Slovenská technická univerzita v Bratislave

Fakulta informatiky a informačných technológii

Ilkovičova 2, 842 16 Bratislava 4

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Zadanie 1.: Analyzátor sieťovej komunikácie

Počítačové a komunikačné siete

Autor: Ivan Gulis

Študijný program: Informatika

Semester: 4.

Ak. rok: 2014/2015

Zadanie úlohy

Navrhnite a implementujte programový “post” analyzátor Ethernet siete, ktorý analyzuje komunikáciev sieti zaznamenané v súbore a poskytuje nasledujúce informácie o komunikáciách.

1) Výpis všetkých komunikácií, t.j. všetkých rámcov v hexadecimálnom tvare postupne tak, ako boli zaznamenané v súbore.

Pre každý rámec uveďte:

- Poradové číslo rámca v analyzovanom súbore.

- Dĺžku rámca v bajtoch poskytnutú paketovým drajverom, ako aj dĺžku tohto rámca

prenášaného po médiu.

- Typ rámca – Ethernet II, IEEE 802.3 (IEEE 802.3 - LLC, IEEE 802.3- LLC - SNAP, IEEE

802.3 – Raw).

- Zdrojovú a cieľovú fyzickú (MAC) adresu uzlov, medzi ktorými je rámec prenášaný.

- Na konci výpisu uveďte IP adresy všetkých vysielajúcich uzlov, ako aj IP adresu uzla, ktorý

sumárne odvysielal (bez ohľadu na príjemcu) najväčší počet bajtov. Počet bajtov taktiež

uveďte. Uvažujte iba IP adresy vnorené v rámci Ethernet II.

Vo výpise jednotlivé bajty usporiadajte po 8, 16 alebo 32 v jednom riadku. Pre prehľadnosť

výpisu je vhodné použiť neproporcionálny font.

2) Študent musí vedieť vysvetliť, aké informácie sú uvedené v jednotlivých rámcoch Ethernet II, t.j. vnáranie protokolov ako aj ozrejmiť dĺžky týchto rámcov.

3) V súboroch so zachytenými komunikáciami analyzujte zadané typy komunikácií. Analýzu cez jednotlivé vrstvy vykonajte len pre rámce Ethernet II a protokoly rodiny TCP/IPv4.

Analyzované komunikácie s protokolmi:

a) HTTP komunikácie

b) HTTPS komunikácie

c) TELNET komunikácie

d) SSH komunikácie

e) FTP riadiace komunikácie

f) FTP dátové komunikácie

g) Všetky TFTP komunikácie

h) Všetky ICMP komunikácie

i) Všetky ARP dvojice (request – reply).

Vo všetkých výpisoch treba uviesť aj IP adresy a pri transportných protokoloch aj porty komunikujúcich uzlov.

V prípadoch komunikácií so spojením vypíšte iba jednu kompletnú komunikáciu - obsahuje

otvorenie (SYN) a ukončenie (FIN na oboch stranách alebo ukončenie FIN a RST alebo ukončenie iba s RST) spojenia a aj prvú nekompletnú komunikáciu, ktorá obsahuje iba otvorenie spojenia. Pri výpisoch vyznačte, ktorá komunikácia je kompletná.

Pre nájdenú a vypísanú kompletnú komunikáciu urobte štatistiku výskytu dĺžky rámca Ethernet II v bajtoch, a to pre dĺžky 0 až19, 20 až 39, 40 až 79, atď. (postupne x až (2x-1)).

V prípade výpisu h) uveďte aj typ ICMP správy (pole Type v hlavičke ICMP), napr. Echo request, Echo reply, Time exceeded, a pod.

V prípade výpisu i) uveďte pri ARP-Request IP adresu, ku ktorej sa hľadá MAC (fyzická) adresa

a pri ARP-Reply uveďte konkrétny pár - IP adresa a nájdená MAC adresa. V prípade, že bolo

poslaných viacero rovnakých rámcov ARP-Request, vypíšte všetky.

Ak počet rámcov danej komunikácie je väčší ako 20, vypíšte iba 10 prvých a 10 posledných

rámcov.

Pri všetkých výpisoch musí byť poradové číslo rámca zhodné s číslom rámca v analyzovanom

súbore.

4) Program musí byť organizovaný tak, aby čísla protokolov v rámci Ethernet II a v IP pakete ako aj čísla portov v transportných protokoloch boli programom určené z externého súboru a pre známe protokoly a porty boli uvedené aj ich názvy.

5) V procese analýzy rámcov pri identifikovaní jednotlivých polí rámca ako aj polí hlavičiek

vnorených protokolov nie je povolené použiť funkcie poskytované použitým programovacím

jazykom. Celý rámec je potrebné spracovať postupne po bajtoch.

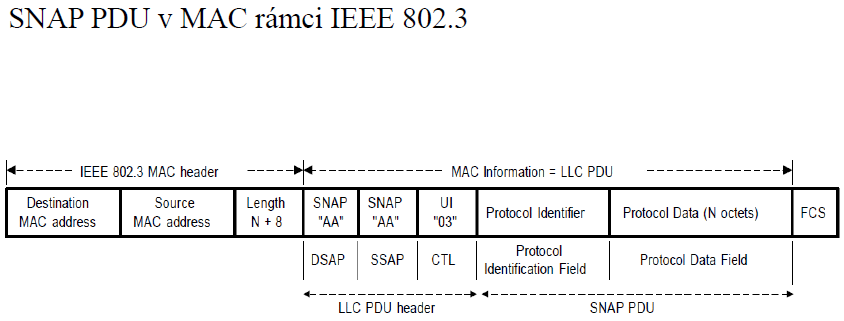
6) Program musí byť organizovaný tak, aby bolo možné jednoducho rozširovať jeho funkčnosť o výpis rámcov podľa ďalších požiadaviek na protokoly v bode 3) - pri doimplementovaní jednoduchej funkčnosti na cvičení.

7) Študent musí byť schopný preložiť a spustiť program v miestnosti, v ktorej má cvičenia!

**Koncepcia riešenia**

Úlohou je spraviť analyzátor sieťovej komunikácie. Pre sieťovú komunikáciu sa využívajú komunikačné protokoly, ktoré sú na určitých vrstvách a cez ne komunikujú vrstvy so susednými vrstvami a paralelne s vrstvami na druhej strane komunikácie. K tomuto sa využívajú PDU (protocol data unit), čiže protokolové dátové jednotky. Každá vrstva si pridá svoju hlavičku (zvyčajne) na začiatok a všetko pošle ďalšej vrstve, ktorá si pridá svoju hlavičku znova na začiatok, tento proces sa nazýva enkapsulácia, a je to vertikálna komunikácia medzi vrstvami. Opakom je de-enkapsulácia, kde si vrstvy na opačnej strane komunikácie postupne odoberajú tieto hlavičky, každá vrstva odoberie hlavičku jej určenú, vytvorenú rovnakou vrstvou na druhej strane (horizontálna komunikácia).

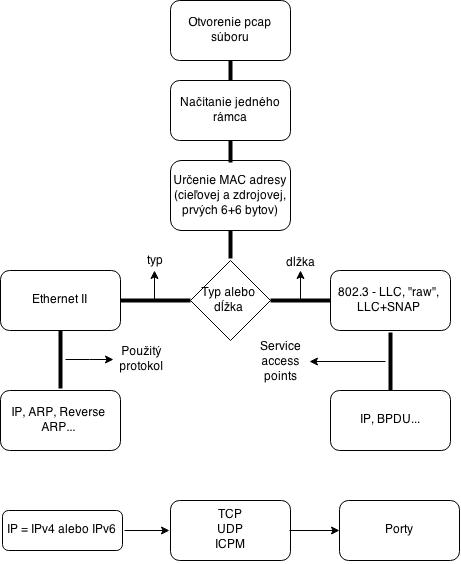
Z tohto reťazca dát je potrebné rozlíšiť jednotlivé hlavičky a postupne ich po bytoch analyzovať.



Príklad z prednášky, PDU kde je možné vidieť hlavičku napr. fyzickej vrstvy (MAC header) a LLC PDU header. Každý úsek má určitú dĺžku v bytoch. Program pristupuje k daným skupinám a ukladá si získané informácie.

**Mechanizmus**

Približný postup analýzy vrstiev:



Toto sa opakuje pre všetky rámce v pcap súbore pomocou funkcie nextEx(). Informácie kam sa v diagrame posunúť získam priamo z rámca prečítaním správnych bytov funkciou getUByte() do hexadecimálneho stringu.

**Analýza komunikácie so spojením**

Toto spojenie sa nazýva relačné, medzi príjemcom a odosielateľom. Týka sa to protokolu TCP, pretože UDP má služby bez spojenia a bez potvrdenia.

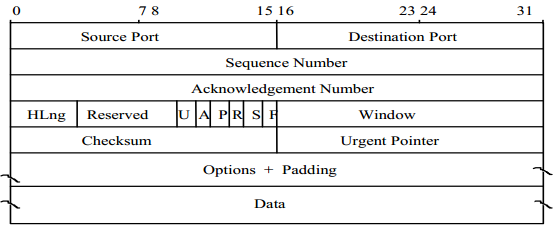
Odosielateľ segment odošle a čaká na potvrdenie (zároveň zapne časovač a po jeho uplynutí, ak nepríde potvrdenie, odošle segment znova). Príjemca musí doručenie segmentu potvrdiť.

Pri tejto komunikácii sa využívajú príznaky (flags) SYN, FIN, ACK.

Spojenie sa nadviaže v 3 krokoch (výmena 3 segmentov):

Strana, ktorá požaduje prenos(klient) odošle správu(segment) s príznakom SYN 1. Strana, ktorá dáta odosiela(server) zas pošle segment s príznakmi SYN 1 a ACK 1. Potom už zostáva prvej strane odoslať segment s príznakom ACK 1.

V nasledujúcich segmentoch sú v TCP hlavičke za portami poradové čísla segmentov a potvrdzovacie čísla, aby sa dodržalo poradie.



Na obrázku je jeden TCP segment, kde je vidieť spomenuté čísla(sequence number a acknowledgement number) a rôzne druhy príznakov. Obrázok je prevzatý z prednášky.

Ukončenie komunikácie sa robí pomocou výmeny segmentov s príznakmi FIN a ACK v štyroch krokoch (výmena 4 segmentov). Klient najprv pošle FIN 1 a ACK 1, server odpovie ACK (spojenie je polouzatvorené). Potom server pošle FIN a ACK príznaky, a klient odpovie ACK, spojenie je uzavreté. Iné možnosti:

1. Klient FIN, Server ACK, Server FIN, Klient ACK

2. Server FIN, Klient ACK, Klient FIN, Server ACK

3. Klient FIN, Server FIN ACK, Klient ACK

4. Server FIN, Klient FIN ACK, Server ACK

5. Server RES

6. Klient RES

Počas komunikácie sa u klienta(časovače vysielača) spúšťa viac druhov časovačov, ako napr. oneskorenie ACK timer, Fin\_Wait, ako aj vyššie spomenutý časovač na „testovanie nulového okna“, ako dlho má čakať na potvrdenie, aby prípadne odoslal segment znova.

Pri analýze tejto komunikácie sa teda sústredím na TCP hlavičku a rozpoznávanie príznakov(flagov), ako aj na IP adresy vysielacích či prijímacích uzlov.

**Externé súbory**

Využívam 1 externý súbor so známymi portami a protokolmi, ktoré načítavam do atribútov classy pre externý súbor. Po riadkoch na striedačku:

IPv4

0800

ARP

0806

ICMP

01

TCP

06

UDP

11

tftp

0045

http

0050

https

01BD

telnet

0017

ssh

0016

ftp-control

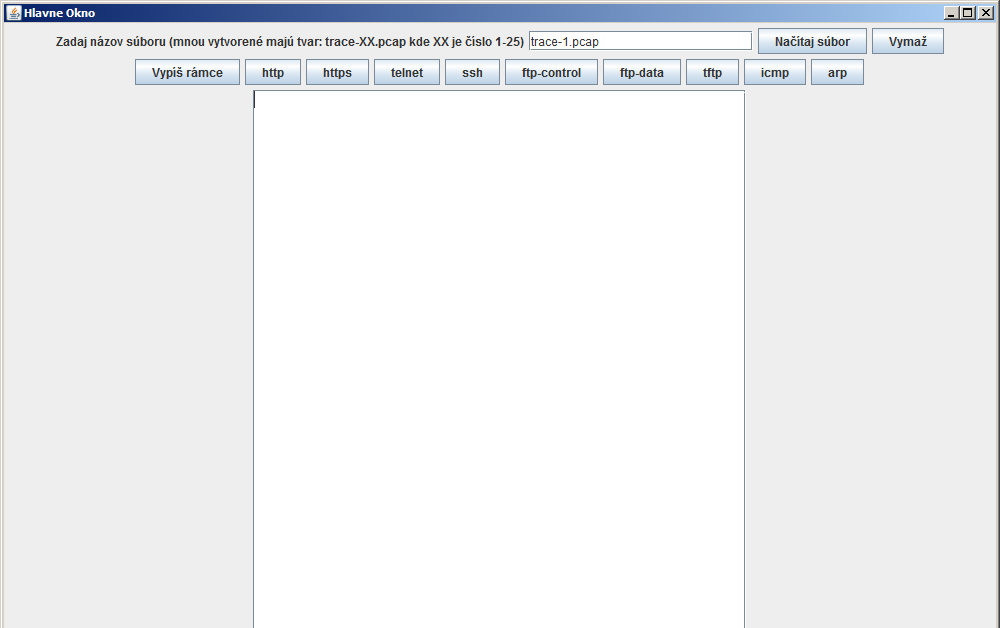
0015

ftp-data

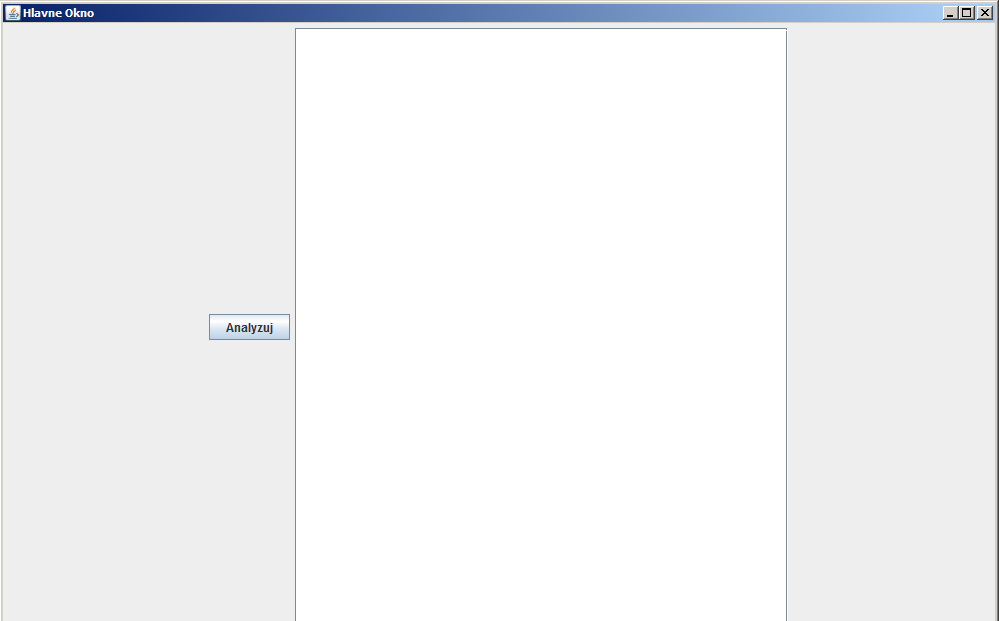
0014

**Používateľské rozhranie:**

V programe používam jednoduché GUI s pár buttonmi na základnú analýzu, zmazanie výstupu či iné potrebné výpisy podľa zadania úloh. Pre výpis použijem TextArea.



Po stlačení tlačidla „Načítaj súbor“:



**Implementačné prostredie**

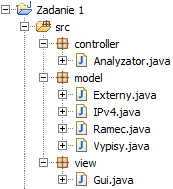
Pre implementáciu programu používam prostredie Eclipse v jazyku Java. Javu som zvolil pre jej pohodlnú prácu s objektami a classami, s jednoduchou tvorbou GUI. S programom som v minulosti celý semester pracoval, tak som sa rozhodol pri nom zostať.

Trochu k impementácii:

Ako návrhový vzor som zvolil MVC model (model – view – controller), kde controller je reprezentovaný main funkciou, funkciami pre GUI, a classou so vzorom Singleton (jedináčik). Využívam knižnucu JNetPcap na prácu s pcap súbormi.

Rámce načítavam pomocou funkcie nextEx(), premienam na hexadecimálny string cez funkciu getUByte() a string ukladám do atribútov objektov v liste Rámcov. Pre každý rámec si v atribútoch classy „Ramec“ ukladám ešte iné potrebné údaje, ako dĺžka rámca či použité protokoly. Tieto údaje neskôr dopĺňam či mením.

Základné rozdelenie class na ukážku MVC modelu:



**Zhrnutie**

K analyzovaniu rámcov využívam jazyk Java v prostredí Eclipse, 3 externý súbor na porty a protokoly. Načítaný vstup zo súboru pcap rozdelím na časti podľa hlavičiek a zistím všetky potrebné údaje ako sú MAC adresy, IP adresy či porty, a následne analyzujem komunikáciu medzi uzlami a rozpoznám fagy v TCP segmentoch. Všetky rámce ukladám do dátovej štruktúry ArrayList, a prejdením tohto listu vypíšem všetky potrebné informácie uložené v objektoch classy „Ramec“.

K úlohe 3 využívam pomocné ArrayListy, ktoré zvlášť analyzujem podľa daného protokolu či portu.