SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE

Fakulta informatiky a informačných technológií
Ilkovičova 2, 842 16 Bratislava 4

Zadanie č. 2 – Komunikácia s využitím UDP protokolu

POČÍTAČOVÉ A KOMUNIKAČNÉ SIETE

Róbert Junas

FIIT STU

Cvičenie: Štvrtok 14:00

24.11.2021

1. Zadanie

Zadanie úlohy

Navrhnite a implementujte program s použitím vlastného protokolu nad protokolom UDP (User Datagram Protocol) transportnej vrstvy sieťového modelu TCP/IP. Program umožní komunikáciu dvoch účastníkov v lokálnej sieti Ethernet, teda prenos textových správ a ľubovoľného súboru medzi počítačmi (uzlami).

Program bude pozostávať z dvoch častí – vysielacej a prijímacej. Vysielací uzol pošle súbor inému uzlu v sieti. Predpokladá sa, že v sieti dochádza k stratám dát. Ak je posielaný súbor väčší, ako používateľom definovaná max. veľkosť fragmentu, vysielajúca strana rozloží súbor na menšie časti - fragmenty, ktoré pošle samostatne. Maximálnu veľkosť fragmentu musí mať používateľ možnosť nastaviť takú, aby neboli znova fragmentované na linkovej vrstve.

Ak je súbor poslaný ako postupnosť fragmentov, cieľový uzol vypíše správu o prijatí fragmentu s jeho poradím a či bol prenesený bez chýb. Po prijatí celého súboru na cieľovom uzle tento zobrazí správu o jeho prijatí a absolútnu cestu, kam bol prijatý súbor uložený. Program musí obsahovať kontrolu chýb pri komunikácii a znovuvyžiadanie chybných fragmentov, vrátane pozitívneho aj negatívneho potvrdenia. Po prenesení prvého súboru pri nečinnosti komunikátor automaticky odošle paket pre udržanie spojenia každých 5 -20s pokiaľ používateľ neukončí spojenie. Odporúčame riešiť cez vlastne definované signalizačné správy.

Program musí mať nasledovné vlastnosti (minimálne):

1. Program musí byť implementovaný v jazykoch C/C++ alebo Python s využitím knižníc na prácu s UDP socket, skompilovateľný a spustiteľný v učebniach. Odporúčame použiť python modul socket, C/C++ knižnice sys/socket.h pre linux/BSD a winsock2.h pre Windows. Iné knižnice a funkcie na prácu so socketmi musia byť schválené cvičiacim. V programe môžu byť použité aj knižnice na prácu s IP adresami a portami:

   arpa/inet.h
   netinet/in.h

2. Program musí pracovať s dátami optimálne (napr. neukladať IP adresy do 4x int).
3. Pri posielaní súboru musí používateľovi umožniť určiť cieľovú IP a port.
4. Používateľ musí mať možnosť zvoliť si max. veľkosť fragmentu.
5. Obe komunikujúce strany musia byť schopné zobrazovať:
   a. názov a absolútnu cestu k súboru na danom uzle,
   b. veľkosť a počet fragmentov.

6. Možnosť simulovať chybu prenosu odoslaním minimálne 1 chybného fragmentu pri prenose súboru (do dátovej časti fragmentu je cielene vnesená chyba, to znamená, že prijímajúca strana deteguje chybu pri prenose).
7. Prijímajúca strana musí byť schopná oznámiť odosielateľovi správne aj nesprávne doručenie fragmentov. Pri nesprávnom doručení fragmentu vyžiada znovu poslať poškodené dáta.
8. Možnosť odoslať 2MB súbor a v tom prípade ich uložiť na prijímacej strane ako rovnaký súbor, pričom používateľ zadáva iba cestu k adresáru kde má byť uložený.

Odovzdáva sa:

1. Návrh riešenia
2. Predvedenie riešenia v súlade s prezentovaným návrhom

Program musí byť organizovaný tak, aby oba komunikujúce uzly mohli prepínať medzi funkciou vysielača a prijímača bez reštartu programu - program nemusí (ale môže) byť vysielač a prijímač súčasne. Pri predvedení riešenia je podmienkou hodnotenia schopnosť doimplementovať jednoduchú funkcionalitu na cvičení.

Hodnotenie

Celé riešenie - max. 20 bodov (min. 7), z toho:

- max. 5 bodov za návrh riešenia;

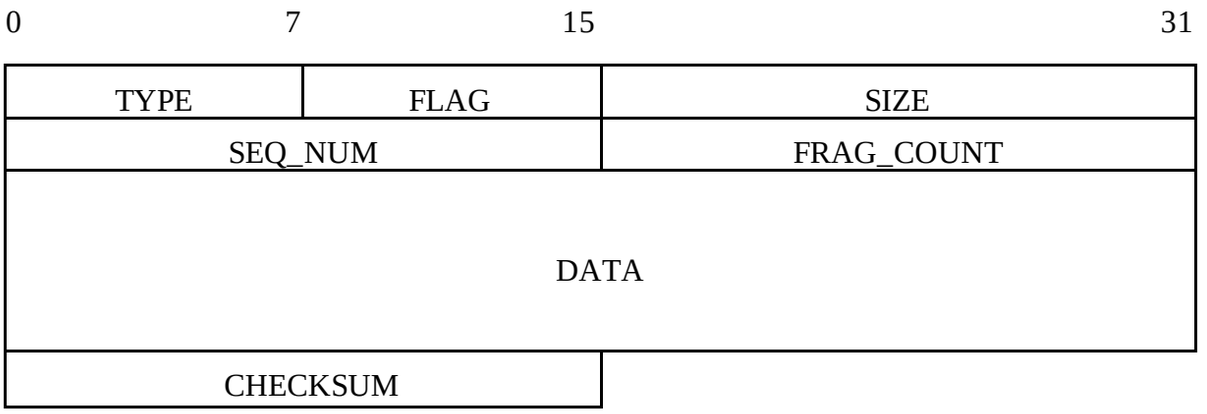
- max. 1 bod za doplnenú funkčnosť (doimplementáciu) priamo na cvičení v požadovanom termíne podľa harmonogramu cvičení; V prípade, ak študent nesplní úlohu zadanú priamo na cvičeniach, nehodnotí sa výsledné riešenie;

- max. 14 bodov za výsledné riešenie.

Návrh a zdrojový kód implementácie študent odovzdáva v elektronickom tvare do AISu v určených termínoch.

2. Návrh

2.1. Hlavička



Rámec môže mať niekoľko typov, ktoré určujú ako sa k nim bude program správať:

- Control – 0x00 – na kontrolné rámce, ako ukončenie, udržanie spojenia alebo na zistenie či je cieľ dosiahnuteľný,
- Text – 0x01 – na posielanie textovej správy,
- File – 0x02 – posielanie súboru.

Bajt, v ktorom je pole flag, určuje o akú správu sa jedná:

- SYN – 0x01
- FIN – 0x02
- ACK – 0x04
- RESEND – 0x08
- KEEP-ALIVE – 0x10
- SWAP – 0x20

FRAG_COUNT a SEQ_NUM majú veľkosť 2 bajty, čo pri maximálnej veľkosti dát 500 bajtov dokáže preniesť 65535*500 = ~32,767 MB, čo je dostačujúce na zadanie.

2.2. Kontrola chýb

Kontrola chýb, resp. checksum pridaný na koniec posielaného rámca, bude vykonávaná pomocou CRC16 s polynómom 0x1021 (4129), resp. 0x11021. Táto funkcionalita bude zložená z dvoch funkcií – jednej na overenie korektnosti prijatých dát a druhej pre vytvorenie kontrolnej hodnoty.

Výpočet bude prebiehať tak, že sa na koniec dát, ktoré chceme zahashovať sa pridajú 2 bajty, ktoré budú reprezentovať checksum. Následne bude v dátach hľadať prvý bit nastavený na 1 a od neho urobí XOR s polynómom a uloží sa do výsledku. Následne sa výsledku budú pridávať ďalšie bity z dát a ak výsledok bude od svojej prvého bitu nastaveného na 1 bude dlhý ako polynóm, tak sa znovu vykoná XOR operácia a zapíše sa do výsledku. Táto operácia sa bude opakovať pokiaľ neprejdeme všetky bity v dátach. Na koniec dát sa pridá vypočítané crc.

Kontrola prebehne rovnako s tým, že sa proces vykoná pre celý prijatý rámec aj s crc a ak po vykonaní výpočtu budú posledné bity (tie na ktorých je uložené crc) samé 0, tak rámec prišiel v poriadku. Inak nebol v poriadku a vyžiada, aby bolo okno poslané znovu.

2.3. ARQ

ARQ metóda použijeme Go Back-N protokol, ktorý si udržiava posuvné okno N rámcov, ktoré musia byť odoslané. Toto okno znovu posiela N rámcov v prípade, že pre prvý rámec v okne neprišlo od prijímača potvrdenie ACK, resp. pri prípadoch, že správa nepríde na prijímač, bude poškodená alebo sa odpoveď ACK stratí. Znovu poslanie bude riešené časovačom na strane vysielača a to práve časovačom, ktorý bude nastavený na pár sekúnd. Ak vysielač dostane odpoveď, tak sa celé okno posunie, časovač sa zresetuje, odošle rámec, ktorý pribudol do okna a čaká na ACK pre ďalší rámec. Ak prídu rámce v zlom poradí, tak rámce budú ignorované, teda sa prijímač ich nezoberie – neodošle pre nich ACK správu.

Ďalej naša implementácia použije aj signalizačnú správu NACK, resp. RESEND + ACK, ktorá by ihneď odoslala celé okno znova, bez čakania či vlastne.

2.4. Udržiavanie spojenia

Po prenesení súboru na prijímač sa spustí posielanie KEEP-ALIVE rámcov, ktoré sa budú posielať každých 10 sekúnd. Ak odpoveď na tieto správy nepríde do 3 pokusov, tak sa spojenie bude brať za ukončené. Rovnako, ak do 10 sekúnd od poslania posledného rámca KEEP ALIVE alebo posledného rámca prenosu dát nepríde ďalší KEEP ALIVE rámec tak sa spojenie preruší. Prípadne sám prijímač alebo vysielač vyšlú požiadavku na ukončenie spojenia alebo na poslanie ďalšieho súboru, v oboch prípadoch sa prestanú posielať KEEP-ALIVE správy.

2.5. Fragmentácia

Môže sa stať, že chceme poslať dáta, ktoré sú väčšie ako je maximálne daná veľkosť jedného rámca. Pri takýchto prípadoch sa správa rozfragmentuje na viac častí, ktoré sa odosielajú po poradí prijímaču, resp. dajú sa do radu packetov, čakajúcich na odoslanie. Následne sa v prijímači poskladajú v poradí akom prišli, pretože nami zvolená ARQ metóda zaručuje, že rámce prídu v poradí akom majú.

Maximálna veľkosť fragmentu bude podobná ako pri TFTP, kde každý prenášaný blok dát má hodnotu najviac 512 bajtov a veľkosť hlavičky je 4 bajty. Pre náš protokol to bude fungovať podobne s tým, že máme väčšiu hlavičku (16 bajtov), teda maximálna veľkosť fragmentovaného bloku bude ~500 bajtov a minimálna bude 1 bajt (pre prenos 2MB súboru by minimum malo byť okolo 40 Bajtov). Rovnako sme sa dozvedeli, že najbezpečnejšia veľkosť pre posielanie nad IPv4 je z 576 bajtov a posielanie väčších rámcov je v poriadku, ak vieme

zaručiť, že sme ho schopný preniesť (https://www.rfc-editor.org/rfc/pdfrfc/rfc791.txt.pdf). Preto viac ako 576 bajtov nebudeme ani podporovať.

2.6. Výmena rolí

Pri výmene rolí pošle vysielač prijímaču požiadavku na výmenu pomocou flagu SWAP, ktorý po prijatí zmení svoj stav z počúvania na posielanie a odošle potvrdenie o výmene role, keď vysielač príjme odpoveď tak zmení svoj stav.

Pred samotnou výmenou musí vysielač poslať aspoň jednu správu na nadviazanie spojenia s prijímačom.

2.7. Začiatok spojenia

Pri začiatku spojenia sa pridá SYN rámec do radu a nastaví sa veľkosť okna na 1 rámec, dokým nepríde potvrdzovací rámec od prijímača. Po tom ako príde potvrdzovací rámec, tak sa zväčší okno na viac rámcov. Zároveň sa prijímač nastaví aby neprijímal rámce od iných zariadení.

2.8. Časti kódu

2.8.1 Knižnice

- Socket
- Threading
- Time – potrebné pre časovač nad ARQ

2.8.2 Prijímač

- Zadať cieľový port, IP adresu a maximálnu veľkosť fragmentu
- Poslať korektnú správu ako aj simulovať chybu
- vypísať názov a absolútnu cestu k súboru, veľkosť a počet fragmentov
- prijať súbor od vysielača
- poslať správu
- ukončiť spojenie

2.8.3 Vysielač

- Nastaviť si port na počúvanie
- Nastaviť miesto kam sa súbor uloží
- Potvrdiť prijatie rámcov
- Pospájať fragmenty
- Vykonať výmenu s klientom
- Detegovať chybu a vyžiadať znovuposlanie rámca
- Oznámiť prijatie nesprávnej správy
- Oznámiť poradie prijatého fragmentu
- Uložiť súbor, vypísať kam sa uložil a počet fragmentov

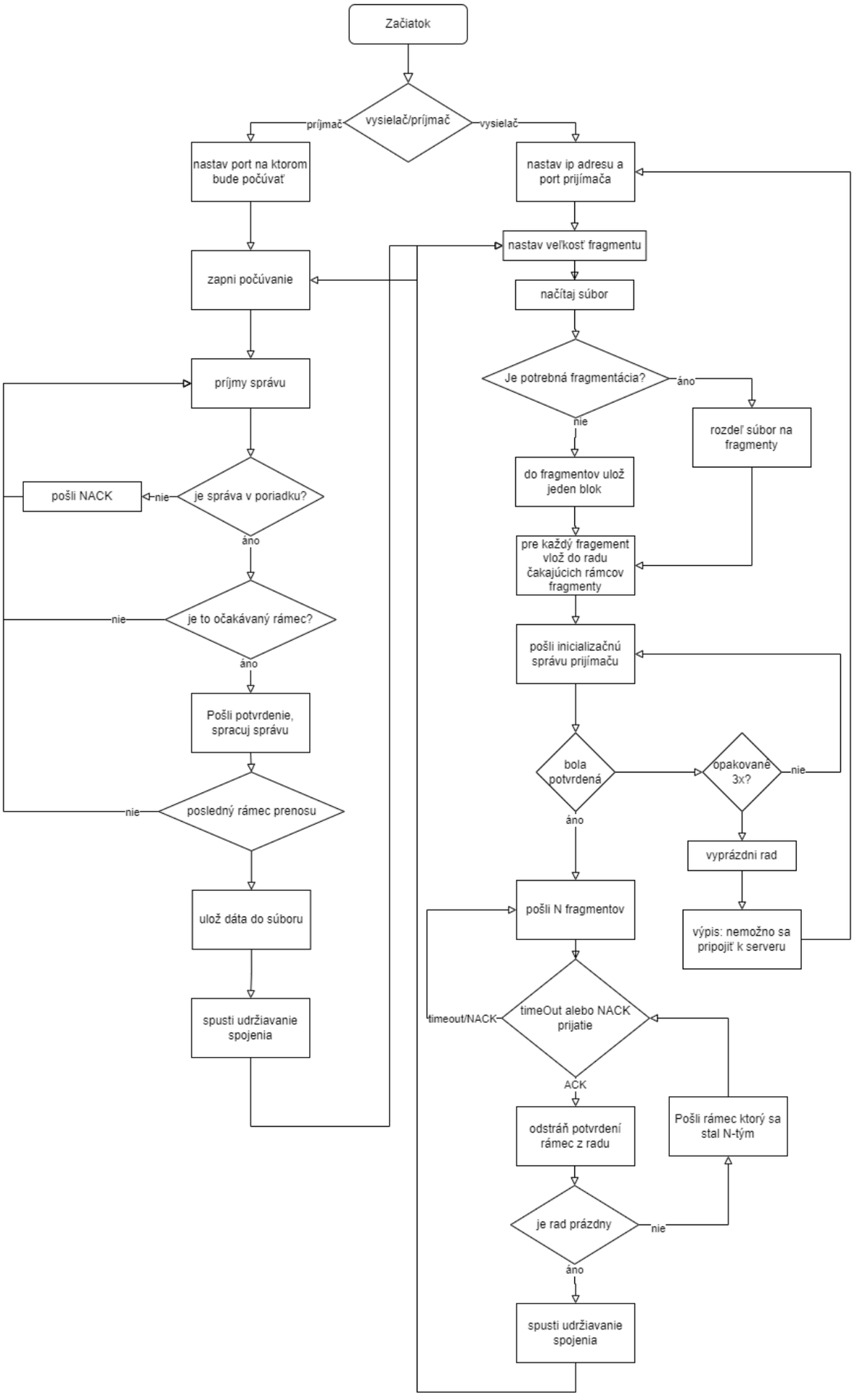
2.8.4 Štruktúry

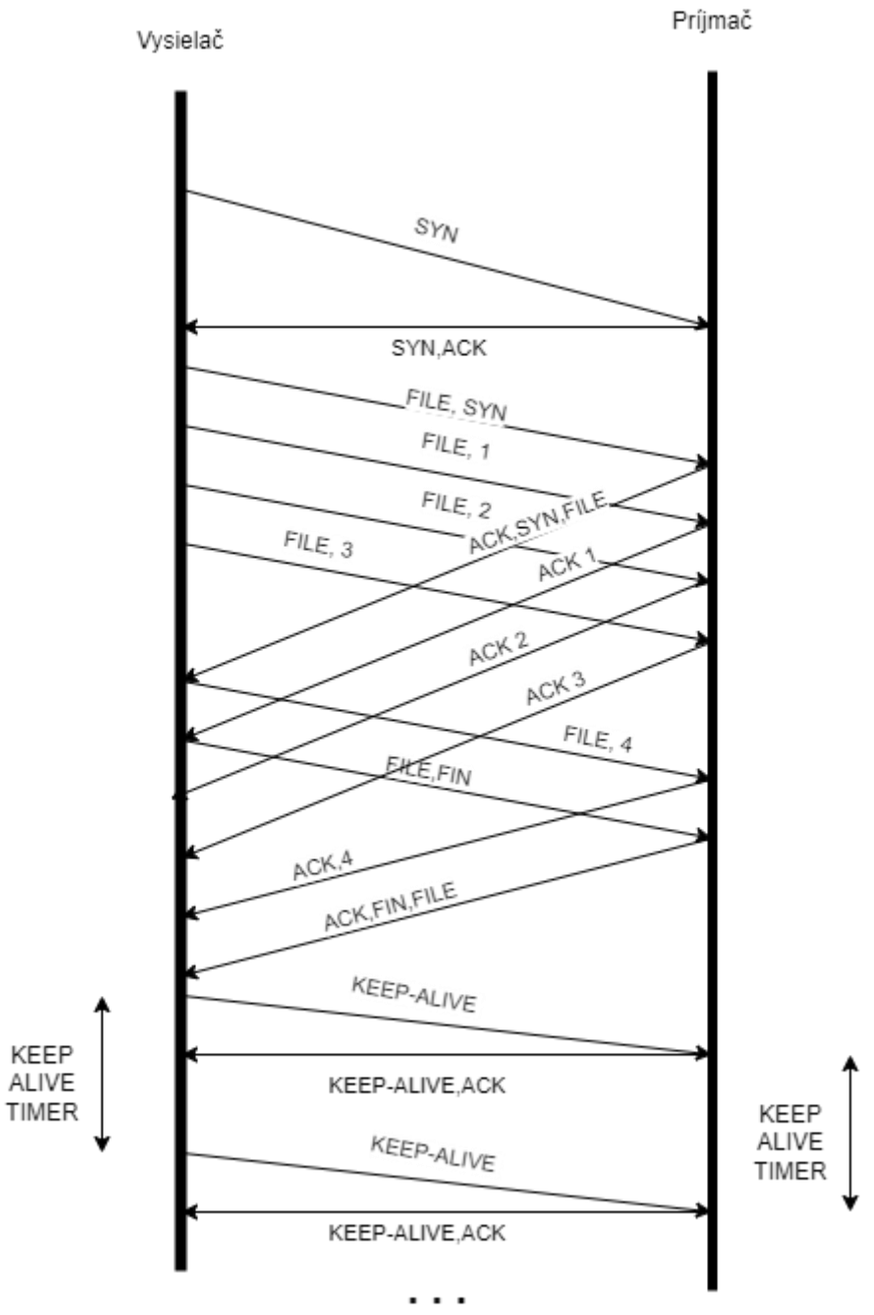
- Spojenie – všetky potrebné údaje na posielanie a získavanie dát
- Konštanty – všetky konštanty ako sú flag-y, typy
- Rámec – štruktúra hlavičky

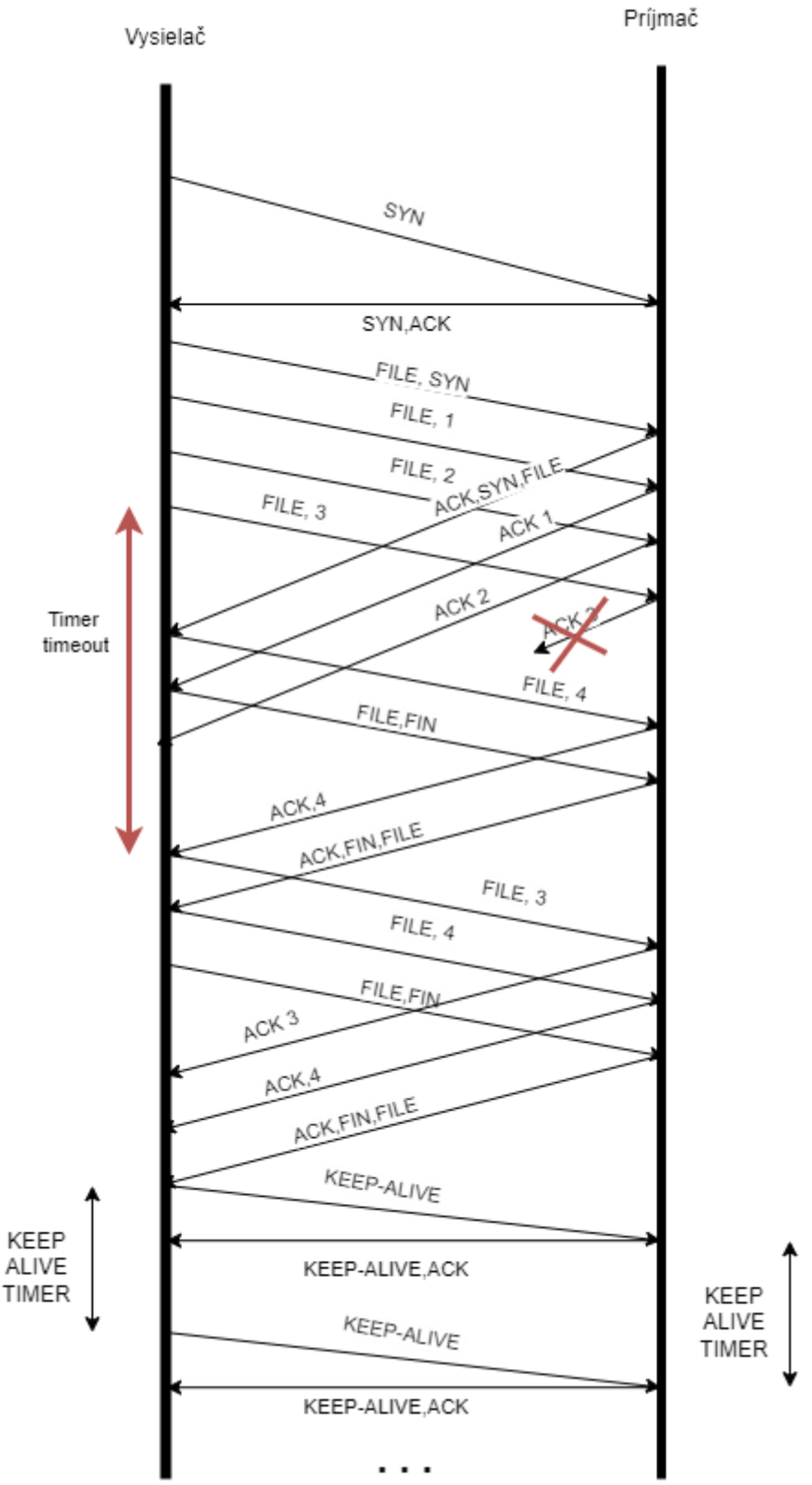
2.8.5 Thready

Program sa bude skladať z troch vlákien a jedného hlavného, v ktorom budeme brať požiadavky od používateľa. Jedno vlákno bude použité na prijímanie a druhé na posielanie rámcov. Tretie vlákno bude slúžiť na posielanie KEEP-ALIVE rámcov a aj na zistenie či prišiel KEEP-ALIVE rámec do určitého času.

2.9. Diagram fungovania




Vysielač
Príjmač
SYN
SYN,ACK
FILE, SYN
FILE, 1
FILE, 2
FILE, 3
ACK,SYN,FILE
ACK 1
ACK 2
ACK 3
FILE, 4
FILE,FIN
ACK,4
ACK,FIN,FILE
KEEP-ALIVE
KEEP-ALIVE,ACK
KEEP-ALIVE
KEEP-ALIVE,ACK
KEEP ALIVE TIMER
KEEP ALIVE TIMER
. . .


Vysielač
Príjmač
SYN
SYN,ACK
FILE, SYN
FILE, 1
FILE, 2
ACK,SYN,FILE
FILE, 3
ACK 1
ACK 2
Timer
timeout
FILE, 4
FILE,FIN
ACK,4
ACK,FIN,FILE
FILE, 3
FILE, 4
FILE,FIN
ACK 3
ACK,4
ACK,FIN,FILE
KEEP-ALIVE
KEEP
ALIVE
TIMER
KEEP-ALIVE,ACK
KEEP-ALIVE
KEEP
ALIVE
TIMER
KEEP-ALIVE,ACK
. . .