Autor: xsviteka@stuba.sk
STU FIIT

Návrh protokolu pre P2P komunikáciu pomocou UDP(User Data Protocol)

Autor: xsviteka@stuba.sk
STU FIIT

Obsah

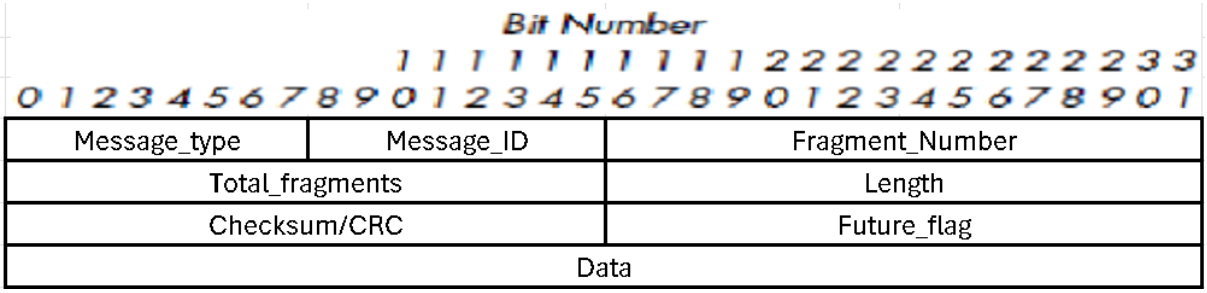
Autor: xsviteka@stuba.sk
STU FIIT

1. Zámer:

Zámerom tohto projektu je navrhnúť program a implementovať P2P(peer to peer)komunikáciu pre výmenu textových správ, súborov a iných informácii. Komunikácia bude sprostredkovaná pomocou vlastného protokolu nad UDP. Tento protokol bude obsahovať mechanizmus na prvotné nadviazanie spojenia medzi užívatelmi ako aj fragmentáciu správ, kontrolu, a udrziavanie spojenia.

Taktiež pri návrhu protokolu a implementácie v kóde sa budú simulovať poškodené dáta a následne sledovať ako sa kód zachová. Navrhnuty kód by mal byť schopný nahradiť stratené dáta pri prenose alebo snažit sa obnoviť spojenie ak sa simuluje jeho prerušenie. Kontrola odoslaných správ sa bude kontrolovať pomocou metódy stop and wait ako aj pomocov CRC 16 a metóda keep-alive sa bude starať o udržiavanie spojenia medzi dvoma klientami.(o týchto metódach nižšie)

2. Návrh hlavičky protokolu



Mnou navrhnutá hlavička nad UDP obsahuje tieto časti:

Message_type(1B): message type udáva, o aký typ správy ide. V mojom prípade budú tieto správy():

- 0x01 => nadviazanie spojenia medzi klientami(0b00000001)

- 0x02 => prenos textu(0b00000010)
- 0x03 => prenos súborov(0b00000011)
- 0x04 => keep alive(0b00000100)
- 0x05 => potvrdenie o prijatí fragmentu(ACK) (0b00000101)
- 0x06 => potvrdenie o neprijatí fragmentu(NACK) (0b00000110)
- 0x08 => správa na ukončenie spojenia(0b00000111)

Message ID (1 B): Unikátne identifikačné číslo priradené každej správe. Umožňuje rozlíšiť správu medzi inými a napríklad zistiť, či ide o duplikát, a zahodiť ju. Pomocou ID môžeme sledovať, či sa správa poslala alebo nie.

Fragment_Number (2 B): Obsahuje údaj, ktorý značí číslo fragmentu v poradí, čo pomáha pri určovaní chýb alebo pri správnom skladaní fragmentov do celku.

Total_fragments (2 B): Je údaj o celkovom počte fragmentov, ktoré sa majú poslať. Je dôležitý na kontrolu, či sa pošlú všetky fragmenty a či sa niektorý náhodou neposlal alebo sa stratil pri posielaní.

Length (2 B): Určuje veľkosť payloadu, preto klient vie, koľko bytov očakávať. Dĺžka pomáha aj pri určení dĺžky, ak ide o rôzne typy správ, ako sú súbory/texty alebo správy o kontrole.

Checksum (CRC16) (2 B): Opísané nižšie.

Future flag (2 B): Pri práci na zadaní som vy dosiel k zaveru ze tuto cas headera je mozne vynechat pretoze nebola zatial potrebna a ostala nevyuzita. Nahradil som ju teda castou s nazvom Future_flag ktora zatial nesplna ziadnu ulohu ale je pripravena na vyuzitie v buducnosti napriklad pri implementacii novych funkcii

3. Dáta

Veľkosť samotných dát, ktoré sa odosielajú druhému užívateľovi, je v počte 1500 B (bajtov). Od týchto bajtov musíme najskôr odpočítať IP hlavičku v počte 20 B (bajtov), potom UDP hlavičku v počte 8 B (bajtov) a nakoniec vytvorenú hlavičku v počte 12 B (bajtov). Výsledná veľkosť dát je teda 1500 – 20 – 8 – 12 = 1460 B (bajtov).

4. CRC16

CRC (Cyclic Redundancy Check) je technika na detekciu chýb v prenosoch dát. CRC funguje tak, že sa na začiatku určípolynómny deliteľ v binárnom tvare. V dalšom kroku sa k dátam pripíše 16 núl ktoré slúžia na kontrolný súčet. Neskôr sa pôvodné dáta vydelia pomocou bitovej funkcie XOR. Po vydelení je tento zvyšok kotrolným súčtom ktorý sa pridá k dátam a odošle spolu s nimi kde presne rovnakú akciu vykoná užívateľ číslo 2. Ak sa hodnoty budú zhodovať znamená to, že dáta sú v poriadku, no ak sa líšia znamená to že časť dát je poškodená a musí sa poslať znova

CRC 16 vs. CRC 32: Pre tento projekt som si vybral CRC 16 pre efektívnosť a dostatočnú veľkosť. CRC 16, tým že obsahuje 16 bitov, je časovo efektívnejší ako 32-bitový CRC 32. Pre posielanie správ a malých súborov je dostatočný.

5. ARQ

ARQ (Automatic Repeat Request) je metóda na zabezpečenie bezchybného prenosu dát. Táto metóda sa používa na detekciu chýb počas prenosu dát.

Stop and Wait: V projekte je zakomponovaná metóda Stop and Wait, ktorá, ako názov napovedá, funguje na princípe, kedy po každom poslanom pakete odosielateľ čaká na ACK alebo NACK, aby mohol poslať ďalší paket. Ak dostane ACK (acknowledgment), znamená to, že paket bol doručený, a môže poslať ďalší. Ak však dostane správu NACK (Negative acknowledgment), znamená to, že paket nebol odoslaný správne, a tým oznamuje odosielateľovi, aby poslal paket znova.

Výhody Stop and Wait: jednoduchší na implementáciu, spoľahlivý.
Nevýhody Stop and Wait: neefektívny, problémy pri vysokej latencii.

6. Keep-Alive

Je technika na overenie, či spojenie medzi používateľmi stále prebieha. Cieľom je zabrániť opakovanému nadväzovaniu spojenia. V projekte funguje Keep-Alive tak, že pri neaktivite sa každých 5 sekúnd posiela správa, aby sa overilo spojenie.

Ak jedna strana nedostane odpoveď 3-krát po sebe, spojenie sa považuje za neaktívne, teda ukončené. Keď sa spojenie ukončí, automaticky sa vypíše správa do terminálu o tejto akcii.

Keep-Alive by mal byť schopný reagovať na chyby, ako napríklad pri strate spojenia, kedy sa správy alebo súbory stále posielajú, ale spojenie sa preruší. V takom prípade by mal automaticky skontrolovať integritu spojenia. Ak nedostane odpoveď, spojenie sa považuje za prerušené.

V prípade, že sa spojenie obnoví, pokúsi sa pokračovať v zasielaní správ alebo súborov ďalej.

7. Fragmentácia

Fragmentácia je proces, ktorým sa dátové pakety prenášajú cez sieť. Ak má paket väčšiu veľkosť, než je povolené, je rozdelený na menšie časti nazývané fragmenty. Maximálna veľkosť fragmentu je obvykle 1500 bajtov (čo je hodnota MTU – **Maximum Transmission Unit**). Ak veľkosť fragmentu presiahne tento limit, je potrebná ďalšia fragmentácia, čo však nie je efektívne.

Každý fragment obsahuje hlavičku s dôležitými údajmi, ako je poradie fragmentu a kontrolné informácie, spolu s dátami, ktoré sa majú odoslať. Tieto fragmenty sa odosielajú nezávisle prijímateľovi, ktorý má za úlohu všetky prijaté fragmenty zrekonštruovať späť do pôvodného celku.

Pri posielaní cez protokol UDP je dôležité zabezpečiť mechanizmus kontroly, či boli všetky fragmenty úspešne prijaté. Keďže protokol UDP je nespoľahlivý, fragmenty sa môžu počas prenosu stratiť. Preto je potrebné implementovať doplnkové metódy, ako sú potvrdenia prijatia (ACK) alebo kontrolné súčty (CRC), na zaistenie správnosti a úplnosti prenesených dát.

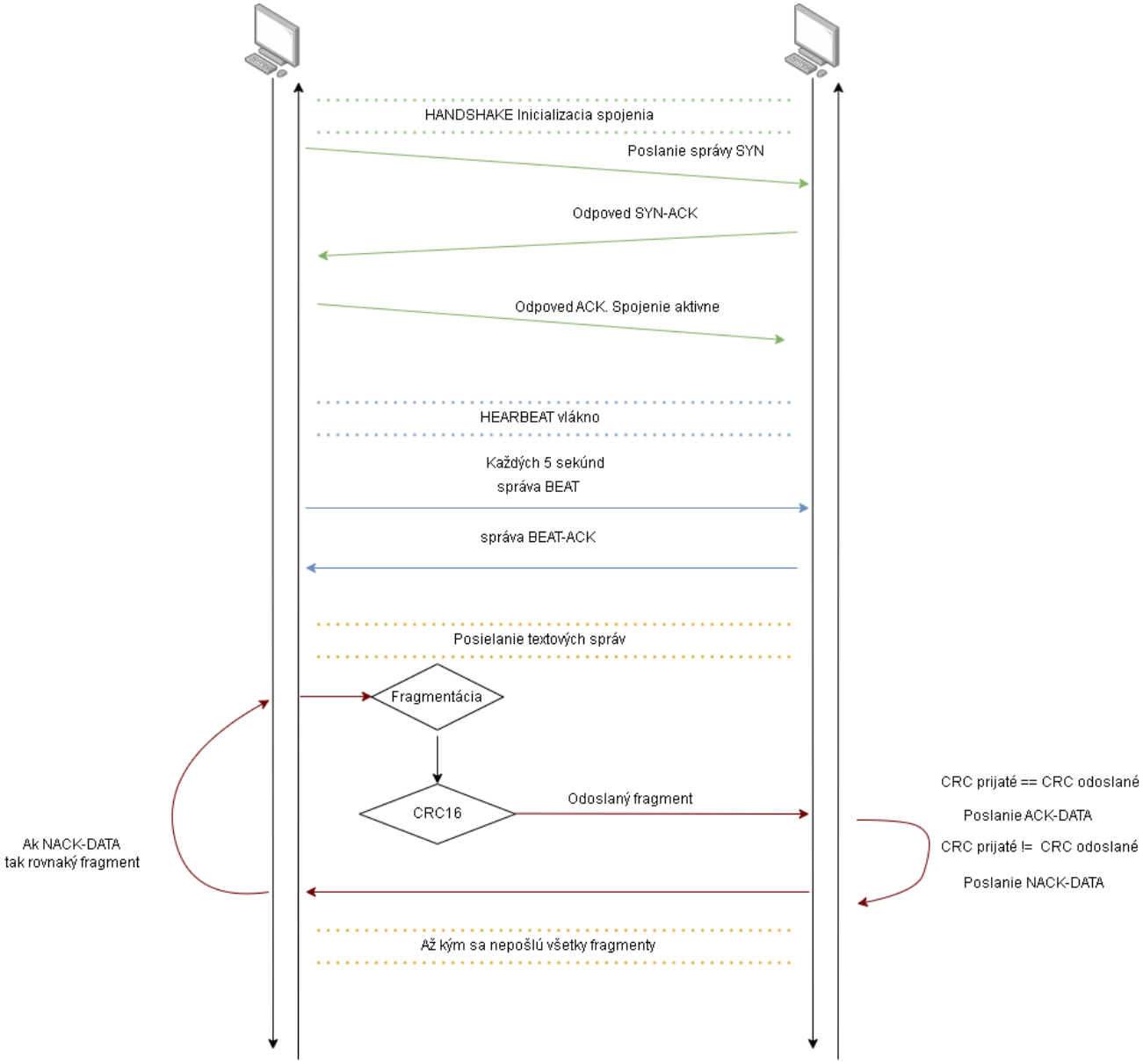
8. Odosielanie suborov/správ

V programe vytvorenom na prenos fragmentov je možné posielať textové správy aj súbory. Pri odosielaní fragmentov si používateľ sám nastaví veľkosť fragmentu na odoslanie (posledný fragment môže byť menší). Po odoslaní fragmentu program čaká na odpoveď od druhého počítača, ktorá potvrdzuje prijatie fragmentu. Ak takúto správu nedostane, pokúsi sa fragment odoslať

znova, a to opakovane, kým nepríde potvrdenie o prijatí. Tento cyklus sa opakuje pre všetky fragmenty, až kým nie sú odoslané všetky časti správy alebo súboru.

Ak nastane situácia, že sa spojenie stratí, prenos sa preruší a program začne testovať integritu spojenia. V takom prípade sa pokúsi znovu nadviazať spojenie. Ak sa mu to podarí, odosielanie fragmentov pokračuje tam, kde bolo prerušené.

9. Diagram



Uvedený diagram znázorňuje funkciu programu navrhnutého na odosielanie a prijímanie fragmentov medzi dvoma zariadeniami súčasne pomocou protokolu UDP.

Prvým krokom je nastavenie parametrov, ako sú port zariadenia 1, IP adresa a port zariadenia 2. Po správnom nastavení týchto parametrov na oboch zariadeniach prebehne proces 3-way Handshake, ktorý overí spojenie medzi zariadeniami. Tento proces začína odoslaním správy SYN z jedného zariadenia, na ktorú druhé zariadenie odpovie správou SYN-ACK. Po prijatí správy SYN-ACK sa odošle správa ACK, čím sa spojenie považuje za nadviazané.

Ďalšou vlastnosťou programu je udržiavanie spojenia počas nečinnosti. Ak program nevykonáva žiadnu akciu, medzi zariadeniami sa každých 5 sekúnd odosiela správa BEAT, na ktorú sa očakáva odpoveď. Ak zariadenie nedostane odpoveď na tri po sebe odoslané správy, spojenie sa považuje za ukončené.

Poslednou časťou diagramu je odosielanie fragmentov, ktoré môžu predstavovať textové správy alebo súbory zvolené používateľom. Pri prenose dát sa dáta najskôr rozdelia na menšie časti nazývané fragmenty. Každému fragmentu sa vypočíta kontrolný súčet CRC16, ktorý sa spolu s hlavičkou odosiela s fragmentom. Prijímacie zariadenie po prijatí fragmentu opäť vypočíta CRC16. Ak sa vypočítaná hodnota zhoduje s prijatou hodnotou, zariadenie odošle potvrdenie ACK-DATA. Ak sa hodnoty nezhodujú, zariadenie odošle správu NACK-DATA, ktorá signalizuje chybný fragment a požaduje jeho opätovné odoslanie. Tento cyklus sa opakuje, až kým sa neprijmú všetky fragmenty správy alebo súboru bez chýb.

Týmto spôsobom program zabezpečuje spoľahlivé odosielanie a prijímanie dát aj nad nespoľahlivým protokolom UDP.

10. Záver

V tejto dokumentácii som pre návrh protokolu P2P nad UDP vysvetlil metódy použité v implementácii a pridal diagram, ktorý stručne znázorňuje fungovanie programu, vrátane procesu odosielania a prijímania fragmentov. Diagram demonštruje kľúčové mechanizmy, ako sú fragmentácia dát a overovanie integrity pomocou CRC. Tieto prvky zabezpečujú spoľahlivý prenos dát aj nad nespoľahlivým protokolom UDP.