Dokumentácia vlastného protokolu – /PKS

Andrejčík Dávid

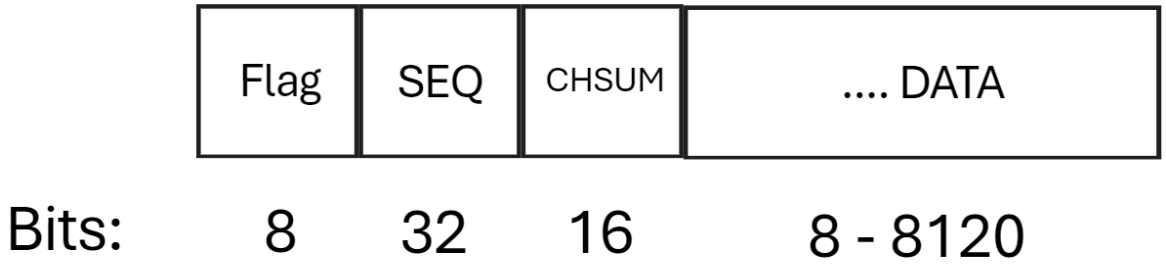
Obsah - Zoznam

OBSAH:

Úvod

Úlohou zadania je navrhnúť vlastný protokol vnorený v UDP protokole na IPV4. UPD protokol je sám o sebe podstatne okresanejší ako spoľahlivejší TCP protokol, dáva však veľa miesta na experimentáciu a tvorenie vlastných metód ošetrovania prichádzajúcich dát.

Hlavička protokolu



Flags

Flag – prvá informácia v hlavičke, hovorí o tom, aký typ správy prichádza.

SYN = 0 → Používa sa na inicializáciu handshake medzi dvoma peerami.

SYN_ACK = 1 → Posiela sa ako odpoveď na SYN a potvrdzuje začatie handshake.

MESS = 2 → Označuje prenos štandardnej textovej správy.

ACK = 3 → Potvrdzuje úspešné prijatie správy alebo fragmentu. Hodnota SEQ zodpovedá naposledy prijatej správe alebo fragmentu, ktorý sa potvrdzuje.

END = 4 → Signalizuje ukončenie spojenia.

KEEP_ALIVE = 5 → Periodicky sa odosiela na zaistenie, že spojenie zostáva aktívne.

NACK = 6 → Požaduje opätovné zaslanie poškodenej alebo chýbajúcej časti správy (fragmentu).

FILE_START = 7 → Označuje začiatok prenosu súboru. Obsahuje názov súboru.

CHUNK = 8 → Predstavuje fragment dát počas prenosu správy alebo súboru.

FILE_END = 9 → Označuje ukončenie prenosu súboru.

MESS_END = 10 → Označuje ukončenie prenosu fragmentovanej textovej správy

SEQ

a) Číslo poslanej správy, posiela sa ako kontrolný údaj
b) Pri ACK , .._REP vráti spätne číslo príchodzej správy
c) Hlavne použité pri vypýtaní si o znovuzaslanie poškodeného fragmentu
d) Na prijímateľovi zabezpečuje aby pri strate spätnej ack správy, nebol fragment obdržaný násobne

Checksum

Hodnota použitá na kontrolu integrity prijatej správy, v tomto protokole – CRC-16 polynóm „CCITT“.

DATA

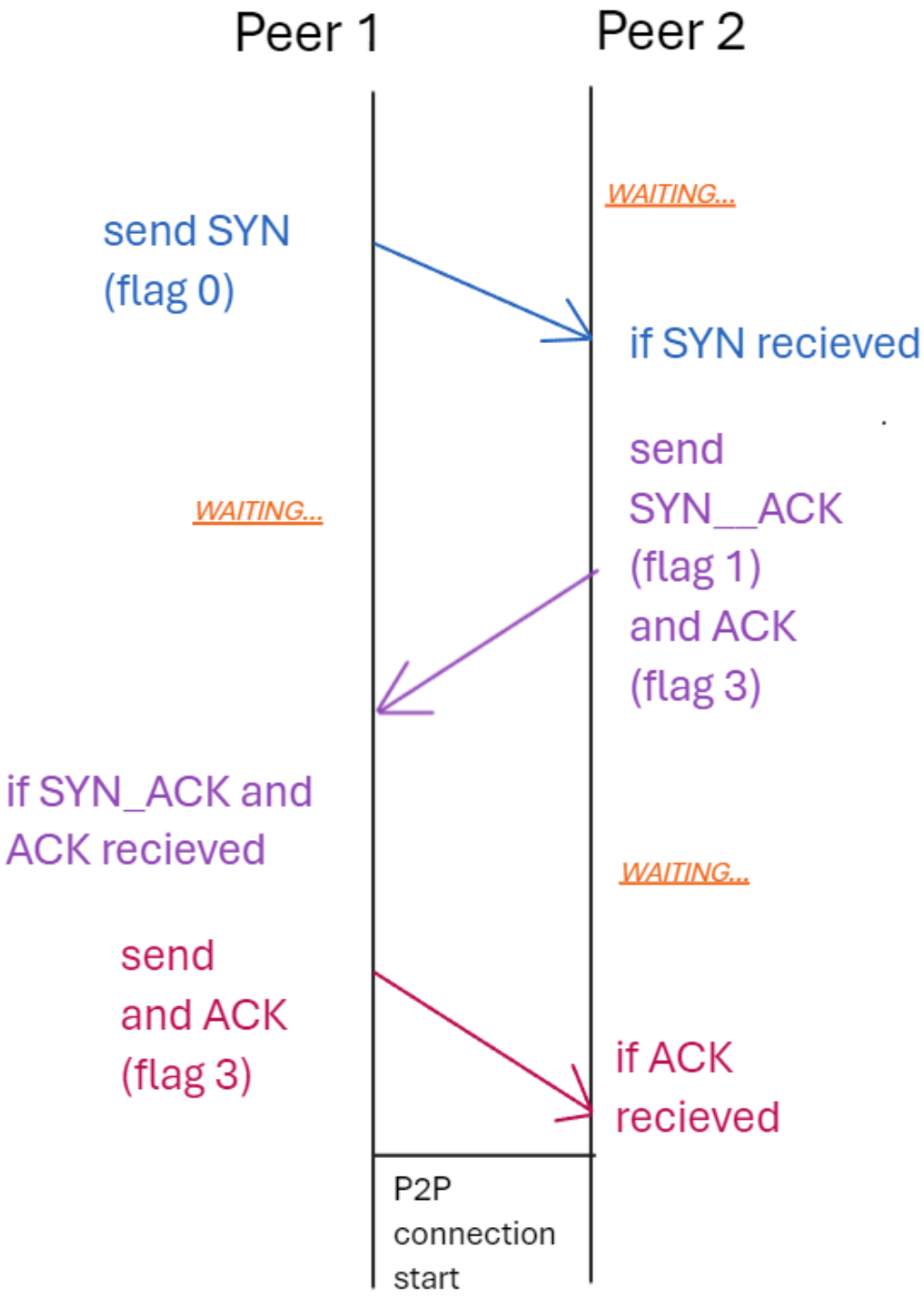
Payload, message...

Časť na uloženie používateľom zvolenej správy alebo jej časti(fragmentu).

Pri obsiahlej správe alebo nastavení na uzle, sa rozdelí na viac fragmentov → pošle sa viac fragmentov, každý s časťou správy. Tá sa na prijímacom uzle poskladá a vypíše.

Handshake – zobrazenie

Za predpokladu že Peer 1 začína s handshakom (je možnosť vybrať si):



PEER 1 – P1

PEER 2 – P2

(pre jednoduchosť, zatiaľ sa spojenie nespráva ako P2P)

P1 zasiela prázdnu správu s flagom na staveným na 0(SYN).

Pokial P2 správu obdrží, zašle späť prázdnu správu s flagom na 1 (SYN_ACK) a zároveň jednu acknowledgment správu.

Pri obdržaní SYN_ACK správy, pošle P1 → P2 jednu acknowledgement správu.

Pokiaľ tento proces prebehne, vytvorí sa P2P spojenie.

Opis metód

Tieto metódy zabezpečujú správny chod programu a riešia problémy ako:

- kontrola integrity
- znovuzaslanie poškodeného fragmentu
- keep-alive

Kontrola integrity

Pomocou CRC-16-CCITT →

CRC-16 z dôvodu časovej a pamäťovej efektivity(rozdiel od Simple Sum, Adler...), zároveň je vhodnejšou kontrolou pre menšie správy (rozdiel od CRC-32)

CCITT je jeden z polynómov používaných kvôli svojím vlastnostiam, vyzerá takto:
0x1021

Kódovanie pomocou CRC-16-CCITT pre správu PKS:

Polynóm: 0x1021

Prevod správy do ASCII hodnôt

P=80 -> 0x50 -> 01010000

K=75 -> 0x4B -> 01001011

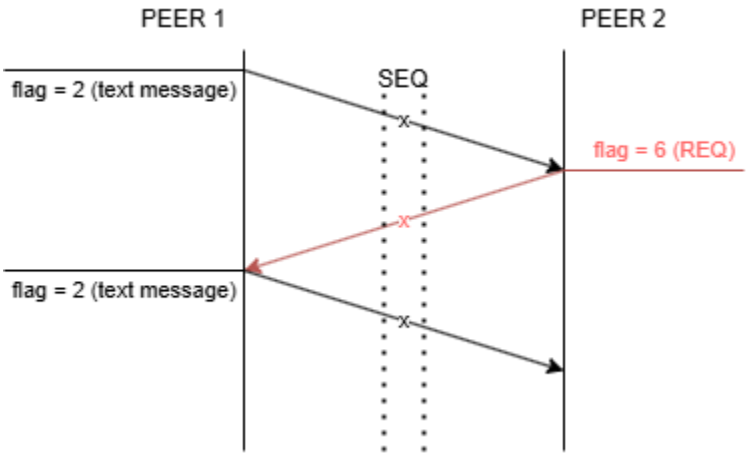
S=83 -> 0x53 -> 01010011

Spolu PKS + append 16*0
01010000 01001011 01010011 00000000 00000000

Predeliť polynómom, zvyšok = 0x29FA (ak presiahne 16bit, najväčšia bitová reprezentácia sa odoberie)

Tento proces sa deje na oboch stranách ale nie len pre stranu, ale pre celý fragment (hlavička + dáta). Ak checksum teda nesúhlasí na oboch stranách pre rovnakú správu, vypýta sa znovu.

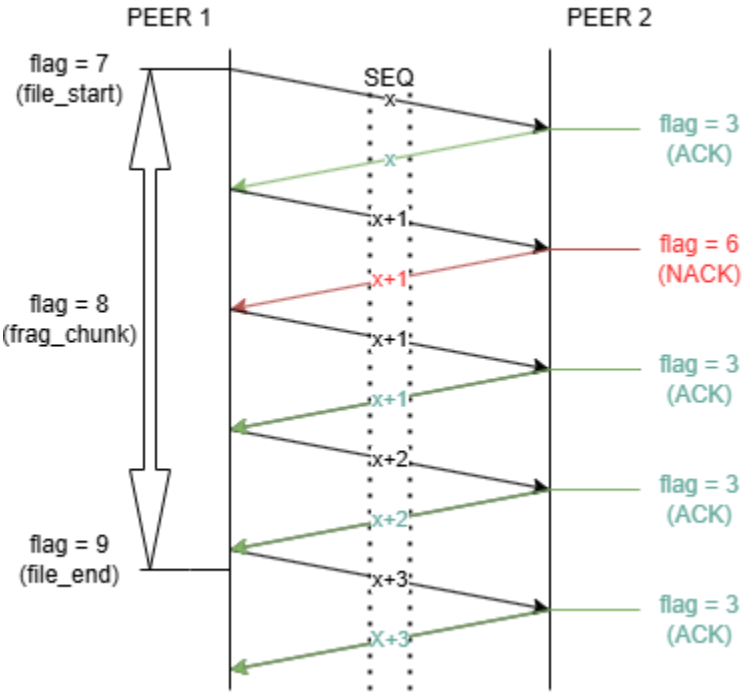
Graficky:



Znovu zaslaný fragment je aj vtedy, ak nie je doručená ACK potvrdzujúca správa.

Poškodený fragment – riešenie

Graficky:

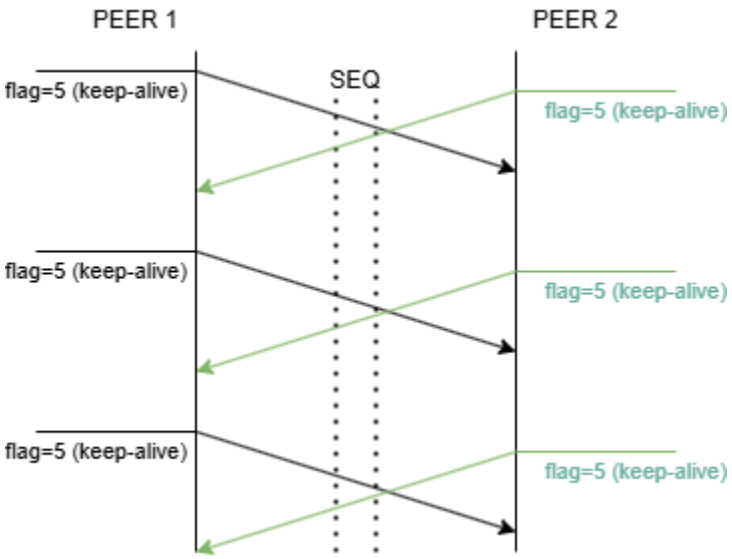


Princíp Go-Back-N ARQ.

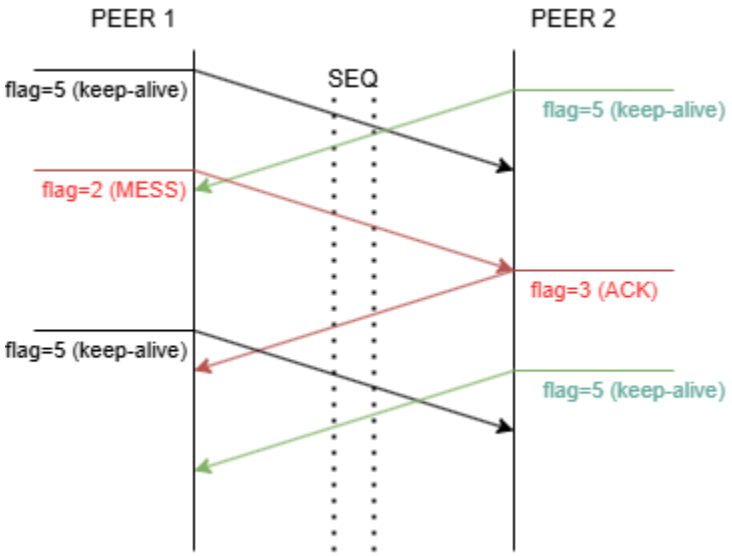
Fragmetny sú posielané po jednom, vždy čakjúce na potvrdenie predošlého fragmentu. Ak sa tak nestane, alebo ak príde správa o poškodenom fragmente, zašle sa fragment znovu.

Keep-alive

Graficky:

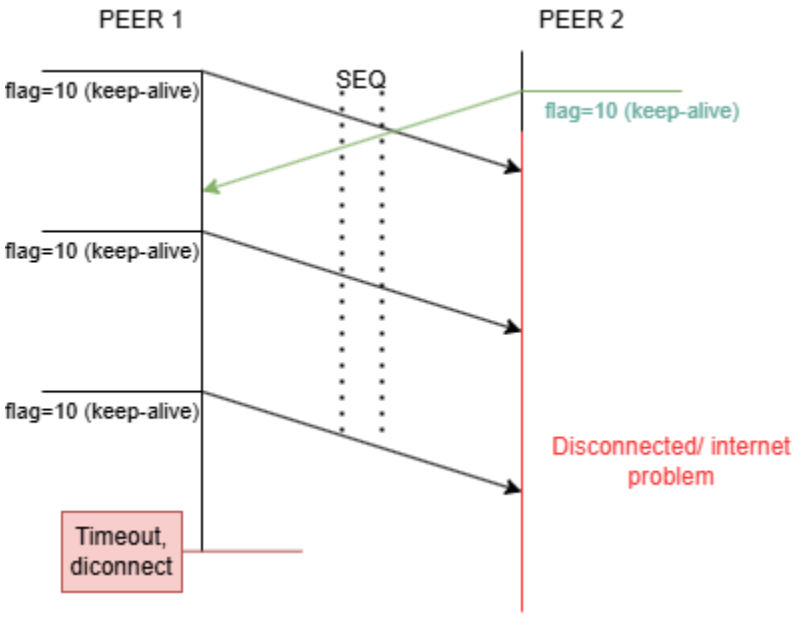


Zaslanie akéhokoľvek fragmentu sa považuje ako potvrdenie spojenia, resetuje sa tak cooldown na posielanie keep-alive spárv.



KEEP_ALIVE /F5/. Tieto fragmenty sú posielané periodicky F5 z oboch strán, nečakajúce na potvrdenie z druhej strany. Ak však nedostane F5 po dobru 3x čakanie na jeden F5, program vyhodnotí že prepojenie bolo stratené.

Prerušenie:



Odchytenie protokolu vo wiresharku

Handshake vo Wiresharku :

22007	898.932077	127.0.0.1	127.0.0.1	MYPROTO	39 50601 → 50605 Len=7
22008	900.935340	127.0.0.1	127.0.0.1	MYPROTO	39 50605 → 50601 Len=7
22009	900.935788	127.0.0.1	127.0.0.1	MYPROTO	39 50601 → 50605 Len=7
22046	906.139464	127.0.0.1	127.0.0.1	MYPROTO	39 50605 → 50601 Len=7
22047	906.139803	127.0.0.1	127.0.0.1	MYPROTO	39 50601 → 50605 Len=7
22057	911.141849	127.0.0.1	127.0.0.1	MYPROTO	39 50605 → 50601 Len=7
22058	911.142387	127.0.0.1	127.0.0.1	MYPROTO	39 50601 → 50605 Len=7

SYN →, SYN_ACK ←, ACK → + KEEP_ALIVE ←/→

Odchytenie posielania správy vo Wiresharku:

File_start

FILE_START-80, ACK-81+2n, CHUNK-82+2n

80	13.129662	127.0.0.1	127.0.0.1	MYPROTO	52	50601 → 50605 Len=20
81	13.130533	127.0.0.1	127.0.0.1	MYPROTO	39	50605 → 50601 Len=7
82	13.132268	127.0.0.1	127.0.0.1	MYPROTO	1056	50601 → 50605 Len=1024
83	13.133294	127.0.0.1	127.0.0.1	MYPROTO	39	50605 → 50601 Len=7
84	13.134916	127.0.0.1	127.0.0.1	MYPROTO	1056	50601 → 50605 Len=1024
85	13.135884	127.0.0.1	127.0.0.1	MYPROTO	39	50605 → 50601 Len=7
86	13.137250	127.0.0.1	127.0.0.1	MYPROTO	1056	50601 → 50605 Len=1024
87	13.137950	127.0.0.1	127.0.0.1	MYPROTO	39	50605 → 50601 Len=7
88	13.139424	127.0.0.1	127.0.0.1	MYPROTO	1056	50601 → 50605 Len=1024
89	13.140133	127.0.0.1	127.0.0.1	MYPROTO	39	50605 → 50601 Len=7
90	13.141278	127.0.0.1	127.0.0.1	MYPROTO	1056	50601 → 50605 Len=1024
91	13.141823	127.0.0.1	127.0.0.1	MYPROTO	39	50605 → 50601 Len=7
92	13.143915	127.0.0.1	127.0.0.1	MYPROTO	1056	50601 → 50605 Len=1024
93	13.144878	127.0.0.1	127.0.0.1	MYPROTO	39	50605 → 50601 Len=7
94	13.146238	127.0.0.1	127.0.0.1	MYPROTO	1056	50601 → 50605 Len=1024
95	13.146951	127.0.0.1	127.0.0.1	MYPROTO	39	50605 → 50601 Len=7
96	13.148389	127.0.0.1	127.0.0.1	MYPROTO	1056	50601 → 50605 Len=1024
97	13.149065	127.0.0.1	127.0.0.1	MYPROTO	39	50605 → 50601 Len=7
98	13.150465	127.0.0.1	127.0.0.1	MYPROTO	1056	50601 → 50605 Len=1024
99	13.151022	127.0.0.1	127.0.0.1	MYPROTO	39	50605 → 50601 Len=7
100	13.152413	127.0.0.1	127.0.0.1	MYPROTO	1056	50601 → 50605 Len=1024
101	13.153002	127.0.0.1	127.0.0.1	MYPROTO	39	50605 → 50601 Len=7

Obdržanie NACK správy na poškodený packet

17864	579.179008	127.0.0.1	127.0.0.1	MYPROTO	39	50605 → 50601 Len=7
17865	579.180439	127.0.0.1	127.0.0.1	MYPROTO	1056	50601 → 50605 Len=1024
17866	579.181060	127.0.0.1	127.0.0.1	MYPROTO	39	50605 → 50601 Len=7
17867	579.182507	127.0.0.1	127.0.0.1	MYPROTO	1056	50601 → 50605 Len=1024
17868	579.183098	127.0.0.1	127.0.0.1	MYPROTO	39	50605 → 50601 Len=7
17869	579.184533	127.0.0.1	127.0.0.1	MYPROTO	1056	50601 → 50605 Len=1024
17870	579.185090	127.0.0.1	127.0.0.1	MYPROTO	39	50605 → 50601 Len=7
17876	582.191016	127.0.0.1	127.0.0.1	MYPROTO	1056	50601 → 50605 Len=1024
17877	582.191679	127.0.0.1	127.0.0.1	MYPROTO	39	50605 → 50601 Len=7
17878	582.192741	127.0.0.1	127.0.0.1	MYPROTO	1056	50601 → 50605 Len=1024
17879	582.193345	127.0.0.1	127.0.0.1	MYPROTO	39	50605 → 50601 Len=7
17880	582.194956	127.0.0.1	127.0.0.1	MYPROTO	1056	50601 → 50605 Len=1024
17881	582.195733	127.0.0.1	127.0.0.1	MYPROTO	39	50605 → 50601 Len=7
17882	582.197093	127.0.0.1	127.0.0.1	MYPROTO	1056	50601 → 50605 Len=1024

Zmeny počas implementácie

- Zjednodušenie KEEP_ALIVE na obojstranne nezávislé posielanie.
  Strana nedostane ack o hearbeate, ak prestane dostavať správy alebo heartbeat správy, usúdi že druhá strana je odpojená
- Prechod na jednoduchšiu STOP N WAIT kontrolu, fragment tak vždy čaká na ack pred poslaním ďalšieho fragmentu. Zanedbateľne pomalšie a podstatne jednoduchšie na implementáciu.
- Text. správa sa fragmentuje rovnakým spôsobom ako súbor, ukončenie však nastane iným flagom. Jednoduchšie používanie buffera, lepšia prehľadnosť kódu.

Záver

Vytvorený návrh protokolu ktorý vypĺňa základné nedostatky UDP protokolu a zároveň implementácia a úprava fungovania kódu tak, aby spĺňal požiadavky ako:
integrita, udržanie spojenia, spoľahlivý prenos dát, prenos súborov a textových správ na P2P typu vzťahu medzi strojmi.