Slovenská technická univerzita

Fakulta informatiky a informačných technológií

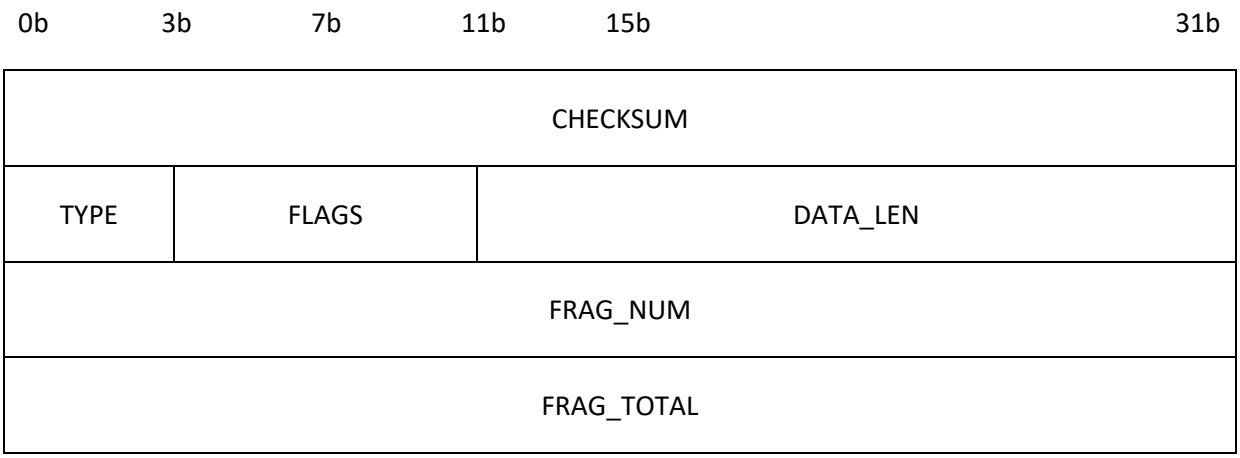
Návrh zadania č. 2

Komunikácia s využitím UDP protokolu

Meno: Dominik Mifkovič

Obdobie: ZS 2022/2023

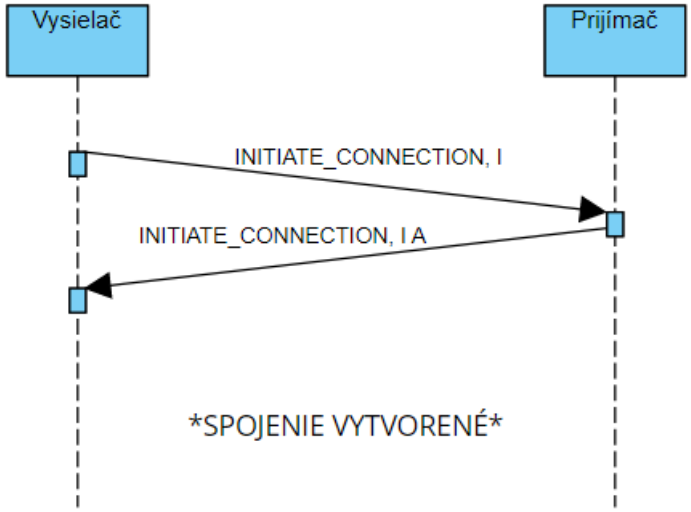
Návrh hlavičky



- CHECKSUM – 4B (je potrebných 32 bitov pretože checksum bude vypočítaný pomocou crc32 funkcie)
- TYPE – 4b
  - 0001 – packet bez dát
  - 0010 – packet pre posielanie textu
  - 0100 – packet pre posielanie súborov
- FLAGS – 1B
  - I – Initiate connection
  - A – Acknowledge
  - E – End connection
  - K – Keep alive
  - N – New data stream
  - R – Request again
  - C – Data stream complete
  - S – Swap mode (receiver/sender)
- DATA_LEN – 20b, dĺžka pola pre dáta
- FRAG_NUM – 4B, poradie daného fragmentu
- FRAG_TOTAL – 4B, totálne množstvo fragmentov pre daný stream

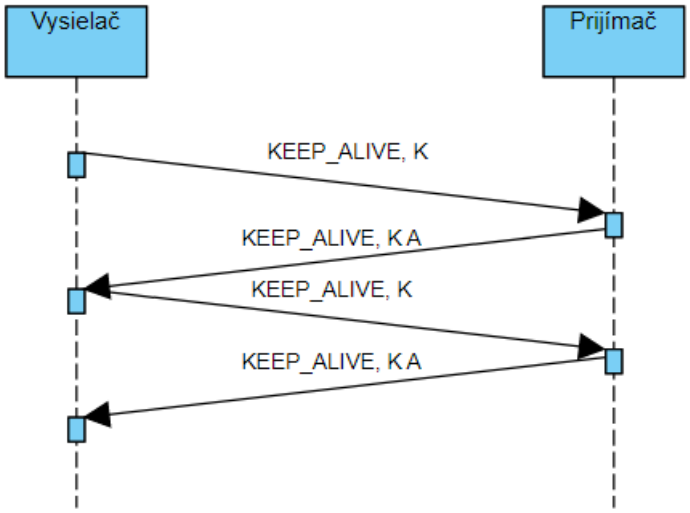
Nadviazanie spojenia

Nadviazanie prvotného spojenia prebehne v podobe 2-way handshake-u. Najskôr predom nastavený vysielač odošle packet bez dát čiže políčko TYPE bude mať hodnotu 0. Tu sa nastaví flag I (initiate connection) a packet sa odošle prijímateľovi. Po úspešnom prijatí prijímateľ odošle späť packet toho istého typu ale s flagmi I a A (Initialize, Acknowledge). Keď vysielač tento packet prijme, spojenie bolo nadviazané.



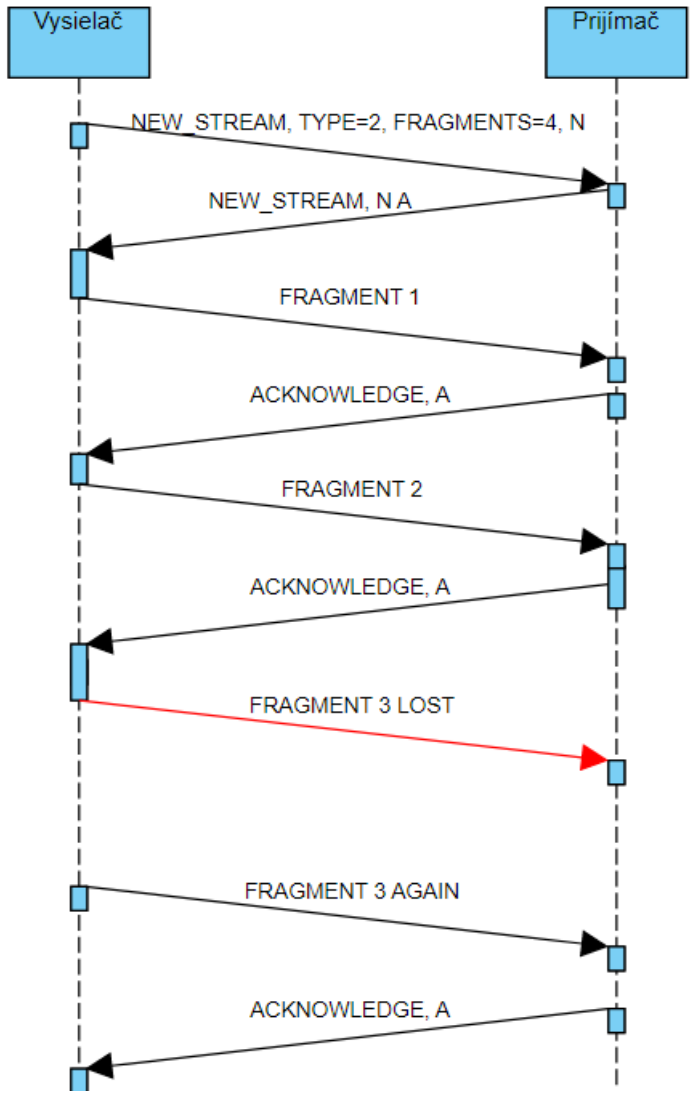
Udržiavanie spojenia

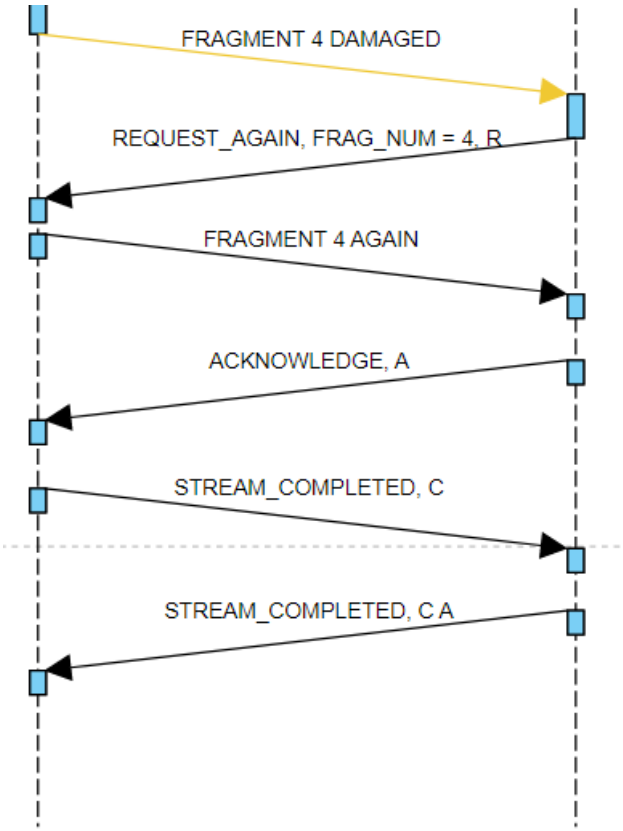
Udržiavanie spojenia bude prebiehať jednoducho. Vysielač najskôr odošle packet typu 0 s flagom K (Keep alive). Po úspešnom prijatí prijímateľ odošle späť packet toho istého typu ale s flagmi K a A (Keep alive, Acknowledge). Tento proces sa opakuje.



Odosielanie dát

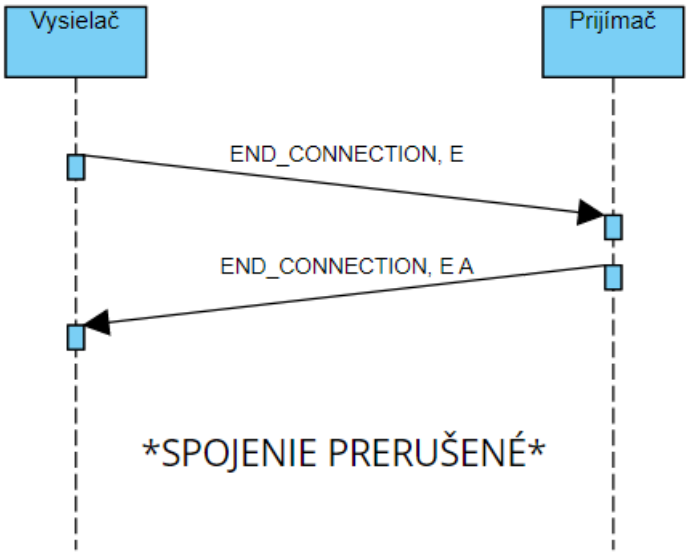
Pri posielaní dát najskôr vysielač oznámi, že chce začať nový dátový tok. To spraví tak, že pošle prijímateľovi packet s potrebným typom (posielanie textu – 2, posielanie súboru – 7), a flagom N (New data stream). Prijímateľ odošle späť packet s flagmi N a A. Ak vysielač úspešne prijme tento packet, prenos sa začne. Ak je veľkosť dát väčšia ako predvolená veľkosť fragmentu, dáta sa rozdelia do fragmentov a posielajú sa postupne. Fragment s dátami nemá žiadny flag. Obsahuje len typ, fragment dát, svoje poradie a celkový počet fragmentov daného toku dát. Odosielanie bude fungovať systémom Stop & Wait. Vysielač odošle fragment a čaká až prijímateľ pošle späť potvrdenie o prijatí fragmentu. Ak prijímateľ nepošle potvrdenie do nejakého časového intervalu, vysielač pošle fragment znova. Ak fragment dorazí poškodený, t.j. hodnota checksum nebude sedieť, prijímateľ pošle vysielaču požiadavku na znovu-poslanie daného fragmentu. To spraví tak, že pošle packet s flagom R (Request again) a nastaveným poradovým číslom fragmentu. Vysielač následne pošle fragment znova. Ak však ku chybe nedôjde, vysielač pošle packet s flagom A. Toto sa opakuje kým sa nepošlú všetky fragmenty. Ak sa posiela súbor, pošlú sa dodatočné fragmenty, ktoré indikujú typ a meno súboru. Po dokončení odosielania vysielač odošle posledný packet typu 0 s flagom C (Data stream complete). Prijímateľ v zápätí odošle packet toho istého typu s flagmi C a A. Prenos sa ukončí a spojenie sa udržiava.





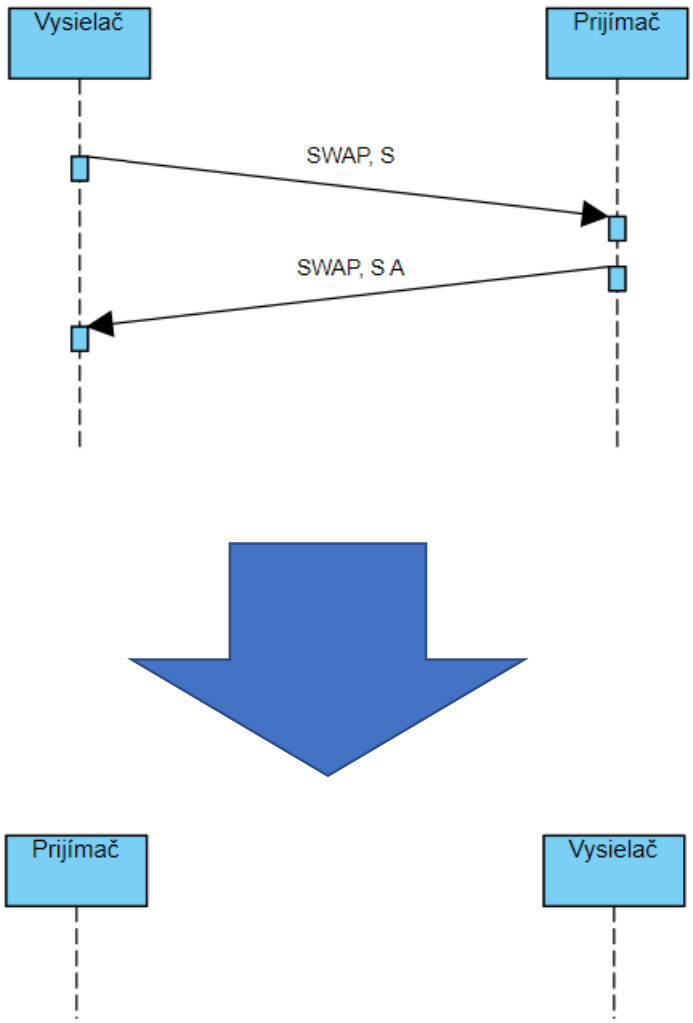
Ukončenie spojenia

Ukončiť spojenie môže hociktorý uzol. Avšak spojenie je možné ukončiť iba vtedy, keď sa neodosielajú dáta. Ukončenie sa inicializuje tým, že jeden uzol pošle druhému packet typu 0 s flagom E (End connection). Po prijatí prijímajúci uzol odošle späť packet rovnakého typu s flagmi E a A. Potom sa spojenie ukončí a packety KEEP ALIVE sa už nebudú posielať.



Prepínanie typu uzlov

Pri prvom spustení programu sa typ nastaví manuálne. Prepínanie medzi prijímačom a vysielačom prebehne poslaním packetu typu 0 s flagom S (Swap mode). Tento packet môže posielať aj prijímač aj vysielač. Po prijatí, prijímajúci uzol odošle packet rovnakého typu s flagmi S a A a typy sa prehodia.



Fragmentácia

Vysielač bude mať možnosť na začiatku zvoliť veľkosť fragmentu. Maximálnu veľkosť by som chcel nastaviť na 1400B aby sa program mohol rozširovať keď to bude nutné (napr. ak bude treba doplniť informácie do hlavičky) a teda aby sa nepresiahla veľkosť 1500B a tým nenastala fragmentácia na linkovej vrstve. Ak odosielané dáta majú menšiu veľkosť ako veľkosť fragmentu, tak sa vytvorí iba jeden fragment. Ak však veľkosť presahujú, tak sa dáta rozdelia podľa predvolenej dĺžky fragmentu, na každom packete fragmentu sa nastaví poradové číslo, totálny počet fragmentov a posielajú sa postupne. Po dokončení toku dát sa prijaté dáta poskladajú do celku podľa poradia FRAG_NUM.

Kontrolovanie chýb CHCECKSUM

Ako už bolo spomenuté, kontrolovanie chýb bude implementované pomocou funkcie crc32() z Python modulu „zlib“. Táto funkcia berie ako vstup pole byte-ov. V tomto prípade to budú dáta fragmentu. Výstup funkcie je 32 bitový integer, preto treba v hlavičke pre CHECKSUM vyhradiť 4B.

CRC32 je populárny algoritmus používaný na detekciu poškodenia údajov. Existuje viacero variant algoritmu, ktoré majú podobné matematické vlastnosti. Najbežnejší variant kontrolného súčtu CRC32, niekedy nazývaný CRC-32b, je založený na nasledujúcom polynóme:

$$g(x) = x32 + x26 + x23 + x22 + x16 + x12 + x11 + x10 + x8 + x7 + x5 + x4 + x2 + x + 1.$$

Príklad pre reťazec „abc“

Vstupy:
Binárny tvar 'abc': 0b011000010110001001100011 = 0x616263
Polynóm: 0b10000010011000001000111011011011 1 = 0x104C11DB7

Začneme s prvými troma byte-mi 'abc':
61 62 63 (hexadecimálne)
01100001 01100010 01100011 (binárne)

Prevrátime postupnosť bitov v každom byte:
10000110 01000110 11000110

Pridáme 32 bitov s hodnotou 0:
10000110010001101100011000000000000000000000000000000000

Aplikujeme XOR na prvé 4 byte-y s hodnotou 0xFFFFFFFF:
01111001101110010011100111111111000000000000000000000000

Potom aplikujeme operáciu tzv. 'CRC Delenie':

'CRC Delenie - algoritmus':
Ak je prvý bit 1, aplikujeme XOR s polynómom.
Ak je prvý bit 0, nerobíme nič.
Posunieme sa doprava o 1 bit
Opakujeme až kým neprídeme na koniec.

'CRC Delenie - príklad':
Vydelíme polynómom 0x104C11DB7:
01111001101110010011100111111111000000000000000000000000
10000010011000001000111011011 0111

 11100010001001011111101001001 0110
 10000010011000001000111011011 0111

 11000000100010101110100100 1000010
 10000010011000001000111011011 0111

 10000101110101001100111111111 01010
 10000010011000001000111011011 0111

 11110110100010000010010111010 0000
 10000010011000001000111011011 0111

 11101001110100010101011000010 1110
 10000010011000001000111011011 0111

 11010111011000111011000110 0110010
 10000010011000001000111011011 0111

 10101010000001100111111010000 1010
 10000010011000001000111011011 0111

 10100001100110111110000010111 10100
 10000010011000001000111011011 0111

 10000111111101101001111101000 01100
 10000010011000001000111011011 0111

 11011000110110110000010111011 0000
 10000010011000001000111011011 0111

 10110101011101110001011000000 1110
 10000010011000001000111011011 0111

 11011100010111100110001101110 0100
 10000010011000001000111011011 0111

 10111100011111011101101101010011

Zostane nám 32 – bitový zvyšok:
0b10111100011111011101101101010011 = 0xBC7DDB53

Na zvyšok aplikujeme XOR s 0xFFFFFFFF:
0b01000011100000100010010010101100 = 0x438224AC

Prevrátime bity:
0b00110101001001000100000111000010 = 0x352441C2 = 891568578

CRC-32 výstup pre 'abc' je: 0x352441C2 v hexadecimálnom tvare a 891568578 v destiatkovom tvare.

Zdroj príkladu: https://www.autohotkey.com/boards/viewtopic.php?f=74&t=35671