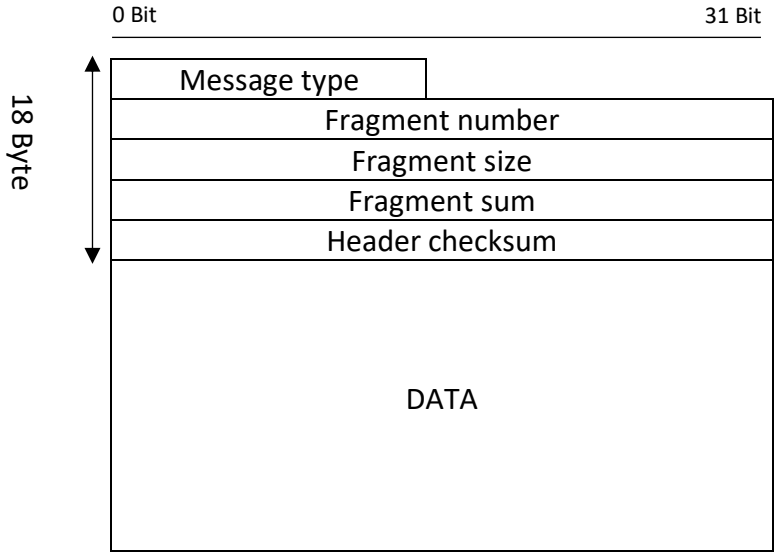
Počítačové a komunikačné siete
Zadanie 2 – Návrh riešenia

Meno: Martin Rudolf
Cvičenie : Štvrtok 8.00

Protokol

Náš protokol ktorý budeme používať bude mať vlastnú hlavičku, zostavenú z týchto častí:

1. Typ správy (bude rozlišovať o aký typ ide, či txt, pdf, png, a iné)
2. Číslo fragmentu nám opisuje poradové číslo fragmentu, slúži na znovu zostavenie správy
3. Veľkosť fragmentu
4. Fragment sum – celkový počet fragmentov
5. Header checksum – kontrolný súčet

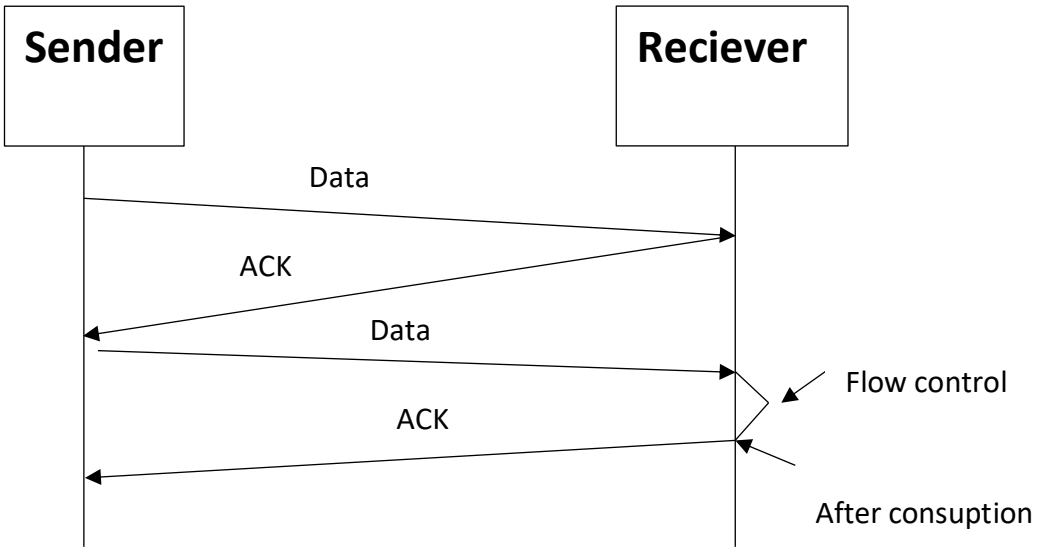



```
function crc(bit array bitString[1..len], int polynomial) {
    shiftRegister := initial value // commonly all 0 bits or all 1 bits
    for i from 1 to len {
        if most significant bit of shiftRegister xor bitString[i] = 1 {
            shiftRegister := (shiftRegister left shift 1) xor polynomial
        } else {
            shiftRegister := (shiftRegister left shift 1)
        }
    }
    return shiftRegister
}
```

Figure 1 Checksum pseudo kod

ARQ metóda

Program bude využívať ARQ, Stop and Wait metódu. Odosielateľ správy odošle paket a čaká na odpoveď (ACK), až tak pošle ďalší paket. Prijímateľ po prijatí paketu odosiela správu o prijatí acknowledgement (ACK). Pri komunikácií môže dôjsť k niekoľkým problémom, k strate dát, strate ACK správy, oneskoreniu dát a ACK. O tieto problémy sa postará naša Stop and Wait metóda ktorá funguje správne na lokálnej sieti, no to sa
nedá povedať pri komunikácii na väčšie vzdialenosti.



Udržanie spojenia

Pre udržanie spojenia použijeme keepalive metódu. Metóda zasiela keepalive správy vo vopred definovanom intervale, na ktoré čaká odpoveď. Ak odosielateľ nedostane odpoveď od prijímateľa, metóda spustí počítadlo ktoré začne počítať odoslané keepalive pakety bez odpovede, ak ich napočíta 3 tak ukončí komunikáciu.

Navrhnuté knižnice

<stdio.h>
<stdlib.h>
<pthread.h> vztvaranie a praca s vláknami
<arpa/inet.h> použijeme na prácu IP adresami
<sys/socket.h> použijeme na spojenie – praca so socketom

Dokumentácia

Užívateľské rozhranie:

```
Vyber funkciu:
1 -Server
2 -Klient
0 -Koniec aplikacie
Zadajte cislo vasej volby:
```

Užívateľ si má možnosť vybrať či sa pripojí ako server alebo ako klient. Ak si zvolí server zadá port na ktorom bude prijímať správy. Ak si zvolí klienta musí nakonfigurovať IP adresu a port na ktorý bude vysielať.

Implementácia

Projekt sa drží návrhu, posielanie a prijímanie správ je spravované ARQ Stop and Wait metódou , keep alive metódu sa z časových tiesni nepodarilo implementovať. Projekt bol implementovaný v jazyku C++ v prostredí visual studio 2019. Rozdiel v návrhu a implementácii je v CRC metóde, kde program využíva CRC16 ako štandardné UDP.

Knižnice a funkcie

Knižnice:

```
#include <stdlib.h>
#include <string>
#include <WinSock2.h>
#include <Ws2tcpip.h>
#include <thread>
#include <iostream>
#include <stdint.h>
#include <thread>
```

V projekte boli vzužité tieto funkcie:
void client(int port) plní funkciu klienta, na základe používateľových požiadaviek.
void server() plní funkciu servera.
void getCRC(const unsigned char* data_p, unsigned char length) vypočíta kontrolnú sumu.

Hlavička protokolu:

```
struct custom_header {
    uint16_t massage_type;
    uint32_t fragment_number;
    uint32_t fragment_size;
    uint32_t fragment_sum;
    uint32_t header_checksum;
};
```