Počítačové a komunikačné siete

Zadanie č.2 - Komunikácia s využitím UDP protokolu

Autor: Marek Čederle
AIS ID: 121193 (xcederlem)
Cvičiaci: Ing. Matej Janeba
(Utorok 16:00)

Návrh

Implementáciu budem robiť v programovacom jazyku python. Je to z dôvodu že má je jednoduchší ako C/C++ s tým že má knižnice na veľa vecí a celkovo sa s ním jednoduchšie pracuje. Síce jeho nevýhodou je že je pomalší ale to v našom prípade nebude tak vadiť. Pôjde o console program, to znamená že výstupy aj vstupy budú robené cez konzolu. Bude to v podstate také TUI (Terminal User Interface). Keďže pracujem na počítači s Windows tak aj následné testovanie bude prebiehať na Windows PC prípadne na localhoste. Môj protokol sa nazýva DNP.

Vlastná hlavička

Horná hlavička reprezentuje Ethernet hlavičku s vnorenými IPv4 a UDP protokolmi a PAYLOADOM, v ktorom sa nachádza môj protokol respektíve moja hlavička a následne dáta.

Preamble	Delimiter	Ethernet	IPv4	UDP	PAYLOAD	Ethernet CRC	IPG
7	1	14	20	8	18-1472	4	12

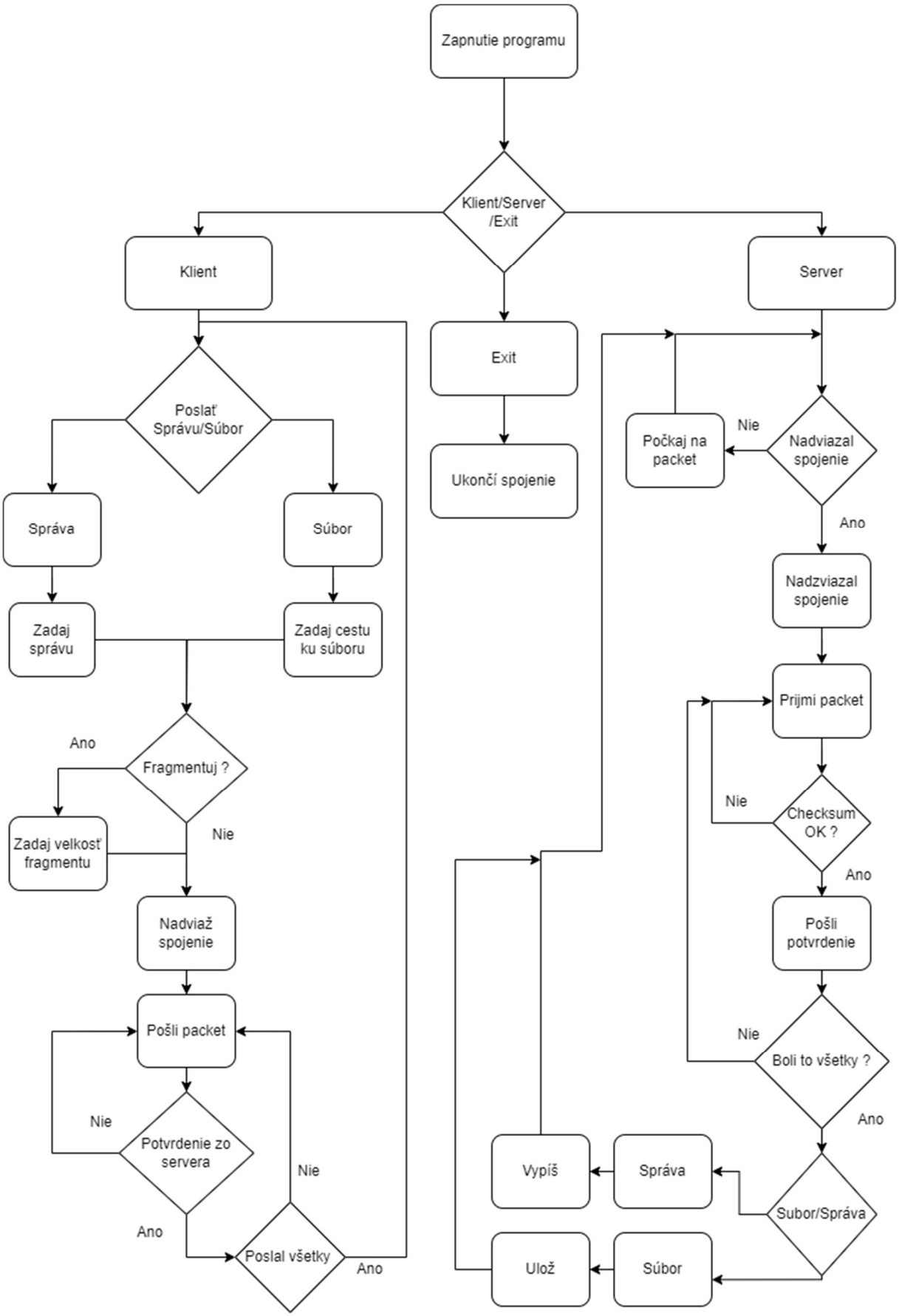
PAYLOAD								
DNP HEADER								
MSG TYPE	PACKET LENGHT	FRAG COUNT	FRAG OFFSET	CHECKSUM	DATA	without header	Ethernet minimal	60
1	2	2	2	1	10-1464	theoretical minimum	Ethernet max	1526
					24-1464	defined minimum		

- MSG TYPE – tip riadiacej správy (1B)
- PACKET LENGHT – veľkosť fragmentu (2B)
- FRAG COUNT – počet fragmentov (2B)
- FRAG OFFSET – offset kde majú pokračovať dáta (2B)
- CHECKSUM – kontrolný súčet pre dátovú časť (nazývam to ako checksum ale v skutočnosti ide o paritu) (1B)
- DATA – dáta vo vnútri packetu

Typy správ v MSG TYPE:

Decimal	Binary	Info
1	00000001	Keep alive
2	00000010	Nadviazanie spojenia
3	00000011	Ukončenie spojenia
4	00000100	Posielanie dát - správa
5	00000101	Posielanie dát - súbor
6	000000110	Chyba pri doručení dát
7	000000111	Doručenie dát bolo OK

Diagram a opis



Používateľ si vyberie či chce byť klient alebo server. Klient je odosielateľ dát a server je prijímateľ dát.

Klient:

Klient musí zadať IP adresu a port servera kam má posielať dáta. Takto sa nadviaže spojenie a potom zadá či chce posielať správu alebo súbor. Pri súbore treba zadať jeho cestu, pri správe treba zadať text. Potom zadá fragmentovanie prípadne počet fragmentov. Potom sa stále posiela keep alive a zároveň sa začnú posielať packety. Pri posielaní čaká na potvrdenie a ak mu server pošle potvrdenie tak posiela ďalej.

Server:

Server si nastaví IP adresu a port na ktorom bude počúvať. Čaká na inicializáciu spojenia. Ak sa klient snaží nadviazať tak server mu pošle správu o prijatí. Server pri prijímaní fragmentov vždy kontroluje každý jeden podľa checksum. Ak nastane chyba vypýta si znova. Keď sa pošlú všetky tak ak to bol súbor tak sa uloží ak správa tak ju vypíše. Potom zase počúva.

ARQ metóda

Budem používať STOP & wait ARQ metódu. Bude fungovať nasledovne:
Jej fungovanie v podstate vyplýva z jej názvu. Klient pošle dáta a ak server nepošle potvrdzovaciu správu že prišli dáta OK tak klient počká 8 sekúnd či náhodou nepríde potvrdzovacia správa. V prípade že príde tak pošle znova posledný packet.

Checksum (Parita)

Parita bude vypočítaná z dátovej časti a to nasledovne:
Keďže má 1B a to je 8 bitov tak sa dátová časť rozdelí na 8 častí a z každej sa vypočíta parita. Tá sa počíta spôsobom že ak v danej časti bol párny počet jednotiek tak dá nulu a ak nepárny počet jednotiek tak dá jednotku. Toto sa urobí pre každú časť a výsledných 8 bitov sa zapíše do hlavičky. Následne server po prijatí správy vypočíta podľa rovnakého postupu paritu a porovná s hlavičkou. Ak sa rovnajú tak prijme packet, ak nie tak ho zahodí a vypýta si ho znova.

Keep alive metóda

Metóda na udržanie spojenia funguje nasledovne:
Každých 12 sekúnd sa bude posielať správa na udržanie spojenia klientom. Server na ňu musí odpovedať. Ak neodpovie tak sa spojenie ukončí.

Zmeny oproti návrhu

Opravená veľkosť Ethernet fragmentu 60 -> 64

Preamble	Delimiter	Ethernet	IPv4	UDP	PAYLOAD	Ethernet CRC	IPG
7	1	14	20	8	18-1472	4	12

PAYLOAD								
DNP HEADER								
MSG TYPE	PACKET LENGHT	FRAG COUNT	FRAG NUMBER	CHECKSUM	DATA	without header	Ethernet minimal	64
1	2	2	2	1	10-1464	theoretical minimum	Ethernet max	1526

Zmena v hlavičke protokolu:

FRAG OFFSET zmenení na FRAG NUMBER

- MSG TYPE – tip riadiacej správy (1B)
- PACKET LENGHT – veľkosť fragmentu (2B)
- FRAG COUNT – počet fragmentov (2B)
- FRAG NUMBER – poradové číslo fragmentu (2B)
- CHECKSUM – kontrolný súčet pre dátovú časť (nazývam to ako checksum ale v skutočnosti ide o paritu) (1B)
- DATA – dáta vo vnútri packetu

Pridanie dvoch signalizačných správ:

Decimal	Info
1	Keep alive
2	Nadviazanie spojenia
3	Ukončenie spojenia
4	Posielanie dát - správa
5	Posielanie dát - súbor
6	Chyba pri doručení dát
7	Doručenie dát bolo OK
8	Výmena rolí
9	Súbor väčší ako 2MB

Keep alive interval bol zmenení na 5 sekund.

Fungovanie programu

Použité knižnice

- socket – knižnica ktorá pracuje so socketmi aby som vedel posielať dáta po sieti
- sys – využil som funkcie na zatvorenie programu
- threading – vytvorenie keep alive vlákna
- os – na prácu s cestami a názvami súborov
- struct – zostavenie hlavičky + dát a následné rozbalenie
- time – uspanie vlákna
- math – matematické funkcie (zaokrúhľovanie)

Opis funkcií programu

checksum(string_message)

Výpočet parity zo správy, ktorá je string. Vráti 1B integer.

checksum_binrary(binary_message)

Výpočet parity z binárnej správy/súboru. Vráti 1B integer.

switch_roles(client_or_server, socket_for_switch, ip_address, port)

Funkcia, ktorá sa zavolá ak prebieha zmena rolí. Funguje tak, že ako parameter sa pošle či sa mení client na server alebo naopak a potom sa rozhodne či chce naozaj zmeniť rolu. Ak nie nič sa nestane. Ak chce zmeniť rolu tak pošle správu na zmenu rolí, opačná strana musí zmenu potvrdiť. Ak nepotvrdí, pošle správu znova. Ak potvrdí tak sa zavrie daný socket na oboch stranách a vytvoria sa nové sockety s vymenenými rolami, s tým že sa vymenili aj porty.

server_choice()

Menu funkcia pre server.

Server(PORT_switch)

Funkcia, kde beží server. Bude vysvetlené ďalej.

client_choice()

Menu funkcia pre klienta.

Client(IP_switch, PORT_switch)

Funkcia, kde beží klient. Bude vysvetlené ďalej.

keep_alive(socket_client, server_ip)

Funkcia na udržanie spojenia. Je vytvorená na zvlášť vlákne a pošle KEEP ALIVE správu a čaká na odpoveď ktorá musí byť KEEP ALIVE správa. Ešte počúva aj na správu na zmenu rolí ak ju iniciuje server. (Zmena rolí zo strany servera bude vysvetlená ďalej)

send_message(socket_client, server_ip)

Funkcia na posielanie správ (iba klient). Má v sebe aj ARQ metódu, teda čaká na potvrdenie fragmentu.

send_file(socket_client, server_ip)

Funkcia na posielanie súborov (iba klient). Má v sebe aj ARQ metódu, teda čaká na potvrdenie fragmentu.

receive(socket_server)

Funkcia na prijímanie správ (iba server). Má v sebe aj ARQ metódu, teda ak sa pošle poškodený packet tak pošle správu o chybnom doručení a tým si vypýta novú.

main()

Podľa výberu z menu spustí Server/Klient alebo ukončí program.

Opis fungovania programu

Opis fungovania je podobný tomu čo je v diagrame.

Na začiatku programu v menu si používateľ vyberie či chce byť server alebo klient.

Server:

Pri spustení serveru sa inicializuje server socket. Následne vypíše svoju IP adresu. Ak to bolo prvé spustenie (nemenili si role) tak si vypýta aby používateľ zadal port na ktorom bude počúvať. Následne binduje socket a počúva na nadviazanie spojenia. Ak klient pošle správu na nadviazanie spojenia server mu ju opätuje. Následne vybehne menu kde si môže vybrať či chce zmeniť rolu alebo pokračovať. Potom sa zavolá funkcia receive, ktorá počúva všetky ostatné správy. Server vie zmeniť rolu tak že keď si z menu vyberie že chce meniť rolu, tak čaká na najbližšiu keep alive správu a na ňu pošle správu na zmenu rolí. A tak sa zmenia role na oboch uzloch. Ak nechce meniť role tak na keep alive správu posiela odpoveď keep alive. Ak klient chce zmeniť role a server zachytí správu na zmenu rolí tak pošle mu odpoveď že OK meníme role a oba zmenia rolu. Ak príde správa na ukončenie spojenia tak server pošle odpoveď na ukončenie a oba uzli ukončia spojenie a program. Ak to nie sú iba signalizačné správy samé o sebe ale posielajú sa väčšie správy aj s hlavičkou a dátami tak server zistí či sa posiela správa alebo súbor a podľa toho adekvátne robí checksum (paritu) a overuje správnosť packetu. Pri chybnom checksume si vypýta packet znovu. Všetky fragmenty si uloží a následne poskladá správu/súbor. Ak bol súbor menší ako 2MB tak si zadá aj názov súboru aj cestu kde chce uložiť. Ak súbor bol väčší ako 2MB tak klient ešte pred tým než začne posielať fragemnty tak pošle signalizačnú správu že súbor bude väčší ako 2MB a server mu na ňu rovnako odpovie. V správe je poslané meno súboru, ktoré si server uloží aby potom vedel ako sa má volať súbor. Ak zistí že sa poslali všetky fragmenty tak vypíše správu o úspešnom doručení a štatistiku koľko bolo prijatých packetov celkovo a koľko bolo chybných. Na konci vypíše aj absolútnu cestu ku súboru a veľkosť.

Klient:

Pri spustení klienta sa inicializuje klient socket. Vypíše jeho IP adresu a následne ak sa nemenili role tak používateľ zadá IP adresu a port servera. Potom pošle správu na nadviazanie spojenia a čaká na odpoveď od servera. Ak sa spojenie nadviaže tak klient spustí keep alive vlákno. Následne dostane menu a vyberá si či chce posielať správu/súbor prípadne zmeniť role alebo ukončiť komunikáciu. Vlákno na keep alive vie zachytiť prípadnú správu na zmenu role od servera a nastaví flag. Ak klient zistí že je flag nastavený tak zmení rolu. Pri posielaní správy zadá používateľ nejakú správu a zvolí si veľkosť fragmentu. Ak zadal zlú veľkosť tak ho nepustí ďalej. Následne sa spýta či chce pridať error do prvého packetu. Potom poskladá hlavičku a dáta a začne posielať fragmenty. Pri potvrdení zo servera posiela ďalej, ak príde že doručenie nebolo OK tak pošle fragment znovu. Posielanie súborov funguje na rovnakom princípe s rozdielom že sa zadáva aj názov súboru a cesta k nemu. Ak sa posiela súbor väčší ako 2MB tak sa pred posielaním fragmentov pošle ešte špeciálna správa s názvom súboru.

Poznámka: Ak sa pri zadávaní veľkosti fragmentu zistí že by nebolo možné poslať súbor lebo by nestačil počet fragmentov tak automaticky nastaví veľkosť fragmentu na maximum a oboznámi používateľa.

Ak klient chce meniť role z menu tak sa zavolá funkcia na zmenu rolí a tá vykoná potrebné kroky.

Ak klient zadá z menu že chce ukončiť program tak sa pošle ukončenie spojenia serveru a oba uzly sa vypnú.

ARQ metóda

Používam metódu Stop and Wait. Je súčasťou funkcií na prijímanie a posielanie správ a bola opísaná vyššie.

Checksum (Parita)

Používam rovnaký princíp ako v návrhu.

Vnesenie chyby

Urobí zmenu v checksume ak si o to klient požiada. Zmena sa dotkne iba prvého fragmentu.

Používateľské rozhranie

Server:

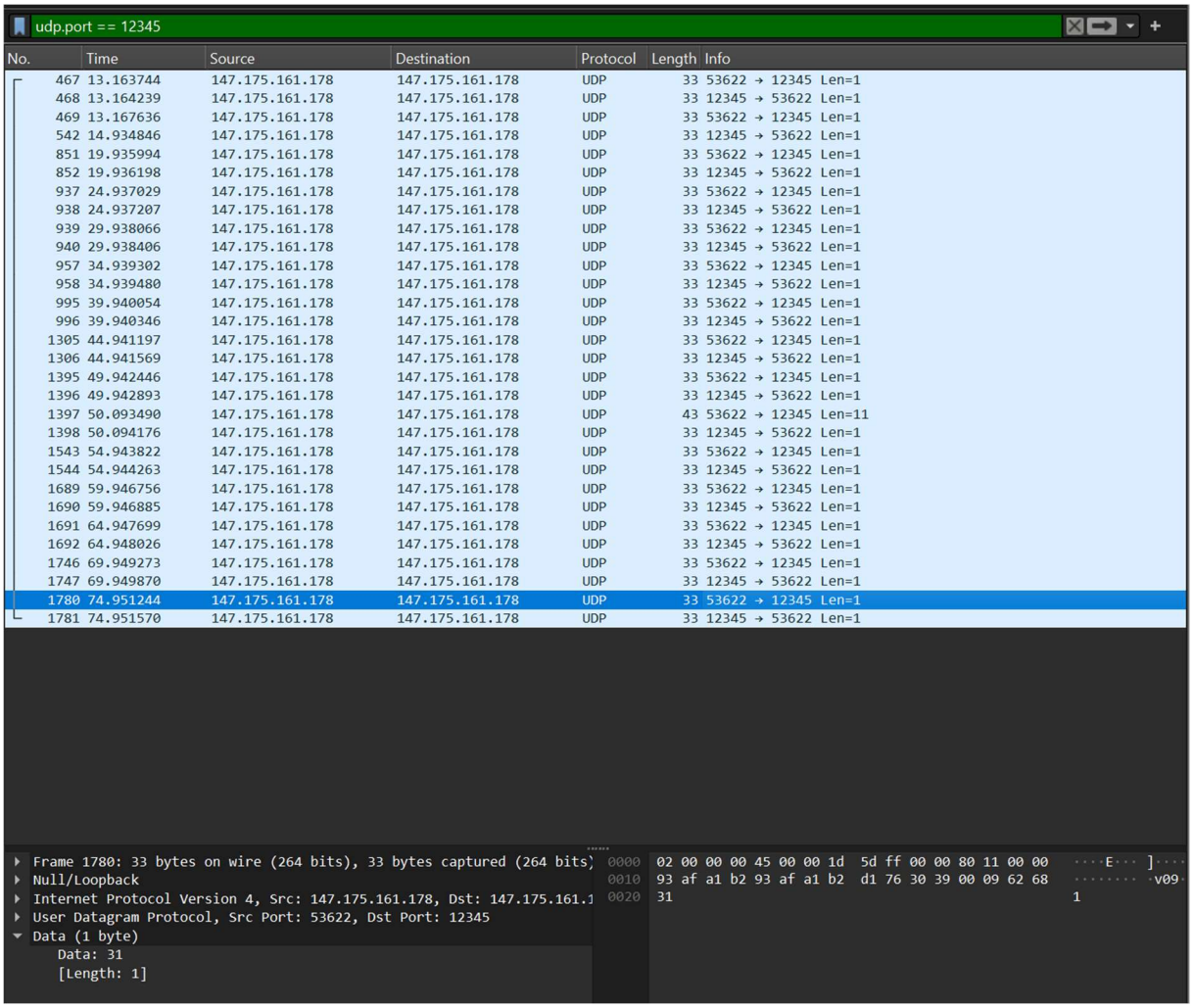
```
Starting program...
Enter your option:
1 - Server
2 - Client
3 - Quit
1
-----You are SERVER-----
Server IP:  147.175.161.178
Enter port: 12345
Connection established with:  ('147.175.161.178', 53622)
Enter your option:
1 - Switch roles
2 - Quit
Press whatever to continue

Nothing changed, continuing...
Fragment 0 received successfully
Message received successfully
Received 1 fragments (damaged included)
Damaged was 0 fragments
Received message:  SUP
Message length: 3
Enter your option:
1 - Switch roles
2 - Quit
Press whatever to continue
```

Klient:

```
Starting program...
Enter your option:
1 - Server
2 - Client
3 - Quit
2
-----You are CLIENT-----
Client IP:  147.175.161.178
Enter port: 12345
Enter server IP: 147.175.161.178
Connection established with:  ('147.175.161.178', 12345)
Enter your option:
1 - Send message
2 - Send file
3 - Switch roles
4 - Quit
1
Sending message...
Enter your message: SUP
Enter fragment size (10-1464): 10
Number of fragments:  1
Do you want to introduce an error? (y/n): n
Fragment 0 delivered successfully
Enter your option:
1 - Send message
2 - Send file
3 - Switch roles
4 - Quit
```

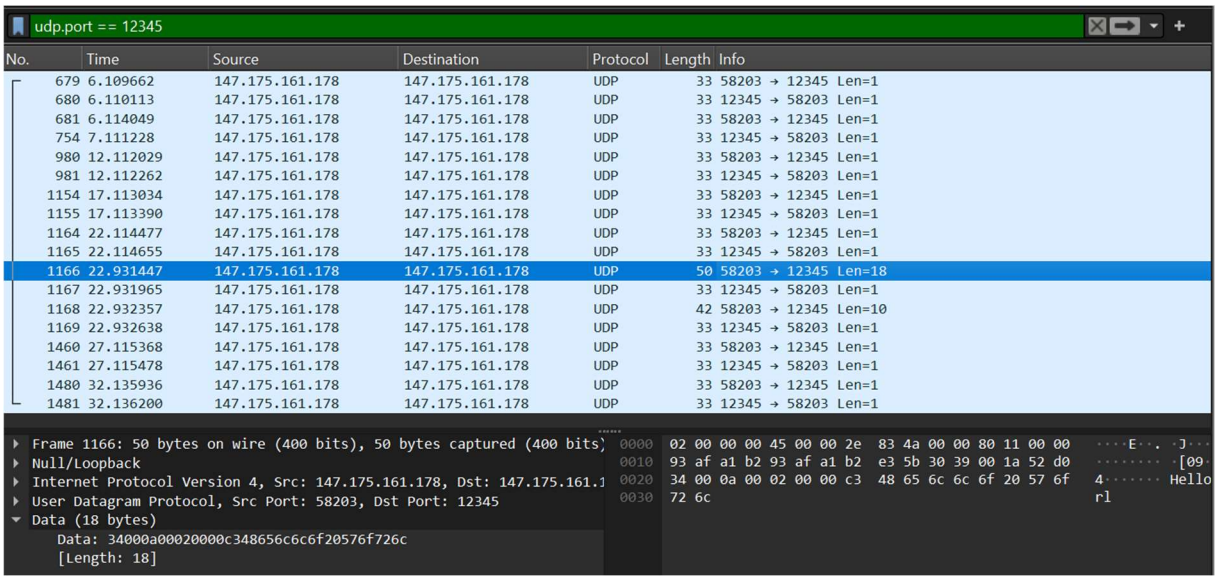
Zachytávanie komunikácie cez Wireshark

Keep alive:

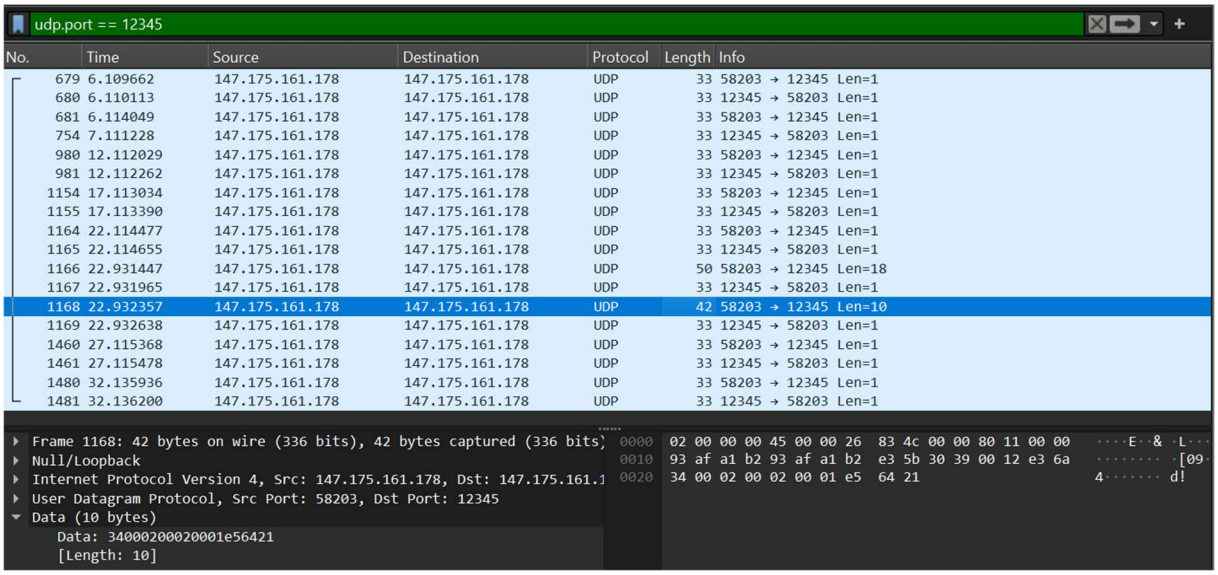


Posielanie správy:

1.fragment



2. fragment



Záver

Medzi kľúčové funkcie programu patrí prepínanie rolí, overovanie checksumu na kontrolu údajov a implementácia metódy Stop and Wait - Automatic Repeat ReQuest (ARQ).

V prípade servera si používateľ môže vybrať možnosti prostredníctvom menu keď server inicializuje socket. Spracúva zmeny rolí a overuje checksum prijatých správ alebo súborov.

Na strane klienta používateľ začne komunikáciu, odosiela správy alebo súbory a môže prepnúť rolu. Klient umožňuje používateľom zaviesť chybu v checksum. Metóda Stop and Wait ARQ zabezpečuje spoľahlivý prenos údajov medzi klientom a serverom.

Program teda prenáša dáta pomocou UDP protokolu a navyše má kontrolné mechanizmy.

Testoval som ako na svojom počítači (localhost) tak aj s kolegom na dvoch počítačoch. Testované boli relatívne krátke správy ale funkčnosť veľkých správ je zaručená tiež. Dokázal som poslať aj súbor väčší ako 2MB. Išlo o obrázok vo vysokej kvalite, ktorý bol tiež doručený korektne.

Zdroje

https://pythontic.com/modules/socket/recv

https://pythontic.com/modules/socket/sendto

https://docs.python.org/3/library/socket.html

https://docs.python.org/3/library/struct.html