Peter Kúdela

PKS

Zadanie 2 (Final)

Komunikácia s využitím UDP protokolu

Fiit

02.12.2020

1. Návrh riešenia – pred implementáciou

Header:
Hlavička nášho protokolu bude obsahovať:
Checksum - 1 bajty slúžiaci na kontrolu správnosti poslaného fragment
Číslo fragmentu - 4 bajty držiace poradie fragmentu
Celkový počet fragmentov - 4 bajty
Typ správy - 1 bajt reprezentujúci či je správa txt, file, switch (server/client)

Client/server switch:
Na prepínanie funkcie bez reštartu programu budem využívať pole typ správy v hlavičke packetu, ak takúto správu client odošle zmení sa funkcionalita programu na server, nasledovne ak server túto správu príjme zmení svoju funkcionalitu na client.

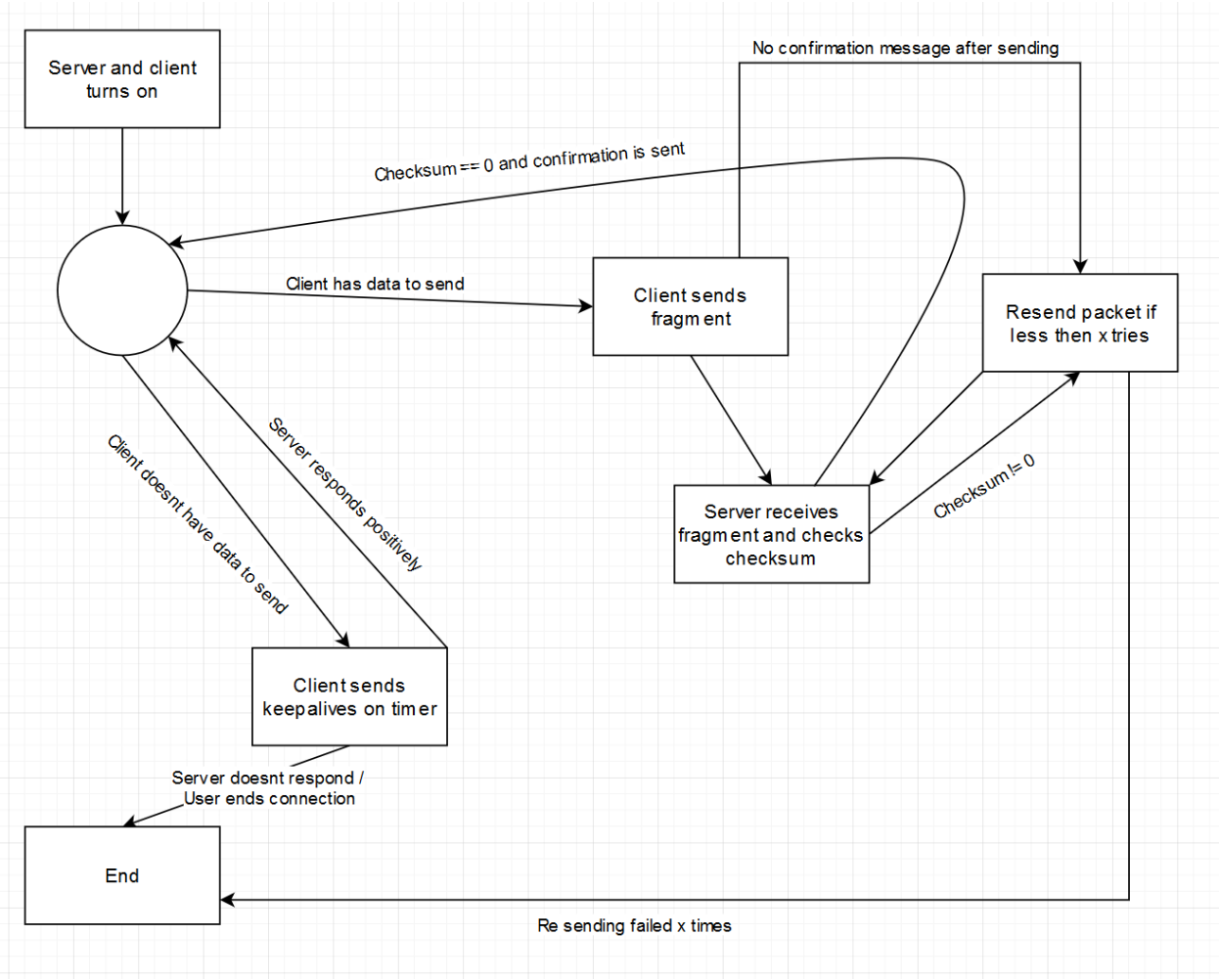
Checksum:
Client side:
Dáta spolu s našou hlavičkou rozdelíme na segmenty dĺžky n bitov, tie nasledovne sčítame (pokiaľ je výsledné číslo dlhšie ako n bitov zvyšné bity pričítame), výsledok ako komplement 1 zapíšeme do hlavičky nášho packetu a odosielame.

Server side:
Proces opakujeme, pripočítavame však aj checksum hodnotu prenesenú v hlavičke, ak nám komplement jednej dáva 0, vieme že data sa preniesli správne.

ARQ:
V prípade že checksum kontrola výde 0 server pošle naspeť acknowledgment správu že dáta prijal správne, v inom prípade pošle správu že nastala chyba, v takomto prípade client posiela packet opätovne (Packety posiela po jednom, ďalší packet sa neodošle kým sa predošlí neodošle správne), toto sa opakuje po limit počtu opakovaní

Udržanie spojenia:
Po úspešnom odoslaní súboru spojenie udržiavame keepalive packetmy ktoré posielame každých x sekúnd (čas bude špecifikovaný v kóde)

Implementácia:
Program bude implementovaný v jazyku python pomocou modulu socket



Obr. 1 Diagram komunikácie na oboch uzloch

2. Interface

Skript sa správa ako klient aj server zároveň, preto je potrebné nastaviť aj server port aj klient port.

Po spustení skriptu je užívateľ vyzvaný na zadanie ip adressy, portu z ktorého bude klient správy odosielať a portu na ktorom bude server počúvať. Ďalej sa užívateľa skript pýta na aký typ správy odosiela a maximálnu veľkosť fragmentu (maximum nastavené na 1472 bytov), text správa bude vypísaná v termináli na prijímajúcom uzli. Pre poslanie súboru vložíme jeho názov, súbor sa

musí nachádzať v rovnakom mieste ako skript, súbor je potom odoslaný a vytvorený na prijímajúcom uzle v adresári kde sa nachádza jeho skript.

3. Implementácia

Server a klient bežia v rovnakom skripte na dvoch threadoch.

3.1 Checksum

Hodnota checksum počítame spočítaním všetkých bytov v hlavičke aj správe, pokiaľ hodnota prevýši 255 nadbytočný bit pričítame k celkovej sume.

3.2 Server

Na začiatku sa server naviaže na ip a port vložený na začiatku programu. Server potom ďalej čaká kým mu nebude odoslaná správa, pri prijatí správy skontroluje či bola správa prijatá správne, ak nie požiada o jej opätovné poslanie, podľa typu správy (zisteného z hlavičky) pripočítavame jej obsah do buď do byte arrayu alebo do prázdnej reťaze znakov. Pokiaľ bola správa rozložená na viacero fragmentov, proces opakujeme, končime ak server príjme posledný fragment (číslo fragmentov v hlavičke).

Pokiaľ sa typ správy rovná inej hodnote ako jednej z definovaných (0-5) server skončí aktivitu.

3.3 Client

Používateľ je vyzvaný vybrať druh správy z nasledujúcich množných:

0- Textová správa (Zobrazí sa v terminály prijímajúceho uzla)
1- Súbor (adresy brané z adries skriptov)
3- Ukončí funkciu klienta
Other- ukončí funkciu servera na prijímajúcom uzle

Po zadaní správy klient zistí či je potrebné ju delíme na fragmenty ktoré nasledovne spájame s našou hlavičkou a posielame. Pri každom odoslanom fragmente čakáme na odpoveď od servera, pokiaľ príde správa, že fragment bol pri prenose poškodený znovu odošleme ten istý fragment.

Pokiaľ klient posiela súbor, prvý fragment bude názov súboru.

3.3.1 Simulácia chyby

Skript je nastavený tak, že prvý poslaný fragment z klienta bude chybný, všetky ostatné odoslané fragmenty fungujú správne.

4. Návrh – zmeny

Hlavička:
Checksum – 1Byte
Typ správy – 1Byte
Počet fragmentov – 2Bytes

Client/server switch:

Switch medzi funkcionalitou nebol implementovaný, namiesto neho skript funguje naraz ako server aj ako klient.

ARQ:

Nebol implementovaný limit počtu opakovaní.

Keepalive:

Neúplná implementácia, keepalive packety sú posielané každých 60 sekúnd po úspešnom odoslaní správy. Nevypisuje sa však správa o prerušení spojenia.