PKS - semestrálna práca

Dokumentácia

HLAVIČKA PROTOKOLU

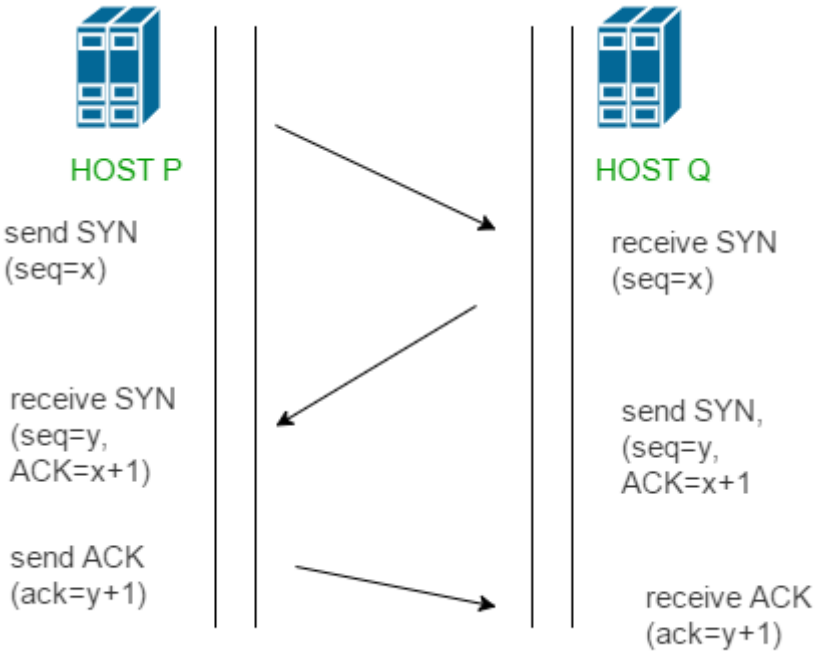
Pole	**Dĺžka BYTE**	**Popis**
Typ spravy (FLAGS)	1	Typ správy (napr. SYN, ACK, DATA, KEEP ALIVE)
SourcePort	2	Zdrojový port
DestinationPort	2	Cieľový port
Sequence Number	2	Sekvenčné číslo pre udržanie poradia správ a fragmentov
Acknowledgment Number	2	Potvrdenie posledneho packetu
Fragment offset	2	Identifikátor fragmentu
Length	2	Dĺžka dát (data field)
Checksum	2	Kontrolný súčet CRC 16
Data	-	Samotná správa, napr. "Hello World"

NADVIAZANIE SPOJENIA

Nadviazanie spojenia medzi dvoma uzlami sa vykonáva pomocou **three-way handshake** podobného TCP, ale implementovaného nad UDP. Tento proces zahŕňa tri správy:

- **SYN**: Prvá správa iniciátora spojenia s nastaveným príznakom **0x01** v poli Flags a náhodne vygenerovaným sekvenčným číslom.
- **SYN-ACK**: Odozva prijímateľa, ktorý potvrdzuje prijatie SYN správy s nastaveným príznakom **0x02**. Obsahuje sekvenčné číslo a potvrdenie **Acknowledgment Number**.
- **ACK**: Finálna správa iniciátora spojenia s príznakom **0x04**, ktorá potvrdzuje nadviazanie spojenia. Obsahuje aktualizované sekvenčné a potvrdené číslo.

Po prijatí **ACK** správy sú oba uzly pripravené na výmenu dát a spojenie je nadviazané. Sekvenčné čísla zabezpečujú správny poriadok správ a eliminujú duplicity.



Po nadviazaní spojenia môžu uzly začať posielať dáta.

POSIELANIE DAT PO FRAGMENTOCH

Nastavenie parametrov fragmentácie pri handshaku

Predtým, ako sa začnú prenášať samotné dáta, uzly (komunikujúce strany) si musia dohodnúť parametre spojenia počas **handshaku**. Okrem štandardných krokov handshaku (napr. odoslanie správ **SYN**, **SYN-ACK** a **ACK**) sa počas tohto procesu môžu dohodnúť aj parametre fragmentácie.

Jedným z kľúčových parametrov, ktoré si uzly môžu dohodnúť, je **maximálna veľkosť jedného fragmentu**

Proces fragmentácie dát

1. Rozdelenie dát na fragmenty

Ak odosielateľ plánuje poslať správu alebo súbor, ktorý presahuje limit veľkosti jedného fragmentu, musí tieto dáta rozdeliť na menšie časti – **fragmenty**. Každý fragment obsahuje časť pôvodných dát spolu s hlavičkou, ktorá obsahuje potrebné informácie o fragmente.

Hlavička fragmentu obsahuje:

- **Typ správy (Flags)**: Udáva, či ide o typ správy ako **SYN**, **ACK**, **DATA**, **KEEP ALIVE**.
- **Zdrojový port (Source Port)**: Port odosielajúceho uzla.

- **Cieľový port (Destination Port)**: Port cieľového uzla.
- **Sekvenčné číslo (Sequence Number)**: Číslo na udržanie poradia správ a fragmentov.
- **Potvrdenie (Acknowledgment Number)**: Potvrdenie prijatia posledného fragmentu alebo správy.
- **Offset fragmentu (Fragment Offset)**: Číslo fragmentu v rámci celej správy (napr. fragment 1, fragment 2, atď.).
- **Celkový počet fragmentov (Total Fragments)**: Informácia o tom, koľko fragmentov celkovo tvorí správu.
- **Dĺžka dát (Length)**: Dĺžka dátového fragmentu.
- **Kontrolný súčet (Checksum)**: Používa sa na overenie integrity fragmentu, aby sme sa uistili, že fragment nebol poškodený počas prenosu.
- **Dáta (Data)**: Samotná časť správy, ktorá bola rozdelená na fragmenty.

2. Odosielanie fragmentov

Odosielateľ po rozdelení dát na fragmenty začne každý fragment postupne posielať. Fragmenty sa odosielajú po poradí a každý z nich je zabalený spolu so svojou hlavičkou, ktorá poskytuje dôležité informácie prijímateľovi.

V tomto kroku odosielateľ sleduje, či prijímateľ prijal všetky fragmenty správne. Na tento účel sa používajú potvrdenia (ACK správy) od prijímateľa.

3. Prijímanie fragmentov a skladanie správy

Prijímateľ prijíma jednotlivé fragmenty jeden po druhom. Na základe informácií v hlavičke fragmentu (**Fragment ID**, **Total Fragments**) identifikuje poradie fragmentov a skontroluje, či fragmenty neboli počas prenosu poškodené (pomocou kontrolného súčtu).

Po prijatí všetkých fragmentov prijímateľ fragmenty zloží do pôvodnej správy alebo súboru.

KONTROLA INTEGRITY POSLANEJ SPRÁVY

Na kontrolu integrity správy používame **kontrolný súčet** (Checksum). Kontrolný súčet umožňuje overiť, či správa nebola počas prenosu poškodená. Pri výpočte kontrolného súčtu sa berú do úvahy všetky bajty správy, vrátane hlavičky a dát. Chcel by som si zvoliť CRC16-ANSI pretože má v tejto aplikácií lepšie využitie a zároveň menšiu zložitosť.

Odosielateľ

1. **Inicializácia CRC registra**:
   - Na začiatku odosielateľ nastaví CRC register na počiatočnú hodnotu, čo je bežne 0xFFFF.
2. **Spracovanie dát**:
   - Odosielateľ spracováva všetky bajty správy vrátane hlavičky a dát:
     - Každý bajt dát sa XORuje s horným bajtom aktuálneho CRC registra.
     - Následne pre každý bajt prebehne 8-bitová iterácia (bit po bite):
       - Ak najvyšší bit v CRC registri je 1, odosielateľ posunie CRC doľava o 1 bit a XORuje ho s polynómom 0x8005.

        - Ak najvyšší bit v CRC nie je 1, posunie CRC doľava bez XOR operácie.
3. **Opakovanie pre všetky bajty**:
    - Tento proces sa opakuje pre každý bajt správy (hlavičky aj dát).
4. **Finálna úprava**:
    - Po spracovaní všetkých bajtov odosielateľ finálne XORuje výsledný CRC s 0x0000. V niektorých aplikáciách sa môže používať XOR s 0xFFFF na inverziu výsledného CRC.
5. **Pridanie CRC k správe**:
    - Odosielateľ následne pripojí vypočítaný CRC16-ANSI kontrolný súčet k správe. Tento CRC je súčasťou hlavičky správy alebo je pripojený ako posledná časť správy.
6. **Odoslanie správy**:
    - Celá správa, vrátane dát, hlavičky a vypočítaného kontrolného súčtu, je odoslaná prijímateľovi.

Prijímateľ

1. **Prijatie správy**:
    - Prijímateľ obdrží celú správu vrátane hlavičky, dát a kontrolného súčtu (CRC).
2. **Oddelenie kontrolného súčtu**:
    - Prijímateľ oddelí pripojený CRC kontrolný súčet od zvyšku správy.
3. **Výpočet CRC z prijatých dát**:
    - Prijímateľ opakuje ten istý proces výpočtu CRC, ako robil odosielateľ:
        - Inicializuje CRC register s počiatočnou hodnotou 0xFFFF.
        - XORuje prijaté dáta vrátane hlavičky a dát.
        - Pre každý bajt opakuje proces XORovania a posúvania bitov s polynómom 0x8005.
        - Tento proces sa opakuje pre všetky prijaté bajty správy (okrem oddeleného CRC).
4. **Porovnanie výsledku**:
    - Po vypočítaní CRC z prijatých dát prijímateľ porovná tento výsledok s CRC, ktorý bol pripojený k správe od odosielateľa.
    - Ak sa výsledné CRC rovná prijatému CRC, správa nebola poškodená a je považovaná za validnú.
    - Ak sa výsledné CRC nezhoduje s prijatým CRC, znamená to, že správa bola počas prenosu poškodená.
5. **Akcia na základe výsledku**:
    - Ak je kontrola úspešná (CRC sa zhoduje), prijímateľ potvrdí prijatie správy (napr. odoslaním **ACK** správy odosielateľovi).

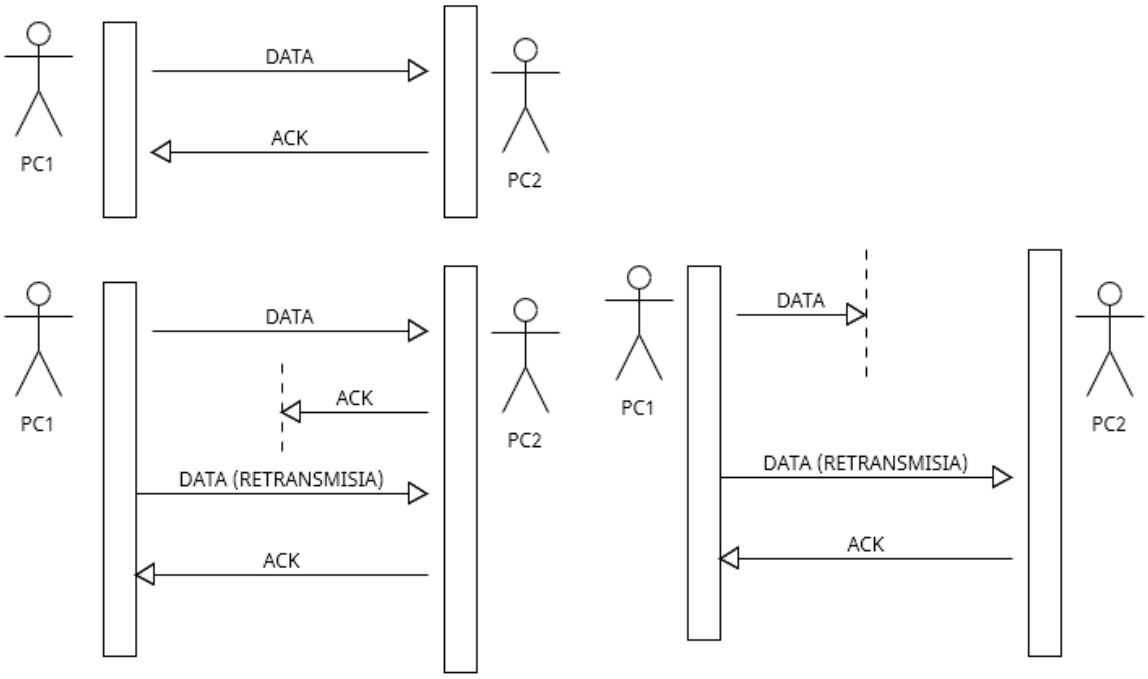
ARQ

ARQ je základným mechanizmom, ktorý umožňuje spoľahlivý prenos dát nad UDP. V našom prípade požijeme **Stop-and-Wait ARQ**, pretože je najjednoduchší na implementáciu.

Stop-and-Wait ARQ

- **Princíp**: Po odoslaní správy odosielateľ čaká na ACK od prijímateľa, kým neodošle ďalšiu správu. Ak ACK nepríde, pošle tú istú správu znova.
- **Výhody**: Jednoduchý a ľahko implementovateľný mechanizmus.

- **Nevýhody**: Neefektívne využitie prenosovej kapacity, pretože vždy čaká na ACK pred odoslaním ďalšej správy.



KEEP-ALIVE

Metóda **Keep-Alive** sa používa na zabezpečenie toho, že spojenie medzi dvoma uzlami zostane aktívne a funkčné, aj keď medzi nimi momentálne neprebieha žiadna dátová komunikácia.

Periodické posielanie Keep-Alive správ:

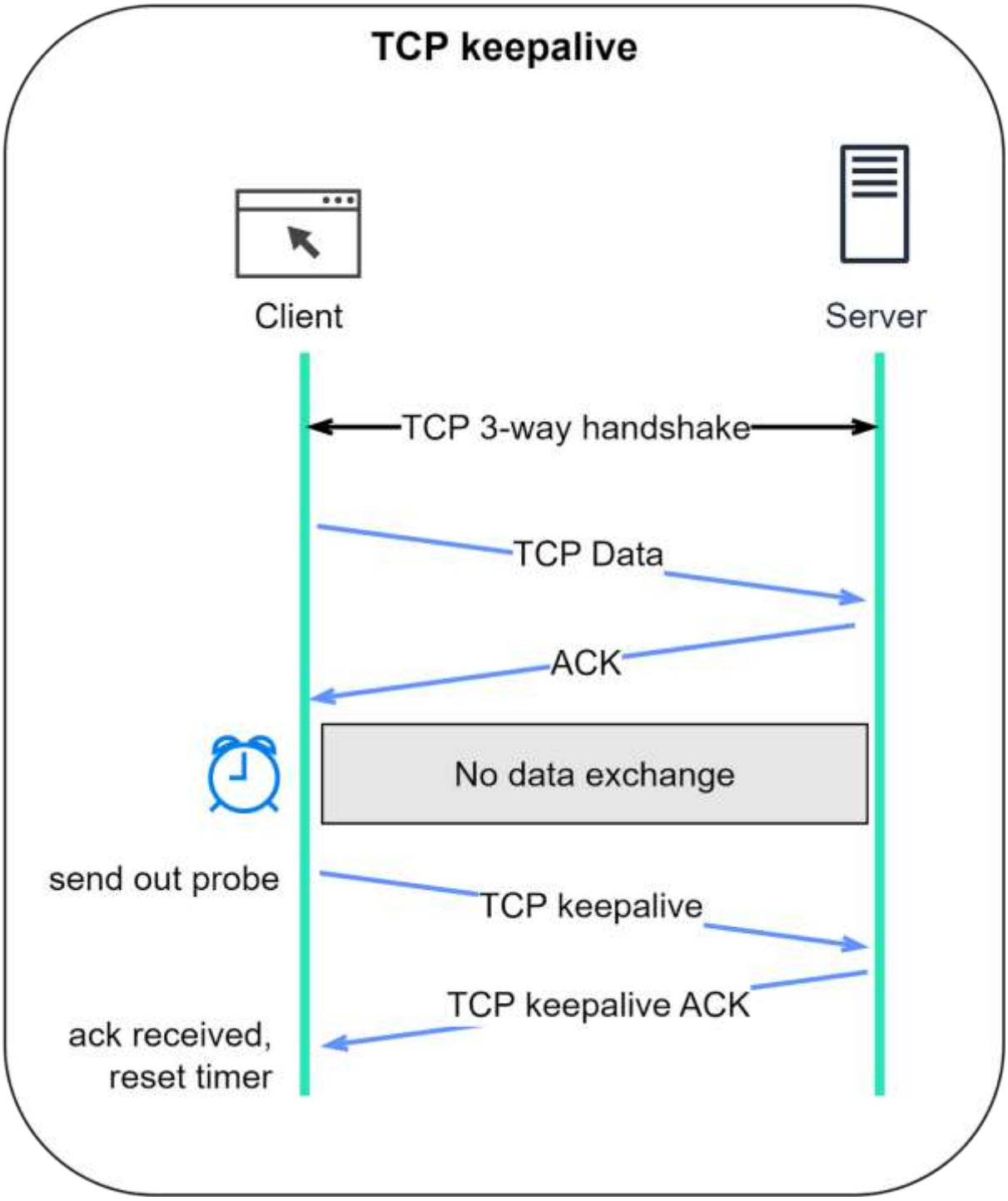
- Uzol A posiela každých X sekúnd správu typu "HEARTBEAT" (prípadne iný špecifický typ Keep-Alive správy).
- Uzol B prijme túto správu a odošle ACK ako potvrdenie, že je stále aktívny.

Časovač pre detekciu neaktívnosti:

- Každý uzol má časovač, ktorý začne po odoslaní Keep-Alive správy bežať. Ak nedostane ACK v stanovenom časovom limite, uzol predpokladá, že druhý uzol je neaktívny.
- Časovač sa resetuje, ak ACK dorazí včas.

Reakcia na neaktívny uzol:

- Ak ACK nepríde ani po niekoľkých pokusoch (napr. troch), uzol môže predpokladať, že druhý uzol sa odpojil, a môže uzavrieť spojenie alebo zobraziť používateľovi chybu.



SIMULOVANIE CHYB

Zmena kontrolného výpočtu CRC:

Odosielateľ môže správne vypočítať CRC pre správu, ale potom ho úmyselne zmeniť pred pripojením k správe. Prijímateľ, ktorý spracuje dáta a vypočíta CRC, zistí nesúlad a odmietne správu.

Príklad scenára: Odosielateľ vypočíta CRC ako 0x1234, ale pred odoslaním zmení tento CRC na 0x5678. Keď prijímateľ vypočíta CRC z dát, jeho výsledok (napr. 0x1234) sa nebude zhodovať so zaslaným CRC 0x5678, čo indikuje chybu.

Manipulácia bitov:

Teoreticky môžeme simulovať chybu náhodnou manipuláciou bitov v správe. Vyberieme náhodný bajt v dátach a náhodne invertujeme jeden z jeho bitov (t.j. ak je 1, zmeníme ho na 0 a naopak). Tým sa naruší integrita správy a prijímateľ zaznamená nesúlad pri výpočte CRC.

Prípadný scenár: V správe "Hello World" vyberieme bajt zodpovedajúci písmenu "H" (binárne 01001000), náhodne zmeníme jeden bit, napr. na 01000000. Prijímateľ dostane nesprávnu hodnotu, čo spôsobí chybu pri kontrole CRC.