Slovenská technická univerzita v Bratislave
Fakulta informatiky a informačných technológií

Počítačové a komunikačné siete

Zadanie č.2

UDP komunikátor - návrh

Akademický rok 2022/2023

Meno: Ján Ágh
Cvičiaci: Ing. Lukáš Mastiľak

Dátum: 30.11.2022
Počet strán: 6

Obsah

1 Stručný opis problematiky a zadania

Cieľom zadania je navrhnúť program umožňujúci komunikáciu dvoch zariadení v lokálnej sieti typu Ethernet s využitím vlastného protokolu nad protokolom User Datagram Protocol (UDP), ktorý pracuje na transportnej vrstve sieťového modelu TCP/IP.

Program sa skladá z dvoch častí, a to vysielacej (klient) a prijímacej (server), a umožňuje používateľovi preposielať jednoduché textové správy a súbory ľubovoľného formátu. Používateľ má taktiež možnosť zadefinovať si maximálnu veľkosť fragmentu, ktorá nesmie presiahnuť veľkosť 1500B, aby nedochádzalo k opätovnej fragmentácii na datalinkovej vrstve.

Program obsahuje kontrolu chýb pri komunikácii a znovuvyžiadanie chybných fragmentov, pričom na túto skutočnosť využíva signalizačné správy odosielané počas komunikácie. Počas doby trvania komunikácie jednotlivé strany v intervale každých 5 sekúnd odosielajú paket slúžiaci na udržiavanie spojenia druhej strane dovtedy, kým sa používateľ nerozhodne manuálne komunikáciu prerušiť alebo nevznikne nečakaná chyba.

2 Navrhnuté riešenie

2.1 Organizácia projektu

Program pozostáva z niekoľkých základných súborov, ktoré sú nevyhnutné pre jeho funkčnosť. Každý zo spomínaných súborov plní istú úlohu počas vykonávania programu.

Prvým súborom je *interface.py* nachádzajúci sa v adresári *src.* Tento súbor obsahuje statickú triedu Interface, ktorá slúži na poskytovanie spätnej väzby používateľovi pomocou konzolových výpisov a taktiež akceptovanie vstupných informácií. Tvoria ju metódy *initial_setup()* na prvotné nastavenie programu (výber možnosti klient – server), *initialize_client()* a *initialize_server()* na zobrazenie menu a akceptovanie vstupu patriaceho ku klientskej, resp. serverovskej časti programu a ďalšie, menej významné a pomocné metódy.

Dvojica súborov *client.py* a *server.py* v adresári *src* obsahujú triedy s konkrétnymi implementáciami príslušných častí programu – klientskej časti v triede *Client* a serverovskej časti v triede *Server*.

Súbor *general.py* a v ňom sídliaca trieda *General* predstavuje rodičovskú triedu pre *Client* aj *Server* a obsahuje dôležité metódy využívané oboma týmito triedami.

Posledným a zároveň najdôležitejším súborom je *main.py*, tvoriaci jadro celého programu a spájajúci zvyšné triedy a súbory do jedného celku. Ide o vstupný bod programu a v ňom sa vytvoria príslušné objekty tried *Server* a *Client*.

2.2 Návrh hlavičky vlastného protokolu

Súčasťou zadania je aj návrh vlastného protokolu, ktorý bude pracovať nad protokolom UDP. Keďže protokol UDP neodosiela žiadne potvrdenia týkajúce sa prijatia dát (a týmpádom druhá strana komunikácie nemá žiadny spôsob dozvedieť sa, či dáta prišli v poriadku a v správnom poradí), úlohou navrhnutého protokolu bude vyriešiť aj túto problematiku. Štruktúra protokolu je opísaná nižšie:

1. Typ paketu – informácia, o aký druh paketu sa jedná:

- **0** – paket *SYN*, označujúci začiatok nadväzovania spojenia
- **1** – paket *SYN+ACK*, odpoveď druhej strany na požiadavku spojenia
- **2 –** paket *ACK*, odpoveď na viacero požiadaviek (definitívne potvrdenie spojenia, odpoveď na keep-alive pakety, odpoveď na prijaté fragmenty…)
- **3** – paket *keep-alive* na overovanie konektivity druhej strany
- **4** – paket *FIN* na ukončenie spojenia
- **5** – paket *FIN+ACK* na odpoveď k požiadavke ukončenia spojenia
- **6** – paket slúžiaci na opätovné vyžiadanie dát
- **7** – paket na inicializáciu odosielania dát (obsahuje typ alebo názov s príponou súboru)
- **8 –** paket na odoslanie súboru alebo textovej správy
- **9** – paket na oznámenie zmeny rolí (z klienta sa stane server a naopak)

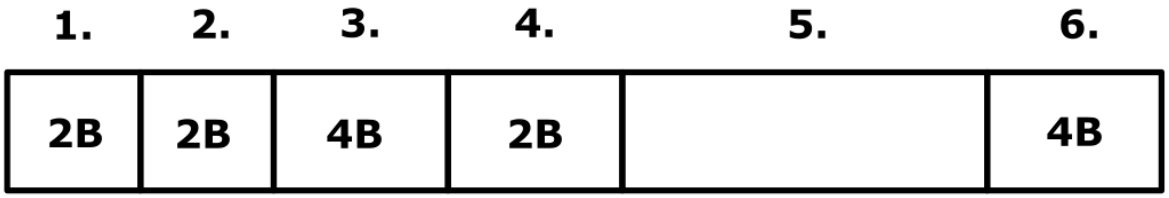
2. Identifikácia prenosu – každý process prenosu má vlastné identifikačné číslo

3. Poradové číslo – slúži na správne zoskupenie fragmentov na strane prijímateľa

4. Veľkosť – veľkosť prenášaných dát (bez ostatných položiek hlavičky)

5. Dáta – prenášané údaje

6. CRC – kontrolný súčet na detekciu chýb v prenášanom pakete

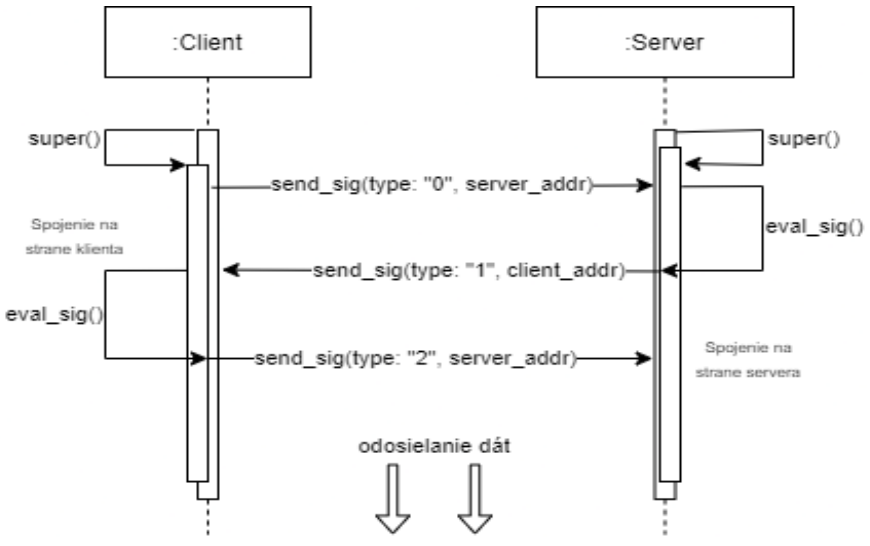


Obr. 2.1 Znázornenie hlavičky protokolu

2.3 Opis spôsobu inicializácie a ukončenia komunikácie

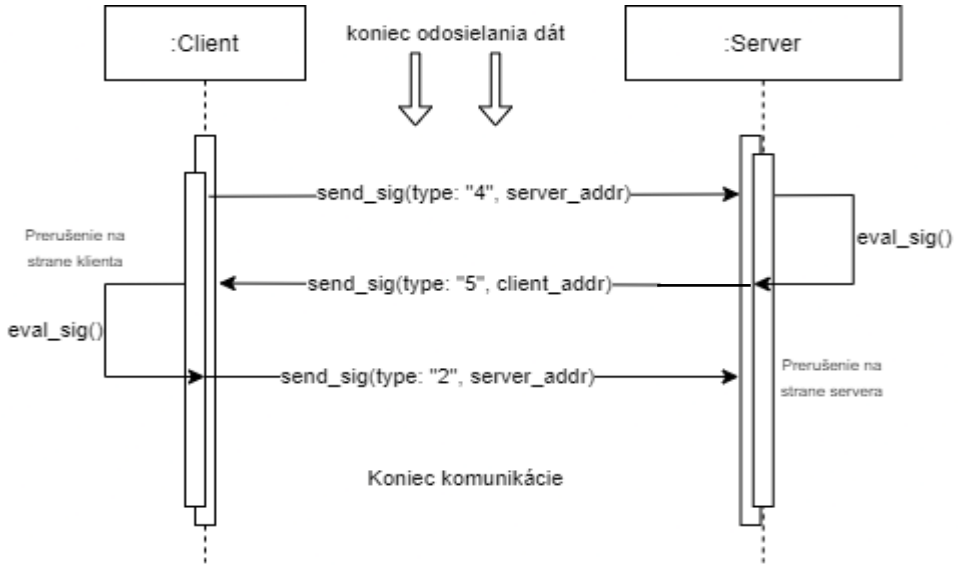
Pred začiatkom každej komunikácie / prenosu je z pohľadu zadania dôležité nadviazať spoľahlivé spojenie s prijímacou stranou. Keďže protokol User Datagram Protocol (UDP) pracuje nespoľahlivo (bez spojenia a overovania), o nadviazanie a rušenie spojenia sa stará vlastný prokotol, navrhnutý v predchádzajúcej podkapitole, pomocou špeciálnych druhov paketov uvedených v časti "Typ paketu".

O začiatok prvotnej inicializácie komunikácie sa stará odosielateľ (klient), ktorý sa snaží nadviazať spojenie so serverom formou *three-way handshake*. Úrobí tak odoslaním inicializačného paketu typu 0 – *SYN*. Akonáhle server identifikuje klientovu požiadavku, odošle mu späť paket typu 1 – *SYN+ACK* na potvrdenie akceptovania požiadavky spojenia (*ACK*) spolu s vlastnou inicializačnou požiadavkou (*SYN*). Po doručení uvedeného paketu sa komunikácia na strane klienta považuje už za nadviazanú, je však potrebné odoslať na stranu servera ešte jeden paket typu 2 – *ACK* na potvrdenia prijatia predchádzajúceho paketu. Komunikácia sa považuje za nadviazanú aj na strane servera až po prijatí tohto paketu. Po úspešnom uskutočnení *three-way handshake* je možné začať prenos dát odoslaním paketu 7.



Obr. 2.2 Sekvenčný diagram nadviazania spojenia

Po úspešnom prenose dát sa klient môže rozhodnúť oficiálne prerušiť komunikáciu. Začiatok prerušenia komunikácie odštartuje odoslaním paketu typu 4 – *FIN*, z ktorého sa server dozvie o snahe ukončiť spojenie. Úlohou servera je odoslať odpoveď vo forme paketu typu 5 – *FIN+ACK*, v ktorom oboznámi klienta o prijatí úvodného paketu typu 4 – *FIN* (*ACK*) a zároveň doručí vlastnú žiadosť o prerušenie komunikácie (*FIN*). Akonáhle klient prevezme spomínaný paket, považuje komunikáciu za úspešne prerušenú a v poslednom kroku odošle na server paket typu 2 – *ACK* na potvrdenie prijatia predchádzajúceho paketu. Server považuje spojenie za úspešne ukončené až po prijatí tohto potvrdzovacieho paketu.



Obr. 2.3 Sekvenčný diagram prerušenia spojenia

2.4 Opis metódy kontrolnej sumy

Výpočet aj evaluácia správnosti kontrolného súčtu pri každom odoslanom a prijatom pakete sa realizuje pomocou metódy *crc32()* pochádzajúcej z knižnice *zlib*. Uvedená metóda ako argument akceptuje dáta spolu s navrhnutou hlavičkou a vráti 32-bitovú celočíselnú hodnotu. Táto hodnota sa v uvedenom formáte vloží do hlavičky protokolu, paket dorazí na cieľovú adresu a tam sa opätovne uskutoční výpočet sumy s cieľom zistiť, či údaje dorazili bezchybne. Chybovosť sa overuje porovnaním nového výpočtu s hodnotou uloženou v hlavičke, pričom ak sa zhodujú, program pokračuje v ďalšej analýze paketu a odošle späť paket typu 2, ale ak nastane nezhoda indikujúca chybu, prijatý paket sa odstráni a odošle sa paket typu 5 na vyžiadanie opätovného odoslania údajov.

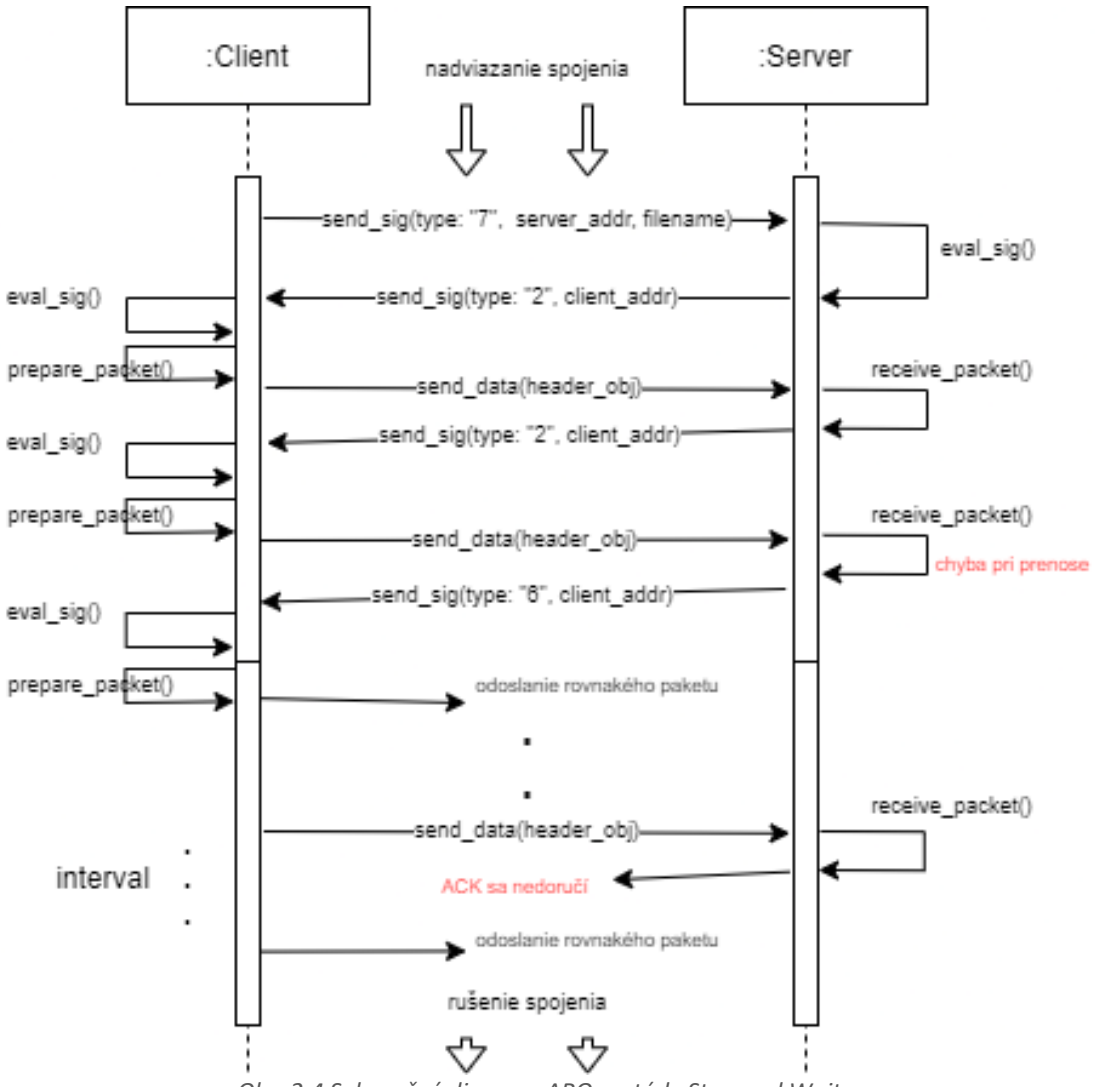
Uvedená metóda počas výpočtu kontrolného súčtu kombinuje dve operácie – binárny posun vľavo a následne operáciu XOR výsledku s binárnou hodnotou 1 0000 0100 1100 0001 0001 1101 1011 0111. Krátky opis samotného algoritmu by vyzeral nasledovne:

- *Obráť vstupnú hodnotu a priraď na jej koniec celkovo 32 núl*
- *Vykonaj operáciu XOR vstupu s hodnotou 0xFFFFFFFF*
- *Ak je prvý bit výsledku rovný 1, vykonaj XOR s vyššie písaným binárnym polynómom*
- *Vykonaj binárny posun celého výsledko vľavo a posuň číslo vpravo*
- *Opakuj kroky 3 a 4, dokým prvých 8 bitov výsledku nebude nulových*
- *Vykonaj operáciu XOR vstupu s hodnotou 0xFFFFFFFF a obráť výsledok*

2.5 Opis použitej ARQ metódy

ARQ (Automatic Repeat Quest) je rozsiahla skupina metód detekcie a korekcie chýb využívaných pri prenose dát. Pre zaistenie spoľahlivého prenosu dát po nespoľahlivom kanáli používa signály na potvrdenie príjmu. Úlohou príjemcu je indikovať odosielateľovi úspešné a bezchybné prijatie paketu odoslaním kladného potvrdenia, pričom ak odosielateľ neobdrží uvedené potvrdenie do doby uplynutia zadefinovaného časového intervalu, odošle daný paket opäť. Počet znovuodoslaní paketov je obmedzený a ak sa prekročí maximálny počet pokusov, nastane chyba v komunikácii a spojenie sa považuje za prerušené.

Komunikátor pre overovanie správnosti odosielania dát využíva ARQ metódu *Stop and Wait*, ktorej princíp je veľmi jednoduchý. Odosielateľ posiela pakety po jednom, pričom po každom odoslanom pakete očakáva signál kladného potvrdenia zo strany príjemcu. Nasledujúci paket je odoslaný až po prijatí spomenutého potvrdenia. Na druhej strane, od prijímateľa sa očakáva, že po prijatí paketu toto potvrdenie odošle. Samozrejme, ak prijatý paket obsahuje chybné údaje (zistené napr. tým, že vypočítaný kontrolný súčet sa nerovná súčtu uvedeného v hlavičke), namiesto jednoduchého potvrdenia prijatia sa odosielateľovi doručí paket typu 6, slúžiaci na znovuvyžiadanie predchádzajúceho paketu.



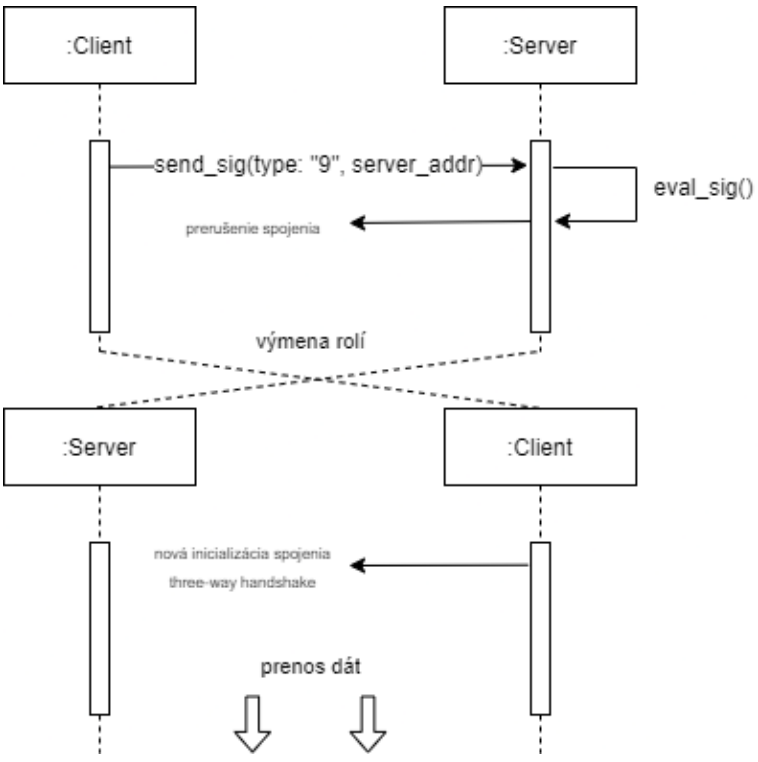
Obr. 2.4 Sekvenčný diagram ARQ metódy Stop and Wait

2.6 Opis metódy na udržiavanie spojenia

Odosielanie paketov typu 3 (na overovanie konektivity druhého komunikujúceho zariadenia) sa uskutočňuje v odlišnom vlákne ako primárna komunikácia, s cieľom predchádzať kolíziám. Uvedené pakety si navzájom posielajú obaja účastníci komunikácie v pravidelných intervaloch dĺžky 5 sekúnd, pričom po odoslaní sa očakáva aj odpoveď prichádzajúca z opačnej strany vo forme paketu typu 2 – *ACK*. Ak odpoveď nedorazí do piatich sekúnd, odošle sa ďalší paket typu 3 a ak ani na tento paket nepríde odpoveď, spojenie sa považuje za nefunkčné a automaticky sa preruší. Týmto by mali byť ošetrené situácie, kedy sa prvý paket typu 2 – *ACK* podarilo odoslať, ale nepodarilo sa ho doručiť (nedojde k automatickému ukončeniu spojenia).

2.7 Opis spôsobu zmeny úloh

Súčasťou programu je aj možnosť automatickej výmeny úloh medzi klientom a serverom. Proces výmeny úloh môže inicializovať hociktorý účastník komunikácie, či sa jedná o klienta, alebo o server. Základnou úlohou inicializačnej strany je doručiť druhému účastníkovi paket typu 9 – oznámenie o zmene rolí, pričom rozdiel oproti iným druhom komunikácie je, že sa tu neočakáva odpoveď druhej strany vo forme paketu typu 2 – *ACK*. Po odoslaní paketu si odosielateľ automaticky zmení svoju rolu, pričom akonáhle druhej strane dorazí paket typu 9, aj ona vykoná rovnaký krok. Dôležitou poznámkou je, že jednotlivé strany budú musieť po výmene úloh inicializovať novú komunikáciu spôsobom, aký je opísaný v podkapitole 2.3 (najprv, pred výmenou prerušiť existujúce spojenie správnym spôsobom, následne vymeniť úlohy a nakoniec inicializovať nové spojenie pomocou *three-way handshake*).



Obr. 2.5 Sekvenčný diagram výmeny úloh