAC adresy je uvedena její velikost
- **Délka IP adresy** – aby protokol **ARP** mohl fungovat pro více protokolů síťové vrstvy je v tomto poli uvedena její velikost
- **Kód zprávy** – 1 = žádost (request), 2 = odpověď (response)
- **Zdrojová MAC adresa** – fyzická adresa zdroje
- **Zdrojová IP adresa** – síťová adresa zdroje
- **Cílová MAC adresa** (v žádosti obsahuje nuly)
- **Cílová IP adresa** – adresa cíle ke kterému chceme zjistit MAC adresu

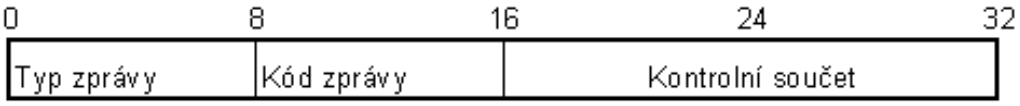
Protokol RARP

Funguje opačně než **ARP**. Při znalosti adresy **MAC** zjistí adresu **IP.**

[7] Rita Pužmanová, TCP/IP v kostce, Kopp 2004, ISBN 80-7232-236-2

3.2.3 Protokol ICMP

Protokol určený k přenosu zpráv o chybách a zvláštních okolnostech přenosu. Většina zpráv neslouží pro uživatele ale pro **IP** software. **ICMP** využívá služeb **IP** v rámci síťové vrstvy.



Obrázek 8 Hlavička ICMP[8]

Obsah ICMP hlavičky

- **Typ zprávy** – specifikuje typ **ICMP** zprávy
- **Kód zprávy** – určuje parametry zprávy
- **Kontrolní součet** – zabezpečení záhlaví proti chybám

Tabulka 1 Nejčastější typy ICMP zpráv[9]

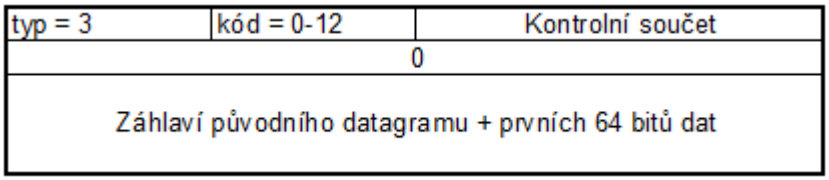
Typ hlášení	Kód zprávy	
0	0	Echo reply - odpověď na ping
8	0	Echo request - žádost o ping
3	0	Nedostupná síť
3	1	Nedostupná stanice
3	2	Nedostupný protokol
3	3	Nedostupný port
3	4	Nemožnost fragmentace když byla potřeba
3	5	Nevydařené směrování
4	0	Informace o zahlcení
5	0	Přesměrování datagramů pro síť
5	1	Přesměrování datagramů pro stanici
5	2	Přesměrování datagramů pro typ služby a síť
5	3	Přesměrování datagramů pro typ služby a stanici
11	0	TTL vypršelo
11	1	Vypršel čas znovusestavení datagramu
13	0	žádost o časové razítko
14	0	odpověď s časovým razítkem
15	0	žádost o adresu sítě
16	0	Odpověd s adresou sítě

[8] Rita Pužmanová, TCP/IP v kostce, Kopp 2004, ISBN 80-7232-236-2

[9] J. Postel Request for Comments: 792 INTERNET CONTROL MESSAGE PROTOCOL [cit. 2013-06-15]. Dostupné z http://www.ietf.org/rfc/rfc792.txt

Destination Unreachable

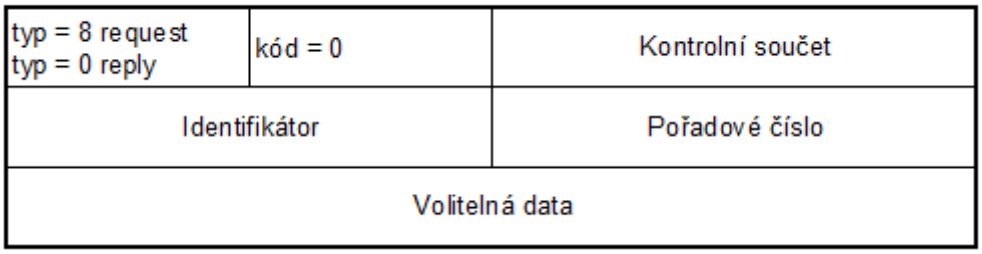
V případě nedostupnosti stanice (sítě, protokolu, portu, atd.) je jako součást datagramu odeslána zpět původní **IP** hlavička a prvních 64 bitů dat.



Obrázek 9 Formát zprávy ICMP destination unreachable[10]

Ping (Packet Internet Groper)

Slouží k ověření dostupnosti stanice nebo zařízení. Pracuje na nejnižší možné vrstvě, je implementován v operačním systému. Stanice vyšle zprávu **ICMP** echo request a očekává od cíle odpověď **ICMP** echo reply.



Obrázek 10 Formát ICMP echo zprávy[11]

[10] Rita Pužmanová, TCP/IP v kostce, Kopp 2004, ISBN 80-7232-236-2

[11] Rita Pužmanová, TCP/IP v kostce, Kopp 2004, ISBN 80-7232-236-2

```
C:\Users\Čábelka>ping www.seznam.cz

Pinging www.seznam.cz [77.75.76.3] with 32 bytes of data:
Reply from 77.75.76.3: bytes=32 time=6ms TTL=250
Reply from 77.75.76.3: bytes=32 time=5ms TTL=250
Reply from 77.75.76.3: bytes=32 time=5ms TTL=250
Reply from 77.75.76.3: bytes=32 time=8ms TTL=250

Ping statistics for 77.75.76.3:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 5ms, Maximum = 8ms, Average = 6ms
```

Obrázek 11 Příklad použití příkazu ping ve Windows

Traceroute

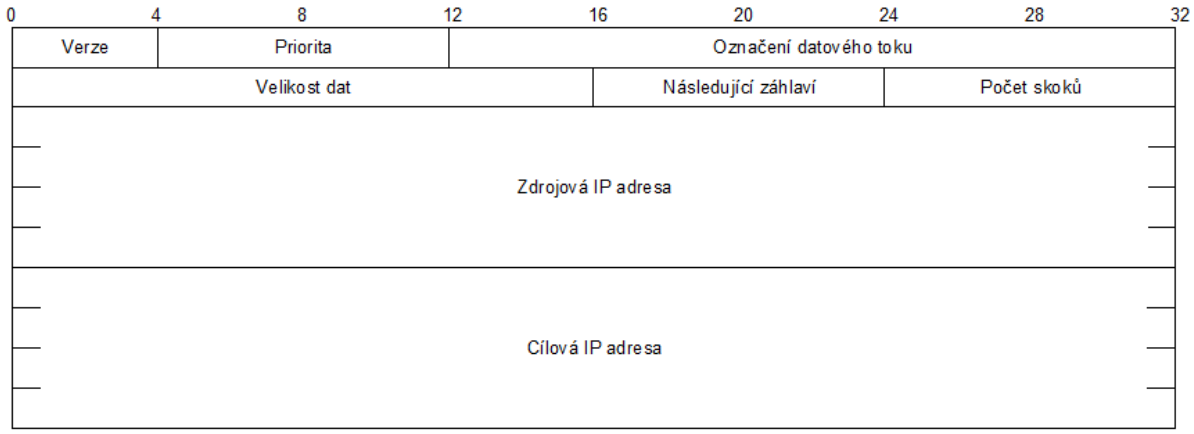
Slouží ke zjištění všech směrovačů v cestě k cíli. Vygeneruje se datagram (ve Windows obvykle **ICMP**, unixové systémy používají **UDP**, ale může být i **TCP**) s hodnotou **TTL** 1 a je ve třech kopiích odeslán k cíli (ten je na prvním směrovači zahozen a směrovač vyšle zpět zprávu **ICMP** Time exceeded − typ = 11, kód = 0), poté se generuje stejný datagram s postupně se zvyšujícím polem **TTL**, až je dosaženo cíle. Z odpovědí od směrovačů po cestě lze sestavit cestu k cíli.

```
C:\Users\Čábelka>tracert www.seznam.cz

Tracing route to www.seznam.cz [77.75.76.3]
over a maximum of 30 hops:

  1     1 ms    <1 ms    <1 ms  192.168.1.100
  2     2 ms     1 ms     1 ms  10.11.2.254
  3     3 ms     2 ms     2 ms  10.11.2.149
  4     5 ms     7 ms     4 ms  95.85.240.89
  5     6 ms     5 ms     5 ms  nix2.seznam.cz [91.210.16.194]
  6     5 ms     5 ms     5 ms  www.seznam.cz [77.75.76.3]

Trace complete.
```

Obrázek 12 Příklad použití příkazu tracert ve Windows

3.2.4 Protokol IP verze 6

Nová verze protokolu **IP**, byla vytvořena již v roce 1995. Nový protokol byl potřeba, protože ve stávajícím protokolu **IPv4** docházely adresy (zejména kvůli jejich plýtvání v počátku). **IPv6** adresa má 128 bitů, což umožňuje vytvoření mnohem většího počtu adres. Mezi další výhody patří zjednodušení hlavičky, lepší bezpečnost díky **IPSec** (bezpečností protokol pro autentizaci, kontrolu integrity atd.).



Obrázek 13 Hlavička IP verze 6[12]

Obsah IPv6 hlavičky

- **Verze** – verze protokolu v **IPv6** je vždy 6
- **Priorita** – priorita rychlosti přenosu, doručení
- **Označení datového toku** (flow label) – označuje datagramy, které vyžadují speciální zacházení, společně s prioritou slouží pro podporu **QoS**
- **Velikost dat** – délka dat uložených v datagramu
- **následující záhlaví** – označuje následující záhlaví
- **maximální počet směrovačů** (hop limit) – označuje počet směrovačů před zahozením datagramu (obdobně jako **TTL** u **IPv4**)
- **zdrojová adresa** – 128 bitová **IPv6** adresa zdroje
- **cílová adresa** – 128 bitová **IPv6** adresa cíle

[12] Abhaya S Induruwa IPv6 Header Format [online] 2001-12-16 [cit. 2013-07-15] Dostupné z: http://agenda.ictp.trieste.it/agenda_links/smr1335/networking/node35.html

IPv6 nepodporuje fragmentaci (stanice si musí zjistit maximální velikost **MTU** před odesláním datagramu), proto v záhlaví není pole pro fragmentaci, dále neobsahuje délku záhlaví, neboť záhlaví je pevné délky, ani kontrolní součet, neboť spoléhá na nižší vrstvu.

3.3 Protokoly vrstvy síťového rozhraní

3.3.1 Ethernet verze 2

Rámec Ethernet podle normy **IEEE802.**3 obsahuje místo pole typ **délku rámce.** Ale nahradil jej Ethernet verze 2.

Rámec Ethernet 2

7bytů	1byt	6bytů	6bytů	2byty	46-1500bytů	4byty
preambule	Označení začátku rámce					
7x 10101010	10101011	Cílová mac adresa	Zdrojová mac adresa	typ	Data	Zabezpečení

Obrázek 14 Rámec Ethernet 2[13]

Obsah rámce Ethernet 2

- **Preambule** – slouží k synchronizaci hodin příjemce a odesilatele
- **Označení začátku rámce** – **SFD** – Start of Frame Delimiter – označuje začátek rámce
- **Cílová MAC adresa** – fyzická adresa cíle
- **Zdrojová MAC adresa** – fyzická adresa zdroje
- **Typ** – označení typu rámce, určuje protokol vyšší vrstvy (**IP, ARP**)
- **Data** – přenášená data
- **Zabezpečení** – kontrolní součet **CRC** (slouží ke kontrole správnosti dat, kontrolní součet počítá zdrojová, cílová stanice a přepínače v režimu store and forward nebo směrovače.

[13] Rita Pužmanová, TCP/IP v kostce, Kopp 2004, ISBN 80-7232-236-2

3.3.2 Další protokoly vrstvy síťového rozhraní

Token Ring

Norma **IEEE802.5**. Přidělování práva na vysílání probíhá v kruhu. Byl vyvinut počátkem 80. let 20. století a dnes byl prakticky vytlačen Ethernetem.

FDDI

Fiber Distributed Data Interface, norma **ISO 9314**. je založeno na dvojité kruhové síti (optické nebo měděné dráty), druhý kruh může být záložní nebo pro zvětšení přenosové kapacity. Byla vyvinuta v 80. letech 20. století původně pro větší sítě **MAN.**

4. Síťová zařízení

4.1 Router (Směrovač)

Aktivní síťové zařízení pracující na síťové vrstvě (3. vrstva). Spojuje a odděluje od sebe dvě nebo více sítí (v každé síti je jiný rozsah **IP** adres). Směrovač může být specializované zařízení nebo obyčejný počítač. Specializované směrovače však nabízí vyšší rychlost. Ke směrování se využívá směrovací tabulka se záznamy nejlepší cesty sítí k cíli. Směrovač může fungovat jako brána do internetu, kdy na jedné straně je vnitřní síť a na druhé je připojen k internetu, uvnitř sítě se používají privátní adresy, které jsou ve směrovači překládány na adresu tohoto směrovače, aby mohly pakety putovat sítí.
Směrovací protokoly: **BGP**, **IGRP**, **EIGRP**, **OSPF**, **RIP**, **RIPv2**.

4.2 Switch a Bridge (Přepínač a most – dvouportový přepínač)

Aktivní síťové zařízení pracující na linkové vrstvě (2. vrstva). Odděluje od sebe segmenty jedné sítě. Odděluje kolizní doménu, protože nepřeposílá rámce na všechny porty, ale pouze na jeden, uživatel sítě tak nemůže vidět komunikaci v jiném segmentu sítě.

Způsoby přeposílání rámců
Store and forward (ulož a pošli) – celý rámec je nejprve přijat uložen do paměti, zkontrolován kontrolní součet a teprve pak je odeslán na příslušný port. Nejpomalejší metoda vhodná do sítí s velkým množstvím kolizí.
Cut through nebo **On the fly** (za letu) – rámec je odesílán na příslušný port hned jak je to možné (po načtení cílové MAC adresy). Snižuje nároky na paměť, zvyšuje rychlost přepínání, ale může odeslat i poškozený paket.
Fragment free – rámec se začne přeposílat po přijetí 64 bytů, kdy je jisté, že nevznikla kolize během odesílání.
Adaptive switching – přepínání mezi metodami store and forward a cut through.

4.3 Hub (opakovač)

Aktivní síťové zařízení pracující na fyzické vrstvě (1. vrstva). Umožňuje větvení sítě, prodlužuje maximální délku segmentu. Př.: 100 metrů u kabelu **cat5e** v Ethernetu. Po 100 metrech už signál začíná být zkreslený a mohou vznikat přeslechy z důvodu rušení signály jiných zařízení a kabelových vedení. Opakovač se vůbec nestará o data, která do něj přicházejí, pouze přijatý signál v podobě 1 a 0 obnoví a odešle na všechny porty. Zpoždění opakovače je tak pouze 1bit. Opakovače jsou v dnešní době nahrazovány přepínači, které jsou lepší z hlediska bezpečnosti. Neodesílají data na všechny porty ale pouze správným směrem.

4.4 Umístění zachytávače

Z hlediska zachytávání by bylo nejjednodušší, aby síť byla tvořena pouze opakovači, aby se veškerá komunikace dostala ke každému uzlu. V praxi je však právě z tohoto důvodu síť oddělena přepínači, tak aby data nešly do každého koutu sítě, ale pouze k cíli.

Například u **Wi-Fi** sítí je možnost (pokud máme fyzický přístup k **AP**) připojit opakovač s nejméně 3porty ke kabelu, který přivádí internetové připojení, tím zůstane zachována funkčnost připojení a na třetí port se umístí zachytávač, který může zachytávat celou komunikaci směřující z bezdrátové sítě do internetu.

4.4.1 Některé útoky na přepínač s cílem získání přístupu ke komunikaci

MAC flood attack

Cílem je pomocí náhodně generovaných rámců **ARP** zahltit **ARP** cache přepínače a ten se pak přepne do režimu opakovače a odesílá data na všechny porty.

ARP spoofing

Útok na přepínač kdy je odeslán **ARP** rámec s podvrženou **IP** adresou tak aby **IP** adresa brány (kam směřuje většina komunikace) byla spojena s **MAC** adresou našeho **PC** kde je spuštěn zachytávač, po zachycení může být komunikace odesílána na pravou bránu a uživatel tak nemusí tušit, že jeho data jsou odposlouchávána.

5. Jazyk Java a síť

Java je multiplatformní objektově orientovaný programovací jazyk, je jedním z nejoblíbenějších programovacích jazyků na světě. Pro spuštění programu stačí mít nainstalovanou podporu Javy.

5.1 Java.net

V jazyce Java existuje knihovna `Java.net`, obsahující různé třídy pro práci se sítí.

Datové typy knihovny `Java.net` rozdělené podle vrstvy, na které pracují.

- Aplikační vrstva – aplikace
- Transportní vrstva – `Socket, ServerSocket, DatagramPacket`
- Síťová vrstva – `InetAddress`
- Vrstva síťového rozhraní

`Socket` – kombinace **IP** adresa a port, slouží k vytvoření soketu, který slouží k navázání spojení.

možné konstruktory:

`Socket(String host, int port)` – vytvoří soket, který se připojí k adrese zadané slovně (například www.seznam.cz)

`Socket(InetAddress addr, int port)` – vytvoří soket, který se připojí k adrese zadané pomocí datového typu `InetAddress`

ServerSocket – soket vytvořený na straně serveru, konstruktor přijímá jako vstup pouze číslo portu, na kterém bude očekávat spojení

ServerSocket(int port)

`InetAddress` – třída reprezentující **IP** adresu, podporuje verze adresy **IPv4** i **IPv6**, konkrétně pomocí svých potomků `Inet4Address` a `Inet6Address`.

Konstruktory pro vytvoření IP adresy:

- `InetAddress.getLocalHost()` – vytvoří adresu na základě adresy vlastního počítače 127.0.0.1
- `InetAddress.getByAddress(byte[] addr)` – vytvoří adresu zadanou jako pole typu `byte`.
- `InetAddress.getByName(String host)` – vytvoří adresu na základě jména počítače
- `InetAddress.getByAddress(String host, byte[] addr)` – vytvoří adresu na základě jména počítače a adresy zadané jako pole typu `byte`.

`DatagramPacket`

Třída reprezentující UDP datagram. Obsahuje metody pro nastavení portu, délky, adresy a dat.

`DatagramSocket`

Třída reprezentující soket pro příjem datagramů na určitém portu. Tuto třídu nelze využít pro obecné zachytávání, neboť blokuje daný port (teoreticky by bylo možné vytvořit 65 tisíc soketů, ale přijímaly by se pouze **UDP** datagramy, některé porty by již mohly být obsazeny a celkově by to mělo negativní dopad na výkon.

`NetworkInterface`

Třída reprezentující síťové rozhraní. Obsahuje metody pro zobrazení **MAC** i **IP** adresy, **MTU**, jména rozhraní.

5.2 JPCAP

Externí knihovna slouží pro zachytávání síťových paketů v jazyce Java. Ve Windows využívá knihovnu WinPcap. Obsahuje datové struktury `NetworkInterface`, `Packet`.

WinPcap

Knihovna obsahující nástroje pro přístup na linkovou vrstvu ve Windows, umožňuje zachytávání a odesílání paketů.

5.2.1 Třída `Packet`

Paket z knihovny Jpcap obsahuje:

- **`int caplen`** a **`int len`** – integer obsahující délku zachyceného paketu v bytech
- **`byte[] header`** – bytové pole obsahující kombinaci hlavičky Ethernet, **IP** hlavičky a dalších (**TCP**, **UDP**, **ICMP** atd.)
- **`byte[] data`** – bytové pole obsahující data
- **`DatalinkPacket datalink`** – rámec linkové vrstvy, umožnuje vypsat zdrojovou, cílovou **MAC** adresu a typ rámce (tyto data jsou uloženy na začátku pole `header`)
- **`long sec`** a **`long usec`** – čas zachycení datagramu v unixovém tvaru (př.: 1363011466139) na datum se převede kontruktorem

  Date(packet.sec*1000 + packet.usec/1000)

Třída `Jpcap.NetworkInterface`

Třída reprezentující síťové rozhraní.

Třída `Jpcap.JpcapCaptor`

Třída umožňující zachytávání paketů na vybraném síťovém rozhraní, a čtení paketů ze souboru.

Třída `Jpcap.Jpcap Sender`

Třída pro odesílání vlastních vytvořených paketů.

Třída `Jpcap.Jpcap Writer`

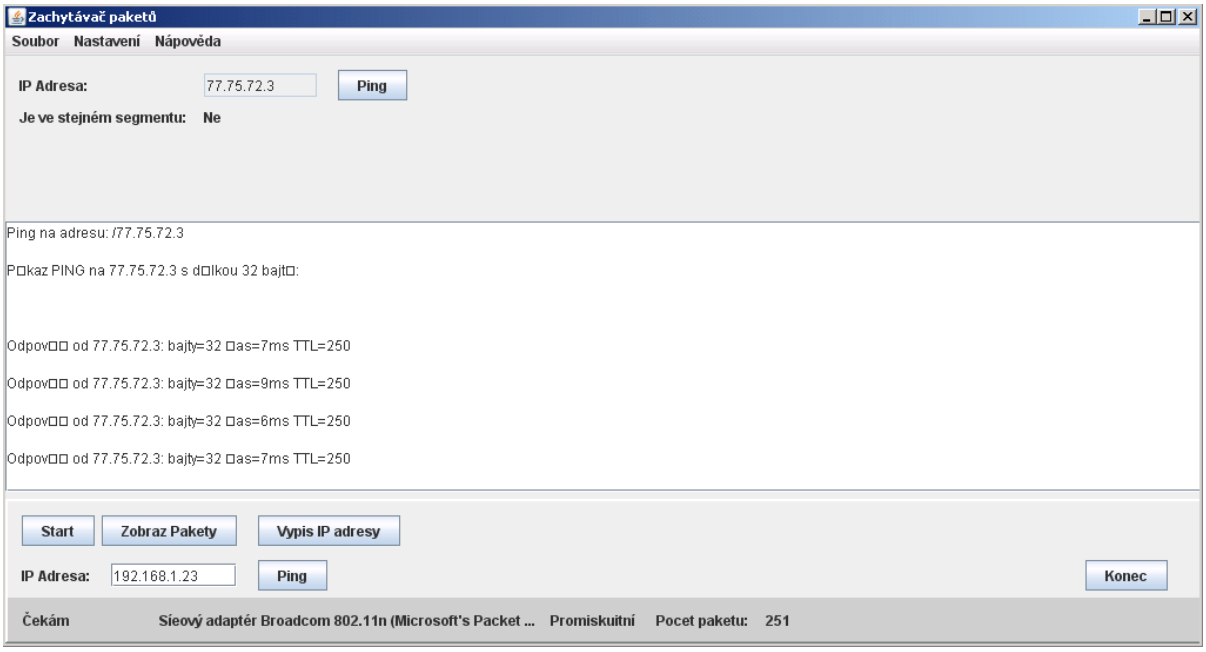
Třída pro zápis paketů do souboru.

Praktická a implementační část

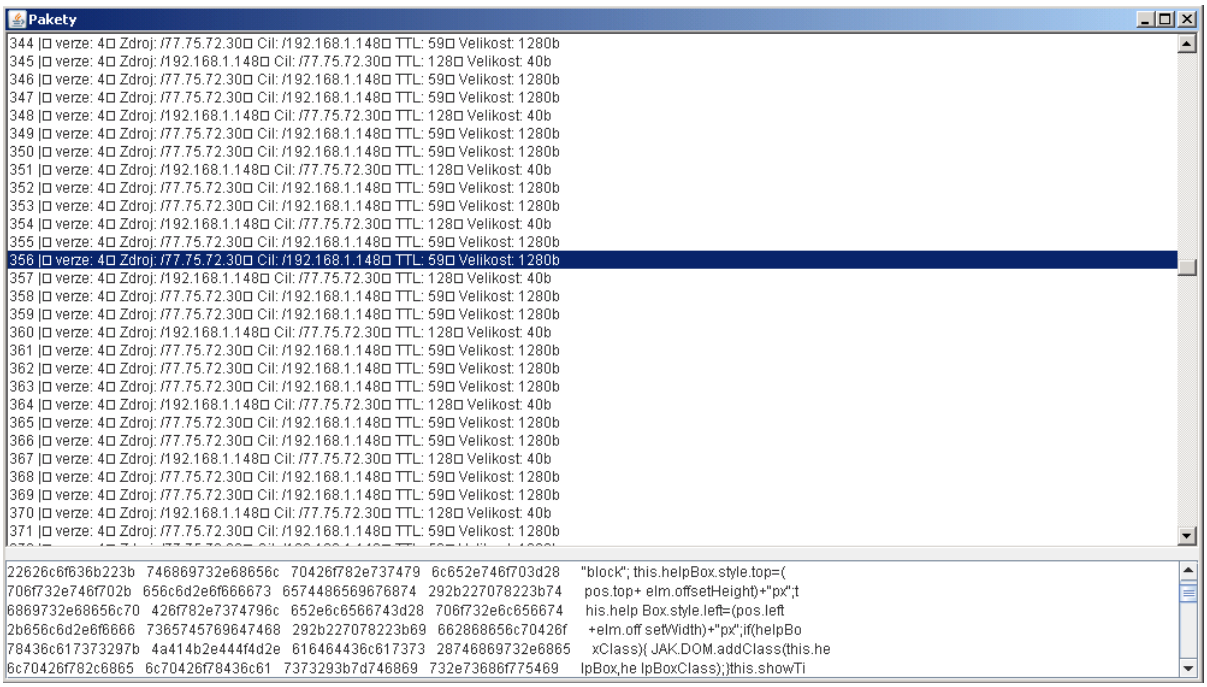
6 Návrh aplikace

6.1 Základ aplikace

Základem aplikace se stala semestrální práce z předmětu IJAV − Zachytávač paketů, kterou bylo třeba doplnit o práci se soubory (ukládání a načítaní), filtry zachycených dat a vylepšený systém analýzy, protože původní aplikace zobrazovala pouze vybrané části hlaviček. Vzhledem k množství potřebných úprav byla nakonec aplikace tvořena zcela od začátku, s využitím zkušeností z tvorby původní aplikace.



Obrázek 15 Hlavní okno semestrální práce: Zachytávač paketů.



Obrázek 16 Výpis paketů v semestrální práci.

6.2 Filtry zachycených paketů

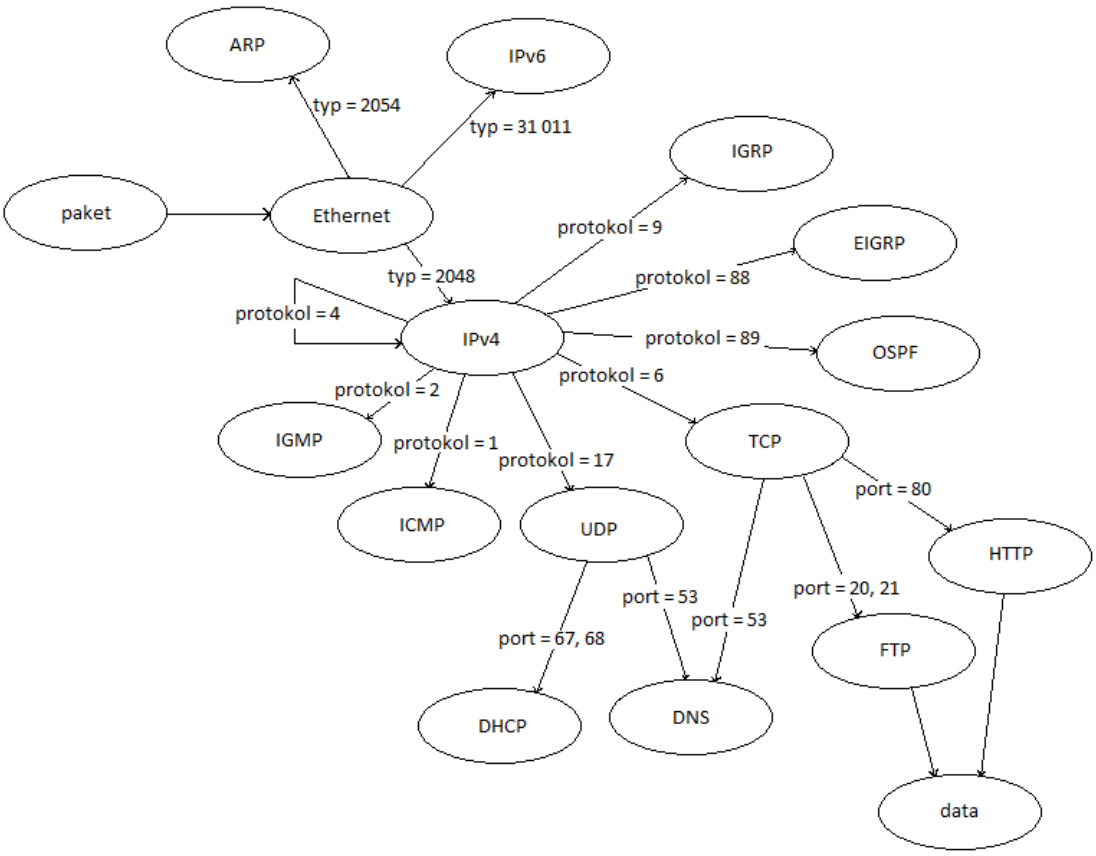
Protože při zachytávání je zachyceno velké množství paketů a tím i dat, je nutné je nějakým způsobem filtrovat. První možností je zachytávat pouze vybrané pakety a ty ukládat. Druhou možností je zachytávat a ukládat všechny pakety a následně odstranit ty které nejsou vhodné pro analýzu, například pokud jsou odeslány z jiného zařízení, jinou aplikací, atd.

Prvky vhodné jako podmínky filtrování

- Zdrojová a cílová MAC adresa
- Zdrojová a cílová IP adresa
- Zdrojový a cílový port
- Typ u rámce Ethernetu
- Čas zachycení paketu
- Typ služby

6.3 Analýza paketů

Aby bylo možné snadno doplňovat další a další analyzátory, bylo nejjednodušší navrhnout pro každou hlavičku vlastní třídu, která by se zabývala pouze touto jednou hlavičkou a následně volala další třídu dle zjištěných informací (typ na linkové vrstvě, protokol na síťové vrstvě, port na transportní vrstvě).



Obrázek 17 Návrh volání jednotlivých tříd v aplikaci.

7 Vývoj aplikace

7.1 Základ aplikace – zachytávací vlákno

Základem aplikace původně měla být semestrální práce z předmětu IJAV – Zachytávač paketů, ale vzhledem k množství potřebných úprav pro splnění zadání byla nakonec práce vytvořena od začátku, s využitím zkušeností z tvorby předešlé práce. Základem nové aplikace (stejně jako původní) je samostatné zachytávací vlákno, které je vytvořeno pomocí knihovny Jpcap.

Testování zachytávacího vlákna

Zachytávací vlákno bylo testováno souběžným spuštěním zachytávacího vlákna a programů Wireshark a JPCAPdumper (vytvořil K. Fujii který v posledních letech udržoval a aktualizoval knihovnu Jpcap. Výsledky (hlavně počty zachycených paketů a základní výpis) byly navzájem porovnávány, aby byla prokázána správná funkčnost zachytávacího vlákna.

7.2 Načítání a ukládání do souboru

Po otestování funkce zachytávacího vlákna následovalo vytvoření systému práce se soubory. Na výběr byly dvě varianty: ověřená knihovní třída `Jpcap.Writer`, která umožňuje zapisovat zachycené pakety do souboru, nebo vytvoření vlastního systému ukládání. Nakonec byla zvolena první varianta, která přináší výhodu v zachování kompatibility s programem `JPCAPdumper`, takže je možno analyzovat i pakety uložené v tomto programu a naopak.

7.3 Analýza paketů

Analyzovaný paket obsahuje dvě hlavní části: bytové pole `header` a bytové pole `data`. V poli `header` jsou za sebou umístěny hlavičky protokolů bez mezer, chybí ovšem úvodní část hlavičky Ethernetu (která by ovšem byla u každého paketu naprosto stejná, takže její vypuštění ušetří nároky na paměť. Pro každou hlavičku byl vytvořen vlastní soubor obsahující datové typy vhodné pro uložení prvků hlavičky a dále řetězec pro jejich výpis formátovaný pro přehlednost.

7.4 Grafické rozhraní – GUI

Grafické rozhraní bylo navrženo tak, aby bylo přehledné, obsahuje horní lištu s menu, kde jsou nástroje pro uložení a načtení ze souboru. Hlavní část rozdělenou na tři menší pro výpis paketu, seznamu jeho hlaviček a analyzované vybrané hlavičky. A dolní lištu s informativním výpisem zvoleného síťového zařízení a výběru promiskuitního módu.

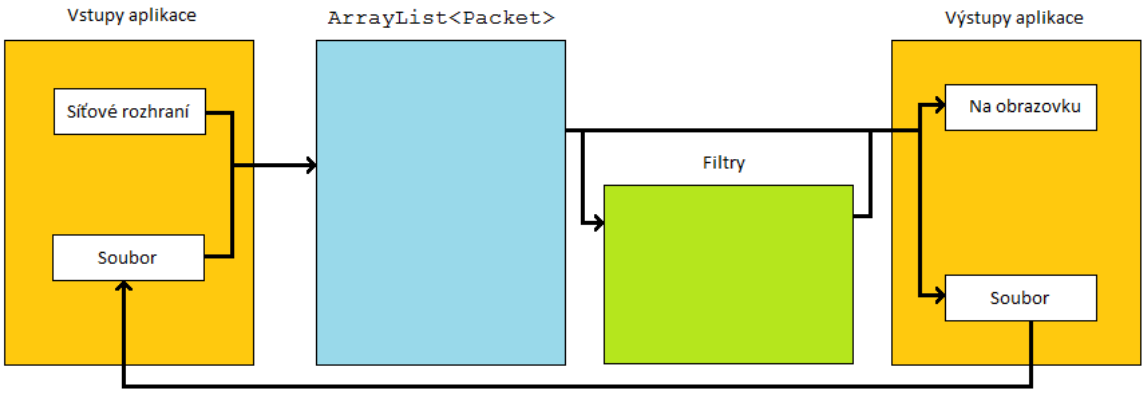
7.5 Testování aplikace

Testování celé aplikace probíhalo podobně jako testování zachytávacího vlákna. Spuštěním zachytávání a porovnáváním zachycených dat s ostatními zachytávači a porovnáváním zjištěných dat s aktuálně prováděnou síťovou komunikací jako například prohlížení webových stránek, spouštění příkazu ping a tracert, spouštění komunikačních nástrojů.

8 Struktura aplikace

Jednotlivé třídy byly rozděleny podle určení do dvou částí.

- Analysers – balíček pro umisťování analyzátorů jednotlivých hlaviček paketů
- Captor – balíček pro umístění zachytávače, hlavní části programu a GUI



Obrázek 18 Průchod paketů aplikací.

Pakety je možné zachytit na vybraném síťovém rozhraní (zachytávají se všechny pakety), nebo načíst ze souboru. V aplikaci jsou uchovány v `ArrayList<Packet>`. `ArrayList<Packet>` je generické pole prvků s proměnnou velikostí. Poskytuje operace přidání, odebrání, zjištění počtu prvků a další. Následně je možno seznam paketů projít filtrem a odstranit pakety, které nevyhoví zadané podmínce. Výstup analýzy se zobrazuje na obrazovce a seznam paketů může být uložen do souboru.

8.1 Použité nástroje

Aplikace byla vyvíjena v jazyce Java SE 7, ve vývojovém prostředí NetBeans IDE 7.1.1 v operačním systému Windows XP s rozšířením knihovnami Jpcap a WinPcap.

8.2 Instalace vývojového prostředí

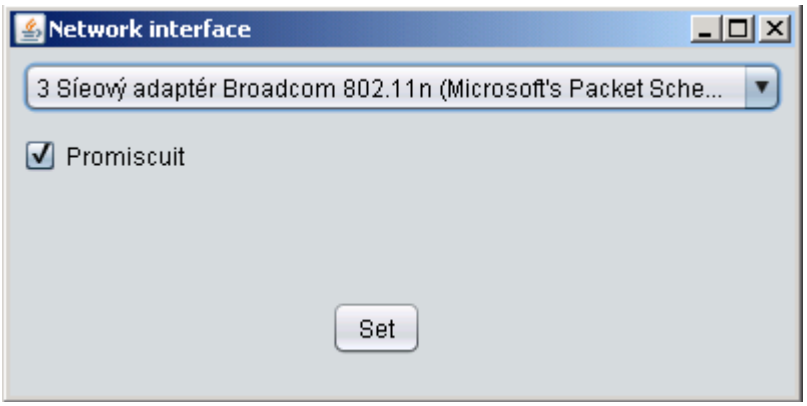
Jako první je třeba nainstalovat Java Development Kit. Po něm přichází na řadu vývojové prostředí NetbBeans IDE 7.1.1. Po instalaci vývojového prostředí je třeba ještě nainstalovat pomocné knihovny Jpcap a WinPcap, obě knihovny mají klasický instalátor, který nainstaluje vše automaticky po odsouhlasení licenčního ujednání.

Pro samotné spuštění aplikace je třeba mít nainstalovanou podporu jazyka Java, a dále knihovny Jpcap a WinPcap.

8.3 Vstupy aplikace

8.3.1 Zachycení paketu na síťovém rozhraní

Zachycení paketu zajišťuje třída `JpcapCaptor`, která pracuje jako samostatné vlákno, ve smyčce čeká na zachycení paketu, který je předán jako argument hlavní aplikaci. V hlavní aplikaci jsou pakety ukládány do `ArrayList<Packet>` pomocí příkazu `packets.add(packet)`. Před spuštěním zachytávání je třeba vybrat síťové rozhraní ze seznamu a zapnout nebo vypnout promiskuitní mód (při zapnutí budou zachytávány i pakety, které míří na jiné zařízení, při vypnutí budou zachytávány pouze pakety směřující na a z daného síťového rozhraní).



Obrázek 19 Okno výběru síťového rozhraní a promiskuitního módu.

```
@Override
public void run() {
    try {
        captor = JpcapCaptor.openDevice(netint, snaplen, promiscuit, timeout);
    } catch (IOException ioe) {
        Logger.getLogger(PacketCaptor.class.getName()).log(Level.SEVERE, null, ioe);
        return;
    }
    while (true) {
        packet = captor.getPacket();
        if (packet != null) {
            setChanged();
            notifyObservers(packet);
        }
        packet = null;
        if (!running) {
            break;
        }
    }
}
```

Obrázek 20 Metoda run vlákna pro zachytávání. Zdroj kód aplikace.

8.3.2 Načtení ze souboru

Načtení ze souboru zajišťuje stejně jako zachytávání knihovní třída `JpcapCaptor`, ovšem čte z vybraného souboru místo ze síťového rozhraní. Po zvolení souboru s pakety je soubor otevřen, pakety jsou čteny jeden za druhým a ukládány do seznamu paketů.

```
public ArrayList<Packet> loadFromFile() {
    ArrayList<Packet> packets = new ArrayList<Packet>();
    JFileChooser fc = new JFileChooser();
    fc.showOpenDialog(fc);
    String filePath = fc.getSelectedFile().getPath();
    a = true;
    try {
        captor = JpcapCaptor.openFile(filePath);
        do {
            packet = captor.getPacket();

            if ((packet.caplen == 0) && (packet.sec == 0) && (packet.usec == 0)) {
                a = false;
            } else {
                packets.add(packet);
            }
        } while (a == true);
    } catch (java.io.IOException ioe) {
    }
    return packets;
}
```

Obrázek 21 Metoda pro načtení paketů ze souboru

8.4 Filtrování paketů

Protože se zachytávají všechny pakety, což může být velké množství dat, je třeba z nich vybrat pouze ty, které splňují určité podmínky. Například pokud chceme odposlouchávat komunikaci mezi jinými počítači v síti a Internetem, jsou veškeré pakety jdoucí k vlastnímu počítači zbytečné a je možné je odstranit tlačítkem `delete my packets.` Naopak pokud chceme analyzovat vlastní komunikaci, jsou zbytečné veškeré pakety směřující k ostatním zařízením.

Postup filtrování

`ArrayList<Packet>` se prochází cyklem `for` a kontroluje se, zda paket vyhovuje zadané podmínce, pokud ne je z listu odstraněn. V aplikaci jsou umístěny dva seznamy paketů, na hlavní seznam se aplikují filtry a odstraňují se nevyhovující pakety. Druhý slouží jako záložní a jsou v něm uchovány všechny pakety. Pokud by byly z hlavního seznamu odstraněny pakety omylem, je možné je znovu načíst ze záložního seznamu tlačítkem `Reload packets`.

```
for (int i = 0; i < packets.size(); i++) {
        packet = packets.get(i);
        if ((packet.header[0] == captor.netint.mac_address[0]
                && packet.header[1] == captor.netint.mac_address[1]
                && packet.header[2] == captor.netint.mac_address[2]
                && packet.header[3] == captor.netint.mac_address[3]
                && packet.header[4] == captor.netint.mac_address[4]
                && packet.header[5] == captor.netint.mac_address[5])
                || (packet.header[6] == captor.netint.mac_address[0]
                && packet.header[7] == captor.netint.mac_address[1]
                && packet.header[8] == captor.netint.mac_address[2]
                && packet.header[9] == captor.netint.mac_address[3]
                && packet.header[10] == captor.netint.mac_address[4]
                && packet.header[11] == captor.netint.mac_address[5])) {
            packets.remove(i);
            i--;
        } else {
        }
    }
updateListPackets();
```

Obrázek 22 Odstranění paketů se shodnou MAC adresou

8.5 Analýza paketů

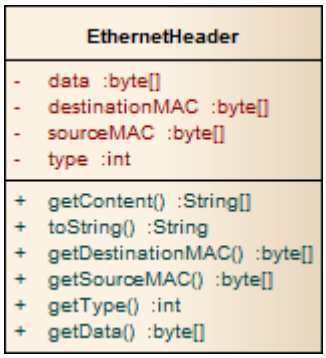
8.5.1 Analyzátory obecně

Konstruktor přijímá bytové pole `dataInput`, z něj převede počet prvků odpovídajících velikosti hlavičky příslušného protokolu a převede je na atributy, zbytek je převeden do pole data a je volán další konstruktor (pokud je znám a existuje příslušný analyzátor), pokud není znám nebo je analyzátor poslední, převede se pole data na řetězec, tímto se simuluje reálná síťová komunikace po vrstvách, kdy si každá vrstva odebere hlavičku a zbytek předá vyšší vrstvě ke zpracování.

8.5.2 EthernetHeader

`JpcapCaptor` vrací paket u kterého pole `header` začíná hlavičkou Ethernetu, ale bez synchronizace (preambule a **SFD**) na začátku a kontrolního součtu na konci, hlavička Ethernetu tak začíná cílovou **MAC** adresou.

Konstruktor přijímá paket, prvních 14 prvků z pole `header` je převedeno na **MAC** adresy a typ. Zbytek je převeden do bytového pole data a podle typu je volán konstruktor dalšího analyzátoru, analyzovaná data jsou vracena metodou `getContent()`, která vrací pole typu `String`, kde každý prvek obsahuje analýzu jedné hlavičky případně dat.



Obrázek 23 Diagram třídy EthernetHeader

```
public EthernetHeader(Packet packet) {
    destinationMacAddress[0] = packet.header[0];
    destinationMacAddress[1] = packet.header[1];
    destinationMacAddress[2] = packet.header[2];
    destinationMacAddress[3] = packet.header[3];
    destinationMacAddress[4] = packet.header[4];
    destinationMacAddress[5] = packet.header[5];
    sourceMacAddress[0] = packet.header[6];
    sourceMacAddress[1] = packet.header[7];
    sourceMacAddress[2] = packet.header[8];
    sourceMacAddress[3] = packet.header[9];
    sourceMacAddress[4] = packet.header[10];
    sourceMacAddress[5] = packet.header[11];
    type = (((packet.header[12] & 0xFF) * 256) + (packet.header[13] & 0xFF));
    size = (packet.header.length + packet.data.length) - 14;
    data = new byte[size];
    int h = 0;
    for (int i = 0; i < packet.header.length - 14; i++) {
        data[i] = packet.header[i + 14];
        h = i;
    }
    System.arraycopy(packet.data, 0, data, h, packet.data.length);
    switch (type) {
        case 2048://IPv4
            IPv4Header iphead = new IPv4Header(data);
            s = iphead.getContent();
            System.arraycopy(s, 0, strings, 1, s.length);
            break;
        case 2054://ARP
            ARP arp = new ARP(data);
            s = arp.getContent();
            System.arraycopy(s, 0, strings, 1, s.length);
            break;
    }
}
```

Obrázek 24 Konstruktor třídy EthernetHeader

8.5.3 Třída IPv4Header

Konstruktor přijímá pole typu `byte dataInput`, z něj odebere prvních 20 prvků a převede je na prvky hlavičky **IP**, pokud je `headerLength` větší než 0, je vytvořeno pole typu `byte options` a do něj je převeden počet prvků rovnajících se rozdílu `headerLength` a standardní velikosti **IPv4** hlavičky což je 20. Zbytek vstupních dat je zkopírován do pole `data`. Následně je volán analyzátor vyšší vrstvy podle hodnoty atributu `protocol`.

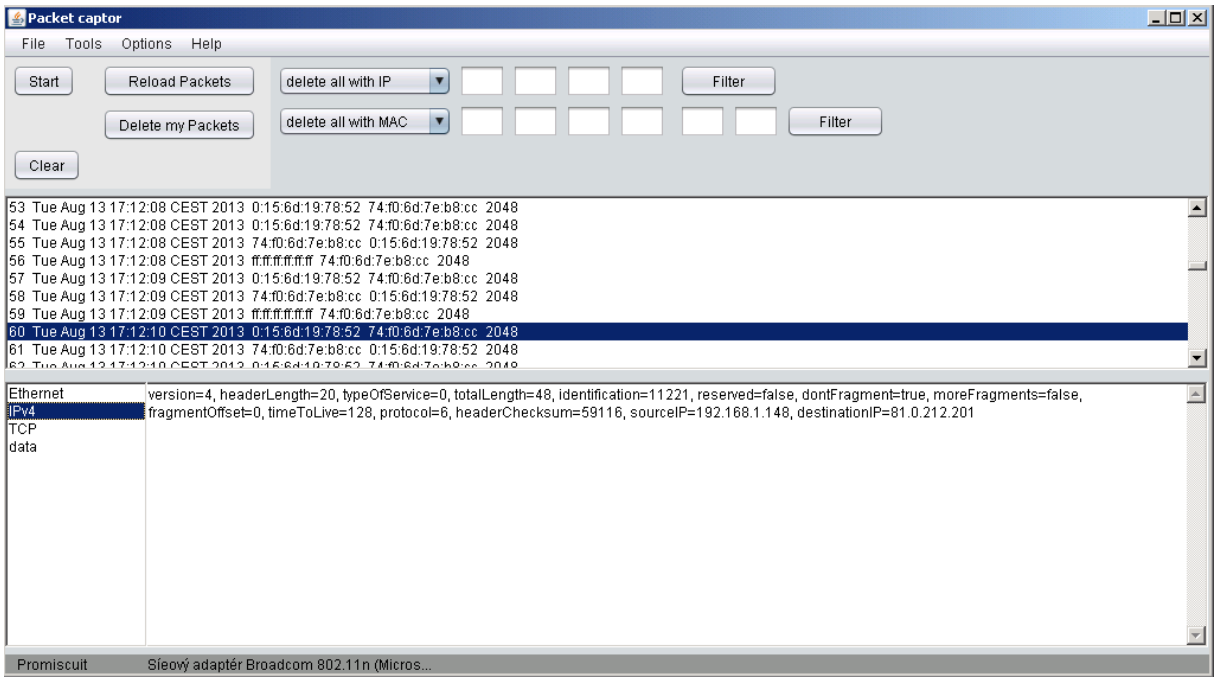
```
public IPv4Header(byte[] dataInput) {
    headerLength = (short) ((dataInput[0] & 0x0F) * 4);
    typeOfService = dataInput[1];
    totalLength = (dataInput[2] & 0xFF) * 256 + (dataInput[3] & 0xFF);
    identification = (dataInput[4] & 0xFF) * 256 + (dataInput[5] & 0xFF);
    if ((dataInput[6] & 0x80) == 0) {
        reserved = false;
    } else {
        reserved = true;
    }
    if ((dataInput[6] & 0x40) == 0) {
        dontFragment = false;
    } else {
        dontFragment = true;
    }
    if ((dataInput[6] & 0x20) == 0) {
        moreFragments = false;
    } else {
        moreFragments = true;
    }
    fragmentOffset = dataInput[6] & 0x1F + dataInput[7] & 0xFF;
    timeToLive = (short) (dataInput[8] & 0xFF);
    protocol = (short) (dataInput[9] & 0xFF);
    headerChecksum = (dataInput[10] & 0xFF) * 256 + (dataInput[11] & 0xFF);
    sourceIP[0] = dataInput[12];
    sourceIP[1] = dataInput[13];
    sourceIP[2] = dataInput[14];
    sourceIP[3] = dataInput[15];
    destinationIP[0] = dataInput[16];
    destinationIP[1] = dataInput[17];
    destinationIP[2] = dataInput[18];
    destinationIP[3] = dataInput[19];
    options = new byte[headerLength - 20];
    for (int i = 0; i < options.length; i++) {
        options[i] = dataInput[i + 20];
    }
    data = new byte[dataInput.length - headerLength];
    for (int i = 0; i < dataInput.length - headerLength; i++) {
        data[i] = dataInput[i + 20];
    }
    String[] s;
    switch (protocol) {
        case 6: //TCP
            TCPHeader tcphead = new TCPHeader(data);
            s = tcphead.getContent();
            System.arraycopy(s, 0, strings, 1, s.length);
            break;
        case 17: //UDP
            UDPHeader udphead = new UDPHeader(data);
            s = udphead.getContent();
            System.arraycopy(s, 0, strings, 1, s.length);
            break;
        }
    }
```

Obrázek 25 Konstruktor třídy IPv4Header

8.6 Výstupy aplikace

Výstup z aplikace lze rozdělit na dvě části. První je výstup analyzovaných paketů na obrazovku, kde se vybraná část paketu zobrazí v části okna. Druhá možnost je ukládat celé pakety do souboru, aby je bylo možno procházet a analyzovat později.

8.6.1 Výstup na obrazovku



Obrázek 26 Hlavní okno aplikace

Pro paket vybraný ve střední části okna se dole v levém rámečku zobrazí nabídka jednotlivých hlaviček a dat, po kliknutí na příslušnou hlavičku se vypíše v pravé části obrazovky. Na přiloženém obrázku je vybrán paket obsahující IPv4 hlavičku zobrazenou v pravé části. V horní části obrazovky jsou vidět tlačítka ke spuštění zachytávání, vymazání všech paketů, obnovení paketů ze záložního seznamu a filtrování. V dolní informační liště je vidět zapnutý promiskuitní mód a vybrané síťové zařízení k zachytávání.

8.6.2 Uložení do souboru

Knihovna `Jpcap` poskytuje funkce pro ukládání zachycených paketů do souboru. Konkrétně se jedná o třídu `Jpcap.Writer`. Ta do vytvořeného souboru ukládá pakety tak, aby je bylo možné znovu načíst a provádět analýzu kdykoliv, ne jen ihned po zachycení.

Pakety v aplikaci držené v `ArrayList<Packet>` je možno uložit do souboru pomocí metody `saveToFile`, která přijímá jediný argument `ArrayList<Packet>`. Po zvolení umístění a názvu souboru je v cyklu uložen každý paket. Takto uložený soubor může otevřít kterákoliv jiná aplikace využívající knihovní třídy `Writer` a `Captor`.

```
public void saveToFile(ArrayList<Packet> packets) {
    if (packets != null) {
        JFileChooser fc = new JFileChooser();
        fc.showSaveDialog(fc);
        File f = fc.getSelectedFile();
        try {
            JpcapWriter writer = JpcapWriter.openDumpFile(captor, f.getPath())
            for (Packet p : packets) {
                writer.writePacket(p);
            }
            writer.close();
        } catch (java.io.IOException ioe) {
        }
    }
}
```

Obrázek 27 Metoda pro uložení paketů do souboru

Závěr

Tato bakalářská práce měla za cíl vytvořit program pro zachytávání a analýzu síťových paketů. V teoretické části byly popsány síťové modely, IP adresy a protokoly dále síťová zařízení, jejich funkce a popis práce se sítí v jazyce Java. Tyto teoretické poznatky jsou následně využity v praktické části a v aplikaci, která umí zachytávat síťovou komunikaci a zobrazovat na obrazovku.

Během programování aplikace jsem narazil na jediný vážnější problém a to bylo, když přestaly fungovat webové stránky knihovny Jpcap, kde byla veškerá dokumentace, návody, instalátory a zdrojové kódy ke knihovně a vzorové aplikace. Naštěstí jsem měl knihovnu i aplikaci staženou a dalo se bez dokumentace obejít.

Aplikace dále nabízí možnost doplnění dalších filtrů, analýzu dalších protokolů a vylepšení stávajících například o textový popis zjištěných vlastností paketu. Přidání funkcí pro odesílání vlastních paketů, například pro ping a traceroute, dále tvorbu statistik, měření objemu komunikace atd.

Tvorba bakalářské práce pro mě byla přínosná, zopakoval jsem si síťové protokoly, vyzkoušel jsem si tvorbu kompletní aplikace v jazyce Java a měl jsem příležitost podívat se, co vše putuje po počítačové síti. Například že pro načtení jedné webové stránky (www.seznam.cz) je potřeba přenést 300 - 600 paketů.

Literatura

Knižní zdroje

[1] Pecinovský,R., Návrhové vzory, Computer Press, 2007 vydání první, ISBN 978-80-251-1582-4

[2] Page-Jones, Meilir, Základy objektově orientovaného návrhu v UML, Grada 2001, ISBN 80-247-0210-X

[3] Rita Pužmanová, TCP/IP v kostce, Kopp 2004, ISBN 80-7232-236-2

Internetové zdroje

[1] J. Postel Request for Comments: 792 INTERNET CONTROL MESSAGE PROTOCOL [cit. 2013-06-15]. Dostupné z http://www.ietf.org/rfc/rfc792.txt

[2] K.Fujii JPCAP library [cit. 2012]. Od roku 2013 nedostupné. Původní adresa: http://www.netresearch.ics.uci.edu/kfujii/jpcap/doc/index.html

[3] Abhaya S Induruwa IPv6 Header Format [online] 2001-12-16 [cit. 2013-07-15] Dostupné z: http://agenda.ictp.trieste.it/agenda_links/smr1335/networking/node35.html

[4] Programming tutorials and source code examples [online] 2009-2012[cit. 2013] Dostupné z: http://java2s.com/

Příloha A – Obsah CD

Aplikace

Zdrojový kód aplikace (NetBeans projekt)

Instalátor knihovny Jpcap

Instalátor knihovny WinPcap

Text bakalářské práce

Příloha B – Paket analyzovaný pomocí aplikace

Pole typu `byte` vypsané v desítkové soustavě

0 21 109 25 120 82 116 240 109 126 184 204 8 0 69 0 1 249 23 156 64 0 128 6 137 216 192 168 1 148 77 75 72 3 4 171 0 80 142 194 238 4 171 125 36 59 80 24 68 112 136 207 0 0

EthernetHeader

sourceMacAddress = 74:f0:6d:7e:b8:cc, destinationMacAddress = 0:15:6d:19:78:52, type=2048

IPv4Header

version=4, headerLength=20, typeOfService=0, totalLength=505, identification=6044, reserved=false, dontFragment=true, moreFragments=false, fragmentOffset=0, timeToLive=128, protocol=6, headerChecksum=35288, sourceIP=192.168.1.148, destinationIP=77.75.72.3

TCPHeader

sourcePort=1195, destinationPort=80, sequenceNumber=2395139588, ACKNumber=4278199355, headerLength=20, reserved=0, urg=false, ack=true, psh=true, rst=false, syn=false, fin=false, windowSize=112, checksum=207, urgentPointer=71, options=[B@1735602

Data

71 69 84 32 47 32 72 84 84 80 47 49 46 49 13 10 72 111 115 116 58 32 119 119 119 46 115 101 122 110 97 109 46 99 122 13 10 85 115 101 114 45 65 103 101 110 116 58 32 77 111 122 105 108 108 97 47 53 46 48 32 40 87 105 110 100 111 119 115 32 78 84 32 53 46 49 59 32 114 118 58 50 50 46 48 41 32 71 101 99 107 111 47 50 48 49 48 48 49 48 49 32 70 105 114 101 102 111 120 47 50 50 46 48 13 10 65 99 99 101 112 116 58 32 116 101 120 116 47 104 116 109 108 44 97 112 112 108 105 99 97 116 105 111 110 47 120 104 116 109 108 43 120 109 108 44 97 112 112 108 105 99 97 116 105 111 110 47 120 109 108 59 113 61 48 46 57 44 42 47 42 59 113 61 48 46 56 13 10 65 99 99 101 112 116 45 76 97 110 103 117 97 103 101 58 32 99 115 44 101 110 59 113 61 48 46 55 44 101 110 45 117 115 59 113 61 48 46 51 13 10 65 99 99 101 112 116 45 69 110 99 111 100 105 110 103 58 32 103 122 105 112 44 32

100 101 102 108 97 116 101 13 10 67 111 111 107 105 101 58 32 104 105 110 116 61 77 84 77 51 78 84 73 52 79 84 103 50 78 105 119 119 79 122 77 49 76 68 69 115 77 84 77 51 77 106 103 48 79 68 85 48 77 121 119 122 59 32 102 116 120 116 95 112 114 101 102 61 113 119 80 116 52 76 65 120 119 101 74 45 52 76 80 120 90 99 74 74 110 99 74 104 48 98 49 57 84 82 80 55 122 98 102 109 110 118 57 111 119 118 122 114 59 32 103 101 116 73 101 49 48 67 111 117 110 116 61 54 48 59 32 100 112 61 99 111 108 117 109 110 106 115 59 32 115 122 110 109 97 105 108 100 111 109 97 105 110 61 115 101 122 110 97 109 46 99 122 59 32 104 119 61 56 54 52 48 48 13 10 67 111 110 110 101 99 116 105 111 110 58 32 107 101 101 112 45 97 108 105 118 101 13 10 13 10

GET / HTTP/1.1

Host: www.seznam.cz

User-Agent: Mozilla/5.0 (Windows NT 5.1; rv:22.0) Gecko/20100101 Firefox/22.0

Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8

Accept-Language: cs,en;q=0.7,en-us;q=0.3

Accept-Encoding: gzip, deflate

Cookie: hint=MTM3NTI4OTg2NiwwOzM1LDEsMTM3Mjg0ODU0Mywz; ftxt_pref=qwPt4LAxweJ-4LPxZcJJncJh0b19TRP7zbfmnv9owvzr; getIe10Count=60; dp=columnjs; sznmaildomain=seznam.cz; hw=86400

Connection: keep-alive