Slovenská technická univerzita v Bratislave
Fakulta informatiky a informačných technológií

Počítačové a komunikačné siete

Zadanie 2 – návrh protokolu

Petra Miková

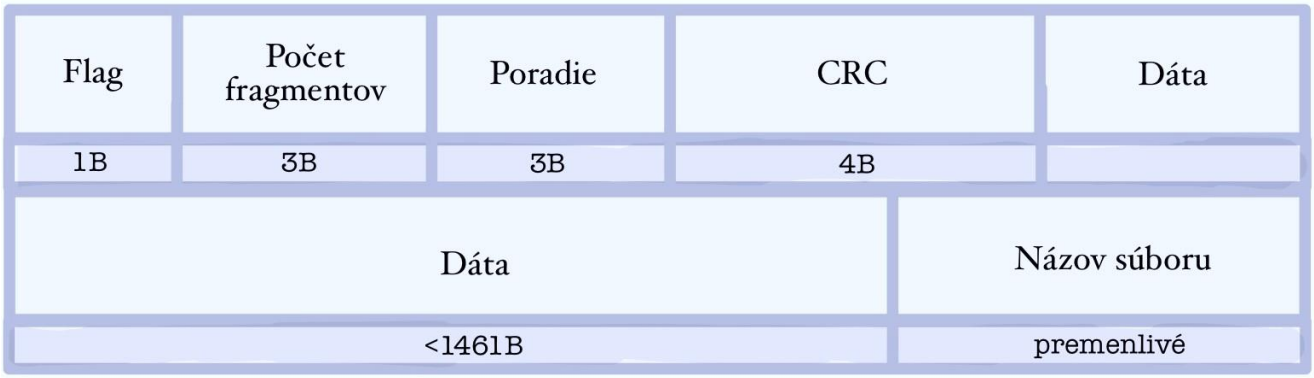
Petra Miková
ID: 120852

Obsah

Vlastná hlavička 1
Sekvenčný diagram 2
Slovný opis fungovania programu 3
- Server 3
- Klient 3
- Komunikácia 3
- Keepalive (udržiavanie spojenia) 3
- ARQ metóda Selective Repeat 3
- Checksum 4

Petra Miková
ID: 120852

Vlastná hlavička



Flag (1B): udržiava informáciu o odosielanom packete
1: nadviazanie spojenia
2: keep alive
3: prenos dát (správa)
4: prenos dát (súbor)
5: správny packet s dátami
6: nesprávny packet s dátami (chyba v CRC)
7: ukončenie spojenia

Počet fragmentov (3B): udržiava informáciu o počte packetov s dátami, ktoré sa budú posielať
Keďže chceme vedieť posielať 2MB súbor, musíme pre toto pole v hlavičke mať takú veľkosť, aby to bolo možné, aj keby klient chce dáta fragmentovať po jednom byte. Ak by to takto bolo, na prenesenie 2MB súboru by sme potrebovali 2^21 fragmentov – z čoho vyplýva že pre toto pole potrebujem 3B.
Toto pole je inicializované iba v inicializačnom packete o tom že sa budú prenášať dáta.

Poradie (3B): udržiava informáciu o poradí packetu odosielaného fragmentu
Toto pole je inicializované len pri prenose packetov s dátami.

CRC (checksum) (4B): udržiava informáciu o zvyšku po delení CRC metódou
Za použitia CRC-32 algoritmu bude táto hodnota o veľkosti 4B, takže toľko dáme aj pre toto pole do hlavičky.

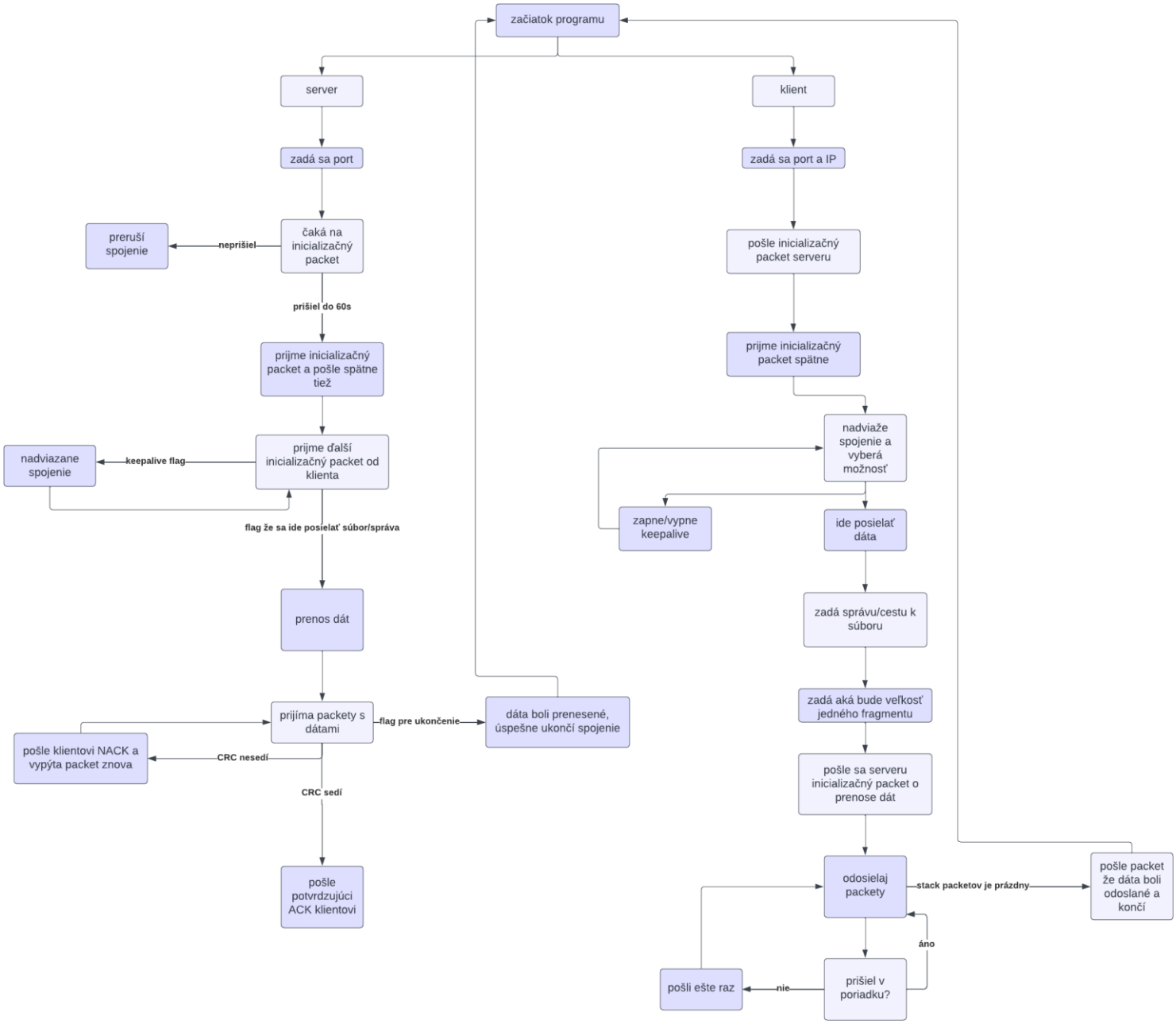
Dáta (<1461): udržiava samotné odosielané dáta
Keďže nechceme, aby došlo k fragmentácii na linkovej vrstve, maximálna veľkosť dát (teda jedného fragmentu) v packete musí byť 1500B – 20B (IP hlavička) – 8B (UDP defaultna hlavička) – 11B (moja hlavička) = 1461B. Tých 1500B je hodnota, ktorá predstavuje veľkosť dátovej časti v Ethernete.

Názov súboru: udržiava názov súboru ktorý sa bude posielať
Toto pole je inicializované len pri inicializačnom packete o tom že sa bude odosielať súbor.

Petra Miková
ID: 120852

Sekvenčný diagram

V tejto časti ukážem sekvenčný diagram fungovania programu. Tento proces fungovania sa samozrejme pri finálnej verzii môže zmeniť.



Petra Miková
ID: 120852

Slovný opis fungovania programu

Server

Vytvorím si server socket a nabindujem doň zadaný port. Zároveň si zadefinujem timeout, dokedy čakám na inicializáciu spojenia zo strany klienta. Ak od klienta príde inicializačný packet na nadviazanie spojenia, prijmem ho a pošlem mu ho spätne taktiež. Ak tento packet do 60 sekúnd nepríde, uzatváram spojenie s chybou nadviazania spojenia. Inak teda čakám na ďalší inicializačný packet od klienta, kde nám môže prísť buď požiadavka o keepalive, zmena funkcie, ukončenie, alebo teda už samotný prenos dát. Ak sme dostali ukončovací packet, uzavrieme spojenie.

Klient

Vytvorím si klient socket a vložím doň zadaný port a IP adresu. Pošlem serveru inicializačný packet a spätne prijmem taktiež jeden od serveru, kde čakám na inicializáciu z jeho strany. Ak k nej teda došlo, nadväzujem spojenie a to zaslaním inicializačného packetu s možnosťami, ktoré som už spomenula vyššie. Ak teda iniciuje klient prenos dát, začína komunikácia.

Komunikácia

Od klienta sa vypýta aký typ dát chce odosielať (správa/súbor), samotné dáta, a akú veľkosť fragmentu v jednom packete chce odosielať. Taktiež má možnosť si vybrať, či chce do prenosu zahrnúť aj chybné packety.

1. **Posielanie dát ARQ metódou Selective Repeat (client side)**
   V loope, ktorý končí až keď máme zásobník packetov prázdny, posielam fragmenty. Pre každý packet s fragmentom sa vypočíta CRC a posiela sa tento dátový packet serveru. Na základe odpovedí od serveru kontrolujeme, či všetky packety došli v poriadku, a ak nie, tak tie chybné posielame znova. Ak sa všetko úspešne odoslalo, pošleme serveru inicializačný packet o ukončení komunikácie. Metóda Selective Repeat bude opísaná hlbšie vo vlastnom odseku.
2. **Počúvanie dát zo strany servera (server side)**
   Znova v loope počúva a prijíma od klienta packety s dátami, kde rozhoduje či sú dobré alebo chybné (na základe toho že si na svojej strane vypočítava CRC a porovnáva ho s tým prijatým. Ak bol packet dobrý, posiela potvrdzujúci packet o úspešnom prenose fragmentu (ACK), a ak nie, posiela packet o chybnom packete (NACK) a žiada opätovné zaslanie. Ak server prijme ukončovací packet, pošle taktiež jeden spätne a ukončuje komunikáciu.

Keepalive (udržiavanie spojenia)

Ak si klient vyberie možnosť keepalive, bude sa každých 15 sekúnd posielať packet s flagom keepaliveu serveru, ktorý pre udržanie spojenia musí odpovedať naspäť. Ak odpoveď nedôjde, klient dedukuje ukončenie spojenia. Tak isto na strane servera, ako som už vyššie spomínala, mám nastavený timeout pri začiatku spojenia, kde ak do 60 sekúnd nedôjde zo strany klienta inicializačný packet, server ukončuje spojenie. Spojenie môže do tretice zaniknúť ešte explicitným ukončením zo strany klienta.

ARQ metóda Selective Repeat

Pri tejto metóde retransmittujeme len nesprávne packety, správne sa naďalej klasicky prijímajú a ukladajú do bufferu. Pre eventuálne prijatie fragmentovaných dát v správnom poradí si potrebujeme pre každý packet ktorý prenáša fragment dát udržiavať v hlavičke poradie fragmentu. Pri prijatí správneho packetu posiela server ACK, pri prijatí chybného posiela NACK, teda vieme jednoducho na strane klienta vidieť ktoré packety potrebujeme poslať znova. Samotná metóda funguje tak, že je zadefinované sliding window s určitou veľkosťou a poradie packetov usporiadane zostupne, a ak klient všetky packety z okna odoslal, server posiela ACK/NACK o tom packete, ktorý došiel ako prvý a

posúva okno o jeden index vľavo. Ak máme chybný packet, server pošle teda spomínaný NACK a klient pošle naspäť LEN tento jeden packet (na rozdiel od iných metód).

Checksum

Checksum je v skratke kontrolný súčet packetu s dátami hovoriaci o tom, či nedošlo k chybe pri prenose. V mojej implementácii som sa rozhodla použiť metódu CRC. CRC pracuje na bitovej úrovni a využíva predom definovaný polynóm na vytvorenie checksumu takým spôsobom, že aj samotné dáta berie ako polynóm a týmto preddefinovaným ho vydelí. Modulo z tohto delenia sa ukladá ako checksum. Generuje sa z hlavičky a dát na strane klienta, ale aj na strane serveru, ktorý následne kontroluje a porovnáva svoj vypočítaný a prijatý checksum od klienta. Na základe tohto vie rozpoznať chybné packety. V implementácii budem chybné packety mockovať tak, že nejakým spôsobom pozmením výsledný checksum na strane klienta.