Slovenská technická univerzita
Fakulta informatiky a informačných technológií
Ilkovičova 3, 842 19 Bratislava 4

Jakub Ivácek

Zadanie PKS
Komunikácia s využitím UDP

Študijný program: Informatika
Ročník: 3.
Predmet: **Počítačové a komunikačné siete**
Ak. rok: 2023/24

Obsah

Zadanie

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 znovuvyžiadanie chybných fragmentov, vrátane pozitívneho aj negatívneho potvrdenia. Po zapnutí programu, komunikátor automaticky odosiela paket pre udržanie spojenia každých 5s pokiaľ používateľ neukončí spojenie ručne. Odporúčame riešiť cez vlastne definované signalizačné správy a samostatné vlákno.

Návrh

Hodnotí sa prehľadnosť a zrozumiteľnosť odovzdanej dokumentácie ako aj kvalita navrhovaného riešenia. 5 bodov získa študent, ktorý má v dokumentácií uvedené všetky podstatné informácie o fungovaní jeho programu. Korektne navrhnutú štruktúru hlavičky vlastného protokolu, opis použitej metódy kontrolnej sumy a fungovania ARQ, metódy pre udržanie spojenia, diagram spracovávania komunikácie na oboch uzloch (sekvenčný), popis jednotlivých častí zdrojového kódu (knižnice, triedy, metódy, ...). Podčiarknuté požiadavky sú minimálne.

Návrh

Program Klient Server

Na začiatku sa bude dať zvoliť či chcem byť server alebo klient . Tieto role sa budú dať vymeniť odhlásením a navolením druhej .

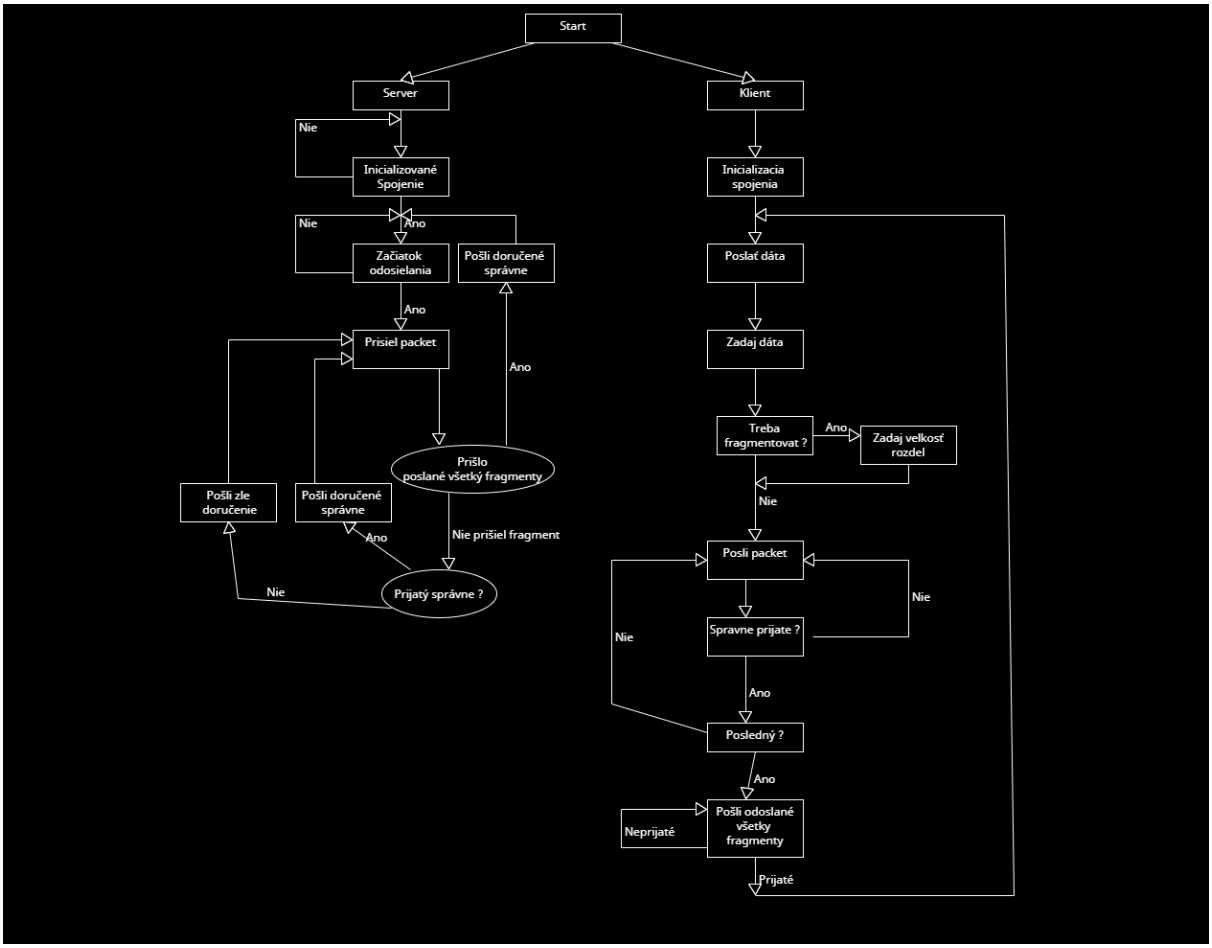
Server

Keď si vyberiem server -> používateľ nastaví ip, port , Server čaká kým nepríde správa , ktorá inicializuje spojenie od klienta . Po tomto sa bude udržiavať keep-alive thread . Potom server čaká na oznámenie o posielaní dát , správy . Fragmenty sa kontrolujú po jednom a to tak že server dostane fragment a odpovie či bol doručený správne alebo nesprávne , keď tak sú fragmenty znovu poslané . Ak už sú všetky fragmenty poslané a prešli kontrolou CRC budú fragmenty spojené a uložené ako súbor alebo správa .Potom klient ukončuje odosielanie tak , že pošle správu odoslané všetky dáta (po tomto server už nebude čakať ďalšie dáta) a odpovie posledným doručením správne. Po skončení prenosu server znovu čaká na začiatok odosielania .

Klient

Keď si vyberiem klient -> používateľ nastaví ip, port , Potom sa inicializuje spojenie a keep-alive thread . Potom si klient môže zvoliť či chce poslať súbor alebo správu . Ak si vyberie súbor zadá aj cestu k súboru čo chce odoslať a kde ho chce uložiť a používateľ ešte zadá max veľkosť fragmentu. Potom Klient pošle správu o začiatku odosielania a čaká kým Server odpovie a začne sa odosielanie . Kontrola a posielanie fragmentov boli vysvetlené už vyššie . Po odoslaní všetkých dát pošle klient Odoslané všetky fragmenty na čo server odpovie posledný krát a odosielanie sa ukončí. Potom sa klientovi navrhne odoslanie správ alebo súborov znovu alebo ukončenie spojenia . Ak zvolí ukončenie spojenia tak sa pošle info Serveru a ukončí sa spojenie .

Diagram spracovania komunikácie



Návrh hlavičky

TYP	VEĽKOSŤ	DÁTA	CRC

Typ 1 B

0001 - Inicializácia spojenia

0010 – Posielanie dát

0011 – Posielanie správy

0100 – Doručené dáta správne(ACK)

0110 – Doručené dáta nesprávne

0111 – Odoslané všetky fragmenty

1000 - Dáta

1100 – Ukončenie spojenia

Veľkosť (2 B) -> veľkosť fragmentu

CRC (1 B) -> zvyšok po delení polynómom

Spolu 4B + IP head (20B) + UDP head (8B) = 32B

Checksum metóda

Ako checksum budem používať CRC ktorý funguje tak , že budem používať polynóm ako deliteľ , ktorý bude dostupný aj pre klienta a aj server napr. Polynóm x ^3 + x ^2 + 1 vygeneruje kľuč 1101 .

Odosielateľ najskôr pridá k-1 núl na koniec dát (Kde k = počtu bitov kľúča polynómu). Použije modulo-2 bin delenie na vydelenie dát kľúčom polynómu a zvyšok po delení sa pripevní na koniec dát .

Prijímateľ najskôr použije modulo-2 bin delenie dát kľúčom a pozrie či zvyšok po delení budú nuly ak hej tak nedošlo k žiadnemu erroru a dáta boli prijaté správne.

ARQ metóda

Stop and Wait ARQ

Funkcia ARQ je na kontrolu či boli pri komunikácií parkety dobre odoslané , prijaté . Stop and Wait ARQ funguje tak , že odosielať pošle packet dát a čaká kým obdrží potvrdenie o prijatí od príjemcu dát a až potom posiela ďalšie dáta. Ak do určeného času dáta neprídu server pošle doručenie nesprávne a dáta sa pošlú znovu inak pošle ak boli doručené ack (doručené správne) a klient sa môže posunúť na ďalšie dáta. Ak boli dáta zle doručené , ľahko sa zistí error a packet sa pošle znovu . Nevýhoda je že táto metóda neefektívna lebo čaká na potvrdenie po každom packete.

Komunikácia

1. Odosielateľ -> Príjemca packet

2. Odosielateľ <- Príjemca (Dobre prijaté, Zle prijaté)

3. Poslanie znovu , alebo prechod na daľší packet

Udržanie spojenia

Udržanie spojenia bude prebiehať na druhom vlákne kde odosielateľ bude automaticky posielať Keep Alive paket každých 10s serveru a ten odpovie potvrdzovaciu správou Alive ak táto správa nepríde alebo odosielať zruší pripojenie ručne tak sa pripojenie ukončí .

Hlavička paket udržanie spojenia vlastné signalizačné správy

1bit

TYP

Typ 1 – 0 Keep Alive

Typ 2 – 1 Alive

Zmeny oproti návrhu

Návrh hlavičky

CRC	DÁTA

Moja nová hlavička , ktorá posiela dáta má len crc a dáta . Pred začatím posielania sa ešte serveru pošle počet fragmentov a názov súboru samostatne potom sa posielajú dáta v tejto forme crc + data.

Udržanie spojenia

TYP

Udržanie posiela viacej typov ako so mal naplánované

0 – keep Alive

1 - Exit

2 – Switch

3 – Posielanie správy

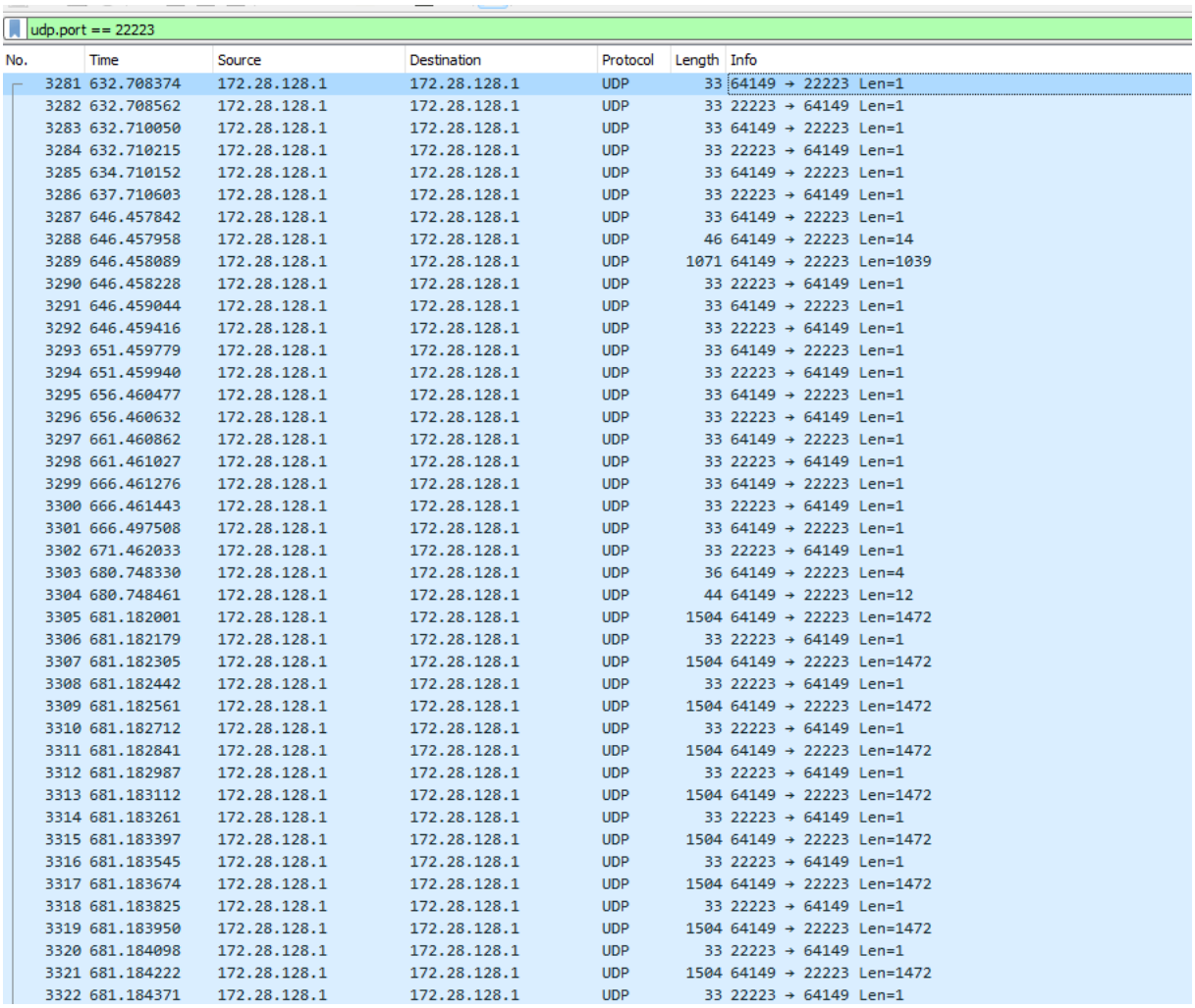
4 – Posielanie súboru

1, 2 , 3 , 4 sa posiela pre oznámenie druhej strany keď sa zaháji switch , send , exit

Rozhranie

```
Input : C for Client
 S for Server
 E for Exit
c
Login CLIENT
Input ip of server : 172.28.128.1
Input port of server: 23333
Connected to address: ('172.28.128.1', 23333)
 Client loop Enter : E for Exit , S for Switch , M for message send, F for file Send
Client KeepAlive from Server arrived conn. on
f
 Client loop Enter : E for Exit , S for Switch , M for message send, F for file Send
Send info to Server file transfer start
Received ack from Server start transfer
PRESS F TO CONTINUE
f
Enter file name : test_2MB.txt
Enter fragment size : max 1470 :1470
Enter 1 if u want bad packet receive : 1
File to send : test_2MB.txt2236998 B
Packet : 1 arrived wrong sending again
Packet : 1 arrived ok
Packet : 2 arrived ok
Packet : 3 arrived ok
Packet : 4 arrived ok
Packet : 5 arrived ok
Packet : 6 arrived ok
```

Simulácia wireshark



udp.port == 22223

No.	Time	Source	Destination	Protocol	Length	Info
3281	632.708374	172.28.128.1	172.28.128.1	UDP	33	64149 → 22223 Len=1
3282	632.708562	172.28.128.1	172.28.128.1	UDP	33	22223 → 64149 Len=1
3283	632.710050	172.28.128.1	172.28.128.1	UDP	33	64149 → 22223 Len=1
3284	632.710215	172.28.128.1	172.28.128.1	UDP	33	22223 → 64149 Len=1
3285	634.710152	172.28.128.1	172.28.128.1	UDP	33	64149 → 22223 Len=1
3286	637.710603	172.28.128.1	172.28.128.1	UDP	33	22223 → 64149 Len=1
3287	646.457842	172.28.128.1	172.28.128.1	UDP	33	64149 → 22223 Len=1
3288	646.457958	172.28.128.1	172.28.128.1	UDP	46	64149 → 22223 Len=14
3289	646.458089	172.28.128.1	172.28.128.1	UDP	1071	64149 → 22223 Len=1039
3290	646.458228	172.28.128.1	172.28.128.1	UDP	33	22223 → 64149 Len=1
3291	646.459044	172.28.128.1	172.28.128.1	UDP	33	64149 → 22223 Len=1
3292	646.459416	172.28.128.1	172.28.128.1	UDP	33	22223 → 64149 Len=1
3293	651.459779	172.28.128.1	172.28.128.1	UDP	33	64149 → 22223 Len=1
3294	651.459940	172.28.128.1	172.28.128.1	UDP	33	22223 → 64149 Len=1
3295	656.460477	172.28.128.1	172.28.128.1	UDP	33	64149 → 22223 Len=1
3296	656.460632	172.28.128.1	172.28.128.1	UDP	33	22223 → 64149 Len=1
3297	661.460862	172.28.128.1	172.28.128.1	UDP	33	64149 → 22223 Len=1
3298	661.461027	172.28.128.1	172.28.128.1	UDP	33	22223 → 64149 Len=1
3299	666.461276	172.28.128.1	172.28.128.1	UDP	33	64149 → 22223 Len=1
3300	666.461443	172.28.128.1	172.28.128.1	UDP	33	22223 → 64149 Len=1
3301	666.497508	172.28.128.1	172.28.128.1	UDP	33	64149 → 22223 Len=1
3302	671.462033	172.28.128.1	172.28.128.1	UDP	33	22223 → 64149 Len=1
3303	680.748330	172.28.128.1	172.28.128.1	UDP	36	64149 → 22223 Len=4
3304	680.748461	172.28.128.1	172.28.128.1	UDP	44	64149 → 22223 Len=12
3305	681.182001	172.28.128.1	172.28.128.1	UDP	1504	64149 → 22223 Len=1472
3306	681.182179	172.28.128.1	172.28.128.1	UDP	33	22223 → 64149 Len=1
3307	681.182305	172.28.128.1	172.28.128.1	UDP	1504	64149 → 22223 Len=1472
3308	681.182442	172.28.128.1	172.28.128.1	UDP	33	22223 → 64149 Len=1
3309	681.182561	172.28.128.1	172.28.128.1	UDP	1504	64149 → 22223 Len=1472
3310	681.182712	172.28.128.1	172.28.128.1	UDP	33	22223 → 64149 Len=1
3311	681.182841	172.28.128.1	172.28.128.1	UDP	1504	64149 → 22223 Len=1472
3312	681.182987	172.28.128.1	172.28.128.1	UDP	33	22223 → 64149 Len=1
3313	681.183112	172.28.128.1	172.28.128.1	UDP	1504	64149 → 22223 Len=1472
3314	681.183261	172.28.128.1	172.28.128.1	UDP	33	22223 → 64149 Len=1
3315	681.183397	172.28.128.1	172.28.128.1	UDP	1504	64149 → 22223 Len=1472
3316	681.183545	172.28.128.1	172.28.128.1	UDP	33	22223 → 64149 Len=1
3317	681.183674	172.28.128.1	172.28.128.1	UDP	1504	64149 → 22223 Len=1472
3318	681.183825	172.28.128.1	172.28.128.1	UDP	33	22223 → 64149 Len=1
3319	681.183950	172.28.128.1	172.28.128.1	UDP	1504	64149 → 22223 Len=1472
3320	681.184098	172.28.128.1	172.28.128.1	UDP	33	22223 → 64149 Len=1
3321	681.184222	172.28.128.1	172.28.128.1	UDP	1504	64149 → 22223 Len=1472
3322	681.184371	172.28.128.1	172.28.128.1	UDP	33	22223 → 64149 Len=1

3289 – 1 fragment send

3305 – 2MB file start send

Príklad wireshark zachytená komunikácia .

Záver

Program je rozdelený na dve časti - jednu na prijímanie a druhú na odosielanie. Po úspešnom pripojení môžu užívatelia komunikovať a vymieňať si dáta alebo si vymeniť pozície . Program najskôr

spracuje vstupnú správu alebo súbor. Po spracovaní sa dáta rozdelia na fragmenty a pridajú sa hlavičky tieto fragmenty sa ďalej odosielajú cez sieť. Po prijatí týchto fragmentov sa kontroluje ich správnosť a v prípade chyby sa požiada o poslanie znovu . Nakoniec sa tieto fragmenty spoja do jednej správy alebo súboru na konci komunikácie. A uložia alebo vypíšu