Az adatkapcsolati réteg

Programtervező informatikus BSc
Számítógép hálózatok és architektúrák
előadás

Az adatkapcsolati réteg

- A fizikai átviteli hibáinak elfedése a hálózati réteg elől
- Keretezés
  - Adatfolyam tördelése
  - Küldés sorrendben
  - Nyugtázás (megbízható szolgáltatás)
  - Kerethatárok felismerése
- Forgalomszabályozás
  - Elárasztás elleni védelem (felsőbb rétegekben is)
  - Adó tájékoztatása a vevő szabad puffereiről
- Összeköttetés iránya (szimplex, fél duplex, duplex)
- Osztott csatornához való hozzáférés szabályozása

Az adatkapcsolati réteg

- Tervezési szempontok
  - Jól definiált interfész biztosítása a hálózati rétegnek
  - Az átviteli hibák kezelése
  - Az adatforgalom szabályozása
- Megvalósítás: Keretezés
  - A hálózati réteg csomagjainak keretbe foglalása
    - keret fejrész
    - adatmező
    - keret farokrész

Szolgáltatások típusai a hálózati réteg felé I.

- Nyugtázatlan összeköttetés mentes szolgálat
  - Egymástól független keretek küldése a forrástól a célig
  - Nincs előzetes kapcsolatépítés és bontás
  - Az elveszett kereteket nem állítja helyre a réteg
  - Alkalmazhatóság
    - Alacsony hibaarány esetén (javítás a felsőbb rétegekben)
    - Valós idejű adatforgalom esetén (hangátvitel)
- Nyugtázott összeköttetés mentes szolgálat
  - Szintén nincs előzetes kapcsolatépítés és bontás
  - Minden egyes keret megérkezését nyugtázza a cél állomás
  - Alkalmazhatóság
    - Pl.: vezeték nélküli kapcsolatoknál

Szolgáltatások típusai a hálózati réteg felé II.

- Nyugtázott összeköttetés alapú szolgálat
  - Összeköttetés felépítése az adatátvitel előtt
  - Sorszámozott keretek
    - minden keret garantáltan megérkezik
    - minden keret garantáltan egyszer érkezik meg
    - minden keret a megfelelő sorrendben érkezik meg
  - Az átvitel folyamata
    - összeköttetés felépítése (számlálók, változók inicializálása)
    - keret(ek) továbbítása
    - összeköttetés bontása (számlálók, változók felszabadítása)

Keretezés

- A fizikai réteg nem garantál hibamentes átvitelt
- A réteg feladata ezen hibák jelzése, javítása
- Keretezés
  - Az adatfolyam feldarabolása kisebb részekre (keret)
  - Ellenőrző összegek számítása minden kerethez
- Az átvitel ellenőrzése
  - A fogadott adatok alapján az ellenőrző összeg kiszámítása
  - A fogadott és az újraszámolt összeg összehasonlítása
- A kerethatárok jelölésének problémái
  - Szünetek beszúrása az egyes keretek közé?

A kerethatárok jelölése - Karakterszámlálás

- Minden keret fejrészében megadásra kerül a keret karaktereinek száma
  - A vevő a fejrészből tudja, hány karakter tartozik a kerethez
  - Illetve, hogy hol lesz a keret vége (a következő eleje)
- Problémák:
  - A karakterszám mezőt is érheti átviteli hiba
    - A vevő kiesik a szinkronból, nem találja a következő keretet
    - Hibás ellenőrző összeg → hibás keret
      - de hol kezdődik a következő keret?
    - Az újraküldés sem megoldás

Kezdő- és végkarakterek karakterbeszúrással

- A kerethatárok jelzése egy különleges karakterrel
  - Régebben eltérő protokollok eltérő bájtot használtak
  - Manapság egységes "jelző bájt" (flag byte) jellemző
- Ha a vevő kiesik a szinkronból csak megvárja a következő flag byte-ot
  - Az első flag byte az adott keret vége
  - A második flag byte a következő keret eleje
- Mi történik ha a flag byte-nak megfelelő karaktert akarunk átvinni? (bináris átvitel)
  - Kivétel bájt (escape byte) beszúrása
- És ha escape byte-ot kell átvinni?

Kezdő- és végjelek bitbeszúrással

- A karakterkódok hossza ne legyen része a keretezésnek
- Tetszőleges számú bit legyen átvihető egy keretben
- Flag byte: 01111110
- Az adó minden ötödik 1-es után beszúr egy 0-t
- A vevő minden ötödik 1-es után törli a követő 0-t
- A jelző bájt is probléma nélkül átvihető
  - Adási oldal: 01111110 → 011111010
  - Vételi oldal: 011111010 → 01111110
- Transzparens átvitel mindkét irányban
- Adatfolyam az adó oldalán: 0110111111011111111111010101
- Átviteli vonal: 0110111110101111101111101101010 1
- Adatfolyam a vevő oldalán: 0110111111011111111111010101

Fizikai rétegbeli kódolássértés

- Alkalmazható ha a fizikai réteg kódolása redundáns
- Például ahol egy bit kódolása is jelváltással történik
  - Előnyös:
    - a vevő könnyen felismeri a bithatárokat
    - Nincsenek szinkronizációs problémák
  - Logikai magas szint: magas → alacsony jelváltás
  - Logikai alacsony szint: alacsony → magas jelváltás
  - Nincs magas → magas vagy alacsony → alacsony átmenet
    - Ezek használhatók a kerethatárok jelzésére
- A gyakorlatban több megoldás kombinációja használt

Forgalomszabályozás

- A lassabb (terhelt) gép védelme elárasztás ellen
  - A túlterhelés keretek elvesztéséhez vezet
- Visszacsatolás alapú forgalomszabályozás
  - A vevő tájékoztatja az adót a pillanatnyi állapotáról
  - A vevő engedélyezi az adónak a további küldést
  - Az adatkapcsolati réteg jellemző megoldása
- Sebesség alapú forgalomszabályozás
  - A protokoll tartalmazza a sebességkorlátozást
  - Ez minden adóra nézve kötelező érvényű

Hibakezelés

- Lehetséges problémák
  - Megérkezett-e minden keret?
  - Minden keret csak egyszer érkezett-e meg?
  - A keretek megfelelő sorrendben érkeztek-e meg?
- Visszacsatolás szükséges a vevő felől
  - Pozitív és negatív nyugták
  - Ha nem érkezik meg a keret?
    - Időzítők alkalmazása
- Keretek újraküldése
  - Többször megérkezhet ugyanaz a keret
  - Kimenő keretek sorszámozása

Egybites és csoportos hibák

- Egybites hibák
- Csoportos hibák
  - Bizonyos közegek esetén sokkal gyakoribb
  - Azonos hibaarány mellett kevesebb hibás keret
  - Jelzése és javítása lényegesen nehezebb
- Példa:
  - 1 adatblokk legyen 1000 bit, a hibaarány = 0,1%
  - Független hibák esetén átlagban minden keret hibás lesz
  - 100 bites csoportos hibáknál 100-ból 1-2 keret lesz hibás

Hibajelző és hibajavító kódok

- Hibajelző kódok
  - Csak a hibás átvitel tényét jelzi → újraküldés
  - Alacsony hibaarány mellett lehet praktikus
- Hibajavító kódok
  - A hibás adatokból helyreállítható a helyes üzenet
  - Alkalmas lehet bizonytalan átviteli közegek esetén
- Redundancia
  - *m* - adatbitek
  - *r* - redundáns bitek
  - $n = m + r$ (teljes keret - *n* bites kódszó)

Hamming-távolság

- Azonos helyen előforduló különböző bitek (XOR)
- Teljes kód Hamming-távolsága
- *d* számú hiba jelzéséhez *d*+1 Hamming-távolságú kód szükséges
- *d* számú hiba javításához 2*d*+1 Hamming-távolságú kód szükséges
- Paritásbit alkalmazása
  - Hamming-távolság: 2
  - Alkalmas 1 bites hibák jelzésére

Hibajavító Hamming-kód

- Az üzenet bitjeit balról jobbra 1-től számozzuk
- 2 egész számú hatványai lesznek az ellenőrző bitek
- Az egyéb bitek az adatbitek

1	2	3	4	5	6	7	8	9	10	11
1	2	1,2	4	1,4	2,4	1,2,4	8	1,8	2,8	1,2,8

- Minden paritás számításában azok a bitek vesznek részt amelyekben kettő adott hatványa szerepel
- Például
  - 1-es paritásbit: 3, 5, 7, 9, 11 bitek
  - 4-es paritásbit: 5, 6, 7 bitek

Hamming-kód példa

1	2	3	4	5	6	7	8	9	10	11
1	2	1,2	4	1,4	2,4	1,2,4	8	1,8	2,8	1,2,8

- Kódolandó bitek: 1100101

1	2	3	4	5	6	7	8	9	10	11
0	0	1	1	1	0	0	0	1	0	1

- Hiba az átvitel során a 6-os számú biten

1	2	3	4	5	6	7	8	9	10	11
0	0	1	1	1	**1**	0	0	1	0	1

- Történt-e átviteli hiba? Hányadik bit a hibás?

Csoportos hibák javítása Hamming-kóddal

- Alapesetben egy bit javítható, csoportos hiba nem
- Rendezzünk *k* darab kódszót egymás alá
- Az átvitel ne kódszavanként és azon belül bitenként
- Hanem oszloponként, azon belül bitenként végezzük
- *k* bitnyi csoportos hiba esetén
  - Minden kódszóban csak egy bitnyi hiba lesz
  - Minden kódszó javítható
- Ellenőrző bitek száma = *kr*
- Adatblokk mérete = *km*
- *k* hosszúságú csoportos hiba javítható!

CRC hibaellenőrzés

- CRC - Cyclic Redundancy Code
  - Polinom kód
- A leggyakrabban alkalmazott megoldás hibajelzésre
- A bitsorozatokat polinomoknak tekintjük
  - pl.: 10011001 → $x^7+x^4+x^3+1$ → hetedfokú polinom
- Műveletek
  - Összeadásnál, kivonásnál nincs átvitel → XOR
  - Osztás: mint a bináris osztás
    - A kivonásnál nincs átvitel
    - Az osztó megvan az osztandóban ha az osztandó ugyanannyi bitet tartalmaz mint az osztó

CRC hibaellenőrzés

- Továbbítandó keret: M(x)
- Generátor polinom G(x)
  - Az adónak és a vevőnek is ismerni kell
  - A legfelső és legalsó bitnek 1-nek kell lennie
  - Rövidebbnek kell lennie mint a továbbítandó keretnek
- Feladat: fűzzünk ellenőrző összeget a kerethez, hogy az így kapott keret (polinom) osztható legyen G(x)-el
- Átvitel ellenőrzése: a vevő elosztja a vett keretet a generátorral, ha van maradék akkor hibás az átvitel

CRC ellenőrző összeg számítása

- Legyen r $G(x)$ foka. Fűzzünk r darab 0 értékű bitet a keret alacsony helyi értékű végéhez → $x^rM(x)$ amely $m + r$ bitből áll.
- Osszuk el a az így kapott keretünket [$x^rM(x)$] a generátor $G(x)$ bitsorozatával.
- Vonjuk ki a maradékot a bővített keretből így megkapjuk az ellenőrző összeggel ellátott továbbítandó keretet → $T(x)$

A CRC erőssége

- Minden egybites hibát képes jelezni
- Két izolált egybites hiba is jelezhető
  - G(x) nem oszthatja $x^k + 1$ -et, ahol k<kerethossz
  - Alacsony fokszámú polinomok, hosszú keretek védelmére
    - Pl.: $x^{15} + x^{14} + 1$ nem osztja $x^k + 1$-et ha k<32768
- Minden páratlan számú hibás bitet tartalmazó hiba felismerhető
  - ha $G(x)$ osztható $x + 1$ -gyel
- Minden r ellenőrző bittel ellátott polinomkód minden maximum r hosszúságú csoportos hibát tud jelezni

Néhány klasszikus CRC polinom

- CRC12 = $x^{12} + x^{11} + x^2 + x^1 + 1$
  - 6 bites karakterek kódolására
- CRC16 = $x^{16} + x^{15} + x^2 + 1$
  - 8 bites karakterek kódolására
  - Képes felismerni:
    - 1 bites hibák, 2 bites hibák
    - Páratlan hibás bitet tartalmazó hibák
    - 16 vagy kevesebb bitet tartalmazó csoportos hibát
    - 17 bites csoportos hibák 99,997%-át
    - 18 vagy nagyobb bitszámú csoportos hibák 99,998%-át
- CRC-CCITT = $x^{16} + x^{12} + x^5 + 1$
  - 8 bites karakterek kódolására (CCITT ajánlás)

CRC problémák?

- Bonyolult algoritmus az ellenőrző összeg számításához?
  - Áramkörileg egyszerűen megvalósítható
    - léptető regiszteres áramkör - Peterson és Brown
  - Gyakorlatilag szinte minden LAN használja
- Véletlenszerű-e a keretek tartalma?
  - Eredeti feltételezés: igen
  - Partridge és társai (1995)
    - Valós adatok elemzése
    - Alaptalan az eredeti feltevés, nem teljesen véletlenszerű
    - Gyakoribbak lehetnek a felderítetlen hibák