Slovenská technická univerzita v Bratislave

Fakulta informatiky a informačných technológií

Počítačové a komunikačné siete  
Komunikácia s využitím UDP protokolu

1. Zadanie

Navrhnite a implementujte program s použitím vlastného protokolu nad protokolom UDP (User Datagram Protocol) transportnej vrstvy sieťového modelu TCP/IP. Program umožní komunikáciu dvoch účastníkov v lokálnej sieti Ethernet, teda prenos textových správ a ľubovoľného súboru medzi počítačmi (uzlami). Program bude pozostávať z dvoch častí – vysielacej a prijímacej. Vysielací uzol pošle súbor inému uzlu v sieti. Predpokladá sa, že v sieti dochádza k stratám dát. Ak je posielaný súbor väčší, ako používateľom definovaná max. veľkosť fragmentu, vysielajúca strana rozloží súbor na menšie časti - fragmenty, ktoré pošle samostatne. Maximálnu veľkosť fragmentu musí mať používateľ možnosť nastaviť takú, aby neboli znova fragmentované na linkovej vrstve. Ak je súbor poslaný ako postupnosť fragmentov, cieľový uzol vypíše správu o prijatí fragmentu s jeho poradím a či bol prenesený bez chýb. Po prijatí celého súboru na cieľovom uzle tento zobrazí správu o jeho prijatí a absolútnu cestu, kam bol prijatý súbor uložený. Program musí obsahovať kontrolu chýb pri komunikácii a znovu vyžiadanie chybných fragmentov, vrátane pozitívneho aj negatívneho potvrdenia. Po zapnutí programu, komunikátor automaticky odosiela paket pre udržanie spojenia každých 5s pokiaľ používateľ neukončí spojenie ručne. Odporúčame riešiť cez vlastne definované signalizačné správy a samostatný thread.

1. Návrh hlavičky

CRC

Odosielané dáta

Poradie fragmentu

Veľkosť fragmentu

Identifikátor správy

* 1. Identifikátor správy (1B)

Určuje typ odosielanej správy :

* 1 - inicializácia spojenia
* 2 - zmena strán (vysielač - prijímač)
* 3 - odosielanie *textovej* správy
* 4 - odosielanie *súboru*
* 5 - odpoveď - úspešná komunikácia
* 6 - odpoveď - neúspešná komunikácia
* 7 - keep alive správa
  1. Veľkosť fragmentu (2B)

Určuje veľkosť odosielaného fragmentu pri odosielaní textovej správy alebo súboru

* 1. Poradie fragmentu (2B)

Určuje poradie odosielaného fragmentu pri odosielaní textovej správy alebo súboru

* 1. Odosielané dáta

Dáta odosielané v správe

* 1. CRC (4B)

Kontrolný súčet na overenie správnosti odosielaných dát

1. Opis programu

Program bude vyvíjaný v jazyku python s využitím knižnice socket. Na začiatku programu sa spustí hlavné menu programu s 3 možnosťami : Vysielač, Prijímač a Ukončenie programu. Program je organizovaný do 4 súborov - triedy reciever.py, sender.py a keep\_alive.py a hlavné menu programu main.py

Prijímač

Pri inicializácií prijímača program čaká na inicializačnú správu (typ správy = 1) od vysielača. Po prijatí tejto správy odpovie inicializačnou odpoveďou (typ správy = 1) a komunikácia je spustená. Program čaká na signál o inicializácií správy (typ správy = 2). Podľa inicializačnej správy vie koľko rámcov má očakávať. Postupne prijíma všetky rámce, kontroluje ich a zapisuje si chybné rámce. Po prijatí všetkých rámcov odošle vysielaču zoznam chybných rámcov alebo správu o úspešnej komunikácií.

Vysielač

Pri inicializácií vysielača pošle vysielač inicializačnú správu (typ správy = 1) a následne čaká na inicializačnú odpoveď od Prijímača. Po prijatí správy je komunikácia spustená a môžeme začať odosielať správy. Na výber máme odosielanie súboru alebo textovej správy. V oboch prípadoch začneme odoslaním správy a inicializácií komunikácie (typ správy = 2). Používateľ si môže nastaviť veľkosť fragmentu v prípade že správu treba fragmentovať. Používateľ si tiež môže zvoliť simuláciu chyby odosielania. V takom prípade je pred odoslaním zmenené vypočítané CRC tak, aby jeho hodnota nesedela pri porovnaní na strane prijímača. Následne vysielač postupne odošle všetky rámce a počká na odpoveď od prijímača. Ak nastala chyba pri prenose rámcov, vysielač odošle chybné rámce na základe požiadavky prijímača. Tento typ ARQ sa nazýva Selective Repeat

Keep Alive

Na funkcionalitu udržiavania spojenia využívam odosielanie keep alive správ vo vlastnom Thread-e. Vysielač odosiela každých 5 sekúnd keep alive správu a pokiaľ nedostane 20 sekúnd odpoveď, ukončí spojenie.

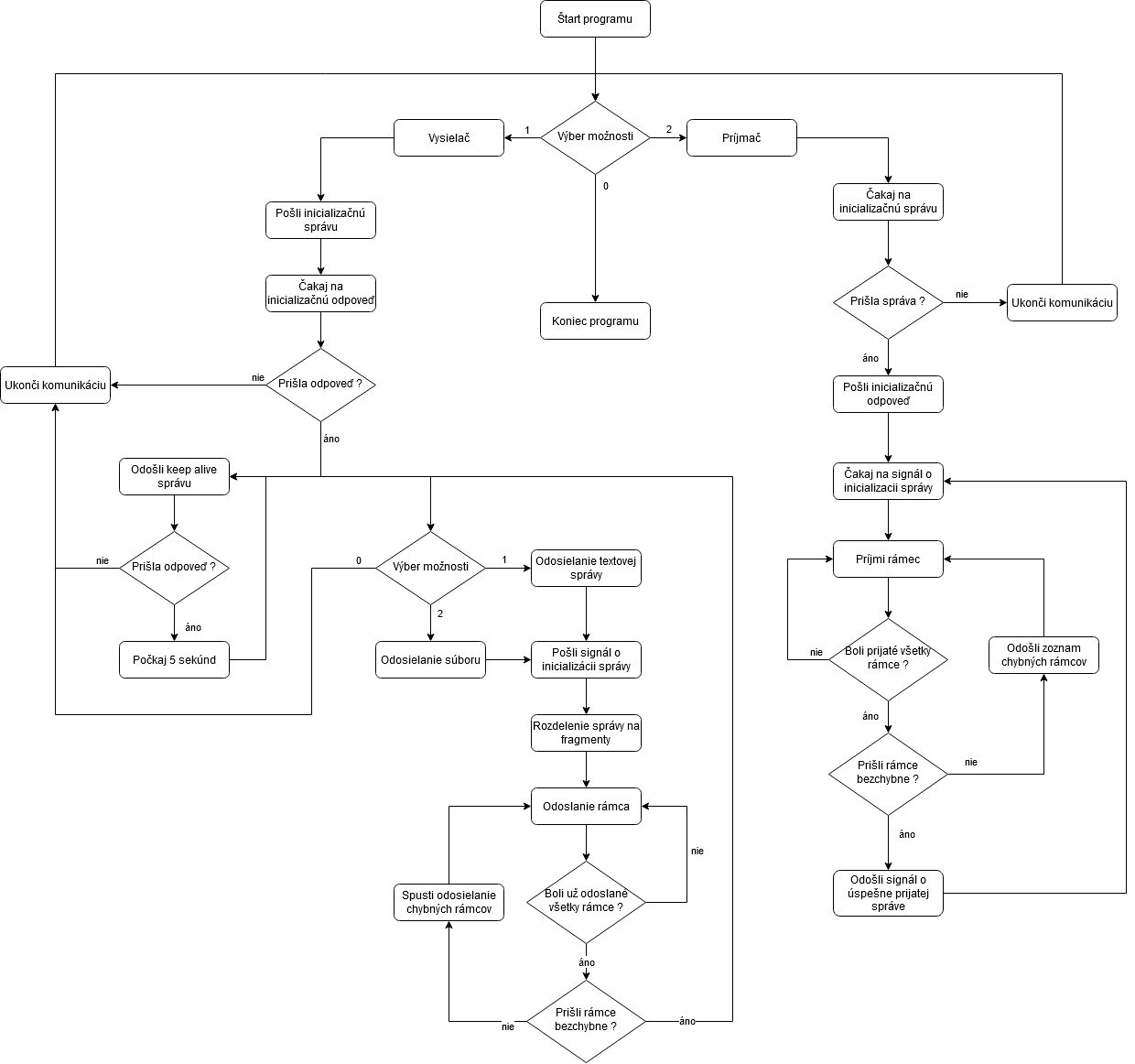
CRC

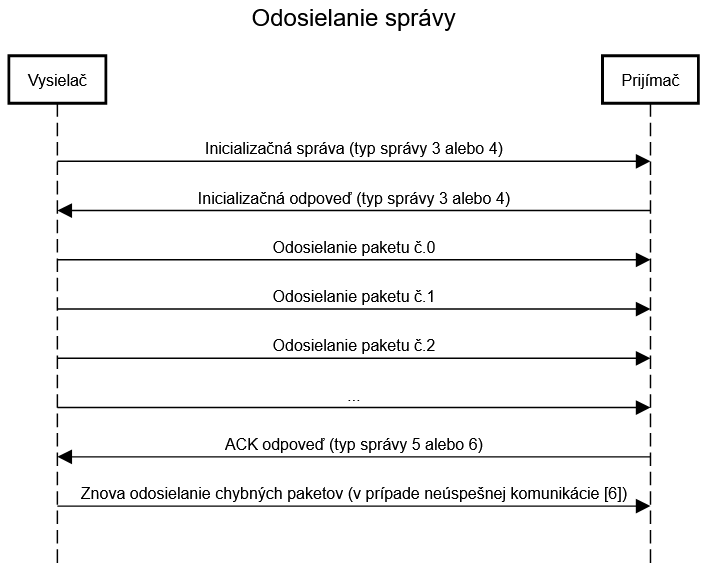
Ako kontrolný súčet využívam CRC-32 s knižnice binascii, CRC je vypočítané nasledujúcim polynómom:

Obrázok, na ktorom je text

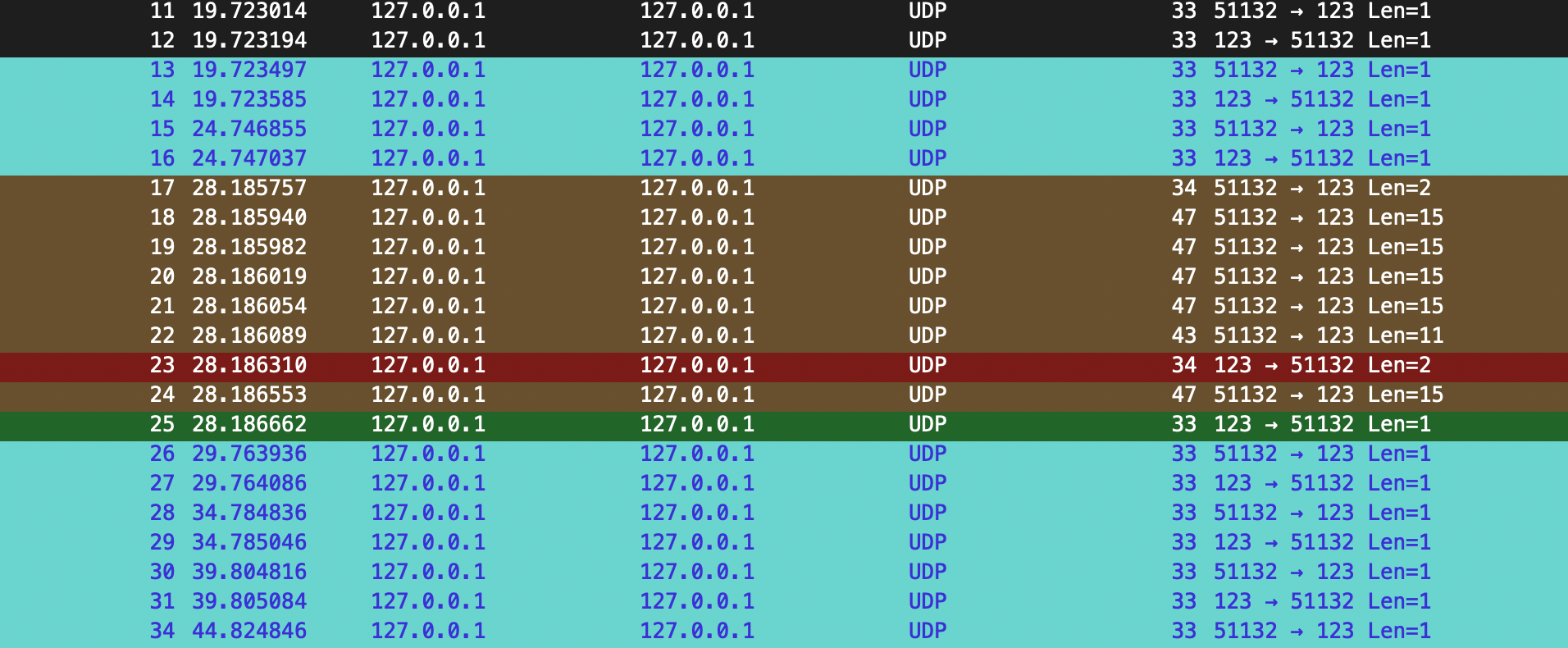
Automaticky generovaný popisx32 + x26 + x23 + x22 + x16 + x12 + x11 + x10 + x8 + x7 + x5 + x4 + x2 + x1 + 1

*Pseudokód algoritmu CRC-32 prevzatý z https://en.wikipedia.org/wiki/Cyclic\_redundancy\_check#CRCs\_and\_data\_integrity*

1. Bloková schéma programu
2. Sekvenčný diagram komunikácie



1. Príklad odosielania správy zachytený vo wiresharku



Prvé dve správy sú inicializácia spojenia

Nasledujú 2 dvojice keep alive správ

Inicializácia odosielania a 5 packetov odosielanej správy

ACK odpoveď – chybný packet

Znova odoslanie packetu

ACK odpoveď – úspešná komunikácia

Ďalšie dvojice keep alive správ

1. Zmeny oproti návrhu

* Typ správy 2 - inicializácia správy bola odstránená a namiesto nej je typ správy 2 využívaný na zmenu strán (rolí) programu.
* Pridaný sekvenčný diagram odosielania správ.
* Lepšie opísaný Keep Alive a CRC