Slovenská technická univerzita v Bratislave, Fakulta informatiky a informačných technológií
Ilkovičova 6276/2, 842 16 Bratislava 4

Počítačové a komunikačné siete
Zadanie 2 – Komunikácia s využitím UDP protokolu
Návrh

Veronika Macejáková
xmacejakova@stuba.sk

Zimný semester 2023

Obsah

1. Zadanie práce

Cieľom práce je navrhnúť a implementovať program s použitím vlastného protokolu nad protokolom UDP transportnej vrstvy sieťového modelu TCP/IP pre komunikáciu medzi 2mi účastníkmi v lokálnej sieti. Program bude pozostávať z vysielacej časti, ktorá posiela dáta a z prijímacej časti, ktorá prijíma vyslané dáta. V prípade, že je súbor väčší ako maximálna používateľom definovaná veľkosť fragmentu je potrebné súbor rozložiť na menšie časti a po dokončení prenosu informovať o úspešnosti.

Program musí obsahovať kontrolu chýb pri komunikácii a znovu vyžiadanie chybných fragmentov. Po zapnutí programu, komunikátor automaticky odosiela paket pre udržanie spojenia každých 5s pokiaľ používateľ neukončí spojenie ručne.

2. Analýza

Pri posielaní dát po sieti sú využívané viaceré protokoly pre zabezpečenie kompletných a korektných dát na strane prijímateľa a hlavne, aby dáta dorazili k správnemu príjemcovi. Na opísanie funkcie komunikačného systému sa používa TCP/IP model a OSI model. TCP/IP model pozostáva zo 4 vrstiev.

Vrstvy TCP/IP modelu :

1. Aplikačná vrstva
2. Transportná vrstva (TCP/UDP)
3. Sieťová/internetová vrstva (IP)
4. Vrstva sieťového prístupu

V našom prípade nás zaujíma hlavne transportná vrstva, kde sa nachádzajú TCP a UDP protokoly.

2.1 TCP protokol

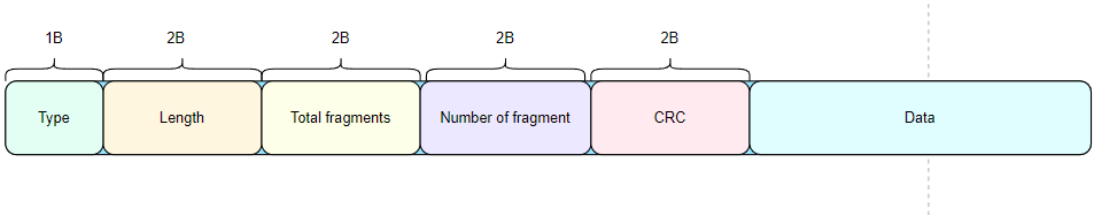
TCP protokol je spojovo orientovaný protokol a je spoľahlivý, pretože zabezpečuje, aby sa údaje dostali na správne miesto v presnom poradí, v akom boli odoslané. V prípade chybného fragmentu, je tento fragment znovu poslaný. TCP protokol sa používa najmä na posielanie dát, pri ktorých je podstatné, aby žiaden fragment nebol stratený na ceste k príjemcovi.

2.2 UDP protokol

UDP protokol nie je spojovo orientovaný a nie je 100% spoľahlivý, kvôli tomu, že neposkytuje kontrolu správnosti doručenia dát ani to, či boli dáta komplet doručené. Kvôli jeho nespoľahlivosti sa používa napríklad na video hovory, kde nie je potrebná spoľahlivosť dát. Od maximálnej veľkosti UDP dát je potrebné odpočítať IP a UDP hlavičku (20B a 8B) a taktiež moju vlastnú UDP hlavičku vo veľkosti 7B, takže 1500B – 20 – 8B – 9B = 1463B je maximálne množstvo dát.

3. Návrh

3.1 UDP hlavička



Obrázok 1 Návrh UDP hlavičky

Hlavička bude pozostávať z 5tich častí :

1. Typ o akú správu ide
2. Celková dĺžka fragmentu – maximálna dĺžka 1463B
3. Celkový počet fragmentov na udržanie prehľadu o spracovaných a nespracovaných fragmentoch
4. Číslo fragmentu v poradí na zabezpečenie v prípade chyby preposlanie správneho fragmentu
5. Kontrolná suma – na základe toho sa zistí, či prišli všetky dáta

3.2 Typ správy

Od typu správu budú závisieť ďalšie kroky programu.

0000 0001	Žiadosť o spojenie
0000 0010	ACK správa
0000 0100	Posielanie dát
0000 1000	Chybný fragment
0001 0000	Korektný fragment
0010 0000	Keep alive
0100 0000	Ukončenie spojenia

Obrázok 2 Typy správ

- Žiadosť o spojenie - bude využívaná na prvotnú inicializáciu spojenia
- ACK správa - využívaná na potvrdenie spojenia alebo ukončenia
- Posielanie dát – posielanie dát zo strany klienta
- Chybný fragment – v prípade chyby bude inicializované preposlanie konkrétneho fragmentu
- Korektný fragment – bude signalizovať, že sa môže poslať ďalší fragment
- Keep alive – posielané každých 5s na udržanie spojenia, inak bude spojenie ukončené
- Ukončenie spojenia – buď zo strany klienta, ak nemá žiadne dáta na poslanie alebo zo strany servera, ak nedostal Keep alive správu

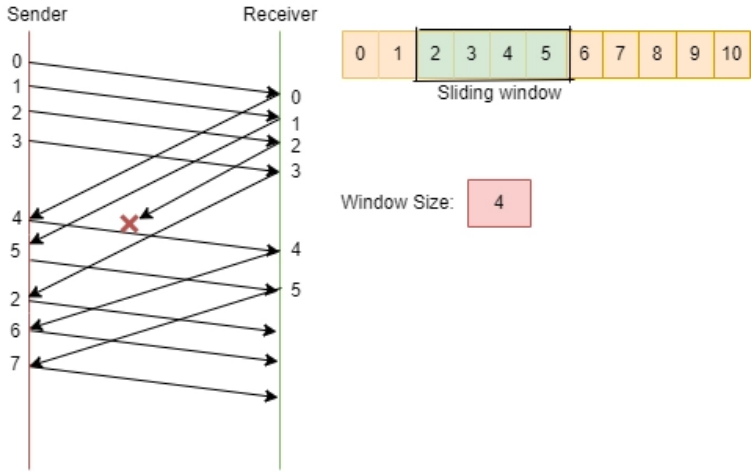
3.3 CRC

CRC (Cyclic Redundancy Code) je metóda používaná na detekciu chyby v prenesených dátach delením dát polynómom veľkosti n (polynóm je dopredu známi) . Na koniec dát pre pripísaných n-1 núl a následne prebieha pre celú dĺžku dát výpočet CRC. Výpočet prebieha tak, že postupne sa aplikuje

XOR (logická nezhoda) na dĺžku dát n a daný polynóm a následne sa aplikuje bitový posun. V prípade, že zvyšok dá nie je možné viac deliť, našiel sa výsledok pre CRC a ten je pripadaný do hlavičky na strane odosielateľa a na strane prijímateľa sa výpočet musí zhodovať s tým čo dostal v hlavičke UDP.

3.4 ARQ – sliding window

ARQ (Automatic Repeat Request) je metóda na kontrolu doručenia dát a to takým princípom, že odosielateľovi dát je poslaná potvrdzujúca správa že dané dáta/fragment dát bol prijatý. ARQ poskytuje viacero metód napr. Stop and Wait, Go Back-N, Selective repeat a Dynamic sliding window, ktorý budem používať.



Obrázok 3 Sliding window

Metóda sliding window využíva danú veľkosť okna n , na základe toho sa pošle prvých n fragmentov dát a čaká na potvrdenie doručenia fragmentu. Po doručení potvrdenia posiela ďalší fragment a posúva sa okno. Fragmenty za oknom predstavujú fragmenty, ktoré boli poslané a bolo prijaté potvrdenie, fragmenty v okne predstavujú poslané fragmenty bez potvrdenia a fragmenty pred oknom sú ešte neposlané. V prípade, že nepríde potvrdenie o prijatí fragmentu, je daný fragment preposlaný.

4. Priebeh komunikácie

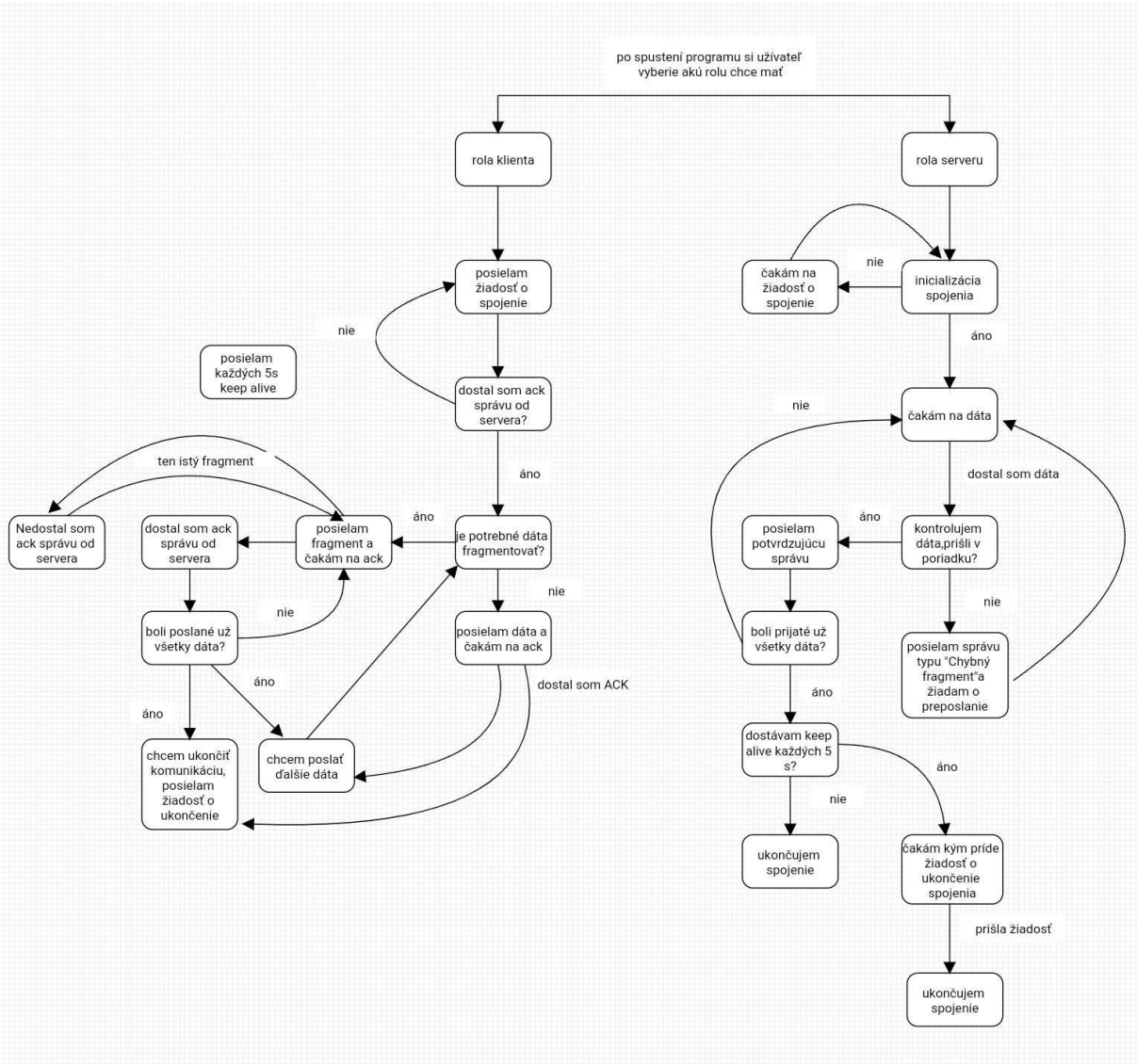
Po spustení programu bude používateľovi umožnené vybrať si rolu klienta alebo rolu servera.

Rola servera

Ako prvé je potrebné, aby užívateľ zadal aká bude IP adresa servera a port na akej bude komunikácia prebiehať. Následne server počúva a čaká, kým mu príde žiadosť o spojenie a pošle potvrdzujúcu správu žiadateľovi. Server čaká, kým mu budú doručené nejaké dáta a následne je potrebné tieto dáta skontrolovať, či prišli v poriadku. Ak dáta prišli v poriadku server posiela potvrdzujúcu správu a čaká na ďalšie dáta. V prípade chybných dát posiela správu, že dáta neprišli v poriadku a čaká na korektné dáta. Ukončenie spojenia môže nastať tak, že server nedostane každých 5s keep alive správu alebo mu príde žiadosť o ukončenie spojenia.

Rola klienta

V úlohe klienta je podstatné posielať keep alive správy každých 5s a na začatie komunikácie poslať žiadosť o pripojenie na server a čakať na potvrdenie spojenia. Ak máme nadviazané spojenie prichádza na rad samostatné posielanie dát. Dáta, ktoré nie je potrebné fragmentovať môžu byť poslané na server a počkať na potvrdenie, ale pri dátach, ktoré je potrebné fragmentovať je dôležité zabezpečiť aj preposlanie chybného fragmentu a za každý fragment dostať potvrdenie o prijatí. Keď sú všetky dáta prijaté môže klient poslať ďalšie rovnakým postupom alebo ukončiť spojenie poslaním žiadosti o ukončenie spojenia.



Obrázok 4 Priebeh komunikácie