Počítačové a komunikačné siete

**Komunikácia s využitím UDP protokolu - návrh**

Emma Macháčová

**Meno cvičiaceho** : Ing. Lukáš Mastiľak

**Čas cvičení** : Štvrtok, 16:00

**Dátum vytvorenia** : 23. Nov. 2021

Obsah

[1. Cieľ 1](#_Toc88565513)

[2. Návrh programu 2](#_Toc88565514)

[Opis programu 2](#_Toc88565515)

[Návrh vlastného protokolu a štruktúra hlavičky 2](#_Toc88565516)

[Opis použitej metódy kontrolnej sumy 2](#_Toc88565517)

[Fungovanie ARQ metódy pre udržanie spojenia 2](#_Toc88565518)

[Opis a diagram spracúvania komunikácie 2](#_Toc88565519)

1. Cieľ

Cieľom projektu je navrhnúť a implementovať program s použitím **vlastného protokolu** ad protokolom UDP transportnej vrstvy sieťového modelu TCP/IP. Program má umožniť komunikáciu dvoch uzlov **v lokálnej sieti Ethernet** (textových správ a súborov).

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 bude **obsahovať kontrolu chýb** pri komunikácii a znovu-vyžiadanie 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.

2. Návrh programu

Opis programu

Program bude implementovaný v jazyku **Python** (ver. 3.9), vo vývojovom prostredí PyCharm (2021.2.2 Professional Edition) s využitím knižníc na prácu s UDP socket (python modul socket).

Program bude fungovať **peer-to-peer**, a teda oba uzly budú schopné zároveň prijímať aj vysielať. Rozdieľ medzi uzlami bude ten, že jeden z nich bude plniť funkciu servera, a teda bude hosťovať spojenie. Používateľovi bude umožnené určiť cieľovú IP a port, a taktiež maximálnu veľkosť fragmentu a pri posielaní správy dáta rozložia podľa požadovanej veľkosti fragmentov.

Obe komunikujúce strany **budú schopné zobrazovať**:

* názov a absolútnu cestu k súboru,
* veľkosť a počet fragmentov

Program bude taktiež schopný simulovať chybu prenosu, a to dodatočným zmenením posielaných dát (po vypočítaní kontrolnej sumy).

Návrh vlastného protokolu a štruktúra hlavičky

Program bude využívať vlastné hlavičky, ktoré pomocou enkapsulácie pridá k už existujúcim dátam packetu. Enkapsulácia sa celkovo koná na štyroch úrovniach – a to pri vlastnom protokole, protokoloch UDP, IP a Ethernet.

Nakoľko je UDP nespoľahlivý protokol, musíme rátať s možnosťou, že sa niektoré packety pri prenášaní stratia, alebo že neprídu v správnom poradí.

Maximálna veľkosť fragmentu bude 1500 bajtov - IP hlavička - UDP hlavička - vlastná hlavička, čo bude cca 1465 bajtov.

Opis použitej metódy kontrolnej sumy

Kontrolná suma sa bude počítať nasledovne:

hash = 0

for i in data:

hash += data[i] % PRVOCISLO

return hash

Fungovanie ARQ metódy pre udržanie spojenia

Táto metóda zabezpečuje to, aby sa klient odpojil a ukončil spojenie v prípade, že server prestane počúvať po poslednej správe.

Klient potrebuje vedieť, že server stále funguje a počúva. Preto v určitom časovom rozmedzí (x sekúnd) bude posielať serveru správu na kontrolu, či stále počúva.

Ak sa keep-alive pošle určitý počet krát a server neodpovie, klient preruší spojenie lebo predpokladá, že server už nepočúva, a vypíše sa chybová hláška.

Opis a diagram spracúvania komunikácie

Na začiatku si používateľ vyberie to, či chce byť server, alebo klient. Nastavia sa IP a port, a veľkosť fragmentu.

* server – hosťuje socket, vytvorí server
  + prijíma a odosiela správy
* klient – pripojí sa na server
  + prijíma a odosiela správy
  + má na starosti keep-alive (ak server nemá s kým komunikovať, vypne sa)

Súbory (textové alebo iné) sa budú spracúvať po bajtoch.

Na začiatku sa vytvorí úvodná správa z tej strany, ktorá posiela dáta. Bude obsahovať CRC, údaje o súbore, počet packetov a názov. Po odoslaní úvodnej správy sa bude očakávať potvrdenie začatie komunikácie.

Na túto správu sa dostane odpoveď – potvrdenie komunikácie – a môžeme začať odosielať packety.

Packety sa začnú postupne posielať, a po každých x packetoch sa bude očakávať potvrdenie prijatia. Po prijatí tejto potvrdzujúcej správy sa bude kontrolovať to, či prišli všetky packety správne (pomocou CRC), a ak nie, tieto sa odošlú znova (chýbajúce a poškodené – selective repeat).

Takto sa to bude opakovať, kým nebudú odoslané úspešne všetky packety.

Na záver sa pošle odpoveď, že všetko prebehlo úspešne a súbor sa uloží.

**Chybný fragment**

Možnosť odoslať chybný fragment bude pri odosielaní súboru – používateľ bude mať na výber, či má dochádzať ku korupcii fragmentov.