FAKULTA INFORMATIKY A INFORMAČNÝCH TECHNOLÓGIÍ
SLOVENSKÁ TECHNICKÁ UNIVERZITA
Ilkovičova 2, 842 16 Bratislava 4

2021/2022

Počítačové a komunikačné siete

Zadanie č.2 – Komunikácia s využitím UDP portu

Cvičiaci: Ing. Miroslav Bahleda, PhD.
Čas cvičení: Štvrtok 18:00 – 19:50

Vypracovala: Monika Zjavková
AIS ID: 105345

Obsah

1. Zadanie

Navrhnite a implementujte program s použitím vlastného protokolu nad protokolom UDP (User Datagram Protocol) transportnej vrstvy sieťového modelu TCP/IP. Program umožní́ 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. Vysielací́ 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 znovuvyžiadanie 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.

2. Návrh

Zadanie bude implementované v jazyku Python s pomocou knižnice socket. Používateľovi umožní určiť cieľovú IP adresu a port pomocou vstupu z konzoly, kde tiež dostane možnosť zvoliť si maximálnu veľkosť fragmentu. Hotový komunikátor bude fungovať na princípe server a klient, bude na používateľovi či si zvolí posielať dáta (klient) alebo prijímať dáta (server) a tiež si môže zvoliť, či chce simuláciu s chybou, v tom prípade sa vyskytne v prvom fragmente.

2.1. Hlavička

Správa	Veľkosť dát	Počet Fragmentov	Poradie fragmentu	CRC	Data

Hlavička bude obsahovať typ správy bude veľká jeden byte a bude obsahovať informáciu o tom, či inicializácia spojenia prebehla úspešne, aký typ dát sa bude posielať, či sa jedná o prenos dát a na záver, či boli dáta poslané úspešne alebo neúspešné a bude treba ich poslať znova. Nasleduje veľkosť posielaných dát, počet fragmentov a poradie fragmentu, aby program vedel, či už je to posledný a na tieto údaje budú vyhradené pravdepodobne 2, presné číslo bude určené podľa potreby implementácie.

Na konci hlavičky bude ešte CRC hodnota, ktorej veľkosť bude určená podľa potrieb pri implementácii, pravdepodobne 2 - 4 byty. Za hlavičkou budú potom nasledovať dáta daného fragmentu.

Veľkosť hlavičky bude maximálne teda 11 bytov.

2.2. Kontrola správnosti rámca – metóda kontrolnej sumy

Na kontrolu správnosti rámec bude použité metóda kontrola cyklickým kódom (CRC). Vypočítaná hodnota bude funkciou z Python knižnice zlib.

Na základe bitov sa vypočíta hodnota rámca pred aj po prijatí na server. Ak sa tieto dva údaje zhodujú, prijatý rámec bol odoslaný bez porušenia. Hodnota sa vypočítava s maskou cez posuvné registre a jej hodnota je určená generujúcim polynómom. V implementácii bude použitý odporúčaný nerozložiteľný polynóm:

$$x^{16} + x^{12} + x^5 + 1$$

V hlavičke sa posielajú sa len koeficienty polynómov a zvyšku po delení. Po prijatí súboru sa vždy skontroluje, či rámec je nepoškodený a pošle správu o úspešnom poslaní. Ak nie, pošle sa rámec znova.

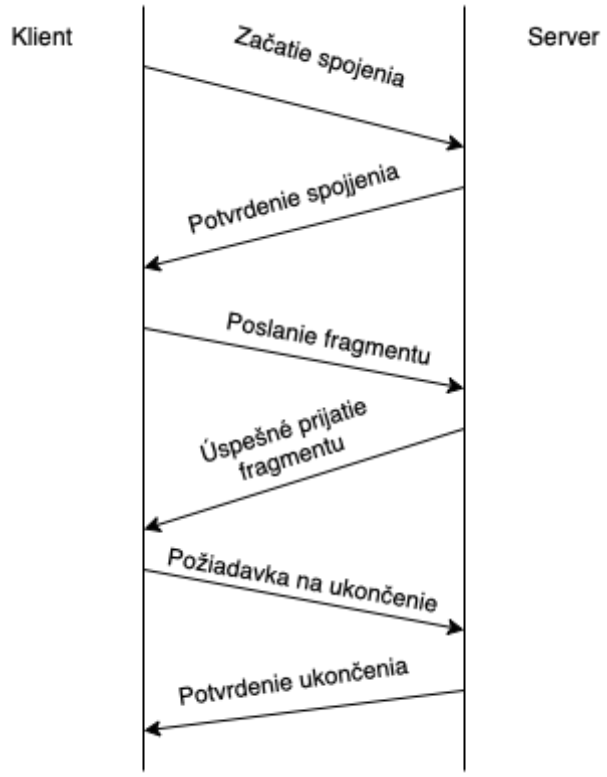
2.3. Fungovanie ARQ

Pri implementácii komunikátora bude použitá Stop-and-wait ARQ, na zaistenie, že všetky rámce prídu v správnom poradí. V programe to teda bude fungovať tak, že nebude poslaný ďalší, kým nepríde správa o potvrdení, že prišiel ten predchádzajúci a v tom prípade nebude treba ani poradie rámca Ak nepríde potvrdzujúca správa v potrebnom čase, pošle sa rámec znova, rovnako aj keď príde poškodený. Keď príde správny fragment, zapamätá sa a poškodený sa zahodí, aby program vedel, s ktorým má pracovať.

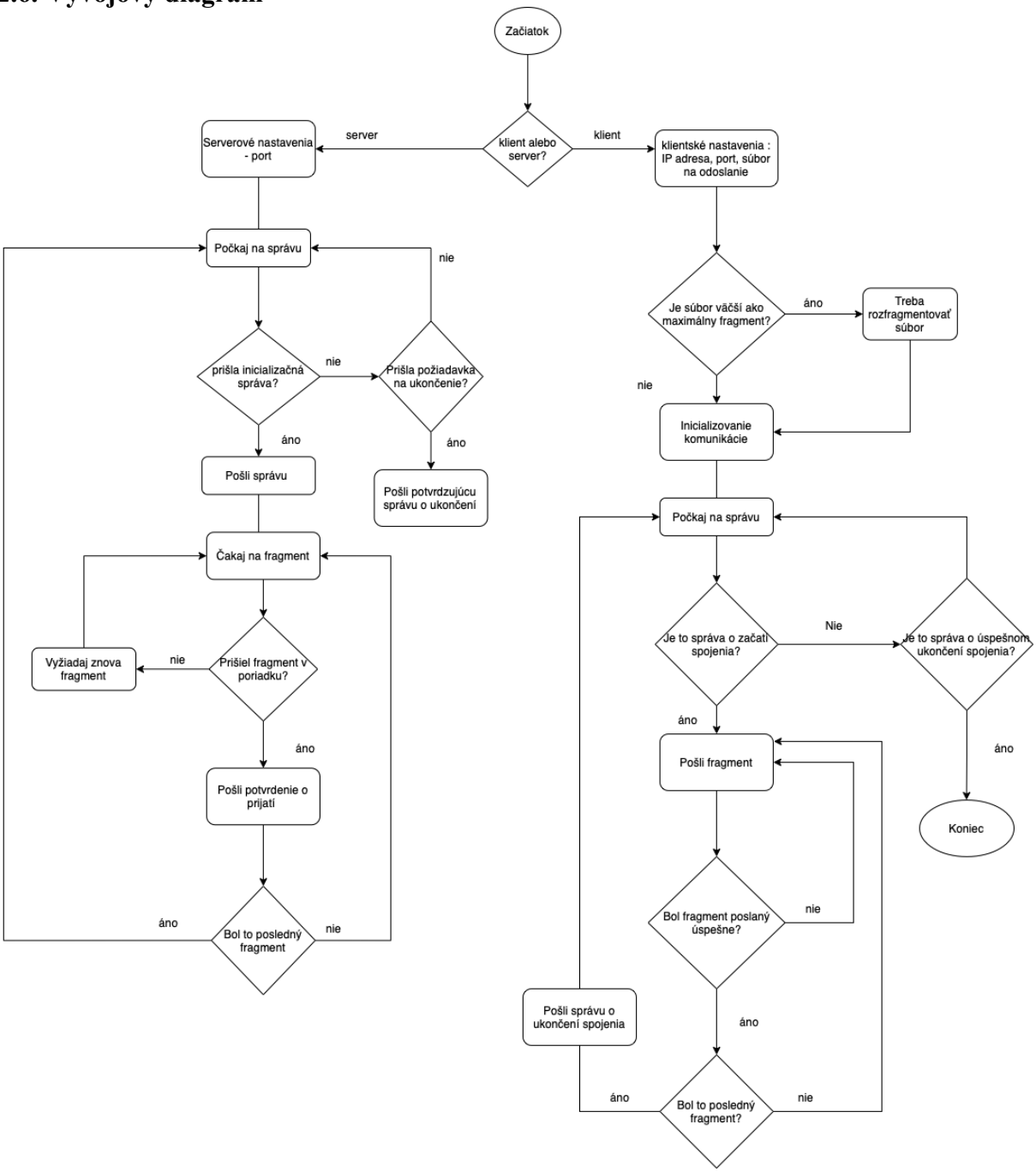
2.4. Metódy udržania spojenia

Po otvorení spojenia, bude udržiavané otvorené najbližších 20 sekúnd posielaním správ na udržanie otvoreného spojenia, ak nepríde ďalšia správa alebo fragment, spojenie sa skončí s možnosťou znovuotvorenia, ak bude klient chcieť poslať ďalší rámec. Tento časovač bude vždy ďalším fragmentom reštartovaný. V prípade ak nastane koniec skôr ako sa odošle celý súbor, bude poslanie uznané ako neúspešné.

2.5. Diagram spracovávania



2.6. Vývojový diagram



3. Implementácia

Pri spustení programu dostane používateľ na výber, či chce program spustiť ako server alebo ako klient. Pre server aj klienta je nastavená defaultná IP adresa ako premenná, pomocou menu sa však bude dať cez konzolu zmeniť.

Zmeniť sa zo serveru na klienta a opačne bude možné spraviť manuálne, ak bude úspešne ukončené spojenie alebo program skoční po vypršaní časovača pre server. V tom prípade je používateľ vrátený do úvodného menu, kde si môže znova zvoliť ako čo sa chce prihlásiť.

3.1. Klient

Po prepnutí na klienta program pošle správu o začatí komunikácie, ak mu od servera príde kladná odpoveď, vypýta si od používateľa, čo chce poslať. Môže to byť správa alebo súbor. V oboch prípadoch je podobný postup. Zistí sa potrebná veľkosť fragmentu a ak ide o súbor tak aj cestu a názov súboru.

Keď už má funkcia všetky potrebné informácie, vypočíta počet fragmentov podľa toho, akú veľkosť si zadal používateľ a pošle to v hlavičke serveru aj s menom súboru. Keď sa posiela správa, je tam iba počet fragmentov.

Každý súbor je otvorený ako binárny, ktorý sa číta po bajtoch podľa zadanej požadovanej veľkosti. Vtedy sa pri pripájaní hlavičky s dátami nič nedeje. V prípade správy je pretransformovaná na bajtový reťazec.

Ak prebehne poslanie všetkých základných informácii, začnú sa dáta posielať v cykle, kým sa nepošlú všetky fragmenty. Na začiatku sa prečítajú dáta podľa zadefinovanej veľkosti a pridá sa tam hlavička s flagom – čiže typom správy o aký fragment ide, poradím fragmentu a CRC na kontrolu správnosti, ktorá sa vypočíta z dátovej časti. Fragmentácia je uplatnená iba na dátovú časť, čiže hlavička sa neberie do úvahy pri delení súboru alebo správy na fragmenty.

Po prijatí, server odošle odpoveď o stave, v akom prišiel fragment. V prípade že bol chybný sa nespraví nič a v ďalšom cykle sa posiela ten istý fragment. Funkcia je nastavená tak, že každý prvý fragment je chybný. Chyba sa simuluje pomocou prehodenia prvého a posledného bajtu.

Ak fragment prišiel v poriadku hodnota, ktorá zaznamenáva poradie fragmentu sa zvýši o 1 a buď sa prečítajú ďalšie bajty zo súboru alebo sa posunie začiatok správy, aby v ďalšom cykle sa poslali nové dáta.

Na konci pošle klient serveru správu, že všetky fragmenty boli poslané a následne vypíše odpoveď či všetky prišli alebo nejaké boli stratené.

Po odoslaní si môže používateľ vybrať, či chce poslať ďalší súbor či správu, alebo ukončiť spojenie, kedy pošle žiadosť na server o ukončenie a keď mu príde požadovaná odpoveď spojenie je považované za skončené a vráti sa do menu.

V prípade, že spojenie ostáva otvorené, posielajú sa v pravidelnom intervale KeepAlive správy, aby server neuzavrel spojenie. Tieto správy sú posielané pomocou použitia threadu, ktorý sa spúšťa vždy, keď je používateľ v menu.

3.2. Server

Server po spustení začne počúvať na porte, ktorý mu prišiel ako vstupný parameter a nastaví sa timeout, po ktorom sa ukončí spojenie s klientom.

Keď príde správa od klienta identifikuje flag a podľa toho sa určí jeho ďalšie správanie. Na začiatku čaká iba na zahájenie komunikácie. Ak by mu prišiel iný flag v hlavičke ako s požiadavkou o začatie spojenia, tak ju ignoruje. Po prijatí pošle potvrdenie, že klient môže posielať súbory a správy pre server.

Po začatí spojenia môže reagovať na správy o ukončení, o prijatí súboru a správy. Pred každým súborom je poslaná hlavička s informáciou o názve súboru a počte fragmentov, ktoré má server prijať. Tie následne slúžia na kontrolu či všetky prišli. Server si pred posielaním súboru vypýta cestu, kde má súbor uložiť, názov je použitý ten istý ako bol u klienta.

Keď prišiel názov súboru, je server pripravený prijímať fragmenty. V prípade súboru dáta sa ukladajú do bajtového reťazca, ktorý je po prijatí všetkých zapísaný do novovytvoreného binárneho súboru. Po prijatí celého súboru je vypísaná do konzoly absolútna cesta, kde je uložený.

Pre správu nie je oddelený flag na začiatok správy, iba v prvom fragmente sa nachádza počet fragmentov, v zvyšných už pokračuje správa. Na rozdiel od súboru je najskôr dekódovaná podľa ascii na obyčajný reťazec a potom pridaná ku zvyšku správy, ktorý prišiel pred tým. Na konci je vypísaná celá správa.

Po prijatí každého fragmentu je skontrolovaná správnosť dát pomocou CRC, ktoré sa znova vypočíta z prijatých dát a porovná sa s hodnotou, ktorá je uložená v hlavičke. Ak sa hodnoty zhodujú, je fragment správne poslaný a môže byť prijatý ďalší. Ak nie, server požiada o poslanie fragmentu ešte raz.

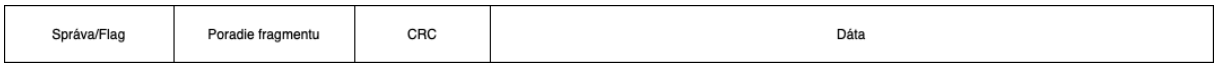
Keď príde správa od klienta, že všetky fragmenty boli poslané. Zapíšu sa údaje do súboru alebo do reťazca a vypíšu sa. Zhodnotí či boli všetky fragmenty úspešne poslané a odošle odpoveď klientovi.

Spojenie zo strany môže byť ukončené ak vyprší zadaný čas, ktorý server prijíma správy od klienta alebo ak klient požiada o ukončenie a server pošle kladnú odpoveď. V prípade, ak sa používateľ nachádza v klientskom menu, klient posiela správy o udržaní spojenia, ktoré sa vypíšu na obrazovku.

V prípade že funkcia serveru je ukončená, používateľ sa vráti do menu, kde si môže odznova zvoliť či chce spustiť program ako klient alebo znova server. Tiež mu je ponúknutá možnosť zmeniť IP adresu alebo ukončiť program.

3.3. Zmeny oproti návrhu

Zmeny oproti návrhu nastali v hlavičke, kde niektoré z uvedených informácii o fragmente nebolo potrebné použiť. Potrebné ostali iba správa alebo flag, ktorý určuje o aký typ fragmentu ide. Poradie fragmentu alebo počet fragmentov, ak je to poslaný fragment o základných údajoch posielanej správy alebo súboru a CRC.



Veľkosti ostávajú pre flag rovnaké zmenili sa len pre poradie fragmentu kvôli tomu, aby bolo možné v ňom zapísať aj väčšie čísla pre prípad, že sa posiela veľký súbor a nastavená veľkosť fragmentu je malá, čím môže vzniknúť veľké množstvo fragmentov. V CRC je zapísaná osem bajtová hodnota.

Veľkosť hlavičky teda je:
Správa: 1 bajt
Poradie fragmentu: 4 bajty
CRC: 8 bajtov.
Spolu 13 bajtov

Typy správ a teda fragmentov, ktoré dokáže program spracovať:
B – začiatok spojenia
C – úspešne začaté spojenie
E – žiadosť o ukončenie spojenia

F- spojenie úspešne ukončené
S – poslaný fragment
D – prijatý fragment úspešne
U – neúspešne prijatý fragment
M - správa
N – meno súboru
K – keep alive správa
A – keep alive správa prijatá úspešne
T – timeout
W – zapíše do súboru všetko, čo bolo poslané
P – vypíše správu, keď všetko bolo poslané
L – jeden alebo viac fragmentov bolo stratených
O – poslanie prebehlo úspešne

Malé zmeny boli spravené aj v diagramoch. V diagrame spracovávania prebehne ešte na konci výmena správ medzi klientom a serverom o tom, či boli všetky fragmenty odoslané a server pošle odpoveď, či prišli všetky.

Vo vývojovom diagrame nastala zmena iba v tom, že súbor je rozfragmentovaný po začatí spojenia a nie pred tým z dôvodu, že informáciu o tom, čo sa ide posielať a koľko nastane až po začatí spojenia, keď používateľ dostane prístup k druhému menu.

3.4. Ukážka komunikácie

Server:

```
Run as:
1.Client
2.Server
3.Change server ip address
4.End
2
UDP server up and listening
Inicialization of connection
Sending message, Number of fragments:  3
Fragmant size: 5
Fragment 1 was not sent successfully, requesting fragment again
Fragment 1 was sent successfully
Fragment 2 was sent successfully
Fragment 3 was sent successfully
Message was sent successfully.
Message:
hello world!
Keep Alive
Reqest to end connection from Client
Sending back confirmation about ending connection
Run as:
1.Client
2.Server
3.Change server ip address
4.End
```

Klient:

```
Run as:
1.Client
2.Server
3.Change server ip address
4.End
1
Starting Client program
Sending message to server to start connection
Starting connection successfully
1. Send file
2. Send message
3. End
2
Enter max. fragment size (1-1459): 5
Message to send:
hello world!
Type: Message, number of fragments: 3
Fragment size: 5
Number of fragments delivered successfuly
Sending fragment N. 1
Fragment was not delivered successfully
Sending fragment N. 1
Fragment delivered successfully
Sending fragment N. 2
Fragment delivered successfully
Sending fragment N. 3
Fragment delivered successfully
Message was sent successfully
1. Send file
2. Send message
3. End
3
Sending request to end connection
Connection ended successfully
```