STU FIIT

Komunikácia s využitím UDP protokolu

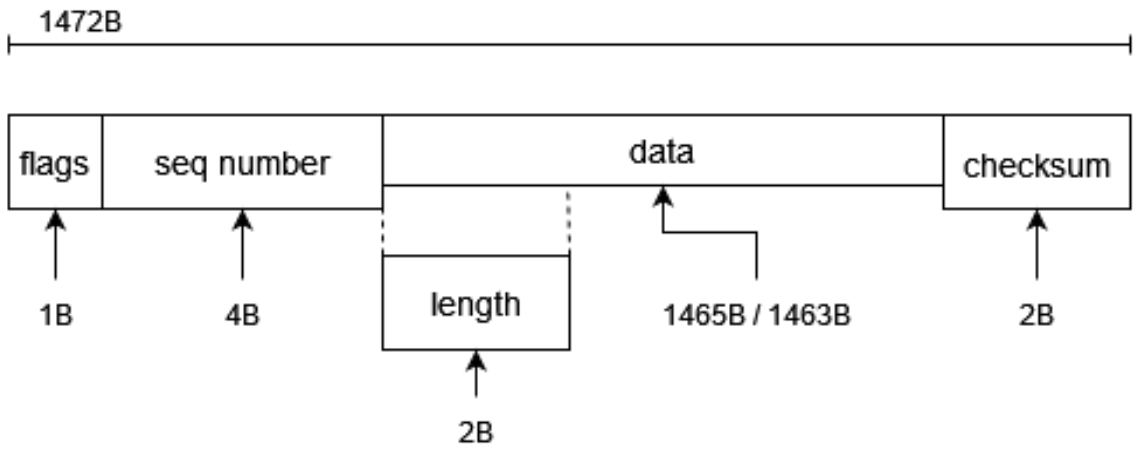
Marko Bukovina
Cvčenie: Ing. Ladislav Zemko, Štvrtok 8.00

Obsah

Implementačné prostredie

Ako moje implementačné prostredie som si zvolil **Python**. A taktiež budem využívať knižnicu **crcmod** ktorá obsahuje všetky potrebné **CRC**(Cyclic Redundancy Check) funkcie na overenie integrity dát.

Štruktúra hlavičky

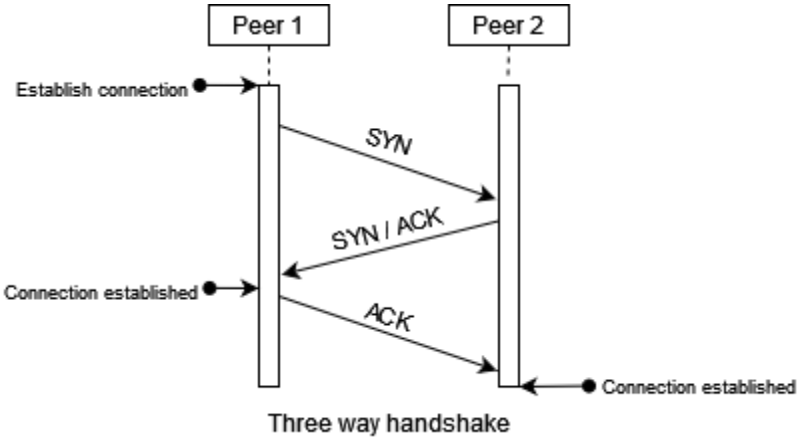


Flag	**Hodnota**	**Význam**
SYN	**1**	
ACK	**2**	
HB	**3**	Heartbeat – Keep alive
FIN	**4**	
STR	**5**	Začiatok posielania dát obsahuje veľkosť fragmentov
FRAG	**7**	Fragment
FRAG_F	**8**	Posledný fragment – obsahuje pole length
MSG	**11**	Packet obsahuje správu

Štruktúra hlavičky bude obsahovať nasledovné polia, **flags**, **sequence number** za pomoci kombinácie flagov a squence number môžem nasimulovať **acknowledge number** no tento spôsob môže podstúpiť zmenám vo finálnej implementácii. Takto môžem ušetriť trošku limitovanej veľkosti packetov. Takťiež môžeme pridať príležitostné pole **length** iba v prípade kedy ho budeme potrebovať a to v poslednom fragmente prenášaneho súboru a pri posielaní správ.

Nadviazanie spojenia – Three way handshake

Na overenia správneho nadviazania spojenia som použil jednoduchší **TCP three way handshake**. Program spustí vlákno na počúvanie prichádzajúceho spojenia. V prípade že zachytíme synchronizačný packet spustíme sekvenciu na obrázku. Program bude mať taktiež možnosť na zaslanie synchronizačného packetu na cieľovú adresu



Three way handshake

Overenie integrity

Cyklická redundančná kontrola je široko používaný mechanizmus na zisťovanie chýb v sieťovej komunikácii a pri ukladaní dát. Spočíva v generovaní krátkej binárnej sekvencie pevnej dĺžky z väčšieho bloku dát. Táto hodnota sa potom pripojí k dátam pred ich prenosom. Po prijatí sa dáta a CRC znova vypočítajú a porovnajú. Ak sa nezhodujú, predpokladá sa, že dáta boli poškodené.

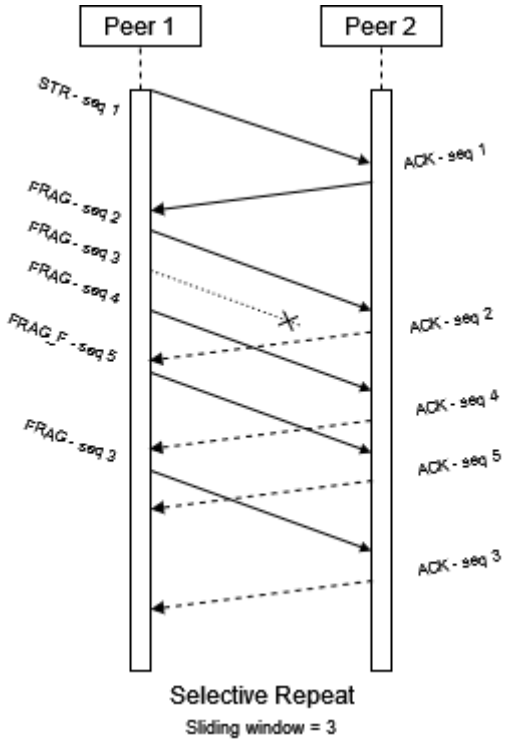
Ako moju metódu overenia integrity dát som si zvolil **crc-16** keďže sa domnievam že crc-32 je z časti nepotrebne rozliahli na toto zadanie ale taktiež uznávam že je to štandard v Ethernet II komunikácii.

Výpočet **CRC-16:**

- Rozdelime data ktore chceme delit na sekvenciu bitov a nakoniec pridame padding 16bitov núl keďže používame crc-16
- Nasledne túto sekvenciu delíme polynómom $x^{16}+x^{12}+x^5+1$. Formov XOR bitovej divízie.
- Po videlení čísla dostávame vyberieme zvyšok ktorý uložíme do packetu na budúce porovnanie po príchode do cieľa
- Ak sa hodnota líši tak došlo k poruche dát

ARQ Selective repeat

Ako moju metodu prenosu dát som si zvolil **ARQ Selective repeat.** Na začiatku spojenia pošle program veľkosť fragmentov ktorú si používateľ na druhej strane zvolil tento špecialny packet bude mať flag **STR**.
SR bude požívať takzvané sliding window, takže metóda môže poslať iba N packetov bez toho aby dostala správu o ich prijatí v prípade že dostane správu o prijatí odošle ďalší packet/packety v prípade že nedostaneme **ACK** pre packet ktorý sme zaslali ale dostaneme packet ktorý sme zaslali po ňom opakuje sap renos packetu.



Udržiavanie spojenia – Keep Alive

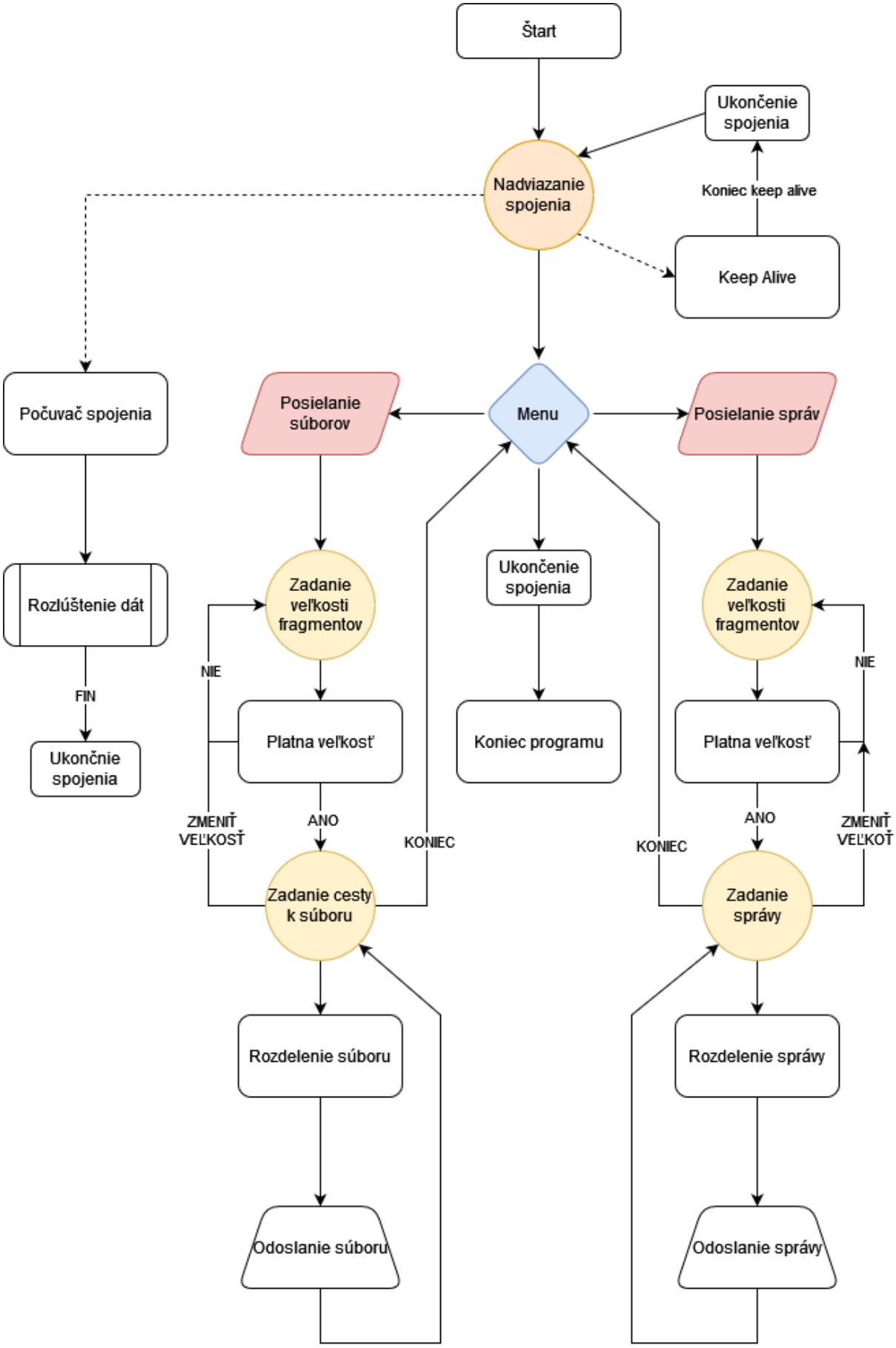
Funkcia bude zasielať periodicky heartbeat flagy v prípade a taktiež počúvať spojenie pre prichádzajúce heart beaty ak program neobdrží *N* heartbeatov po sebe, spustí sa terminácia spojenia a aplikácia sa ukončí.

Simulácia poškodených dát

Poškodené dáta vytvorým obrateným jedného alebo viacerých bitov v už pripravenom packete na odoslanie, keďže bude kontrolna hodnota už vypočítana po opätovnom výpočte na cieľovom uzle dôjde k chybe.

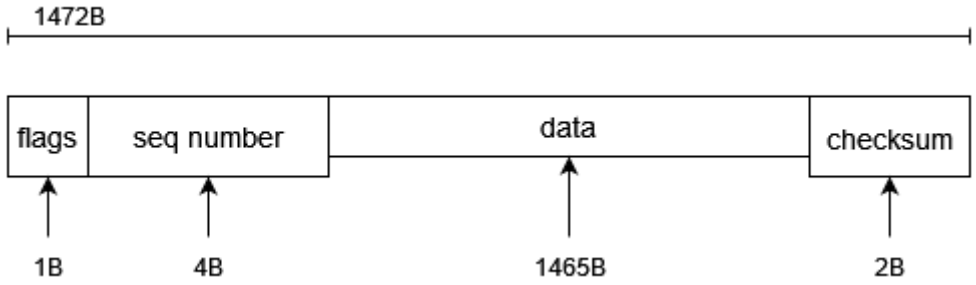
Funkcionalita program

Na začiatku programu bude potrebné uviesť počúvaci port a cieľovú adresu a cieľový port. Aplikácia následne spustí vlákno na počúvanie prichádzajúceho TCP handshaku a taktiež sa spustí separatné vlaknu na posielanie synchronizačných packetov na inicializaciu TCP handshaku v prípade že si to tak zvolíme.
Po úspešnom nadviazaní spojenia sa sa vypíše menu možností z ktorých si môžeme vybrať sposob odosielanie dát a taktiež možnosť na ukončenie programu. Spustia sa dve dôležité asynchronné funkcie **keep_alive** a hlavný počúvač prichádzajúcich packetov tento bude mať na starosti riadiť všetky prichádzajúce packety do ich náležitostných funkcií.
V prípade oboch módov odosielanie si zvolíme veľkosť fragmentov po overení ich správnej veľkosti a dáta sa rozdelia na veľkosť ktorá bola zadaná nakoniec sa odošle **STR** packet aby oboznámil cieľový uzol o veľkosti ktorú má očakávať. Po prijatí tohto packetu sa začnú odosielať dáta spôsobom **ARQ Selective repeat**. Konkrétne budú dáta na oboch stranách rozdelené do hash máp aby bolo jednoduchšie spravovať ich správne doručenie pomocou ich sekvenčného čísla.


Štart
Ukončenie spojenia
Nadviazanie spojenia
Koniec keep alive
Keep Alive
Počuvač spojenia
Posielanie súborov
Menu
Posielanie správ
Rozlúštenie dát
Zadanie veľkosti fragmentov
Ukončenie spojenia
Zadanie veľkosti fragmentov
NIE
FIN
Platna veľkosť
Koniec programu
Platna veľkosť
NIE
Ukončnie spojenia
ZMENIŤ VEĽKOSŤ
ANO
KONIEC
KONIEC
ANO
ZMENIŤ VEĽKOŤ
Zadanie cesty k súboru
Zadanie správy
Rozdelenie súboru
Rozdelenie správy
Odoslanie súboru
Odoslanie správy

Zmeny vykonané v implementácii

Hlavička



Odstránil som pole **length** oproti originalu keďže som pre dané pole nemal využitie. Posledný fragment má príponu flagu **_F** a jeho daná velkosť sa dá vypočítať po jeho zachytení.

Zmena v počte flagov bola taktiež minimálna jediný pridaným prvkov bol **KAACK – Keep alive acknowledge** ktorý zjednodušil pracovanie s keep alive funkciou.

Detailná implemntácia komunikátora

Písanie do konzoly

Kvôli početným problémom som sa rozhodol vytvoriť samostatné vlákno na čítanie vstupu do konzoly toto vlákno načítava vstup a okamžite ho zapisuje do asynchronnej rady **queue.Queue()** v pythone. Následne sa všetok potrebný vstup odoberá z tejto rady.

Spracovanie prichádzajúcich packetov

Na spracovanie packetov používam dve vlákna jedno vlákno zachytáva packety a vkladá ich do rady prijatých packetov. Druhé vlákno vyberá packety z tejto rady a spracováva ich kontroluje ich checksum, rozhoduje ako ich spracovať podla flagu atď.

Z počiatku sa o príjmanie a spracovanie staral jedeno vlákno no po analýze spojenia som zistil že veľa packetov sa prehliadne kvôli prebiehajúcim výpočtom.

Keep alive

Funkcia keep alive kontroluje čas kedy naposledy bol prijatý nejaký packet. Ak je tento čas vyšší ako 5 sekúnd tak sa odošle heartbeat packet. ACK na tento heartbeat resetuje časovač naposledy prijatého packet. V prípade že tento ack nepríde viackrát resp. 3krát spojenie sa zruší.

Posielanie suborov

Posielanie suborov a textových správ funguje na veľmi podobnom princípe. Po zvolení danej možnosti v menu sa zadá správa alebo cesta k súboru. Načítané dáta sa rozdelia podľa zadanej veľkosti fragmentov a odošlú sa do posielača packetov ktorý je implementáciou ARQ selective repeat

ARQ Selective repeat

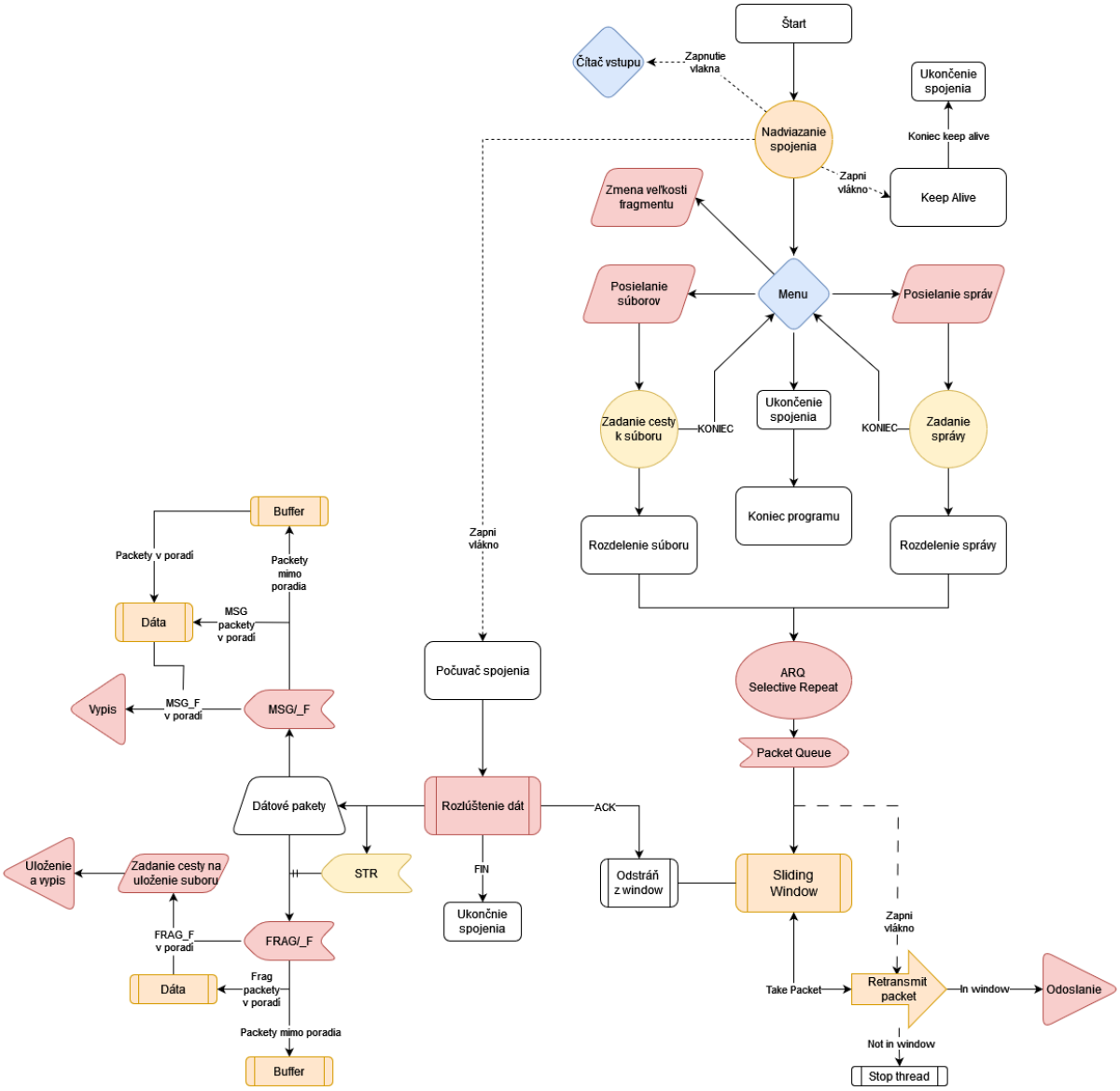
Packety sa zadajú do mapy/dictionary/**sliding window** podla sekvenčného čísla **ACK** packetu na packet vložený do sliding window. Následne sa spustí samostatný thread ktorý neustále posiela tento packet, s časovým rozmedzím, až kým nám nedôjde **ACK** ktorý následne odoberie packet zo **sliding window,** teda posielanie tohto packetu sa zruší a **sliding window** sa môže naplniť o nové packety.

Moja implementacia **ARQ Selective receive** funguje na princípe **žiaden ACK = NACK.**

Príjimanie dát

Dáta sa dajú rozdeliť na dva typy fragmenty a posledný fragment. Po príchode dát sa skontroluje či nám príšlo očakávane sekvenčne číslo ak áno tak sa priradí k dátam ak nám prišlo väčšie sek. č. tak ho priradime do dočasneho bufferu a po príchode správneho packetu sa tento buffer prehľadá aby sme mohli priradiť ďalšie dáta v sekvencii.

V prípade že došlo k chybe počas prenosu teda nikdy nepríde finalny packet alebo nikdy nam nepride dalsi packet v poradi tak sa dáta v pamäti vmažú a všetky prichádzajúce packety typu **FRAG** sa začnú ignorovať čím sa dá najavo posielači na druhej strane že sa niečo pokazilo a všetky dáta na posielanie sa vyčistia


Štart
Čítač vstupu
Zapnutie vlakna
Ukončenie spojenia
Nadviazanie spojenia
Koniec keep alive
Zapni vlákno
Keep Alive
Zmena veľkosti fragmentu
Menu
Posielanie súborov
Posielanie správ
Zadanie cesty k súboru
KONIEC
Ukončenie spojenia
KONIEC
Zadanie správy
Koniec programu
Rozdelenie súboru
Rozdelenie správy
Zapni vlákno
Počuvač spojenia
ARQ Selective Repeat
Packet Queue
Buffer
Packety v poradí
Packety mimo poradia
Dáta
MSG packety v poradí
Vypis
MSG_F v poradí
MSG/_F
Dátové pakety
Rozlúštenie dát
ACK
Odstráň z window
Sliding Window
Uloženie a vypis
Zadanie cesty na uloženie suboru
STR
FIN
Ukončnie spojenia
FRAG_F v poradí
FRAG/_F
Dáta
Frag packety v poradí
Packety mimo poradia
Buffer
Zapni vlákno
Take Packet
Retransmit packet
In window
Odoslanie
Not in window
Stop thread