Slovenská technická univerzita
Fakulta informatiky a informačných technológii
Ilkovičova 3, 812 19, Bratislava

Komunikácia s využitím UDP protokolu

Počítačové a komunikačné siete

Ak. Rok 2016/17

Ondrej Harnúšek
ID: 79545

1. Zadanie

Nad protokolom UDP (User Datagram Protocol) transportnej vrstvy sieťového modelu TCP/IP navrhnite a implementujte program, ktorý umožní komunikáciu dvoch účastníkov v sieti Ethernet, teda prenos správ ľubovoľnej dĺžky medzi počítačmi (uzlami).

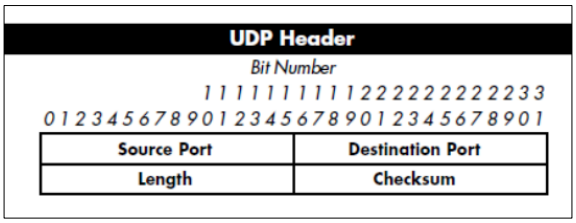
Program bude pozostávať z dvoch častí – vysielacej a prijímacej. Vysielací uzol pošle správu inému uzlu v sieti. Predpokladá sa, že v sieti dochádza k stratám dát. Vysielajúca strana rozloží správu na menšie časti - fragmenty, ktoré samostatne pošle. Správa sa fragmentuje iba v prípade, ak je dlhšia ako max. veľkosť fragmentu. Veľkosť fragmentu musí mať používateľ možnosť nastaviť aj menší ako je max. prípustný pre transportnú vrstvu.

Po prijatí správy na cieľovom uzle tento správu zobrazí. Ak je správa poslaná ako postupnosť fragmentov, najprv tieto fragmenty spojí a zobrazí pôvodnú správu.

Komunikátor musí vedieť usporiadať správy do správneho poradia, musí obsahovať kontrolu proti chybám pri komunikácii a znovuvyžiadanie rámca, vrátane pozitívneho/negatívneho potvrdenia. Pri nečinnosti komunikátor automaticky odošle paket pre udržanie spojenia. Odporúčame riešiť cez vlastne definované signalizačné správy.

2. Analýza

2.1. Enkapsulácia a hlavičky

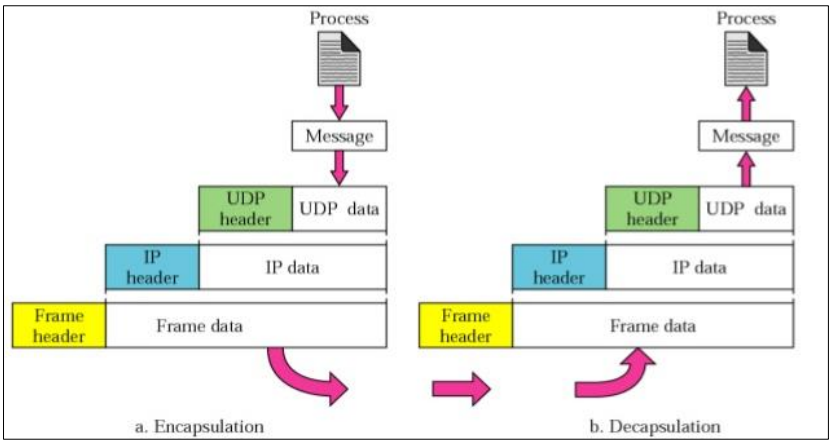


Obr. 1 [1]

Našou úlohou je navrhnúť program na protokolom UDP, ktorého hlavička je zobrazená na obr. 1. Jej veľkosť je 8B. UDP datagram je enkapsulovaný v dátovej časti IPv4 paketu, ktorého štandardná hlavička má veľkosť 20B.

Takže, praktický limit pre dĺžku dát je 65507B = (65535B − 8B UDP hlavička− 20B IPv4 hlavička).

Na obr. 2 je zachytený proces unkapsulácie počas odosielania správy a dekapsulácie počas jej prijímania.



Obr. 2 [2]

2.2. Hlavné vlastnosti protokolu UDP z pohľadu zadania

UDP protokol je tzv. "nespoľahlivý" protokol, ktorý nezabezpečuje mnohé vlastnosti, ktoré sú pre korektnú implementáciu nášho zadania potrebné:
• nezriaďuje spojenie pred prenosom dát
• nepotvrdzuje prijaté dáta
• nedeteguje straty
• nie je možnosť požadovať opakovanie prenosu dát
• negarantuje doručenie dát
• nezaručuje, že dáta sú prijímané v rovnakom poradí ako boli vyslané
• nemá mechanizmus na riadenie toku dát medzi koncovými uzlami resp. na riadenie zahltenia

2.3. Chyby v prenose a znovuvyžiadanie rámca

Chyby, ktoré môžu počas prenosu nastať sú kontrolované pomocou mechanizmu CRC (Cyclic Redundancy Checksum).

Po každom prijatom rámci prijímateľ skontroluje správnosť rámca (jeho index a checksum) a odošle pozitívne (ACK) alebo negatívne (NAK) potvrdenie prijatia. Pri negatívnom potvrdení odošle daný rámec znovu.

V prípade, že odosielateľovi neprišlo do stanoveného časového limitu žiadne potvrdenie, pošle takisto rámec znova.

2.4. Veľkosť fragmentu

Správy, ktoré presahujú maximálnu veľkosť fragmentu je potrebné pri odosielaní fragmentovať a pri prijímaní defragmentovať.

Maximálnu veľkosť fragmentu nastavuje odosielateľ. Tá však nemôže presiahnuť veľkosť dátovej časti UDP protokolu.

3. Špecifikácia požiadaviek

Program bude implementovaný v jazyku JAVA, pomocou knižníc na prácu s UDP datagrami java.net a java.io. Kontrolný súčet bude generovaný pomocou triedy CRC32, ktorá používa generujúci polynóm: `x^32 + x^26 + x^23 + x^22 + x^16 + x^12 + x^11 + x^10 + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1`

Program je navrhnutý pre dva typy používateľov:

1) **Odosielateľ** - Vysielací uzol pošle správu inému uzlu v sieti.
   - Určuje cieľovú IP adresu a port.
   - Nastavuje max. veľkosť fragmentu.
   - Má možnosť odoslať chybný rámec.
   - Môže odoslať textovú správu, alebo súbor z priečinku, v ktorom je program..

2) **Prijímateľ** - Prijímací uzol zachytí správu a odošle potvrdenie.
   - Nastavuje port.
   - Pri prijatí textovej správy sa mu zobrazí.
   - Pri prijatí súboru program zobrazí jeho meno a súbor uloží do priečinku, v ktorom je program.

4. Návrh riešenia

4.1. Návrh vlastného protokolu

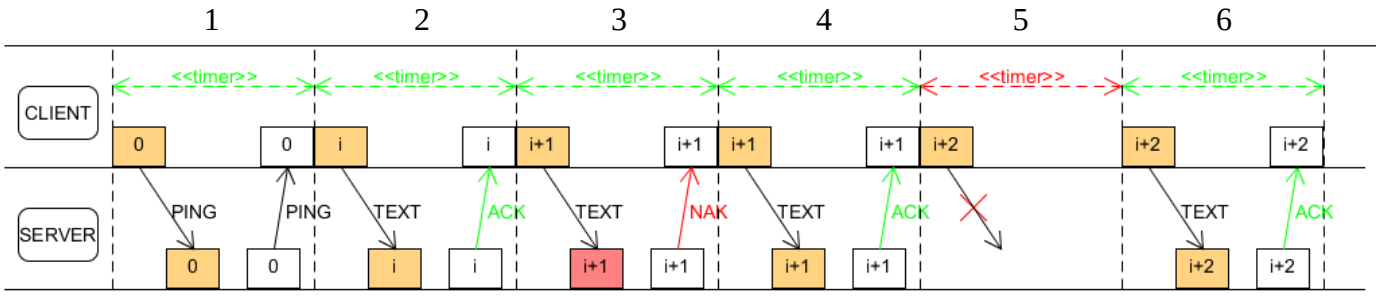
Požiadavky na spoľahlivosť komunikačného systému splníme definovaním vlastného protokolu, ktorý bude enkapsulovaný v dátovej časti UDP protokolu. Jej veľkosť môže byť max 65507B.

Index of Packet	Count of Packets	Flags	Length	Data	CheckSum
4B	*4B*	*1B*	*2B*	*0B – 65 492B*	*4B*

Popis hlavičky nášho protokolu:

Index of Packet		poradové číslo fragmentu
Count of Packets		počet fragmentov
Flags	1= TEXT	identifikuje textovú správu
	2= FILE	identifikuje súbor
	4= ACK	identifikuje pozitívne potvrdenie
	8= NAK	identifikuje negatívne potvrdenie
	16= PING	identifikuje paket pre udržanie spojenia
Length		dĺžka dátovej časti
Data		dátová časť
CheckSum		kontrolný súčet

4.2. Diagram posielania a príjmu datagramov



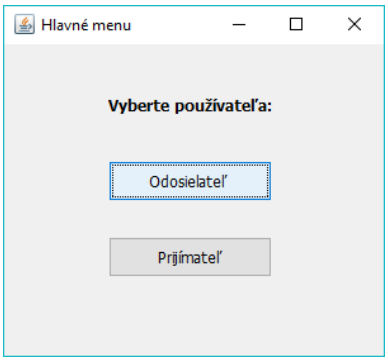
Obr. 3

Na obr.3 je znázornený proces posielania 6 datagramov.

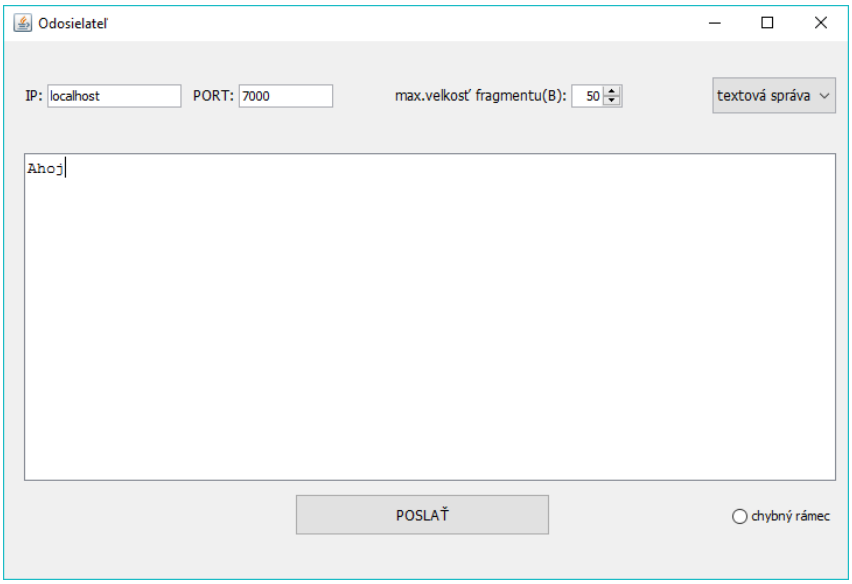
1) Poslanie PING správy, odpoveď PING správa
2) Poslanie i-teho fragmentu TEXT správy, pozitívne potvrdenie
3) Poslanie i+1 fragmentu TEXT správy, negatívne potvrdenie
4) Opätovné poslanie i+1 fragmentu TEXT správy, pozitívne potvrdenie
5) Poslanie i+2 fragmentu TEXT správy, žiadna odpoveď (vypršanie času)
6) Opätovné poslanie i+2 fragmentu TEXT správy, pozitívne potvrdenie

Datagramy, ktoré nie sú správne (zlý index fragmentu alebo checksum) prijímateľ zahadzuje.
Opätovné poslanie sa opakuje maximálne 5 krát – po týchto pokusoch sa vypíše chybová hláška.

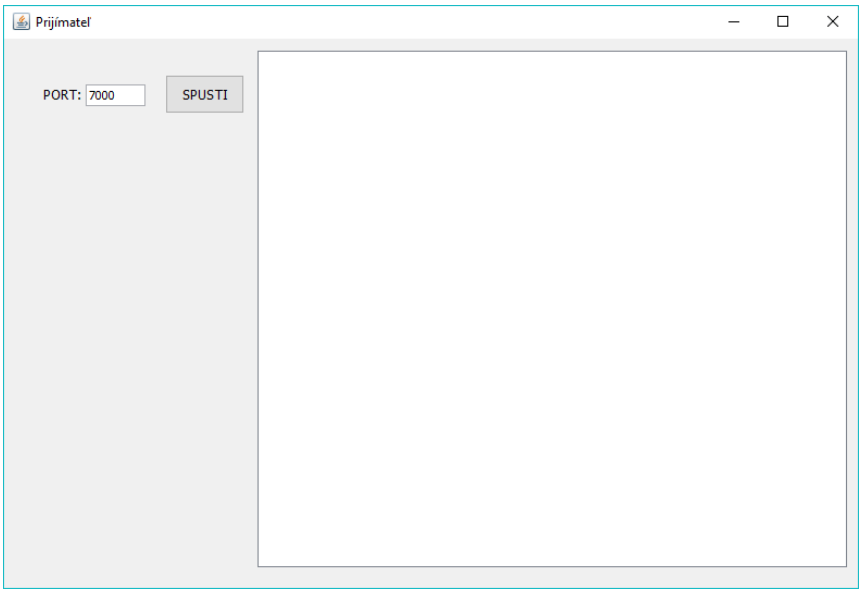
4.3. Používateľské rozhranie



Hlavné menu – možnosť výberu používateľa.



Okno pre odosielateľa. Nastavenie IP, portu, max. veľkosti fragmentu a typu správy. Možnosť odoslať chybný rámec.



Okno pre prijímateľa. Nastavenie portu na ktorom počúva.

5. Implementácia

5.1. Zmeny oproti návrhu

Maximálna veľkosť fragmentu je 65507B a minimálna 16B (čo znamená len 1B pre dátovú časť navrhnutého protokolu). Veľkosť poľa Length v hlavičke navrhnutého protokolu je 16 bitov.

5.2. Použité balíky a triedy

Program je implementovaný v jazyku JAVA. Použitie balíkov a tried:

- **java.net**: *DatagramSocket*, *DatagramPacket* -práca s UDP datagramom
- **java.nio**: *ByteBuffer* -práca pri hlavičky a enkapsulácii
- **java.io**: *FileInputStream*, *FileOutputStream* -práca so súborom
- **java.util.zip**: *CRC32* -generovanie kontrolného súčtu

5.3. Organizácia programu

Program je organizovaný do troch balíkov:

- **GUI** – *HlavneOkno* - obsahuje main(), *OdosielatelOkno*, *PrijimatelOkno* (viď 5.5)

- **comm.sender** – obsahuje triedy, ktoré používa Odosielateľ:
  - *Client* - obsahuje informácie o príjemcovi (adresa, port). Na začiatku sa otvorí soket. Funkcia control() uloží premenné a vytvára objekt *Send*.
  - *Send* - posiela dva typy správ: TEXT/FILE. Pričom pri posielaní súboru pošle najskôr jeho meno vo forme správy TEXT a následne samotný súbor vo forme FILE.
    -podľa typu správy vytvára hlavičku(4.1) a fragmentuje správu. Postupne odosiela fragmenty a po každom príjíma potvrdenie.
  - *Ping* - v novom vlákne posiela každých 30sec správu PING a prijíma odpoveď.

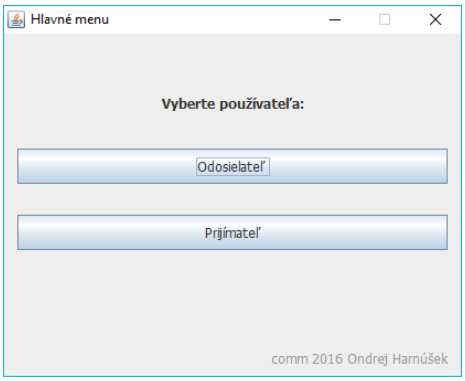
- **comm.receiver** – obsahuje triedy, ktoré používa Prijímateľ:
  - *Server* - obsahuje port na ktorom počúva. Funkcia control() uloží premenné a vytvára nové vlákno *Receive*.
  - *Receive* - otvorenie soketu a neustále prijímanie paketov:
    0. Nová správa = 0 fragmentov
    1. Prijatie i-teho fragmentu
    2. Prečítanie hlavičky
    3. Výpočet kontrolného súčtu
    4. Ak je správa PING - odoslanie odpovede. →exit();
    5. Ak sedí kontrolný súčet a index fragmentu – odoslanie ACK. Inak NAK. →exit();
    6. Uloženie fragmentu do poľa bytov.
    7. Ak to je posledný fragment, tak podľa typu TEXT/FILE výpis alebo uloženie. →bod 0.

5.4. Zámerne chybná správa

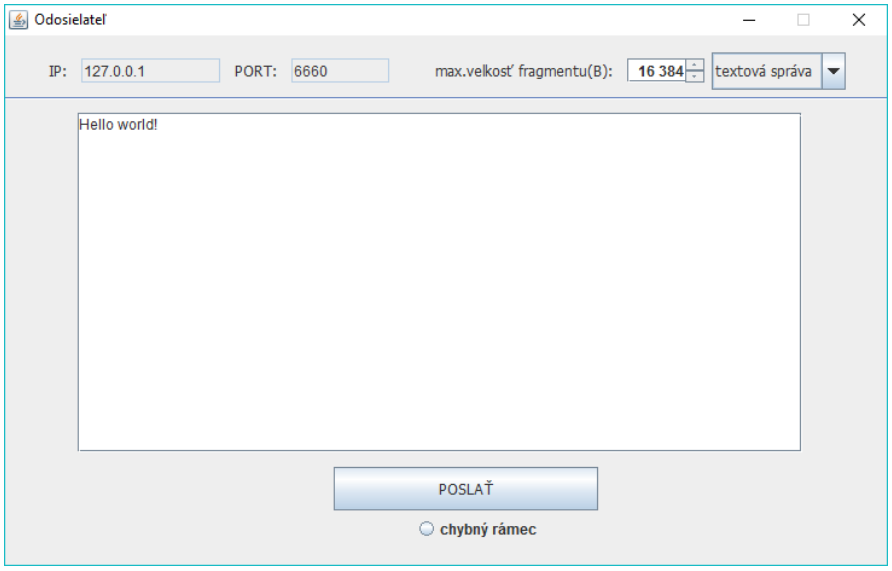
Ak je vyznačený checkbox „chybný rámec“(t.j. boolean jeChybny=true), tak v prvom fragmente správy sa pred odoslaním rámca prepíše pole hlavičky Flags na 0. boolean jeChybny sa vynuluje.

Takže keď server vypočíta kontrolný súčet nesedí s tým v hlavičke, odošle NAK klientovi a ten odošle fragment znovu.

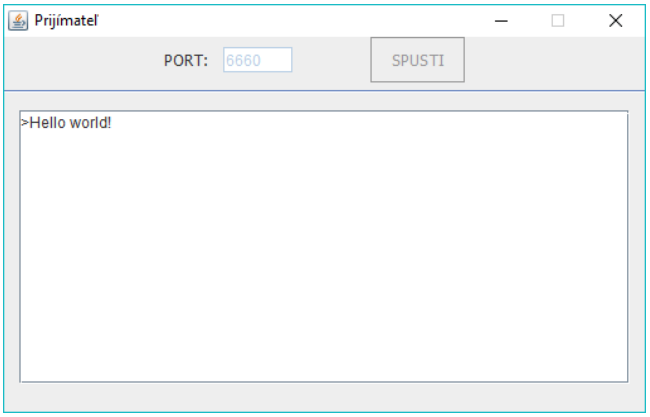
5.5. Používateľská príručka - GUI



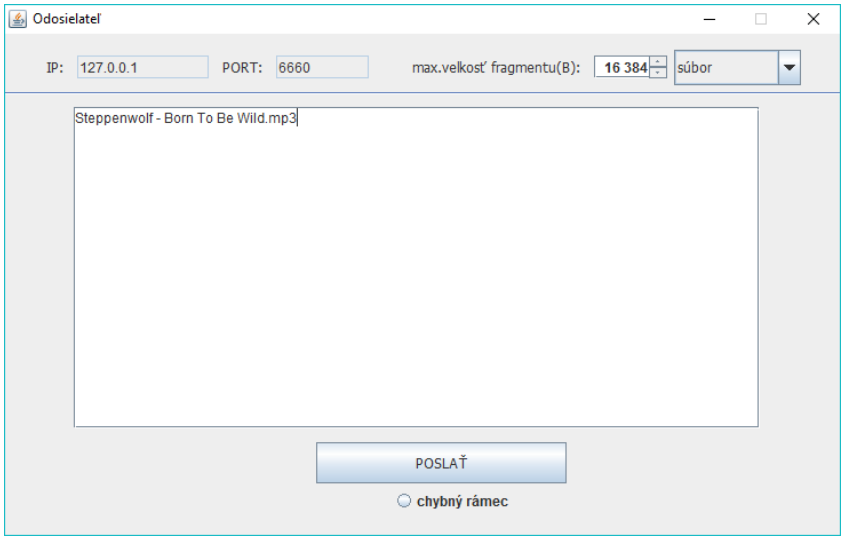
Hlavné okno – zvolenie typu užívateľa



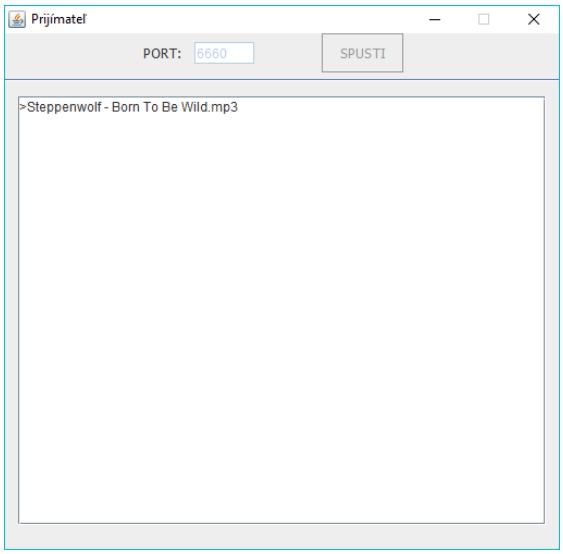
Okno Odosielateľ – vyplnenie polí
odoslanie textovej správy



Okno Prijímateľ – zadanie portu a spustenie servera
Zobrazenie prijatej textovej správy



Okno Odosielateľ – odoslanie súboru
Súbor je umiestnený v rovnakom priečinku ako program. Do poľa sa napíše jeho meno.prípona



Okno Prijímateľ – zobrazenie názvu prijatého súboru
Súbor je uložený do rovnakého priečinka ako program.

6. Zhodnotenie

Program vďaka návrhu vlastného protokolu umožnuje spoľahlivú komunikáciu medzi dvomi účastníkmi, pričom pracuje nad protokolom UDP transportnej vrstvy sieťového modelu TCP/IP.

Pri odosielaní správu fragmentuje a pri jej prijímaní ju defragmentuje. Dokáže kontrolavať, či nedošlo k poškodenie dát a požiadať o opätovné odoslanie. Pri nečinnosti komunikátor automaticky odošle paket pre udržanie spojenia.

Odosielateľ má možnosť odosielať textové správy ako aj súbor. Má možnosť vytvoriť a odoslať zámerne chybnú správu. Dokáže nastaviť maximálnu veľkosť fragmentu.

7. Bibliografia

[1] SANS Technology Institute: POCKET REFERENCE GUIDE, https://www.sans.org/security-resources/tcpip.pdf

[2] Jamal:. Encapsulation and decapsulation TCP/IP Protocol, http://www.slideshare.net/noctorouskhan/chap-11-udp