SLOVENSKÁ TECHNICKÁ UNIVERZITA

Fakulta informatiky a informačných technológií

v Bratislave

Počítačové a komunikačné siete

Zadanie 2

Dokumentácia

Adam Tomčala

Prednášajúci: prof. Ing. Ivan Kotuliak, PhD.

Cvičiaci: Ing. Pavol Helebrandt, PhD.

Čas cvičení: Štvrtok 8:00

Adam Tomčala
ID: 103164

Zadanie 2: Komunikácia s využitím UDP protokolu

Zadanie úlohy

Navrhnite a implementujte program s použitím vlastného protokolu nad protokolom UDP (User Datagram Protocol) transportnej vrstvy sieťového modelu TCP/IP. Program umožní komunikáciu dvoch účastníkov v lokálnej sieti Ethernet, teda prenos textových správ a ľubovoľného súboru medzi počítačmi (uzlami).

Program bude pozostávať z dvoch častí – vysielacej a prijímacej. Vysielací uzol pošle súbor inému uzlu v sieti. Predpokladá sa, že v sieti dochádza k stratám dát. Ak je posielaný súbor väčší, ako používateľom definovaná max. veľkosť fragmentu, vysielajúca strana rozloží súbor na menšie časti - fragmenty, ktoré pošle samostatne. Maximálnu veľkosť fragmentu musí mať používateľ možnosť nastaviť takú, aby neboli znova fragmentované na linkovej vrstve.

Ak je súbor poslaný ako postupnosť fragmentov, cieľový uzol vypíše správu o prijatí fragmentu s jeho poradím a či bol prenesený bez chýb. Po prijatí celého súboru na cieľovom uzle tento zobrazí správu o jeho prijatí a absolútnu cestu, kam bol prijatý súbor uložený.

Program musí obsahovať kontrolu chýb pri komunikácii a znovu vyžiadanie chybných fragmentov, vrátane pozitívneho aj negatívneho potvrdenia. Po prenesení prvého súboru pri nečinnosti komunikátor automaticky odošle paket pre udržanie spojenia každých 5-20s pokiaľ používateľ neukončí spojenie. Odporúčame riešiť cez vlastne definované signalizačné správy.

Adam Tomčala
ID: 103164

Základné informácie:

- **Programovací jazyk:**
  - Použil som jazyk Python 3.9.
- **Použité knižnice:**

```
import socket
import os
import math
import struct
import binascii
import threading
import time
```

- Knižnicu **socket** som použil pri práci so soketmi prijímača a vysielača.
- Použil som z nej nasledujúce funkcie:

socket()	vytvorenie jednotlivých soketov
bind()	funkcia viaže soket s adresou (IP adresa, port)
sendto()	funkcia slúži na odoslanie dát z jedného uzla do druhého
recvfrom()	funkcia slúži na prijatie dát
settimeout()	funkcia nastavý časový limit pre blokovanie operacií soketu

- Knižnicu **os** som použil pri práci zo súborom. Použil som z nej nasledujúce funkcie:

path.isfile()	funkcia zisťuje či cesta vedie k existujúcemu súboru
path.basename()	funkcia z cesty zistí meno súboru
path.getsize()	funckia zisťuje veľkosť súboru
path.abspath()	funkcia zistí absolútnu cestu k súboru
path.urandom()	Funkcia vytvára náhodnú postupnosť byte-ov, podľa zadanej dĺžky (využil som ju pri generovaní chyby).

- Z knižnice **math** som použil funkciu **math.ceil()**, ktorú využívam pri počítaní počtu fragmentov, ktorú budem musieť odoslať. Zaokrúhľuje výsledok smerom nahor.
- Knižnicu **struct** som použil pri práci s dátami v hlavička. Knižnica slúži na konverziu medzi Python a C dátovými typmi. Využil som tieto 2 funkcie:

struct.pack()	funkcia vytvára bytes object s “zapackovaným” dátami (packoval som **int** na **unsigned int** kvôli šetreniu miesta)
struct.unpack()	funkcia “unpackuje” tieto data

- Z knižnice **binascii** som použil funkciu **crc_hqx()** na vytvorenie kontrolnej sumy z dát.

- Knižnicu **threading** som použil na vytvorenie objektu **Thread()**. Nový thread som vytváral kvôli implementovaniu keep-alive správ.
- Z knižnice **time** som použil funkciu **sleep(),** ktorá zastaví vykonávanie behu programu. Využil som to pri posielaní keep-alive správ.

Popis prijímateľa:

- Potom ako si používateľ vyberie, že bude pokračovať ako prijímateľ musí zadať **číslo portu**, na ktorom bude počúvať.
- Následne prijímateľ čaká na inicializačnú správu, ktorá značí začiatok komunikácie, správa s typom nastaveným na „**I**“ (inicializácia). Po prijatí tejto správy prijímač odpovedá na inicializačný paket, posiela správu s typom nastavený na „**0**“ (potvrdenie).
- Ak po zapnutí prijímača nepríde žiadna inicializačná správa do **20 sekúnd,** prijímač sa automaticky vypne.
- Hneď po prijatí inicializačného paketu bude prijímač dostávať každých **5 sekúnd** keep-alive správy. Na tieto správy bude prijímač odpovedať potvrdzovacou správou (typ bude nastavený na hodnotu **„6“**).
- Následne bude prijímač očakávať správu, ktorej obsahom bude to, aký druh správy sa bude odosielať.
- Ak príde správa s typom nastaveným na „**T**“, tak sa bude odosielať text.
- Ak príde správa s typom nastaveným na „**F**“, tak sa bude odosielať súbor. V ďalšej časti správy bude uložený názov súboru + prípona. Následne bude musieť zadať miesto, kde sa daný súbor uloží a až potom odošle potvrdzovaciu správu.
- Po oboch typoch týchto správ sa odošle potvrdzovacia správa s typom nastaveným na „**0**“ (potvrdenie).
- Keďže som si vybral **Stop and Wait ARQ** metódu, prijímateľ čaká, kým k nemu nedorazí práve 1 paket. Po obdržaní paketu musí prijímateľ vypočítať **checksum** obdržaného paketu a porovná ho s **checksum**-om v hlavičke. Ak sa zhodujú, znamená to, že prišli správne dáta a pošlem potvrdzovaciu správu (typ nastavený na hodnotu **„2“** čo znamená správne prijatie dát). Ak sa nezhodujú, znamená to, že sme neobdržali správne dáta. V tom prípade prijímateľ odošle správu so žiadosťou o opätovné odoslanie dát (typ nastavený na hodnotu **„3“**).
- Ak sa už odoslali všetky pakety dát, prijímač dostáva správu s typom nastaveným na „**4**“. Následne prijímač odpovedá potvrdzovacou správou (typ nastavený na hodnotu **„5“**).
- Po ukončení prenosu dát bude prijímač prijímať keep-alive správy, na ktoré bude odpovedať rovnako ako predtým.
- Ak vysielač už nebude chcieť odosielať ďalšie dáta, prijímač dostane správu o ukončení komunikácie (typ nastavený na hodnotu **„E“,** ukončenie komunikácie). Následne sa prijímač vypne a program sa dostane do hlavného menu.

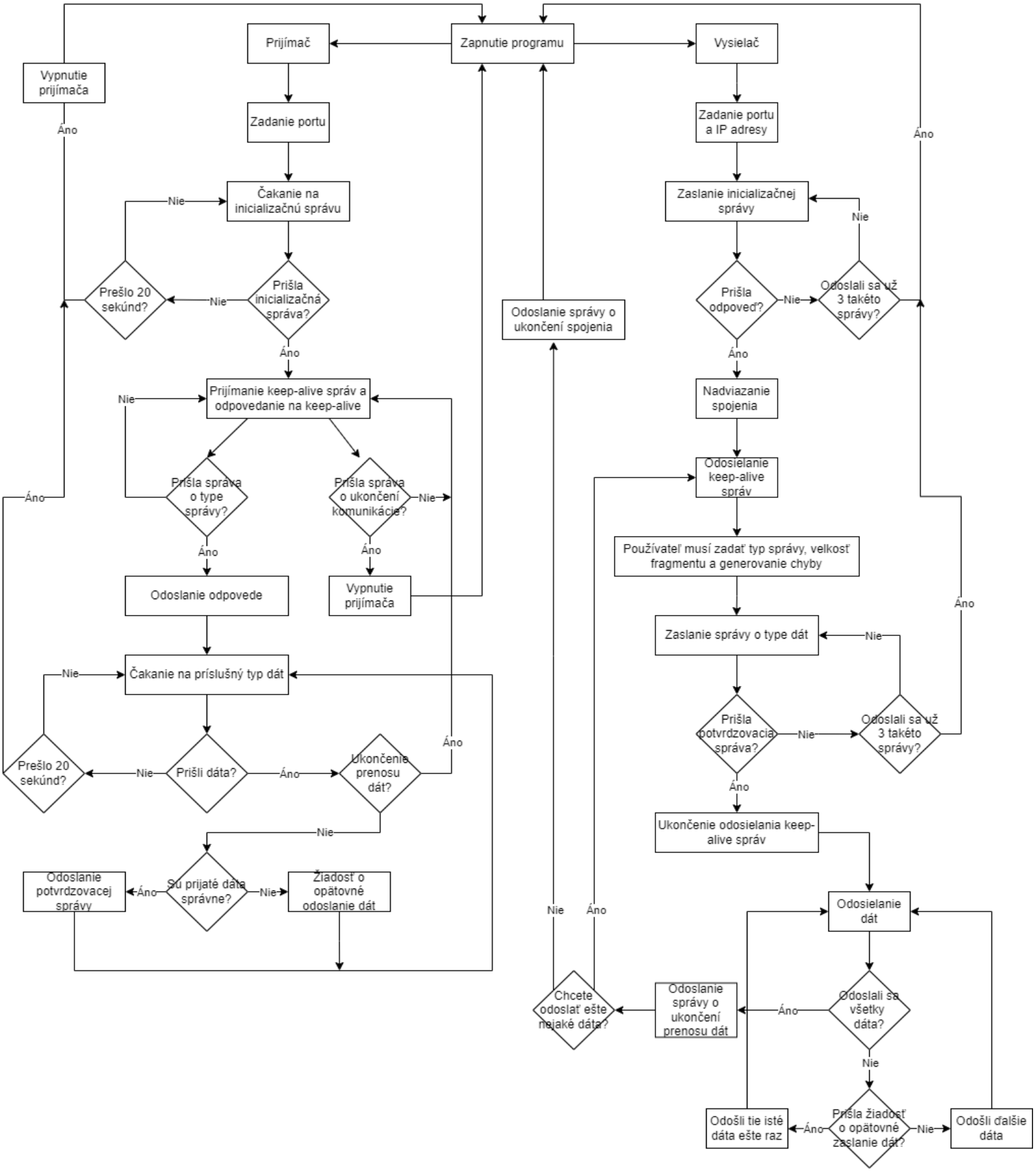
Adam Tomčala
ID: 103164

Popis vysielača:

- Potom ako si používateľ vyberie, že bude pokračovať ako vysielač musí zadať **číslo portu**, na ktorom bude prijímateľ počúvať. Taktiež je potrebné zadať aj **IP adresu** tohto prijímateľa.
- Ako náhle sa zadajú tieto informácie, vysielač pošle inicializačný paket (typ nastavený na hodnotu **„I“**). Následne čaká na potvrdenie.
- Po potvrdení sa zahájі komunikácia a vysielač bude v intervale každých **5 sekúnd** posielať keep-alive správy.
- Po nadviazaní spojenia si používateľ bude môcť vybrať či chce odoslať dáta, zmeniť role alebo ukončiť program.
- Ak si vyberie, že chce odoslať dáta, vyberie typ správy, ktorý bude chcieť poslať:
    - *Textová správa*:
        - Text zo štandardného vstupu.
    - *Súbor*
        - Používateľ zadá cestu k súboru a overí sa či súbor existuje
- Potom používateľ bude musieť zadať veľkosť fragmentu v intervale od **1-1463 B**
  (1500 – IPv4 hlavička – UDP hlavička – moja hlavička (5B) = 1463).
- Následne si používateľ vyberie či chce simulovať chybu.
- Po zadaní týchto informácií sa odošle správa vysielaču so správou, aký typ správy má očakávať. Ak sa odosiela text, odošle sa správa s typom nastaveným na hodnotu **„T“**. Inak sa odošle správa s typom nastaveným na hodnotu „**F**“ **+ meno súboru.**
- Ak príde potvrdenie, začnú sa posielať dáta, typ bude nastavený na hodnotu **„1“**
- Ak vysielač obdrží správu s obsahom **„2“**, odosielajú sa ďalšie fragmenty. Ak obdrží správu s obsahom **„3“**, príslušný paket sa odošle ešte raz.
- Ak počas odosielanie dát vysielač nedostane potvrdzovaciu správu do **5 sekúnd**, príslušný paket sa odošle ešte raz. Ak nepríde potvrdzovacia správa 3krát, vysielač sa ukončí a program skočí do hlavného menu.
- Ak sa už odoslali všetky fragmenty správne, odošle sa správa s typom nastaveným na hodnotu **„4“** (ukončenie prenosu dát). Ak príde potvrdenie, obsah správy bude **„5“**, začnú sa opäť posielať keep-alive správy v intervaloch každých **5 sekúnd** pre udržanie spojenia.
- Následne si používateľ bude môcť vybrať či chce ešte odoslať nejaké dáta, zmeniť rolu alebo ukončiť program.
- Ak si vyberie odosielanie dát, program bude pokračovať rovnako od zadávania typu správ.
- Ak si vyberie zmenenie rolí alebo ukončenie programu, skončí sa odosielanie keep-alive správ a zašle sa práva o ukončení komunikácie (typ nastavený na hodnotu **„E“**).
- Pri zmene rolí sa program vráti do hlavného menu a používateľ si môže vybrať ako bude pokračovať.
- Ak sa všetky dáta úspešne prenesú, odosielateľ pošle správu ukončení prenosu dát a očakáva potvrdzovaciu správu.

Adam Tomčala
ID: 103164

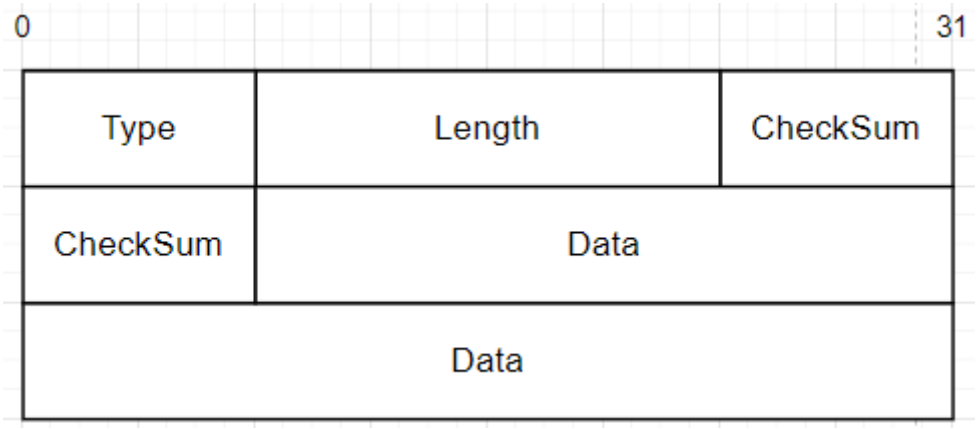
Diagramy programu:



Adam Tomčala
ID: 103164

Navrhnutá hlavička:

- V mojej implementácií budem používať dva druhy hlavičiek podľa toho, aký druh správy sa bude odosielať.
- Rôzne hlavičky budem používať preto, lebo pri správach určitého typu budú určité informácie zbytočné.
- **Hlavička na prenos dát** bude vyzerať nasledovne:



- **Popis jednotlivých hodnôt:**

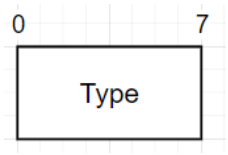
Type (1B)	Určuje typ danej správy
Length (2B)	Určuje veľkosť prenášaných dát
CheckSum (2B)	Kontrolná suma

- **Popis hodnôt, ktoré môže nadobúdať políčko Type:**

I	Inicializačná správa
T	Správa, ktorá hovorí, že sa bude odosielať text
F	Správa, ktorá hovorí, že sa bude odosielať súbor
K	Keep-alive správa
1	Prenos dát
2	Potvrdenie správneho prijatia dát
3	Žiadosť o znovu odoslanie dát
4	Ukončenie prenosu dát
5	Potvrdenie ukončenia prenosu dát
6	Potvrdenie o prijatí keep-alive správy
0	Potvrdzovací typ správy
E	Ukončenie komunikácie

- Celková veľkosť hlavičky je **5B**.

- **Hlavička, ktorá nebude určená na prenos dát** budú vyzerať nasledovne**:**



Začiatok komunikácie:

- Pri inicializačnej správe pošle vysielač prijímateľovi správu, ktorá bude mať políčko **Type** nastavené na **„I“.**
- Potom čo správa dorazí k prijímateľovi, prijímateľ odošle potvrdenie o nadviazaní spojenia a políčko **Type** bude nastavené na **„0“** (začnú sa posielať **keep-alive správy**)**.**
- Následne vysielač odošle správu, ktorá bude obsahovať typ prenášanej správy, ak to bude text odošle sa správa, ktorá bude mať políčko **Type** nastavené na **„T“.** Ak sa bude prenášať súbor, **Type** bude nastavený na hodnotu **„F“** + odošle sa aj názov súboru s príponou.
- Potom čo správa dorazí k prijímateľovi, prijímateľ odošle potvrdenie o začatí komunikácie (prenosu dát) a políčko **Type** bude nastavené na **„0“** (zastaví sa odosielanie **keep-alive správ**)**.**

Potvrdenie prijatia dát:

- Potom čo dorazia dáta k prijímačovi nasleduje kontrola dát (checksum).
- Ak sú dáta správne (checksum sa zhoduje), odošle sa potvrdenie o tom, že dáta sú správne. Políčko **Type** bude nastavené na hodnotu **„2“**.
- Ak dáta nebudú správne (checksum sa nezhoduje), odošle sa potvrdenie o tom, že dáta sú nesprávne. Políčko **Type** bude nastavené na hodnotu **„3“**.

Ukončenie prenosu dát:

- Po skončení prenosu dát vysielač odošle správu prijímaču s informáciou o ukončení prenosu dát. Políčko **Type** bude nastavené na hodnotu **„4“**.
- Potom čo prijímač dostane túto správu, odošle sa z prijímača potvrdenie konca prenášania dát. Políčko **Type** bude nastavení na hodnotu **„5“** (opäť sa začnú posielať **keep-alive správy** pre udržanie spojenia)**.**
- Ak používateľ zvolí ďalší prenos dát, bude musieť zadať potrebné informácie, ale spojenie už bude nadviazané.
- Ak používateľ už nebude chcieť odosielať ďalšie dáta, pošle sa správa a políčko **Type** bude nastavené na hodnotu **„E“** (ukončí sa spojenie).

Keepalive správa:

- Pri keepalive správe bude políčko **Type** nastavené na hodnotu **„K“.**
- Pri potvrdení keepalive správy bude políčko **Type** nastavené na hodnotu **„6“.**

ARQ:

- Pre detekciu chýb pri prenose dát použijem metódu **Stop and Wait ARQ**.
- Táto metóda sa využíva pri obojstrannej komunikácií (half-duplex) a spočíva v tom, že odosielateľ neodosiela ďalšie pakety (dáta) dovtedy, dokým mu nepríde potvrdzovacia správa o správnom obdržaní dát.
- Odosielateľ si taktiež musí udržiavať kópiu tohto paketu, pretože ak prijímateľ nedostane správne dáta alebo odosielateľ nedostane správu o potvrdení do určitého času, musí odosielať paket dovtedy, dokým sa paket neodošle správne.
- V mojom programe som to implementoval tak, že ak príde správa o správnom prijatí dát, idú sa odosielať ďalšie časti dát. Inak sa príslušný paket bude odosielať, až kým sa správne nedoručí.
- Ak vysielaču nepríde nejaká potvrdzovacia správa do **5 sekúnd**, pošle sa príslušný paket ešte raz. Ak nepríde odpoveď na 3 takéto správy za sebou, vysielač prestáva posielať dáta (vypíše sa chybová hláška, že neprišla odpoveď) a program skočí do hlavného menu.

Kontrolná suma (checksum):

- Kontrolná suma je mechanizmus, ktorý slúži na ochranu a kontrolu dátového bloku počas jeho prenosu. Túto hodnotu si vypočítam pomocou funkcie **crc_hxq()** z knižnice **binascii**.
- Táto funkcia vracia vypočítanú CRC hodnotu, funkcia funguje nasledovne:
- Funkcia používa **CRC-CCITT** polynóm, ktorý vyzerá nasledovne: $\mathbf{x^{16} + x^{12} + x^5 + 1.}$
- Polynóm si môžeme reprezentovať ako postupnosť bitov, počet bitov bude 16 + 1 = 17.
- **1 0001 0000 0010 0001** – polynóm $\mathbf{x^{16} + x^{12} + x^5 + 1.}$
- Na výpočet CRC hodnoty potrebujeme tento polynóm a dáta ako postupnosť bitov.
- K týmto dátam pridám na koniec 16 núl (veľkosť CRC generátora/polynómu - 1).
- Následne sa vykonáva operácia **XOR** medzi dátami (zo vstupných dát zoberiem toľko bitov zľava, aká je veľkosť polynómu) a polynómom. Začína sa zľava.
- Ak zľava vzniknú nuly, odstránim ich a zoberiem z dát toľko bitov, koľko som odstránil núl zľava (aby bol počet bitov z dát rovnaký ako veľkosť polynómu) a pokračujem s vykonávaním **XOR** operácie.
- Vykonáva sa to až kým sa nedôjde nakoniec.
- Táto funkcia sa zavolá na strane vysielača, pred odoslaním paketu.

- Túto CRC hodnotu v mojom program rátam z dát + hlavička (okrem samotnej hodnoty CRC, ktorú potom pridám na koniec hlavičky).
- Po prijatí dát na strane prijímača sa opäť zavolá táto funkcia sa, vypočíta sa hodnota CRC.
- Ak sa vypočítaná hodnota bude zhodovať s hodnotou v hlavičke, dáta sa prijali správne a pošle sa správa o správnom prijatí dát.
- Ak sa vypočítaná hodnota nebude zhodovať s hodnotou v hlavičke, pošle sa žiadosť o opätovné odoslanie príslušného paketu.

Keepalive správy:

- Keep alive je typ správy, ktorý bude posielať vysielač na udržanie spojenia.
- V mojom programe som keep-alive správy implementoval pomocou vytvorenia nového **threadu**.
- Ako náhle prijímač a vysielač nadviažu spojenie, posielajú sa z druhého vlákna (z vysielača) keep-alive správy každých **5 sekúnd.**
- Prijímač na tieto správy odpovedá potvrdzovacou správou.
- Keep-alive správy sa odosielajú, od nadviazania spojenia do začatia prenosu dát (medzi tým používateľ na strane vysielača musí vyplniť potrebné informácie pre odoslanie príslušných dát).
- Hneď ako sa začnú posielať dáta, toto vlákno zaniká.
- Ak nepríde odpoveď na 3 keep-alive správy (prijímač sa pravdepodobne vypol) preruší sa spojenie a ukončí sa komunikácie a program skočí do hlavného menu.
- Po skončení prenosu dát sa používateľ na strane vysielača bude musieť rozhodnúť ako bude chcieť pokračovať (ďalší prenos dát, zmena rolí alebo ukončenie programu). Medzi tým sa opäť vytvorí vlákno na odosielanie keep-alive správ, kým sa používateľ nerozhodne ako pokračovať.
- Ak si vyberie posielanie ďalších dát, keep-alive správy sa budú odosielať až do začatie odosielania dát.
- Ak si vyberie inú možnosť ukončí sa spojenie a vlákno pre keep-alive správy zanikne.

Simulácia chyby:

- V mojom programe si používateľ musí určiť či chce generovať chybu. Ak zadá, že chce chybu generovať, chyba sa vygeneruje len v prvom pakete dát.
- Chybu generujem pomocou funkcie **os.urandom()** z knižnice **os**, ktorá generuje náhodnú postupnosť byte-ov určitej veľkosti.
- Veľkosť chybných dát závisí od veľkosti aktuálne prenášaných dát.
- Po odoslaní chybných dát si prijímač vyžiada znovu odoslanie príslušného paketu.
- Následne pošlem už správne dáta a pokračujem v ďalšom odosielaní bez generovania chyby.

Adam Tomčala
ID: 103164

Popis jednotlivých častí zdrojového kódu:

- V mojom zdrojovom kóde som dôkladne okomentoval jednotlivé časti kódu a vytvoril som ku každej funkcií DocString.

Dôležité časti kódu a popis fungovania programu:

- Po spustení programu si používateľ vyberie rolu (funkcia main):

```
Zapnutie programu ako vysielac -> zadajte 1.
Zapnutie programu ako prijimac -> zadajte 2.
Zadajte hocico ine pre ukoncenie programu.
Zadajte vstup:
```

- Ak si používateľ vyberie pokračovať ako vysielač spustí sa funkcia:

```
def vysielac_login():
    """
    Funkcia zisti od pouzivatela port a IP adresu prijimatela a nasledne ide inicializacia.
    :return:    Nic.
    """
    cislo_portu = int(input("Zadajte cislo portu prijimatela: "))

    ip_adresa = input("Zadajte IP adresu prijimatela: ")

    inicializacia(cislo_portu, ip_adresa)
```

- Ak si používateľ vyberie pokračovať ako prijímač spustí sa funkcia:

```
def prijimac_login():
    """
    Funkcia sluzi na vytvorenie uzla na prijimanie sprav.
    :return:    Nic.
    """
    port = int(input("Zadajte port: "))

    socket_2 = socket.socket(family=socket.AF_INET, type=socket.SOCK_DGRAM)
    socket_2.bind(("", port))
```

- Funkcia inicializácia slúži na zahájenie spojenia medzi uzlami:

```
print("Odosielanie inicializacneho paketu serveru...")

socket_1 = socket.socket(family=socket.AF_INET, type=socket.SOCK_DGRAM)
adresa = (ip, port)

socket_1.sendto(str.encode("I"), adresa)
socket_1.settimeout(5)

data, adresa = socket_1.recvfrom(1500)
data = data.decode()
```

  - Odoslanie inicializačného paketu a čakanie na odpoveď.

- Ak príde potvrdenie vytvára sa vlákno na keep – alive správy:

```
if data == "0":
    print("Zahajenie komunikacie.")
    # Ak sa zahaji komunikacia, vytvaram novy thread na keep-alive spravy.
    global je_thread
    je_thread = True
    keep_vetva = threading.Thread(target=keep_alive, args=(socket_1, adresa))
    keep_vetva.start()
```

- Následne si používateľ vyberie ako pokračovať ďalej:

```
Odosielanie inicializacneho paketu serveru...
Zahajenie komunikacie.

Chcete odoslat data? -> 1
Chcete ukoncit komunikaciu a zmenit role? -> 2
Chcete ukoncit program? -> 3
Zadajte vstup:
```

- Medzi tým prijímač už bude dostavať keep-alive správy:

```
Potvrdenie o zacati komunikacie.
Potvrdenie o prijatie Keep-Alive spravy.
Potvrdenie o prijatie Keep-Alive spravy.
```

- Na odoslanie dát slúži funkcia **odosli_data()**, ktorá odošle daný typ dát:

```
def odosli_data(vysielac_socket, adressa, flag, meno_suboru, velkost_dat, velkost_frag, data, chyba):
    """
    Funkcia sluzi na odoslanie zadanych sprav na server.
    :param vysielac_socket:     Socket vysielaca.
    :param adressa:             Tuple adresa prijimaca (ip, port)
    :param flag:                Typ spravy.
    :param meno_suboru:         Meno suboru, ak odosielame subor.
    :param velkost_dat:         Celkova velkost prenasanych dat.
    :param velkost_frag:        Velkost jedneho fragmentu.
    :param data:                Zakodovane data, ktore odosielam.
    :param chyba:               Urcenie ci generujem chybu alebo nie.
    :return:                    Bool, True  -> ak komunikacia prebehla spravne.
                                      False -> ak komunikacia neprebehla spravne.
    """
```

- Pred odoslaním dát sa vypíšu dôležité informácie na strane vysielača:

```
Pre odoslanie textu  zadajte 't'.
Pre odoslanie suboru zadajte 's'.
Zadajte typ spravy: s
Zadajte cestu k suboru: obrazok.jpg
Absolutna cesta k suboru C:\Users\User\PycharmProjects\PKS2\obrazok.jpg.
Velkost suboru je 2101546 B.
Zadajte velkost jedneho fragmentu (maximalna velkost je 1463 B):
Zadajte velkost fragmentu: 1463
Chcete generovat chybu? (A/N): N
```

  - Cesta k súboru a jeho veľkosť

- Po úspešnom odoslaní dát sa vypíšu dôležité informácie na strane prijímača:

```
Potvdenie prijatia 1437. paketu. Velkost je 678 B.
Ukoncenie prenosu dat..
Absolutna cesta k suboru b'C:\\Users\\User\\PycharmProjects\\PKS2_druhy\\obrazok.jpg'.
Prenieslo sa 1437 paketov, celkova velkost suboru je 2101546 B.
```

  - Cesta k novému súboru, veľkost nového súboru (musia sa zhodovať), počet prenesených paketov a pri každom pakete je jeho veľkosť v B.

- Po prenesení dát s ana strane vysielača opäť objaví menu:

```
Odosielanie inicializacneho paketu serveru...
Zahajenie komunikacie.

Chcete odoslat data? -> 1
Chcete ukoncit komunikaciu a zmenit role? -> 2
Chcete ukoncit program? -> 3
Zadajte vstup:
```

- Ak chce používateľ zmeniť role alebo ukončiť program, ukončí sa komunikácia zruší sa vlákno pre keep-alive:

```
elif vstup == 2:
    je_thread = False
    keep_vetva.join()
    socket_1.sendto(str("E").encode(), adresa)
    print("Hlavne menu.")
    return
```

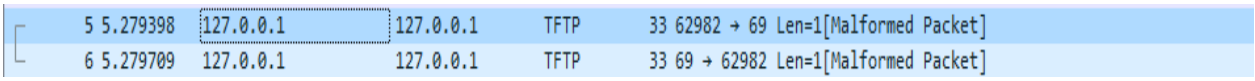
```
elif vstup == 3:
    je_thread = False
    keep_vetva.join()
    print("Ukoncenie programu. ")
    exit(0)
```

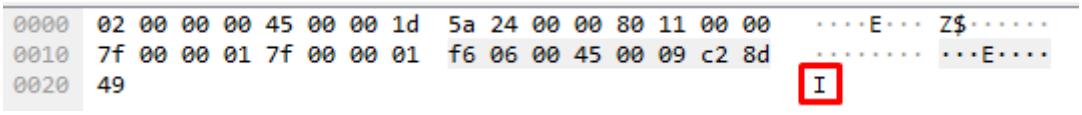
Adam Tomčala
ID: 103164

Testovanie vo Wireshark-u:

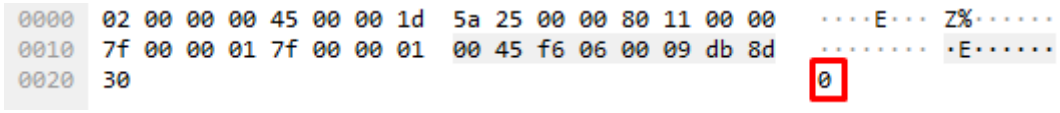
- Skúšal som prenášať rôzne typy súborov s rôznymi veľkosťami.
- Nasledujúci obrázky demonštrujú prenos dát zachytených vo Wireshark-u.

- 2 inicializačné pakety:



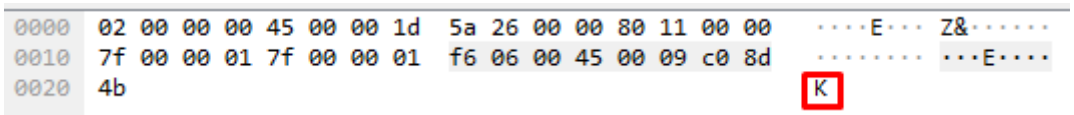


- Paket číslo 5 obsahuje inicializačnú správu **“I”.**

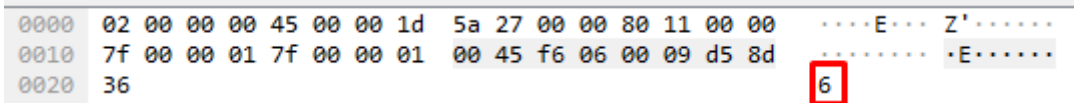


- Paket číslo 6 potvrdzuje nadviazanie spojenia **“0”**.

- Kým používateľ zadáva potrebné informácie na prenos dát, už začalo odosielanie keep-alive správ.

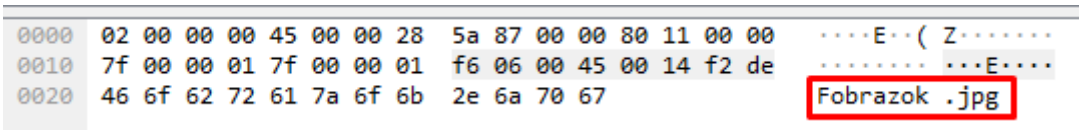


- Paket číslo 7 má nastevený typ na **“K”**.



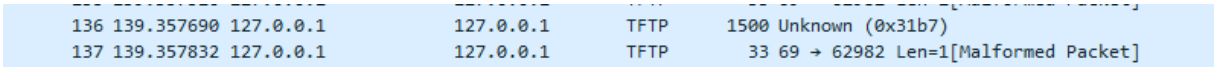
- Paket číslo 8 potvrdzuje prijatie keep-alive správy, typ nastavený na **“6”**.

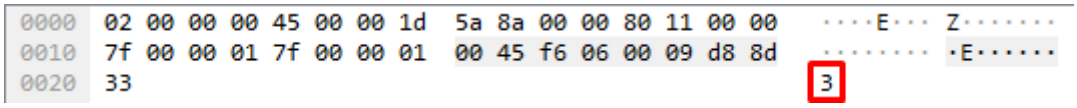
- Pre tento príklad som sa rozhodol prenášať súbor obrazok.jpg.
- Veľkosť fragmentu som nastavil na 1463 B.



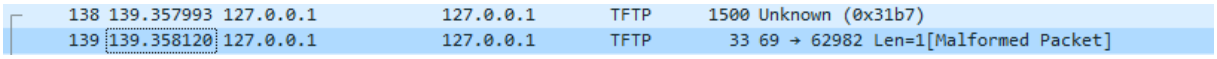
- Paket číslo 134 obsahuje informáciu o tom, aký typ dát sa ide prenášať (v tomto prípade je typ nastavený na **“F” + meno súboru** – obrazok.jpg).
- Paket číslo 135 obsahuje potvrdzovaciu správu podobne ako všetky potvrdzovacie správy.

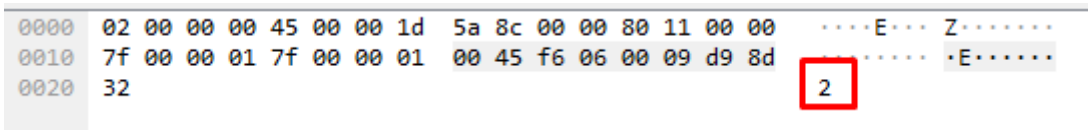
- Pre tento prípad som generoval chybu na prvom pakete (136).



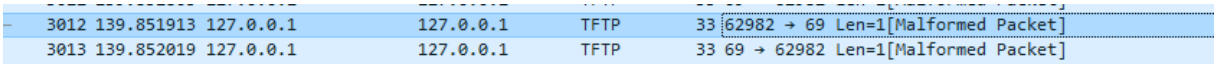


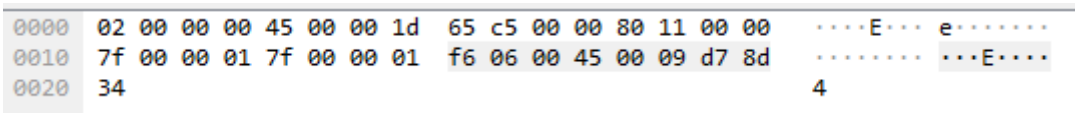
- Paket číslo 137 obsahuje žiadosť o opätovné odoslanie dát (**“3”**).
- Vysielač odošle príslušný paket ešte raz (paket číslo 138).



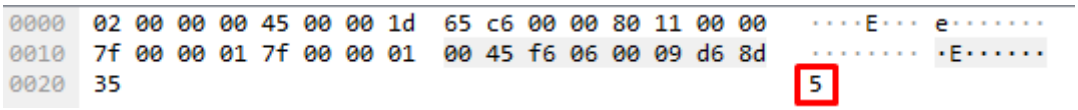


- Dáta, ktoré prišli už boli korektné a pride správa o správnom doručení (**“2”**).



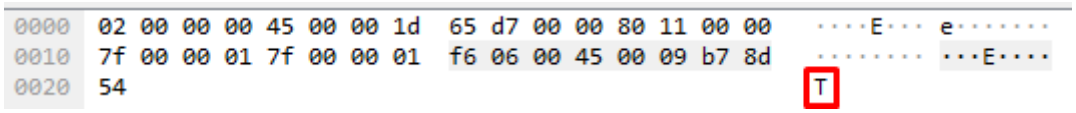


- Paket číslo 3012 obsahuje informáciu o skončení prenosu dát s typom nastaveným na hodnotu **“4”.**

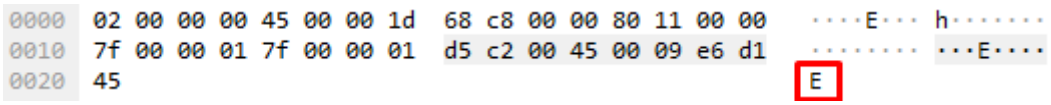


- Následne sa posiela potvrdenie o skončení prenosu dát (paket číslo 3013 s typom nastaveným na **“5”**).
- Potom sa opäť začnú posielať keep-alive správy.

- Takto by vyzeral paket s informáciou, že prenášam text:



- Keď ukončuje spojenie medzi uzlami, paket vyzerá nasledovne (typ nastevený na **„E“**):



Zmeny oproti návrhu:

- Aktualizoval som hlavičku na prenášanie súboru tým, že som pridal ďalšie možnosti pri políčku **Type.**
- Aktualizoval a prispôsobil som návrhové diagramy podľa mojej implementácie.
- Dôkladnejšie som popísal metódu CRC a keep-alive správy.
- Pridal som všetky knižnice a dôležíté funkcie, ktoré používam v mojom programe.
- Dôkladnejšie som popísal vysielač a prijímač (podľa mojej implementácie).