UPS

Cvičení 7

http://siroky.cz/vyuka/ups/

Opakování

- ISO/OSI
- TCP/UDP
- základní/přeložené pásmo
- modulace
- $\log_2(L)$
- bit/baud
- asynchronní, arytmický, synchronní
- Manchaster

Dobrovolné odevzdání

- příští cvičení
- protokol
- prototyp serveru
  - rychle předvést pár dotazů a odpovědí (např. nc jako klient)

Chyba přenosu

- dojde ke ztrátě či záměně dat
    - zkreslení signálu, rušení, šum
- bezpečnostní kódy
    - detekce chyb x oprava chyb
- Uvažuje binární symetrický přenosový kanál bez paměti:
    - binární: přenáší se 0/1
    - symetrický: 0/1 se přenáší se stejnou pravděpodobností
    - bez paměti: nezáleží co se přeneslo v předchozím kroku

Chyba při přenosu

- pravděpodobnost správného přenosu 1 bitu
  $P_1 = p_1$
  - Kontrolní otázka: jaká je nejmenší použitelná pravděpodobnost?

- pravděpodobnost správného přenosu N bitů
  $P_N = p_1^N$

Příklad

- $P_1=0.9999$
- $P_n=0.9$
- $N=?$

Příklad

- $P_1 = 0.9999$
- $P_n = 0.9$
- $N = ?$
- $P_N = P_1^N$: $0.9 = 0.9999^N$

Příklad

- $P_1=0.9999$
- $P_n=0.9$
- N=?
- $P_N=P_1^N$: $0.9 = 0.9999^N$
- $\log(x^y) = y \log(x)$: $\log(0.9) = N \log(0.9999)$

Příklad

- $P_1=0.9999$
- $P_n=0.9$
- $N=?$
- $P_N=P_1^N$: $0.9 = 0.9999^N$
- $\log(0.9) = N \log(0.9999)$
- $N = \log(0.9) / \log(0.9999)$

Příklad

- $P_1 = 0.9999$
- $P_n = 0.9$
- $N = ?$
- $P_N = P_1^N$: $0.9 = 0.9999^N$
- $\log(0.9) = N \log(0.9999)$
- $N = \log(0.9) / \log(0.9999)$
- $N = 1053$

Bezpečnostní kódy

- přidáme nějaké bity navíc nebo pozměníme data
- čím více bitů navíc tím účinnější metoda
- detekční – kontrola zda jsou data správně
- samoopravné – chybu rozpoznají a opraví

Parita

- přidáváme jeden paritní bit
- sudá 0 = sudý počet 1, 1 = lichý počet 1
  - vždy sudý počet 1 ve zprávě
  - umí jen detekovat, nevíme co je špatně
- lichá parita je analogie k sudé

Parita

- př.: doplňte lichý paritní bit do zpráv
  - 01001101
  - 111101

Parita

- př.: odesílatel: 10101101
  - Jaká je parita?
  - Které přijaté zprávy jsou "správné"?
    - 10101001
    - 10101000
    - 11111110

Checksum

- kontrolní součet – pro celý blok dat
- jednotlivé znaky chápeme jako čísla bez znaménka
- provádíme sčítání modulo 2^8 nebo 2^{16}
  - Kontrolní otázka: proč 2?
  - Kontrolní otázka: proč 8 nebo 16?
- výsledek je číslo o délce 1 nebo 2 bytů
- výpočet probíhá postupně

Checksum

- př.: spočítejte checksum modulo 2^8 zprávy

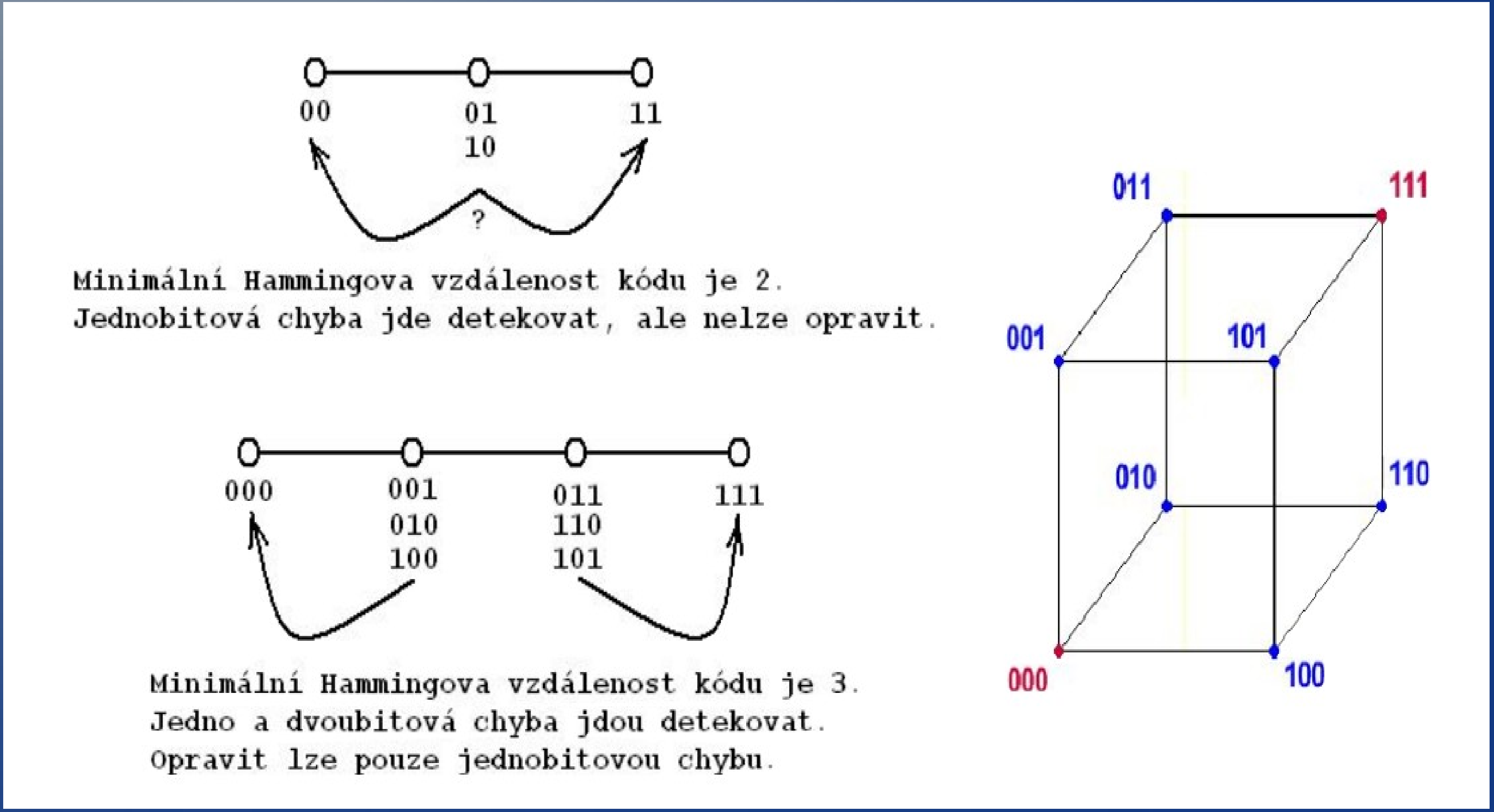
  0x3a 0x10 0x00 0xab 0x9f

Hammingova vzdálenost

- počet míst v němž se dvě kódová slova liší
  - např.: 000 a 001 mají vzdálenost 1, 010 a 101 mají vzdálenost 3
- minimální Hammingova vzdálenost d_{min} = minimální vzdálenost mezi všemi možnými páry vektorů
  - př. {0000, 1011, 1111}, d_{min}=?
  - př. {00000, 10110, 11100}, d_{min}=?

Hammingova vzdálenost

- 0 a 1 budu kódovat více bity, např. 00 nebo 111



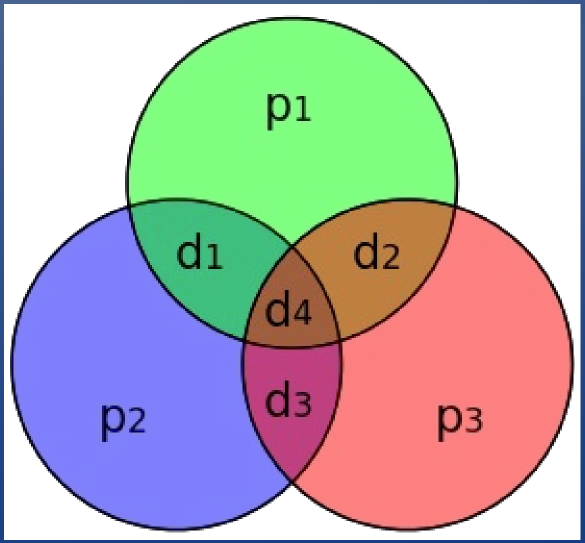
Hammingova vzdálenost

- Pro detekci n bitových chyb platí
  - $d_{min} > n$
- Pro detekci a korekci n bitových chyb platí
  - $d_{min} > 2n$

- Př.: 0:0000, 1:1111 - co dokážu říct o přijaté zprávě 0101?
- Př.: 0:00000, 1:11111 - co dokážu říct o přijaté zprávě 01010?

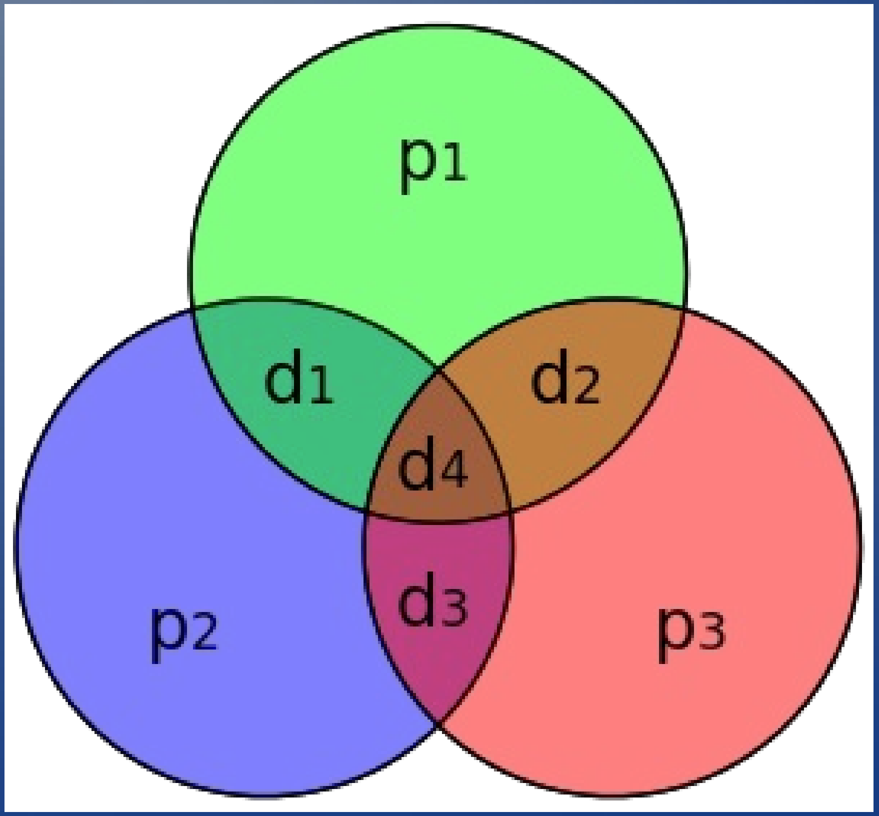
Hammingův kód (7,4)

- dovoluje detekovat dvojitou a opravit jednoduchou chybu
  - Kontrolní otázka: jaká je d_{min}?
- 4 datové bity, 3 sudé paritní
  - p1, p2, d1, p3, d2, d3, d4



Hammingův kód (7,4)

- p1, p2, d1, p3, d2, d3, d4
- př.: zakódujte zprávy:
  - 0000, 1111, 1011, 0010



Hammingův kód (7,4)

$$\mathbf{G} := \begin{pmatrix} 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \qquad \mathbf{H} := \begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix}.$$

- t = Gm
- c = Ht
- př.: zakódujte a ověřte zprávu:
  - 0110

CRC

- cyklický redundantní součet
- jednotlivé datové bity tvoří koeficienty polynomu
  - např. ...1001... -> $...1x^{14} + 0x^{13} + 0x^{12} + 1x^{11}...$ $= ...x^{14} + x^{11}...$
- tento se vydělí tzv. charakteristickým polynomem
  - např. CRC16: $x^{16} + x^{15} + x^{2} + 1$
    - Kontrolní otázka: kolika bitový je?
- data = podíl * charpolynom + **zbytek**
- **zbytek** se použije pro zabezpečení

CRC

- př.: zapište polynom pro 1011
- př.: zapište bitovou posloupnost polynomu $x^7 + x^3 + x^2$

CRC

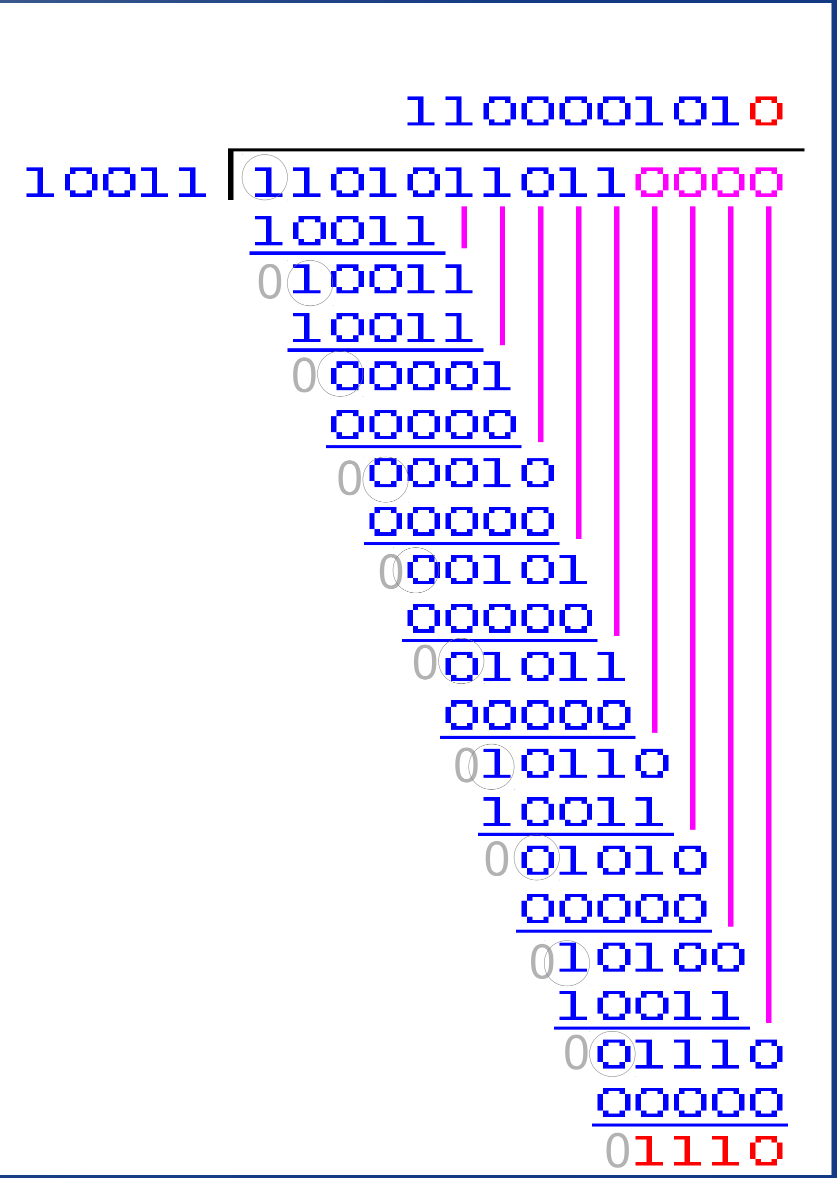
- vypočteme zbytek po dělení R(x) = M(x) % G(x)
- odesíláme T(x) = M(x) | R(x)
- po přijetí provedeme T(x) % G(x)
- pokud je zbytek nula, je přenos v pořádku
- označení jako CRC 16, 32 atp. podle stupně polynomu G(x)

CRC příklad

- $M(x)=1101011011$
- $G(x) = 10011$ ($x^4 + x + 1$)
  - polynom stupně 4
- za zprávu přidám 4 nuly ($M(x) * x^4$) a dělím
  - 1101011011 0000 / 10011
- $R(x) = 1110$

CRC příklad

- postup dělení
- stejné jako dělení pod sebe
- operaci odečítání nahrazuje operace XOR
- odesíláme M(x) | R(x)
  - 1101011011 | 1110



CRC příklad

- Ověření přijaté zprávy

```
                 1100001010
10011 | 11010110111110
        10011
         10011
         10011
          00001
          00000
           00010
           00000
            00101
            00000
             01011
             00000
              10111
              10011
               01001
               00000
                10011
                10011
                 00000
                 00000
                  0000
```

CRC samostatně

- $M(x) = 1010001100$
- $G(x) = x^5 + x^4 + x^2 + 1$
- $R(x) =$
- $T(x) =$

CRC samostatně

Zabezpečení

```
                 1101010111
110101 | 101000110000000
         110101
          111011
          110101
           011101
           000000
            111010
            110101
             011110
             000000
              111100
              110101
               010010
               000000
                100100
                110101
                 100010
                 110101
                  101110
                  110101
                   11011
```

Kontrola

```
                 1101010111
110101 | 101000110011011
         110101
          111011
          110101
           011101
           000000
            111010
            110101
             011110
             000000
              111101
              110101
               010001
               000000
                100010
                110101
                 101111
                 110101
                  110101
                  110101
                   00000
```