

Slovenská technická univerzita v Bratislave
Fakulta informatiky a informačných technológií

Počítačové a komunikačné siete
Semestrálne zadanie

Rok: 2024-2025

Obsah

1. Uvod.	3
2. Štruktúra hlavičky protokolu.....	3
3. Metóda kontroly integrity prenesenej správy.....	4
4. Spoľahlivý prenos dát (ARQ)	5
5. Description of the Handshake.....	7
6. Udržiavanie spojenia (Keep-Alive)	8
7. Mechanizmus na simuláciu chýb.....	9
8. Koncové pripojenie.....	9
9. Diagramy.....	10
10. Skript Wireshark Lua.....	12
11. Záver.....	13
12. Použité zdroje.....	14

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 posielaý medzi uzlami bude obsahovať hlavičku so základnými informáciami:

NEW:

Číslo fragmentu		
Typ správy	Konečný rámček	Kontrola integrity (CRC)

OLD:

Typ správy	Veľkosť dát	Identifikátor
Číslo fragmentu		
Celkový počet fragmentov		
Kontrola integrity (CRC)		Sekvenčné číslo

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): Identifikátor - 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 polynomicom 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 budem 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. Prijemca 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:

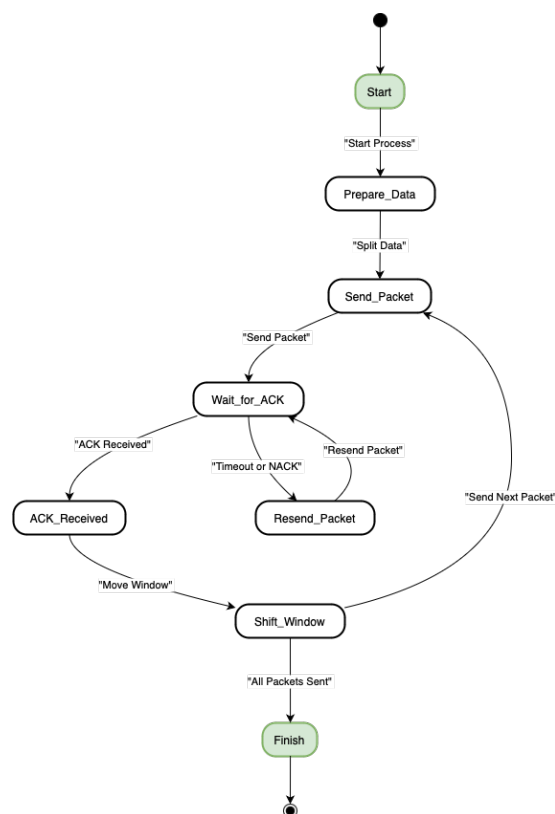
1. Prijemca po prijatí každého fragmentu odošle spätné potvrdenie (ACK) s informáciou, ktorý fragment úspešne prijal.
2. Ak sa na strane odosielateľa neobdrží ACK v definovanom časovom limite, fragment sa odošle znova.
3. V prípade, že príjemca obdrží poškodený fragment (CRC kontrola zlyhá), odošle negatívne 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. **Vyrovňavacia 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 poradi.



~ **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.

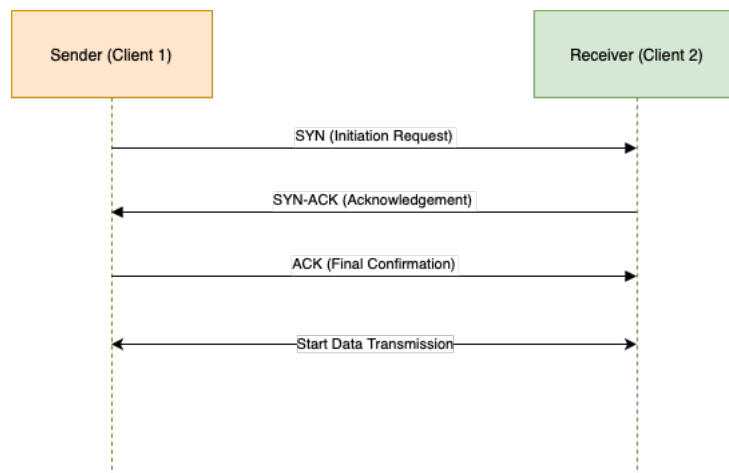
~**Resend Packet:** Posielajú sa len stratené alebo poškodené balíky.

~**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 prispôbený 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:

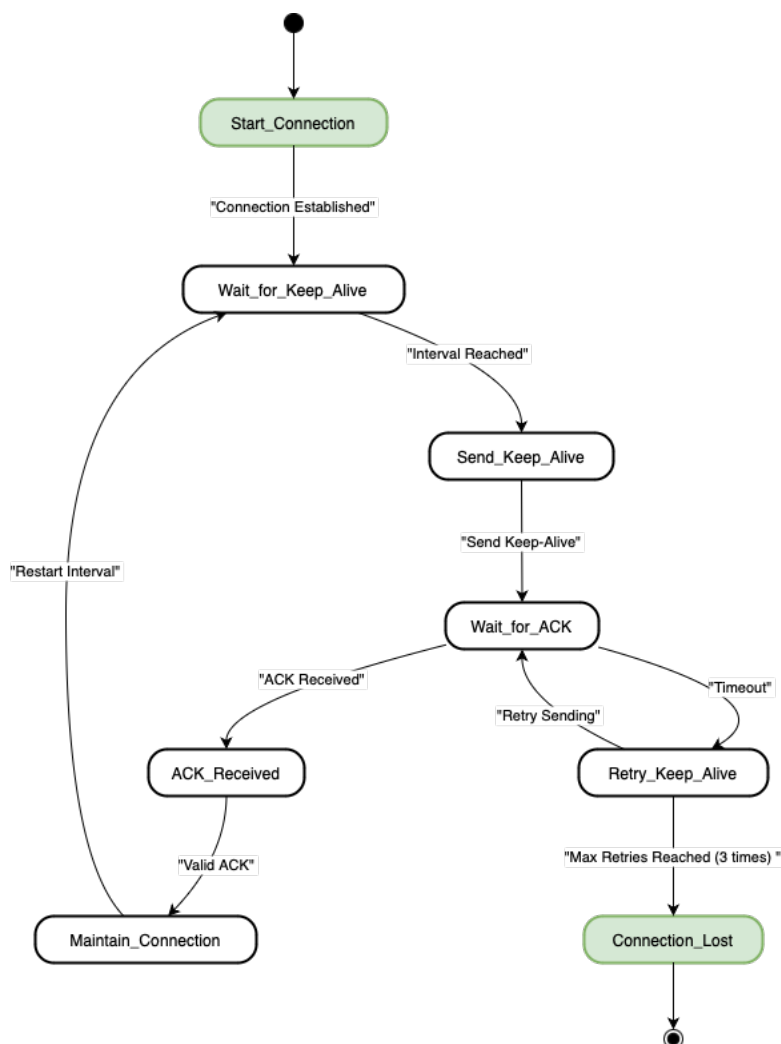
6	16.830850	127.0.0.1	127.0.0.1	Kisil	40	Message Type: Handshake
7	16.831484	127.0.0.1	127.0.0.1	Kisil	40	Message Type: Handshake
8	16.831551	127.0.0.1	127.0.0.1	Kisil	40	Message Type: Handshake

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

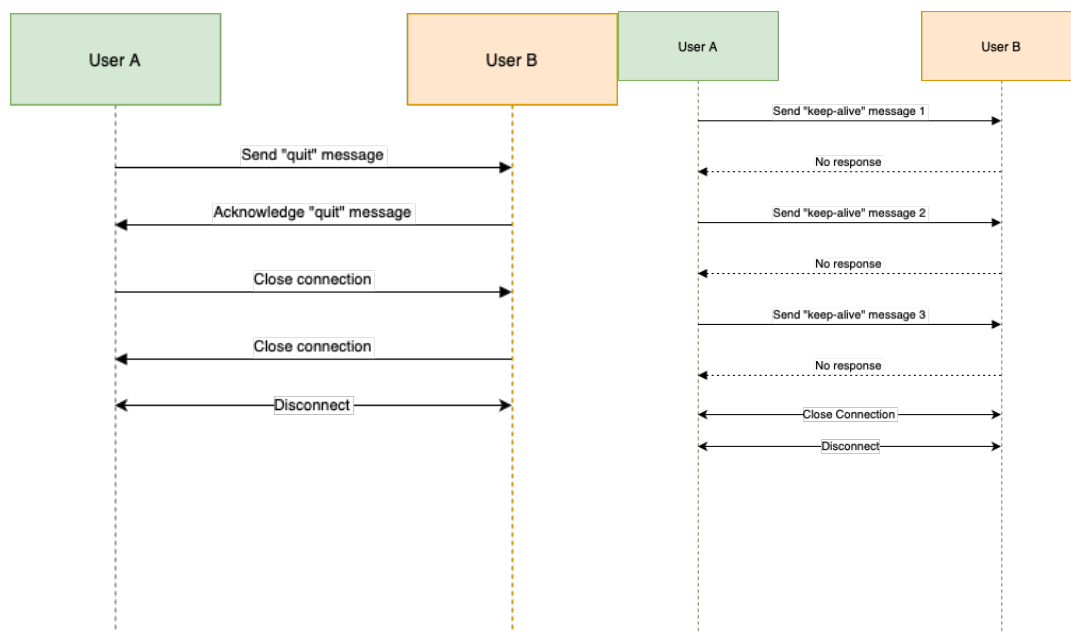
Mechанизм на симуляцию ошибок(updated)

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.

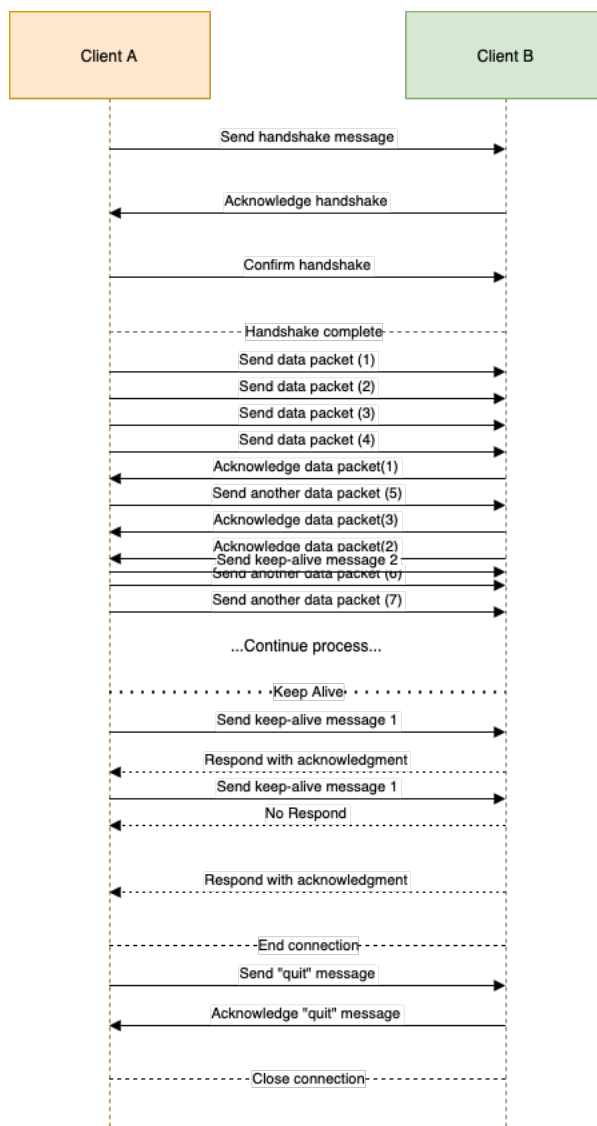
Koncové pripojenie(updated)

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 správy keep-alive. Mechanizmus keep-alive sa zvyčajne používa na overenie, či je spojenie stále aktívne.

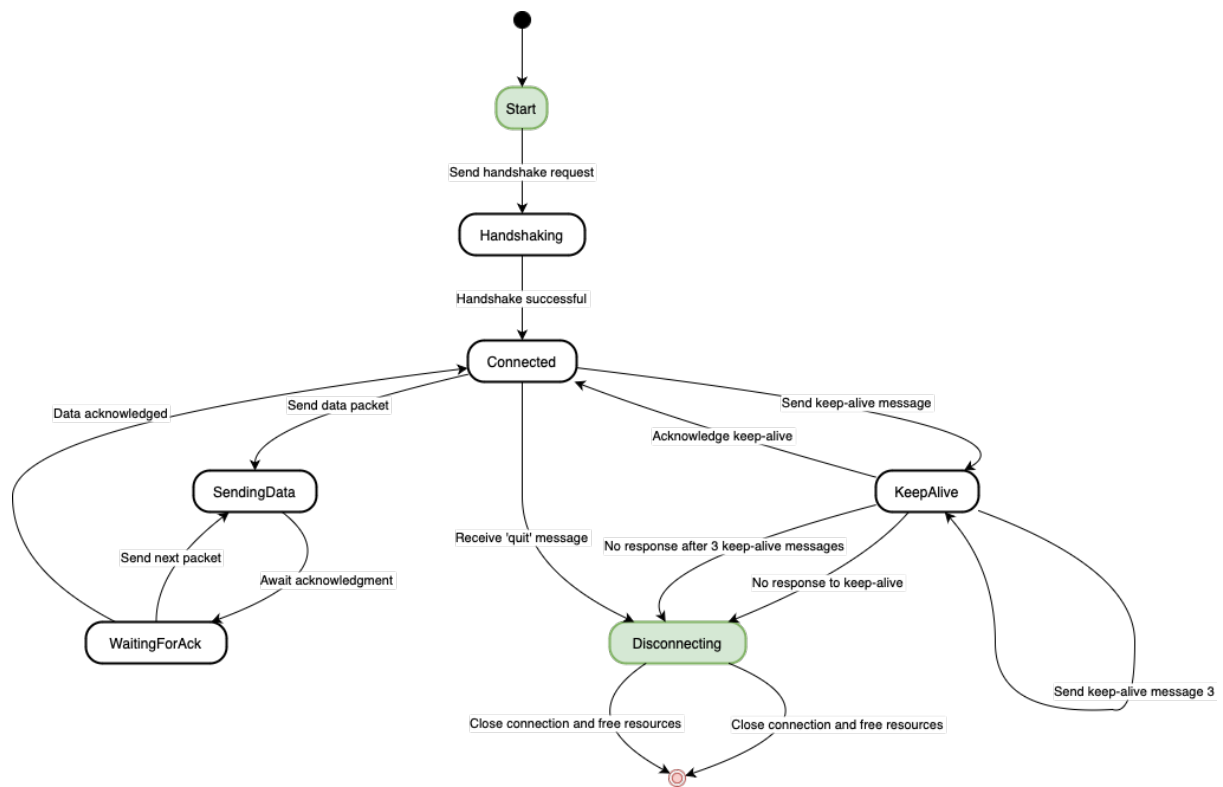


Diagramy:

Sekvenčný diagram:



Stavový diagram:



Skript Wireshark Lua

345	53.441858	127.0.0.1	127.0.0.1	Kisill	48	Message Type: Handshake
346	53.441859	127.0.0.1	127.0.0.1	Kisill	48	Message Type: Handshake
347	53.441538	127.0.0.1	127.0.0.1	Kisill	48	Message Type: Handshake
348	58.445599	127.0.0.1	127.0.0.1	Kisill	40	Message Type: Heartbeat
349	58.445923	127.0.0.1	127.0.0.1	Kisill	48	Message Type: Heartbeat Reply
350	62.738519	127.0.0.1	127.0.0.1	Kisill	45	Message Type: Text
351	62.738559	127.0.0.1	127.0.0.1	Kisill	45	Message Type: Text
352	62.738587	127.0.0.1	127.0.0.1	Kisill	45	Message Type: Text
353	62.738619	127.0.0.1	127.0.0.1	Kisill	45	Message Type: Text
354	62.739044	127.0.0.1	127.0.0.1	Kisill	48	Message Type: ACK
355	62.739065	127.0.0.1	127.0.0.1	Kisill	45	Message Type: Text
356	62.739180	127.0.0.1	127.0.0.1	Kisill	48	Message Type: ACK
357	62.739182	127.0.0.1	127.0.0.1	Kisill	48	Message Type: ACK
358	68.450272	127.0.0.1	127.0.0.1	Kisill	40	Message Type: Heartbeat
359	68.450599	127.0.0.1	127.0.0.1	Kisill	48	Message Type: Heartbeat Reply
360	73.455548	127.0.0.1	127.0.0.1	Kisill	40	Message Type: Heartbeat
361	73.456121	127.0.0.1	127.0.0.1	Kisill	48	Message Type: Heartbeat Reply
362	77.744692	127.0.0.1	127.0.0.1	Kisill	52	Message Type: Name File
363	77.746290	127.0.0.1	127.0.0.1	Kisill	1248	Message Type: File
364	77.747658	127.0.0.1	127.0.0.1	Kisill	1248	Message Type: File
365	77.748044	127.0.0.1	127.0.0.1	Kisill	48	Message Type: ACK
366	77.749615	127.0.0.1	127.0.0.1	Kisill	1248	Message Type: File
367	77.749775	127.0.0.1	127.0.0.1	Kisill	48	Message Type: ACK
368	77.750376	127.0.0.1	127.0.0.1	Kisill	1248	Message Type: File
369	77.751438	127.0.0.1	127.0.0.1	Kisill	48	Message Type: ACK
370	77.751755	127.0.0.1	127.0.0.1	Kisill	1248	Message Type: File
371	77.752722	127.0.0.1	127.0.0.1	Kisill	48	Message Type: ACK
372	77.754322	127.0.0.1	127.0.0.1	Kisill	48	Message Type: ACK
373	77.755578	127.0.0.1	127.0.0.1	Kisill	1248	Message Type: File
374	77.756731	127.0.0.1	127.0.0.1	Kisill	1248	Message Type: File
375	77.756848	127.0.0.1	127.0.0.1	Kisill	48	Message Type: ACK
376	77.758017	127.0.0.1	127.0.0.1	Kisill	1248	Message Type: File
377	77.758258	127.0.0.1	127.0.0.1	Kisill	48	Message Type: ACK
378	77.759215	127.0.0.1	127.0.0.1	Kisill	1248	Message Type: File
379	77.759458	127.0.0.1	127.0.0.1	Kisill	48	Message Type: ACK
380	77.760383	127.0.0.1	127.0.0.1	Kisill	1248	Message Type: File
381	77.760723	127.0.0.1	127.0.0.1	Kisill	48	Message Type: ACK
382	77.761404	127.0.0.1	127.0.0.1	Kisill	1248	Message Type: File
383	77.761723	127.0.0.1	127.0.0.1	Kisill	48	Message Type: ACK
384	77.762445	127.0.0.1	127.0.0.1	Kisill	1248	Message Type: File
385	77.762730	127.0.0.1	127.0.0.1	Kisill	48	Message Type: ACK
386	77.763485	127.0.0.1	127.0.0.1	Kisill	1248	Message Type: File
387	77.763747	127.0.0.1	127.0.0.1	Kisill	48	Message Type: ACK
388	77.764523	127.0.0.1	127.0.0.1	Kisill	1248	Message Type: File
389	77.764790	127.0.0.1	127.0.0.1	Kisill	48	Message Type: ACK
390	77.765519	127.0.0.1	127.0.0.1	Kisill	1248	Message Type: File
391	77.765806	127.0.0.1	127.0.0.1	Kisill	48	Message Type: ACK
392	77.766576	127.0.0.1	127.0.0.1	Kisill	1248	Message Type: File
393	77.766801	127.0.0.1	127.0.0.1	Kisill	48	Message Type: ACK
394	77.767695	127.0.0.1	127.0.0.1	Kisill	1248	Message Type: File
395	77.767936	127.0.0.1	127.0.0.1	Kisill	48	Message Type: ACK
396	77.768675	127.0.0.1	127.0.0.1	Kisill	1248	Message Type: File
397	77.768815	127.0.0.1	127.0.0.1	Kisill	48	Message Type: ACK
398	77.769601	127.0.0.1	127.0.0.1	Kisill	1248	Message Type: File
399	77.769806	127.0.0.1	127.0.0.1	Kisill	48	Message Type: ACK
400	77.770663	127.0.0.1	127.0.0.1	Kisill	1248	Message Type: File
401	77.770736	127.0.0.1	127.0.0.1	Kisill	48	Message Type: ACK
402	77.771493	127.0.0.1	127.0.0.1	Kisill	1248	Message Type: File
403	77.771613	127.0.0.1	127.0.0.1	Kisill	48	Message Type: ACK
404	77.772491	127.0.0.1	127.0.0.1	Kisill	1248	Message Type: File
405	77.772581	127.0.0.1	127.0.0.1	Kisill	48	Message Type: ACK
406	77.773396	127.0.0.1	127.0.0.1	Kisill	1248	Message Type: File
407	77.773427	127.0.0.1	127.0.0.1	Kisill	48	Message Type: ACK
408	77.774358	127.0.0.1	127.0.0.1	Kisill	1248	Message Type: File
409	77.774377	127.0.0.1	127.0.0.1	Kisill	48	Message Type: ACK
410	77.775258	127.0.0.1	127.0.0.1	Kisill	1248	Message Type: File
411	77.775277	127.0.0.1	127.0.0.1	Kisill	48	Message Type: ACK
412	77.776096	127.0.0.1	127.0.0.1	Kisill	1248	Message Type: File
413	77.776576	127.0.0.1	127.0.0.1	Kisill	48	Message Type: ACK
414	77.776801	127.0.0.1	127.0.0.1	Kisill	1248	Message Type: File
415	77.776815	127.0.0.1	127.0.0.1	Kisill	48	Message Type: ACK
416	77.777944	127.0.0.1	127.0.0.1	Kisill	1248	Message Type: File
417	77.777973	127.0.0.1	127.0.0.1	Kisill	48	Message Type: ACK
418	77.778049	127.0.0.1	127.0.0.1	Kisill	1248	Message Type: File
419	77.778068	127.0.0.1	127.0.0.1	Kisill	48	Message Type: ACK
420	77.779773	127.0.0.1	127.0.0.1	Kisill	1248	Message Type: File
421	77.779794	127.0.0.1	127.0.0.1	Kisill	48	Message Type: ACK
422	77.780849	127.0.0.1	127.0.0.1	Kisill	1248	Message Type: File
423	77.780853	127.0.0.1	127.0.0.1	Kisill	48	Message Type: ACK
424	77.780748	127.0.0.1	127.0.0.1	Kisill	128	Message Type: File
425	77.781446	127.0.0.1	127.0.0.1	Kisill	48	Message Type: ACK
426	77.781549	127.0.0.1	127.0.0.1	Kisill	48	Message Type: ACK
427	83.461806	127.0.0.1	127.0.0.1	Kisill	48	Message Type: Heartbeat
428	83.461825	127.0.0.1	127.0.0.1	Kisill	48	Message Type: Heartbeat Reply
429	88.466456	127.0.0.1	127.0.0.1	Kisill	48	Message Type: Heartbeat
430	88.467157	127.0.0.1	127.0.0.1	Kisill	48	Message Type: Heartbeat Reply
431	93.471562	127.0.0.1	127.0.0.1	Kisill	48	Message Type: Heartbeat
432	93.471848	127.0.0.1	127.0.0.1	Kisill	48	Message Type: Heartbeat Reply
433	98.476222	127.0.0.1	127.0.0.1	Kisill	48	Message Type: Heartbeat
434	98.476379	127.0.0.1	127.0.0.1	Kisill	48	Message Type: Heartbeat Reply
441	101.479228	127.0.0.1	127.0.0.1	Kisill	48	Message Type: Heartbeat
442	101.479598	127.0.0.1	127.0.0.1	Kisill	48	Message Type: Heartbeat Reply
443	104.487206	127.0.0.1	127.0.0.1	Kisill	44	Message Type: Text
444	104.487385	127.0.0.1	127.0.0.1	Kisill	48	Message Type: ACK
445	104.487899	127.0.0.1	127.0.0.1	Kisill	44	Message Type: Text
446	104.488058	127.0.0.1	127.0.0.1	Kisill	48	Message Type: ACK

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:

350	62.738510	127.0.0.1	127.0.0.1	Kisil	45	Message Type: Text
351	62.738559	127.0.0.1	127.0.0.1	Kisil	45	Message Type: Text
352	62.738587	127.0.0.1	127.0.0.1	Kisil	45	Message Type: Text
353	62.738919	127.0.0.1	127.0.0.1	Kisil	40	Message Type: NACK
354	62.739044	127.0.0.1	127.0.0.1	Kisil	40	Message Type: ACK
355	62.739065	127.0.0.1	127.0.0.1	Kisil	45	Message Type: Text
356	62.739100	127.0.0.1	127.0.0.1	Kisil	40	Message Type: ACK
357	62.739162	127.0.0.1	127.0.0.1	Kisil	40	Message Type: ACK
358	62.739277	127.0.0.1	127.0.0.1	Kisil	40	Message Type: Heartbeat

> Frame 350: 45 bytes on wire (360 bits), 45 bytes captured (360 bits) on interface 0		0000	02 00 00 00 45 00 00 29 e3 20 00 00 00 11 00 00E..)
> Null/Loopback		0010	7f 00 00 01 7f 00 00 01 f1 88 c5 aa 00 15 fe 283..
> Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1		0020	00 00 00 00 00 00 33 08 61 73 64 66 613..
> User Datagram Protocol, Src Port: 61832, Dst Port: 50602				
Source Port: 61832				
Destination Port: 50602				
Length: 21				
Checksum: 0xfe28 [unverified]				
[Checksum Status: Unverified]				
[Stream index: 1]				
[Stream Packet Number: 4]				
> [Timestamps]				
UDP payload (13 bytes)				
My Protocol Data				
Fragment Number: 0				
Last Fragment: No (0)				
Message Type: Text (0)				
CRC: 0x3308				

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

362	77.744692	127.0.0.1	127.0.0.1	Kisil	52	Message Type: Name File
363	77.746290	127.0.0.1	127.0.0.1	Kisil	1240	Message Type: File
364	77.747650	127.0.0.1	127.0.0.1	Kisil	1240	Message Type: File
365	77.748044	127.0.0.1	127.0.0.1	Kisil	40	Message Type: NACK
366	77.749015	127.0.0.1	127.0.0.1	Kisil	1240	Message Type: File
367	77.749775	127.0.0.1	127.0.0.1	Kisil	40	Message Type: ACK
368	77.750378	127.0.0.1	127.0.0.1	Kisil	1240	Message Type: File
369	77.751430	127.0.0.1	127.0.0.1	Kisil	40	Message Type: ACK
370	77.751755	127.0.0.1	127.0.0.1	Kisil	1240	Message Type: File
371	77.752722	127.0.0.1	127.0.0.1	Kisil	40	Message Type: ACK
372	77.754322	127.0.0.1	127.0.0.1	Kisil	40	Message Type: ACK

Zobrazí sa začiatok odosielenia 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 odosielenie 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/WflhQ3o2xow?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/#>