Zadanie 1: Komunikácia s využitím UDP protokolu

Branislav Trstenský

Úvod

Úlohou programu odoslať alebo prijať súbor a textové správy prostredníctvom UDP protokolu zo zabezpečením spoľahlivej komunikácie. Tiež je potrebné poslať súbor fragmentovať podľa užívateľom špecifikovanej veľkosti fragmentu.

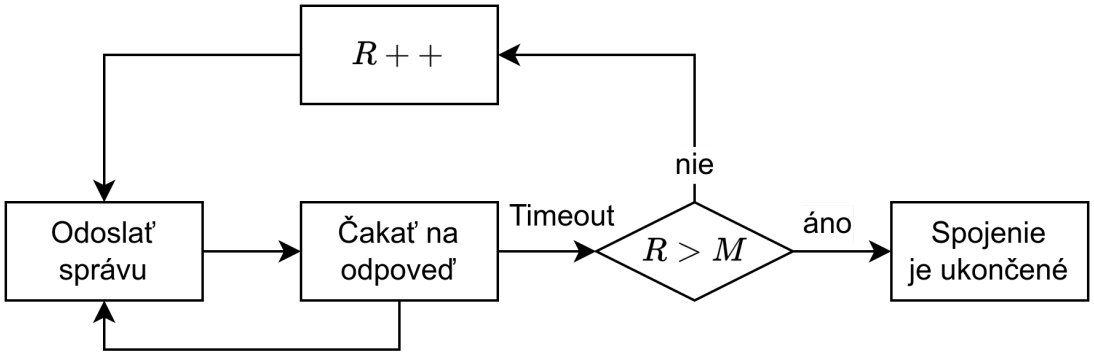
Protokol

Protokol využíva krátke správy na manažovanie komunikácie a dátové segmenty pre prenos dát. Obidva spôsoby komunikácie majú metódu spoľahlivého doručenia správy a zaistenie správneho poradia.

Protokol tiež aktívne udržiava komunikáciu z ukončením spojenia ak komunikant neodpovedá a kontrolu správnosti doručených dát pomocou kontrolnej sumy algoritmu CRC32.

Odosielani správ

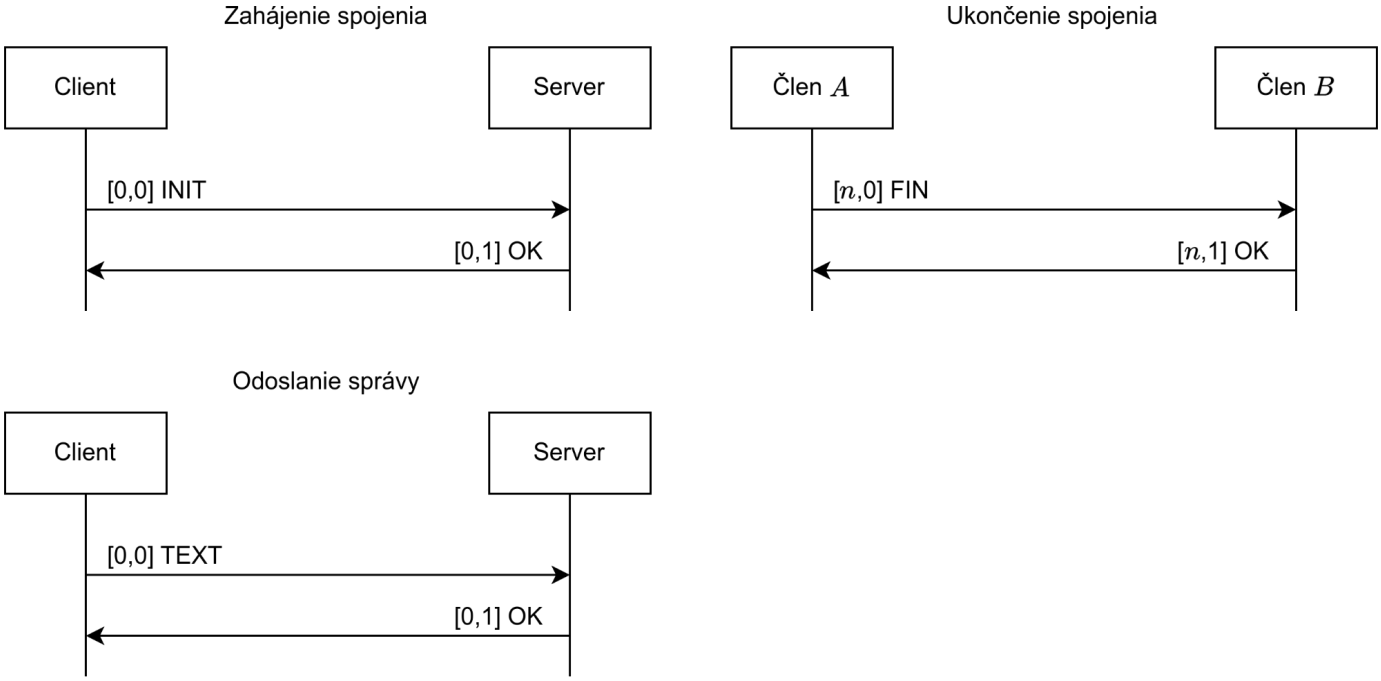
- Krátky prenos na manažovanie komunikácie
- Správy existujú v prúdoch, každý prúd ma ID, nový prúd má $+2$ od predošlého
- Klient počíta od 0, server od 1 - žiadna kolízia
- V jednom prúde sa odosielateľ a prijímateľ striedajú
- Každá správa má ID, vždy $+1$ ako predošlá v prúde
- Prúd ukončený správou OK, odpoveď sa neočakáva
- Ak nedôjde odpoveď na správu do $\mathrm{TIMEOUT}$ správa je odoslaná znova
- Ak príde správa v prúde s nesprávnym ID (moc veľkým/malým) je ignorovaná



Udržiavanie komunikácie

- Každý program sleduje čas od posledného odoslaného segmentu
- Ak je čas prekročí $\mathrm{PING\_INTERVAL}$ je odoslaná PING správa
- Program túto správu opakuje až kým nedostane odpoveď, alebo nie je dosiahnutý $\mathrm{REPEAT\_LIMIT}$
- Po dosiahnutí limitu je spojenie ukončené

Operácie protokolu



Kontrolná suma

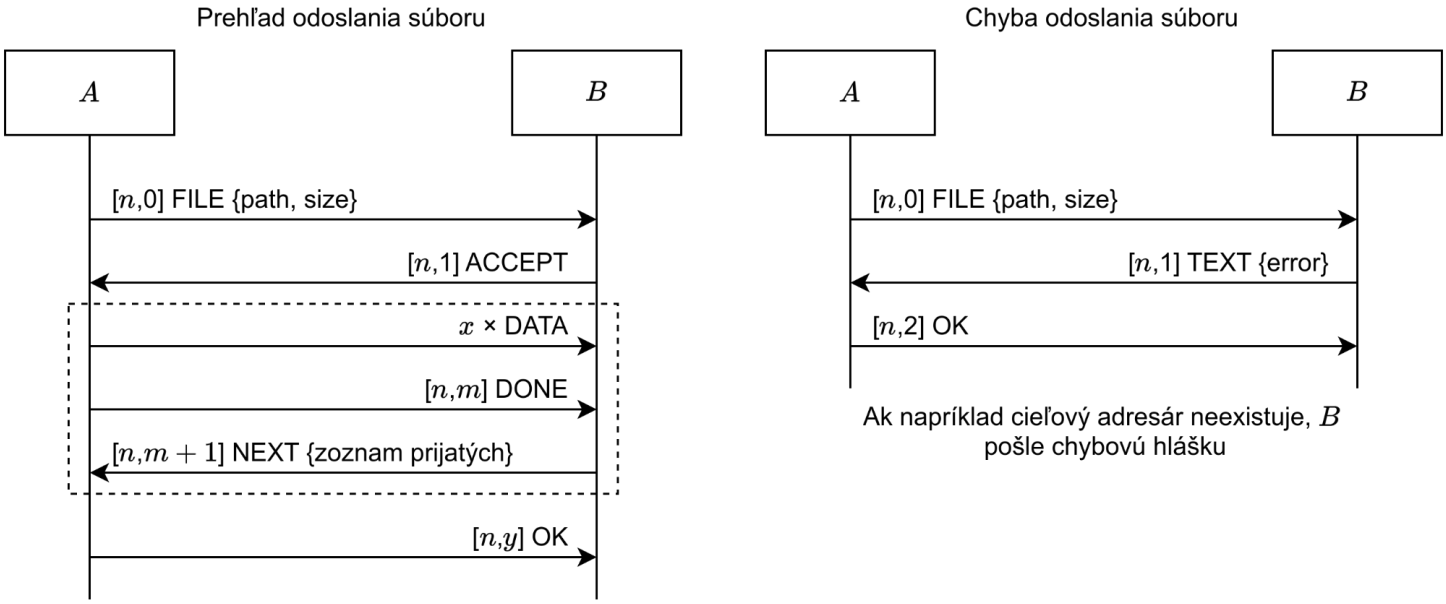
- Použitý algorimus CRC32 v štandardnej knižnici Pythonu
- Najprv sa vytvorí v pamäti cely segment - hodnota bude 0
- Vypočíta sa CRC32 a je vložené do hlavičky
- Na strane prijímateľa je hodnota uložená do dočasnej premennej a v hlavičke vynulovaná
- Vypočíta sa CRC32 a porovná s dočasnou premennou

Hlavička protokolu

- Spoločné:
  - Typ segmentu (1 bajt)
  - Dĺžka dát (2 bajty) ⇒ v bajtoch, dĺžka hlavičky je konštanta pre type segmentu
  - ID prúdu (4 bajty)
  - Kontrolná suma (4 bajty) ⇒ algoritmus CRC32 v štandardnej knižnici Pythonu
- Správy
  - ID správy (2 bajty)
  - Obsah (n bajtov)
- Fragmenty
  - Číslo fragmentu (2 bajty)
  - Dáta (n bajtov)

Odosielanie súboru

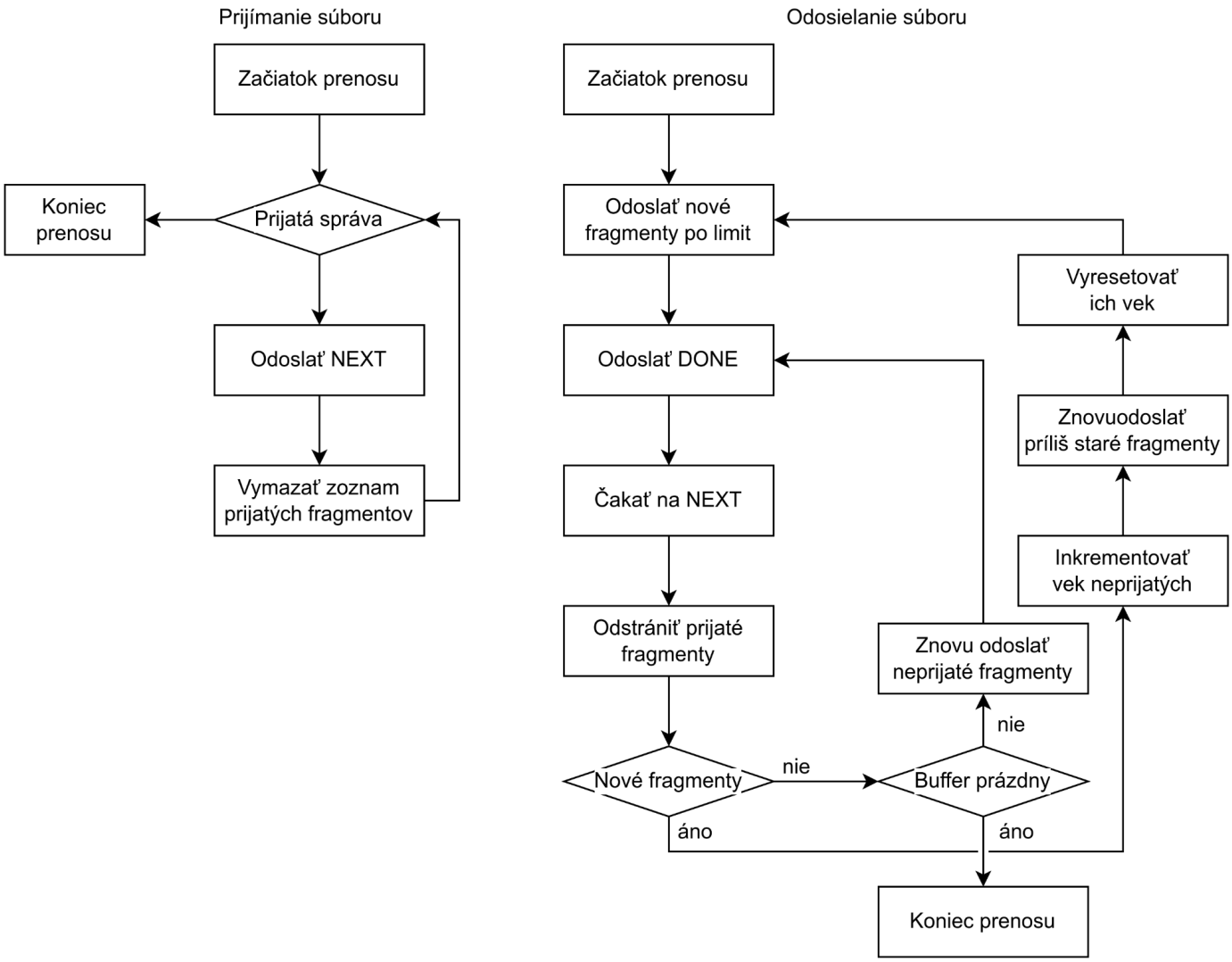
- Program plní úlohy vysielača a prijímača súčasne
- Každý používateľ môže poslať správu FILE na inicializáciu prenosu



- Odosielanie fragmentov prebieha v cykloch
- Odosielateľ si drží buffer odoslaných fragmentov veľkosti WINDOW_SIZE
- Keď je dosiahnutý limit bufferu odošle správu DONE
- Prijímateľ odpovedá správou NEXT zo zoznamom prijatých fragmentov
- Odosielateľ odstráni prijaté fragmenty z bufferu a odošle ďalšie až po WINDOW_SIZE
- Všetky neprijaté fragmenty majú inkrementovaný age
- Ak niektoré dosiahnu FRAGMENT_MAX_AGE sú znova odoslané a vek je resetovaný
- Ak už nie sú nové fragmenty na odoslanie, sú neprijaté fragmenty odoslané ihneď
- Odosielateľ odošle správu DONE a opakuje sa kým nie sú všetky fragmenty odoslané

Zaručenie poradia fragmentov

- Prijaté fragmenty sa ukladajú do prioritnej fronty podľa ich offsetu
- Počítadlo začína na nule
- Čaká sa kým nebude na začiatku fronty fragment offsetu rovného počítadla
- Packet je odstránený z fronty a zapísaný do súboru
- Počítadlo je inkrementované
- Ak je na začiatku fronty fragment menšieho offsetu ako počítadlo, je vyradený



Program

Program bol napísaný pre Python 3.10.12. Program nevyžaduje žiadne neštandardné knižnice.

K programu je tiež priložený súbor `wireshark.lua`, pre Wireshark podporu tohto protokolu.

Používateľské rozhranie

Program mení svoju rolu podľa konzolových parametrov:

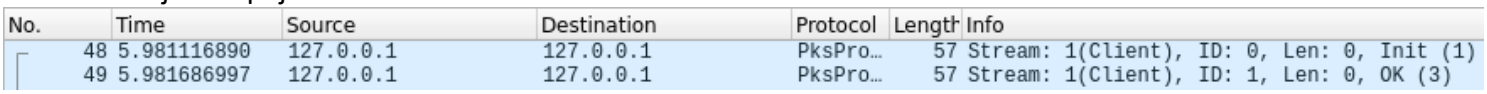
- server <port> ⇒ počúva na porte na pripojenie klienta
- client <ip> <port> ⇒ pripojí sa na server zo špecifikovanov IP adresov a portom

Po pripojení program prijíma vstup zo štandardného vstupu. Akýkoľvek vložený text je odoslaný ako textová správa okrem nasledujúcich príkazov:

- FIN ⇒ ukončí spojenie
- FILE <názov súboru>, <cesta kam sa má súbor uložiť> ⇒ odošle súbor
- SIZE <veľkosť> ⇒ nastaví maximálnu veľkosť fragmentu pri odosielaní súboru
- LOSS <počet> ⇒ pri ďalšom odosielaní súboru bude počet segmentov poškodených tak aby boli vyradené

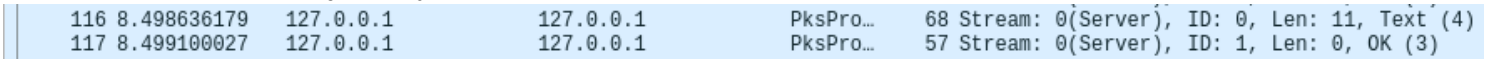
Testovací scenár

- Zahájenie spojenia



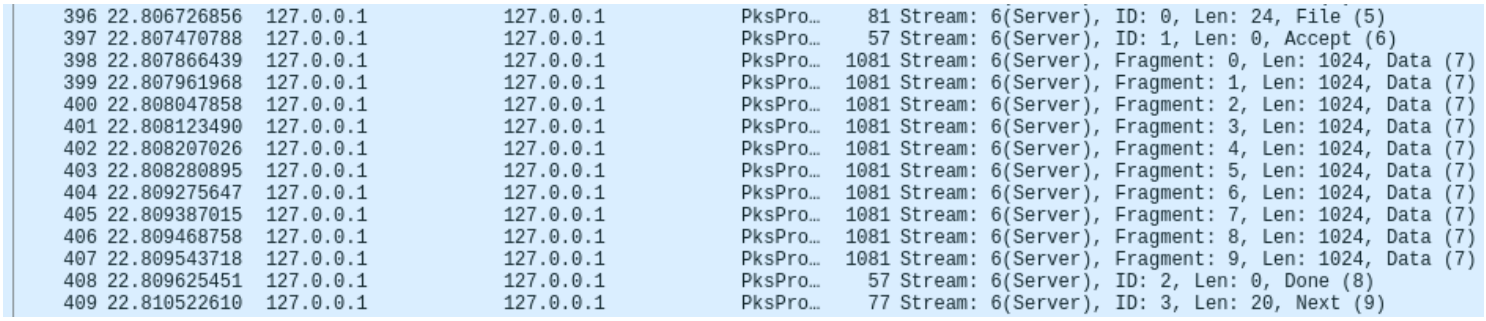
No.	Time	Source	Destination	Protocol	Length	Info
48	5.981116890	127.0.0.1	127.0.0.1	PksPro…	57	Stream: 1(Client), ID: 0, Len: 0, Init (1)
49	5.981686997	127.0.0.1	127.0.0.1	PksPro…	57	Stream: 1(Client), ID: 1, Len: 0, OK (3)

- Odoslanie textovej správy



116	8.498636179	127.0.0.1	127.0.0.1	PksPro…	68	Stream: 0(Server), ID: 0, Len: 11, Text (4)
117	8.499100027	127.0.0.1	127.0.0.1	PksPro…	57	Stream: 0(Server), ID: 1, Len: 0, OK (3)

- Odoslanie súboru

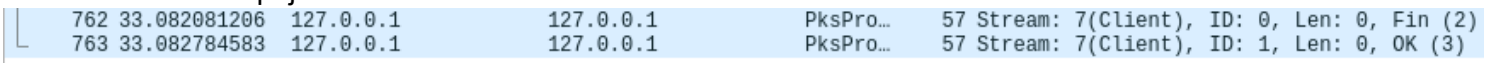


396	22.806726856	127.0.0.1	127.0.0.1	PksPro…	81	Stream: 6(Server), ID: 0, Len: 24, File (5)
397	22.807470788	127.0.0.1	127.0.0.1	PksPro…	57	Stream: 6(Server), ID: 1, Len: 0, Accept (6)
398	22.807866439	127.0.0.1	127.0.0.1	PksPro…	1081	Stream: 6(Server), Fragment: 0, Len: 1024, Data (7)
399	22.807961968	127.0.0.1	127.0.0.1	PksPro…	1081	Stream: 6(Server), Fragment: 1, Len: 1024, Data (7)
400	22.808047858	127.0.0.1	127.0.0.1	PksPro…	1081	Stream: 6(Server), Fragment: 2, Len: 1024, Data (7)
401	22.808123490	127.0.0.1	127.0.0.1	PksPro…	1081	Stream: 6(Server), Fragment: 3, Len: 1024, Data (7)
402	22.808207026	127.0.0.1	127.0.0.1	PksPro…	1081	Stream: 6(Server), Fragment: 4, Len: 1024, Data (7)
403	22.808280895	127.0.0.1	127.0.0.1	PksPro…	1081	Stream: 6(Server), Fragment: 5, Len: 1024, Data (7)
404	22.809275647	127.0.0.1	127.0.0.1	PksPro…	1081	Stream: 6(Server), Fragment: 6, Len: 1024, Data (7)
405	22.809387015	127.0.0.1	127.0.0.1	PksPro…	1081	Stream: 6(Server), Fragment: 7, Len: 1024, Data (7)
406	22.809468758	127.0.0.1	127.0.0.1	PksPro…	1081	Stream: 6(Server), Fragment: 8, Len: 1024, Data (7)
407	22.809543718	127.0.0.1	127.0.0.1	PksPro…	1081	Stream: 6(Server), Fragment: 9, Len: 1024, Data (7)
408	22.809625451	127.0.0.1	127.0.0.1	PksPro…	57	Stream: 6(Server), ID: 2, Len: 0, Done (8)
409	22.810522610	127.0.0.1	127.0.0.1	PksPro…	77	Stream: 6(Server), ID: 3, Len: 20, Next (9)

[…]

586	22.862452888	127.0.0.1	127.0.0.1	PksPro…	1081	Stream: 6(Server), Fragment: 156, Len: 1024, Data (7)
587	22.862506649	127.0.0.1	127.0.0.1	PksPro…	1081	Stream: 6(Server), Fragment: 157, Len: 1024, Data (7)
588	22.862558826	127.0.0.1	127.0.0.1	PksPro…	1081	Stream: 6(Server), Fragment: 158, Len: 1024, Data (7)
589	22.862612947	127.0.0.1	127.0.0.1	PksPro…	106	Stream: 6(Server), Fragment: 159, Len: 49, Data (7)
590	22.862667099	127.0.0.1	127.0.0.1	PksPro…	57	Stream: 6(Server), ID: 32, Len: 0, Done (8)
591	22.863784540	127.0.0.1	127.0.0.1	PksPro…	77	Stream: 6(Server), ID: 33, Len: 20, Next (9)
592	22.865048997	127.0.0.1	127.0.0.1	PksPro…	57	Stream: 6(Server), ID: 34, Len: 0, OK (3)

- Ukončenie spojenia



762	33.082081206	127.0.0.1	127.0.0.1	PksPro…	57	Stream: 7(Client), ID: 0, Len: 0, Fin (2)
763	33.082784583	127.0.0.1	127.0.0.1	PksPro…	57	Stream: 7(Client), ID: 1, Len: 0, OK (3)

Záver

Program dokáže odosielať súbor zo zaručením prijatia každého segmentu. Odosielateľ prepošle len tie segmenty, ktoré neboli prijaté (Selective Repeat). Jedna správa o prijatých segmentoch môže obsahovať čísla viacerých fragmentov, takže sa neposiela pre každý prijatý segment.

Pri udržiavaní komunikácie sa PING správa odošle len vtedy ak nie je za určitú dobu, detegovaná žiadna komunikácia, takže pri aktívnej komunikácií (napr. posielanie text. správy alebo súboru) nie je potrebné tieto správy odosielať. Ak je správa PING stratená pri prenose, je znovu odoslandá do určitého limitu, takže exituje nebude spojenie ukončené ihneď pri jednej chýbajúcej odpovedi.