Univerza
v Ljubljani

Pedagoška
fakulteta

Računalniške komunikacije

Povezavna plast: Hammingovo kodiranje, CRC kodiranje

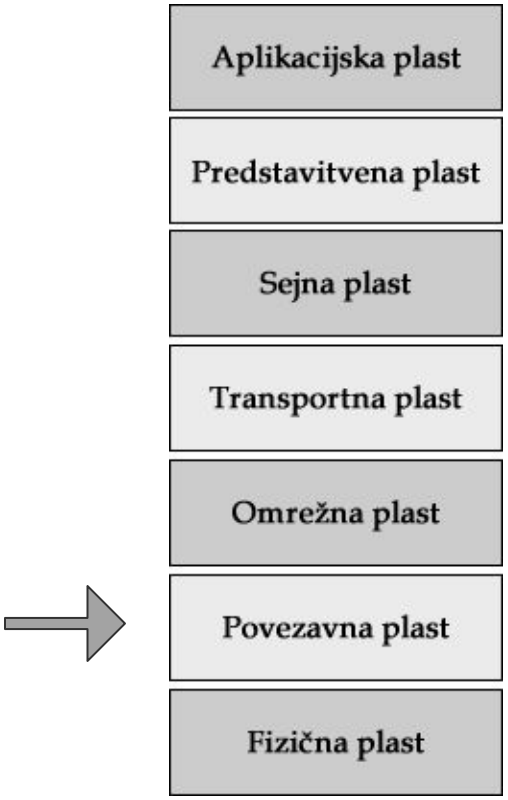
Nosilec predmeta: dr. Jože Rugelj
joze.rugelj@pef.uni-lj.si
Asistent: Matej Zapušek
matej.zapusek@pef.uni-lj.si

Povezavna plast

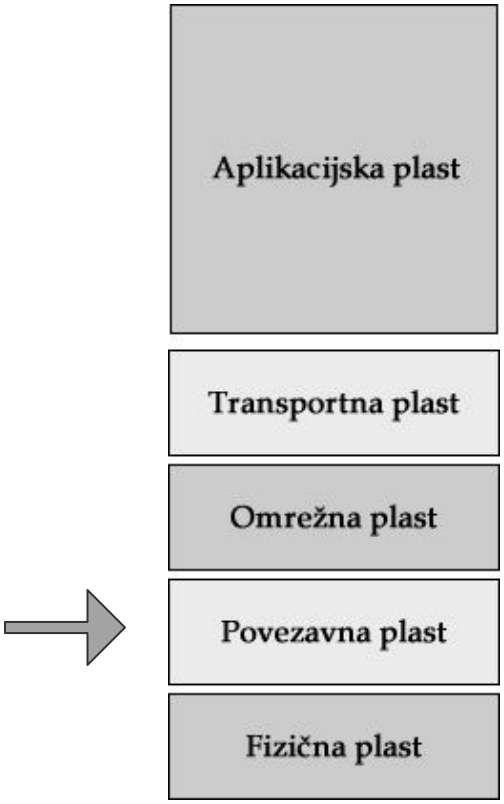
HAMMINGOVO KODIRANJE

Kje se nahajamo v modelih?

Model ISO/OSI



Model TCP/IP



Hammingovo kodiranje

- Zgodovina: Leta 1940 je Richard Hamming ugotovil, da je za nadaljnji razvoj računalnikov potrebna večja zanesljivost pri prenosu podatkov. Še posebno je bilo pomembno razviti sistem, ki bi napake odpravljal ne le detektiral.

- Razvil je metodo za odkrivanje in korekcijo napake na 1-bitu, ki jo imenujemo Hammingovo kodiranje

Uporaba

- V telekomunikacijah:
  - Linearno kodiranje z odpravo napak
  - Odpravi največ 1 bitno napako
  - Odkrije največ 2 bitno napako
- Matematično:
  - Razred linearnih dvojiških kodiranj
- Pogosta uporaba:
  - Pomnilniki RAM (npr. ECC registered RAM)

Hammingovo kodiranje

- Bistvo algoritma: uporaba dodatnih paritetnih bitov za detekcijo enojnih napak

- Kreiranje kodne besede:
  - Določimo pariteto: **liha** ali **soda**
  - Vrednost paritetnega bita postavimo na 1 oz. 0 glede na izbrano pariteto.
  - Na pozicije 2^n, kjer n>0 vstavimo mesta za paritetne bite.
  - Ostali biti namenjeni podatkom oz. začetni kodi, ki jo pošiljamo. Vpišemo jih zaporedno na preostala mesta.
  - Vrednost paritetnega bita se izračuna glede na vrednosti podatkovnih bitov, katerih binaren zapis pozicije ima bit na poziciji enaki poziciji paritetnega bita postavljen na 1.

- Da si lažje “zapomnimo”:
  - Pozicija 1: preskoči 0, upoštevaj 1, preskoči 1... (1,3,5,7,9,11...)
  - Pozicija 2: preskoči 1, upoštevaj 2, preskoči 2, (2,3,6,7,10,11...)
  - Pozicija 4: preskoči 3, upoštevaj 4, preskoči 4, (4,5,6,7,12,13...)
  - ...
  - Pozicija n: preskoči n-1, upoštevaj n, spusti n.

- Vrednost par. bita je odvisna od izbrane paritete. Če pariteta:
  - Soda: postavimo na 1 če število enic v bitih, ki jih preverjamo liha in na 0 če soda
  - Liha: postavimo na 1, če število enic v bitih sodo oz. na 0, če število liho

Primer

Vhodni podatek: 10011010

Kreiramo podatkovno besedo, pustimo prostorčke za paritetne bite:

_ _ 1 _ 0 0 1 _ 1 0 1 0

- Za poz. 1 preverimo bite 1,3,5,7,9,11 oz.
  ? _ **1** _ **0** 0 **1** _ **1** 0 **1** 0

ker št. 1 sodo, nastavimo prvi bit na 0

- Za poz. 2 preverimo bite 2,4,7,8,11 oz.
  0 **?** **1** _ 0 **0** **1** _ 1 **0** **1** 0
  ker št. 1 liho, postavimo drugi bit na 1

- Za poz. 4 preverimo bite 4,5,6,7,12 oz.
  0 1 1 **? 0 0 1** _ 1 0 1 **0**

ker št. 1 liho, postavimo četri bit na 1

- Za poz. 8 preverimo bite 8,9,10,11,12 oz.
  0 1 1 1 0 0 1 **? 1 0 1 0**

ker št. 1 sodo, postavimo 8 bit na 0

- Kodna beseda: 011100101010
- Kaj pa na drugem koncu?
- Recimo, da je prejemnik dobil kodno besedo: 011100101110
- Preverimo, če je v redu...
- Vidimo, da se ne ujema v 2. in 8. bitu.
- Pozicijo napake dobimo, če seštejemo poziciji paritetnih bitov.
- V našem primeru na poz. 10.

Dodatni primeri

- 010101100011
- 111110001100
- 000010001010

Povezavna plast

CRC KODIRANJE

CRC kodiranje

- Koda za **odkrivanje** napak
- Ciklične redundančne kode (Cyclic Redundancy Code/Check)
- Gre za funkcijo, ki poljubni količini bitov priredi neko fiksno vrednost
- Postopek: deljenje s polinomom
  - Generacijski polinom
  - Koeficienti so dvojiška števila
  - Pošiljatelj in prejemnik morata uporabljati enak polinom

Algoritem

- Binarno število dopolnimo z r ničlami
  - R je stopnja polinoma G(x)
- Dopolnjeno število delimo z G(X)
  - "deljenje" z operacijo XOR
  - ne upoštevamo izposoj in prenosov
- Dobljeni ostanek je koda CRC

Primer

- Imejmo niz bitov: 01010010_2
- Uporabimo polinom $G(x) = x^4 + x^2 + 1$

- 1. G(x) pretvorimo v binarno obliko
  - $G(x) = 1*x^4 + 0*x^3 + 1*x^2 + 0*x^1 + 1$
  - Koeficiente zapišemo v obliki binarnega števila
  - G(x) = 10101

- 2. Kodi dodamo **r** ničel
  - r je stopnja polinoma G(x) = 10101
  - v našem primeru je stopnja 4
  - dodamo 4 ničle

1.	2.	3.	4.	5.	6.	7.	8.	9.	10.	11.	12.
0	1	0	1	0	0	1	0	0	0	0	0

- Delimo z G(X) = 10101
  - Če je MSB bit deljenca enak 0 => prepišemo deljenec
  - Če je MSB bit deljenca enak 1 => XOR

  - To počnem, dokler ne pridem do konca..
  - Dobimo rezultat: ostanek pri deljenju (v našem primeru 1011)
  - Ta ostanek prepišemo na mesto prejšnjih r dodanih bitov in dobimo končni rezultat:

1.	2.	3.	4.	5.	6.	7.	8.	9.	10.	11.	12.
0	1	0	1	0	0	1	0	1	0	1	1

Na prejemnikovi strani…

- Sam izračuna kodo CRC (za število bitov, ki smo jih poslali)
  - Uporabiti mora enak G(x)
  - Če je izračunana CRC enaka sprejeti CRC => pri prenosu ni prišlo do napak
- Izračuna CRC kodo celotnega niza prejetih bitov
  - Če je tako izračunani CRC enak 0 => pri prenosu ni prišlo do napak