Návrh riešenia pre zadanie č. 2

Počítačové a komunikačné siete

2023 – 24

# Programovací jazyk a prostredie

Svoj protokol budem programovať v programovacom jazyku Python 3.11, v programe JetBrains PyCharm 2023.2.3 s použitím knižníc socket, threading , time a crcmod.

# Návrh projektu

V mojom protokole budú dva typy paketov: riadiace a prenosné – každý typ bude mať vlastnú hlavičku podľa potrieb komunikácie. Úlohou riadiacich paketov bude sprostredkovať komunikáciu medzi uzlami na začiatku spojenia aj v priebehu, čiže medzi prenosmi dát. Prenosné pakety budú slúžiť výlučne na prenos dát. Úlohy odosielateľa a prijímateľa budú rozdelené do dvoch samostatných zdrojových kódov sender.py a receiver.py a komunikácia medzi dvoma uzlami bude bežať v main.py, kde budú vytvorené dva objekty, „uzly“, ktoré budú implementovať funkcie z oboch vedľajších súborov.

Na začiatku si používateľ zvolí či chce byť odosielateľ alebo prijímateľ. Ak si používateľ zvolí úlohu odosielateľa, program si od neho vypýta číslo portu a IP adresu prijímateľa, s ktorým sa chce spojiť a prebehne nadviazanie spojenia. Po nadviazaní spojenia sa automaticky spustí vlákno pre udržiavanie spojenia – „keep alive“ a spustí sa aj kontrolné vlákno, cez ktoré si môže zvoliť čo chce robiť:

1. Odoslať správu
2. Odoslať súbor
3. Zmeniť úlohu na prijímateľa
4. Nerobiť nič

Tieto rozhodnutia sa budú vykonávať pomocou riadiacich paketov. Ak sa rozhodne poslať súbor alebo správu, vlákno udržiavania spojenia sa skončí a po skončení odosielania sa znovu automaticky zapne. Odosielanie dát budú realizovať prenosné pakety.

Ak sa používateľ rozhodne, že chce byť prijímateľ, program si vypýta číslo portu, na ktorom bude počúvať prichádzajúce správy. Po obdržaní riadiaceho paketu, so žiadosťou o nadviazanie spojenia automaticky odošle riadiaci paket s potvrdením a spojenie je nadviazané. Následne čaká na správy od odosielateľa.

V oboch prípadoch môže používateľ odoslať žiadosť o zmenu úlohy (odosielateľ / prijímateľ). Túto žiadosť môže samozrejme poslať iba vtedy, keď neprebieha prenos správ alebo súborov – čiže keď beží len udržiavanie spojenia. Po odoslaní žiadosti čaká na potvrdenie od druhej strany a po obdržaní potvrdenia si role prehodia.

# Návrh hlavičky – riadiace pakety

|  |
| --- |
| TYP  (1B) |

Riadiace pakety budú vyzerať nasledovne:

Žiadne ďalšie informácie pri nich nebudú potrebné.

Typy paketov budú podľa čísel označené nasledovne:

1. Paket pre udržiavanie spojenia – „keep alive“
2. Paket pre nadviazanie spojenia na začiatku komunikácie
3. Paket s potvrdením – „ACK“
4. Paket s negatívnym potvrdením – „NACK“
5. Paket so žiadosťou o odoslanie správy
6. Paket so žiadosťou o odoslanie súboru
7. Paket s informáciou o prijatí všetkých dát
8. Paket so žiadosťou o zmenu úloh
9. Paket so žiadosťou o koniec komunikácie

# Návrh hlavičky – prenosné pakety

Prenosné pakety budú podľa môjho protokolu vyzerať nasledovne:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Veľkosť fragmentu (2B) | Počet fragmentov (2B) | Poradie fragmentu (2B) | Dáta | CRC  (2B) |

Veľkosť fragmentu – veľkosť odosielaného fragmentu

Počet fragmentov – celkový počet fragmentov správy

Poradie fragmentu – poradie odosielaného paketu

CRC – kontrolný údaj pre stanovenie správnosti prenášaných dát

Veľkosť mojej hlavičky je 8B, veľkosť UDP hlavičky je 8B a veľkosť IP hlavičky je 20B, preto maximálna veľkosť fragmentu, ktorý môžem odoslať je 65 000 – 36 = 64 964 Bajtov.

# Metóda ARQ

V mojom programe budem používať ARQ metódu „Go Back-N“. Pointa tejto metódy spočíva v tom, že odosielateľ rozdeľuje dáta na pevne definované bloky, nazývané okná o veľkosti N. Každý blok je odoslaný bez čakania na potvrdenie, až kým nie je celé okno naplnené. Prijímateľ potvrdzuje prijaté dáta pomocou riadiacich paketov typu 2 – „ACK“ alebo v prípade korupcie prijatých dát pomocou riadiacich paketov typu 3 – „NACK“.

V prípade, že odosielateľ dostane negatívnu odpoveď, alebo v stanovenom čase nedostane kladnú odpoveď, odosielateľ opakuje odoslanie všetkých dát od posledného potvrdeného bodu. Prijímateľ zahadzuje všetky prichádzajúce bloky okrem požadovaného bloku.

Túto metódu som si zvolil, lebo zistenie a oprava chyby prenosu paketu je pri nej pomerne jednoduchá. Jej nevýhodou však je, že v prípade chyby prenosu môže dôjsť k neefektívnemu opakovanému odosielaniu dát.

# Metóda Checksum

Na kontrolu integrity prijatého paketu bude môj protokol používať CRC algoritmus.

Vzhľadom na povolenie cvičiaceho budem používať 16-bitový kontrolný súčet pomocou knižnice crcmod s polynómom „0x8005“. Týmto polynómom sa vydelia dáta zakódované do reťazca Bajtov a zvyšok po tomto delení sa zakóduje do reťazca o veľkosti dvoch Bajtov a pridá sa na koniec reťazca dát. Takto zakódované dáta potom prijímateľ znovu vydelí stanoveným polynómom a ak je výsledok tohto delenia 0, dáta boli prenesené správne. Ak je výsledok hocijaký iný, nastala korupcia dát pri prenose.

Túto metódu som si vybral pretože vie veľmi spoľahlivo zachytávať chyby v pakete a pridáva iba malé množstvo informácii k paketu.

# „Keep alive“ metóda

Metóda udržania spojenia medzi komunikujúcimi uzlami vyplýva zo zadania. Odosielateľ musí každých 5 sekúnd odoslať riadiaci paket typu 0. Po obdržaní takéhoto paketu prijímateľ automaticky posiela riadiaci paket typu 2 – „ACK“. Obidva uzly majú v svojom kóde implementovaný časovač vypršania spojenia, ktorý sa úspešným prijatím takýchto paketov zakaždým resetuje. Časovač je nastavený na 16 sekúnd. Ak časovač vyprší, spojenie sa terminuje a pre ďalšiu komunikáciu je potrebné ho znovu nadviazať.

# A black background with white rectangular shapes Description automatically generatedDiagram spracovávania komunikácie