Slovenská technická univerzita v Bratislave

Fakulta informatiky a informačných technológií

Počítačové a komunikačné siete

Semestrálne zadanie

Rok: 2024-2025

Obsah

Dokument reprezentuje: Návrh komunikačného protokolu pre spoľahlivý prenos dát a plán implementácie jednotlivých mechanizmov

Vypracoval: Yulian Kisil

Úvod

Cieľom tejto dokumentácie je predstaviť návrh komunikačného protokolu pre spoľahlivý prenos dát medzi dvoma stranami. Tento protokol umožňuje nadviazanie spojenia, prenos správ po fragmentoch, kontrolu integrity správ, opätovné poslanie stratenej alebo poškodenej správy a udržiavanie aktívneho spojenia. Protokol je navrhnutý tak, aby podporoval simuláciu chýb v správach, čo umožňuje testovanie jeho robustnosti. Na vývoj projektu budem používať jazyk python

Štruktúra hlavičky protokolu(updated)

Pre účely prenosu správ medzi uzlami je potrebné definovať štruktúru hlavičky. Každý paket posielaný medzi uzlami bude obsahovať hlavičku so základnými informáciami:

NEW:

<table>
<tr><td colspan="3">Číslo fragmentu</td></tr>
<tr><td>Typ správy</td><td>Konečný rámček</td><td>Kontrola integrity (CRC)</td></tr>
</table>

OLD:

<table>
<tr><td>Typ správy</td><td>Veľkosť dát</td><td>Identifikátor</td></tr>
<tr><td colspan="3">Číslo fragmentu</td></tr>
<tr><td colspan="3">Celkový počet fragmentov</td></tr>
<tr><td colspan="2">Kontrola integrity (CRC)</td><td>Sekvenčné číslo</td></tr>
</table>

Vysvetlenie polí:

- **Číslo fragmentu(4 bajt)**: Poradie fragmentu v rámci jednej správy. Umožňuje rekonštrukciu správy na strane príjemcu
- **Typ správy(1 bajt)**: Udáva, či ide o dátovú správu, potvrdenie, kontrolnú správu pre Keep-Alive, alebo chybnú správu
- **Konečný rámček(1 bajt):** Príznak, ktorý označuje, že ide o koncový fragment. Môže byť 1(true) alebo 0 (false) **(updated)**
- **Kontrola integrity (CRC) (2bajt):**Používa sa na zistenie, či došlo k poškodeniu správy počas prenosu.

Čo bolo odstránené:

1.Identifikátor(2 bajt): Indentifikátor - je unikátna hodnota pre každý balík. Ak je balík fragmentovaný, všetky fragmenty rovnakého paketu budú mať rovnaký identifikátor

2.Celkový počet fragmentov(4 bajt): Označuje, koľko fragmentov má kompletná správa

3.Veľkosť dát(1 bajt): Umožňuje dynamickú veľkosť fragmentov podľa preferencií používateľa

4.Sekvenčné číslo (2 bajt): je jedinečná hodnota pre každý fragment. Ak je paket fragmentovaný, každý fragment bude mať jedinečné poradové číslo

Aké príznaky sa používajú v „Typ správy“:

1. MESSAGE_TYPE_TEXT (0) - Toto je typ správy označujúci textovú správu.
2. MESSAGE_TYPE_FILE (1) - Toto je typ správy, ktorý označuje prenos súboru
3. MESSAGE_TYPE_HANDSHAKE (2)- Toto je typ správy, ktorý označuje počiatočné nadviazanie spojenia medzi klientom a klientom (handshake).
4. MESSAGE_TYPE_NAME_FILE (3) - Toto je typ správy, ktorý označuje prenos názvu súboru.
5. MESSAGE_TYPE_ACK (4) - Toto je typ správy, ktorá sa používa na potvrdenie prijatia fragmentu.
6. MESSAGE_TYPE_HEARTBEAT (5) - Toto je typ správy používanej na kontrolu stavu spojenia
7. MESSAGE_TYPE_HEARTBEAT_REPLY (6) - Toto je typ správy, ktorá je odpoveďou na žiadosť o "heartbeat".
8. MESSAGE_TYPE_NACK (7) - Toto je typ správy, ktorá označuje negatívne potvrdenie (NACK) alebo chybu.

Metóda kontroly integrity prenesenej správy

Na kontrolu integrity údajov použijem CRC16 (Cyclic Redundancy Check). CRC16 je bežná a účinná metóda detekcie chýb. Výpočet CRC16 je založený na polynomickom násobení nad binárnymi údajmi. Tento proces zabezpečuje, že ak príjemca správy dostane iný CRC16 ako odosielateľ, znamená to, že údaje boli počas prenosu poškodené.Pre CRC16 budu som používať polynóm : x^16+x^15+x^2+1

Výpočet CRC16:

1. Dátové pole správy sa považuje za veľký binárny reťazec.
2. Tento reťazec sa vydelí generujúcim polynómom, čo vytvorí zvyšok.
3. Zvyšok je pripojený k správe a odoslaný spolu s ňou.
4. Príjemca správy opäť vydelí prijatý dátový reťazec tým istým polynómom.
5. Ak zvyšok je 0, správa je nepoškodená. Inak došlo k chybe.

Príklad:

Vstupné údaje: b'1234' má bajty: '1' → 0x31 '2' → 0x32 '3' → 0x33 '4' → 0x34

Počiatočná hodnota CRC je 0xFFFF

Pre každý bajt (počnúc 0x31 pre „1“) sa vykoná XOR s aktuálnou hodnotou CRC, po ktorom nasleduje 8-bitový posun, pričom v každom kroku sa kontroluje najmenší bit.

Po výpočte pre všetky bajty je výsledok pre reťazec b'1234' 0xF1B0. (==61872)

Spoľahlivý prenos dát (ARQ)

Pre zabezpečenie spoľahlivého prenosu dát sa použije mechanizmus ARQ (Automatic Repeat reQuest). Protokol bude implementovať systém selektívneho opakovania, čo znamená, že ak príjemca deteguje chýbajúci alebo poškodený fragment, požiada o opätovné zaslanie len toho konkrétneho fragmentu, nie celej správy.

Proces ARQ:

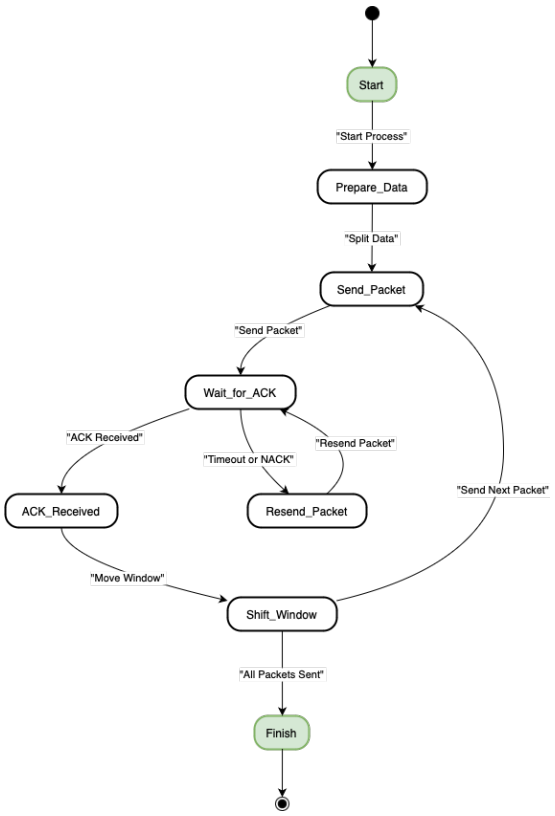
1. Príjemca po prijatí každého fragmentu odošle spätné potvrdenie (ACK) s informáciou, ktorý fragment úspešne prijal.
2. Ak sa na strane odosielateľa neobdrží ACK v definovanom časovom limite, fragment sa odošle znova.
3. V prípade, že príjemca obdrží poškodený fragment (CRC kontrola zlyhá), odošle negatívne potvrdenie (NACK) a požiada o opätovné zaslanie daného fragmentu.
4. Ak čas skončí a ACK nepríde, fragment sa odošle znova.

Na prenos textu/súborov - na kontrolu, či sú údaje poškodené alebo stratené, použijem algoritmus: Selective Repeat (SR)

Zakladne princípy práce Selective Repeat:

1. **Rozdelenie na balíky a číslovanie balíkov:** Dáta sú rozdelené do malých paketov a každý balík je odoslaný s jedinečnou sekvenciou číslom.

2. **Prenos paketov**: Odosielateľ môže odoslať viacero paketov (veľkosť je nastavená v veľkosť dát)
3. **Potvrdenie prijatých paketov (ACK):** Keď príjemca úspešne prijme odošle späť odosielateľovi potvrdenie (ACK), v ktorom uvedie poradie prijatých paketov. prijatého paketu. Ak je niektorý z paketov počas prenosu poškodený, prijímač odošle NACK a odosielateľ tento fragment odošle znova . Ak bol niektorý z paketov pri stratený, prijímač si vyžiada opakované odoslanie len tohto konkrétneho paketu
4. **Vyrovnávacia pamäť**: Prijímam pakety mimo poradia (napr. ak niektoré pakety prídu rýchlejšie ako iné). Na tento účel má k dispozícii vyrovnávaciu pamäť, do ktorej ukladá pakety, kým sa nevyskytne chýbajúci paket (s nižším číslom) nie je prijatý. Potom môže údaje znovu zoradiť do správneho poradí.



~ Prepare Data : Súbor je rozdelený na fragmenty (pakety) s jedinečnými sekvenčnými číslami. Pre každý paket sa vypočíta CRC16 (kontrolný súčet) na overenie integrity

~ Send Packet: Pakety sa prenášajú podľa ich sekvenčných čísel. Údaje sa odosielajú prostredníctvom UDP zásuvky.

~Wait for ACK: Čaká na ACK (potvrdenie) pre každý paket. Ak sa prijme ACK, označí príslušný paket ako doručený. Ak sa ACK neprijme do určeného časového limitu, pokračuje v opakovanom odosielaní.

~ACK Received: Aktualizuje stav paketu (doručený). Posunie hranice posuvného okna, aby sa mohol odoslať ďalší paket.

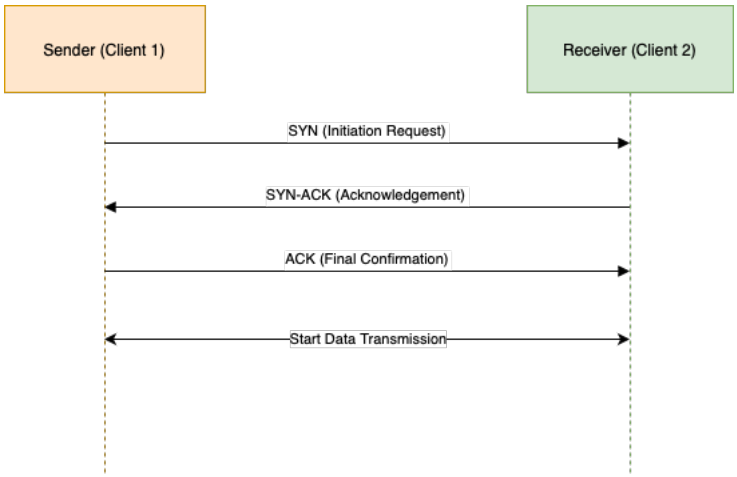
~Resend Packet: Posielajú sa len stratené alebo poškodené balíky.

~Shift Window: Určuje ďalšie pakety, ktoré sa majú odoslať. Kontroluje, či boli doručené všetky pakety.

Description of the Handshake(updated)

Mechanizmus handshake na vytvorenie spojenia peer-to-peer (P2P), podobný trojcestnému handshake TCP, ale prispôsobený kontextu P2P. Tu je krátky opis tohto procesu:

1. **Iniciovanie žiadosti (SYN):** Odosielateľ pošle príjemcovi žiadosť o komunikáciu - správu SYN
2. **Potvrdenie pripravenosti (SYN-ACK):** Odosielateľ pošle príjemcovi žiadosť o komunikáciu - správu SYN.
3. **Konečné potvrdenie (ACK):** Odosielateľ dostane od príjemcu správu SYN-ACK a odošle konečné potvrdenie - správu ACK.



Algoritmus Handshake v mojom projekte vytvára spojenie medzi odosielateľom a príjemcom pred začatím prenosu údajov. Ide o kritickú fázu, ktorá zabezpečuje, aby boli obe strany synchronizované a pripravené na prenos.

Snímka obrazovky handshake s Wireshark:

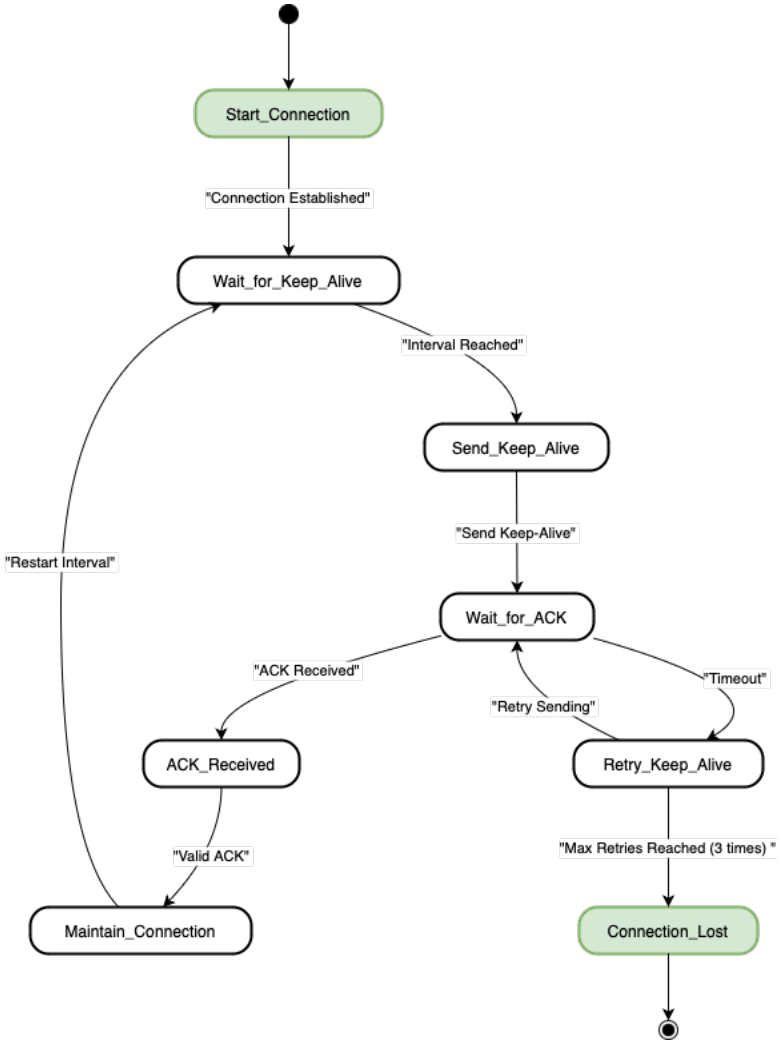
```
     6 16.830850    127.0.0.1       127.0.0.1       Kisil     40 Message Type: Handshake
     7 16.831484    127.0.0.1       127.0.0.1       Kisil     40 Message Type: Handshake
     8 16.831551    127.0.0.1       127.0.0.1       Kisil     40 Message Type: Handshake
```

Udržiavanie spojenia (Keep-Alive)

Mechanizmus keep-alive sa používa na udržiavanie aktívneho spojenia medzi dvoma uzlami. Každý uzol bude pravidelne posielať kontrolné správy, ak medzi nimi nie je žiadna aktivita, aby potvrdil, že uzol na druhej strane je stále k dispozícii. Perióda je 5 sekúnd. na odpojenie je potrebné odoslať 3 nepotvrdené prípady

Proces Keep-Alive:

1. Po nadviazaní spojenia odosielateľ aj príjemca nastaví časovač.
2. Ak v stanovenom čase neobdrží Keep-Alive správu, uzol predpokladá, že spojenie bolo stratené a pokúsi sa o obnovenie.
3. Uzol odošle ďalšiu žiadosť Ak na tri žiadosti nedostane žiadnu odpoveď, uzol predpokladá, že spojenie bolo stratené, a odpojí sa od druhého uzla.



Snímka obrazovky programu Wireshark:

427	83.461006	127.0.0.1	127.0.0.1	Kisil	40	Message Type: Heartbeat
428	83.461625	127.0.0.1	127.0.0.1	Kisil	40	Message Type: Heartbeat Reply
429	88.466456	127.0.0.1	127.0.0.1	Kisil	40	Message Type: Heartbeat
430	88.467157	127.0.0.1	127.0.0.1	Kisil	40	Message Type: Heartbeat Reply
431	93.471662	127.0.0.1	127.0.0.1	Kisil	40	Message Type: Heartbeat
432	93.471848	127.0.0.1	127.0.0.1	Kisil	40	Message Type: Heartbeat Reply
433	98.476222	127.0.0.1	127.0.0.1	Kisil	40	Message Type: Heartbeat

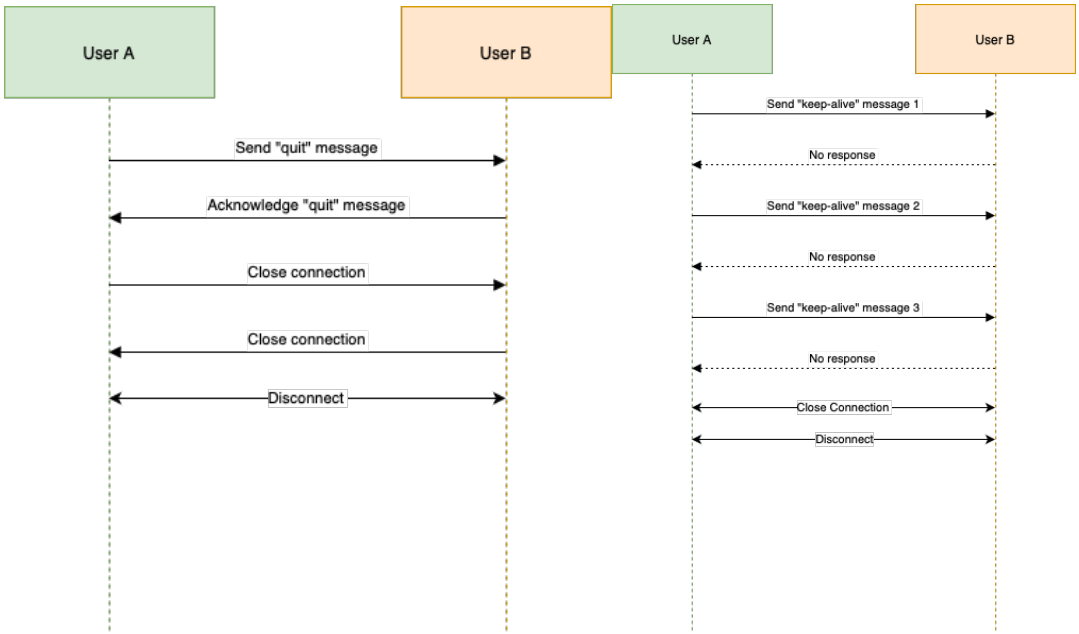
Mechanizmus na simuláciu chýb(updated)

Na účely testovania sa do protokolu pridá funkcia na simuláciu chýb v správach. Táto funkcia náhodne vyberie pozíciu v dátovom pakete a nahradí bajt náhodnou hodnotou, čím simuluje chybu v prenášaných údajoch. Potom metóda vráti upravený paket s poškodeným bajtom a zobrazí správu o mieste, kde sa chyba vyskytla. Táto funkcia umožní skontrolovať spoľahlivosť implementovaných mechanizmov detekcie a opravy chýb.

Projekt obsahuje aj mechanizmus na simuláciu neočakávaného odpojenia používateľa na 10 s. Ak používateľ zadá číslo 2, načítavanie súborov sa pozastaví na 10 s.

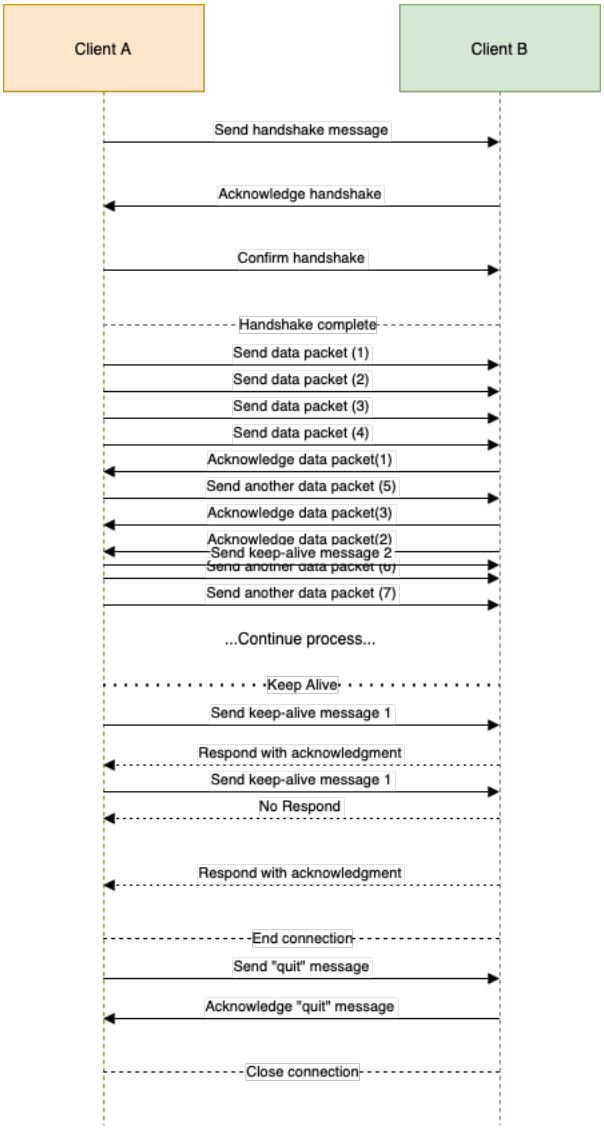
Koncové pripojenie(updated)

Diagramy znázorňujú dva procesy ukončenia spojenia v sieťových protokoloch. Prvý predstavuje milosrdné odpojenie, pri ktorom sa obe strany vzájomne dohodnú na ukončení spojenia. Tento sa zvyčajne vyskytuje počas riadených vypnutí. Druhý mechanizmus sa používa na automatické uzavretie spojenia, keď jedna zo strán nedostane odpoveď na 3 správ keep-alive. Mechanizmus keep-alive sa zvyčajne používa na overenie, či je spojenie stále aktívne.

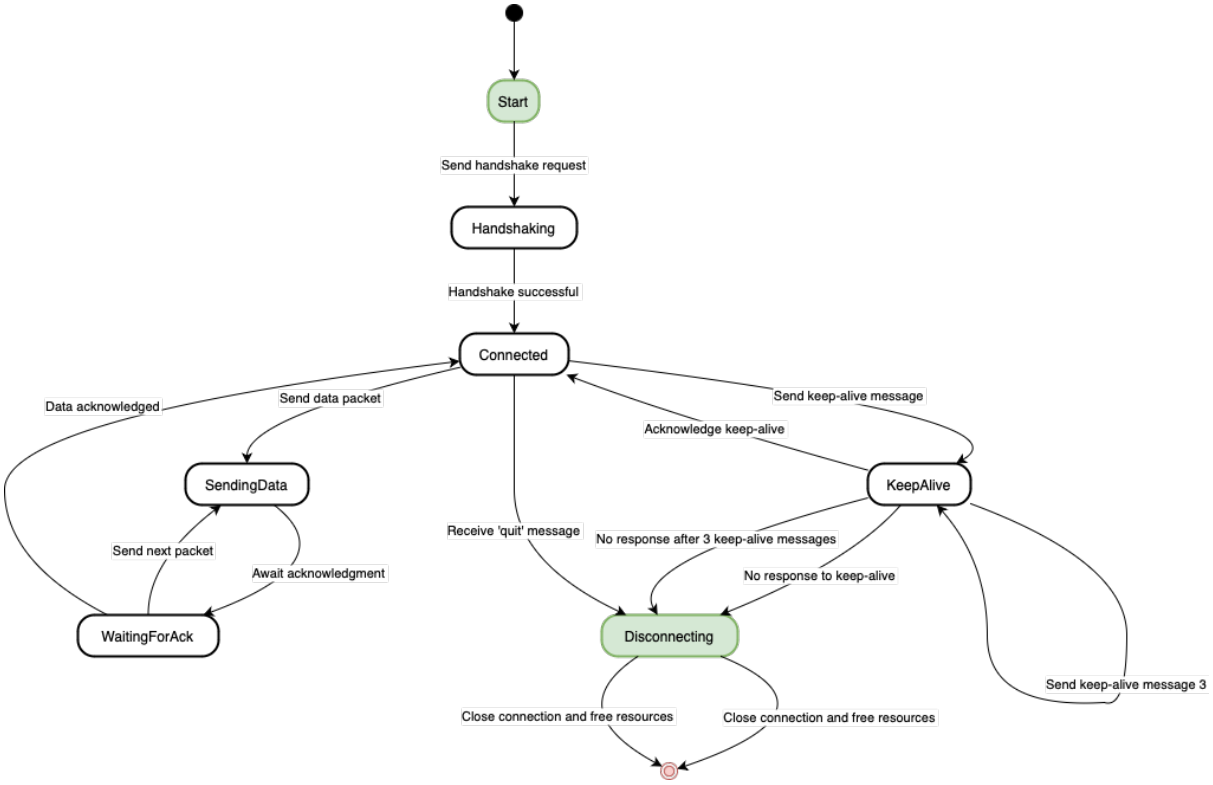


Diagramy:

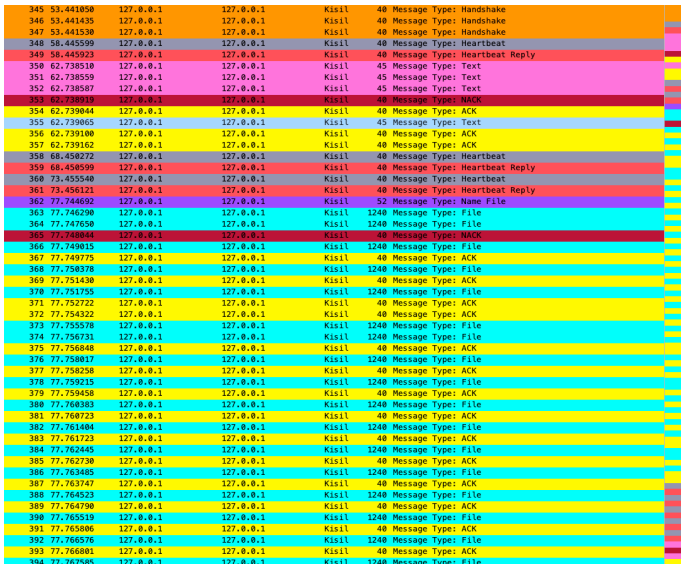
Sekvenčný diagram:

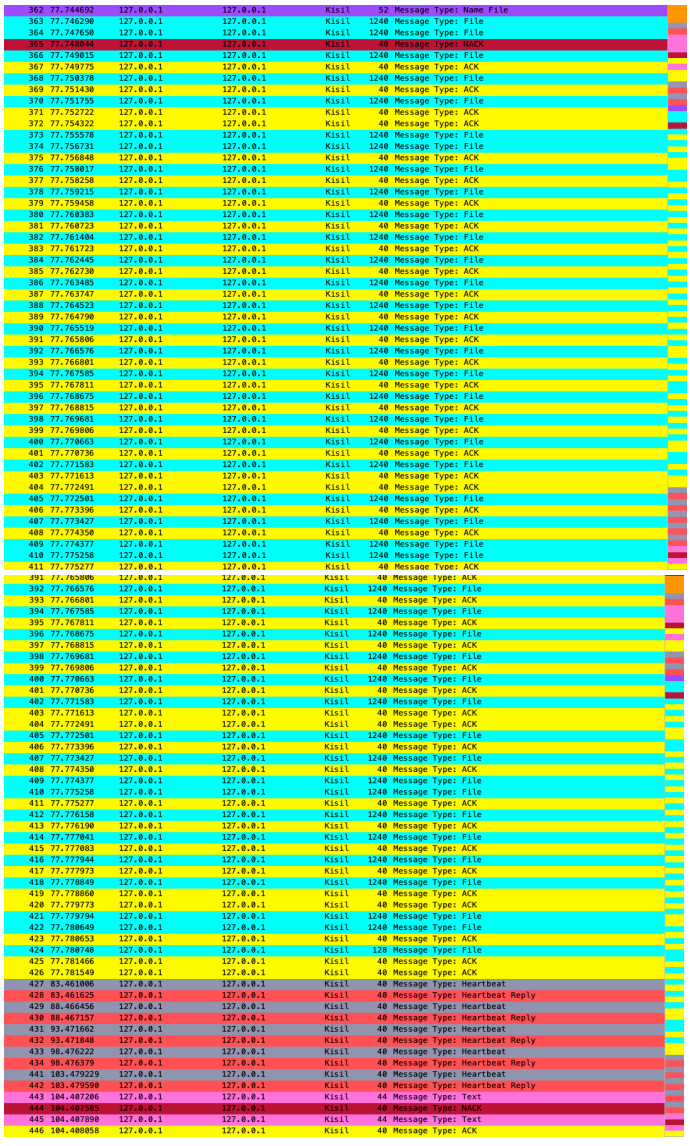


Stavový diagram:



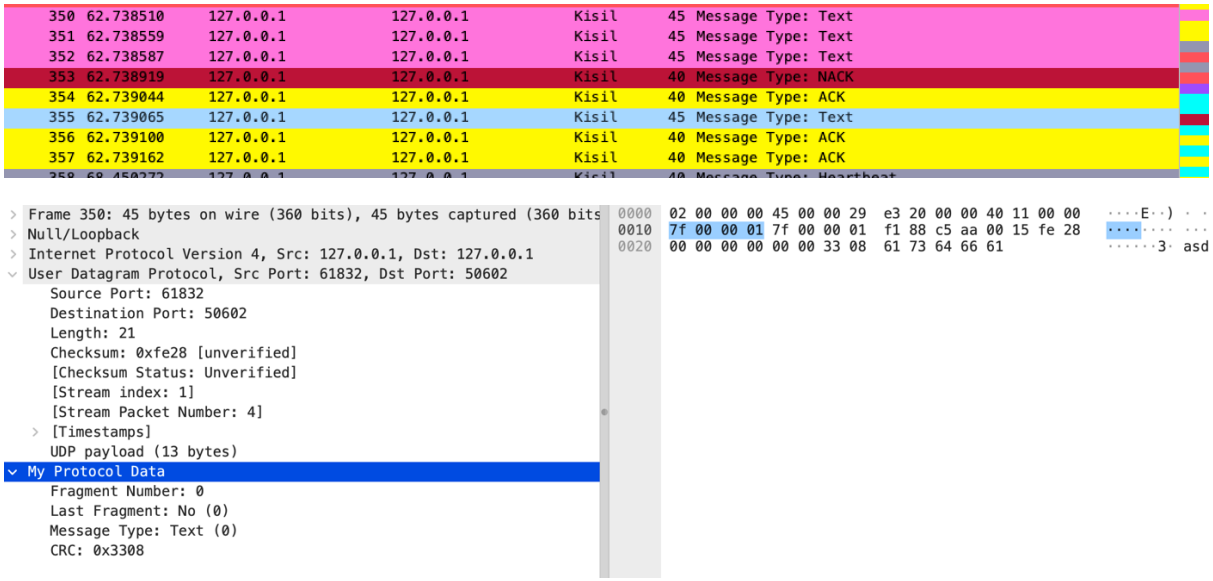
Skript Wireshark Lua



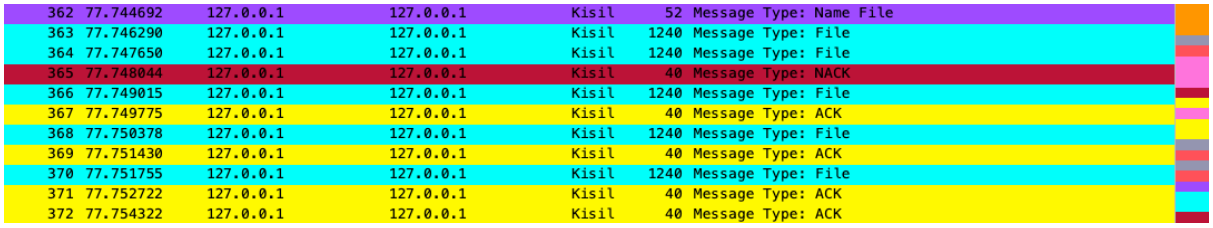


Tieto prezentácie ukazujú, ako sa dva uzly navzájom pripájajú prostredníctvom Handshake.Potom majú keep alive. Potom sa odošle text rozdelený na 3 fragmenty. Potom bol odoslaný text. rozdelený na 31 fragmentov. V prvom fragmente sa vyskytla chyba. Potom sa medzi uzlami vykonalo KeepAlive. a nakoniec sa spojenie ukončilo správou „quit“.

Odoslať reťazec:



Zobrazuje, ako bola správa odoslaná. V prvom fragmente sa vyskytla chyba. Potom bol fragment opäť odoslaný



Zobrazí sa začiatok odosielania súboru. Najprv sa odoslal názov súboru.Potom sa začal odosielať súbor.V prvom fragmente sa vyskytla chyba. Tento fragment bol odoslaný znova

Záver

Táto štruktúra protokolu zabezpečuje spoľahlivý prenos údajov medzi dvoma uzlami vrátane mechanizmov na kontrolu integrity, opätovné odosielanie chybových fragmentov a udržiavanie aktívneho spojenia. Protokol podporuje aj simuláciu chýb,

ktorá umožňuje testovať jeho odolnosť voči rôznym typom zlyhaní. Schémy opisujú rôzne fázy komunikácie a správanie uzlov v rôznych stavoch. Po prvej prezentácii projektu boli odstránené polia „Identifikátor“, „Veľkosť dát“, „Sekvenčné číslo“, pretože sa považovali za nepotrebné; logika protokolu umožňuje vynechanie týchto polí na úsporu miesta. Pole „Celkový počet fragmentov“ bolo nahradené poľom „Konečný rámček“, aby sa ušetrila pamäť a zabezpečilo efektívnejšie spracovanie správ. Všetky požiadavky boli splnené a aplikácia počas testovania fungovala správne. Testovanie sa uskutočnilo v lokalnej sieti aj v sieti eduroam.

Použité zdroje

1. https://youtu.be/WfIhQ3o2xow?si=DjkPQkLjJANeT8Ys
2. https://www.geeksforgeeks.org/sliding-window-protocol-set-3-selective-repeat/
3. https://www.baeldung.com/cs/selective-repeat-protocol
4. https://www.imperva.com/learn/performance/http-keep-alive/
5. https://www.geeksforgeeks.org/tcp-3-way-handshake-process/
6. https://www.linkedin.com/pulse/understanding-tcp-3-way-handshake-computer-networking-haque
7. https://www.scaler.com/topics/fragmentation-in-networking/#