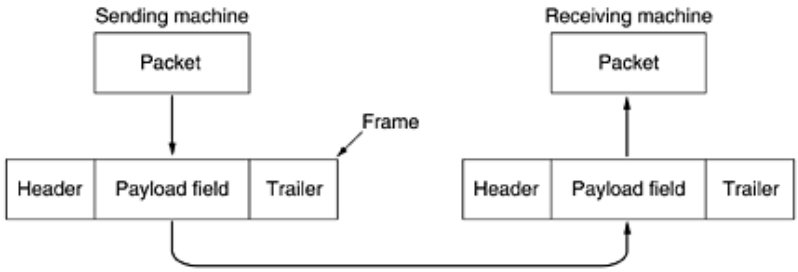
Računarske mreže i mrežne tehnologije

4. termin

1. Sloj veze podataka (data link layer)

- Namena sloja veze podataka je **postizanje pouzdane i efikasne komunikacije između dva susedna računara**, tj. pružanje pouzdane i efikasne usluge prenosa bitova mrežnom sloju.
- Naizgled jednostavan zadatak znatno se komplikuje na **nepouzdanoj liniji** (fizičkom sloju).
- Sloj veze podataka ima sledeće funkcije:
    - Dobro definisan uslužni interfejs ka mrežnom sloju
    - Obrada grešaka pri prenosu (**error control**)
    - Upravljanje tokom podtaka (**flow control**), kako ciljni računar ne bi bio preplavljen
- **Osnovna jedinica prenosa je okvir (frame)**, koji sadrži zaglavlje (*header*), polje za korisničke podatke i završni blok (*trailer*).

Figure 3-1. Relationship between packets and frames.

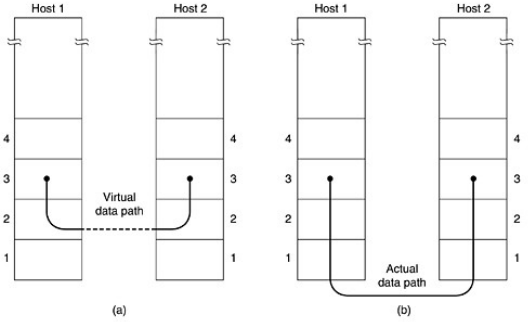


- Principi koji se koriste u sloju veze podataka često se koriste i u višim slojevima (npr. transportnom sloju).

2. Usluge koje se obezbeđuju za mrežni sloj

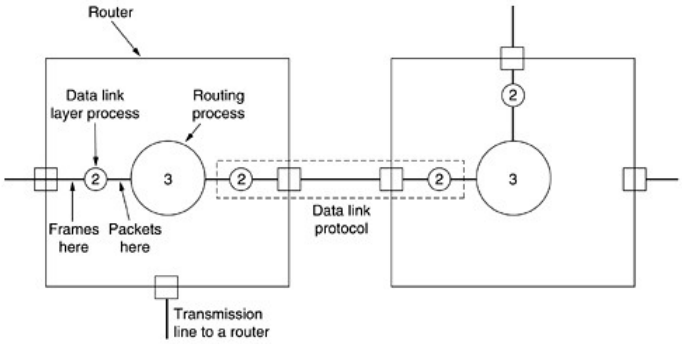
- Osnovna usluga je prenos podataka od mrežnog sloja izvorišnog do mrežnog sloja odredišnog računara.

Figure 3-2. (a) Virtual communication. (b) Actual communication.



- Sloj veze podataka tipično pruža sledeće 3 usluge:
    - prenos podataka bez uspostavljanja direktne veze, bez potvrde o prijemu
    - prenos podataka bez uspostavljanja direktne veze, sa potvrdom o prijemu
    - prenos podataka sa uspostavljanjem direktne veze, sa potvrdom o prijemu
- U većini **lokalnih mreža**, u sloju veze se koristi usluga prenosa bez uspostavljanja direktne veze i bez potvrde o prijemu podataka.
- Na pouzdanim kanalima, npr. optičkim, nema potrebe opterećivati mrežu bezbrojnim potvrdama o prijemu, dok je kod bežičnih kanala to obavezna stvar.
- Kod najsloženije usluge **sa uspostvaljanjem direktne veze**, okviri moraju da pristižu na odredište **istim redom kojim su poslati**. Ova vrsta usluge obezbeđuje pouzdan tok bitova.
- Protokoli sloja veze podataka najčešće su ugrađeni u sam hardver mrežnog interfejsa (npr. Ethernet). Mesto protokola sloja veze **unutar rutera (usmerivača)** dato je na slici:

Figure 3-3. Placement of the data link protocol.



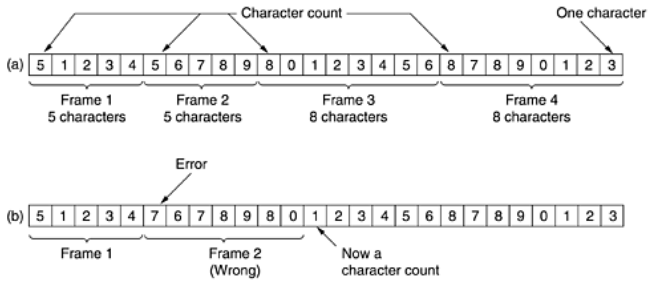
3. Uokviravanje

- **Da bi pružio usluge mrežnom sloju, sloj veze koristi usluge fizičkog sloja**. Fizički sloj prihvata tok sirovih podataka i pokušava da ih isporuči, ali bez garancije pouzdanosti.
- Sloj veze treba da **otkrije greške pri prenosu** i da ih po potrebi **ispravi**.
- Osnovni kvant u mrežnom sloju je **okvir (*frame*)**.
- Deljenje podataka na okvire nije tako jednostavno. Jedna od ideja je da se između okvira ubaci **prazan vremenski interval**, ali je to loše zbog nedostatka sinhronizacije, kao i gubljenja dragocenog vremena.
- Uglavnom su u upotrebi sledeće četiri metode uokviravanja:
    1. Prebrojavanje znakova
    2. Indikatorski bajtovi uz umetanje bajtova
    3. Početni i završni indikatori uz umetanje bitova

3.1 Prebrojavanje bajtova

- Prebrojavanje bajtova je jedna od najstarijih metoda uokviravanja. U zaglavlju okvira postoji polje sa brojem znakova unutar tog okvira. Na slici su data 4 okvira dužine 5, 5, 8 i 8 znakova respektivno.

Figure 3-4. A character stream. (a) Without errors. (b) With one error.

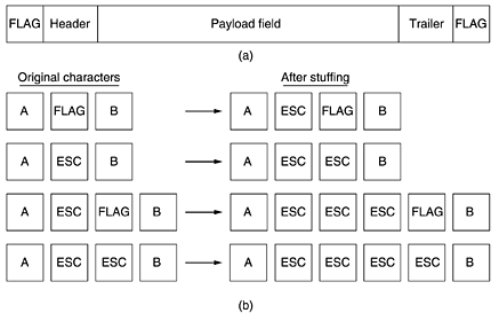


- Problem može da nastane usled pogrešnog očitavanja broja znakova kada ova metoda potpuno gubi sinhronizaciju.

3.2 Indikatorski bajtovi uz umetanje znakova

- Kod ove metode uokviravanja, početak i kraj svakog okvira obeležava se tzv. **indikatorskim bajtom** (*flag byte*).
- U slučaju da odredišni računar izgubi korak, sve što treba da uradi je da pronađe naredni **FLAG** bajt, kojim je označen kraj aktuelnog okvira. **Dva FLAG bajta** označavaju kraj jednog i početak drugog okvira.
- Problem može da nastane pri prenosu binarnih podataka, jer se lako događa da se **FLAG bajt nađe unutar samih podataka** koje treba preneti. Ovo se rešava umetanjem kontrolnog karaktera (ESC) ispred takvog bajta.

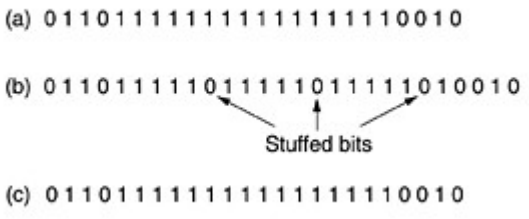
Figure 3-5. (a) A frame delimited by flag bytes. (b) Four examples of byte sequences before and after byte stuffing.



- Poznati **PPP (Point to Point Protocol)** protokol sloja veze podataka koristi sličnu shemu.
- Problem ovog pristupa je što **nisu svi kodni rasporedi osmobitni**, recimo UNICODE.

3.3 Početni i završni indikatori uz umetanje bitova

- Svaki okvir počinje i završava se specijalnom sekvencom bitova, tj. indikatorskim bajtom **01111110**.
- Kada god sloj veze pošiljaoca naiđe na 5 uzastopnih jedinica (u podacima koji se šalju), automatski umeće nulu u izlazni tok, što je analogno umetanju ESC.
- Kada primalac u dolaznom toku pronađe 5 uzastopnih jedinica iza kojih sledi nula, on **automatski izbacuje tu nulu**.



- Primenom ove metode, nedvosmisleno se utvrđuje granica dva okvira na osnovu indikatroskog koda.

4. Kontrola grešaka

- **Glavno pitanje:** Kako biti siguran da su okviri isporučeni mrežnom sloju primaoca, i to ispravnim redosledom?
- Pouzdanost isporuke se utvrđuje tako što **pošiljalac dobija povratnu informaciju** šta se događa na drugom kraju linije.
- Ako se okvir potpuno izgubi, pošiljalac može večno da čeka potvrdu. Ovaj problem se rešava uvođenjem **tajmera**.
- Posle određenog vremena tajmer ukazuje pošiljaocu da je došlo do problema, nakon čega **pošiljalac ponovo šalje isti okvir**.
- Međutim, uvek postoji rizik da se isti okvir primi više puta. Da bi se to sprečilo, okvirima koji se šalju dodeljuju se **redni brojevi** (*sequence numbers*).

5. Kontrola toka podataka

- Kontrola toka se bavi pitanjem "**Šta raditi sa pošiljaocem koji sistematski šalje okvire brže nogo što primalac može da ih prihvati?**"
- Usled takve situacije, može doći do gubljenja okvira.
- Da bi se adresiralo ovo pitanje, primenjuju se dve vrste pristupa:
    1. **upravljanje tokom na osnovu povratnih informacija** (*feedback-based flow control*)
    2. **upravljanje zasnovano na ograničenju brzine** (*rate-based flow control*)
- Sistemi zasnovani na ograničenu brzine nikad se ne ugrađuju u sloj veze podataka, već u više slojeve.

6. Otkrivanje i ispravljanje grešaka

- Telefonski sistem se sastoji iz tri dela: skretnica, vodova i lokalnih linija. U savremenoj telefoniji, **jedini analogni deo su lokalne linije** i u njima su greške još uvek česte.
- **Podaci se uvek šalju u blokovima**, recimo veličine 1000 bita. Ako je učestanost grešaka 0.001 po bitu, moglo bi se zaključiti da je svaki blok pogrešan.
- Međutim, greške se obično ne javljaju nezavisno nego **u rafalima** (*burst errors*).

- Dve su osnovne strategije za obradu grešaka:
    1. **Kodovi za ispravljanje grešaka** (*error-correcting codes*), koja uz podatke šalje i njihov "višak" koji je dovoljan da, i pored greške primalac zaključi šta su bili stvarni podaci. Koristi se npr. u **bežičnim mrežama** sa visokim šumom.
    2. **Kodovi za otkrivanje grešaka** (*error-detecting codes*), koja uključeni višak podataka koristi samo da otkrije da greška postoji i da zahteva ponovno slanje okvira.Koristi se kod **optičkih vlakana** i **lokalnih mreža**.

6.1 Kodovi za ispravljanje grešaka

- Okvir se sastoji od ***m* bitova podataka i *r* redundantnih (kontrolnih bitova)**:

  $n=m+r$

- Jedinica od n bitova koja sadrži i podatke i kontrolne bitove zove se **kodna reč** (*codeword*).
- **Hamingovo rastojanje** se definiše kao broj pozicija bitova u kojima se razlikuju dve kodne reči, u primeru 3:

  10001001
  10110001
  00111000 XOR

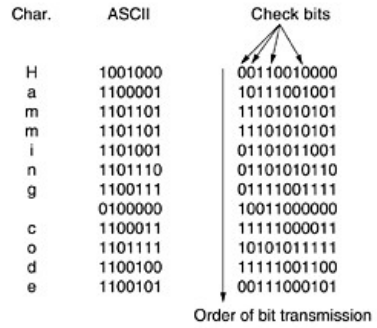
- Za dve kodne reči sa Hamingovim rastojanjem *d* potrebno je ***d* jednobitnih grešaka** da bi se jedna konvertovala u drugu.
- U opštem slučaju, dozvoljeno je slanje svih 2^m kombinacija bitova podataka, ali **nije dozvoljeno slanje svih 2^n** kodnih reči, kako bi se sačuvalo određeno Hamingovo rastojanje.
- **Hamingovo rastojanje celog koda** je minimalno Hamingovo rastojanje dozvoljenih kodnih reči određenih algoritmom za računanje *r* kontrolnih bitova.
- Mogućnost koda da otkrije i ispravi greške **zavisi od njegovog Hamingovog rastojanja**.
- **Da bi se detektovalo *d* grešaka Hamingovo rastojanje koda mora biti *d*+1**, jer tada *d* grešaka nemaju šanse da promene jednu kodnu reč u drugu važeću kodnu reč.
- **Da bi se ispravilo *d* grešaka Hamingovo rastojanje koda mora biti 2*d*+1**, jer je tada, čak i uz *d* grešaka originalna kodna reč bliža od svih drugih kodnih reči, pa se može nedvosmisleno utvrditi.
- Jedan primer je **bit parnosti**, gde se podacima dodaje nula ako je zbir jednica paran, a jedinica ako je zbor jedinica neparan. Na primer, reč 1011010 postaje 10110100, jer je zbir jedinica paran. Ovaj kod ima Hamingovo rastojanje 2, pa može poslužiti za detekciju 1-bitnih grešaka jer izmena bilo kog bita dovodi do pogrešne parnosti.
- **Jednostavan primer koda za ispravljanje grešaka** je kod koji ima samo četiri važeće kodne reči:

  0000000000 0000011111 1111100000 1111111111

  Hamingovo rastojanje ovog koda je 5, što znači da može da ispravi ($2d+1=5$), odnosno $d=2$ greške. Ako pristigne kodna reč 0000000111, primalac zna da je to 0000011111 kojoj su izmenjena 2 bita. Ako, međutim, trostruka greška izmeni 0000000000 u 0000000111, greška se ne može potpuno ispraviti.
- **Koliko je kontrolnih bitova (*r*) dovoljno za ispravljanje pojedinačnih grešaka?** Svaka od 2^m važećih poruka ima *n* nevažećih kodnih reči na rastojanju 1 od sebe. Tako svaka od 2^m poruka pokriva $(n+1)$ 2^m sekvenci bitova, tj. mora da važi:

$(n+1)2^m \leq 2^n$ ili $(m+r+1) \leq 2^r$

- **Hamingov kod** se uvodi tako što se bitovi kodne reči numerišu s leva na desno, bitovi čija je pozicija stepen broja 2 (1,2,4,8,16,...) predstavljaju kontrolne bitove dok su ostali bitovi bitovi podataka (ukupno *m*). Svaki kontrolni bit vodi računa o tome da li određeni skup bitova ima paran ili neparan broj jedinica. Na primer, o bitu na 13. poziciji, računa vode kontrolni bitovi 8,4 i 1 jer je 13=8+4+1. Primer:



- Gornji primer pokazuje kako se neki znaci iz 7-bitnog ASCII koda prevode u 11-bitni Hamingov kod.
- **Hamingov kod može da ispravlja samo pojedinačne greške**, dok se **rafalne greške** koje se često javljaju mogu ispraviti uz pomoć jednog trika. Naime, sekvenca od *k* kodnih reči poređa se u matricu, kao na slici. Umesto po vrstama, podaci se šalju u kolonama, počevši od krajnje leve. Ako dođe do rafalne greške dužine *k* (12 u primeru), ona će izmeniti najviše po jedan bit u svakoj kodnoj reči, što će Hamingov kod savršeno ispraviti.

6.2 Kodovi za ispravljanje grešaka

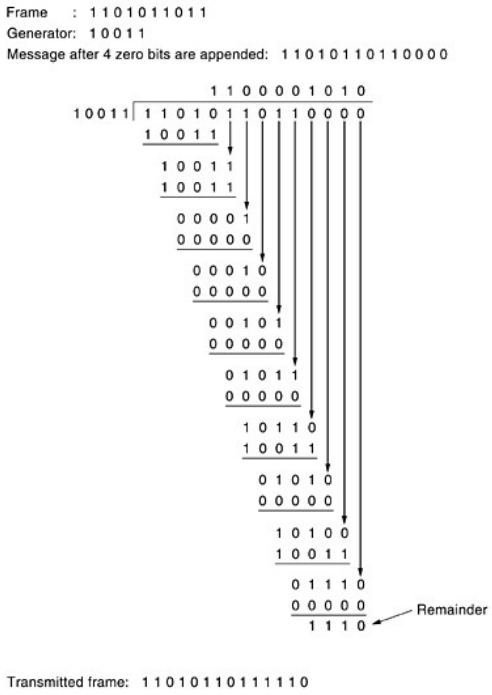
- **Učestanost pojave grešaka na bakarnom i optičkom kablu je dosta manja nego pri bežičnom prenosu**, tako da se umesto kodova za ispravljanje grešaka koriste kodovi za otkrivanje grešaka, jer je efikasnije tek povremeno ponovo poslati podatke umesto uključiti veliku količinu redundantnih podataka.
- Neka je frekvencija pojave grešaka 10^{-6} po bitu, a dužina bloka 1000 bita. **Da bi se sprovelo ispravljanje grešaka** Hamingovim kodom bilo bi potrebno $(m+r+1) \leq 2^r$ 10 kontrolnih bitova, tj. za 1Mbit podataka 10000 kontrolnih bitova.
- S druge strane, **da bi se samo otkrilo postojanje 1-bitne greške**, dovoljan je jedan bit parnosti po bloku. To znači da na 1000 blokova (1Mbit) treba poslati još samo 1001 bit parnosti. Uz ponovno slanje jednog bloka to je ukupno 2001 bit, tj. 5 puta manje opterećenje nego kod metode ispravljanja grešaka.
- **Ako se bloku priključi samo jedan bit parnosti**, a on bude izobličen dugačkom rafalnom greškom, **verovatnoća da ta greška bude otkrivena iznosi svega 0.5**, što je daleko od zadovoljavajućeg.
- Verovatnoća se može znatno povećati ako se **podaci šalju u obliku matrice sa *n* kolona i *k* vrsta**. Bit parnosti se izračunava nezavisno za svaku kolonu i priključuje kao poslednji red matrice. Na taj način, verovatnoća da svaka kolona slučajno ima ispravnu parnostje 0.5, tako da je ukupna verovatnoća da će neispravan blok biti prihvaćen kao ispravan 2^{-n}.
- U praksi se primenjuje jedna druga metoda, tzv. **polinomski kod**, poznata i ako **CRC** (*Cyclic Redundancy Check*).
- Prema CRC metodi, sekvenca bitova se smatra polinomom čiji su koeficijenti samo nule i

jedinice. Okvir od *k* bitova smatra se polinomom stepena *k*-1, sa članovima x^{k-1} do 1. Na primer $110001 \Leftrightarrow x^5+x^4+x^0=x^5+x^4+1$.

- Izrazi sa polinomima računaju se po modulu 2, dok su i sabiranje i oduzimanje definisani operacijom XOR, dok delilac "ide u" deljenik ako imaju isti broj bitova.

```
  10011011      00110011      11110000      01010101
+ 11001010    + 11001101    - 10100110    - 10101111
----------    ----------    ----------    ----------
  01010001      11111110      01010110      11111010
```
XOR

- Kod CRC-a, pošiljalac i primalac moraju se složiti oko tzv. **generatorskog polinoma** *G*(x). Da bi se izračunao **kontrolni zbir** za okvir od *m* bitova, okvir mora biti duži od generatorskog polinoma. Namera je da se kontrolni zbir doda na kraj okvira tako da polinim koji predstavlja okvir sa kontrolnim zbirom bude deljiv sa G(x). Ukoliko to nije slučaj, došlo je do greške i zahteva se ponovno slanje okvira.
- Algoritam se može opisati u tri stavke:
  1. Neka je *r* stepen polinomima *G*(x). Na kraj okvira dodaje se r nula, tako da on sada sadrži *m*+*r* bitova i odgovara mu polinom $x^r M(x)$.
  2. $x^r M(x)$ se deli po modulu 2 sa G(x).
  3. Oduzima se ostatak od prethodnog deljenja od $x^r M(x)$ i time se dobija okvir sa kontrolnim zbirom koji treba poslati.
- Kao primer dat je okvir 1101011011 uz generator $G(x)=x^4+x+1$.



Transmitted frame: 11010110111110

- Ako sekvenca sadrži grešku, umesto polinoma T(x), na drugom kraju linije će se pojaviti polinom T(x)+E(x), pa kad se on podeli po modulu 2 sa G(x) dobija se:
  $\frac{T(x)}{G(x)}+\frac{E(x)}{G(x)}$.

  Pošto je se G(x) sadrži u T(x), ostatak treba da se pojavi samo pri deljenju E(x)/G(x).
- Ako G(x) sadrži monom "1", **mogu se otkriti sve jednobitne greške**, jer se $E(x)=x^k$ ne može podeliti sa G(x) koje sadrži jedinicu

- **Dve jednobitne greške** se u polinomskom obliku mogu napisati kao
  $E(x)=x^j(x^{i-j}+1)$
  Može se pokazati da dovoljan uslov za otkrivanje dvostrukih grešaka da se sa G(x) ne može podeliti x^{k+1} gde je $k \in \{1, i-j\}$.
- Polinom dužine *r* bitova otkriće sve rafalne greške koje nisu duže od *r*. Ako se rafalna grešla predstavi kao
  $x^i(x^{k-1}+...+1)$,
  gde i označava udaljenost greške od desne strane okvira, očigledno je da ako G(x) sadrži "1" neće imati x^i kao činilac, a ako je polinom u zagradi manjeg stepena od G(x), rafal će biti otkriven.
- Polinom koji se koristi kao standard u mrežama IEEE802 izgleda ovako:
  $x^{32}+x^{26}+x^{23}+x^{22}+x^{16}+x^{12}+x^{11}+x^8+x^7+x^5+x^4+x^2+x+1$