

Dokumentacia

Počítačové a Komunikačné siete

Communication over UDP

Jakub Martinák

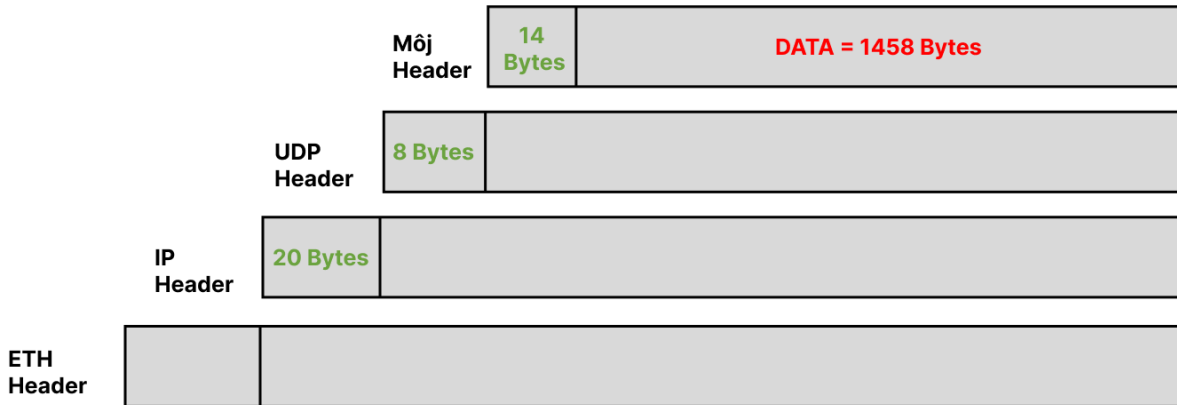
Faculty of Informatics and Information Technology, Slovak Technical
University

Obsah

1	Prvotný Návrh	2
1.1	Checksum metóda	3
1.2	ARQ metóda	3
1.3	Metóda pre udržanie spojenia	3
2	Konečné riešenie	4
3	Použité funkcie/Triedy	4
3.1	CustomHeader	4
3.2	send_packet	5
3.3	file_to_chunks	6
4	Hlavný chod programu	7
4.1	Klient	7
4.2	Server	8
5	Wireshark	9
5.1	Init Packet + poslanie súboru	9
5.2	Koniec posielania súboru	9
5.3	Posielanie Správy	10
5.4	Switch Packet	10
5.5	Switch Packet na mŕtvy server	11
6	Diagram	12

1 Prvotný Návrh

Navrhnutý protokol bude vnorený do protokolu UDP ako je znázornené na obrázku 1. Maximálna veľkosť packetu je 1500B(ETH) - 20B(IP Header) - 8B(UDP Header) - 14B(Môj Header) = **1458 B**



Obr. 1: Návrh packetu

Typ	Sekvencia	Identifikátor súboru	Checksum (CRC16)	Payload	Flags	Data
1B	4B	4B	2B	2B	1B	-

Tabuľka 1: Môj header

Časť	Význam	Možnosti
Typ	Rola packetu	0x01(data), 0x02(ACK), 0x03(Controla), atď
Sekvencia	Na kontrolu správneho poradia Packetov	-
Identifikátor súboru	Unikátny identifikátor súbor	Pomoc pri fragmentovaní
Checksum(CRC16)	Pre kontrolu errorov	-
Veľkosť Payloadu	Veľkosť payloadu v bytoch	Dĺžka packetu indikujúca koľko dát packet obsahuje
Vlajky	Kontrolne Informacie	Vlajka pre znovuoslanie, atď

Tabuľka 2: Typy

Komunikátor bude implementovaný v jazyku Python. Pri spustení programu bude mať užívateľ možnosť si vybrať či sa spustí klient alebo server. Pri výbere klienta sa bude musieť zadávať IP adresa servera a port, na ktorom počúva, veľkosť fragmentu a správu / súbor. Pri spustení programu ako server, užívateľ bude musieť zadať port, na ktorom má server počúvať a bude čakať na inicializačný packet od klienta. Keď klient zadá všetky potrebné informácie, vyšle sa inicializačný packet a počká na odpoveď ACK od serveru, a následne sa začnú posielat dáta. Ak je veľkosť packetu väčšia ako nastavená veľkosť na posielanie, súbor/správa sa rozdelí na fragmenty a bude sa posielat jednotlivo. Po každom odoslanom fragmente server overí pomocou checksumu packet, ak sedí pošle sa ACK a čaká na ďalší odoslaný packet.

Server bude fungovať na zvolenej metóde ARQ Stop and Wait, kde bude klient čakať s posielaním ďalšieho packetu až kým nedostane od serveru ACK o kompletnom prijatí packetu, ak by klient nedostal odpoveď do 6 sekúnd, packet sa pošle znovu. Po dokončení posielania, server čaká na ďalší signál od klienta.

1.1 Checksum metóda

Checksum je využívaný na overenie správy, či bola počas prenášania poškodená. V projekte bude použitý checksum CRC16(Cyclic Redundancy Check) z knižnice binascii (Python). Polynomičná reprezentácia CRC16:

$$x^{16} + x^{15} + x^2 + 1$$

Binárna reprezentácia:

11000000000000101

Hlavná operácia v CRC je binárne delenie, je to ale čisto založené na XOR operáciach, bez odčítania. Dáta, ktoré chceme skontrolovať sú považované za veľké binárne číslo a to je vydelené polynómom CRC.

Pri inicializácii 16 núl je pridelených na koniec dát pred začiatkom delenia. To sa deje kvôli tomu že dĺžka CRC checksumu je 16 bitov. A zvyšok po delení CRC16 je 16-bitové číslo.

1.2 ARQ metóda

V projekte bude použitá ARQ metóda **Stop and Wait**(Automatic Repeat reQuest) .

Táto metóda je jednoduchá ale účinná, zabezpečuje, že dáta prídu v poradí akom boli poslané. Funguje na princípe, že každý packet, ktorý bol odoslaný od klienta na server, klient čaka na spätné ACK aby mohol poslať ďalší. Ak klient do určitého časového limitu nedostane ACK pokúsi sa Packet poslať znovu, čo môže v nejakých prípadoch spôsobiť duplicitu. Táto metóda nie je efektívna na posielanie packetov o veľkej veľkosti alebo na nestabilných sieťach, kde kvôli čakaniu na ACK sa veľmi znižuje jeho efektívnosť.

1.3 Metóda pre udržanie spojenia

Spojenie bude udržiavané pomocou užívateľského rozhrania, kde si užívateľ bude môcť vybrať či chce zmeniť svoju rolu z klienta na server a vice versa. Po dokončení odosielania, bude toto menu opätovne zobrazené na klientovej strane. Ak po skončení odosielania nedostane server po dobu 15 sekúnd packet od klienta na vymenenie ról alebo opätovné posielanie tak sa spojenie terminuje.

2 Konečné riešenie

V konečnom riešení som zmenšil svoj header o 5B z pôvodných 14B sa zmenšil na 9B. Zmenil som svoj prístup k posielaniu cesty a názvu súboru, z posielania týchto informácií v headery na posielanie ich v data časti Init packetu. Taktiež som nepotreboval ďalší 1B na vlajky a ani 2B na veľkosť payloadu. Pridal som do headru 3B na konečný počet packetov, ktoré majú byť poslané.

Typ	Sekvencia	Počet fragmentov	Checksum (CRC16)	Data
1B	3B	3B	2B	-

Tabuľka 3: Konečný header

3 Použité funkcie/Triedy

3.1 CustomHeader

Táto trieda *CustomHeader* pomáha v serializovaní a deserializovaní headra ako taktiež v jeho inicializovaní.

```
9 usages  ▲ JakobMartinak
class CustomHeader:
    ▲ JakobMartinak
    def __init__(self, command, sequence_number, fragment_count, crc):
        self.command = command
        self.sequence_number = sequence_number
        self.fragment_count = fragment_count
        self.crc = crc
        # self.flags = flags

7 usages  ▲ JakobMartinak
    def serialize(self):
        # The sequence_number and file_path must be converted to bytes if they are not already
        # CRC is a numerical value, computed over the data
        # Flags is a numerical value, fitting within 1 byte
        return struct.pack('!B3s3sH', *vars(self), self.sequence_number, self.fragment_count, self.crc)

1 usage  ▲ JakobMartinak
    @staticmethod
    def deserialize(data):
        unpacked_data = struct.unpack('!B3s3sH', data)
        return CustomHeader(*unpacked_data)
```

Obr. 2: Kod Classy CustomHeader

3.2 send_packet

Send_packet funkcia, obsahuje celé riadenie posielanie packetov, vrátane čakania na odpovede kde je aplikovaná Stop&Wait ARQ metóda, čiže po každom odoslanom packete sa čaká na ACK/FINACK ak nepríde do 5 sekúnd pošle sa packet znovu, toto sa zopakuje ešte 2x, ak server dovedy neodpovie spojenie sa ukončí.

```
9 usages  ▴ JakobMartinak
def send_packet(sock, packet, address, timeout=5):
    try:
        sock.settimeout(timeout)
        sock.sendto(packet, address)

        # Wait for ACK
        response, IP_SERVER = sock.recvfrom(1024) # Buffer size for ACK
        if int.from_bytes(response, byteorder='big') == PACKET_TYPES.get("ACK"):
            return response
        elif int.from_bytes(response, byteorder='big') == PACKET_TYPES.get("ERROR"):
            return response
        elif int.from_bytes(response, byteorder='big') == PACKET_TYPES.get("CHANGE"):
            return response
        elif int.from_bytes(response, byteorder='big') == PACKET_TYPES.get("FINACK"):
            return response
    except socket.timeout:
        count = 0
        while count < 3:
            print("Timeout, resending packet...")
            sock.sendto(packet, address)
            try:
                ack, _ = sock.recvfrom(1024) # Buffer size for ACK
                return ack
            except socket.timeout:
                count += 1
        print("Server not responding, closing connection...")
        return None
```

Obr. 3: Kod send_packet

3.3 file_to_chunks

V tejto funkcii sa vykonáva logika čítania súboru do bajtov, a delí sa na fragmenty podľa zadanej veľkosti.

```
1 usage  ▴ JakobMartinak
def file_to_chunks(file_p, chunk_size):
    """
    Generator that reads a file in chunks of chunk_size bytes.
    :param file_p:
    :param chunk_size:
    :return: chunk of data
    """

    with open(file_p, 'rb') as file:
        chunk_count = 0
        while True:
            chunky = file.read(chunk_size)
            if not chunky:
                break
            chunk_count += 1
            yield chunky
```

Obr. 4: Kod file_to_chunk

4 Hlavný chod programu

Pri spustení programu bude užívateľ vyzvaný na vybratie roly, ktorú chce mať, server/klient.

4.1 Klient

Ak si užívateľ vyberie klienta, bude vyzvaný na zadanie IP servera ako taktiež jeho port. Po zadaní týchto základných informácií, užívateľ môže poslať správu alebo súbor na daný server. Ak chce môže simulovať zkoruptovaný packet, aby si ho server vypýtal znovu. Po odoslaní všetkých packetov či už správy alebo súboru, bude užívateľ 3 sekundy počúvať na možnú výmenu rolí zo strany serveru, ak nič nepríde užívateľ si môže znovu vybrať medzi poslaním súboru alebo správy a taktiež môže poslať packet na výmenu rolí. Ak sa rozhodne zmeniť role, bude vyzvaný na zadanie portu, na ktorom má server bežať.

```
repik@pop-os [14:08:53] [~/Documents/STU/PKS/Zadania/Zadanie 2] [main *]
-> % python3 commu.py
=====
Please choose if you want to be a client or server
After sending message/file, you will be asked to choose again
If you want to exit input 0 or press CTRL+C
Or wait for timeout
=====
Choose input: 1 - client, 2 - server, 3 - switch: 1
Enter server IP: 192.168.100.98
Enter server port: 8000
192.168.100.70
CLIENT IP : 192.168.100.70
Choose input: 1 - text, 2 - file, 3 - switch, 0 - exit: 2
Do you want to corrupt packets? (y/n): y
Enter packet data size (MAX 1460): 1460
Enter path to save the file: ./
Enter path to file with file name: big_file.txt
```

Obr. 5: Klientova strana

```
Choose input: 1 - text, 2 - file, 3 - switch, 0 - exit: 1
Do you want to corrupt packets? (y/n): n
Enter packet data size (MAX 1460): 1460
Enter the message: heello
Response: b'\x03'
Connection Innitiated
last packet, sending FIN
Waiting for switch packet
No switch packet received
Choose input: 1 - text, 2 - file, 3 - switch, 0 - exit: 2
Do you want to corrupt packets? (y/n): n
Enter packet data size (MAX 1460): 1460
Enter path to save the file: ./
Enter path to file with file name: big_file.txt
Connection Innitiated
last packet, sending FIN
File sent successfully = 716 packets
Waiting for switch packet
No switch packet received
```

Obr. 6: Klientova strana na ďalšie poslanie

4.2 Server

Ak si užívateľ vyberie rolu servera bude vyzvaný na zadanie portu, na ktorom chce počúvať. Po zadaní portu ihneď začne počúvať a zároveň sa začne timer na timeout, ak nedostane žiaden packet po dobu 15 sekúnd, zavrie port a ukončí sa počúvanie. Užívateľ má potom možnosť znovu si vybrať či si želá byť server alebo užívateľ. Ak server obdrží init packet čaká na zvyšok packetov, či už správy alebo súboru. Na konci keď príde posledný packet na server, je server vyzvaný či si želá zmeniť role alebo nie, ak nie ostane počúvať na rovnakom porte ako doteraz, ak sa rozhodne zmeniť rolu tak musí ako klient zadať port servera a môže začať posielať.

```
repik@pop-os [13:06:55] [~/Documents/STU/PKS/Zadania/Zadanie 2] [main]
-> % python3 commu.py
=====
Please choose if you want to be a client or server
After sending message/file, you will be asked to choose again
If you want to exit input 0 or press CTRL+C
Or wait for timeout
=====
Choose input: 1 - client, 2 - server, 3 - switch: 2
192.168.100.70
Enter server port: 8000
Starting up on 192.168.100.70 port 8000
```

Obr. 7: Štart servera

```
Waiting to receive message...
[+DEBUG+]Received 13 bytes from 192.168.100.98
Init packet received

Waiting to receive message...
[+DEBUG+]Received 19 bytes from 192.168.100.98
FIN packet received
Received header: Command=4, Sequence Number=0, Total Fragments=0, CRC=9c5c,
Received data: eeeee1111
Do you want to switch ? (y/n): n

Waiting to receive message...
```

Obr. 8: Vyzvanie servera

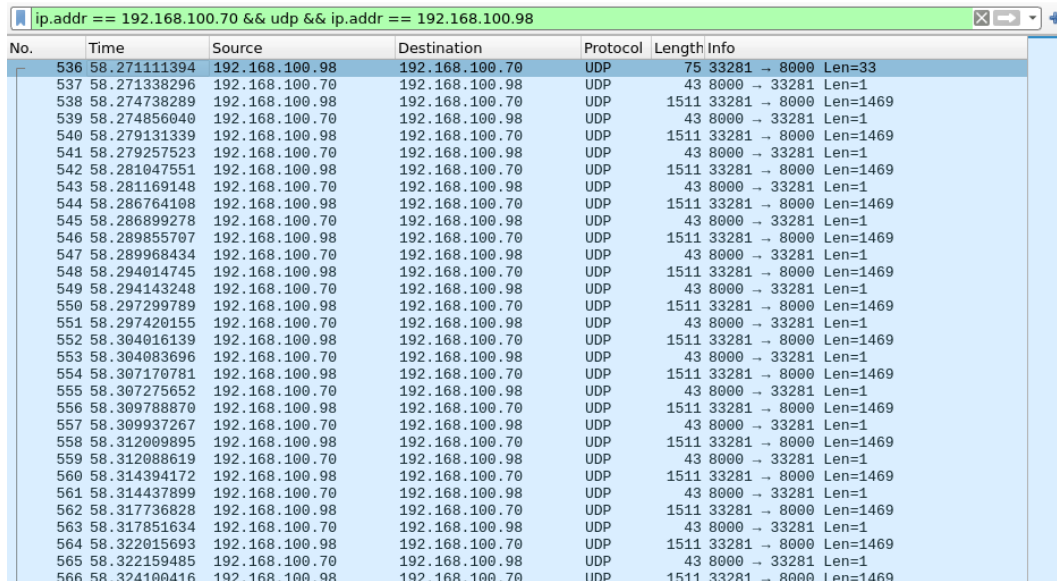
```
Received header: Command=4, Sequence Number=717, Total Fragments=716, CRC=3fe6,
Constructing File
File constructed
File is save at ./sent/big_file.txt
Do you want to switch ? (y/n): y
Switching... from server
Enter server port: 8000
Choose input: 1 - text, 2 - file, 3 - switch, 0 - exit: 1
```

Obr. 9: Switch servera

5 Wireshark

V tejto sekcii sú zobrazené sledovania z wiresharku, ako server s klientom komunikuje a vice versa.

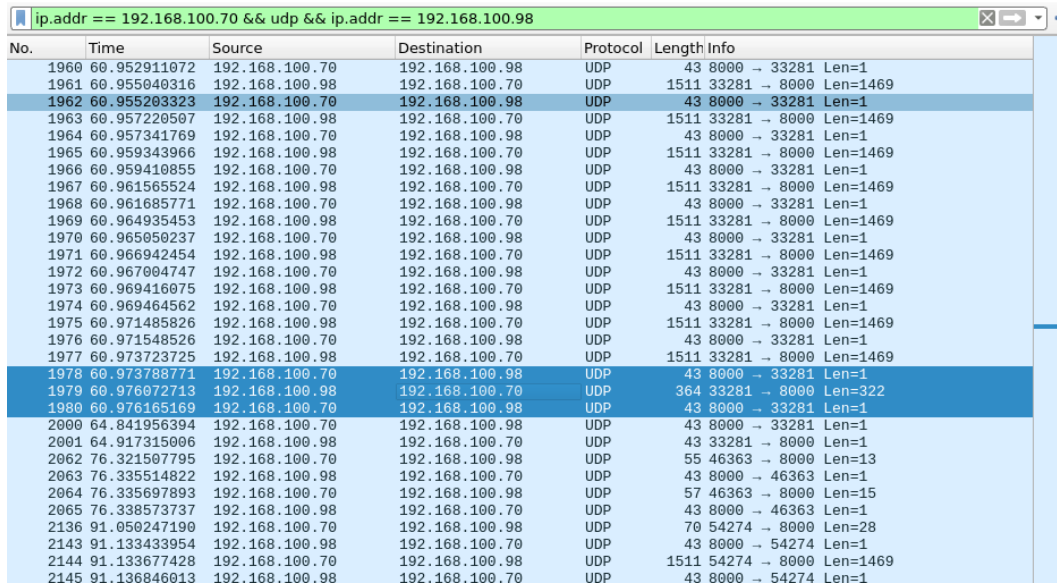
5.1 Init Packet + poslanie súboru



No.	Time	Source	Destination	Protocol	Length	Info
536	58.271111394	192.168.100.98	192.168.100.70	UDP	75	33281 → 8000 Len=33
537	58.271338296	192.168.100.70	192.168.100.98	UDP	43	8000 → 33281 Len=1
538	58.274738289	192.168.100.98	192.168.100.70	UDP	1511	33281 → 8000 Len=1469
539	58.274856040	192.168.100.70	192.168.100.98	UDP	43	8000 → 33281 Len=1
540	58.279131339	192.168.100.98	192.168.100.70	UDP	1511	33281 → 8000 Len=1469
541	58.279257523	192.168.100.70	192.168.100.98	UDP	43	8000 → 33281 Len=1
542	58.281047551	192.168.100.98	192.168.100.70	UDP	1511	33281 → 8000 Len=1469
543	58.281169148	192.168.100.70	192.168.100.98	UDP	43	8000 → 33281 Len=1
544	58.286764108	192.168.100.98	192.168.100.70	UDP	1511	33281 → 8000 Len=1469
545	58.286899278	192.168.100.70	192.168.100.98	UDP	43	8000 → 33281 Len=1
546	58.289855707	192.168.100.98	192.168.100.70	UDP	1511	33281 → 8000 Len=1469
547	58.289968434	192.168.100.70	192.168.100.98	UDP	43	8000 → 33281 Len=1
548	58.294014745	192.168.100.98	192.168.100.70	UDP	1511	33281 → 8000 Len=1469
549	58.294143248	192.168.100.70	192.168.100.98	UDP	43	8000 → 33281 Len=1
550	58.297299789	192.168.100.98	192.168.100.70	UDP	1511	33281 → 8000 Len=1469
551	58.297420155	192.168.100.70	192.168.100.98	UDP	43	8000 → 33281 Len=1
552	58.304016139	192.168.100.98	192.168.100.70	UDP	1511	33281 → 8000 Len=1469
553	58.304083696	192.168.100.70	192.168.100.98	UDP	43	8000 → 33281 Len=1
554	58.307170781	192.168.100.98	192.168.100.70	UDP	1511	33281 → 8000 Len=1469
555	58.307275652	192.168.100.70	192.168.100.98	UDP	43	8000 → 33281 Len=1
556	58.309788870	192.168.100.98	192.168.100.70	UDP	1511	33281 → 8000 Len=1469
557	58.309937267	192.168.100.70	192.168.100.98	UDP	43	8000 → 33281 Len=1
558	58.312009895	192.168.100.98	192.168.100.70	UDP	1511	33281 → 8000 Len=1469
559	58.312088619	192.168.100.70	192.168.100.98	UDP	43	8000 → 33281 Len=1
560	58.314394172	192.168.100.98	192.168.100.70	UDP	1511	33281 → 8000 Len=1469
561	58.314437899	192.168.100.70	192.168.100.98	UDP	43	8000 → 33281 Len=1
562	58.317736828	192.168.100.98	192.168.100.70	UDP	1511	33281 → 8000 Len=1469
563	58.317851634	192.168.100.70	192.168.100.98	UDP	43	8000 → 33281 Len=1
564	58.322015693	192.168.100.98	192.168.100.70	UDP	1511	33281 → 8000 Len=1469
565	58.322159485	192.168.100.70	192.168.100.98	UDP	43	8000 → 33281 Len=1
566	58.324100416	192.168.100.98	192.168.100.70	UDP	1511	33281 → 8000 Len=1469

Obr. 10: Prvý init packet a poslanie súboru

5.2 Koniec posielania súboru



No.	Time	Source	Destination	Protocol	Length	Info
1960	60.952911072	192.168.100.70	192.168.100.98	UDP	43	8000 → 33281 Len=1
1961	60.955040316	192.168.100.98	192.168.100.70	UDP	1511	33281 → 8000 Len=1469
1962	60.955203323	192.168.100.70	192.168.100.98	UDP	43	8000 → 33281 Len=1
1963	60.957220507	192.168.100.98	192.168.100.70	UDP	1511	33281 → 8000 Len=1469
1964	60.957341769	192.168.100.70	192.168.100.98	UDP	43	8000 → 33281 Len=1
1965	60.959343966	192.168.100.98	192.168.100.70	UDP	1511	33281 → 8000 Len=1469
1966	60.959410855	192.168.100.70	192.168.100.98	UDP	43	8000 → 33281 Len=1
1967	60.961565524	192.168.100.98	192.168.100.70	UDP	1511	33281 → 8000 Len=1469
1968	60.961685771	192.168.100.70	192.168.100.98	UDP	43	8000 → 33281 Len=1
1969	60.964935453	192.168.100.98	192.168.100.70	UDP	1511	33281 → 8000 Len=1469
1970	60.965050237	192.168.100.70	192.168.100.98	UDP	43	8000 → 33281 Len=1
1971	60.966942454	192.168.100.98	192.168.100.70	UDP	1511	33281 → 8000 Len=1469
1972	60.967004747	192.168.100.70	192.168.100.98	UDP	43	8000 → 33281 Len=1
1973	60.969416075	192.168.100.98	192.168.100.70	UDP	1511	33281 → 8000 Len=1469
1974	60.969464562	192.168.100.70	192.168.100.98	UDP	43	8000 → 33281 Len=1
1975	60.971485826	192.168.100.98	192.168.100.70	UDP	1511	33281 → 8000 Len=1469
1976	60.971548526	192.168.100.70	192.168.100.98	UDP	43	8000 → 33281 Len=1
1977	60.973723725	192.168.100.98	192.168.100.70	UDP	1511	33281 → 8000 Len=1469
1978	60.973788771	192.168.100.70	192.168.100.98	UDP	43	8000 → 33281 Len=1
1979	60.976072713	192.168.100.98	192.168.100.70	UDP	364	33281 → 8000 Len=322
1980	60.976165169	192.168.100.70	192.168.100.98	UDP	43	8000 → 33281 Len=1
2000	64.841956394	192.168.100.70	192.168.100.98	UDP	43	8000 → 33281 Len=1
2001	64.917315006	192.168.100.98	192.168.100.70	UDP	43	33281 → 8000 Len=1
2062	76.321507795	192.168.100.70	192.168.100.98	UDP	55	46363 → 8000 Len=13
2063	76.335514822	192.168.100.98	192.168.100.70	UDP	43	8000 → 46363 Len=1
2064	76.335697893	192.168.100.70	192.168.100.98	UDP	57	46363 → 8000 Len=15
2065	76.338573737	192.168.100.98	192.168.100.70	UDP	43	8000 → 46363 Len=1
2136	91.050247190	192.168.100.70	192.168.100.98	UDP	70	54274 → 8000 Len=28
2143	91.133433954	192.168.100.98	192.168.100.70	UDP	43	8000 → 54274 Len=1
2144	91.133677428	192.168.100.70	192.168.100.98	UDP	1511	54274 → 8000 Len=1469
2145	91.136846013	192.168.100.98	192.168.100.70	UDP	43	8000 → 54274 Len=1

Obr. 11: Posledný packet a FINACK

5.3 Posielanie Správy

ip.addr == 192.168.100.70 && udp && ip.addr == 192.168.100.98						
No.	Time	Source	Destination	Protocol	Length	Info
1960	60.952911072	192.168.100.70	192.168.100.98	UDP	43	8000 → 33281 Len=1
1961	60.955040316	192.168.100.98	192.168.100.70	UDP	1511	33281 → 8000 Len=1469
1962	60.955203323	192.168.100.70	192.168.100.98	UDP	43	8000 → 33281 Len=1
1963	60.957220507	192.168.100.98	192.168.100.70	UDP	1511	33281 → 8000 Len=1469
1964	60.957341769	192.168.100.70	192.168.100.98	UDP	43	8000 → 33281 Len=1
1965	60.959343966	192.168.100.98	192.168.100.70	UDP	1511	33281 → 8000 Len=1469
1966	60.959410855	192.168.100.70	192.168.100.98	UDP	43	8000 → 33281 Len=1
1967	60.961565524	192.168.100.98	192.168.100.70	UDP	1511	33281 → 8000 Len=1469
1968	60.961685771	192.168.100.70	192.168.100.98	UDP	43	8000 → 33281 Len=1
1969	60.964935453	192.168.100.98	192.168.100.70	UDP	1511	33281 → 8000 Len=1469
1970	60.965050237	192.168.100.70	192.168.100.98	UDP	43	8000 → 33281 Len=1
1971	60.966942454	192.168.100.98	192.168.100.70	UDP	1511	33281 → 8000 Len=1469
1972	60.967004747	192.168.100.70	192.168.100.98	UDP	43	8000 → 33281 Len=1
1973	60.969416075	192.168.100.98	192.168.100.70	UDP	1511	33281 → 8000 Len=1469
1974	60.969464562	192.168.100.70	192.168.100.98	UDP	43	8000 → 33281 Len=1
1975	60.971485826	192.168.100.98	192.168.100.70	UDP	1511	33281 → 8000 Len=1469
1976	60.971548526	192.168.100.70	192.168.100.98	UDP	43	8000 → 33281 Len=1
1977	60.973723725	192.168.100.98	192.168.100.70	UDP	1511	33281 → 8000 Len=1469
1978	60.973788771	192.168.100.70	192.168.100.98	UDP	43	8000 → 33281 Len=1
1979	60.976072713	192.168.100.98	192.168.100.70	UDP	364	33281 → 8000 Len=322
1980	60.976165169	192.168.100.70	192.168.100.98	UDP	43	8000 → 33281 Len=1
2000	64.841956394	192.168.100.70	192.168.100.98	UDP	43	8000 → 33281 Len=1
2001	64.917315006	192.168.100.98	192.168.100.70	UDP	43	33281 → 8000 Len=1
2062	76.321507795	192.168.100.70	192.168.100.98	UDP	55	46363 → 8000 Len=13
2063	76.335514822	192.168.100.98	192.168.100.70	UDP	43	8000 → 46363 Len=1
2064	76.335697893	192.168.100.70	192.168.100.98	UDP	57	46363 → 8000 Len=15
2065	76.338573737	192.168.100.98	192.168.100.70	UDP	43	8000 → 46363 Len=1
2136	91.050247190	192.168.100.70	192.168.100.98	UDP	70	54274 → 8000 Len=28
2143	91.133433954	192.168.100.98	192.168.100.70	UDP	43	8000 → 54274 Len=1
2144	91.133677428	192.168.100.70	192.168.100.98	UDP	1511	54274 → 8000 Len=1469
2145	91.136846013	192.168.100.98	192.168.100.70	UDP	43	8000 → 54274 Len=1

Obr. 12: Poslanie správy

5.4 Switch Packet

ip.addr == 192.168.100.70 && udp && ip.addr == 192.168.100.98						
No.	Time	Source	Destination	Protocol	Length	Info
1960	60.952911072	192.168.100.70	192.168.100.98	UDP	43	8000 → 33281 Len=1
1961	60.955040316	192.168.100.98	192.168.100.70	UDP	1511	33281 → 8000 Len=1469
1962	60.955203323	192.168.100.70	192.168.100.98	UDP	43	8000 → 33281 Len=1
1963	60.957220507	192.168.100.98	192.168.100.70	UDP	1511	33281 → 8000 Len=1469
1964	60.957341769	192.168.100.70	192.168.100.98	UDP	43	8000 → 33281 Len=1
1965	60.959343966	192.168.100.98	192.168.100.70	UDP	1511	33281 → 8000 Len=1469
1966	60.959410855	192.168.100.70	192.168.100.98	UDP	43	8000 → 33281 Len=1
1967	60.961565524	192.168.100.98	192.168.100.70	UDP	1511	33281 → 8000 Len=1469
1968	60.961685771	192.168.100.70	192.168.100.98	UDP	43	8000 → 33281 Len=1
1969	60.964935453	192.168.100.98	192.168.100.70	UDP	1511	33281 → 8000 Len=1469
1970	60.965050237	192.168.100.70	192.168.100.98	UDP	43	8000 → 33281 Len=1
1971	60.966942454	192.168.100.98	192.168.100.70	UDP	1511	33281 → 8000 Len=1469
1972	60.967004747	192.168.100.70	192.168.100.98	UDP	43	8000 → 33281 Len=1
1973	60.969416075	192.168.100.98	192.168.100.70	UDP	1511	33281 → 8000 Len=1469
1974	60.969464562	192.168.100.70	192.168.100.98	UDP	43	8000 → 33281 Len=1
1975	60.971485826	192.168.100.98	192.168.100.70	UDP	1511	33281 → 8000 Len=1469
1976	60.971548526	192.168.100.70	192.168.100.98	UDP	43	8000 → 33281 Len=1
1977	60.973723725	192.168.100.98	192.168.100.70	UDP	1511	33281 → 8000 Len=1469
1978	60.973788771	192.168.100.70	192.168.100.98	UDP	43	8000 → 33281 Len=1
1979	60.976072713	192.168.100.98	192.168.100.70	UDP	364	33281 → 8000 Len=322
1980	60.976165169	192.168.100.70	192.168.100.98	UDP	43	8000 → 33281 Len=1
2000	64.841956394	192.168.100.70	192.168.100.98	UDP	43	8000 → 33281 Len=1
2001	64.917315006	192.168.100.98	192.168.100.70	UDP	43	33281 → 8000 Len=1
2062	76.321507795	192.168.100.70	192.168.100.98	UDP	55	46363 → 8000 Len=13
2063	76.335514822	192.168.100.98	192.168.100.70	UDP	43	8000 → 46363 Len=1
2064	76.335697893	192.168.100.70	192.168.100.98	UDP	57	46363 → 8000 Len=15
2065	76.338573737	192.168.100.98	192.168.100.70	UDP	43	8000 → 46363 Len=1
2136	91.050247190	192.168.100.70	192.168.100.98	UDP	70	54274 → 8000 Len=28
2143	91.133433954	192.168.100.98	192.168.100.70	UDP	43	8000 → 54274 Len=1
2144	91.133677428	192.168.100.70	192.168.100.98	UDP	1511	54274 → 8000 Len=1469
2145	91.136846013	192.168.100.98	192.168.100.70	UDP	43	8000 → 54274 Len=1

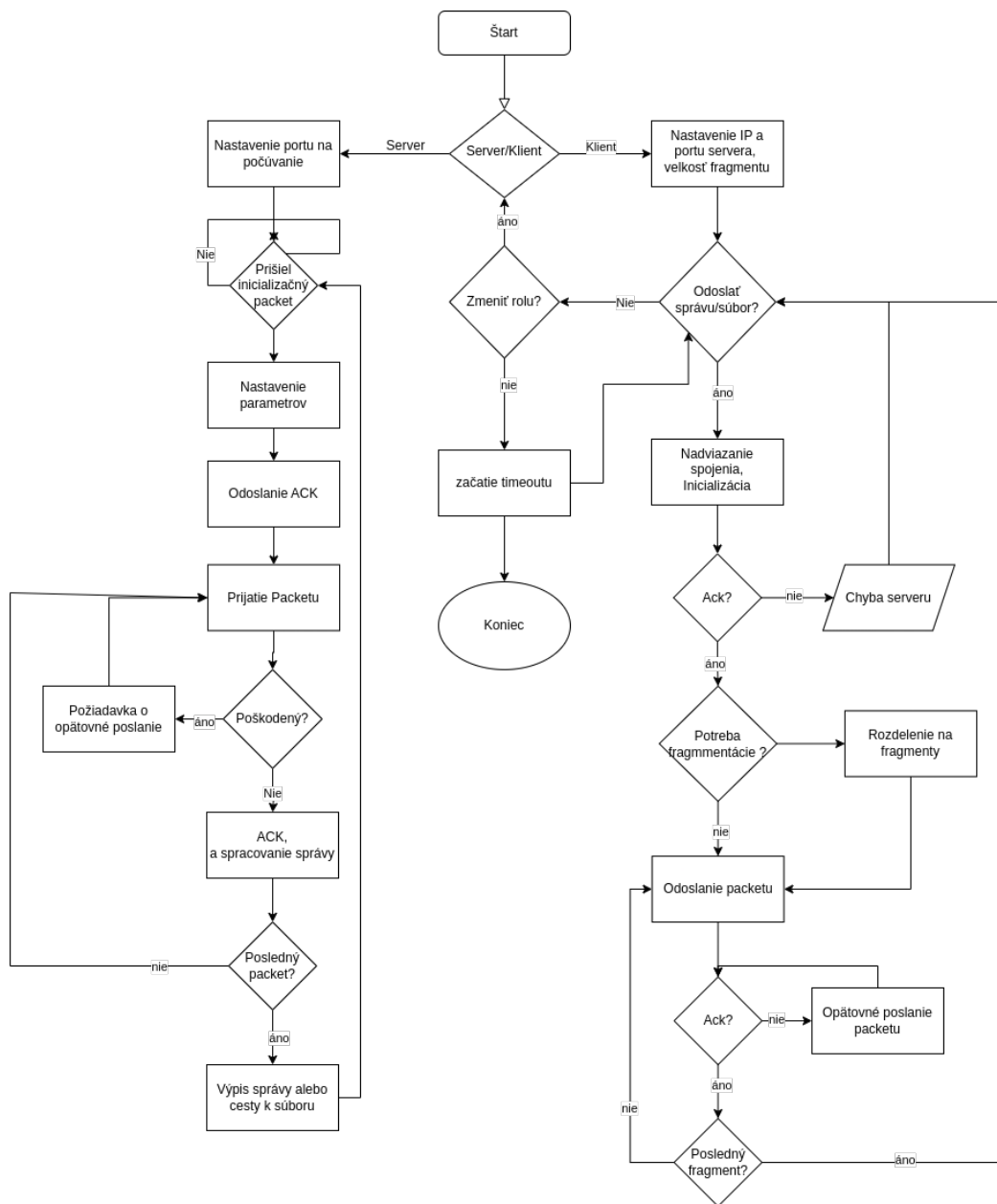
Obr. 13: Switch packet

5.5 Switch Packet na mŕtvy server

ip.addr == 192.168.100.70 && udp && ip.addr == 192.168.100.98						
No.	Time	Source	Destination	Protocol	Length	Info
3570	94.148738698	192.168.100.70	192.168.100.98	UDP	1511	54274 → 8000 Len=1469
3571	94.154030912	192.168.100.98	192.168.100.70	UDP	43	8000 → 54274 Len=1
3572	94.154082772	192.168.100.70	192.168.100.98	UDP	1511	54274 → 8000 Len=1469
3573	94.159345921	192.168.100.98	192.168.100.70	UDP	43	8000 → 54274 Len=1
3574	94.159391787	192.168.100.70	192.168.100.98	UDP	1511	54274 → 8000 Len=1469
3575	94.164937113	192.168.100.98	192.168.100.70	UDP	43	8000 → 54274 Len=1
3576	94.164987709	192.168.100.70	192.168.100.98	UDP	1511	54274 → 8000 Len=1469
3577	94.171308446	192.168.100.98	192.168.100.70	UDP	43	8000 → 54274 Len=1
3578	94.171351741	192.168.100.70	192.168.100.98	UDP	1511	54274 → 8000 Len=1469
3579	94.176716740	192.168.100.98	192.168.100.70	UDP	43	8000 → 54274 Len=1
3580	94.176839966	192.168.100.70	192.168.100.98	UDP	1511	54274 → 8000 Len=1469
3581	94.183087614	192.168.100.98	192.168.100.70	UDP	43	8000 → 54274 Len=1
3582	94.183211886	192.168.100.70	192.168.100.98	UDP	1511	54274 → 8000 Len=1469
3583	94.189465936	192.168.100.98	192.168.100.70	UDP	43	8000 → 54274 Len=1
3584	94.189602057	192.168.100.70	192.168.100.98	UDP	1511	54274 → 8000 Len=1469
3585	94.195749404	192.168.100.98	192.168.100.70	UDP	43	8000 → 54274 Len=1
3586	94.195919034	192.168.100.70	192.168.100.98	UDP	364	54274 → 8000 Len=322
3587	94.198262783	192.168.100.98	192.168.100.70	UDP	43	8000 → 54274 Len=1
3633	99.802108064	192.168.100.70	192.168.100.98	UDP	51	44301 → 8000 Len=9
3634	99.835785591	192.168.100.98	192.168.100.70	UDP	43	8000 → 44301 Len=1
3690	111.099096583	192.168.100.98	192.168.100.70	UDP	55	40601 → 8000 Len=13
3691	111.099255211	192.168.100.70	192.168.100.98	UDP	43	8000 → 40601 Len=1
3692	111.103033545	192.168.100.98	192.168.100.70	UDP	57	40601 → 8000 Len=15
3693	111.103199307	192.168.100.70	192.168.100.98	UDP	43	8000 → 40601 Len=1
4017	130.763323971	192.168.100.98	192.168.100.70	UDP	51	40592 → 8000 Len=9
4018	130.763347889	192.168.100.70	192.168.100.98	ICMP	79	Destination unreachable (Port unreach)
4248	135.789110225	192.168.100.98	192.168.100.70	UDP	51	40592 → 8000 Len=9
4249	135.789159042	192.168.100.70	192.168.100.98	ICMP	79	Destination unreachable (Port unreach)
4455	140.775583978	192.168.100.98	192.168.100.70	UDP	51	40592 → 8000 Len=9
4456	140.775622386	192.168.100.70	192.168.100.98	ICMP	79	Destination unreachable (Port unreach)
4947	145.823826701	192.168.100.98	192.168.100.70	UDP	51	40592 → 8000 Len=9
4948	145.823852999	192.168.100.70	192.168.100.98	ICMP	79	Destination unreachable (Port unreach)

Obr. 14: Packet na výmenu, pri mŕtvom serveri

6 Diagram



Obr. 15: Flow chart Server/Klient