Andrejčák Daniel, ID: 120746
Utorok 16:00

PKS – Zadanie 2: Komunikácia s využitím UDP protokolu– dokumentácia riešenia

Spustenie a používanie programu

Tento program bol vytvorený v jazyku Python 3.11.5. Na jeho korektné spustenie sú potrebné knižnice *socket, threading, os.path, time.* Po spustení programu má používateľ na výber medzi vysielačom, prijímačom a ukončením programu. Obe strany fungujú na princípe, že jedno vlákno slúži na vytváranie, posielanie, prijímanie a spracovanie packetov a druhé vlákno slúži ako command line interface pre používateľa, kde zadáva príkazy a správy na interakciu s programom.

Ak sa používateľ rozhodne pre vysielač, tak musí zadať IP a port na ktorom bude počúvať.

```
$ python main.py
R - RECIEVER, T - TRANSMITTER, K - KILL R
IP: 169.254.1.150
PORT: 54321
```

Ak sa používateľ rozhodne pre vysielač, tak musí zadať IP a port z ktorého bude vysielať a IP a port na ktorý bude vysielať (IP a port prijímača)

```
$ python main.py
R - RECIEVER, T - TRANSMITTER, K - KILL T
IP: 169.254.78.163
PORT: 12345
Reciever IP: 169.254.1.150
Reciever PORT: 54321
```

Funkcie vysielača a prijímača

Strana vysielača dokáže posielať správy, alebo súbory. Používateľ ma taktiež kontrolu nad veľkosťou dát, ktoré sa pošlú v jednom packete (fragment), ktorý môže byť veľkosti 1B – 1427B. Túto veľkosť dokáže používateľ nastaviť príkazov FRAG SIZE <veľkosť>.

Poslanie správy – používateľ do interaktívneho okna napíše správu, ktorú chce poslať a stlačí Enter. Táto správa sa podľa potreby rozdelí na fragmenty a pošle sa prijímačovi na adresu, ktorú používateľ predtým zadal.

Strana vysielača

```
$ python main.py
R - RECIEVER, T - TRANSMITTER, K - KILL T
IP: localhost
PORT: 12345
Reciever IP: localhost
Reciever PORT: 54321
Active transmitter on localhost on port 12345
Hello reciever!
FRAG SIZE 50
Hello reciever!
```

Andrejčák Daniel, ID: 120746
Utorok 16:00

Strana prijímača

```
$ python main.py
R - RECIEVER, T - TRANSMITTER, K - KILL R
IP: localhost
PORT: 54321
Active reciever on localhost on port 54321
Hello reciever! (4 fragments with max fragment size 4B, total size 15B
from ('127.0.0.1', 12345))
Hello reciever! from ('127.0.0.1', 12345)
```

Ako je vidieť, ak je správa rozdelená na fragmenty, tak na strane prijímača sa okrem správy zobrazí aj počet fragmentov a celková veľkosť správy, ktorú prijal. Po zvýšení maximálnej veľkosti fragmentu sa rovnaká správa pošle v jednom packete.

Posielanie súborov – ak chce používateľ poslať súbor tak do interaktívneho okna napíše FILE <cesta k súboru>. Rovnako ako pri posielaní správ, aj tu sa podľa maximálnej veľkosti súbor rozdelí a pošle. Jediný rozdiel oproti správam je to, že pri súbore to sú vždy minimálne 2 packety, kde prvý z nich obsahuje názov súboru a druhý jeho obsah. Na strane prijímača, po prijatí celého súboru, musí používateľ zadať cestu, kde sa má súbor uložiť.

Strana vysielača

```
$ python main.py
R - RECIEVER, T - TRANSMITTER, K - KILL T
IP: localhost
PORT: 12345
Reciever IP: localhost
Reciever PORT: 54321
Active transmitter on localhost on port 12345
FRAG SIZE 100
FILE C:\Users\Lenovo\Desktop\test.pdf
```

Strana prijímača

```
$ python main.py
R - RECIEVER, T - TRANSMITTER, K - KILL R
IP: localhost
PORT: 54321
Active reciever on localhost on port 54321
You have recieved a file, please type "SAVE <path with \ as delimiter>" to save the file
SAVE testfolder\
test.pdf was succesfully saved in C:\Users\Lenovo\Documents\stu_fiit\3.semester\PKS\PKS_zada
nie2\testfolder
(1641 fragments with max fragment size 100B, total size 164070 B from ('127.0.0.1', 12345))
```

Vysielač a rovnako aj prijímač majú možnosť vymeniť si svoju rolu. Po napísaní SWITCH do interaktívneho okna (na ktorejkoľvek strane). Dôjde k výmene prijímača a vysielača bez ukončenia programu. Podmienky sú, že prepínanie sa nemôže vykonať počas posielania správy, alebo súboru a prijímač nemôže inicializovať výmenu skôr ako dostane aspoň 1 správu, pretože predtým ešte nepozná IP a port vysielača, s ktorým sa má vymeniť.

Andrejčák Daniel, ID: 120746
Utorok 16:00

Udržiavanie spojenia

Na udržiavanie spojenia je určený špeciálny typ správy. Tieto správy začne vysielač posielať, ak používateľ nepošle nič aspoň 5 sekúnd. Vysielač na takúto správu odpovie správou rovnakého typu. To sa opakuje každých 5 sekúnd, až kým sa používateľ nerozhodne niečo poslať.

Ak prijímač na takúto správu neodpovie, tak na strane vysielača vypíše správa o tom, že prijímač neodpovedá a navrhne možnosť ukončiť spojenie. Rovnako tak, ak prijímač nedostane 25 sekúnd nejakú správu, tak vypíše správu o tom, že mu neprišla žiadna správa a navrhne spojenie ukončiť .

Strana vysielača

```
Could not reach, type "CLOSE TRANSMITTER" to close the transmitter
```

Strana prijímača

```
No message recieved from transmitter, if you wish not to continue type "CLOSE RECIEVER"
```

Andrejčák Daniel, ID: 120746
Utorok 16:00

Štruktúra hlavičky navrhnutého protokolu

Hlavička navrhnutého protokolu je dĺžky 5 bytov. V prvom byte sa nachádza typ správy a informácie o fragmentácii správy. V ďalších 2 bytoch je ID správy, čo je v podstate poradie správy v komunikácii. To sa využíva pri odosielaní ACK packetov a pri kontrole správnosti prenosu správ. Ďalšie 2 byty obsahujú kontrolnú sumu, ktorá slúži na detekciu chyby pri prenose správy.

Oproti pôvodnému návrhu pribudli typy Switch a Close

0,5	1	2	3	4
Type	Fragment	Identifier	Checksum ...	
... checksum		Data ...		

Type:

0- ACK
1- Message
2- File transfer – file name – názov súboru, ktorý sa posiela
3- File transfer – file content – obsah (časť obsahu) posielaného súboru
4- Close – informačná správa o ukončení vysielača
5- Switch – správy na inicializáciu a potvrdenie výmeny vysielača a prijímača
6- Remain connection – správy na udržanie spojenia (pripadne na zistenie prerušenia spojenia)
7- Error

Fragment:

1- no fragment
2- first fragment
3- more fragments
4- last fragment

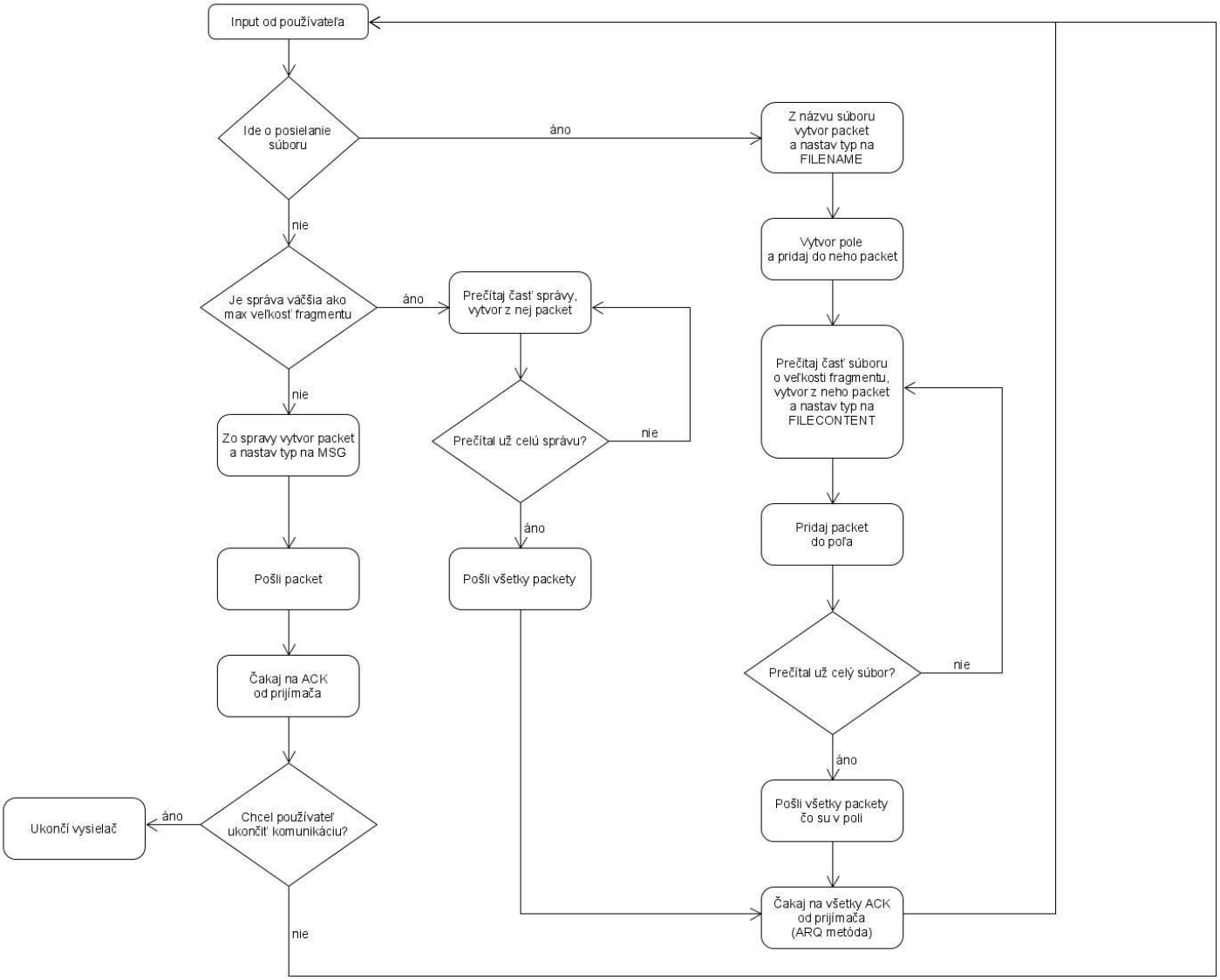
Trieda Protocol

V tomto programe je implementovaná vlastná trieda Protocol, ktorá reprezentuje packet, ktorý je odoslaný / prijatý v komunikácii. Je to vrstva abstrakcie nad surovými dátami, ktoré sa prenášajú v komunikácii, ktorá slúži na jednoduchšiu prácu s týmito dátami a zaručuje väčšiu prehľadnosť kódu. Po načítaní dát, ktoré chce vysielač poslať, sa vytvorí objekt tejto triedy, čo môžeme chápať že sa vytvorí korektná hlavička (nastavenie typu, fragment flagov, nastavedie ID, výpočet checksum), ku ktorej sa pripoja načítané dáta a to sa spolu pošle. Prijímač, po prijatí týchto dát opäť vytvorí objekt tejto triedy a následne tak vie osobitne pracovať s hlavičkou a samotnými dátami, ktoré chcel vysielač poslať.

Andrejčák Daniel, ID: 120746
Utorok 16:00

Diagramy spracovania komunikácie

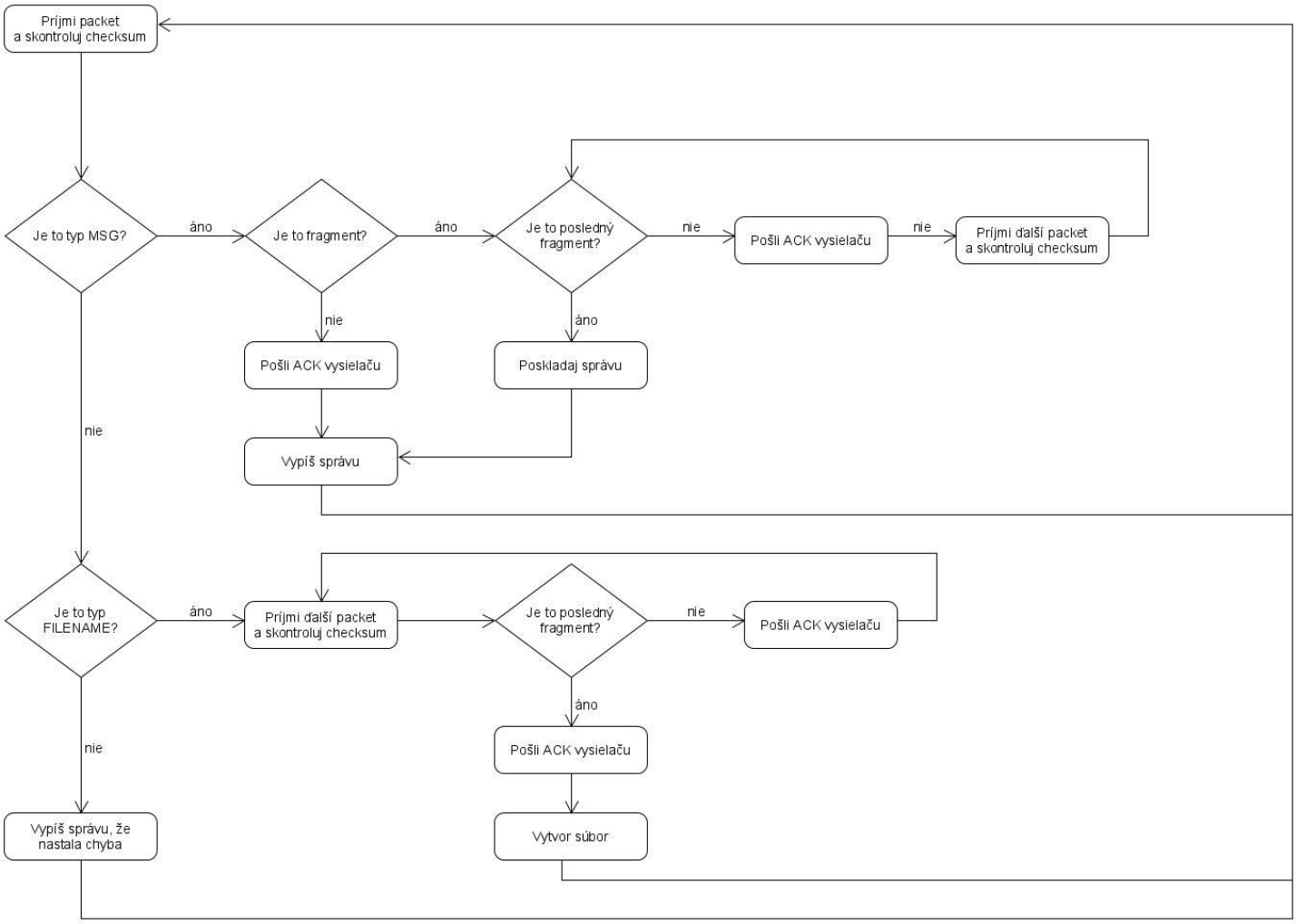
Vysielač



Vytváranie packetov môžeme chápať ako vytvorenie objektu Protocol, nastavenie dát objektu (správa zadaná používateľom, alebo časť súboru), nastavenie fragment flagov, nastavenie identifier, vypočítanie a nastavenie checksum.

Andrejčák Daniel, ID: 120746
Utorok 16:00

Prijímač



Ak pri kontrole checksum program zistí, že prenesené dáta boli poškodené, tak vysielaču pošle správu typu error a prejde do stavu, kedy čaká na prijatie správy

Kontrolná suma – checksum

V tomto programe sa na overenie integrity a správnosti posielaných správ používa metóda podobná CRC (cyclic redundancy check). Na strane vysielača sa po vytvorení objektu Protocol vypočíta kontrolná suma, ktorá sa uloží do poľa Checksum. Na strane prijímača sa po prijatí správy opäť spraví rovnaký výpočet a ak sa výsledok rovná s kontrolnou sumou prijatou zo správy, tak môžeme správu považovať za korektnú.

Výpočet kontrolnej sumy prebieha tak, že vysielač a prijímač majú určenú rovnakú masku (v tomto prípade **0x11021**), ktorou sa dáta (reprezentované ako číslo v hexadecimálnom tvare) vydelia (použitie operácie XOR). Zvyšok po delení sa bude brať ako kontrolná suma.

Andrejčák Daniel, ID: 120746
Utorok 16:00

ARQ metóda

Metóda na zabezpečenie spoľahlivého prenosu dát v tomto programe funguje na princípe Selective Repeat ARQ. Na strane vysielača sa nastaví sliding window o veľkosti N a toľko packetov sa pošle. Vysielač potom čaká na ACK packet od prijímača. Akonáhle vysielač príjme ACK, tak odošle ďalší packet a opäť čaká na ACK packet od prijímača.

Vysielačom odoslané packety sa postupne pridávajú do queue a postupne ako prichádzajú jednotlivé packety ACK, tak sa z queue odoberajú. To znamená, že v určitý moment je v danej queue maximálne N odoslaných packetov.

Ak nastane situácia, že na určitý odoslaný packet nepríde ACK a na packet, ktorý bol odoslaný až po ňom príde, tak sa tento packet odoberie z queue, pošle sa znova a opäť sa pridá do queue. To znamená, že odoslaných packetov a prijatých ACK bude n+m, kde n je počet packetov, ktoré sa majú odoslať a m je počet packetov, ktoré boli odoslané znova.

Simulácia chyby + ukážka z Wireshark

Program podporuje ukážku – simuláciu chyby a jej následnú opravu. Ak na strane vysielača používateľ napíše SIMULATE ERROR, tak sa spustí simulácia chyby. Do kontrolnej sumy jedného z packetov je zámerne vnesená chybná hodnota, ktorú prijímač rozpozná a následne si opäť vyžiada tento chybný packet. Na oboch stranách je vidno, že došlo k “poškodeniu“ packetu.

Strana vysielača

```
$ python main.py
R - RECIEVER, T - TRANSMITTER, K - KILL T
IP: localhost
PORT: 12345
Reciever IP: localhost
Reciever PORT: 54321
Active transmitter on localhost on port 12345
SIMULATE ERROR
Retransmission of packet 5
```

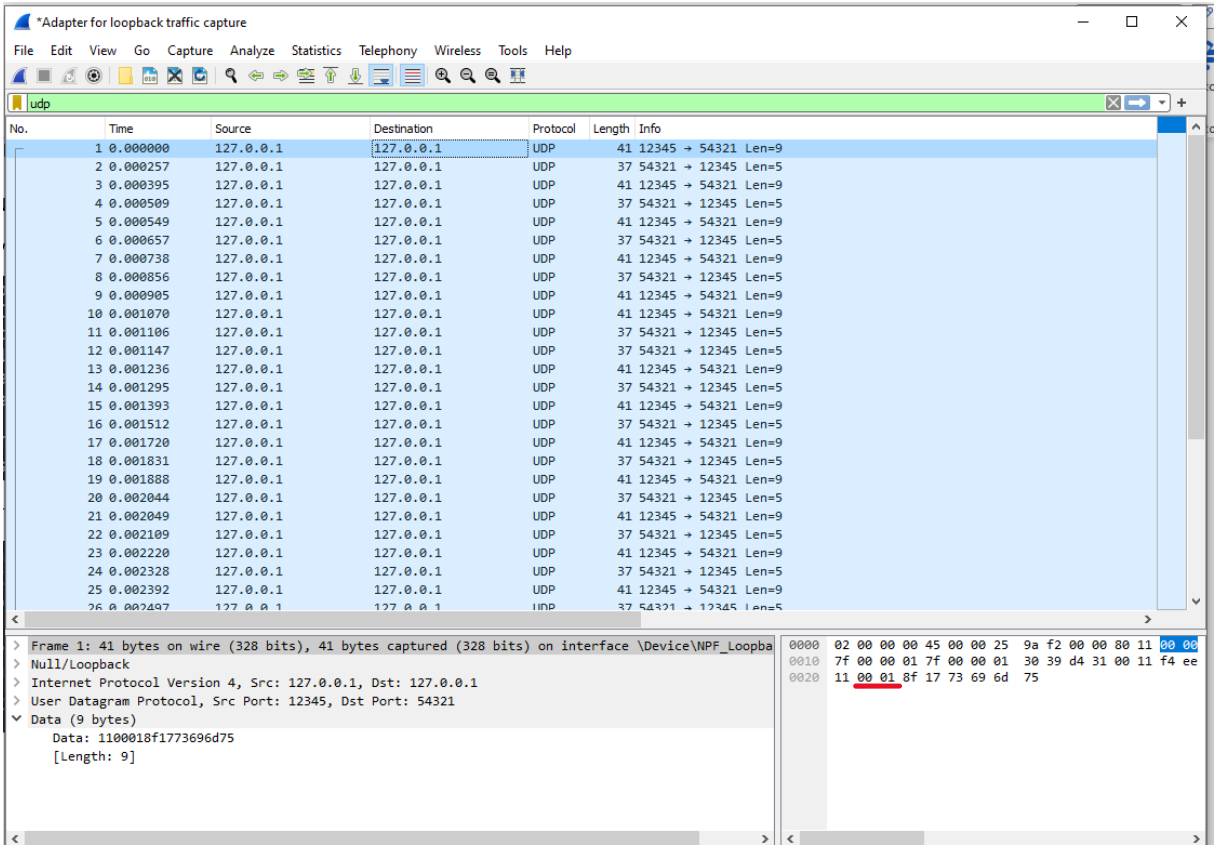
Strana prijímača

```
$ python main.py
R - RECIEVER, T - TRANSMITTER, K - KILL R
IP: localhost
PORT: 54321
Active reciever on localhost on port 54321
Bad checksum on packet 5
simulating error, 5th fragment will have bad checksum (14 fragments with
 max fragment size 4B, total size 53B from ('127.0.0.1', 12345))
```

V nasledujúcich screenshotoch z Wireshark je vidieť ako ARQ metóda selective repeat opäť pošle iba packet, ktorý bol prijatý ako poškodený.

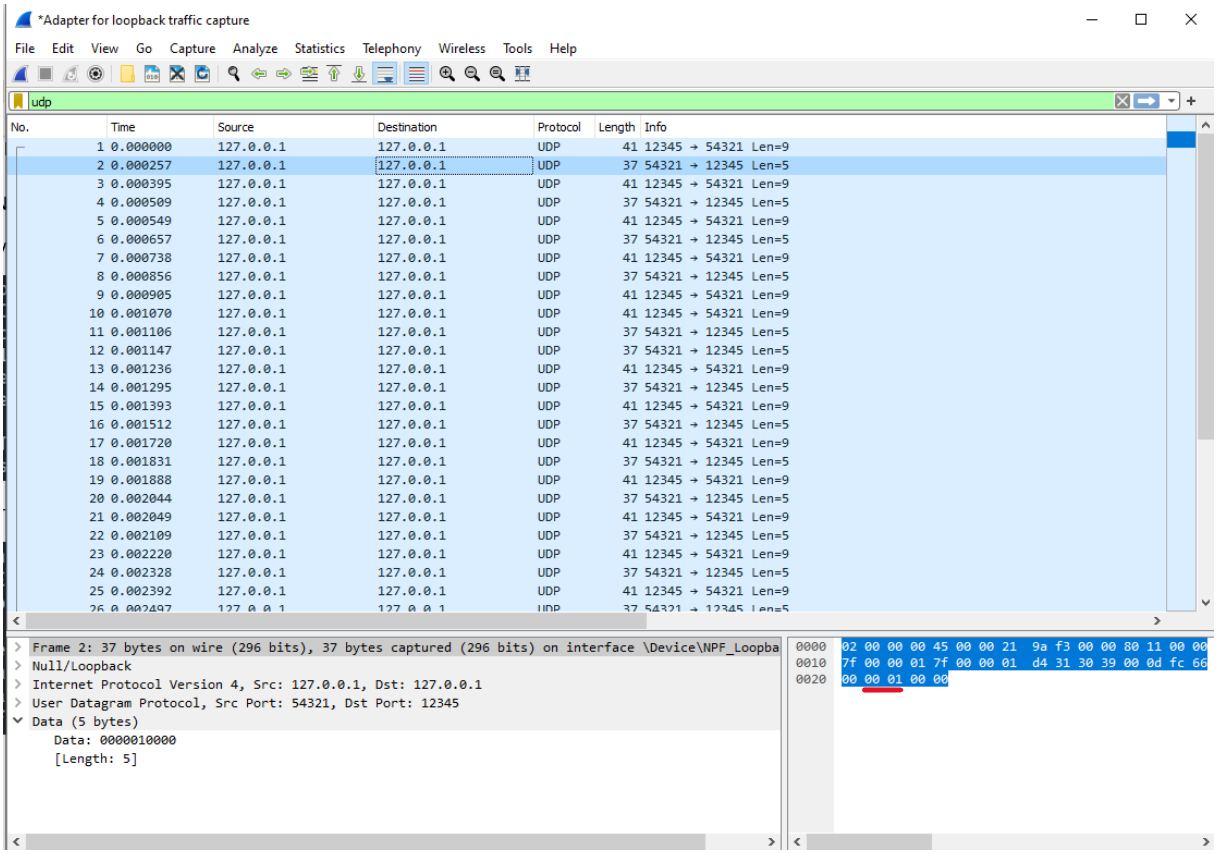
Andrejčák Daniel, ID: 120746
Utorok 16:00

Ako prvý sa poslal packet s ID 0001, čo môžeme vidieť na tomto screenshote. V hlavičke je ID 2. a 3. byte. 11 v hlavičke znamená, že sa jedná o správu a o prvý fragment.



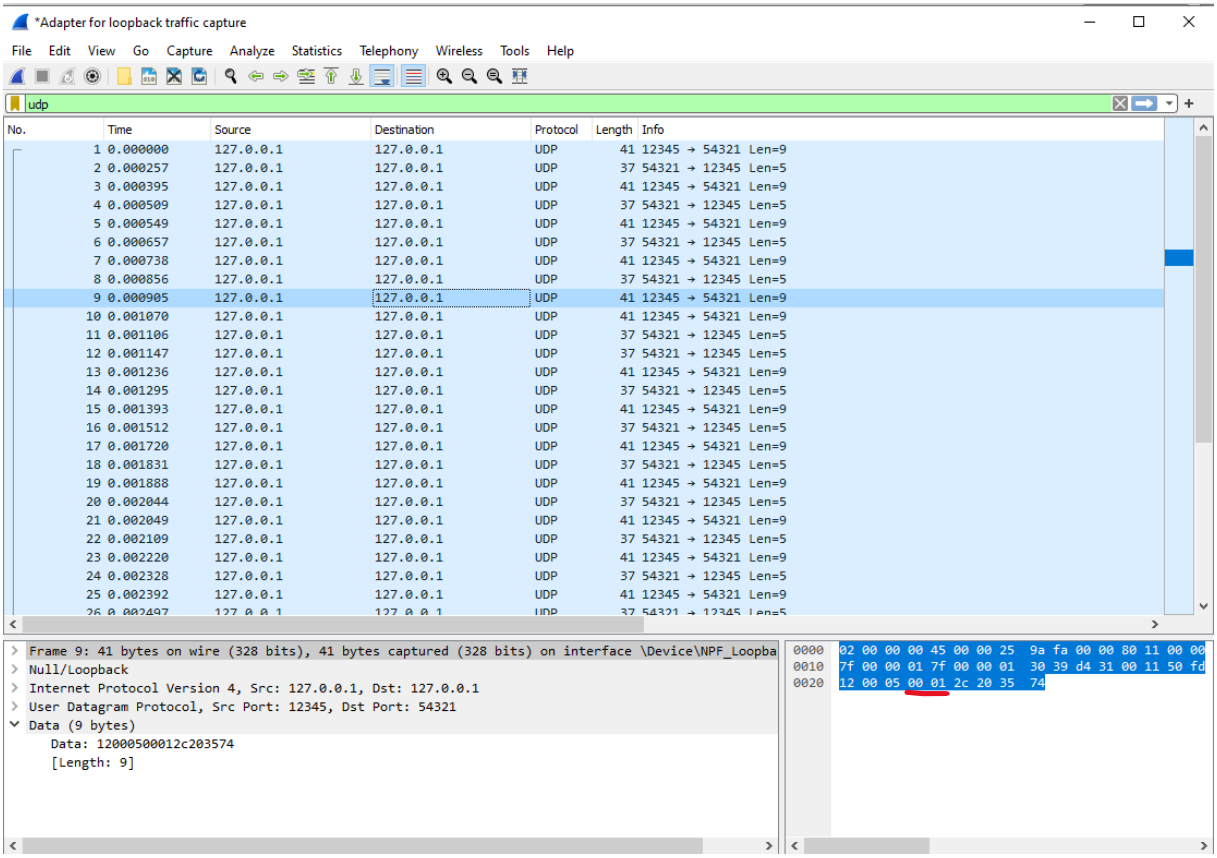
V druhom screenshote je vidieť ACK na prvý packet, ktorý poslal prijímač (Wireshark to zobrazuje takto, ale program v skutočnosti poslal prvé 4 packety a až potom prijal ACK na prvý).

Ako môžeme vidieť, ID sa zhodujú, takže prijímač tento packet prijal ako korektný.

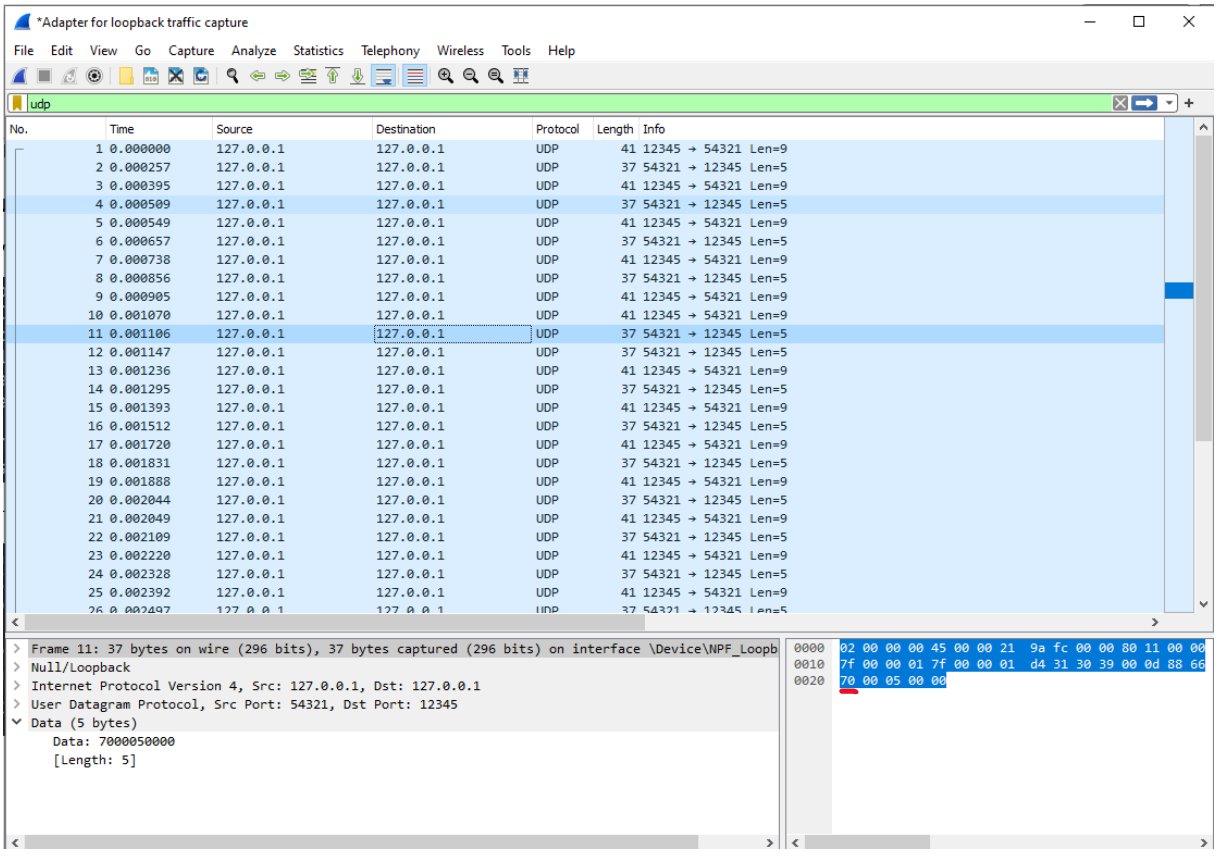


Andrejčák Daniel, ID: 120746
Utorok 16:00

Takto sa korektne pošlú a príjmu prvé 4 packety. 5. packet je však poškodený (checksum – 4. a 5. byte je manuálne nastavený na 0001).

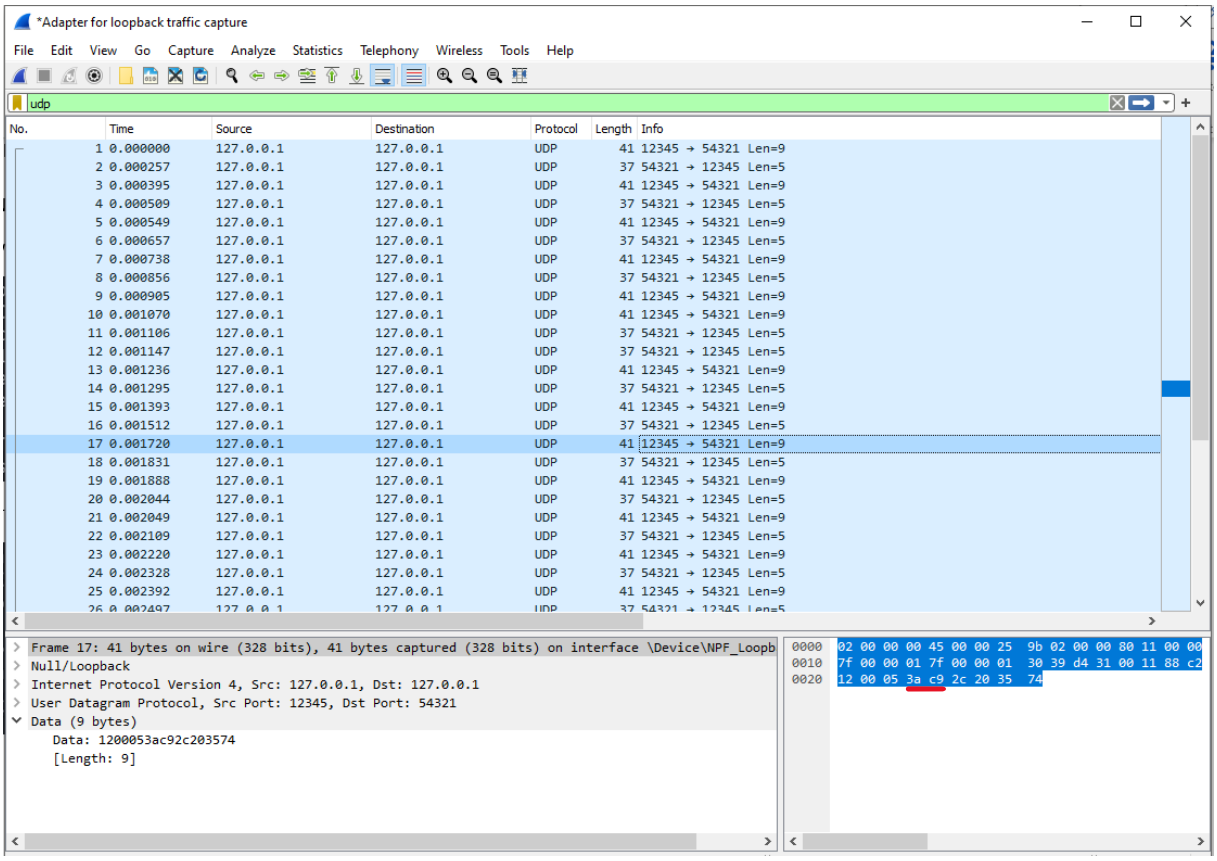


Prijímač to zistil a poslal správu typu ERR (prvý byte je nastavený na 7x), čo značí, že daný packet nie je považovaný za korektné prijatý. Môžeme si všimnúť, že tieto packety majú rovnaké ID (0005). Podľa toho vie vysielač rozoznať, pre aký packet, práve prijal ACK (alebo ERR).

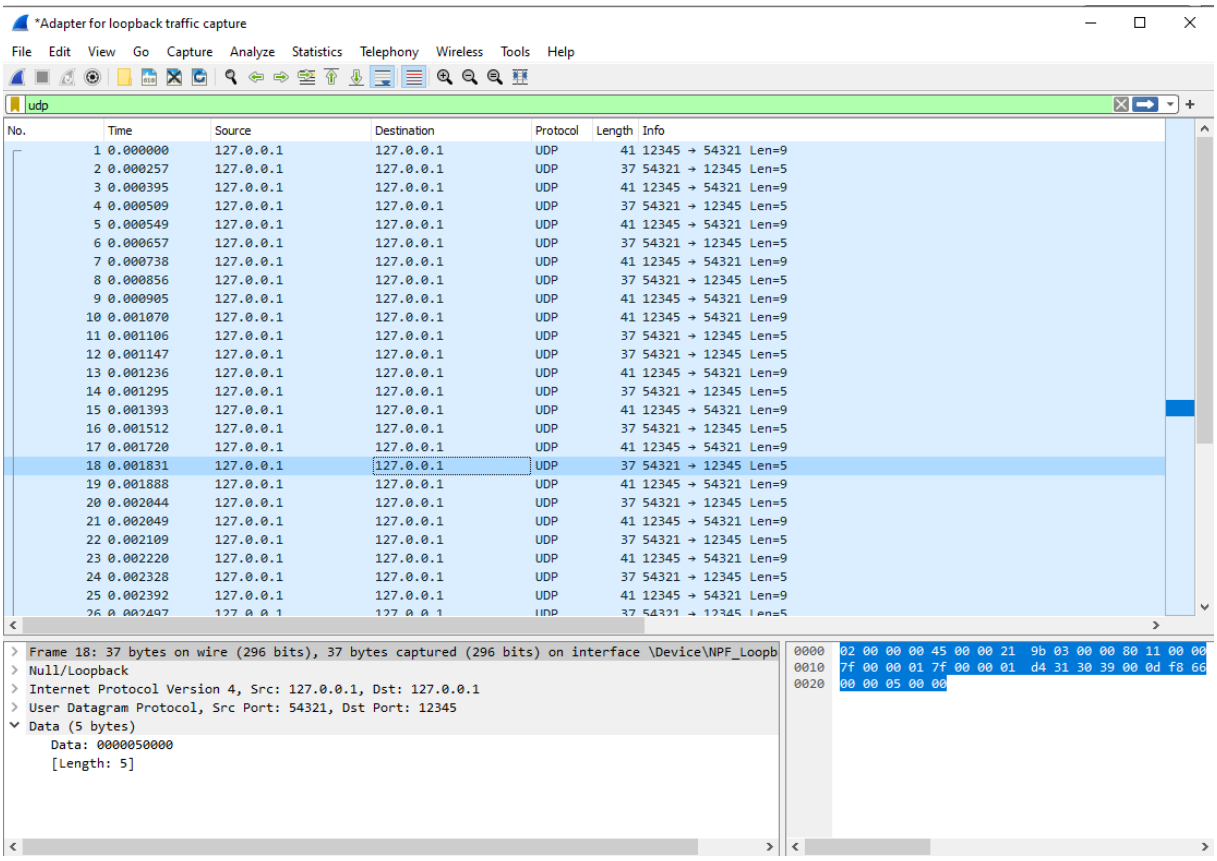


Andrejčák Daniel, ID: 120746
Utorok 16:00

Po prijatí ERR packetu, vysielač vie, že niekde nastala chyba a tak tento packet (ID: 0005) pošle znova (kedže ide o simuláciu chyby, tak sa mu nastaví korektný checksum).



Tento krát už prijímač potvrdí korektné prijatie packetu tým, že pošle ACK s ID 0005.

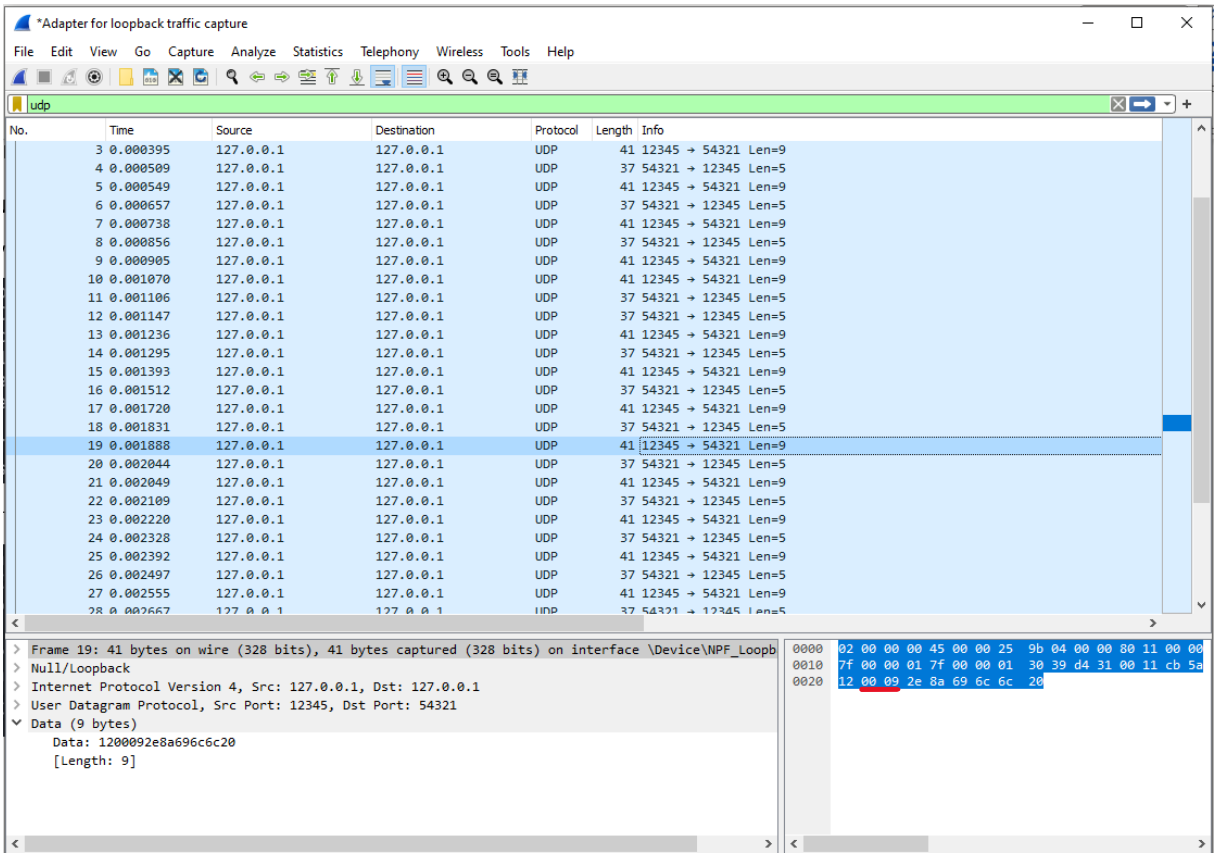


Andrejčák Daniel, ID: 120746
Utorok 16:00

Môžeme si všimnúť, že tento “opravený“ packet sa neposlal ihneď, ale až po niekoľkých ďalších packetoch. To je preto, že ARQ metóda mala ešte v queue niekoľko packetov, ktoré poslala predtým, ako sa dostala k ERR pre “poškodený“ packet. To však nevadí, pretože prijímač dokáže spätne skonštruovať správu / súbor, aj keď tieto fragmenty nedostane v poradí a to vďaka ID.

Ďalej môžeme vidieť, že nastalo znova-poslanie iba tohto jedného “poškodeného“ packetu a nie žiadnych iných. To je taktiež jednou z výhod ARQ metódy selective repeat, že zbytočne znova nepošle packety, ktoré boli prijaté korektne.

17. poslaný packet bol opravený packet s ID 0005, 18. poslaný packet bol ACK pre packet s ID 0005 a 19. packet má ID 0009, čo znamená, že naozaj nedošlo k znova-poslaniu žiadného iného packetu a komunikácia pokračuje ďalej.



Andrejčák Daniel, ID: 120746
Utorok 16:00

Potvrdenie korektného prijatia packetu s ID 0009.