Počítačové a komunikačné siete: Semestrálne zadanie - Komunikácia s využitím UDP protokolu

Marián Kurčina

ID: 127211

xkurcinam@stuba.sk

Cvičenie: Štvrtok 16.00 - 18.00

Obsah

Úvod

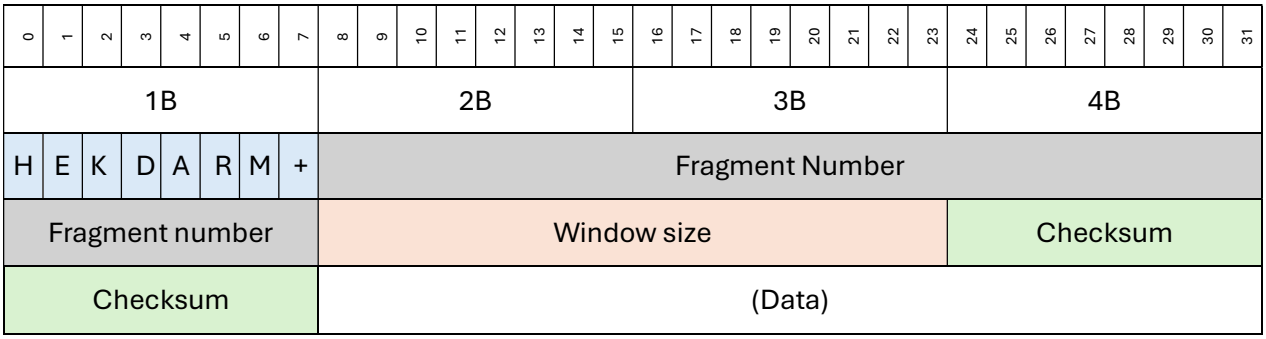
Mojou úlohou je implementovať P2P aplikáciu s využitím vlastného protokolu a UDP, ktorá bude schopná posielať správy a súbory. Hlavnou časťou bolo navrhnúť vlastný protokol, ktorý bude spĺňať nasledujúce požiadavky:

a) Nadviazanie spojenia medzi dvoma stranami a dohodnutie si parametrov spojenia
b) Umožniť poslať dáta po fragmentoch s veľkosťou zadanou zo vstupu od používateľa
c) Overenie integrity poslanej správy na strane príjemcu
d) Vedieť znova poslať stratenú alebo poškodenú správu
e) Zabezpečenie vzájomnej kontroly medzi uzlami, či je účastník na druhej strane spojenia stale aktívny
f) Umožniť úmyselne vytvoriť chybu v niektorej z prenášaných správ s cieľom simulovať poškodené dáta

Na základe týchto podmienok som navrhol vlastný protokol a taktiež som opísal procesy, ktoré sú vykonávané počas chodu aplikácie.

Aplikáciu som programovať v jazyku Python s použitím knižníc os, socket, threading, time, struct, random, tkinter a queue.

Navrhnutý protokol



Môj protokol má veľkosť hlavičky 9B, skladá sa z Flagov, Fragment Number (Number of fragments), Window size (Max window size), Checksum a Data.

Prvou časťou hlavičky sú Flags, táto časť slúži na identifikovanie typu správy, keď tento bit informácie je v binárnom zápise každá číslica hovorí, či je táto správa typu, ku ktorému číslica patrí (0-nie, 1-áno).
Handshake – slúži na identifikáciu Handshake správy.
Exit – slúži na identifikáciu správy ukončujúcu spojenie.
Keepalive – slúži na identifikáciu správy Keep-Alive.
Datatransfer – slúži na identifikáciu správ slúžiacich na prenos informácií.
ACK – slúži na identifikáciu správ potvrdzujúcich prijatie.
REQ – slúži na identifikáciu správ pýtajúce si opätovné poslanie poškodených dát.
Msg – slúži na rozoznanie správy (text bez uloženia).

Additional information – slúži pre identifikáciu 3. správy handshaku a na identifikáciu fragmentovanej správy.

Ďalšou časťou je Fragment number, táto časť identifikuje poradie fragmentu pri fragmentácii. Pri dohode parametrov pred posielaním súboru identifikuje počet posielaných fragmentov.

Window size mi určuje veľkosť okna pri Selective repeat a pri dohode parametrov určuje maximálnu veľkosť okna.

Checksum slúži na kontrolu správnosti dát. Vypočítam ho pomocou CRC 16-CCITT.

Poslednou časťou je časť Data, táto časť slúži na prenos konkrétnych dát. Pri dohadovaní parametrov prenosu súboru táto časť obsahuje názov súboru.

Typy správ

10000000 – prvá správa handshaku
10001000 – druhá správa handshaku
10001001 – tretia správa handshaku

01000000 – exit správa
01001000 – potvrdenie exit správy

00100000 – keepalive správa
00101000 – potvrdenie keepalive

00010010 – nefragmentované poslanie textu
00011010 – potvrdenie nefragmentovaného textu
00010110 – vyžiadanie preposlania nefragmentovaného textu

10010010 – správa na dohodu parametrov posielania fragmentovaného textu
10011010 – potvrdenie prijatia správy na dohodu parametrov posielania fragmentovaného textu
10010110 – vyžiadanie preposlania správy na dohodu parametrov posielania fragmentovaného textu

10010000 – správa na dohodu parametrov posielania súboru
10011000 – potvrdenie prijatia správy na dohodu parametrov posielania súboru
10010100 – vyžiadanie preposlania správy na dohodu parametrov posielania súboru
00010000 – správa posielania časti súboru
00011000 – potvrdenie prijatia časti súboru
00010100 – vyžiadanie preposlania časti súboru

Opis metód

Začiatok spojenia

Na začiatku sa užívateľ 1 pokúsi o spojenie a to tak, že pošle správu handshake1, užívateľ 2 ju prijme a pošle acknowledgement – handshake2, keď táto správa dorazí používateľovi 1 pošle užívateľovi 2 acknowledgement. Tento proces otestuje spojenie z oboch smerov a od tohoto okamihu bude spojenie pravidelne testovať pomocou metódy Keep-Alive.

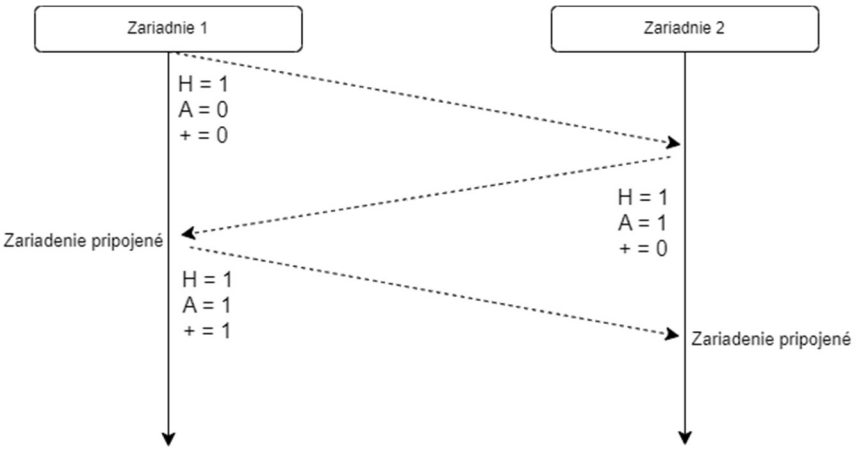


Diagram 1: Nadviazanie spojenia

Ukončenie spojenia

Pre ukončenie spojenia pošle užívateľ 1 používateľovi 2 správu exit, používateľ 2 ju prijme a odošle acknowledgement. Po skončení procesu sa ukončí testovanie Keep-Alive.

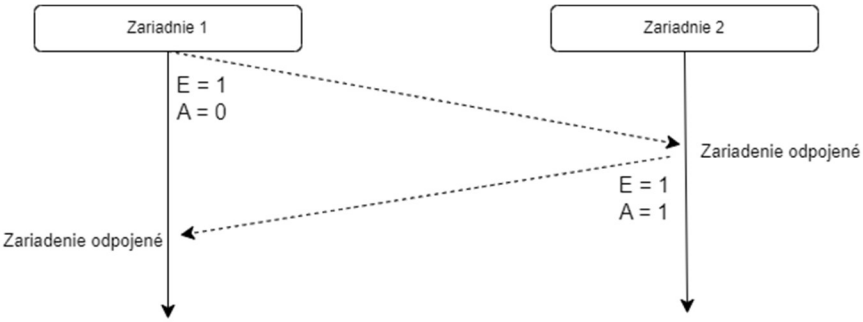


Diagram 2: Ukončenie spojenia

Udržanie spojenia

Pre udržanie spojenia bude spojenie testované metódou Keep-Alive, pravidelným posielaním správ keepalive z oboch komunikátrov. Počas nečinnosti (neposielania správ) bude každých 5 sekúnd overené spojenie, po 3 nezodpovedaných správach bude spojenie ukončené, procesy budú

zastavené a užívateľ bude povinný pokúsiť sa o opätovné spojenie. Ak sa tak stane, môže pokračovať v procesoch pred ukončením spojenia.

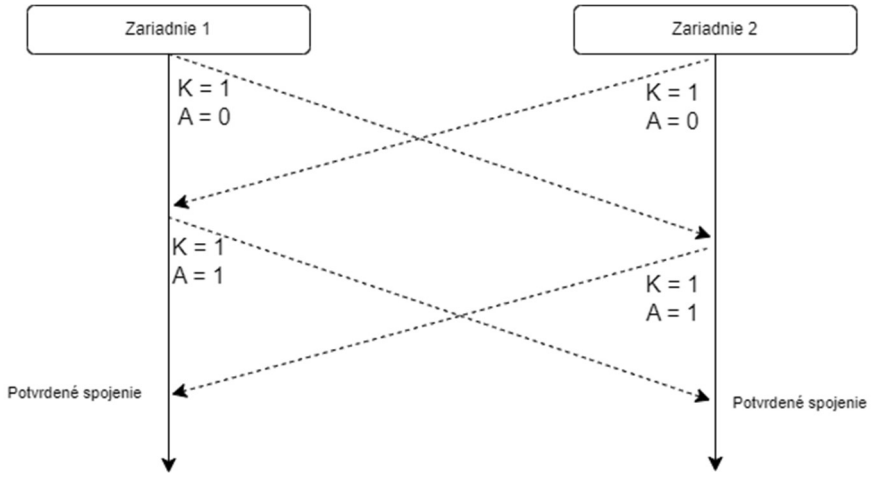


Diagram 3: Udržanie spojenia

Posielanie nefragmentovaného textu

Pri posielaní správy užívateľ 1 pošle užívateľovi 2 správu, ktorej Fragment Number a Window size bude prázdne pole. Užívateľ 2 nasledovne pošle ackowledgement alebo si vypýta opätovné zaslanie správy.

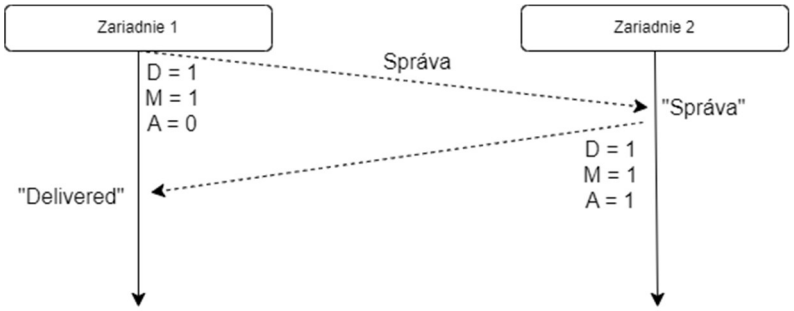


Diagram 4: Poslanie nefragmentovaného textu

Dohodnutie parametrov pre poslanie

Pri dohodovaní parametrov pred poslaním súboru alebo fragmentovaného textu, pošle odosielateľ správu ktorá informuje prijímajúceho o parametroch dôležitých pre poslanie a uloženie súboru alebo textu. Pole Fragment Number bude obsahovať číslo posledného fragmentu (pre nefragmentovaný presun súboru bude toto pole mať hodnotu 0). V poli Window Size bude maximálna hodnota pre window. V časti Data bude pri posielaní súboru uvedené meno súboru, pri posielaní fragmentovaného textu bude toto pole prázdne. Po prijatí parametrov pošle užívateľ acknowledgement.

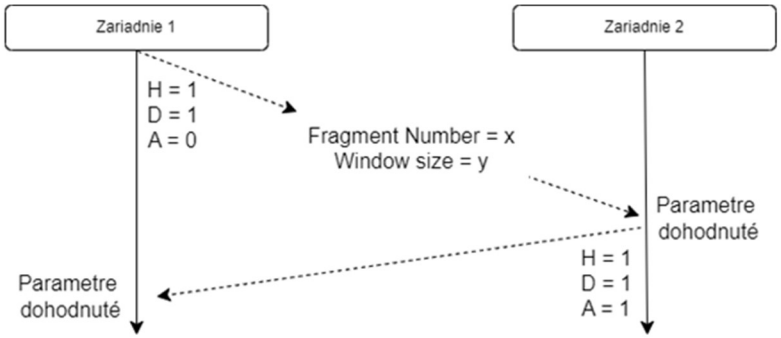


Diagram 5: Dohoda parametrov pre poslanie textu

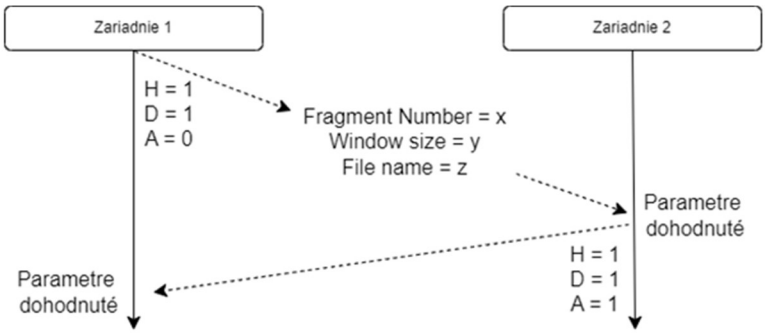


Diagram 6: Dohoda parametrov pre posielanie súboru

Posielanie súboru a fragmentovaného textu

Pri poslaní nefragmentovaného súboru bude odoslaná iba jedna správa s dátami. Ak ju prijímateľ úspešne prijme pošle acknowledgement.
Pri poslaní fragmentovaného súboru alebo textu odosielateľ pošle toľko fragmentov, koľko mu povoľuje Window Size, inak povedané, posiela postupne všetky fragmenty v okne, pričom okno je definované prvým prvkom (prvý prvok na ktorý odosielateľ nedostal acknowledgement) a veľkosťou okna. Prijímateľ sa snaží ukladať prijaté fragmenty v poradí. Po uložení fragmentov do pamäte pošle acknowledgement o poslednom uloženom fragmente. Ak nemôže uložiť ani jeden fragment, pošle request o fragment, ktorý má uložiť ako nasledujúci. Odosielateľ po prijatí requestu opäť odošle fragment. Po prijatí ackonowledgementu posunie svoje okno na nasledujúce miesto od potvrdeného fragmentu.

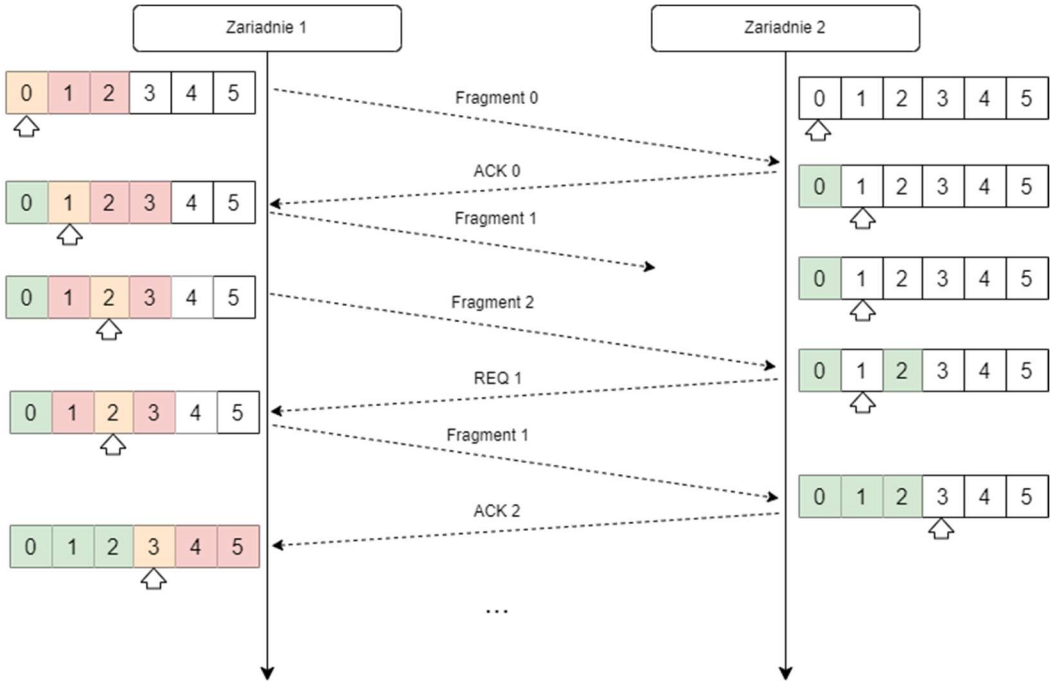


Diagram 7: Poslanie fragmentovaných dát

Ak počas presunu dát dôjde k náhlemu výpadku spojenia, komunikátory po 5 sekundách neaktivity overia spojenie pomocou 3 keepalive správ s časovou medzerou 5 sekúnd. Ak komunikátor dostane ack prenos ďalej pokračuje. Ak ani na jednu z 3 správ nedostane ack, spojenie je ukončené.

Kontrola poškodenia a strát dát

Na kontrolu poškodenia bude slúžiť pole Checksum. Toto pole bude mať hodnotu, ktorá bude vypočítaná z dát.
Na vypočítanie checksumu som sa rozhodol použiť CRC-16-CCITT. Tento checksum sa počíta nasledovne:
Pre každý bajt v dátach sa vykoná posun o 8 bitov doľava a nasledovne prebehne bitová operácia XOR s aktuálnou hodnotou CRC (Na začiatku FFFF v hexadecimálnej sústave). Nasledovne vykonáme 8 krát operáciu na základe hodnoty prvého bitu, ak je prvý bit 1, posunieme aktuálne CRC doľava a vykonáme operáciu XOR. Ak je 0, posunieme iba CRC doľava.
Tento proces opakujeme pre každý bajt informácie.

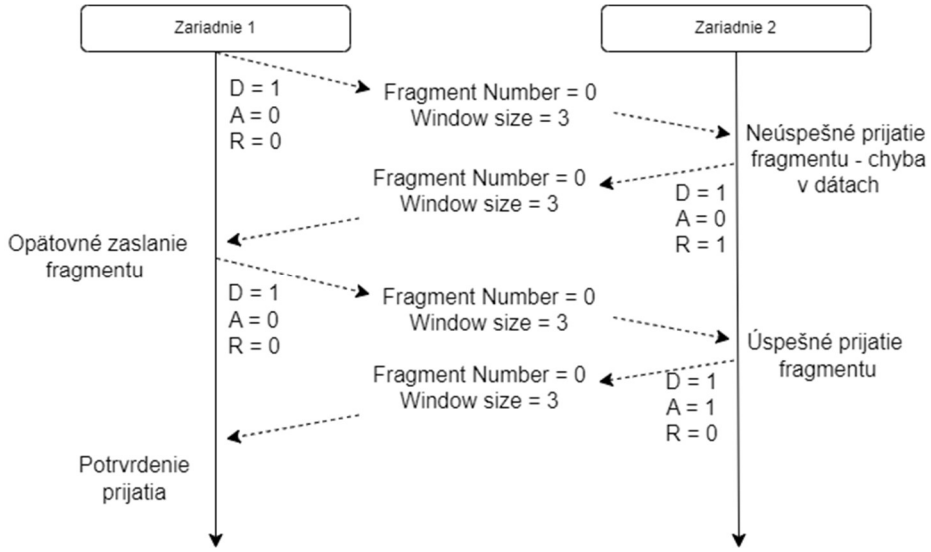


Diagram 8: Prijatie chybnej sprsomávy

Simulácia poškodenia dát

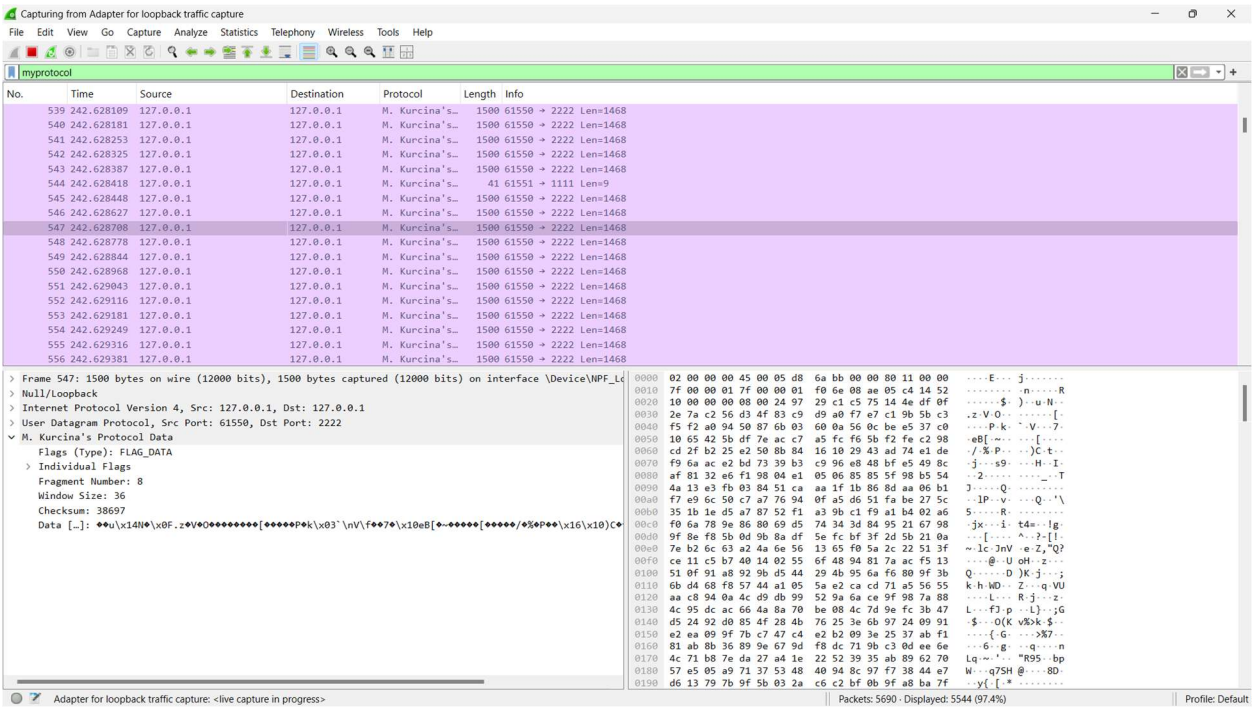
Pri posielaní má používateľ možnosť nastavenia mieru chybnosti pri odosielaní dát, táto miera môže byť 0-50. Pri tvorbe správy bude na konci po vypočítaní checksumu zmenená hodnota jedného bajtu na iný.

Opis aplikácie

Po spustení aplikácie je vyžadované od užívateľa zadanie parametrov spojenia, svoju IP adresu a port a IP adresu a port druhého používateľa. Po zadaní týchto parametrov sa zobrazí GUI okno, v ktorom bude používateľ riadiť komunikáciu. Pred tým ako môže čokoľvek urobiť, musí si nastaviť komunikátor, je potrebné nastaviť adresu ukladania súborov, veľkosť jedného fragmentu (maximálna veľkosť je nastavená tak, aby bola celá správa menšia ako 1500B) a mieru poškodenia správ. Tieto nastavenia sa počas behu programu môžu meniť. Po uložení sa užívateľ môže pokúsiť o pripojenie. Po pripojení má možnosť odpojenia, po odpojení je opätovná možnosť pripojenia.
Po úspešnom pripojení prebieha kontrola spojenia pomocou správ keepalive. Užívateľ má možnosť poslania správ alebo súborov. Pre poslanie správ stačí napísať správu do požadovaného okna a poslanie bude úspešné. Ak ide o poslanie správy menšej ako je nastavená veľkosť fragmentu, správa bude poslaná v jednom celku. Ak bude správa fragmentovaná, bude poslaná po fragmentoch. Celá správa sa objaví vo výpisovom okne u oboch používateľoch. Ak chce užívateľ poslať súbor, zadá jeho adresu. Vo výpisovom okne sa zobrazí informácia o odoslaní/prijatí súboru a jeho adresa. Všetky ostatné dôležité informácie o správach, spojení a čase sú vypisované v termináli. Počas behu programu používateľ v aplikácii vidí informáciu o stave jeho komunikátora, či je pripojený, odpojený alebo v procese posielania.
Po vypnutí aplikácie sú všetky vlákna vypnuté pomocou globálnej premennej a socket je ukončený.

Detekcia protokolu vo Wireshark

Na detekciu môjho protokolu sme mali vytvoriť lua skript. Môj skript je nastavený na dva porty, 1111 a 2222. Ak budú pri posielaní použité iné porty, wireshark nebude schopný jeho detekcie. Môj skript identifikuje typ správy podľa flagov a výpis jednotlivých flagov s pravdivostnými hodnotami. Ďalej vypíše hodnoty pre fragment number, window size, checksum a data. Vo wiresharku som si navyše vytvoril coloring rule, ktorý zvýrazní mnou poslané správy



Obrázok 1: Ukážka rozpoznávania častí môjho protokolu pomocou lua skriptu a coloring rules

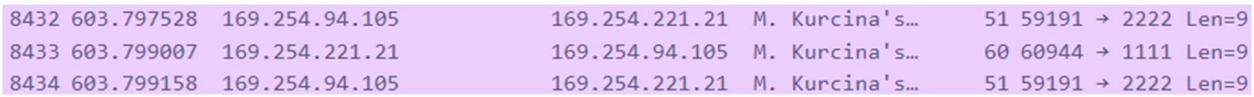
Vykonané zmeny

Počas implementácie som bol nútený zmeniť metódy, svoj protokol a kód aplikácie. Vykonal som nasledovné zmeny:

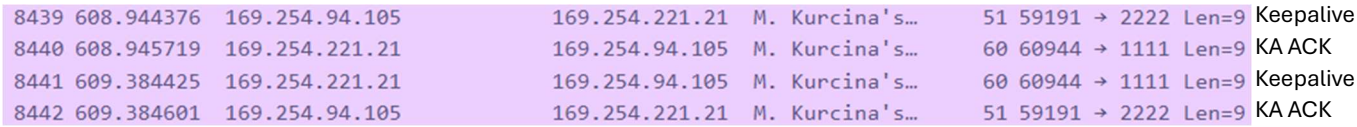
- Hlavička protokolu bola zväčšená o 2B z dôvodu neschopnosti poslať 2MB súbor s použitím fragmentácie o veľkosti 1B.
- V pôvodnej verzii pri dohode parametrov odosielania fragmentovaného súboru/textu bola možnosť prenášať viac informácií potrebných k prenosu dát. Pri implementácii som zistil, že mi stačí na úspešné uloženie meno súboru
- Zmena metódy Keep-alive, pôvodne bola táto metóda udržiavania spojenia vykonávaná pomocou jednostranného posielania keepalive správ a prijímania potvrdenia o potvrdení, keepalive správy posielala iba jedna strana. Zmenou je, že vo finálnej verzii posielajú keepalive správy obe strany a čakajú na potvrdenie o prijatí.
- Zadefinovanie riešenia problému s náhlym výpadkom spojenia počas presunu dát

- Zmena metódy na simuláciu poškodenia dát, v kontrolnom bode bola chyba simulovaná pomocou prepnutia jedného bitu na opačný, vo finálnej verzii je chyba realizovaná pomocou zmeny celého Bajtu
- Zmena metódy overenia správnosti dát správy, v kontrolnom bode som kontroloval správnosť dát pomocou spočítania zapnutých bitov, vo finálnej verzii používam CRC-16-CCITT
- Na iplementáciu používam viac knižníc ako v kontrolnom bode, používam knižnice os, socket, threading, time, struct, random, tkinter a queue, oproti pôvodným socket, threading, time a struct.
- Pôvodná aplikácia mala 3 vlákna, jedno na odosielanie, jedno na prijímanie a jedno na input užívateľa, v mojej finálnej verzii na tieto úlohy používam 2 vlákna a tkinter okno. Tkinter okno slúži na input užívateľa a odosielanie správ, jedno vlákno spĺňa funkciu keepalive a druhé slúži na prijímanie správ a posielanie ack správ

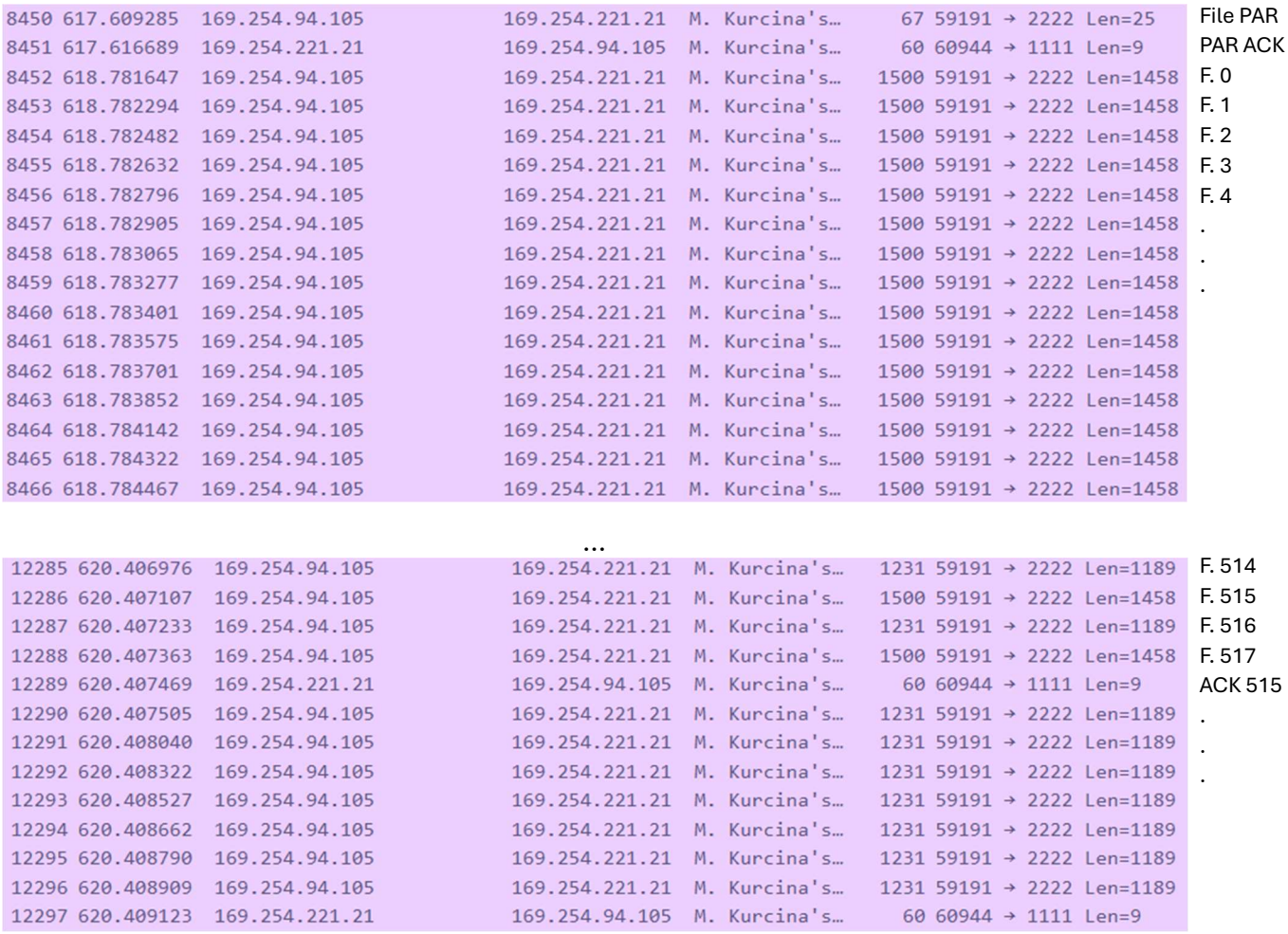
Ukážka testovacieho scenára



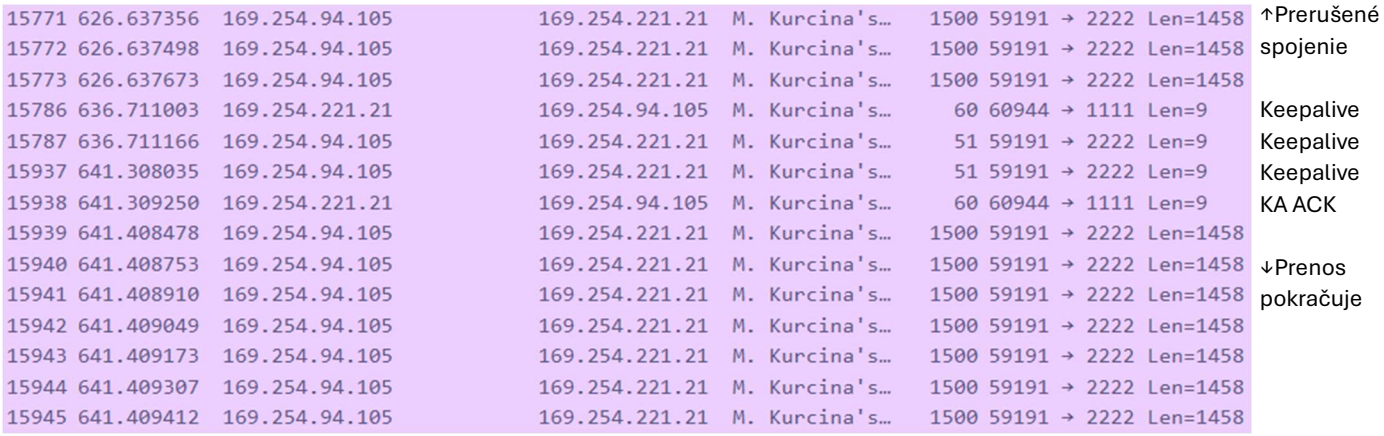
Obrázok 2: Otvorenie spojenia



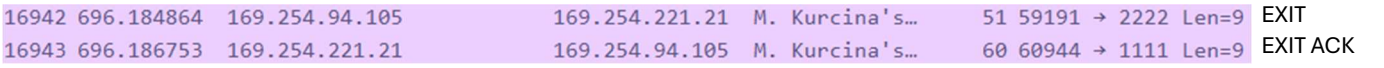
Obrázok 3: Keep-alive



Obrázok 4: Prenos súboru bez chyby



Obrázok 5: Prenos súboru s chybou spojenia a následnou opravou



Obrázok 6: Ukončenie spojenia

Záver

Mojou úlohou bolo navrhnúť protokol, ktorý by vyhovoval podmienkam v zadaní a vytvoriť aplikáciu na nadviazanie spojenia a posielanie správ pomocou môjho protokolu. Úlohu sa mi podarilo splniť. Moja aplikácia dokáže nadviazať spojenie, posielať text a súbory, fragmentované aj nefragmentované, kontrolovať spojenie a ukončiť spojenie, za využitia mnou navrhnutého protokolu. Môj protokol je navrhnutý tak, aby dokázal plniť funkcionality zo zadania. Tieto funkcionality som v svojej dokumentácii opísal a aplikáciu sa mi podarilo doimplementovať.