Predavanje 1 (23. 2. 2026)

- Ena visja od fizicne
- Prva plast, ki prejemnike/naslovnike ze naslavlja
  - Ti naslovi niso isto ko IP (logicni), ampak fizicni (doloceni z napravo)
- Enota -> **okvir** (*zaokrozena celota bitov*)
- Glavna naloga
  - prenos okvirja po povezavi med **sosednjima vozliscama (npr. racunalnik, usmerjevalnik)**, kjer upostevamo se tip medija

1 Kaj lahko (ampak ne rabi) izvaja povezavna plast?

- okvirjanje datagramov
- zaznavanje in odpravljanje napak
- dostop do medija
- zagotavljanje zanesljive dostave
- kontrola pretoka

2 Okvir

- "enota" na povezavni plasti
- opredeli zacetek in konec podatkov
- podatkom doda *glavo* (header) in *rep* (trailer), ki so potrebni za uspesen prenos

3 Enkapsulacija in dekapsulacija

- Enkapsulacija: dodajanje glave (in mogoce noge) k trenutnem paketu, ko se premika **dol** skozi plasti. S tem je paket ovit
- Dekapsulacija: odstranjevanje glave (in noge) od paketa, da lahko pridemo do podatkov, ko se paket premika **gor** skozi plasti

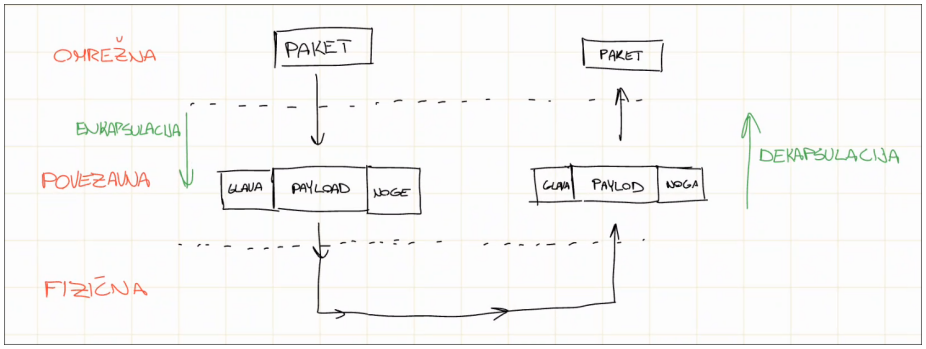

Figure 1: Prenos paketa

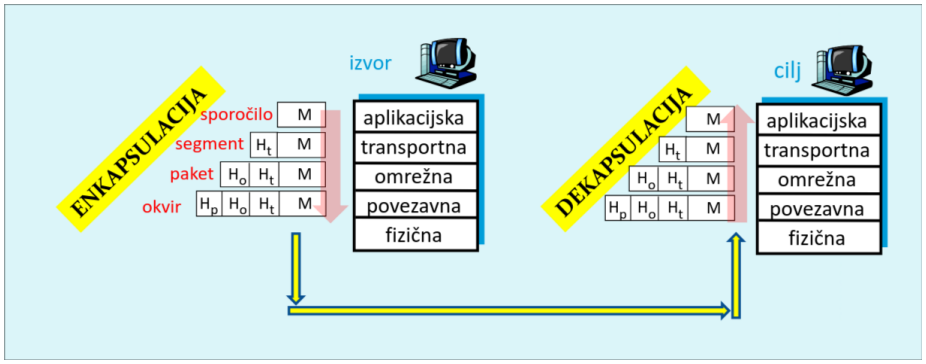

Figure 2: Enkapsulacija in dekapsulacija

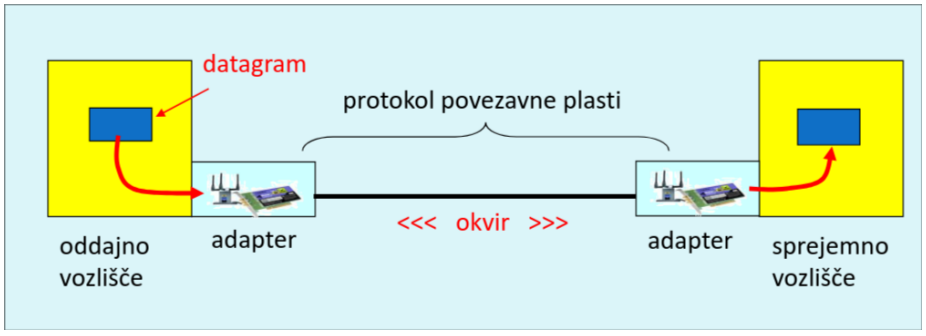

Figure 3: Implementacija povezavne plasti

4 Implementacija povezavne plasti

5 Zaznavanje in odpravljanje napak

- Zaradi suma je lahko tezko videti, ali je signal enako 0 ali 1

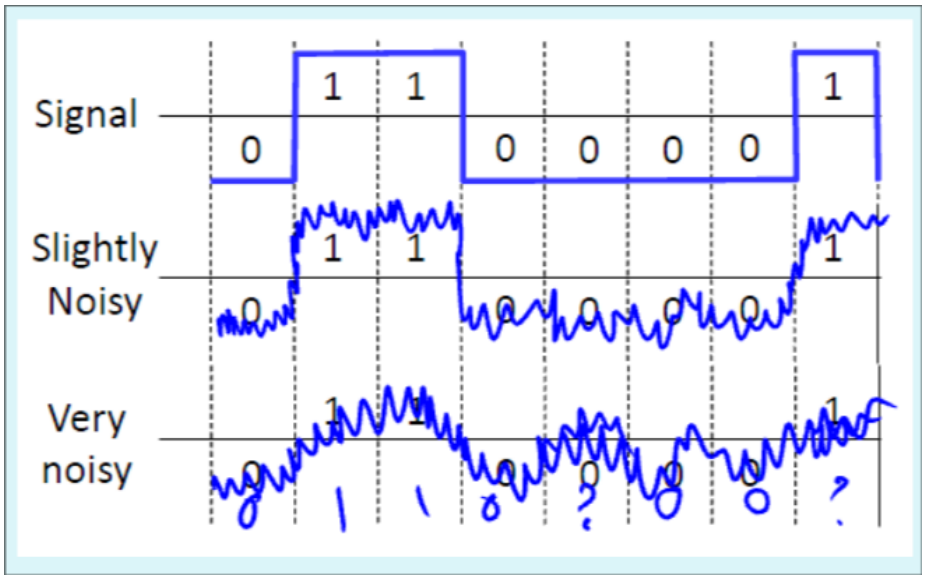

Figure 4: Razlog iz probleme/napake

- Da lahko zaznavamo, dodamo dodatne *bite za preverjanje pravilnosti* (**EDC** - Error Detection Code)
  - Napake so se vedno mozne
  - Vec bitov -> vecja moznost zaznave

5.1 Parnost

- Nacin za dolocenje EDC bitov
- Dodamo en *paritetni bit*
- Soda paritetna shema: 0 -> sodo stevilo enic, 1 -> liho stevilo enic
- Liha paritetna shema: 0 -> liha stevilo enic, 1 -> sodo stevilo enic
- **boljse pravilo**:
  - soda pariteta: stevilo enic skupaj z paritetnim bitom je *sodo*
  - liha pariteta: stevilo enic skupaj z paritetnim bitom je *liho*
- Problem? : ce se dva, stiri, ... biti pokvarijo, se pariteta ohrani in napaka ni zaznana

5.2 Parnost v dveh dimenzijah

- Dodamo paritetni bit za vsako vrstico in vsak stolpec
- Posebni bit -> kumulativni bit ali meta-paritetni bit
  - sestejemo vse paritetne bite in jih glede na pariteto dolocimo se en bit

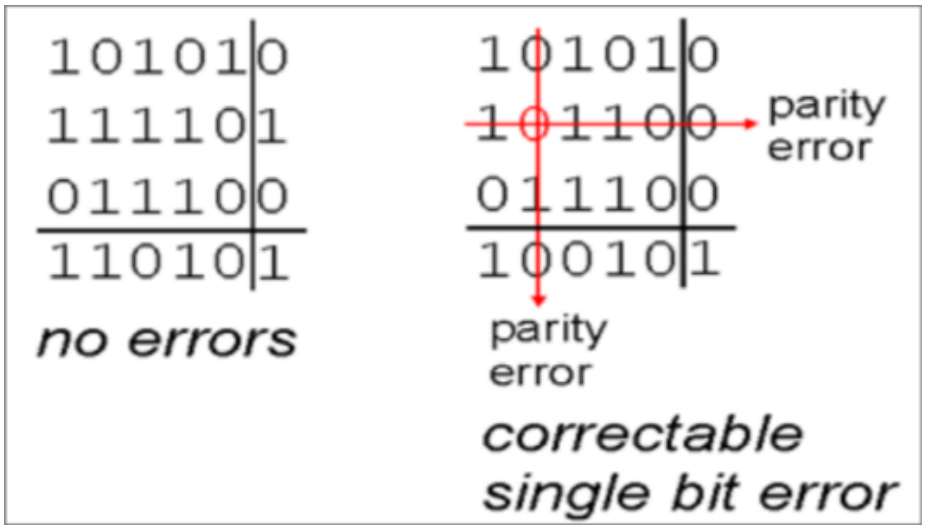

Figure 5: Pariteta v dveh dimenzijah (spodaj desno je kumulativni bit)

5.3 Hemmingova koda

- Se en nacin za dolocanje EDC bitov
- Primer: podatek = $0101_{(2)}$, moramo dodati 3 kontrolne bite
  - zakodiramo s sodo paritetno shemo (glede na to dolocamo bite)
- Prejemnik izracuna **sindrom** -> ce so v sindromu samo nicle, **ni** prislo do napake, drugace moramo popraviti bit, ki je na mestu enak sindromu
  - sindrom izracunamo tako, da sestejemo vrednosti tistih bitov (po modulu 2), ki smo jih uporabili pri dolocanju Hemmingovega bita
  - te bite zapisemo v nasprotnem vrstnem redu

6 Protokoli z dostop do skupinskega medija

- Dve vrsti povezav:
  - dvotockovna: vsaka povezava ima le enega naslovnika in posiljatelja
  - oddajna: deljeni medij, vec vozlisc naenkrat
- Da lahko trke preprecimo, potrebujemo nek *protokol za koordinacijo dostopa*
- Za idealni protokol velja:

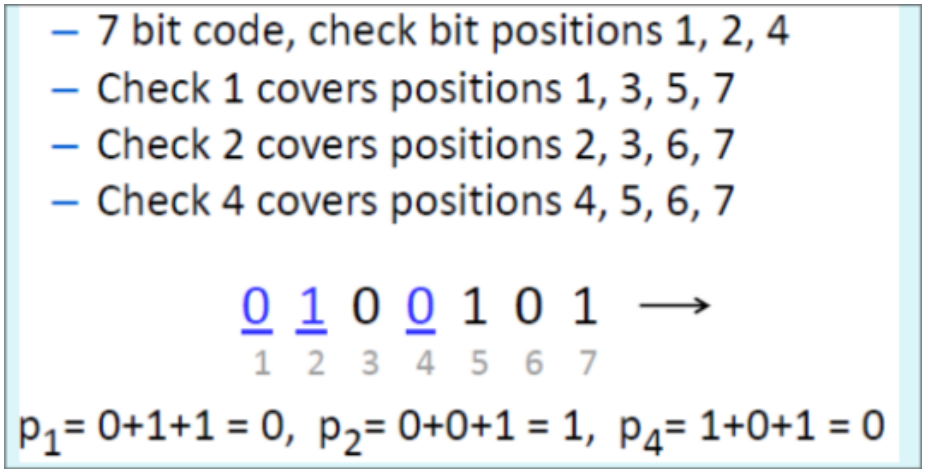

Figure 6: Primer generiranje Hemmingove kode

Figure 7: Dolocanje sindroma in zaznavanje (in odpravljanje) napake

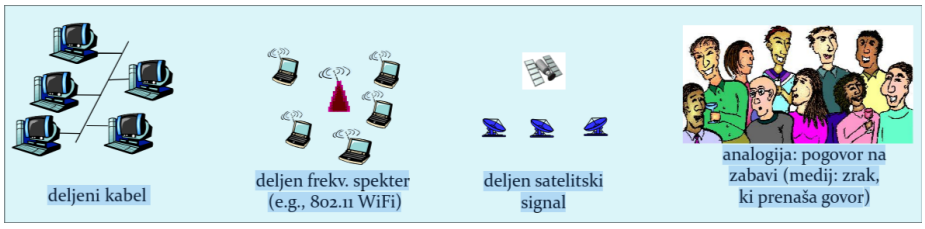

Figure 8: Primeri povezav (kjer lahko pride do kolizij/trkov)

1. Ce oddaja samo eno vozlisce, oddaja s hitrostjo R (eno vozlisce izkoristi celotno hitrost)
2. Ce oddaja M vozlisc, oddajajo s povprecno hitrostjo R/M (vsak pride do svojega deleza)
3. Protokol je decentriliziran (ni centralnega vozlisca, ki ga opravlja)
4. Je enostaven

7 Izogibanje in razresevanje kolizij

- Imamo 3 druzine protokolov:
  1. Delitev kanala (ni kolizij)
  2. Nakljucni dostop (so kolizije)
  3. Izmenicni dostop (ni kolizij)

7.1 Delitev kanala

7.1.1 TDMA (Time Division Multiple Access)

- vsaka postaja dobi enak casovni interval (1 paket)

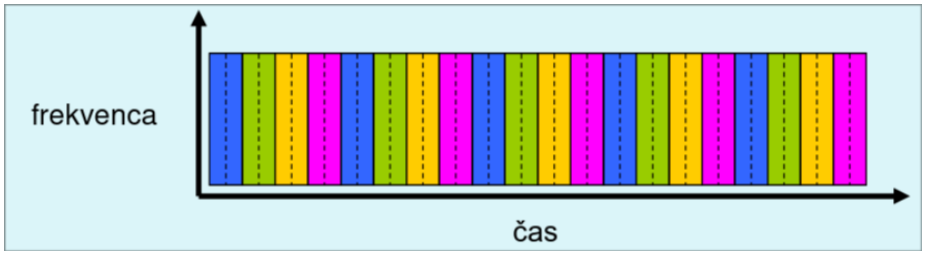

Figure 9: Primer TDMA protokola

7.1.2 FDMA (Frequency Division Multiple Access)

- vsaka postaja dobi *svoj frekvencni pas*

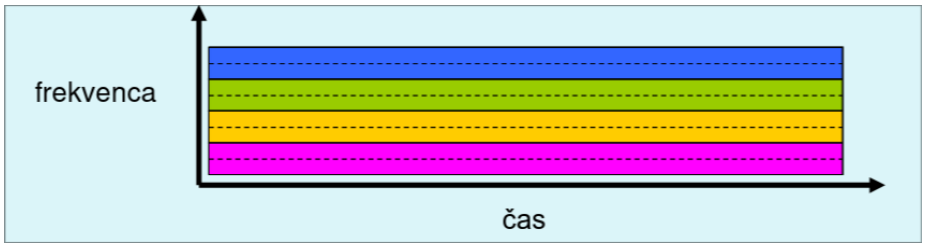

Figure 10: Primer FDMA protokola

7.2 Protokoli za nakljucni dostop

- Dolocajo:
  - kako zaznati kolizijo in
  - kako z njo ravnati
- Postaja poslje kadar hoce in uporabi celotno hitrost kanala R
- Primeri:
  - ALOHA
  - razsekana ALOHA
  - CSMA, CSMA/CA, CSMA/CD

7.2.1 ALOHA

- Paket je vedno ranljiv
- Ce paket ne pride (**pride do kolizije**), je paket ponovno poslan po nekem nakljucnem casu
- Izkoristek/Prepustnost == **18%** (zelo slabo)

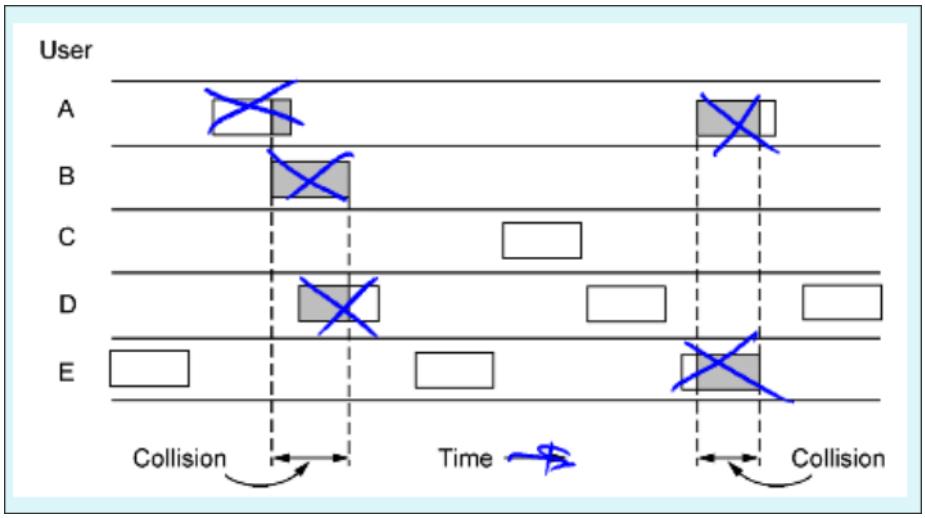

Figure 11: Protokol ALOHA

7.2.2 Razsekana ALOHA

- Cas razsekamo na dolocene casovne intervalne
  - Cas ni vec *zvezen* !!!
  - Vozlisca so *sinhronizirana*, vsa posiljajo na zacetku intervala
- Ce pride do kolizije -> okvir ponovno poslan v naslednjem intervalu z verjetnostjo `p`
- Izkoristek/Prepustnost == **37%** (zelo slabo)

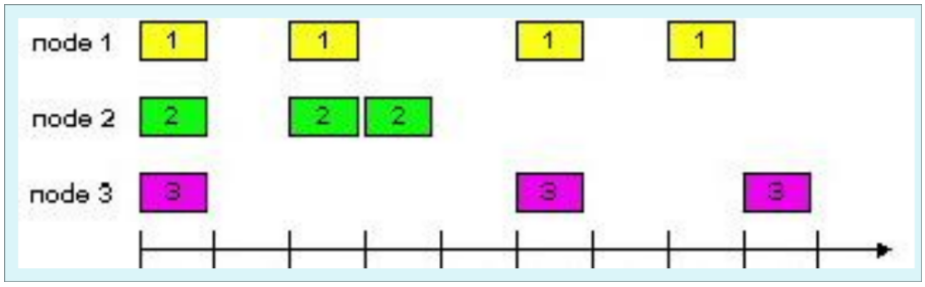

Figure 12: Razsekana ALOHA