UNIVERZA V LJUBLJANI

Fakulteta za elektrotehniko

Stanislav Kovačič

Komunikacije v avtomatiki

(študijsko gradivo)

Ljubljana, 19. oktober 2001

Kazalo

1 Uvod

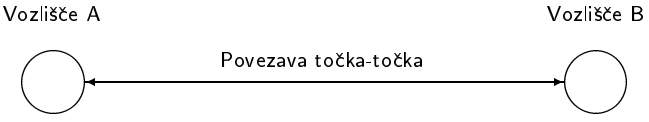
Osnovni problem na področju vodenja krajevno porazdeljenih sistemov je prenos informacij med krajevno ločenimi procesi ali pa med posameznimi deli istega procesa. Ni dolgo tega, kar se je ta problem reševal z nalašč za določen primer uporabe razvitimi komunikacijskimi sistemi. Takšni sistemi, ki so bili načrtovana za čisto specifične potrebe, so se s časom izkazali za toge in neprilagodljive. Razvoj tako imenovanih "inteligentnih" naprav, to je senzorjev, aktuatorjev, krmilnikov in podobnih naprav, ki so sposobne samostojno zajemati in postavljati signale, prenos velikih količin raznovrstnih podatkov, prožnost proizvodnje, združevanje proizvodnih in poslovnih informacijskih sistemov ter sprejemljivi stroški izvedbe in vzdrževanja so zahteve, ki jih takšni komunikacijski sistemi težko zagotovijo. V zadnjih letih je napredoval razvoj komunikacijskih tehnologij vseh vrst. Njihova morda najbolj pomembna lastnost je, da so splošnega značaja. V sodobnih komunikacijskih sistemih je postal računalnik nepogrešljiv sestavni del, prinesel pa je tudi nov način razmišljanja.

Kljub temu, da se komunikacijski sistemi bistveno razlikujejo v tem, kakšno informacijo prenašajo in čemu so namenjeni, pa temeljijo na podobnih osnovnih načelih. Ta načela bomo skušali spoznati v tem delu.

1.1 Osnovni gradniki komunikacijskih omrežij

Osnovna elementa omrežja sta *vozlišče* (ang. Node) in *povezava* (ang. Link). Vozlišče je naprava, ki v omrežju opravlja komunikacijske naloge. Večkrat je za vozlišče bolj primeren izraz *postaja*. Postaja je vozlišče, ki opravlja poleg komunikacijskih še druge naloge, ki niso neposredno vezane na komunikacijo, na primer daljinska postaja, delovna postaja, grafična postaja, ipd. V tem primeru je vozlišče sestavi del postaje, ki je priključena na omrežje.

Komunikacijsko omrežje je sistem med seboj povezanih vozlišč. Osnovni način povezovanja vozlišč je povezavanje *točka-točka*[1] (ang. Point-To-Point). Povezava točka-točka neposredno povezuje dve sosednji vozlišči med seboj, kot je narisano na sliki 1.



Slika 1: Povezava točka-točka.

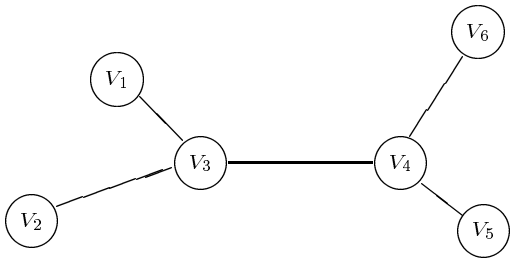
Komunikacijsko omrežje torej združuje množico vozlišč. V primeru, da je

[1] Tudi točka s točko ali točka v točko.

vsako vozlišče neposredno povezano s svojo povezavo z vsakim drugim vozliščem, nastane popolnoma povezano omrežje. V primeru, da je v popolnoma povezanem omrežju N vozlišč, gre iz vsakega vozlišča $(N-1)$ povezav, skupno število povezav pa je:

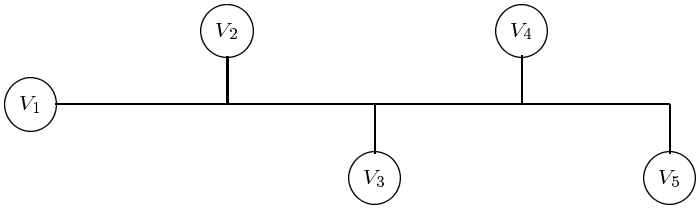
$$(N-1)+(N-2)+(N-3)+\cdots+2+1=\frac{(N-1)\times N}{2}.$$

Popolnoma povezano omrežje je smiselno za majhno število vozlišč. Z naraščanjem števila vozlišč število povezav strmo narašča. V delno povezanem omrežju si dve ali več vozlišč deli skupno povezavo, kot je narisano na sliki 2.



Slika 2: Končna (V_1, V_2, V_5 in V_6) in vmesna (V_3 in V_4) vozlišča v delno povezanem omrežju.

V narisanem primeru vozlišče V_1 komunicira z vozliščem V_6 posredno preko vozlišč V_3 in V_4. Komunikacijsko omrežje v splošnem sestavljata dve vrsti vozlišč: *končna* in *vmesna*[2] vozlišča. V končnem vozlišču informacija nastaja ali pa se koristi. V vmesnih vozliščih informacija niti ne nastaja niti se ne koristi. Vmesno vozlišče deluje kot posrednik informacije: informacijo sprejme, jo po potrebi začasno shrani, preoblikuje in pošlje naprej v zahtevani smeri. Na primer, končna vozlišča so telefoni, senzorji, aktuatorji, daljinske postaje, središče vodenja in tako dalje. Vmesna vozlišča so ponavljalniki (ang. repeaters), mostovi (ang. bridges), usmerjevalniki (ang. routers), koncentratorji, prehodi (gateways), ipd. Zaporedje povezav od enega končnega vozliša preko vmesnih vozlišč do drugega končnega vozlišča sestavlja prenosno pot.



Slika 3: Večtočkovno povezovanje vozlišč.

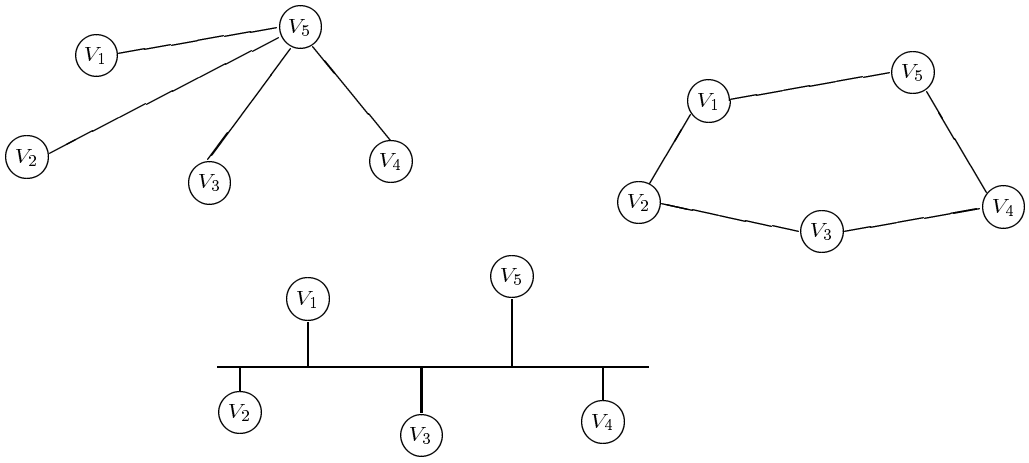
Druga možnost za povezovanje vozlišč je večtočkovno povezovanje (ang. Multi-point ali Multi-drop). Slika 3 prikazuje takšen način povezovanja. Obstajata torej dve osnovi vrsti povezovanja: povezovanje točka-točka in večtočkovno

[2]Tudi komunikacijska vozlišča.

povezovanje. Večtočkovno povezovanje se je zaradi številnih dobrih lastnosti močno uveljavilo v industrijskih omrežjih.

1.2 Osnovne oblike omrežij

S povezovanjem vozlišč nastane omrežje določene oblike. Osnovne oblike (topologije) delno povezanih komunikacijskih omrežij so: zvezda, obroč (tudi znaka) in vodilo. Na sliki 4 so skicirane osnovne oblike omrežij. Tipično komunikacijsko omrežje je *splošne ali mešane oblike* in je kombinacija omrežij osnovnih oblik (Slika 5).

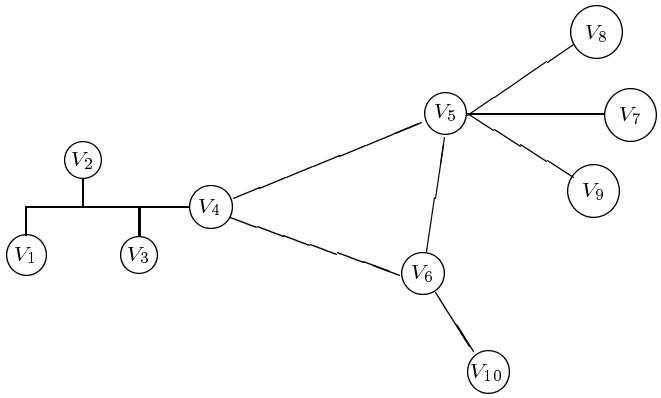


Slika 4: Osnovne oblike omrežij: zvezda, obroč (zanka) in vodilo.

V zvezdastem in zankastem omrežju so povezave med vozlišči tipa točka-točka. V omrežju zvezdaste oblik (ang. Star) komunicirajo zunanja (končna) vozlišča preko osrednjega (vmesnega) vozlišča. Pretok informacije v omrežju ureja osrednje vozlišče. V omrežju, ki je po obliki obroč oz. zanka (ang. Ring, Loop) informacija kroži v predvideni smeri od vozlišča do vozlišča. Nadzor nad pretokom podatkov v obroču je lahko centraliziran ali decentraliziran. V decentraliziranem obroču so si s tega stališča vsa vozlišča enakovredna, v centraliziranem pa ureja pretok podatkov eno od vozlišč.

V omrežju tipa vodilo (ang. Bus) se koristi večtočkovno povezovanje, zato ga imenujemo tudi večtočkovno omrežje. Večtočkovno omrežje je tipičen primer omrežij z množičnim dostopom (ang. Mutiple-Access), ki se med seboj razlikujejo ravno po tem, kako se vozliščem dodeljuje ali kako vozlišča dostopajo do skupnega prenosnega sredstva (medija). Večtočkovna omrežja se zaradi številnih dobrih

lastnosti veliko uporabljajo v industrijskih okoljih za povezovanje enostavnejših naprav (senzorjev, aktuatorjev, programljivih krmilnikov, i.t.d.).

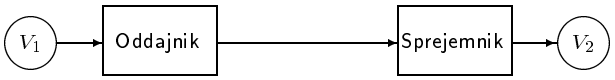


Slika 5: Splošna oblika omrežja. Vozlišča V_4, V_5 in V_6 so vmesna vozlišča in tvorijo komunikacijski podsistem, na katerega so vezana končna vozlišča $V_1, V_2, V_3, V_7, V_8, V_9$ in V_{10}.

Večja omrežja se le redko kdaj javljajo v eni od osnovnih oblik. Omrežje, ki je sestavjeno iz omrežij zvezdaste in večtočkovne oblike ter obroča, je mešano omrežje.

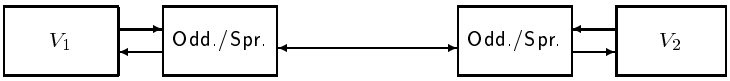
1.3 Smernost prenosa

Glede na možnost sočasnosti prenosa podatkov v obe smeri med dvema oddaljenima napravama govorimo o treh vrstah prenosa. Imenujemo jih enosmerni prenos ali simpleks, pol-dvosmerni ali poldupleks (ang. half-duplex ali HDX) in dvosmerni prenos ali dupleks (ang. full-duplex ali FDX). Prenos vrste *simpleks* omogoča prenos podatkov le v eni smeri, kot to ponazarja slika 6. Tak način prenosa pride v poštev pri zbiranju podatkov.



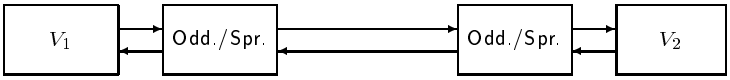
Slika 6: Prenos podatkov vrste simpleks.

Prenos podatkov vrste *poldupleks* omogoča prenos podatkov v obe smeri, vendar pa ne istočasno. Naprava, ki je sporočilo oddala, se mora 'preklopiti' (spremeniti način delovanja) z oddaje na sprejem in počakati na odgovor. Ko sprejme odgovor, se ponovno preklopi na oddajo in odda naslednje sporočilo, glej sliko 7. Informacija se torej prenaša enkrat v eni smeri, drugič v drugi smeri.



Slika 7: Prenos podatkov vrste poldupleks.

Prenos podatkov vrste dupleks (ali polni dupleks), omogoča prenos podatkov v obeh smereh istočasno, glej sliko 8. Realiziramo ga s prenosom po dveh vodih (štirižično) ali po enem samem vodu (dvožično) z multipleksiranjem.

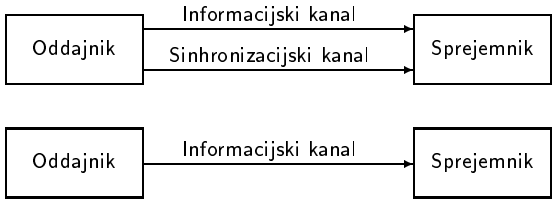


Slika 8: Prenos podatkov vrste dupleks.

1.4 Sinhronost prenosa

Pri prenosu digitalnih signalov se mora sprejemna naprava sinhronizirati z oddajno napravo. To pomeni, da mora natančno ugotoviti, kdaj se začne in kdaj konča posamezen bit, kdaj se začne in konča posamezen znak (skupina bitov), pa tudi kdaj se sporočilo začne in kdaj se konča. Glede na to, kako se sprejemnik sinhronizira z oddajnikom, govorimo o dveh vrstah prenosa, sinhronem in asinhronem prenosu (imenujejo ju tudi sinhronski in asinhronski prenos). Pri sinhronem prenosu tečeta generatorja takta sprejemnika in oddajnika *sinhrono*, torej se morata ujemati tako po frekvenci kot po fazi. Zato se od oddajnika do sprejemnika poleg informacijskega signala prenaša tudi sinhronizacijski signal, ali pa je informacijski signal tak, da se da iz njega pridobiti informacijo o taktu oddajnika (slika 9).

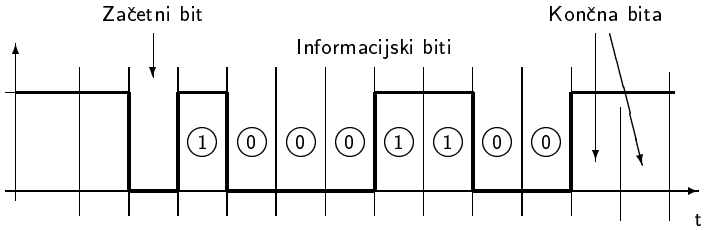
Pri asinhronem načinu prenosa tečeta sprejemni in oddajni generator asinhrono, informacija o frekvenci in fazi oddajnika pa se ne prenaša. Zato pa morata biti frekvenci sprejemega in oddajnega generatorja enaki.



Slika 9: Razlika med sinhronim (zgoraj) in asinhronim (spodaj) načinom prenosa.

1.4.1 Asinhroni prenos

Pri asinhronem (ali asinhronskem) prenosu se sinhronizacija sprejemnika z oddajnikom opravlja na nivoju krajšega zaporedja bitov (na nivoju osmih bitov ali znaka). Vsak znak v sporočilu ima svoj 'okvir' ali takoimenovane 'sinhronizacijske bite'. To so začetni bit (START bit) in eden, eden in pol ali dva končna bita (STOP bita), glej sliko 10.

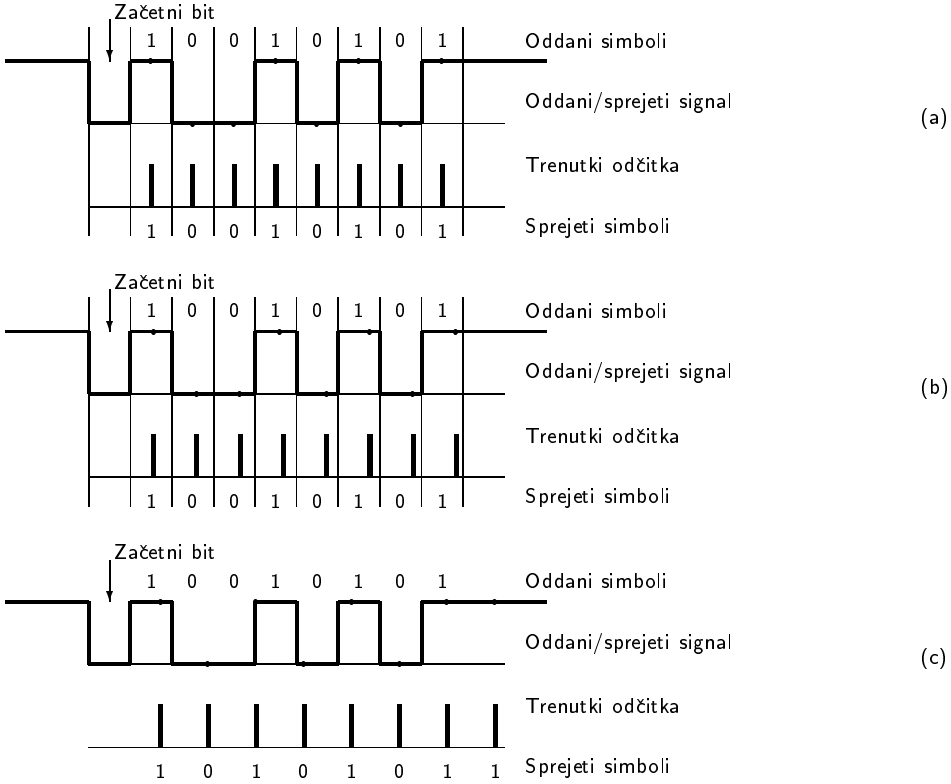


Slika 10: Primer signala za asinhroni prenos z začetnim, z osmimi informacijskim biti in dvema končnima bitoma.

Hitrost oddajanja se mora ujemati s hitrostjo sprejemanja, ujemati pa se mora tudi število bitov na znak in število končnih bitov oddajnika in sprejemnika. Ker ima vsak znak označen svoj začetek in svoj konec so lahko časovni presledki (premori) med zaporednimi znaki v sporočilu poljubno dolgi. Asinhroni prenos se uporablja pri nižjih hitrostih prenosa. Primeren je tam, kjer prihajajo podatki v nepredvidljivih časovnih presledkih. Njegova slabost je nizek izkoristek prenosne poti zaradi časovnih presledkov med zaporednimi znaki in zaradi relativno velikega števila sinhronizacijskih (odvečnih) bitov. Sprejemnik se sinhronizira z oddajnikom tedaj, ko zazna prisotnost začetnega bita (slika 11.a), sicer pa teče njegov generator takta, ki določa trenutke odčitkov, popolnoma asinhrono z oddajnikom. Če se hitrost sprejemnika preveč razlikuje od hitrosti oddajnika, se sprejeti signal odčita napačno, (glej sliki 11.b in 11.c). Končni biti so potrebni za izravnavo razlike med hitrostjo oddajnika in hitrostjo sprejemnika. Pravzaprav določajo najkrajši časovni presledek med zaporednima znakoma, sicer pa je presledek poljubno daljši.

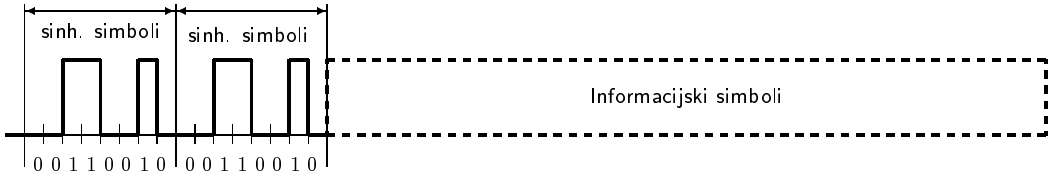
1.4.2 Sinhroni prenos

Pri sinhronem (ali sinhronskem) prenosu podatkov prenašamo znak za znakom brez vmesnih presledkov, sinhronizacija sprejemnika z oddajnikom pa se vzpostavi pred prenosom daljšega zaporedja znakov (okvirja) in se nato vzdržuje cel čas prenašanja. Za sinhronizacijo služi predpisano zaporedje uvodnih znakov, ki jim pravimo sinhronizacijski znaki. Oddajna naprava pred začetkom oddajanja vsebine okvirja najprej oddaja zaporedje enega ali več sinhronizacijskih znakov,



Slika 11: Asinhroni prenos podatkov. a) Oddajna hitrost je enaka sprejemni hitrosti: sprejemnik pravilno odčita informacijski signal. b) Oddajna hitrost je višja od sprejemne: odčitki sprejemnika so še pravilni. c) Oddajna hitrost je previsoka: sprejemnik napačno odčita sprejeti signal.

sinhronizacijske znake pa po potrebi odda tudi sredi oddajanja samega okvirja. Sinhronizacijski znaki so domenjeni, tako da jih sprejemna naprava 'spozna' in loči od koristne informacije. Ko sprejemna naprava sprejme sinhronizacijski znak, ga izloči iz koristne informacije in skuša sinhronizirati svoj generator urnih impulzov z generatorjem oddajnika oziroma nastaviti hitrost (in fazo) urinih impulzov svojega generatorja.



Slika 12: Sinhroni prenos podatkov: najprej gredo sinhronizacijski simboli in za njimi daljše zaporedje informacijskih simbolov.

Pred začetkom oddajanja okvirja (ang. frame) se znaki zbirajo v vmesnem pomnilniku oddajne naprave in naprava med tem ne potrebuje prenosne poti (kanala). Šele ko je okvir v celoti pripravljen, ga oddajna naprava odda.

Dolžina okvirja je lahko od nekaj znakov do nekaj tisoč znakov. To je odvisno od velikosti medpomnilnikov, od lastnosti prenosne poti, od hitrosti prenosa in izvedbe komunikacijskih naprav. Možna je stalna ali spremenljiva dolžina okvirjev. Spremenljiva dolžina omogoča boljšo izrabo prenosne poti. V tem primeru mora biti na koncu okvirja še znak za konec okvirja ali pa mora biti na začetku okvirja podatek o številu znakov. Običajno sledi končnemu znaku še znak za preverjanje pravilnosti prenosa.

1.5 Podatek in informacija

V vsakodnevnem pogovoru največkrat ne razlikujemo med podatkom in informacijo, vendar med njima obstaja razlika, ki je včasih zelo pomembna. Informacijo v tehniki opredelimo z zmanjšanjem nedoločenosti oziroma negotovosti o določeni stvari. Z zmanjšanjem nedoločenosti torej nastane informacija. Podatek ni nujno povezan z zmanjšanjem nedoločenosti. Na primer, izjava ob lepem vremenu *danes je lepo vreme*, ni informacija. Je pa podatek, le da je njegova informacijska vsebina majhna, praktično nič. Rečemo lahko, da je podatek nosilec informacije oziroma da vsebuje informacijo, ki pa je lahko v skrajnem primeru tudi nič. Podatek je simboličen zapis informacije.[3]

1.6 Bit in binarni simbol

Izraz bit ima dva pomena. Po eni strani je *bit* enota za množino informacije. Informacijo enega bita dobimo z odgovorom na vprašanje, na katerega sta možna dva enakoverjetna odgovora. Ali, informacijo enega bita nosi signal z dvema enakoverjetnima stanjema, t.j. dvovrednosti ali binarni signal.

Bit ima še drugi pomen. Z njim na kratko imenujemo binarni simbol (ang. BIT = BInary digiT[4]). Binarni simbol je eden od dveh možnih simbolov binarne abecede. Z njim je možno zapisati 1 bit informacije. [5] Tudi mi bomo izraz bit uporabljali v obeh oblikah, kot enoto za množnino informacije in kot ime za binarni simbol, le v primeru dvoumnosti se bomo zatekli k daljšemu izrazu binarni simbol.

[3]Slovar slovenskega knjižnega jezika: (1) Podatek je dejstvo, ki nam o določeni stvari kaj pove ali se nanjo nanaša, (2) Podatek je informacija v obliki primerni za računalnik.

[4]Digit v angleškem jeziku pomeni prst ali cifro pri štetju do deset. Binary Digit je dvojiška cifra oziroma dvojiški simbol.

[5]Ko rečemo, da je nek podatek sestavljen iz osmih bitov, s tem ne mislimo, da vsebuje informacijo osmih bitov, ampak da ga sestavlja osem binarnih simbolov. Res pa je, da bi lahko vseboval tudi toliko informacije.

1.7 Množina informacije in entropija signala

Hartley je leta 1928[6] predlagal, da bi signalu S, ki zmore zavzeti eno od N diskretnih vrednosti ali stanj $s_i, (i = 1, 2, \ldots, N)$ pripisali informacijsko vsebino

$$I(S) = \log_2 N \ \ [bitov]. \tag{1}$$

Po definiciji (1) je informacijska vsebina tem večja, čim več stanj zmore zavzeti signal,

$$N_1 < N_2 \Longrightarrow (I_1 = log_2 N_1) < (I_2 = log_2 N_2). \tag{2}$$

Vzemimo, da signal z N stanji opazujemo v G trenutkih, tako da poznamo G med seboj *neodvisnih* realizacij $S^{(t)}$ signala,

$$S_G = S^{(1)}, \ S^{(2)}, \ldots, S^{(t)}, \ldots, S^{(G)}. \tag{3}$$

Možnih je N^G različnih in enako verjetnih načinov, kako se to zgodi. Zato je

$$I(S_G) = log_2 N^G = G \times log_2 N = G \times I(S). \tag{4}$$

Izraz (4) pravi, da iz zaporedja G med seboj neodvisnih opazovanj signala S dobimo G-krat toliko informacije, kot jo dobimo iz enega samega opazovanja. To smo tudi pričakovali.

Informacijsko vsebino signala se da razložiti tudi takole. Vzemimo, da zmore signal S zavzeti eno od N stanj oziroma vrednosti, vendar zaenkrat ne vemo, v katerem od stanj se v resnici nahaja. O tem obstaja dvom ali nedoločenost. Čim več je takih stanj, tem večji je dvom (težje uganemo, v katerem stanju je signal), zato pa se nedoločenost tem bolj zmanjša, ko zvemo, v katerem stanju je. Z drugimi besedami, ko zvemo, v katerem od stanj se signal dejansko nahaja, dobimo o signalu informacijo. Sprejeta informacija je tem večja, čimbolj se zmanjša nedoločenost, oziroma čim večja je bila nedoločenost pred tem.

Hartleyevi izsledki veljajo za signale z enakoverjetnimi stanji. E.C. Shannon je Hartleyeve izsledke posplošil in leta 1948 [1] postavil temelje sodobne teorije informacij.

Shannon je izhajal iz predpostavke, da je informacijska vsebina signala (in z njo povezana nedoločenost) odvisna od verjetnostne porazdelitve stanj. Stanja z veliko verjetnostjo so manj nedoločena in zato nosijo manj informacije od tistih, ki imajo majhno verjetnost. Nedoločenost *posameznega stanja* signala je definiral z logaritmom njegove verjetnosti:

$$h(s_i) = log_2 \frac{1}{p_i}, \quad (i = 1, 2, \ldots, N). \tag{5}$$

[6] R.V.L. Hartley, Transmission of Information, *Bell System Technical Journal*, vol.7, pp.535–563, 1928.

Stanja z manjšo verjetnostjo so po tej definiciji bolj nedoločena, jih je težje uganiti ali napovedati in zato nosijo tudi več informacije. Shannon je definiral še povprečno nedoločenost $H(S)$ na stanje signala S (nakljućne spremenljivke – signal),

$$H(S) = \sum_{i=1}^{N} p_i \, log_2 \; h(s_i) = \sum_{i=1}^{N} p_i \, log_2 \; \frac{1}{p_i}. \tag{6}$$

Ker je $\log_2 1/p_i = -log_2 p_i$, dobimo:

$$H(S) = -\sum_{i=1}^{N} p_i \; log_2 \; p_i \tag{7}$$

Kadar nas dejanska vrednost izraza (7) ne zanima, hočemo pa vseeno poudariti, da je njegova vrednost odvisna od porazdelitvenega zakona, zapišemo zraven porazdelitveni zakon:

$$H(S) = H(p_1, p_2, \ldots, p_i, \ldots, p_N). \tag{8}$$

Povprečno nedoločenost na stanje imenujemo *entropija* signala. Entropija je *nenegativna* funkcija verjetnostne porazdelitve signala, in narašča z N. Velja tudi pravilo seštevanja entropij. Kaj lahko se prepričamo, da je entropija $H(S_G)$ zaporedja G (*neodvisnih*) opazovanj signala:

$$S_G = S^{(1)}, \; S^{(2)}, \ldots, S^{(t)}, \ldots, S^{(G)} \tag{9}$$

G-krat večja od entropije enega samega opazovanja:

$$H(S_G) = G \times H(S). \tag{10}$$

Entropija je merilo za 'nered' signala. Entropija je pri danem N največja, ko je signal najbolj 'neurejen'. To je tedaj, ko so vsa stanja enako pričakovana - enako verjetna. V tem primeru velja:

$$H(S) = -\sum_{i=1}^{N} p_i \; log_2 p_i = -\sum_{i=1}^{N} p_i \; log_2 \frac{1}{N} = logN, \tag{11}$$

kar sovpada s Hartleyevo definicijo.

Informacijska vsebina signala ali *množina informacije* je povezana z entropijo. Čim večja je entropija signala, tem več informacije nosi signal. Naj bo njegova entropija v danem trenutku enaka $H(S)$. Ko spoznamo signal, se njegova nedoločenost zmanjša na nič. S tem dobimo (v povprečju)

$$I(S) = H(S), \tag{12}$$

informacije. To pomeni, da bi pri opazovanju signala skozi daljši čas dobili z vsako realizacijo signala v povprečju $H(S)$ informacije.

1.8 Informacijski pretok in kapaciteta kanala

Leta 1924 je H. Nyquist dokazal, da se da poljuben frekvenčno omejen signal s frekvenčno vsebino 0 do F Hz popolnoma rekonstruirati, če signal vzorčimo v enakomernih časovnih presledkih $t_s = 1/(2 \times F)$. Edini pogoj za popolno rekonstrukcijo je, da vzorci signala niso moteni s šumom. Vzorčenje z višjo frekvenco ne prispeva h kvaliteti rekonstrukcije. Povedano drugače, $2 \times F$ vzorcev na sekundo vsebuje vso informacijsko vsebino signala. Edini pogoj za popolno rekonstrukcijo je, da vzorci signala niso moteni s šumom. Nyquistovo spoznanje je temeljnega pomena za prenos informacije po brezšumnem informacijskem kanalu[7] (prenosni poti) z omejeno frekvenčno širino. Da je signal s frekvenčno vsebino $0 - F$ še možno neokrnjeno prenašati po kanalu, mora biti frekvenčna širina kanala enaka F. Tak signal popolnoma opišemo z $2 \times F$ signalnimi elementi na sekundo. Če je za vsak element signala možnih V diskretnih vrednosti (stanj), lahko vsak nosi največ $log_2\ V$ bitov informacije. Najvišja možna hitrost prenašanja informacije potem je:

$$C = 2 \times F \times log_2\ V \quad \frac{\text{bitov}}{\text{sekundo}}. \tag{13}$$

Najvišjo možno hitrost prevajanja informacije imenujemo kapaciteta informacijskega kanala in jo običajno označimo s C. Na primer, po brezšumni prenosni poti s frekvenčno širino 3000 Hz (frekvenčna širina telefonskega voda) teoretično ne moremo prenašati dvovrednostnega (binarnega) signala z višjo hitrostjo od

$$C = 2 \times 3000 \times log_2\ 2 = 6000\ \frac{b}{s},$$

v praksi pa je ta hitrost zaradi prisotnosti šuma in nepopolnosti komunikacijskih naprav še nižja.

Kapaciteto kanala C moramo razumeti kot lastnost, ki je dana z izvedbo prenosne poti in ni odvisna od dejanske hitrosti prenosa. Dejansko hitrost prenosa narekuje oddajna naprava in je lahko poljubno nižja od kapacitete kanala. Oddajna naprava ne sme nikoli oddajati hitreje kot je kapaciteta kanala. V nasprotnem primeru je izgubljanje informacije neizbežno.

Enačba (13) pravi, da se da doseči višjo hitrost prenosa pri isti frekvenčni širini kanala, če za prenos koristimo večvrednostni signal. Denimo, da je $V = 256$. V tem primeru lahko vsak signalni element nosi osem bitov informacije. Kapaciteta kanala pri enaki frekvenčni širini kot prej potem je

$$C = 2 \times 3000 \times log_2\ 256 = 6000 \times 8 = 48000\ \frac{b}{s}.$$

[7] *Informacijski kanal* je sredstvo, medij ali predpis za prenos informacije skozi prostor in/ali čas.

Je pa v tem primeru na sprejemni strani težje razpoznati odposlano vrednost signala - signal je težje ločiti od šuma. Enačba (13) ne pove ničesar o tem, kako moč šuma vpliva na kapaciteto kanala. Dejstvo je, da z naraščanjem moči šuma kapaciteta upada.

C. E. Shannon je ugotovil (1948), da je kapaciteta kanala s pasovno širino F v prisotnosti aditivnega gaussovega šuma (oziroma ob predpostavki, da sta tako signal kot šum porazdeljena normalno) in pri razmerju *moč signala/moč šuma* = S/N podana z enačbo:

$$C = F \times log_2\ (1 + \frac{S}{N}). \tag{14}$$

Iz enačbe vidimo, da je pri manjši frekvenčni širina kanala manjša tudi njegova kapaciteta. Podobno, če moč šuma raste, potem kapaciteta pada. Kapaciteto kanala se da povečati pri istem razmerju $\frac{S}{N}$ z večanjem frekvenčne širine ali pa pustimo frekvenčno širino nespremenjeno in povečamo moč signala ali pa zmanjšamo šum. Pri tem ni važno, koliko nivojev (stanj) zmore zavzeti informacijski signal. Shannonova formula kapacitete kanala neposredno ne povezuje s številom možnih stanj (nivojev) signala, vendar se jo da tolmačiti tudi na ta način. Zapišimo Shannonovo enačbo še enkrat v nekoliko drugačni obliki:

$$C = F \times log_2\ (1 + \frac{S}{N}) = 2 \times F log_2 \sqrt{\frac{S+N}{N}}$$

Enačni (13) in (14) postaneta identični za $V = \frac{\sqrt{S+N}}{\sqrt{N}}$ iz česar sledi, da pri dani moči signala moč šuma pravzaprav določa število razpoznavnih stanj signala.

Za številčni primer vzemimo, da je frekvenčni pas kanala 3000 Hz in da je razmerje signal/šum = 1000 (nekako tako, kot pri telefonskem vodu). Teoretično največja možna hitrost prenosa podatkov po kanalu znaša:

$$C = 3000 \times log_2(1 + 1000) \approx 3000 \times 10 \approx 30000\ \frac{b}{s}.$$

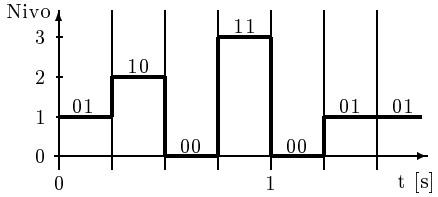
Gornjo formulo je treba razumeti kot teoretično zgornjo mejo hitrosti prenosa, kar pa ne pomeni, da je takšno hitrost tudi enostavno doseči.S sodobnimi postopki kodiranja in modulacije se danes dosega 33600 $[b/s]$ [8], dejanska hitrost prenosa pa seveda zavisi od kvalitete telefonskih linij.

1.9 Baud in bit na sekundo

Število (možnih) sprememb informacijskega signala na časovno enoto podajamo v **baud**ih (izgovorimo bod).[9] V baudih izražamo hitrost spreminjanja signala

[8] Priporočila ITU-T V.90, ITU-T V.34

[9] Baud (**Bd**) je dobil ime po francozkem strokovnjaku E. Baudot-u kot priznanje za njegove dosežke na področju telekomunikacij v preteklem stoletju.



Slika 13: Štirivrednostni signal, ki zmore manjati stanje štirikrat na sekundo, signalna hitrost je 4 Baude. Ker vsako od štirih možnih stanj nosi dva bita informacije, je hitrost prenosa v bitih na sekundo enaka $4 \times 2 = 8$.

oziroma število signalnih elementov na sekundo in ji zato večkrat pravimo tudi signalna ali modulacijska hitrost. Ta nam ne pove veliko o dejanski hitrosti prenosa informacije. Na hitrost prenosa informacije v bitih na sekundo namreč ne vpliva samo hitrost spreminjanja (frekvenca) signala, temveč tudi to, koliko informacije nosi posamezen signalni element, glej enačbo (13). Zato je hitrost prenosa izražena v baudih le izjemoma številčno enaka hitrosti prenosa, ki jo podamo v bitih na sekundo.

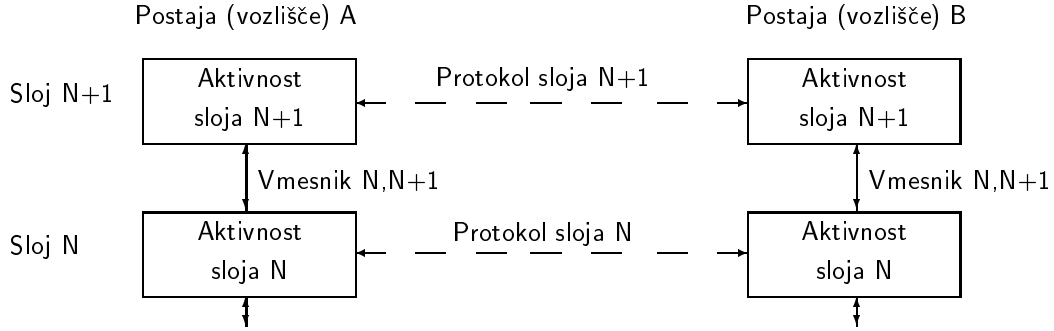
Odnos med baudi in biti na sekundo smo ponazorili na sliki 13. Vsako sekundo oddamo štiri signalne elemente. Signal zmore spremeniti stanje štirikrat na sekundo in hitrost spreminjanja signala je 4 Bd. Vsak signalni element lahko zavzame eno od štirih diskretnih vrednosti (stanj). Z vsakim stanjem zapišemo (zakodiramo) dva binarna simbola (dva bita informacije). Vsako sekundo lahko s takim signalom prenesemo $4 \times 2 = 8$ bitov informacije in hitrost prenosa informacije je zato osem bitov na sekundo.

Torej, ko rečemo, da prenašamo podatke s hitrostjo 2400 baudov, mislimo s tem povedati, da zmore signal spremeniti stanje 2400-krat na sekundo oziroma, da prenašamo 2400 signalnih elementov na sekundo. V primeru, da je signal večvrednosten (denimo V = 16), nosi vsak signalni element več bitov informacije (za V = 16 štiri bite) in najvišja možna hitrost prenosa v bitih na sekundo postane (štirikrat) višja od tiste, s katero se spreminja signal. V tem primeru bi morali torej povedati, da prenašamo 9600 bitov na sekundo z 2400 baudi. Hitrost prenosa izražena v baudih se številčno ujema s hitrostjo izraženo v bitih samo tedaj, ko je signal dvovrednosten (binaren).

2 Arhitektura omrežij

2.1 Slojnost omrežij

Komunikacijska omrežja sodijo med kompleksne sisteme. Da bi zmanjšali zahtevost in stroške snovanja, izgradnje, obratovanja, vzdrževanja in uporabe takih sistemov, je večina sodobnih omrežij zasnovana modularno kot sestav *slojev*, pri čemer je vsak višji sloj nadgrajen nad neposredno nižji sloj. Skupna lastnost sodobnih omrežij je slojna urejenost postopkov in naprav. Osrednji element v tej hierarhični večslojni zgradbi omrežja je *sloj* (Ang. Layer), imenovan tudi plast ali nivo (Ang. Level). Število slojev, imena slojev, kot tudi vsebina in funkcija posameznih slojev, se od omrežja do omrežja razlikujejo. Kar je značilno za sloj v vsakem omrežju je predvem to, da nudi storitve neposredno višjemu sloju, pri tem pa skrije podrobnosti o izvedbi storitev pred uporabnikom storitev. Sloj, ki nudi storitev, se imenuje dajalec storitve (Ang. Service provider). Sloj, ki uporablja storitev, se imenuje koristnik ali uporabnik storitve (Ang. Service user).



Slika 14: Načelo slojnosti omrežja.

Med sosednjima slojema je vmesnik. Vmesnik slojev N in $N+1$ natančno določa storitve, ki jih sloj N nudi sloju $N+1$, pa tudi osnovne operacije za izvedbo teh storitev.

Proces sloja N na eni postaji komunicira s procesom sloja N na drugi postaji. Pravimo, da komunikacija v omrežju poteka med istorodnimi procesi (ang. peer-to-peer). Pravila in dogovori, ki jih upoštevata istorodna procesa na obeh straneh na sloju N, imenujemo s skupnim imenom *komunikacijski protokol* sloja N. Prenos podatkov med postajama na istem sloju (v vodoravni smeri) pa je zgolj navidezen. Resnični tok podatkov gre v navpični smeri. Višji sloj preda podatke (skupaj z nadzorno informacijo) naslednjemu nižjemu sloju, ta pa zopet nižjemu. Končno pride na najnižjem sloju po prenosne poti do dejanskega prenosa informacije z ene na drugo stran, kjer gre informacija od sloja do sloja navzgor - v navpični smeri. Dejanski vodoraven prenos informacije je prisoten samo na najnižjem nivoju - na nivoju prenosne poti.

Množica slojev in protokolov ter vmesnikov, oziroma storitev, ki so dane na vmesnikih, določa *arhitekturo omrežja*. Določila mrežne arhitekture morajo zadoščati za realizacijo vseh postopkov in naprav. Sam način izvedbe protokolov in vmesnikov ne sodi k arhitekturi. Ta je z arhitekturnega stališča popolnoma svoboden. Arhitektura omrežja je torej povezana z vprašanjem *kaj*, ne pa *kako*.

2.1.1 Primer večslojnosti komuniciranja

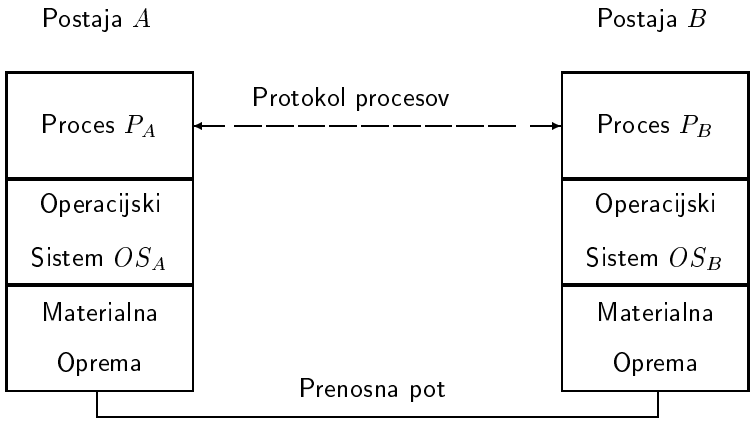
Da bi se znebili občutka nestvarnosti večslojne komunikacije in da bi razjasnili razliko med vodoravno in navpično komunikacijo, si zamislimo naslednji netehnični primer. Denimo, da želi slovenski diplomat D_S komunicirati s finskim diplomatom D_F. Med diplomati obstajajo dogovorjena pravila (protokol diplomatov), ki se jih je morata oba držati. Če želi vsak od diplomatov komunicirati v svojem jeziku, sta potrebna slovenski P_S in finski P_F prevajalec. Prevajalca s prevajanjem nudita storitve (prevajanje) diplomatoma. Kako in v kakšnem jeziku (recimo francoskem) se v resnici sporazumevata prevajalca (protokol prevajalcev), diplomatov ne zanima. Za oba diplomata je važno le, da jima prevajalca omagočata komunikacijo. Prevajalca v tem primeru realizirata storitve, ki jih dajeta diplomatoma (višjemu nivoju) s komunikacijo na svojem nivoju - s protokolom prevajalcev.

Diplomata dejansko ne govorita neposredno drug z drugim, ampak vsak s svojim prevajalcem. V resnici teče informacja v navpični smeri (vmesnik diplomat-prevajalec). Kljub temu se diplomata vedeta kot da govorita neposredno drug z drugim. Dejstvo, da govorita vsak s svojim prevajalcem, je za vsebino njunega pogovora nepomembno.

Tudi za prevajalca obstaja več možnosti komunikacije. Lahko komunicirata eden z drugim po telefonu, po običajni pošti, po elektronski pošti, i.p.d. Na tem, v našem primeru najnižjem nivoju, se zgodi dejanski prenos informacije med diplomatoma.

Poglejmo bolj tehnični primer. Razmislimo, kako razmišlja programer, ki se ukvarja s problemom komuniciranja procesa P_A s procesom P_B na krajevno ločenih postajah A in B, ki sta na nek način povezani, glej sliko 15.

Postaji delujeta pod nadzorom operacijskega sistema (OS). Problematiko komuniciranja bi programer na primer razčlenil na vzpostavljanje zveze med procesoma, na prenos podatkov v obe smeri in na sproščanje zveze, ko ta ni več potrebna. Njegov program bi med drugim verjetno obsegal podprograme z imeni kot so: *VzpostaviZvezo(Proces)* , *PošljiPodatke(Proces)*, *SprejmiPodatke(Proces)* in *SprostiZvezo(Proces)*. Programer pri tem očitno razmišlja tako, kot da procesa P_A in P_B dejansko komunicirata neposredno drug z drugim v vodoravni smeri. Dejstvo, da procesa v resnici komunicirata 'navzdol' z operacijskim sistemom je na tem nivoju abstraktnosti drugotnega pomena. Je pa zagotovo



Slika 15: Večslojnost komuniciranja, vodoraven in navpičen prenos podatkov.

pomembno pri realizaciji podprogramov. Kar vzemimo primer realizacije podprograma *PošljiPodatke(Proces)*. Naloga podprograma je, da poskrbi za prenos določenih podatkov od enega procesa do drugega procesa. Programer za realizacijo podprograma (realizacijo prenosa podatkov) uporablja tisto, kar mu nudi operacijski sistem. Na primer, podatke, skupaj z nadzorno informacijo preda operacijskemu sistemu, ki s krmiljenjem ustrezne materialne opreme poskrbi za dejanski prenos podatkov. Podprogram torej 'pošlje podatke navzdol' operacijskemu sistemu in ne procesu na drugi strani. Kako operacijski sistem poskrbi za dejanski prenos podatkov na drugo stran, je programerju (podprogramu) popolnoma prikrito in je zanj na tem nivoju tudi nepomembno. Po drugi strani pa je s stališča operacijskega sistema popolnoma nepomembno, kaj pomenijo podatki, ki jih prenaša v korist obeh procesov.

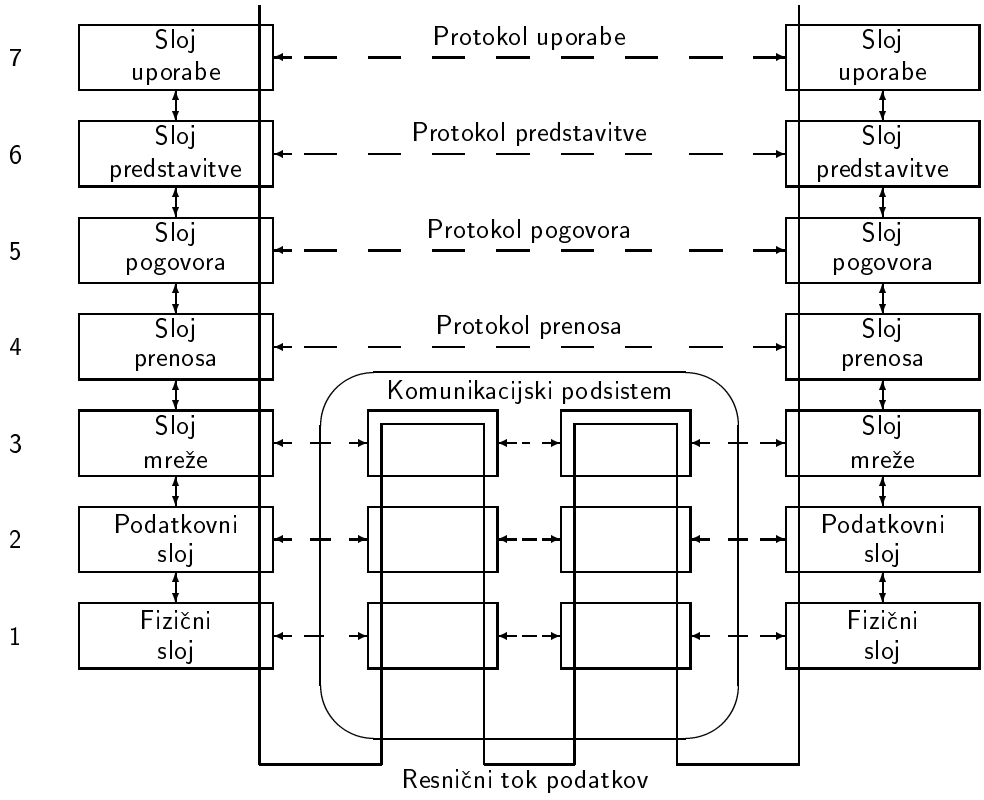
2.2 ISO OSI referenčni model

Da bi zagotovili združljivost sistemov različnih proizvajalcev ter omejili število različnih izvedb komunikacijskih postopkov in opreme, je mednaroda organizacija za standardizacijo ISO v začetku osemdesetih let predlagala model, po katerem naj bi gradili računalniška omrežja.[10] Referenčni model OSI (ang. Open Systems Interconnection), kot pove tudi ime, se nanaša na odprte sisteme, to je sisteme, ki so odprti za komunikacijo z drugimi odprtimi sistemi. Model OSI obsega sedem slojev. Do sedmih slojev so prišli ob upoštevanju naslednjih načel[2]: sloj mora opravljati natančno določeno nalogo, nov sloj je potreben tam, kjer je potreben nov nivo abstrakcije, upoštevati je treba možnosti standardizacije, sloji morajo biti izbrani tako, da se čimbolj zmanjša pretok informacije med sloji (na

[10]ISO 7498-1984: Information Processing Systems - Open systems Interconnection - Basic Reference Model.

vmesnikih), število slojev naj se zadrži v razumnih mejah, da ne bi s prevelikim številom slojev po nepotrebnem večali obsežnosti, vendar naj bo slojev dovolj tako, da ne bi bili prisiljeni združiti dveh načelno različnih aktivnosti v isti sloj.

Referenčni model ISO OSI je narisan na sliki 16. Slika prikazuje odnos med dvema končnima vozliščema, ki ju povezuje komunikacijski podsistem. Model OSI sam po sebi ni arhitektura mreže, saj ne določa niti storitev niti protokolov posameznih slojev. Določa le funkcije posameznih slojev. Res je ISO izdelal ali privzel tudi standarde za posamezne sloje, vendar ti standardi niso sestavni del modela. Za posamezne sloje se lahko izbere poljubno kombinacijo protokolov. Iz tega sledi, da dve postaji, ki sicer komunicirata po referenčnem modelu OSI, a uporabljata različen protokol vsaj na enem sloju, med seboj ne moreta[11] komunicirati. Postaje lahko med seboj komunicirajo samo, če uporabljajo enak sklad protokolov ali enak protokolovni 'profil' (ang. Protocol Stack ali Protocol Suite ali Protocol Profil), enake protokole na vseh nivojih. V praksi se nekatere bolj uveljavljene protokolovne sklade imenuje na kratko kar protokol.[12] Čeprav to ni popolnoma dosledno, bi ne smelo biti razloga za dvoumnost.



Slika 16: Referenčni model ISO OSI.

[11]Lahko s pomočjo protokolovnih prevajalnikov.
[12]Na primer TCP/IP protokol ali MAP protokol

Oglejmo si glavne naloge posameznih slojev, zaporedoma od najnižjega do najvišjega sloja.

Fizični sloj (ang. Physical Layer) skrbi za prenos informacijskih signalov po komunikacijskem kanalu. Osnovna informacijska enota tega sloja je bit. Sloj določa mehanske, električne in postopkovne lastnosti naprav in tokokrogov. Tipična vprašanja v zvezi s tem slojem so: napetostni nivoji signalov, hitrost prenosa, oblike signalov, vrste modulacij, konektorji in število priključkov na konektorjih, uporabljeni prenosni medij. Najbolj znani standardi in priporočila tega sloja so RS232, RS422, RS423, RS449, RS485, V.24, V.28, V.10, V.11, V.12, X.21, X.26, X.27, V.22, V.32, V.42, i.t.d.

Podatkovni sloj ali linijski sloj (ang. Data Link Layer) uporablja storitve fizičnega sloja (prenos signalov) in zagotavlja naslednjemu višjemu sloju (mrežnemu sloju) visoko zanesljivost prenosa podatkov v obe smeri med sosednjimi vozlišči. V ta namen deli daljša zaporedja bitov na manjše okvirje (Ang. Frames) ter skrbi za pravilen prenos posameznih okvirjev od vozlišča do vozlišča s preverjanjem pravilnosti prenešenega okvirja in s potrjevanjem pravilnosti sprejema. Tipična vprašanja v zvezi s tem slojem so: kako označiti začetek in konec okvirja, kako ugotavljati prisotnost napak na okvirjih, kako zahtevati ponoven prenos pokvarjenega okvirja in kako rešiti problem podvojenih okvirjev (dveh enakih okvirjev) v primeru, da se izgubi potrdilo že pravilno sprejetega okvirja. Tipični primeri protokolov (standardov), ki opravljajo funkcije tega sloja so: IBM-ova protokola BSC ali z drugim imenom BISYNC (Binary Synchronous Communication) in SDLC (Synchronous Data Link Control), ISO HDLC (High-Level Data Link Control), ki je povzet po SDLC. Tudi podatkovni protokol LAPB (Link Access Procedure - Balanced) v javnih podatkovnih omrežjih po priporočilu X.25 izhaja iz protokola HDLC. Za prenos podatkov med manjšimi računalniki sta zelo pribljubljena Kermit in XMODEM ter njegove izpeljanke (YMODEM, ZMODEM, i.t.d.). Naslednja važna naloga tega sloja v lokalnih omrežjih je nadzor nad dostopom do skupnega kanala (Ang. Media Access Sublayer - MAC). Morda najbolj poznani standardi s tem v zvezi so Ethernet II, IEEE 802.3, IEEE 802.4, IEEE 802.5 ter podatkovni protokol LLC (Logical Link Control) po standardu IEEE 802.2.

Mrežni sloj (ang. Network Layer) skrbi za delovanje komunikacijskega podsistema. Na sliki 16 vidimo, da vmesna vozlišča, ki tvorijo komunikacijski podsistem, opravljajo naloge samo prvih treh (spodnjih) slojev, do vključno mrežnega sloja. Osnovna podatkovna enota tega sloja je paket (Ang. Packet). Ključna naloga tega sloja je zagotavljanje *poti prenosa* od oddajnega do sprejemnega vozlišča. Pot od oddajnika do sprejemnika je lahko določena čisto statično (vnaprej z izvedbo omrežja in se nikoli ne spremeni), lahko pa je drugačna za vsako novo zvezo, za vsako novo sporočilo ali celo za vsak posamezen paket (dinamično). Mrežni sloj torej skrbi za usmerjanje paketov od vozlišča do vo-

zlišča (ang. Routing), naslavljanje vozlišč, pa tudi za to, da ne pride do preobremenjenosti dela omrežja (ang. Congestion Control), opravlja razne (statistične) izračune, ki omogoča optimizacijo omrežja in/ali obračunavanje storitev. Med najbolj znanimi protokoli mrežnega sloja sta IP (Internet Protocol) v omrežju omrežij Internet, in PLP (Packet Layer Protocol) po priporočilu X.25 organizacije CCITT. V lokalnih omrežjih tega sloja ni ali pa je zelo 'tanek'. V omrežjih z Novellovim mrežnim operacijskim sistemom NetWare opravlja (približno) funkcije mrežnega sloja protokol IPX (Internetwork Packet Exchange) ki se vzgleduje po Xerox XDN ISD protokolu.

Prenosni sloj (Ang. Transport Layer) streže sosednjemu višjemu sloju (sloju pogovora). Od njega prevzema podatke, jih po potrebi deli na manjše enote - podatkovne enote prenosnega sloja (Ang. Transport Protokol Data Unit - T-PDU),[13] zatem pa predaja naslednjemu nižjemu (mrežnemu) sloju ter skrbi, da podatki pogovornega nivoja pridejo pravilno na drugo stran. Torej skrbi za *prenos* informacije na drugo stran, kot pove tudi njegovo ime. Prenosni sloj je prvi 'pravi' sloj 'od-konca-do-konca' (ang. End-to-End). Z drugimi besedami, proces na eni končni postaji komunicira neposredno s sorodnim procesom na drugi končni postaji, brez posrednikov. Za primer protokola tega sloja navedimo protokol v omrežjih Internet TCP (Transmission Control Protocol), ki se pojavlja skupaj s protokolom IP, in ISO 8073, ki se uporablja v omrežjih MAP (Manufacturing Automation Protocol) ter TOP (Technical and Office Protocol).

Pogovorni sloj (Ang. Session Layer) omogoča 'pogovor' med uporabniškimi procesi na različnih postajah in s tem skrbi za organizacijo in strukturiranje dialoga. Podobno kot prenosni sloj omogoča prenos podatkov, poleg tega pa nudi tudi nekatere kvalitetnejše storitve specifične za določene aplikacije. Če bi za prenosni sloj rekli, da skrbi za prenos informacije med enim in drugim končnim vozliščem, skrbi pogovorni sloj za pogovor med istorodnima procesoma enega in drugega vozlišča. Večina omrežij tega sloja spoh nima. Na tem sloju je običajno (a ni nujno) realiziran sistem daljinskih klicev (Ang. Remote Procedure Call), kot eden izmed možnih načinov komunikacije med krajevno porazdeljenimi procesi.

Predstavitveni sloj (Ang. Presentation Layer) opravlja pogoste storitve, ki zahtevajo splošno rešitev za večino uporabnikov teh storitev. Tipične storitve tega sloja so: kodiranje in prekodiranje (npr. ASCII V EBCDIC), šifriranje, zgoščevanje podatkov in podobno.

Sloj uporabe (ang. Application Layer) predstavlja vmesnik med končnim uporabnikom in komunikacijskim sistemom. Vsebuje morda največ storitev in protokolov. Najbolj znane med njimi so: prenos datotek (File Transfer), navidezni terminal (Virtual Terminal), elektronska pošta (E-mail) X.400, mrežne direktorijske storitve X.500, i.t.d.

[13] Od tega sloja navzgor za PDU nimamo posebnih imen kot paket in okvir.

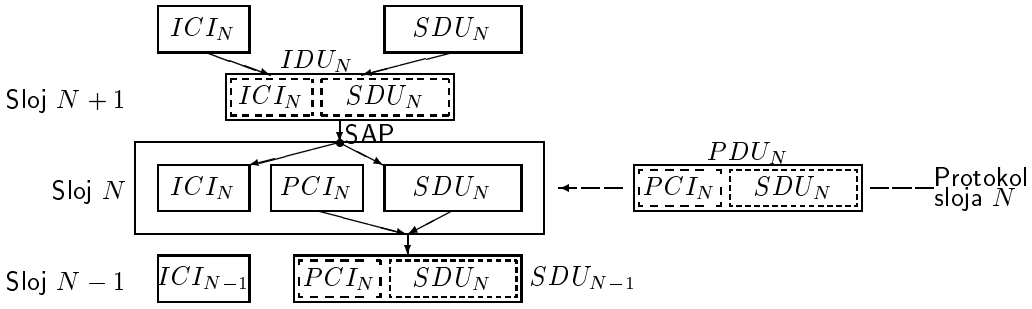
2.3 OSI, storitve, protokoli in načelo ovojnice

Temeljna naloga vsakega sloja v modelu OSI je, da daje storitve sosednjemu višjemu sloju. Aktivne elemente za izvedbo storitev imenujemo *aktivnosti* (ang. Entities). Na primer, aktivnosti prenosnega sloja imenujemo aktivnosti prenosnega sloja. Aktivnosti istega sloja na različnih postajah imenujemo istorodne aktivnosti (ang. Peer Entities). Istorodne aktivnosti sloja N skrbijo za izvedbo storitev, ki jih sloj N daje sloju $N+1$. Pri tem se poslužujejo protokola sloja N. Za realizacijo svojih storitev se sme sloj N poslužiti storitev, ki jih njemu nudi nižji sloj $N-1$.

Storitve sloja so dostopne na dostopni (ali pristopni) točki storitev (ang. Service Access Point), glej sliko 17. Dostopna točka storitve sloja N je mesto, kjer so njegove storitve dane sloju $N+1$. Pri izmenjavi informacije na vmesniku morata sosednja sloja upoštevati pravila vmesnika. Ta pravila v bistvu definirajo vmesnik. Na primer, na tipičnem vmesniku preda aktivnost sloja $N+1$ podatkovno enoto vmesnika IDU (ang. Interface Data Unit) aktivnosti sloja N. Podatkovno enota vmesnika sestavljata dva dela: podatkovna enota storitve SDU (ang. Service Data Unit) in nadzorna informacija vmesnika ICI (ang. Interface Control Information). Nadzorno informacijo vmesnika ICI potrebuje aktivnost sloja N za sebe in ni del koristnih podatkov. Iz nje razbere, kaj mora narediti za koristnika storitve. Podatkovna enota storitve SDU pa je tista informacija, ki se mora prenesti po omrežju na drugo stran k istorodni aktivnosti sloja $N+1$. Pri prenosu podatkovne enote storitve SDU na drugo stran, se istorodni aktivnosti sloja N obeh strani sporazumevata po protokolu sloja N. Podatkovne enote, ki si jih izmenjujeta preko omrežja, imenujemo protokolne podatkovne enote PDU (ang. Protocol Data Unit). Na primer, protokolna podatkovna enota sloja 2 je okvir, protokolna podatkovna enota tretjega sloja je paket in podobno. Protokolno podatkovno enoto sestavlja koristni del (SDU) in protokolna nadzorna informacija PCI ali 'glava' (ang. Protocol Control Information ali tudi Header), ki jo istorodni aktivnosti rabita za medsebojno sporazumevanje po protokolu sloja N.

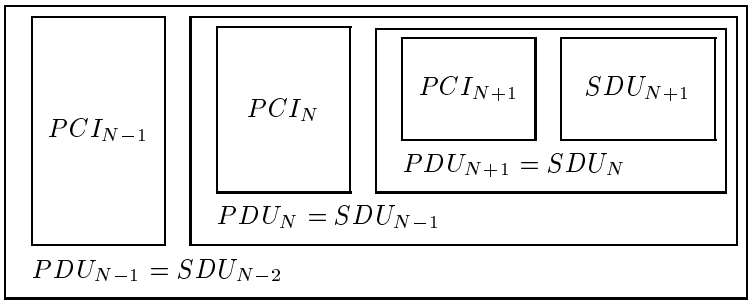
Za realizacijo storitve, ki jo sloj N daje sloju $N+1$ se sloj N poslužuje storitev sloja $N-1$ pod njim. Torej, opisane razmere, ki vladajo na vmesniku slojev $N+1$ in N se v podobni obliki ponovijo na vmesniku slojev N in $N-1$. Protokolni podatkovni enoti sloja N se doda nadzorna informacija vmesnika (ICI) in tako tvori podatkovno enoto vmesnika (IDU). Torej je protokolna podatkovna enota sloja N s stališča dajalca storitve, ki je v tem primeru sloj $N-1$, podatkovna enota storitve (SDU). V splošnem velja, da je $PDU_N = SDU_{N-1}$. Tako smo spoznali *načelo ovojnice*, glej sliko 18.

Istorodne aktivnosti sloja $N+1$ si izmenjujejo protokolne podatkovne enote PDU_{N+1}. Pri tem koristijo storitve sloja N. Aktivnosti na sloju N pa obrav-

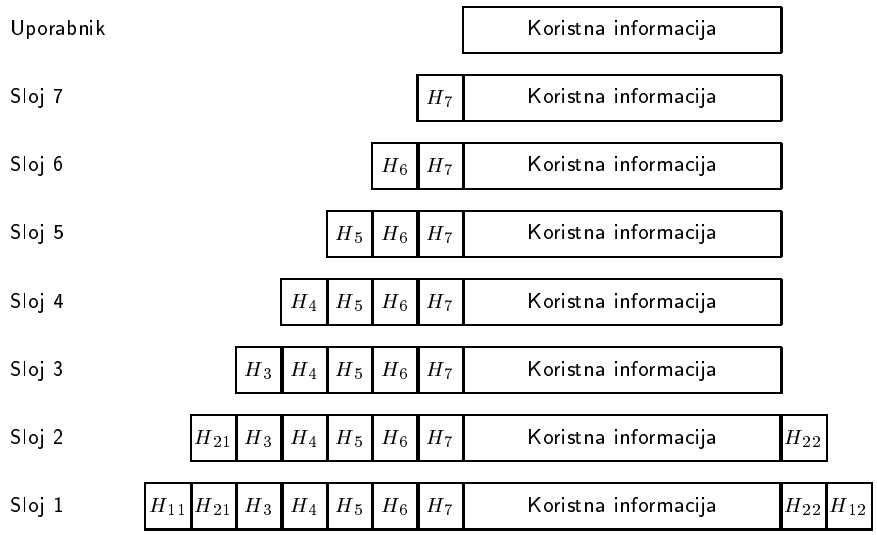


SAP (Service Access Point)	=	Dostopna točka storitve
IDU (Interface Data Unit)	=	Podatkovna enota vmesnika
ICI (Interface Control Information)	=	Nadzorna informacija vmesnika
SDU (Service Data Unit)	=	Podatkovna enota storitve
PCI (Protocol Control Information)	=	Protokolna nadzorna informacija
PDU (Protocol Data Unit)	=	Protokolna podatkovna enota

Slika 17: Podatkovne strukture in razmere na vmesnikih med sloji $N+1$, N in $N-1$.



Slika 18: Načelo ovojnice.



Slika 19: Načelo ovojnice. Koristni informaciji uporabnika se na vsakem sloju doda kontrolna informacija protokola (glava H).

navajo PDU_N zgolj kot podatke SDU_N, ki jih morajo prenesti na drugo stran, njihova vsebina pa jih ne zanima. Podatkom PDU_{N+1} pred pošiljanjem dodajo spremno informacijo (podatke 'zapre v ovojnico'), kot določa protokol na sloju N. Tako nastane protokolna podatkovna enota PDU_N, ki jo sloj N preda nižjemu sloju $N-1$, ta pa spet nižjemu, in tako dalje.

Slika 19 shematično prikazuje načelo ovojnice v sedemslojnem modelu OSI. Na vsakem sloju se koristni informaciji doda kontrolna informacija protokola tega sloja (glava) ter na koncu vse skupaj odda v kanal. Na sprejemni strani teče postopek v obratni smeri.

Pojma storitve ne smemo zamenjati s pojmom protokol. *Storitev* je določena z množico osnovnih operacij, ki jih sloj nudi višjemu sloju. Storitev določa, kaj (kakšne operacije) je nek sloj pripravljen opravljati za uporabnika storitev. Pri tem ni važno, kako so realizirane te operacije. Storitev se nanaša na vmesnik med sosednjima slojema (na navpično komunikacijo). Nasprotno temu pa se protokol nanaša na vodoravno komunikacijo med istorodnima procesoma. *Protokol* je množica pravil, ki jih morata upoštevati procesa (aktivnosti) znotraj nekega sloja, pa tudi določila o obliki in pomenu okvirjev, paketov ali sporočil. Aktivnosti znotraj sloja uporabljajo protokol za realizacijo storitev. S stališča uporabnika storitev je povsem nepomembno kakšen protokol je izbran za realizacijo storitev. Uporabniku storitev mora biti to popolnoma prikrito. Protokol se lahko spremeni (zamenja), ne da bi se spremenile storitve. Na primer *prenos datotek* je storitev aplikacijskega nivoja. Za realizacijo prenosa datotek je potreben *protokol za prenos datotek*. Prenos datotek je lahko realiziran na ta ali oni način,

s takim ali drugačnim protokolom. Za uporabnika storitev (prenosa datotek) je važno samo to, da je datoteka verno prenešena na drugo stran in prav nič, *kako* so realizirane storitve (kakšen je protokol za prenos datotek).

2.3.1 Osnovne operacije s storitvami

Storitev je definirana z množico osnovnih operacij (Ang. Primitives), ki so na voljo uporabniku storitve (ali kakšni drugi aktivnosti). S temi osnovnimi operacijami uporabnik storitve določi, kaj naj dajalec storitve naredi zanj. V ISO OSI referenčnem modelu obstajajo štiri vrste osnovnih operacij: zahteva (Request), naznanilo (Indication), odziv (Response) in potrdilo (Confirm):

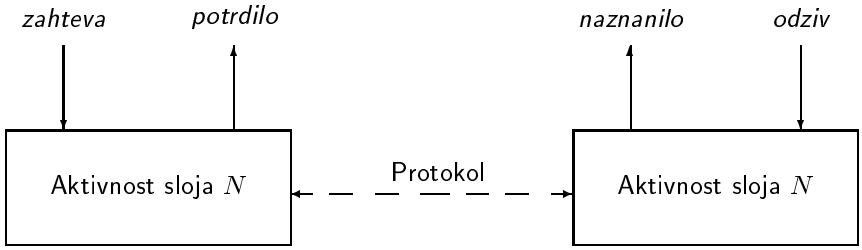
Zahteva	aktivnost zahteva od dajalca storitve neko dejanje
Naznanilo	aktivnost je obveščena o nekem dejanju
Odziv	aktivnost se odzvove na neko dejanje
Potrdilo	aktivnost je obveščena o realizaciji svoje zahteve

Zahteva je operacija, s katero neka aktivnost zahteva (kot pove ime) od dajalca storitve neko dejanje, na primer vzpostavitev zveze. Ko je zveza vzpostavljena, je istorodna aktivnost o tem obveščena z *naznanilom*. Aktivnost, ki dobi naznanilo se odzove, uporabi operacijo *odziv* ter tako pove, da v našem primeru pristaja na vzpostavitev zveze. Končno, aktivnost, ki je opravila zahtevo dobi obvestilo o vzpostavitvi zveze z operacijo *potrdilo*. Vse štiri operacije so prikazane na sliki 20.

Storitve so lahko *potrjene* ali *nepotrjene*. Potrjena storitev omogoča vse naštete štiri osnovne operacije. Za nepotrjeno storitev obstajata samo *zahteva* in *naznanilo*. Sedaj pa poglejmo primer, ki nam je najbližji. Denimo, da želi oseba A po telefonu govoriti z osebo B. Možno zaporedje operacij je:

Storitev	Vzpostavitev zveze
Zahteva	Oseba A pozove osebo B (Zahteva vzpostavitev zveze),
Naznanilo	Oseba B sliži znak poziva (Naznanilo zahteve za vzpostavitev zveze),
Odziv	Oseba B dvigne slušalko (Pristanek na zahtevo za vzpostavitev zveze),
Potrdilo	Oseba A zazna prestanek pozivanja (Zveza je vzpostavljena),
	Sledi pogovor
Storitev	Sprostitev zveze
Zahteva	Oseba A odloži slušalko (Zahteva sprostitev zveze)
Naznanilo	Oseba B sliši zahtevo za prekinitev zveze in odloži slušalko.

V tem primeru je vzpostavljanje zveze potrjena storitev, sprostitev zveze pa nepotrjena storitev.



Slika 20: Osnovne operacije za storitev *vzpostavi zvezo*.

Toliko o storitvah in operacijah s storitvami.

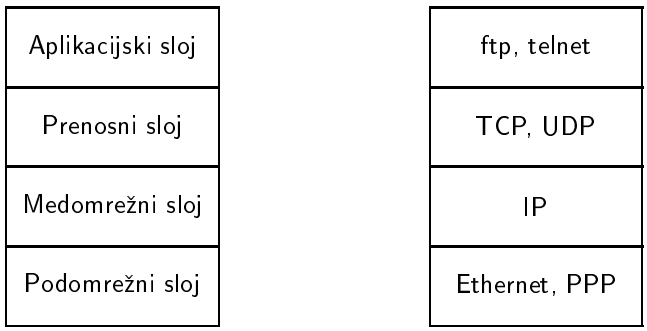
2.4 Primeri mrežnih arhitektur

Čeprav je arhitekturni model OSI sedem slojni pa to ne pomeni, da arhitekture z drugačnim številom slojev niso možne ali ne bi smele obstajati. Drugačne mrežne arhitekture so obstajale dosti prej, predno je nastal model OSI, na primer IBM-ova mrežna arhitektura SNA (System Network Architecture), Digital-ova arhitektura DNA, in druge. Najbolj prepričljiv dokaz za smisel ne-OSI arhitektur je globalno omrežje Internet, ki se dnevno potrjuje z vse večjim številom vozlišč. Ne glede na to je model OSI gotovo prispeval k bolj sistematičnemu in bolj enotnemu obravnavanju zgradbe in delovanja omrežij. Denimo, marsikatero omrežje, ki je nastalo pred samim modelom in se obravnavalo kot omrežje "v enem kosu", se danes obravnava razslojeno z vidika modela OSI.

2.4.1 Arhitektura omrežja Internet

Z imenom Internet (z veliko začetnico) označujemo globalno svetovno omrežje, ki povezuje omrežja tipa TCP/IP. Ko govorimo o omrežjih TCP/IP, mislimo na omrežja, ki uporabljajo protokolovni sklad TCP/IP (Transmission Control Protocol/Internet Protocol) in pa tudi druge protokole, ki se bistveni za delovanje teh omrežij (npr. SNMP, UDP, i.t.d.). Omrežje Internet temelji na štirislojnem arhitekturnem modelu, ki ima svoje korenine v omrežju ARPANET iz konca šestedestih let. Internet ni grajen po modelu OSI. To nikakor ne pomeni, da bi Internet ne bilo "odprto" omrežje ali da ne bi imelo možnosti realizacije vseh potrebnih omrežnih funkcij. Internet je omrežje, ki "deluje". To potrjuje dejstvo, da je TCP/IP omrežij danes daleč več kot vseh drugih omrežij skupaj. Slika 21 prikazuje arhitekturni model in primer arhitekture. Spodnji trije sloji so mrežnega značaja, zgornji sloj je uporabniško usmerjen. V arhitekturi na levi strani slike so zajeti le nakateri najbolj tipični protokoli izmed sicer možnih protokolov. V

lokalnih omrežjih Internet se v spodnjem sloju (ang. Network Interface Layer) največ uporablja Ethernet protokol. Za povezovanje oddaljenih vozlišč točka-točka po podemskih linijah se uporabljata protokola SLIP (Serial Link IP) in PPP (Point-to-Point-Protocol). Ker spodnji sloj realizira funkcije spodnjih dveh slojev modela OSI, pravimo tudi, da je arhitektura petslojna. Če primerjamo Internet model z modelom OSI, bi medomrežni sloj (ang. internet layer) najbolj ustrezal mrežnemu sloju, prenosni sloj prenosnemu, aplikacijski sloj pa zgornjim trem slojem modela OSI.



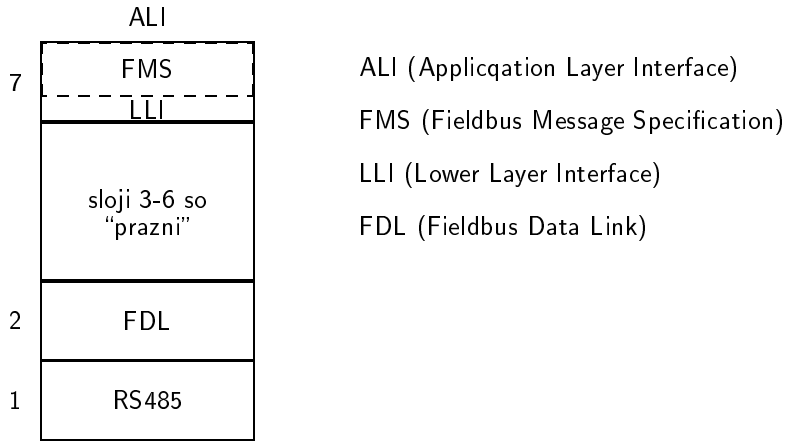
Slika 21: Model in arhitektura omrežja Internet.

2.4.2 Arhitektura Profibus

Procesno področno vodilo Profibus (ang. Process Field Bus) so razvili nemški proizvajalci industrijske komunikacijske opreme v sodelovanju s še petimi nemškimi inštituti. Namenjen je za povezovanje naprav v industrijskih okoljih (senzorjev in aktuatorjev, programljivih krmilnikov - PLC, numerično krmiljenih strojev - NC, CNC, industrijskih računalnikov - PC, i.t.d.). V arhitekturni zasnovi so razvijalci omrežja Profibus upoštevali model OSI, očitno pa so se zgledovali po arhitekturi MiniMAP. Da bi zadostili ostrim zahtevam po kratkih odzivnih časih, a hkrati ohranili možnost interoperabilnosti (zmožnosti skupnega delovanja naprav različnih proizvajalcev), so definirali samo tri sloje: fizični in podatkovni sloj ter sloj aplikacije. Funkcionalnost ostalih slojev so vgradili v podsloj aplikacije LLI (Lower Layer Interface). Obstajajo tri različice omrežja Profibus: Profibus-FMS, Profibus-PA in Profibus-DP. Slika 22 prikazuje arhitekturo omrežja Profibus-FMS. O procesnih področnih vodilih bomo še govorili.

Literatura

[1] J. Day, H. Zimmermann, "The OSI Reference Model", *Proc. of IEEE*, 71, pp.1334-1340, Dec 1983.



Slika 22: Arhitektura omrežja Profibus-FMS.

[2] V. C. Jones, *MAP/TOP Networking*, McGraw-Hill 1988.

[3] J. Morris, *Introduction to Communication, Command, and Control Systems*, Pergamon Press, 1978.

[4] D. M. Piscatello, A. L. Chapin, *Open Systems Networking, TCP/IP and OSI*, Addison–Wesley, 1993.

[5] E. Parr, *Programable Controllers*, B.H. 1993.

[6] T. Ramtheke, *Networks*, Prentice–Hall, 1994.

[7] W. R. Stevens, *TCP/IP Illustrated, Vol. 1*, Addison–Wesley, 1994.

[8] A. Tanenbaum, *Computer Networks*, 3rd Edition, Prentice-Hall 1996.

[9] J. Virant, *Teleinformatika in ISDN*, Univerza v Ljubljani, Fakulteta za elektrotehniko in računalništvo, Ljubljana 1989.

[10] Michael Volz, *PROFIBUS*, PROFIBUS Nutzerorganisation, Technische Druckschrift, 1994.

[11] *Digital Industrial Networks Guidebook*, Digital, 1988.

[12] ISO/IEC 7498-1:1994 Information technology – Open Systems Interconnection – Basic Reference Model: The Basic Model, 1994. (Ekvivalenten priporočilu ITU-T X.200)

[13] *Mikroračunalniški telekomandni sistem TI30/11*, ISKRA Avtomatika, Ljubljana 1980.

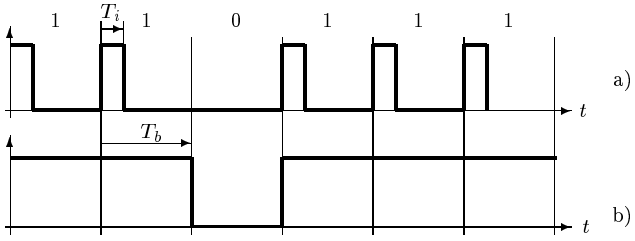
[14] *Open Systems Interconnection, Technology Overview*, Novell 1991.

3 Elementi fizičnega sloja

Fizični sloj skrbi za prenos informacijskih signalov po komunikacijskem kanalu. Osnovna informacijska enota tega sloja je bit. Tipična vprašanja v zvezi s tem slojem so: kakšna je hitrost prenosa, kako se sprejemnik sinhronizira z oddajnikom, kakšne so oblike informacijskih signalov, kakšni so postopki modulacije, kakšen prenosni medij je potreben, in podobno. V tem poglavju si bomo skušali odgovoriti na nekatera od teh vprašanj.

3.1 Oblike digitalnih signalov

Informacijo, ki jo zapišemo z binarnimi simboli, moremo prenašati v osnovnem frekvenčnem pasu brez modulacije (ang. Base Band Transmission) z impulznimi signali različnih oblik. Eno od možnosti za predstavitev zaporedja osmih simbolov smo skicirali na sliki 23.a. Simbol ena smo predstavili s prisotnostjo impulza širine T_i in simbol nič z odsotnostjo impulza. Tak signal prenašamo od oddajnika do sprejemnika, ta pa mora iz oblike sprejetega signala razbrati njegov pomen. Verjetnost za pravilno detekcijo prisotnosti impulza narašča z energijo impulza. Zato si s tega stališča želimo, da bi bila širina impulza čimvečja. Če povečamo trajanje impulza do največje možne mere ($T_i = T_b$), dobimo v našem primeru signal 23.b.



Slika 23: Dve možni obliki informacijskega signala.

Signal ostane v nespremenjenem stanju cel čas oddajanja bita. Tak signal se pri dolgih zaporedjih simbolov enake vrednosti malo spreminja in za prenos zahteva manjšo frekvenčo širino kanala. Po drugi strani pa zaradi odsotnosti sprememb v signalu sprejemna naprava težje ugotovi začetek in konec posameznega simbola oziroma težje ugotovi tisti trenutek, ko naj odčita vrednost signala. Sinhronizacija sprejemnika z oddajnikom je težja.

V narisanem primeru smo za simbol nič izbrali ničelno vrednost signala (odstotnost impulza). Zato nam ničelna vrednost signala pomeni dvoje, bodisi binarni simbol nič bodisi odsotnost informacijskega signala. Narisani signal

vsebuje še enosmerno komponento, ki se s časom spreminja v odvisnosti od informacijske vsebine signala. To je v praksi nezaželeno. Lahko pa bi za simbol nič izbrali nivo signala različen od nič, ali pa bi se odločili za signal, ki od ničelne vrednosti niha v pozitivni in v negativni smeri. Možnosti za prenos informacije z valom pravokotnih impulzov je torej veliko. Vsaka ima svoje prednosti in slabosti. Kakšno obliko signala naj potem v danih okoliščinah izberemo? Bistvena vprašanja pri tem so: kakšne so možnosti za sinhronizacijo sprejemnika z oddajnikom iz sprejetega signala (ugotavljanje takta), kakšna je potrebna frekvenčna širina kanala pri zahtevani hitrosti prenosa in kakšna je odpornost na motnje oziroma zmožnost sprejemnika za razpoznavanje prave vrednosti signala. K temu dodajmo še vprašanje, ali je v signalu prisotna enosmerna komponenta in nenazadnje kakšna je zahtevnost realizacije naprav. Oblike, ki bi hkrati zadostila vsem kriterijem, ni. V splošnem velja, da je sinhronizacija sprejemnika z oddajnikom lažja, če je v signalu več sprememb. Čimveč pa je sprememb v signalu, višje frekvence vsebuje in večja je zahtevana frekvenčna širina kanala za doseganje iste hitrosti prenosa. Čimveč stanj zmore zavzeti signal, več bitov informacije lahko nosi in višja bo hitrost prenosa, zato pa je težje razpoznati pravo vrednost signala.

Signale za prenos binarne informacije je možno razvrstiti po več kriterijih, na primer:

- glede na to, ali signal vstraja v nespremenjenem stanju cel čas trajanja binarnega simbola ali je ta čas krajši,
- glede na usmerjenost impulzov,
- glede na to, ali prenašamo tako 'binarno ena' kot 'binarno nič',
- glede na število različnih možnih stanj signala.

Po prvem kriteriju delimo informacijske signale na signale brez povratka na nič, kjer se nivo signala v času trajanja simbola ne spremeni (ang. **N**on-**R**eturn-to-**Z**ero ali s kratico NRZ) in na signale s povratkom na nič, pri katerih se nivo signala po oddaji vsakega binarnega simbola povrne na nevtralen-ničti nivo, (ang. **R**eturn-to-**Z**ero ali s kratico RZ). Po drugem kriteriju delimo informacijske signale na enopolarne, če so vsi impulzi enako usmerjeni in na dvoploarne, če so impulzi usmerjeni v obe smeri od mirovne vrednosti, to je v pozitivno in v negativno smer od ničelne vrednosti. Glede na tretji kriterij, je signal lahko popolnoma (ali čisto) binaren, če prenašamo tako 'binarno ena' kot 'binarno nič', ali polovično binaren, če prenašamo le 'binarno ena', 'nič' pa pomeni odsotnost impulza ali odsotnost spremembe signala. Po zadnjem kriteriju delimo signale na N-nivojske, kjer je N število nivojev (stanj) signala. V tem primeru pomeni vsak od prenašanih nivojev ustrezno zaporedje binarnih simbolov. Na primer, če je signal štirinivojski,

lahko pomeni vsak od štirih možnih nivojev signala enega izmed parov binarnih simbolov (dibitov): 00, 01, 10 ali 11.

V tem razdelku bomo obravnavali nekaj važnejših skupin oblik, ki so se uveljavile za prenos informacije v osnovnem frekvenčnem pasu ter jih primerjali med seboj. Te skupine so:

- signali brez povratka na nič (NRZ),
- signali s povratkom na nič (RZ),
- bifazni signali (bi-Φ) in
- večnivojski.

Slike 24, 25, 27 in 28 prikazujejo različne oblike signala za zaporedje binarnih simbolov 1 0 1 1 0 1 0 0 1 0.

3.1.1 Signali brez povratka na nič - NRZ

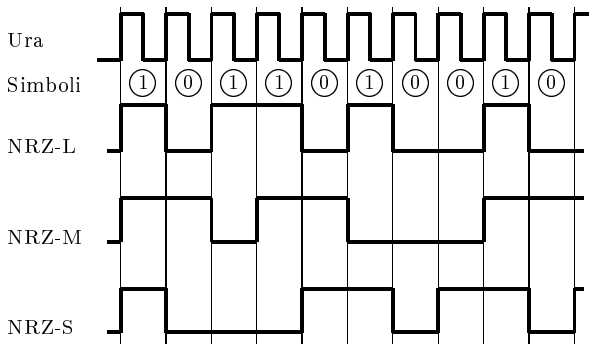
Oblika NRZ se največ uporablja. Obstajajo tri različice signala brez povratka na nič: NRZ-L (Level), NRZ-M (Mark) in NRZ-S (Space), v enopolarni ali dvopolarni izvedbi, glej sliko 24. Nivojska ali NRZ-L oblika je najobičajnejša oblika. Signal NRZ-M se spremeni vedno in samo v primeru, ko oddajamo simbola ena. NRZ-M obliko imenujemo tudi NRZI (NRZ-Invert-On-One). Podobno se pri signalu NRZ-S stanje menja samo pri oddaji simbola nič.

V primeru daljšega zaporedja enakih simbolov (ničel ali enic) se NRZ-L signal malo spreminja in sprejemnik se težko sinhronizira z oddajnikom. Signal NRZ se zato koristi pri asinhronem serijskem načinu prenosa. Po drugi strani pa zahtevajo signali NRZ pri isti hitrosti prenosa enkrat manjšo frekvenčno širino kanala kot signali RZ. Enosmerni komponenti, ki se poleg tega spreminja v odvisnosti od informacijske vsebine signala se popolnoma ne moremo izogniti.

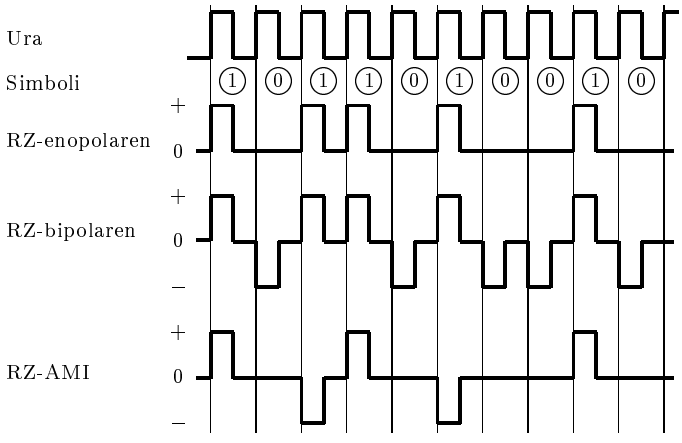
3.1.2 Signali s povratkom na nič - RZ

Slika 25 prikazuje nekaj pogostejših oblik signalov s povratkom na nič: enopolaren in dvopolaren RZ ter RZ-AMI (RZ-Altarnate-Mark-Inversion). RZ oblika omogoča dobro sinhronizacijo sprejemnika z oddajnikom, potrebuje pa pri isti hitrosti prenosa kot NRZ v splošnem dvojno frekvenčno širino kanala.

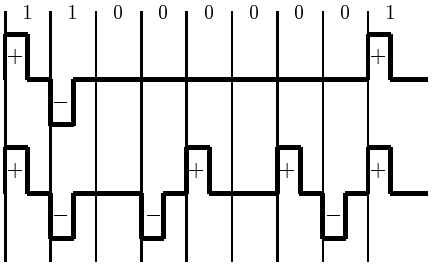
Za prenos podatkov je najpomembnejša oblika RZ-AMI. S tem signalom se enice prenašajo izmenoma s pozitivnim in negativnim impulzom, ničlo pa pomeni



Slika 24: Signali brez povratka na nič (NRZ).



Slika 25: Tipične oblike signalov s povratkom na nič (RZ).



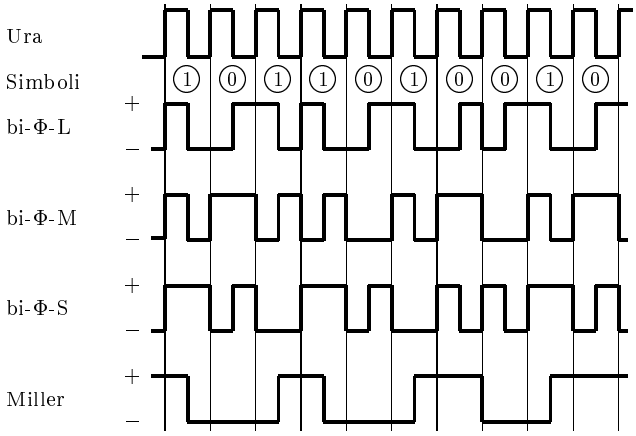
Slika 26: Nadomeščanje zaporednih ničel. Zgoraj: brez nadomeščanja, spodaj: B6ZS nadomeščanje.

odsotnost impulza. Enosmerna komponenta signala je praktično nič, potrebna frekvenčna širina kanala pa taka kot pri signalu NRZ. V primeru daljšega zaporedja ničel se signal sploh ne spreminja in lahko pride do izpadanja sinhronizacije. Da se to ne zgodi, se vsako neprekinjeno zaporedje šestih ničel nadomesti s signalom bolj živahne oblike. Postopek je znan pod kratico B6ZS (ang. Binary-Six-Zero-Substitute). V primeru, da je zadnja enica imela negativen predznak, oddajnik šest ničel nadomesti z vzorcem 0 - + 0 + -, glej sliko 26. V primeru, da je zadnja enica imela pozitiven predznak, oddajnik šest zaporednih ničel nadomesti z vzorcem 0 + - 0 - + . Ker si v sprejetem signalu sledita drug za drugim impulza z enako polariteto sprejemnik ve, da drugi impulz ne pomeni enice, ampak da gre za nadomeščanje zaporedja ničel.

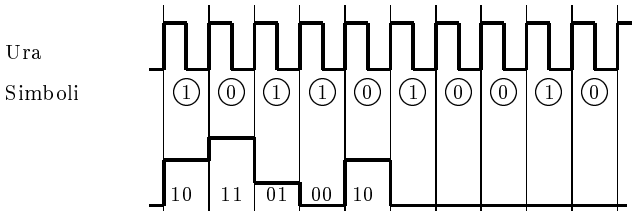
3.1.3 Bifazni signali

Bifazni signali so vedno dvopolarni signali. Poznamo tri različice teh signalov: bi-Φ-L, bi-Φ-M in bi-Φ-S, glej sliko 27. Daleč največ se uporablja oblika bi-Φ-L in je bolj znana pod imenom 'Manchester'. Binarni simbol je kodiran s spremembo signala v sredini bitne celice. Sprememba navzdol pomeni enico, sprememba navzgor pa ničlo. Signal ne vsebuje enosmerne komponente, dober je v pogledu sinhronizacije. Potrebuje sicer dvojno frekvenčno širino kanala, vendar ker se koristi v lokalnih omrežij za prenos v osnovnem frekvenčnem pasu, to ni tako pomembno, ker frekvenčna širina ni tako dragocena kot recimo v javnih omrežijih. Pri signalu bi-Φ-M so razmere podobne: signal se vedno spremeni na začetku bitne celice, v primeru enice pa tudi v sredini. Pri signalu bi-Φ-S je obratno: signal se vedno spremeni na začetku, v sredini pa le pri oddaji ničle.

Pomembna je še takoimenovana Millerjeva oblika signala. Pri oddaji enice se signal spremeni v sredini. Pri oddaji ničle se signal ne spremeni, razen če ne sledi ponovno ničla. Tedaj se signal spremeni na koncu prve in pred začetkom naslednje ničle.



Slika 27: Oblike bifaznih signalov.

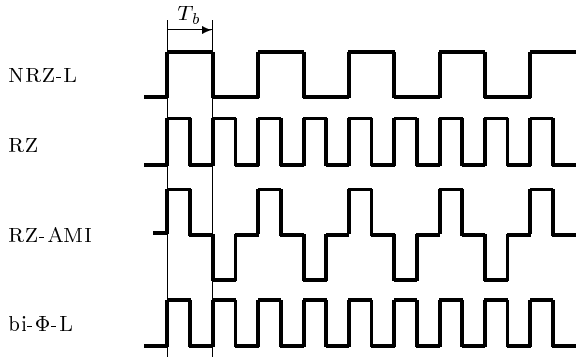


Slika 28: Primer večnivojskega (štirinivojskega) signala.

3.1.4 Večnivojski signali

Veliko signalov uporablja namesto dveh nivojev (stanj) za prenos informacije več nivojev. Tudi bipolarni RZ in RZ-AMI uporabljata več (tri) nivojev, vendar nosita informacijo samo dva. S tremi stanji še vedno kodiramo samo en bit informacije (dva simbola).

Smisel večnivojskih signalov je v tem, da se da v primeru večjega števila dovoljenih stanj prenašati z vsakim signalnim elementom več binarnih simbolov in s tem večjo množino informacije pri sicer enaki frekvenčni širini kanala. Tak način kodiranja je torej potreben kadar je frekvenčna širina zelo dragocena (kot je to v primeru javnega telefonskega omrežja). Slika 28 prikazuje primer večnivojskega signala.

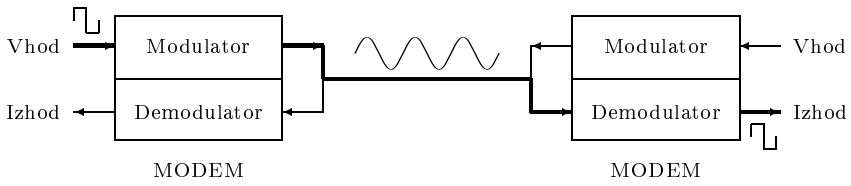


Slika 29: Ponazoritev potrebne frekvenčne širine kanala pri enaki hitrosti prenosa za signale NRZ-L, RZ, RZ-AMI in bi-Φ-L.

3.1.5 Signali NRZ, RZ, RZ-AMI, bi-Φ in frekvenčna širina kanala

Poglejmo kakšna je potrebna frekvenčna širina kanala za signale NRZ, RZ, RZ-AMI in bi-Φ za doseganje enake hitrosti prenosa. Teoretično gledano bi za popolnoma veren prenos pravokotnih impulzov v vseh primerih potrebovali neskončen frekvenčni pas, kajti vlak pravokotnih impulzov vsebuje poleg prve harmonske komponente f_1 še neskončno lihih višjeharmonskih komponent. Ker za razpoznavanje dveh logičnih vrednosti popolna rekonstrukcija signala ni potrebna, zadostuje v praksi že prenos nekaj nižjih harmonskih komponent. Poglejmo sedaj, kakšna je potrebna frekvenčna širina kanala za prenos enakega števila harmonskih komponent. V ta namen opazujmo signale NRZ, RZ, RZ-AMI in bi-Φ za tako zaporedje binarnih simbolov, pri katerem se signal najhitreje spreminja, glej 29.

Signal NRZ se najhitreje spreminja tedaj ko si izmenoma sledijo ničle in enice. Frekvenca prve harmonske komponente je $f_{NRZ} = 1/(2 \times T_b)$. Signal RZ se najhitreje spreminja ko si neposredno sledijo enice, frekvenca prve harmonske komponente pa je še enkrat višja, $f_{RZ} = 2 \times f_{NRZ} = 1/T_b$. Signal RZ-AMI se najhitreje spreminja ko si zaporedoma sledijo enice, frekvenca prve harmonske je $f_{RZ-AMI} = f_{NRZ} = 1/(2 \times T_b)$. Signal bi-Φ-L se najhitreje spreminja ko si sledijo simboli z enako vrednostjo. Frekvenca prve harmonske je $f_{bi-\Phi-L} = f_{RZ} = 1/T_b$. Za prenos istega števila harmonskih komponent (oziroma za doseganje enake hitrosti prenosa) je torej pri signalih RZ in bi-Φ-L v primerjavi s signali NRZ in RZ-AMI potrebna dvojna frekvenčna širina kanala.



Slika 30: Modem in prenos podatkov po telefonskem vodu.

3.2 Modemi in modulacije

Modem je komunikacijska naprava, ki omogoča prenos digitalnih signalov po analognih prenosnih poteh, kot na primer analognem telefonskem omrežju. Frekvenčni pas naročniškega voda je namenoma omejen na frekvence med 300 in 3400 Hz. To v pogledu razumljivosti govora popolnoma zadostuje.[14] Zaradi omejene frekvenčne širine telefonskega voda pa po njem ne moremo prenašati pravokotnega informacijskega signala neposredno. Popačenju signala se izognemo tako, da ga frekvenčno premaknemo v območje frekvenc telefonskega voda. To dosežemo z modulacijo. Napravo, ki podatke na oddajni strani modulira, na sprejemni strani pa povrne v prvotno obliko (demodulira), imenujemo modem, glej sliko 30. Razen modulacije in demodulacije pa sodobni modemi opravljajo številne druge funkcije, ki so potrebne pri prenosu podatkov.

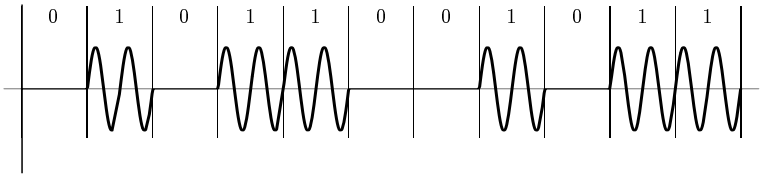
3.2.1 Modulacija

Modulacija je izraz za spreminjanje analognega, običajno sinusnega signala v odvisnosti od informacijskega signala. Sinusni signal je v pomoč pri prenosu informacijskega signala in mu zato pravimo *nosilni signal* ali *nosilec*. Z modulacijo dobimo moduliran signal. Zapišimo cosinusni nosilni signal z amplitudo U_c, kotno frekvenco ω_c in faznim zasukom ϕ_c:

$$u_c(t) = U_c \cos(\omega_c t + \phi_c). \tag{15}$$

V odvisnosti od informacijskega signala $u_i(t)$ se nosilcu lahko spreminja amplituda, frekvenca ali faza. Če se v odvisnosti od informacijskega signala spreminja amplituda nosilca, $U_c(t) = U_c(u_i(t))$, pravimo, da je signal amplitudno moduliran. V primeru, da informacijski signal direktno vpliva na frekvenco nosilca, $\omega_c(t) = \omega_c(u_i(t))$ govorimo o frekvenčni modulaciji. S spreminjanjem faze nosilnega signala v odvisnosti od informacijskega signala, $\phi_c(t) = \phi_c(u_i(t))$, dobimo fazno moduliran signal. Frekvenčni ali fazni modulaciji rečemo tudi kotna modulacija.

[14] Govorni signal sicer obsega frekvence med 80 in 6000 Hz. Zdravo človeško uho zaznava frekvence med 20 Hz in 20000 Hz.



Slika 31: Amplitudna modulacija z binarnim signalom.

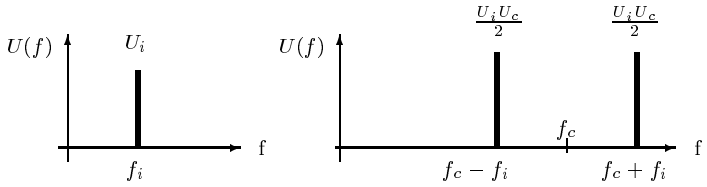
V naslednjih razdelkih bomo govorili o osnovah modulacije analognega signala z digitalnim informacijskih signalom oziroma o tehniki prenosa digitalnih signalov z analognim nosilcem.

3.2.2 Amplitudna modulacija z binarnim signalom

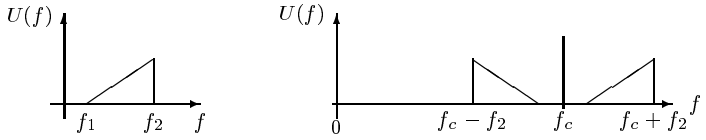
Pri amplitudni modulaciji se v odvisnosti od informacijskega signala spreminja velikost nosilnega signala. Če je informacijski signal binaren, ima amplitudno modulirani nosilec samo dve različni amplitudi, recimo 0 in U voltov. Amplitudni modulaciji z binarnim signalom pravimo *modulacija z amplitudnim odmikom* ali **ASK** (ang. **A**mplitude **S**hift **K**eying). V tem primeru moduliramo tako, da v odvisnosti od informacijskega signala preprosto vklapljamo in izklapljamo nosilni signal. Na sliki 31 je narisan primer amplitudno moduliranega signala, pri čemer pomeni odsotnost nosilca binarno vrednost 'nič', prisotnost nosilca pa vrednost 'ena'. V primeru, da informacijski signal vsebuje daljša zaporedja simbolov 'nič', vsebuje modulirani signal daljše časovne premore brez spremembe in sprejemnik se težko sinhronizira z oddajnikom. Zato raje prenašamo tudi simbol nič z od nič različno amplitudo.

Amplitudna modulacija se uporablja za počasne prenose. Hitrost prenosa lahko povečamo, če povečamo število različnih možnih stanj signala. Na primer, s štirinivojskim signalom prenašamo podatke dvakrat hitreje, ker vsako od štirih možnih stanj (nivojev) nosi dva bita informacije. Večje število različnih stanj signala sicer omogoča višje hitrosti prenosa, vendar pa se po drugi strani veča vpliv šuma.

Napačno bi bilo sklepati, da amplitudno modulirani signal vsebuje samo frekvenco nosilnega signala, na kar morda navaja slika 31. Pa si poglejmo, kako je s frekvenčno vsebino moduliranega signala. Frekvenčni premik signala $u_i(t)$ iz osnovnega frekvenčnega področja v neko drugo, višje frekvenčno področje, dosežemo z množenjem signala $u_i(t)$ z ustreznim cosinusnim signalom $u_c(t)$. Naj bo signal $u_i(t)$, ki ga želimo frekvenčno premakniti, zaradi lažjega razumevanja tudi cosinusen, $u_i(t) = U_i \cos \omega_i t$. Množimo signal $u_i(t)$ z $u_c(t)$. Dobimo moduliran signal



Slika 32: Amplitudni spekter prvotnega signala (levo) in frekvenčno premaknjenega signala (desno).



Slika 33: Frekvenčni spekter osnovnega signala (levo) in amplitudno moduliranega signala (desno).

$u_m(t)$,

$$u_m(t) = u_i(t)\ u_c(t) = U_i \cos\omega_i t\ U_c \cos\omega_c t = \frac{U_iU_c}{2}\cos(\omega_c+\omega_i)t + \frac{U_iU_c}{2}\cos(\omega_c-\omega_i)t,$$

ki ga sestavljata dva cosinusna vala z enako amplitudo in s frekvenco, ki je enaka vsoti in razliki frekvenc f_c in f_i. Amplitudni spekter prvotnega in frekvenčno premaknjenega signala prikazuje slika 32. Za signal $u_i(t)$ z bogatejšo frekvenčno vsebino bi bile razmere podobne.

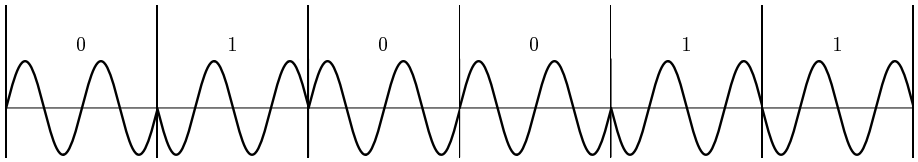
Frekvenčno premaknjen signal $u_m(t)$, iz katerega se da relativno enostavno rekonstruirati prvotni signal $u_i(t)$, dobimo s prištevanjem nosilnega vala $u_c(t)$ k produktu obeh,

$$u_m(t) = U_c[1 + \frac{u_i(t)}{U_c}]\cos\omega_c t = U_m(t)\cos\omega_c t.$$

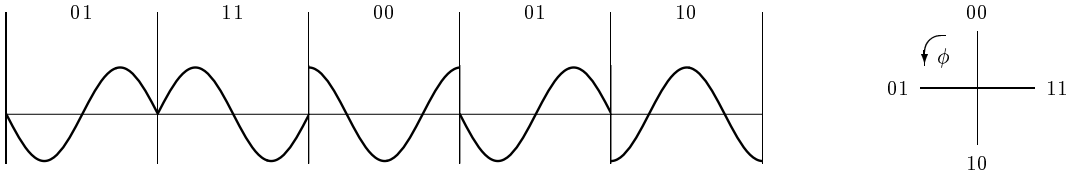
Nosilnemu signalu se v odvisnosti od informacijskega signala spreminja amplituda $U_m(t)$, tako da je ovojnica moduliranega signala kar enaka informacijskemu signalu. Frekvenčna vsebina amplitudno moduliranega signala je podobna frekvenčni vsebini frekvenčno premaknjenega signala, le da je v amplitudno moduliranem signalu prisoten nosilni val, glej sliko 33.

3.2.3 Fazna modulacija (PM)

Pri fazni modulaciji se se v odvisnosti od amplitude informacijskega signala spreminja fazni kot nosilca. Fazna modulacija z binarnim informacijskim signalom se imenuje modulacija s faznim premikom (ang. **P**hase **S**hift **K**eying ali s kratico **PSK**). Naj binarni modulacijski signal $u_i(t)$ zavzema vrednosti nič voltov



Slika 34: Modulacija s faznim premikom (PSK).



Slika 35: Fazna modulacija s štirimi faznimi skoki.

za binarno nič in $+U$ voltov za binarno ena. Fazno moduliran signal

$$u_m(t) = U_c \sin[\omega_c t + \phi_c(t)]$$

ima konstantno amplitudo U_c in frekvenco ω_c, fazni kot $\phi_c(t)$ pa je ali nič ali π:

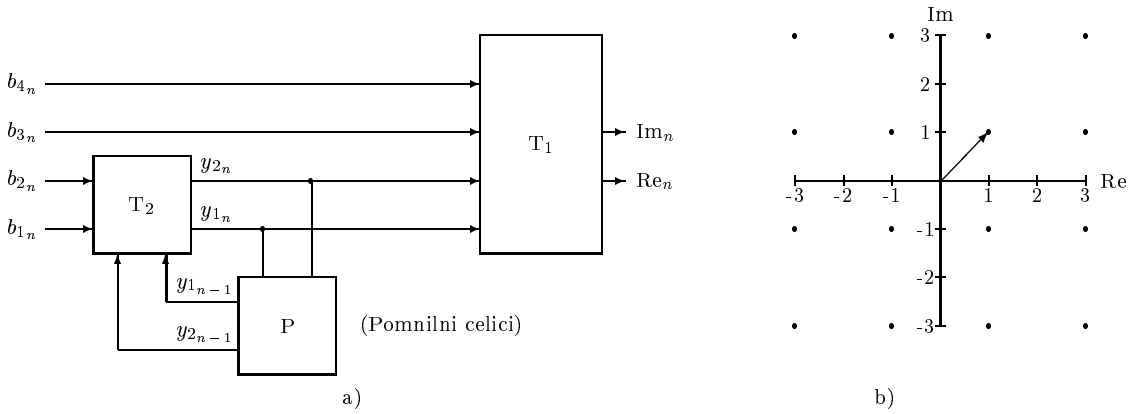
$$u_m(t) = \begin{cases} +U_c \sin \omega_c t, & \text{za } u_i(t) = 0, \\ -U_c \sin \omega_o t, & \text{za } u_i(t) = U. \end{cases}$$

Slika 34 prikazuje primer fazno moduliranega signala. Pri vsaki spremembi stanja infomacijskega signala (iz nič v ena ali obratno) se spremeni tudi faza nosilnega signala za 180°.

Za doseganje višjih hitrosti prenosa se uporabljajo fazne modulacije z večjim številom faznih skokov. Fazna modulacija s štirimi faznimi skoki (4-PSK) omogoča dvakrat višjo hitrost prenosa pri isti hitrosti spreminjanja signala, ker v tem primeru vsak od štirih signalnih elementov nosi informacijo dveh bitov. Na sliki 35 je narisan primer tako moduliranega signala in pravilo kodiranja. Na primer, moduliran signal s parom bitov 00 je fazno premaknjen za 90° glede na referenčni signal.

Za doseganje višje hitrosti prenosa uporabljajo modemi osem (8-PSK), šestnajst (16-PSK) in več faznih skokov. Modulacijo s štirimi ali več faznimi skoki imenujemo tudi kvadraturna modulacija.

Poleg navadne fazne modulacije je v uporabi diferenčna fazna modulacija (DPSK). Za primer na sliki 35 so binarni simboli kodirani z absolutnimi faznimi



Slika 36: Kvadraturna amplitudna modulacija po priporočilu CCITT V.22bis. Blokovna shema vezja za preslikavo četverke bitov v realno in imaginarno komponento signalnega elementa (a) in fazni diagram (b). Točke pomenijo dovoljene amplitude in faze signalnih elementov.

premiki glede na referenčni signal. Zato je referenčni signal potreben tudi za dekodiranje (demoduliranje). Pri diferenčni PSK pa so binarni simbili kodirani relativno glede na fazo predhodnega signalnega elementa. Referenčni signal za dekodiranje ni potreben.

3.2.4 Kvadraturna amplitudna modulacija (QAM)

Kvadraturna amplitudna modulacija (**Q**uadrature **A**mplitude **M**odulation - QAM) je ime za kombinacijo fazne in amplitudne modulacije. Število možnih stanj signala označimo s številom pred kratico QAM. Oznaka 16-QAM tako pomeni, da je modulirani signal sposoben zavzeti eno izmed 16 stanj. Teh 16 stanj se da doseči na več načinov. Na primer, z osmimi fazami nosilnega signala enake amplitude (osem stanj) in štirimi fazami pri dveh amplitudah, ki sta različni od amplitude prvih osmih faz. (spet osem možnosti). V seštevku imamo torej dvanajst faz, pri štirih od teh sta možni dve različni amplitudi. Ker vsako stanje signala nosi štiri bite informacije, se pri dani modulacijski hitrosti doseže štirikrat višja hitrost prenosa v bitih na sekundo.

Slika 36 prikazuje fazni diagram signalnih elementov in načelno blokovno shemo vezja za preslikavo četverk bitov v signalni element in sicer za modeme z modulacijo 16-QAM po priporočilih CCITT V.22bis in V.32. V modulator vstopajo hkrati skupine štirih bitov, glej sliko 36.b.

Prva dva bita določata (kodirata) kvadrant signalnega elementa, druga dva bita določata signalni element znotraj kvadranta. Kvadrant signalnega elementa je kodiran diferenčno glede na fazo (kvadrant) prejšnjega signalnega elementa (preslikava T_2) po pravilnostni tabeli 1. Končno preslikavo T_1 četverke bitov v

Tabela 1: Diferenčno kodiranje kvadranta signalnega elementa v trenutku n po priporočilu V.22bis (Preslikava T_2).

b_{1_n}	b_{2_n}	$y_{1_{n-1}}$	$y_{2_{n-1}}$	Faza v °	y_{1_n}	y_{2_n}
0	0	0	0	90	0	1
0	0	0	1	90	1	1
0	0	1	0	90	0	0
0	0	1	1	90	1	0
0	1	0	0	0	0	0
0	1	0	1	0	0	1
0	1	1	0	0	1	0
0	1	1	1	0	1	1
1	0	0	0	180	1	1
1	0	0	1	180	1	0
1	0	1	0	180	0	1
1	0	1	1	180	0	0
1	1	0	0	270	1	0
1	1	0	1	270	0	0
1	1	1	0	270	1	1
1	1	1	1	270	0	1

signalni element prikazuje tabela 2.

Modemi po priporočilu V.22bis omogočajo dvosmerno (dupleksno) delovanje s frekvenčno ločenima kanaloma za prenos v eni in v drugi smeri (frekvenčno multipleksiranje kanalov). Nosilni frekvenci znašata 1200 Hz za prenos v eni smeri in 2400 Hz za prenos v drugi smeri. Hitrost prenosa je 2400 b/s pri 600 Bd. Modemi po priporočilu V.32 omogočajo dvosmeren dvožičen prenos s hitrostjo do 9600 b/s pri 2400 Bd s frekvenco nosilca 1800 Hz. Dvosmeren prenos se dosega z dušenjem odmeva (ang. Echo Cancellation).

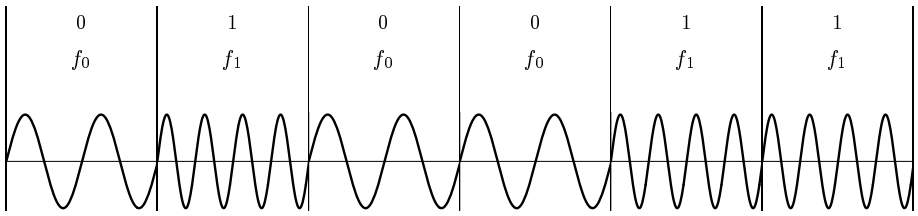
Uporabljajo se tudi druge kombinacije amplitud in faz. Na primer modemi po priporočilu CCITT V.29 se uporabljajo za dupleksni štirižični prenos po najetih telefonskih linijah. Uporabljajo osem različnih faz pri dveh različnih amplitudah, kar da v seštevku spet 16 stanj in spet hitrosti do 9600 b/s pri 2400 Bd.

3.2.5 Frekvenčna modulacija (FM)

Pri frekvenčni modulaciji se v odvisnosti od modulacijskega signala neposredno spreminja frekvenca nosilnega vala. Ker se frekvenca v splošnem od trenuka do trenutka spreminja, ji rečemo tudi *trenutna frekvenca*. Frekvenčno modulacijo z binarnim modulacijskim signalom imenujemo *modulacijo s frekvenčnim premikom* (ang. **F**requency **S**hift **K**eying ali s kratico **FSK**). Frekvenčno moduliran signal $u_m(t)$ je konstantne amplitude U_c, trenutna frekvenca pa se mu spreminja v odvisnosti od amplitude modulacijskega signala okrog frekvence nosilnega vala

Tabela 2: Preslikava četverke binarnih simbolov v trenutku n v signalni element (preslikava T_1) po priporočilu V.22bis.

y_{1_n}	y_{2_n}	b_{3_n}	b_{4_n}	Re_n	Im_n
0	0	0	0	-1	-1
0	0	0	1	-3	-1
0	0	1	0	-1	-3
0	0	1	1	-3	-3
0	1	0	0	1	-1
0	1	0	1	1	-3
0	1	1	0	3	-1
0	1	1	1	3	-3
1	0	0	0	-1	1
1	0	0	1	-1	3
1	0	1	0	-3	1
1	0	1	1	-3	3
0	1	0	0	1	1
0	1	0	1	3	1
0	1	1	0	1	3
0	1	1	1	3	3



Slika 37: Primer modulacije s frekvenčnim premikom (FSK).

f_c,

$$u_m(t) = U_c \cos(\omega_c \pm \Delta\omega)t = U_c \cos 2\pi(f_c \pm \Delta f).$$

Moduliran signal vsebuje dve trenutni frekvenci, ena frekvenca ($f_1 = [f_c + \Delta f)$ pomeni binarni simbol ena, druga frekvenca ($f_0 = [f_c - \Delta f]$ pomeni binarni simbol nič. Običajno je $\Delta f << f_c$ in modulirani signal ima še vedno sinusu podobno obliko, le da se mu od trenutka do trenutka spreminja frekvenca. Primer frekvenčno moduliranega signala prikazuje slika 37. Kadar je frekvenca signala ničle in enice mnogokratnik hitrosti oddajanja binarnih simbolov, kar pa nikakor ni nujno, govorimo o frekvenčni modulaciji brez faznega skoka ali s kratico MSK (ang. Minimum Shift Keying). V narisanem primeru gre torej za MSK modulacijo.

Modulacija s frekvenčnim premikom se uporablja predvsem v (asinhronih) modemih za počasnejše prenose, tja do 2400 b/s.

3.2.6 Pojasnilo k fazni in frekvenčni modulaciji

Zapišimo še enkrat cosinusni signal z amplitudo 1:

$$u(t) = 1 \cos \alpha(t) = \cos(\omega t + \Phi). \tag{16}$$

Argument kotne funkcije $\alpha(t)$ se linearno spreminja s časom, kotna frekvenca $\omega = \frac{d\alpha(t)}{dt}$, torej sprememba kota v časovni enoti, pa je konstantna. Signal niha s konstantno hitrostjo oziroma frekvenco f in prehodi skozi nič si sledijo v enakomernih časovnih presledkih. Tako spreminjanje signala imenujemo harmonično nihanje (valovanje).

Naj se sedaj faza Φ s časom spreminja v odvisnosti od informacijskega signala, ω pa naj bo konstantna in enaka ω_c, $\alpha(t) = \omega_c t + \Phi(t)$. Tak signal smo imenovali *fazno moduliran signal*. Fazno moduliranemu signalu pa se s časom spreminja tudi frekvenca, saj je

$$\omega(t) = \frac{d\alpha(t)}{dt} = \omega_c + \frac{d\Phi(t)}{dt}$$

in $f(t) = f_c + \frac{1}{2\pi}\frac{d\Phi(t)}{dt}$. Torej je fazno moduliran signal moduliran tudi frekvenčno. Ker se $\omega(t)$ od trenutka do trenutka spreminja jo imenujemo *trenutna kotna frekvenca*. Iz enakih razlogov imenujemo $f(t)$ *trenutna frekvenca*.

Sedaj pa vzemimo, da je od informacijskega signala $u_i(t)$ neposredno odvisen frekvenčni odmik trenutne frekvence $f(t)$ od centralne frekvence f_c,

$$2\pi(f(t) - f_c) \propto u_i(t).$$

V skladu z ustaljeno terminologijo smo signal v tem primeru imenovali frekvenčno moduliran signal. Toda trenutna frekvenca se v tem primeru spreminja enako kot bi se spreminjala v primeru fazne modulacije, le da je *odvod* faznega kota $\Phi(t)$ direktno odvisen od informacijskega signala in ne sam fazni kot,

$$\frac{d\Phi(t)}{dt} \propto u_i(t).$$

Ko govorimo o frekvenčni modulaciji bi lahko govorili tudi o fazni modulaciji. Iz podobnih razlogov bi lahko fazno modulacijo obravnavali kot frekvenčno modulacijo. Vendar govorimo o fazni modulaciji tedaj, kadar se v odvisnosti od informacijskega signala *neposredno* menja faza in rečemo, da je modulacija frekvenčna, če se v odvsinosti od informacijskega signala *neposredno* spreminja frekvenca.

Za $\frac{1}{2\pi}\frac{d\Phi(t)}{dt} << f_c$ se trenutna frekvenca bistveno ne oddaljuje od centralne frekvence. Časovni potek moduliranega signala ima še vedno sinusu podobno obliko, prehodi skozi nič pa se vrstijo v neenekomernih časovnih presledkih. To drži tako za frekvenčno kot za fazno (skratka *kotno*) moduliran signal. Iz časovnega poteka signala se ne da ugotoviti za katero vrsto modulacije gre.

3.2.7 Frekvenčni spekter kotno moduliranega signala

Na osnovi slike 37 bi bilo napačno sklepati, da frekvenčni spekter moduliranega signala vsebuje le frekvenco ničle in enice. Kot bomo videli, je za prenos frekvenčno ali fazno moduliranega signala teoretično potreben neskončen frekvenčni pas. Na srečo je skoraj vsa energija signala zgoščena okrog centralne frekvence nosilca in za kvaliteten prenos zadostuje že frekvenčna širina kanala, ki ni dosti večje od dvakratne najvišje frekvence modulacijskega signala.

Opazujmo fazno modiliran kosinusni signal frekvence $f_c, \omega_c = 2\pi f_c$. Naj bo mudulacijski signal frekvence $\omega_i = 2\pi f$ sinusen

$$u_m(t) = \cos(\omega_c t + \beta \sin \omega_i t), \tag{17}$$

Faza signala je odvisna od amplitude nosilca, in se maksimalno odmakne za $\pm\beta$. Sorazmerno konstanto β imenujemo fazni odmik ali tudi modulacijski indeks. Trenutna frekvenca moduliranega signala je $f(t) = f_c + \beta f_i \cos \omega_i t)$. Maksimalen frekvenčni odmik je $\Delta f = \beta f_i$ in $\beta = \frac{\Delta f}{f_i}$. Trenutna frekvenca leži na intervalu $f_c \pm \Delta f$, kar pa ne pomeni, da *vse* spektralne komponente ležijo na tem intervalu. Za izraz (17) velja naslednja enakost:

$$\begin{aligned}&\cos(\omega_c t + \beta sin\omega_i t) = \\ &\cos\omega_c t \cos(\beta \sin \omega_i t) - \sin\omega_c t \sin(\beta \sin \omega_i t) = A(t)\cos\omega_c t - B(t)\sin\omega_c t.\end{aligned}$$

Amplitudi $A(t)$ in $B(t)$ sta odvisni od modulacijskega signala. Lahko si razlagamo, da je signal (17) nastal iz razlike dveh amplitudno moduliranih signalov. Funkcija $A(t)$ je soda periodična funkcija s kotno frekvenco ω_i. Podobno je funkcija $B(t)$ liha in periodična funkcija z enako kotno frekvenco ω_i. Torej se da obe razviti v Fourierjevo vrsto. Z razvojem dobimo:

$$\begin{aligned}\cos(\beta \sin \omega_i t) &= J_0(\beta) + 2J_2(\beta)\cos 2\omega_i t \\ &+ 2J_4(\beta)\cos 4\omega_i t + \cdots + 2J_{2n}\cos 2n\omega_i t + \cdots \\ \sin(\beta \sin \omega_i t) &= 2J_1(\beta)\sin\omega_i t + 2J_3(\beta)\sin 3\omega_i t + \\ &+ \cdots + 2J_{2n-1}(\beta)\sin(2n-1)\omega_i t + \cdots\end{aligned}$$

pri čemer so amplitude spektralnih komponent $J_n(\beta)$ Besselove funkcije prvega reda stopnje n. Če vstavimo rezultat (18) v (17) in upoštevamo enakosti

$$\begin{aligned}\cos\alpha\cos\beta &= 1/2\cos(\alpha-\beta) + 1/2\cos(\alpha+\beta), \\ \sin\alpha\sin\beta &= 1/2\cos(\alpha-\beta) - 1/2\cos(\alpha+\beta),\end{aligned}$$

dobimo:

$$\begin{aligned}u_m(t) = J_0(\beta)\cos\omega_c t &- J_1(\beta)[\cos(\omega_c-\omega_i)t - \cos(\omega_c+\omega_i)t] \\ &+ J_2(\beta)[\cos(\omega_c-2\omega_i)t - \cos(\omega_c+2\omega_i)t] \\ &- J_3(\beta)[\cos(\omega_c-3\omega_i)t - \cos(\omega_c+3\omega_i)t].\end{aligned} \tag{18}$$

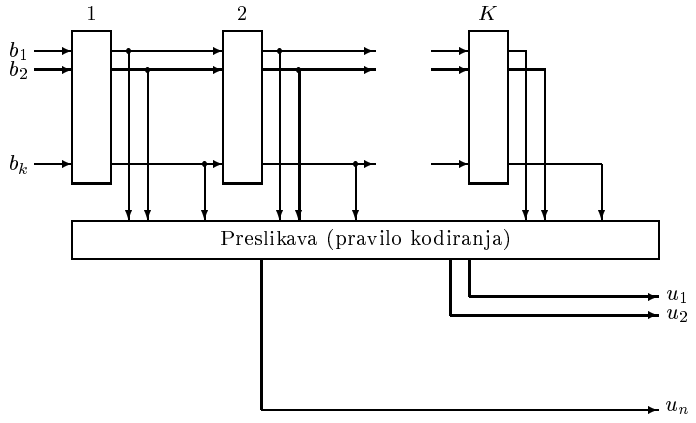
Amplitudni spekter fazno (ali frekvenčno) moduliranega signala $\cos \omega_c t$, ki je moduliran s sinusnim signalom s frekvenco ω_i, je sestavljen iz frekvence nosilca ω_c in neskončnega števila višjih harmonskih komponent $(\omega_c \pm n\omega_i)t \quad (n = 1, 2, \ldots,$ ki se razprostirajo na obeh straneh nosilca s korakom ω_i. Za prenos kotno moduliranega signala bi teoretično potrebovali neskončen frekvenčni pas. Izkaže se, da je kar 98° moči signala strnjenega okrog centralne frekvence in za dovolj kvaliteten prenos signala zadostuje že frekvenčna širina kanala:

$$F = 2(\Delta f + f_i). \tag{19}$$

Formula (19) je znana pod imenom *Carsonovo pravilo* in velja za modulacijo z analognim ali digitalnim signalom.

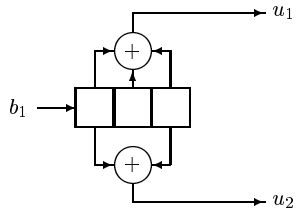
3.2.8 Mrežno kodiranje in konvolucijski kodi

Mrežno ali trellis[15] kodiranje ima ime po trellis diagramu, s katerim je možno predstaviti konvolucijski kodirnik in/ali dekodirnik. *Konvolucijski kodi* spadajo v družino linearnih (grupnih) kodov. Uporabljajo se za kodiranje dolgih nizov simbolov. Slika 38 shematično ponazarja konvolucijski kodirnik. V kodirnik vstopa neprekinjen niz simbolov. Niz simbolov se deli v skupine po k simbolov, $\mathbf{b} = \{b_1, b_2, \ldots, b_k\}$. Skupina simbolov $\mathbf{b}$ sočasno vstopi v kodirnik. Kodirnik pomni tekočo skupino simbolov in še $(K-1)$ preteklih skupin. Teh $(K-1)$ preteklih skupin definira trenutno stanje kodirnika. Možnih je torej $2^{(K-1)}$ stanj. Tekoča vhodna skupina simbolov se v odvisnosti od tekočega stanja kodirnika preslika (zakodira) v n izhodnih simbolov $\mathbf{u} = \{u_1, u_2, \ldots, u_n\}$. Konvolucijski kod opišemo s tremi parametri (n, k, K) ter pravilom kodiranja, ki preslika vhodne simbole v izhodne simbole.



Slika 38: Konvolucijski kodirnik (n, k, K).

[15] V angleškem jeziku pomeni *trellis* mrežno oporo, po kateri se vzpenja rastlina plezalka, npr. vinska trta



Slika 39: Konvolucijski kodirnik s parametri $(n, k, K) = (2, 1, 3)$.

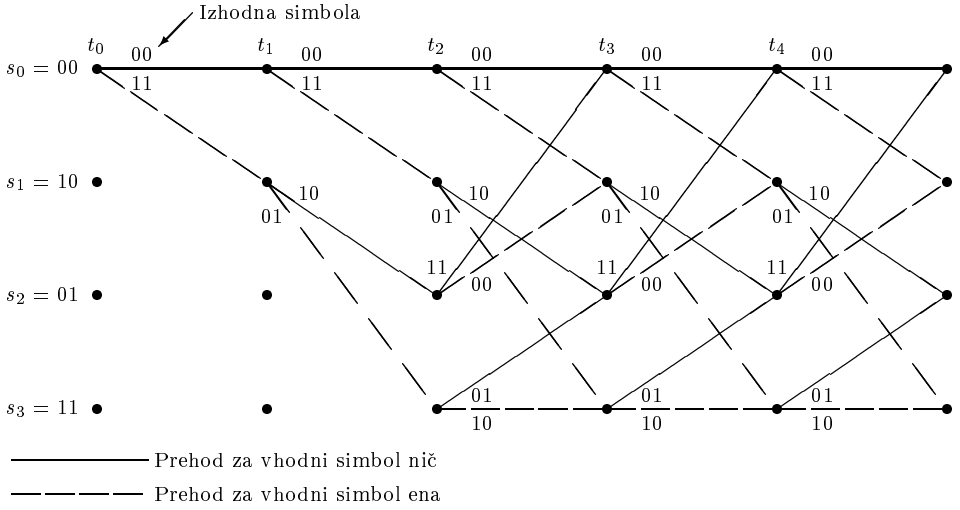
Za primer vzemimo konvolucijski kod $(n, k, K) = (2, 1, 3)$. V kodirnik posamično $(k = 1)$ vstopajo simboli, ki se z dvema predhodnima simboloma $(K = 3)$ preslikajo v po dva $(n = 2)$ izhodna simbola, glej sliko 39.

Preslikavo vhodnega simbola v dva izhodna simbola lahko opišemo z diagramom prehajanja stanj. Predhodna dva simbola definirata štiri stanja, $s_0 = 00, s_1 = 10, s_2 = 01$ in $s_3 = 11$. Kodirnik spremeni stanje v odvisnosti od vhodnega simbola, ki je nič ali ena. Pri prehodu nastaneta dva izhodna simbola. Diagram prehajanja stanj, ki vsebuje še časovno dimenzijo, prikazuje slika 40. Temu diagramu pravimo trellis diagram. V začetnem trenutku t_0 se kodirnik nahaja v stanju $s_0 = 00$. V primeru, da pride simbol nič, oddamo 00 in še naprej ostajamo v stanju s_0. V primeru, da pride simbol ena, pa oddamo 11 in se selimo v stanje s_1. Ob vsakem vhodnem simbolu se premaknemo po trellis diagramu za eno stopnjo v desno.

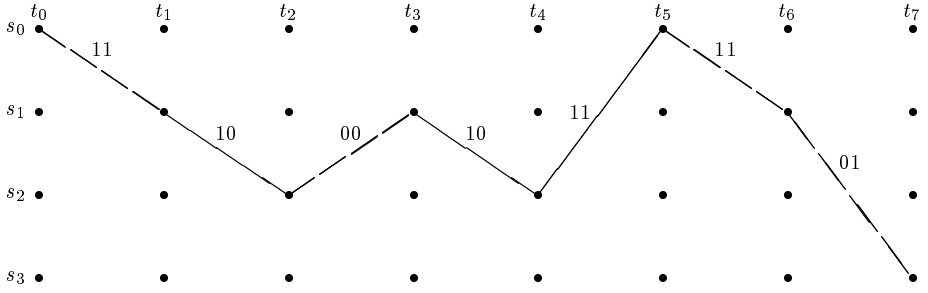
Vsakemu zaporedju vhodnih simbolov ustreza v trellis diagramu ena sama **pot** od začetnega do končnega trenutka. Isti trellis diagram uporabimo tudi pri dekodiranju. Pri oddajanju kodiramo po trellis diagramu in pri sprejemanju dekodiramo po trellis diagramu. V primeru, da med prenosom ne pride do napake, je pot skozi diagram na sprejemni strani enaka poti skozi diagram na oddajni strani. V nasprotnem primeru za sprejeto zaporedje simbolov pot skozi trellis ne obstaja, razen če se oddano zaporedje zaradi napak popolnoma ne spremeni v kakšno drugo možno zaporedje oddanih simbolov. Naloga dekodirnika je, da na osnovi sprejetih simbolov in z upoštevanjem trellis diagrama (pravila kodiranja) poišče tisto pot, ki je bila najverjetneje oddana oziroma tisto, ki se od sprejetega zaporedja najmanj razlikuje. Algoritem dekodiranja, ki išče optimalno pot skozi trellis diagram, je predlagal Viterbi in se tudi po njem imenuje.

3.2.9 Dekodiranje po Viterbiju

Delovanje algoritma Viterbija bomo razložili na primeru kodirnika s slike 39. Predpostavimo, da je zaporedje vhodnih simbolov $\mathbf{b} = 1\ 0\ 1\ 0\ 0\ 1\ 1$. Naj bo s_0 začetno stanje kodirnika (register vsebuje vrednost nič). Temu zaporedju vhodnih simbolov ustreza pot skozi trelis diagram, ki je skicirana na sliki 41. Zakodirano zaporedje izhodnih simbolov je $\mathbf{u} = 11\ 10\ 00\ 10\ 11\ 11\ 01$. To



Slika 40: Trellis diagram za obravnavani primer kodirnika.

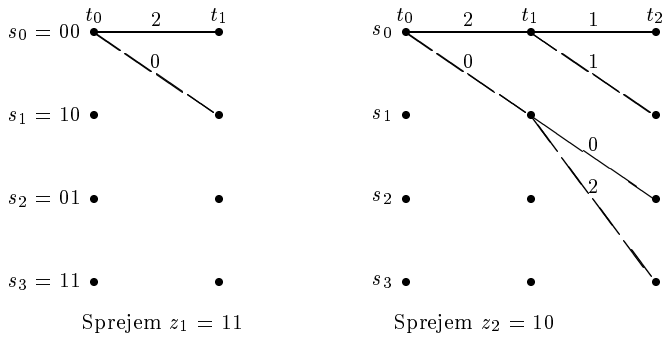


Slika 41: Pot skozi trellis za zaporedje vhodnih simbolov 1 0 1 0 0 1 1.

zaporedje pošljemo v kanal. Naj bo sprejeto zaporedje simbolov na drugi strani kanala enako $\mathbf{z} = 11\ 10\ 10\ 10\ 11\ 11\ 01$. Sprejeto zaporedje se od oddanega razlikuje na tretjem paru simbolov (peti binarni simbol).

Viterbijev algoritem skuša za sprejeto zaporedje simbolov najti tisto možno zaporedje oddanih simbolov, ki se od sprejetega najmanj razlikuje. V ta namen sproti računa razdaljo med sprejetim zaporedjem simbolov in potencialno možnimi potmi skozi trelis diagram. Ker vsaka pot ustreza enemu od možnih zaporedij oddanih simbolov bo na ta način našel tudi najverjetnejše zaporedje. Algoritem temelji na odstranjanju slabih in ohranjanju dobrih poti. Če v nekem trenutku v isto stanje vodita dve poti, algoritem eliminira tisto z večjo razdaljo oziroma večjo "težo". Tako algoritem ohranja do vsakega stanja izmed dveh možnih poti samo ugodnejšo pot.

Poglejmo, kako deluje Viterbijev algoritem na našem primeru. Ker še nismo



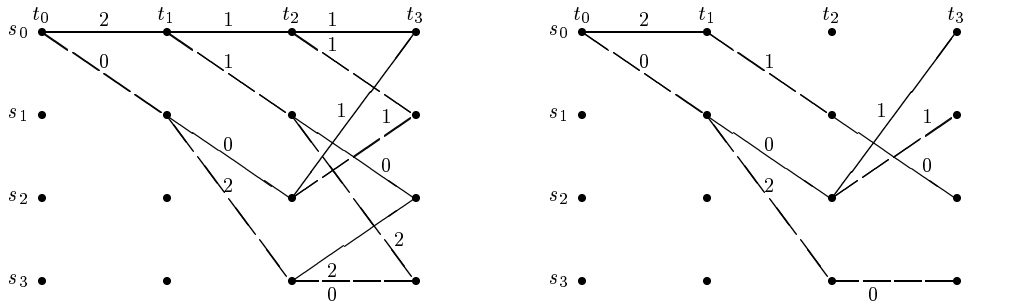
Slika 42: Dekodiranje po algoritmu Viterbija - stanje po sprejemu prvega in drugega para simbolov.

izbrali definicije razdalje, naredimo to sedaj. Naj bo to Hammingova razdalja[16] Ob času t_1 sprejemnik sprejme prvi par simbolov $z_1 = 11$. Iz stanja s_0 ob času t_0 sta možni dve poti. Razdalja sprejetega para od veje, ki kodira simbol ena je enaka nič, $d(11, z_1) = d(11, 11) = 0$. Razdalja do veje, ki kodira simbol nič pa je $d(00, 11) = 2$. Vsako od vej ustrezno veji označimo z razdaljo oziroma "utežjo". V stanji s_0 in s_1 ob času t_1 vodi samo po ena pot in eliminacija ni možna (slika 42). Postopek ponovimo ob sprejemu naslednjega para $z_2 = 10$. V vsako od štirih stanj ob času t_2 pelje še vedno samo po ena pot in eliminacija tudi tokrat ni potrebna. V naslednjem trenutku t_3 sprejmemo par $z_3 = 10$ in spet izračunamo razdaljo do vsake veje. Teh je sedaj osem. V vsako stanje v času t_3 vodita po dve veji oziroma poti, zato po eno eliminiramo (slika 43). Odločimo se, da obdržimo pot z manjšo težo, saj je "bližja" sprejetemu zaporedju simbolov in ustreza "verjetnejšemu" zaporedju oddanih simbolov. Težo poti določa vsota uteži posameznih vej. Na primer, v stanje $s_0(t_3)$ peljeta dve poti. Teža ene je $2 + 1 + 1 = 3$ in druge $0 + 0 + 1 = 1$. Prvo odstranimo in obdržimo drugo. Podobno postopamo s potmi do ostalih treh stanj. V času t_4 sprejmemo zaporedje $z_4 = 10$ (slika 44). Spet poiščemo razdalje do vej in teže poti ter ohranimo samo verjetnejše. Rezultat vidimo na zadnji skici. Opazimo, da je za prehod iz trenutka t_0 v trenutek t_1 ostal samo še en prehod in sicer po veji, ki kodira simbol ena. Zato se odločimo, da je sprejeti simbol enica in ta je tudi v resnici bil oddan. Na enak način nadaljujemo dekodiranje do konca zaporedja.

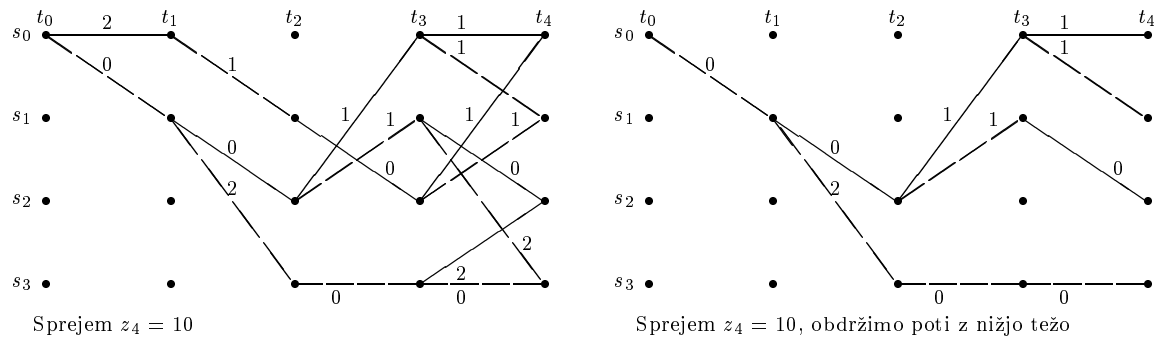
3.2.10 Modemi

Modeme običajno delimo po načinu sinhronizacije, po hitrosti prenosa, po smernosti prenosa in po vrsti modulacije. Modem imamo za napravo fizičnega sloja, vendar opravljajo moderni modemi številne funkcije višjih slojev. Pri sinhronih

[16] Hammingova razdalja je definirana s številom mest, na katerem se dve enako dolgi zaporedji binarnih simbolov razlikujeta. Na primer, razdalja med 1001 in 0011 je ena.



Slika 43: Dekodiranje po algoritmu Viterbija - stanje po sprejemu tretjega para simbolov.



Slika 44: Dekodiranje po algoritmu Viterbija - stanje po sprejemu četrtega para simbolov.

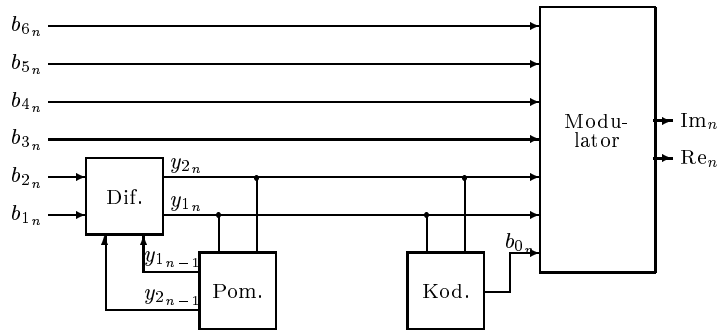
modemih se morata generatorja takta oddajnega in sprejemnega modema pred prenosom sinhronizirati-izenačiti po frekvenci in fazi. Zato mora biti modulacija, ki se koristi v sinhronih modemih taka, da lahko sprejemni modem iz moduliranega signala obnovi tudi signal za sinhronizacijo svojega generatorja takta z oddajnim. Nasprotno se v asinhronih modemih informacija o taktu ne prenaša.

Hitrost prenosa je praktično vedno povezana z zahtevnostjo izvedbe in s ceno. Hitri modemi za prenos po telefonski liniji danes dosegajo hitrosti 33600 bitov na sekundo in več (priporočilo ITU-T V.90) , s čemer je teoretična zgornja meja analogne telefonske linije s frekvenčno širino 3000 Hz praktično dosežena. Visoke hitrosti prenosa se danes dosegajo z naprednimi postopki faznega in amplitudnega moduliranja ter mrenega (ang. trellis) kodiranja, ob boljši kvaliteti prenosnih poti.

Glede na sočasnost prenosa so modemi pol-dupleksni ali dupleksni. Sočasnost prenosa se dosega ali s frekvenčnim multipleksiranjem ali z 'izničenjem odmeva' ali pa se koristi štirižični prenos. Komutiran naročniški vod je dvožičen, zato je za štirižičen prenos potrebna najeta (zakupljena) linija.

Modemi se še najbolj razlikujejo po dodatnih funkcijah, ki jih razen modulacije še opravljajo. Med pomembne dodatne funkcije sodi izenačevanje slabljenja, mešanje signala (scrambling), kodiranje za odkrivanje in popravljanje napak, zgoščevanje podatkov in šifriranje, avtomatsko pozivanje in odzivanje, avtomatski preklop na nižjo hitrost in drugo.

Mešanje signala (ang. Scrambling) se koristi v vseh kvalitetnejših modemih. Na oddajni strani se zaporedje simbolov, ki vstopa v modem, naključno meša, zatem po potrebi kodira in nato modulira. Na sprejemni strani je potreben obraten postopek. Z mešanjem se skuša preprečiti periodično ponavljanje enakih bitnih vzorcev in daljših zaporedij ničel ali enic. S tem se deseže skoraj konstantno moč signala, poveča se odpornost na motnje, sinhronizacija sprejemnika (obnovitev takta iz sprejetega signala) pa je manj odvisna od resničnega zaporedja informacijskih bitov. Za odkrivanje in tudi popravljanje napak se v modemih koristijo konvolucijskimi kodi. *Trellis* kodiranje in dekodiranje ali s kratico TCM (ang. Trellis Coded Modulation) najdemo v vsakem sodobnem modemu za višje hitrosti prenosa. Na primer, 14400 b/s modem po priporočilu V.32 bis deli prihajajoče zaporedje bitov v skupine po šest bitov, dva izmed šestih bitov se najprej diferenčno kodirata glede na predhodna bita, podobno kot po priporočilu V.22 bis (slika 36). Diferenčno kodirana bita služita za vhod v konvolucijski kodirnik, glej sliko 45. Konvolucijski kodirnik generira dodatni (redundančni bit), ki skupaj s šestimi informacijskimi biti določa enega od 128 signalnih elementov modulatorja, od tu oznaka 128-TCM. Namesto 64 signalnih elementov jih imamo zdaj 128 ali še enkrat več, kar sprejemniku omogoči popravljanje nekaterih napak. Število (neodkritih) napak na telefonski liniji se na ta način efektivno zmanjša tudi za



Slika 45: 128-TCM modulacija po priporočilu CCITT V.32 bis.

tri velikostne razrede.

Na pomenu pridobivajo tudi postopki za zgoščevanje podatkov, kajti vse več je naprav, ki ustvarjajo velike količine podatkov z visoko stopnjo redundance. Z zgoščevanjem visoko redundantnih podatkov se da dejansko hitrost prenašanja informacije večkrat povečati. S tem se uporabniku ustvarja vtis, da je hitrost prenašanja podatkov večja, kot pa je v resnici.

Z razvojem modemov so napredovali tudi standardi in priporočila, ki predpisujejo zahtevane tehnične lastnosti modemov. Omenili smo že priporočilo V.32 bis[17] iz leta 1991, ki predvideva dupleksni prenos z dušenjem odmeva po komutirani telefonski liniji pri hitrosti do 14400 bitov na sekundo s trellis načinom kodiranja. Priporočilo V.29 predpisuje modeme za faksimile (Faxmodeme), priporočilo V.33 je za prenos po najetih linijah do hitrosti 14400 b/s, V.42 se nanaša na postopke za popravljanje napak in je združljiv z de facto standardom MNP4 (Microcom Networking Protocol), V.42 bis priporoča zgoščevanje podatkov, podobno kot de facto standard MNP5, a z njim ni združljiv, i.t.d. Leta 1994 je stopil v veljavo standard V.34, ki se nanaša na modeme s hitrostjo do 28800 b/s, kasneje je bil dopolnjen s specifikacijo za modeme do 33600 b/s. Naslednja tabela prikazuje hitrost prenosa, modulacijsko hitrost, vrsto modulacije in frekvenco nosilca nekaterih bolj znanih modemov za prenos po javnih komutiranih linijah po priporočilih CCITT (ITU-T).

[17] *bis* pomeni drugo predelavo priporočila, *ter* pa tretjo

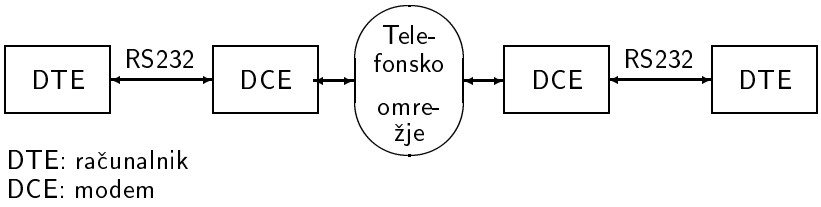
Oznaka	Hitrost prenosa [b/s]	Mod.hitrost [Baud]	Modulacija	Nosilna frekvenca [Hz]
V.90	56k/33.6k	3429	PCM/TCM	1800
V.34 (1994)	28800	3200	960-TCM	1829/1920
V.32 bis	14400	2400	128-TCM	1800
V.32	9600	2400	32-TCM	1800
V.32 uncoded	9600	2400	16-QAM	1800
V.32	4800	2400	4-DPSK	1800
V.29	9600	2400	16-QAM	1700
V.27 ter	4800	1600	8-PSK	1800
V.23	1200/75	1200/75	FSK	1700 ± 400/ 420 ± 30
V.22 bis	2400	600	16-QAM	1200/2400
V.22 (Bell 212A)	1200	600	4-DPSK	1200/2400
V.21 (Bell 103F)	300	300	FSK	1080 ± 100/ 1750 ± 100

Danes prevladujejo "pametni" modemi. Ti modemi ne potrebujejo nadzornih sponk (kot so RTS, CTS, ...) za usklajevanje z računalnikom. Usklajevanje poteka z ukaznimi zaporedji znakov na oddajni in sprejemni sponki, na katerih teče tudi prenos podatkov.

3.3 Nekateri standardi fizičnega sloja

Standardizacija je dejavnost pooblaščenih narodnih in mednarodnih organizacij, kot so ISO (International Organization for Standardization), CCITT[18](Comite Consultatif International Telephonique et Telegraphique), IEC (International Electrotechnical Commission), IEEE (Institute of Electrical and Electronics Engineers) in druge. Zaradi vpliva ameriških proizvajalcev računalniške opreme na svetovnem trgu so ameriški standardi tudi svetovnega pomena. Omeniti je treba vsaj ANSI (American National Standards Institute) in EIA (Electronic Industries Association). Standardi, ki jih izdajajo za to pooblaščene organizacije, so 'uradni' ali (lat.) *de iure* standardi. Na standardizacijo vplivajo tudi veliki proizvajalci, na primer IBM (International Bussines Machines), HP (Hewlett Packard), manjši proizvajalci so enostavno prisiljeni izdelovati opremo, ki je združljiva z opremo velikih, ta pa tako postane 'standardna'. Standardi vplivnih proizvajalcev v resnici niso pravi standardi, ampak so posledica dejanskih razmer in jim zato pravimo *de facto* standardi. Res pa je, da so de facto standardi večkrat podlaga za dokončno uradno standardiziranje.

[18]CCITT je svetovalni svet mednarodnega združenja za telekominikacije, ki deluje pod okriljem združenih narodov ITU (International Telecommunication Union). Leta 1994 je CCITT prešel v ITU-T



Slika 46: Povezava računalniša z računalnikom preko telefonskega omrežja.

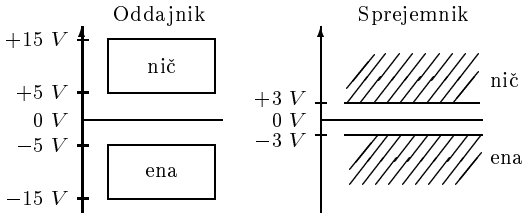
V tem podpoglavju bomo obravnavali nekatere predpise EIA in CCITT (ITU-T) , ki se navezujejo na fizični sloj in določajo lastnosti postopkov, vmesnikov in naprav za prenos podatkov v asinhroni ali sinhroni (bitno) serijski obliki. Ameriški EIA standardi imajo predpono RS, *priporočila* svetovalnega komiteja CCITT pa imajo predpono V. ali X. Priporočila V. se nanašajo na prenos podatkov v javnih telefonskih omrežjih, priporočila X. pa se nanašajo na prenos podatkov v javnih podatkovnih omrežjih.

3.3.1 Standard RS232 in priporočilo V.24

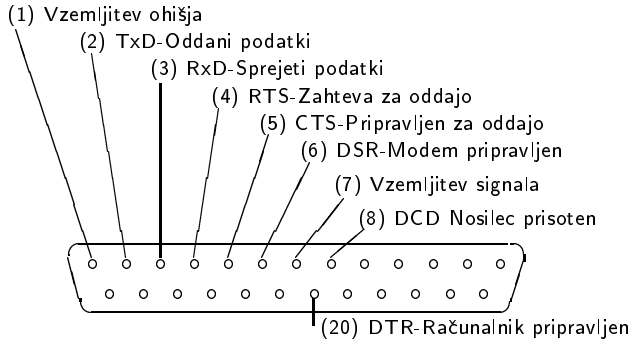
Standard EIA RS232 je verjetno eden najbolj znanih komunikacijskih standardov. RS232C/D (popravek C iz leta 1969 in popravek D iz leta 1986) uradno določa, kako naj povežemo napravo tipa DTE (Data Terminal Equipment) z napravo tipa DCE (Data Communication Equipment). DTE je standardna oznaka za računalniško napravo, kamor sodijo tako računalnik kot terminal ali tiskalnik. DCE je oznaka za komunikacijsko napravo. Sem sodijo različne vrste modemov. Da povežemo dve računalniški napravi (postaji) preko telefonskega omrežja, sta potrebna dve povezavi RS232, kot prikazuje slika 46.

Standard RS232 sestavljajo štirje deli, ki določajo mehanske, električne in funkcionalne lastnosti naprav in povezav. Priporočilo CCITT V.24 je podobno standaru RS232, električne zahteve so predpisane ločeno v priporočilu V.28 (RS232C $\approx$ V.24 + V.28). Standard predpisuje uporabo 25 polnega spojnika (DB25) in tudi razporeditev signalov na spojniku. Od 25 sponk je predpisanih 21, ostale so proste. Še to, kadar proizvajalec jamči združljivost (kompatibilnost) opreme po standardu RS232C, to še ne pomeni, da so v celoti realizirane vse možnosti, ki jih predvideva standard. Običajno so realizirani le nekateri tokokrogi, pač odvisno od konkretne naprave. Vendar pa tisti tokokrogi, ki so realizirani, ne smejo v ničemer kršiti določil standarda.

Standard določa, da je nosilec informacije napetostni nivo signala, glej sliko 47. Oddajna naprava mora za nivo signala logične enice (ang. Mark) zagotoviti manj od $-5\ V$, tipično $-12\ V$. Sprejemna naprava mora pravilno razpoznati nivo signala za logično enico, če je nivo signala vsaj $-3\ V$. Zahtevani nivo signala



Slika 47: Mejne vrednosti napetostnih nivojev signala na oddajni in sprejemni strani po standardu RS232.



Slika 48: Izgled konektorja po standardu RS232 in imena važnejših sponk (DTE stran).

logične ničle (ang. Space) na oddajni strani je več od $+5\ V$, tipično $+12\ V$, na sprejemni strani pa mora biti višji od $+3\ V$. Trajanje spremembe signala z enega logičnega nivoja na drugi logični nivo ne sme presegati 4 odstotkov trajanja enega nivoja (bita). Na prehodni čas vpliva kapacitivnost povezovalnega kabla, ki je tipično med 120 in 150 pikofaradov na meter. Trajanje nivoja signala enega bita je odvisno od hitrosti prenosa. Ker znaša najvišja dovoljena kapacitivnost povezovalnega kabla 2500 pF, je najdaljša dovoljena dolžina kabla približno 15 metrov ($\approx$ 50 ft - feetov). Ko smo že pri hitrostih, navedimo standardizirane hitrosti prenosa (v bitih na sekundo): 19200, 9600, 4800, 2400, 1200, 600, 300, 150, 110, 75, 50. Standard RS232 predvideva precej kratko dolžino kabla, vendar se v praksi dosega popolnoma zadovoljiv prenos tudi pri večkrat daljšem kablu kot je dovoljena. Dejstvo je tudi, da se pri kratkih povezavah nekaj metrov da doseči hitrosti do 115000 b/s.

Funkcionalne zahteve standarda določajo pomen posameznih signalov in tokokrogov pri prenosu podatkov. Predpisi določajo takšne stvari, kot je na primer nadzor nad modemom. Določen je postopek vzpostavljanja zveze, nadzor nad zvezo in podobno. Standard RS232 ne določa postopka za delo z avtomatskim odzivnikom, to je predpisano v spremljajočem dokumentu z oznako EIA RS366, ki je primerljiv s pripočilom CCITT V.25. Tabela 3 prikazuje pomen (ime) važnejših tokokrogov oziroma priključkov na konektorju ter standardno oznako po predpisih EIA in priporočilih CCITT, slika 48 pa izgled konektorja. Imena signalov dajo

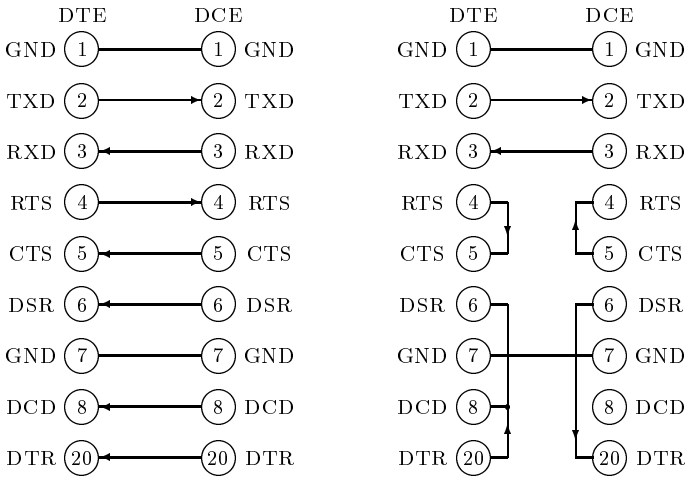
vedeti, da imajo signali pravi pomen pravzaprav le pri povezavi naprave DTE z napravo DCE. Kadar računalnik obratuje, postavi signal DTR (Data Terminal Ready). Podobno velja za modem. Kadar modem deluje, postavi signal DSR (Data Set Ready).[19] Če računalnik želi oddajati, postavi modemu zahtevo za oddajo na sponki RTS (Request To Send). Modem se pripravi za oddajo (začne generirati nosilni signal) in po nekem krajšem času (tipično 200 ms) postavi signal CTS (Clear To Send). Modem na sprejemni strani javi prisotnost nosilnega signala na sponki DCD. Sedaj ima lokalni računalnik možnost, da na sponki TxD (Transmited Data) odda podatke, ki jih računalnik na oddaljeni strani sprejema na sponki RxD (Received Data). Dve možni povezavi naprave DTE z napravo DCE prikazuje slika 49.

Št. sponke	Oznaka EIA	Oznaka CCITT	Ime (pomen) signala
1	AA	101	Vzemljitev ohišja (Protective Ground)
7	AB	102	Vzemljitev signala (Signal Ground)
2	BA	103	Oddani podatki (TxD - Transmited Data)
4	CA	105	Zahteva za oddajo (RTS - Request To Send)
20	CD	108.2	Terminal pripravljen (DTR - Data Terminal Ready)
3	BB	104	Sprejeti podatki (RxD - Received Data)
5	CB	106	Pripravljen na oddajo (CTS - Clear To Send)
6	CC	107	Modem pripravljen (DSR - Data Set Ready)
8	CF	109	Nosilec prisoten (DCD - Data Carrier Detected)
22	CE	125	Znak poziva (RI - Ring Indicator)

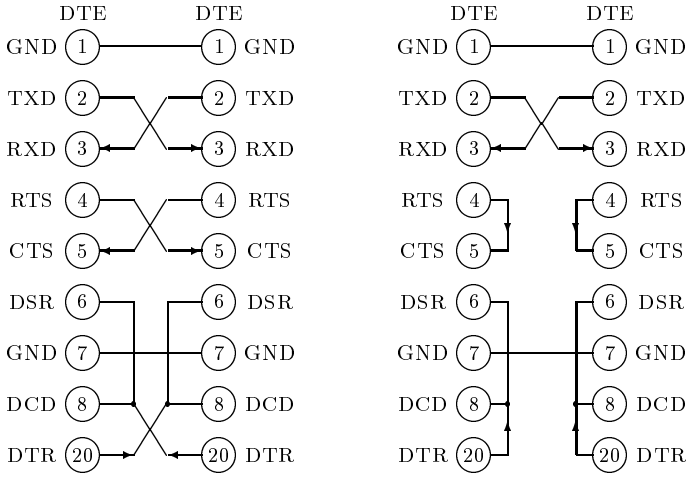
Tabela 3: Razporeditev signalov na konektorju (DTE stran) ter oznake in pomen važnejših signalov.

Standard RS232 uradno ne predpisuje, kako naj neposredno povežemo dve napravi tipa DTE (računalnik z računalnikom, tiskalnik ali terminal z računalnikom). Sicer pa standard tega ne prepoveduje in v praksi se največkrat izkorišča prav ta možnost. Pri neposredni povezavi dveh naprav tipa DTE je večina nadzornih signalov odveč. Za neposredno povezavo računalnika s terminalom (ali računalnika z računalnikom) zadostuje že trižični kabel. Povezati moramo vzemljitvi signala (sponki 7) na obeh konektorjih, oddajno sponko (2) s sprejemno sponko (3) in obratno. V žargonu rečemo, da je potreben križan kabel. Napravi, ki ju povežemo neposredno, 'prevaramo' tako, da na obeh konektorjih (na obeh straneh kabla) kratko vežemo sponko 4 s sponko 5 ter sponki 6 in 8 s sponko 20. Ostale tri sponke (oddaja, sprejem, zemljitev) povežemo tako, kot je narisano na sliki 50 (levo). Neposredna povezava dveh računalniških naprav z možnostjo usklajevanja (ang. handshaking) je narisana na sliki 50 (desno).

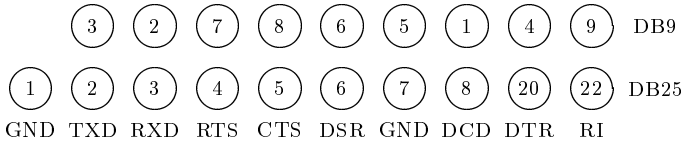
[19] Data Set je Bell-ovo ime za modem



Slika 49: Povezava DTE in DCE po standardu RS232 z usklajevanjem (levo) in brez usklajevanja (desno) z nadzornimi signali.



Slika 50: Dve najpogostejši izvedbi ničelnega modema.



Slika 51: Sponke in signali na DB25 ter odgovarjajoče sponke na DB9.

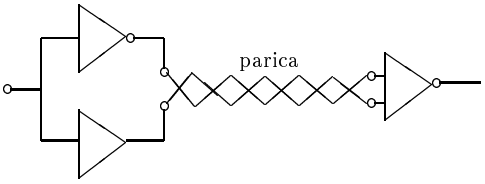
Oddajna sponka ene naprave (oziroma konektorja) je povezana s sprejemno sponko druge naprave ($2 \rightarrow 3$). Sponka RTS (zahteva za oddajo) ene naprave je povezana s sponko CTS (pripravljen za oddajo) druge naprave ($4 \rightarrow 5$). Torej, če naprava na levi strani postavi zahtevo za oddajo, naprava na desni strani to čuti tako, kot da je 'modem' na levi strani pripravljen za oddajo. Napravi druga drugi dajeta občutek, da je modem na drugi strani pripravljen in da smeta oddajati podatke. Podobno pravilo velja za sponko DTR (računalniška naprava pripravljena), ki je vezana na sponki DCD in DSR ($20 \rightarrow 6, 8$). Če naprava na levi strani postavi signal DTR, naprava na drugi strani 'misli', da je modem na levi strani pripravljen, nosilec signala prisoten (sponka DCD) in podatki, ki prihajajo na sprejemni sponki, so 'veljavni'. Kablu s konektorjema DB25 na obeh straneh in z narisano povezavo sponk pravimo *ničelni modem* (ang. Null Modem). S pojavom osebnih računalnikov je DB25 marsikje nadomestil 9-polni konektor enake oblike DB9. Tudi ta priključek se v pogovoru imenuje RS232 priključek. Razporeditev odgovarjajočih signalov na DB25 in DB9 nahajamo na sliki 51.

3.3.2 RS422, RS423, RS449 in RS485

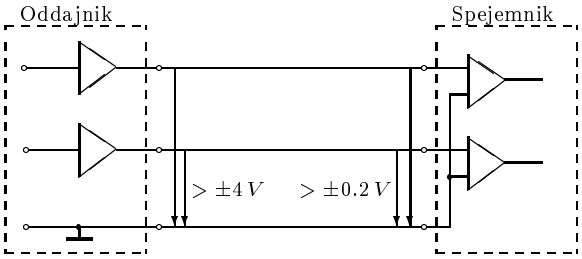
Standarda RS423A in RS422A predvidevata višje hitrosti prenosa in večje razdalje kot RS232. do deset milijonov bitov na sekundo pri razdalji do 40 ft (približno 13 metrov). Oba določata samo električne zahteve tokokrogov. Po standardu RS422A se vsak posamezen signal prenaša dvožično - kot razlika dveh napetostnih nivojev, glej sliko 52. Potencial zemlje ne služi za napetostno referenco. Tako se izognemo problemom zaradi potencialne razlike med napravama, podvojimo pa število povezav.

Ker se informacija prenaša simetrično kot razlika napetostnih nivojev (ang. Balanced Transmission) po parici (ang. Twisted Pair), ki sta enako izpostavljena motilnim vplivom, se zmanjša tudi vpliv motenj. Zato na sprejemni strani zadostuje med nivojema signala ničle in enice že napetostna razlika 400 milivoltov. Če je napetostna razlika med sponkama na sprejemni strani višja od $+200\ mV$, pomeni to za sprejemnik logično ničlo. Če je napetostna razlika nižja od $-200\ mV$ pomeni to logično enico. Naprave RS232C in RS422A med seboj niso združljive.

Naprava izdelana po standardu RS423A je združljiva z napravami tipa



Slika 52: Prenos signalov po standardu RS422A.

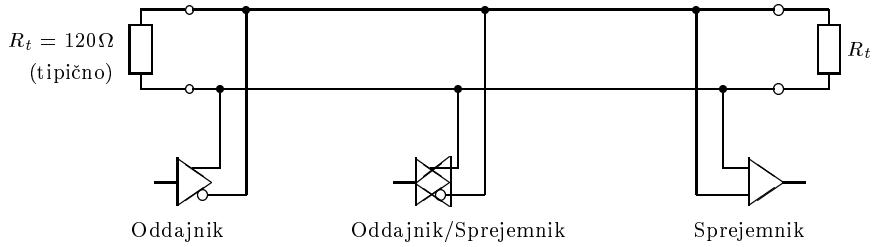


Slika 53: Prenos signalov po standardu RS423A. Narisan je prenos dveh od večjega števila signalov. Vsi signali, ki se prenašajo v isti smeri, imajo skupno povratno pot in ozemljitev samo na eni strani.

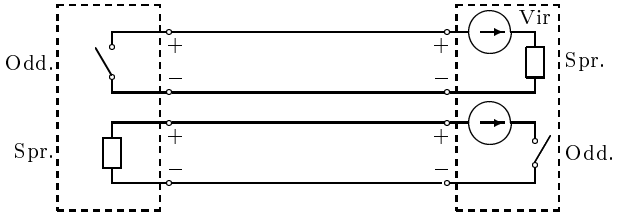
RS232 in tipa RS422A. Standard predpisuje le električne značilnosti povezave dveh naprav. Dovoljuje krajše razdalje in nižje hitrosti prenosa kot RS422A. Slika 53 shematično prikazuje izvedbo povezave dveh signalov po tem standardu. Vsaka smer prenosa uporablja svoj povratni vodnik z vzemlitvijo na oddajni strani.

V pogledu števila povezav je standard RS423A bolj ekonomičen od standarda RS422A. Ker ni vzemljitve tokokrogov na obeh straneh, ni izravnalnega toka, ki bi povzročal potencialne razlike med obema stranema. Potencialna razlika med napravama, ki sta povezani po standardu RS232, pa pogosto predstavlja resen problem. Standard RS423A zahteva, da je na oddajni strani napetostna razlika med nivojema ničle in enice vsaj 8 voltov, kar je dovolj, da tudi sprejmna naprava tipa RS232 pravilno razpozna stanji logične ničle in logične enke. Sprejemnik tipa RS423A pa mora pravilno delovati tudi pri razliki nivojev nad 400 mV, enako kot po standardu RS422A. To zagotavlja združljivost naprave RS423A z obema standardoma.

Standard RS449 naj bi bil naslednik standarda RS232. Predpisuje mehanske in funkcionalne zahteve za izvedbo povezave dveh naprav. Električne zahteve so zajete v standardih RS422A in RS423A. RS449 je kasnejšega datuma (1977) kot RS232C in naj bi izboljšal tehnične lastnosti povezave svojega predhodnika RS232C. Predpisuje kar 46 tokokrogov, razporejenih na dveh priključnikih s 37 in z 9 sponkami. Razlog za to je najbrž dejstvo, da za večino povezav signalov drugega priključnika ne potrebujemo.



Slika 54: Večtočkovno povezovanje in standard RS485.



Slika 55: Povezava naprav s tokovno zanko.

Za nas zanimiv je še standard RS485 iz leta 1983. Ta določa električne karakteristike naprav za večtočkovne povezave, ki jih srečujemo v industrijskih okoljih, glej sliko 54. Na skupen simetričen kanal se sme povezati do 32 naprav, maksimalna dolžina kabla je 1200 metrov, maksimalna hitrost pa 10 Mb/s (ne istočasno). Vsaka naprava je lahko samo oddajnik, samo sprejemnik ali oddajnik in sprejemnik. V nekem trenutku sme seveda oddajati samo eden, sprejemajo pa lahko vsi. Oddajni nivoji signala morajo presegati $\pm 1.5\ V$, sprejemnik mora ločiti logični stanji že pri razliki $\pm 200\ mV$.

3.3.3 Tokovna zanka

Nosilec informacije je tok. Sklenjen tokokrog (tok teče) pomeni logično enico. Razklenjen tokokrog (tok ne teče) pomeni logično ničlo. Uporabljata se izvedbi povezave s tokom 20 mA in s tokom 60 mA. 'Tokovna zanka' niti ni pravi standard. Ohranila se je še iz časov elektromehanskih telegrafskih naprav. Tokovna zanka omogoča razdalje do 1500 ft (500 metrov) pri hitrosti prenosa 9600 b/s.

Obstajata dve vrsti priključka tipa 'tokovna zanka': aktiven in pasiven. Aktiven priključek (ali aktivna tokovna zanka) je generator toka. Pasivna tokovna znaka je porabnik toka. Neposredno smemo povezati samo aktiven priključek s pasivnim priključkom, sicer pa je potreben vmesnik z optosklopnikom. Možni sta obe kombinaciji: aktiven oddajnik in pasiven sprejemnik ali pasiven oddajnik in aktiven sprejemnik. Na sliki 55 je narisana ena od možnosti. Tabela 4 podaja nekaj lastnosti nekaterih izmed obravnavanih standardov.

Tabela 4: Primerjava lasnosti nekaterih standardov.

Oznaka EIA in CCITT	RS232C/D V.24+V.28	RS422A V.10, X.26	RS423A V.11, X.27	RS485 -
Povezava	nesimetrična	nesimetrična	simetrična	simetrična
Število oddajnikov sprejemnikov	 1 1	 1 10	 1 10	 32 32
Nivoji oddajnika sprejemnika	$\pm 5V$ Min $\pm 15V$ Max $\pm 3V$	$\pm 3.6V$ Min $\pm 6.0V$ Max $\pm 0.2V$	$\pm 2V$ Min $\pm 0.2V$	$\pm 1.5V$ Min $\pm 0.2V$
Max. razdalja Max. hitrost	15m 20kb/s	1200m 100kb/s	1200m 10Mb/s	1200m 10Mb/s

Literatura

[1] C. Shannon, *Mathematical Theory of Communications*, Urbana 1948.

[2] L. Gyergyek, *Teorija informacij*, ZAFER 1988.

[3] B. Sklar, *Digital Communications, Fundamentals and Applications*, Prentice-Hall, 1998.

[4] H. Taub, D. Schilling, *Principles of Communication Systems*, McGraw-Hill, 1971.

[5] D. Tugal, O. Tugal, *Data Transmission*, 2nd ed., McGraw-Hill, 1989.

[6] P. Horowitz, W. Hill, *The Art of Electronics*, 2nd.Ed., Cambridge Press 1989.

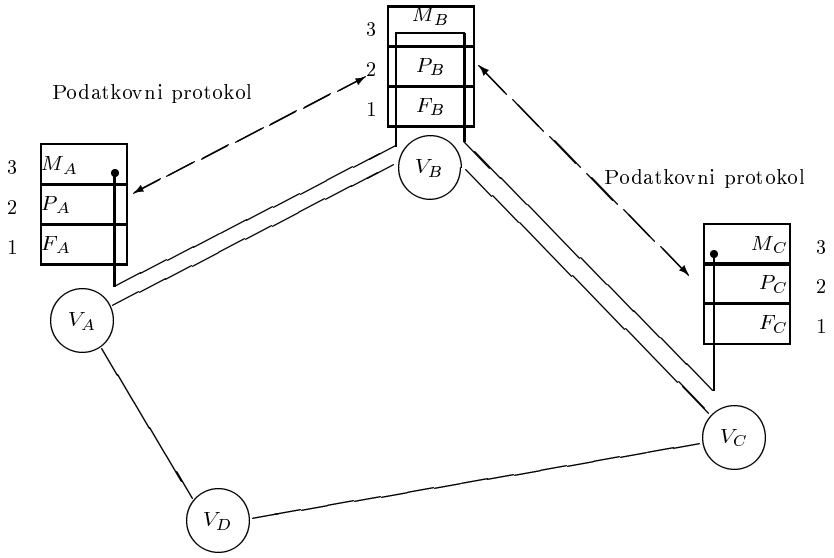
[7] A. Tanenbaum, *Computer Netorks*, Prentice-Hall, 1998.

[8] I. Witten, "Welcome to the Standards Jungle", Byte, Vol. 8, No. 2, Feb. 1983, pp. 146:178.

[9] J. Campbell, *C Programmers Guide to Serial Communications*, Howard W. Sams & Company, 1987.

4 Elementi podatkovnega sloja

Podatkovni sloj nudi storitve neposredno višjemu mrežnemu sloju. Njegova naloga je, da poskrbi za prenos podatkov z visoko stopnjo zanesljivosti od procesa (aktivnosti) mrežnega sloja na eni strani do procesa mrežnega sloja na drugi strani, glej sliko 56. Proces mrežnega sloja M_A vozlišča V_A želi komunicirati s procesom M_C vozlišča V_C. Zato "prosi" podatkovni sloj, ki skrbi za prenos podatkov med V_A in V_B, naj mu omogoči komunikacijo s procesom M_B vozlišča V_B. Podobno proces M_B prosi podatkovni sloj, ki je zadolžen za povezavo od V_B do V_C, naj mu omogoči komunikacijo s procesom M_C. Vmesno vozlišče V_B v tem primeru opravlja funkcije posrednika. Posredništvo se zgodi na tretjem (mrežnem) nivoju. Takemu vozlišču navadno pravimo 'usmerjevalnik' (ang. Router). Če bi se posredništvo zgodilo na podatkovnem sloju brez angažiranja mrežnega sloja, bi vozlišče V_B imenovali most (ang. Bridge).



Slika 56: Podatkovni sloj nadgradi nad nezanesljivim prenosom bitov fizičnega sloja zanesljiv prenos paketov mrežnega sloja. Aktivnosti podatkovnega sloja sosednjih vozlišč se pri tem sporazumevajo po podatkovnem protokolu.

Podatkovni sloj se poslužuje storitev sosednjega nižjega - fizičnega sloja. Ta mu zagotavlja prenos binarnih simbolov (signalov) od vozlišča do vozlišča, ki pa zaradi nepopolnosti naprav in povezav ter zunanjih motilnih vplivov v splošnem ni nikoli popolnoma zanesljiv. Nezanesljiv prenos podatkov je s stališča višjih slojev premalo kvalitetna storitev. Zato podatkovni sloj nadgradi nad nezanesljivim fizičnim slojem zanesljiv prenos podatkov med sosednjimi vozlišči. To doseže z delitvijo zaporedja bitov na okvirje ali okvirjenjem (ang. Framing) in z nedvoumnim označevanjem začetka in konca okvirja, prenosom posameznih okvirjev od vozlišča do vozlišča skupaj z ugotavljanjem in redkeje s popravljanjem napak, ponavljanjem prenosa poškodovanih okvirjev in reševanjem problema podvojenih

in izgubljenih okvirjev (ang. Flow and Error Control). Podatkovni sloj opravlja še druge naloge v zvezi s prenosom podatkov, denimo vzpostavljanje, vzdrževanje in sproščanje zveze. Zelo važna naloga tega sloja posebno v lokalnih omrežjih pa je nadzor nad dostopom do skupnega prenosnega medija (ang. Media Access Control - MAC). Pravila in dogovore, ki jih oddalejene (istorodne) aktivnosti podatkovnega sloja pri tem upoštevajo, imenujemo protokol podatkovnega sloja. Protokol bi torej lahko imenovali tudi krajevno porazdeljen algoritem za prenos podatkov.

4.1 Okvirjenje

Delitev neprekinjenega niza bitov na krajše enote, tako imenovane okvirje, imenujemo okvirjenje. Z okvirjenjem nedvoumno označimo kdaj okvir začne in kdaj konča. Štirje najpogostejši načini okvirjenja so:

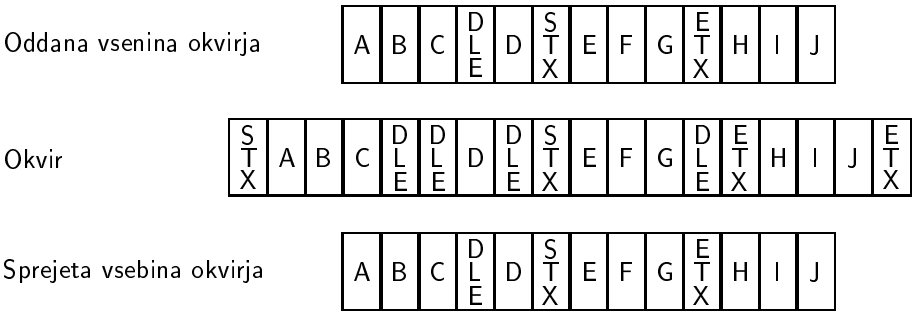
- označevanje začetka in konca okvirja z domenjenimi nadzornimi znaki ter vrivanje napovednega znaka pred podatke, ki so morebiti enaki nadzornim znakom,
- označevanje začetka in konca okvirja z domenjenim bitnim vzorcem ter vrivanje 'polnilnih' bitov pred podatkovne bite v primeru dvoumnosti,
- označevanje začetka in konca okvirja z drugačno obliko signala, kot je predvidena za prenos podatkov,
- zapis dolžine okvirja na začetku okvirja.

Zadnji način predvideva na začetku (v "glavi" ali "zaglavju") okvirja polje, ki vsebuje število podatkov v okvirju. Če je to polje okvirja med prenosom poškodovano, sprejemnik ne more ugotoviti dolžine tekočega okvirja, lahko zgreši tudi začetek naslednjega okvirja in sprejemnik izgubi sinhronizacijo z oddajnikom. Brez dodatnih mehanizmov je praktično nemogoče ponovno vzpostaviti sinhronizacijo sprejemnika z oddajnikom na začetek okvirja. Zato se zapis števila podatkov v okvirju za okvirjenje skoraj nikoli ne uporablja samostojno, ampak skupaj z enim od drugih treh načinov.

Znakovno usmerjen način okvirjenja reši problem sinhronizacije sprejemnika na začetek okvirja z vnaprej dogovorjenim *začetnim znakom*, ki je tipično ASCII[20] kodiran znak z imenom STX (Start of TeXt). Pred oddajo vsebine okvirja oddajna naprava odda začetni znak, ki ga pričakuje sprejemnik. Podobno se konec okvirja označi z domenjenim *končnim znakom* ETX (End of TeXt) ali ETB (End

[20] v IBM EBCDIC kodu ima enako ime

of Block). Ker se ASCII kod uporalja za kodiranje besedila, pri prenosu 'čistega' besedila ni težav, saj se nadzorni znaki kot sta znaka STX in ETX nikoli ne pojavijo med podatki znotraj okvirja. Težave se začnejo pri prenosu "binarnih" podatkov, kot so stanja stikal, vrednosti meritev, i.t.d., kjer je za podatke možna vsaka kombinacija binarnih simbolov. Ena od rešitev je avtomatično vstavljanje napovednega znaka (na primer ASCII DLE - Data Link Escape) na oddajni strani pred vsak podatek, ki je enak nadzornemu znaku. Sprejemna naprava razume napovedni znak DLE kot opozorilo, da bo sledil podatek in ne nadzorni znak, odstrani napovedni znak in ohrani podatek, ki je sicer enak nadzornemu znaku. Primer 'polnjenja' znakov je narisan na sliki 57. Napovedni znak je potreben tudi tedaj, ko se njegova koda DLE pojavlja kot podatek znotraj okvirja.



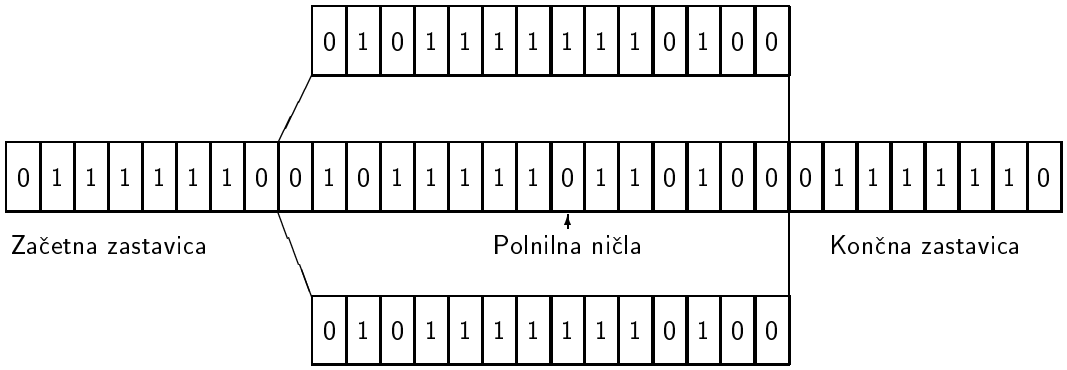
Slika 57: Okvirjenje z začetnim in končnim znakom in vstavljanje napovednega znaka. Zgoraj: prvotno zaporedje podatkov (vsebina okvirja). V sredini: okvir v času oddaje dopolnjen z začetnim, s končnim in z napovednimi znaki pred podatki, ki so enaki nadzornim znakom. Spodaj: 'očiščeni' sprejeti podatki.

To ni edina možnost za rešitev problema dvoumnosti med podatki in nadzornimi znaki. Možen je tudi obraten dogovor, in sicer: napovedni znak naj služi kot opozorilo, da bo sledil nadzorni znak in ne podatek. Okvir začne z DLE in STX ter konča z DLE in ETX, medtem ko se nadornih znakov znotraj okvirja ne napoveduje, razen kadar se kot podatek pojavi DLE.

Označevanje začetka in konca okvirja z domenjenimi nadzornimi znaki nas veže na osembitne podatke in način (ASCII) kodiranja. Uporablja se, kot pravimo, v znakovno usmerjenih podatkovnih protokolih za sinhroni in asinhroni prenos. Znakovne protokole so danes skoraj popolnoma izrinili bitno usmerjeni protokoli, ki niso vezani na način kodiranja.

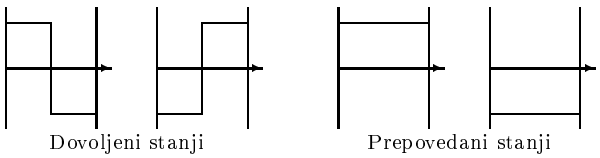
Okvirjenje z začetnim in končnim bitnim vzorcem je neodvisno od načina kodiranja (ASCII, EBCDIC), in dovoljuje poljubno število bitov na podatek. Okvir začne z domenjenim začetnim bitnim vzorcem, ki ji rečemo 'otvoritvena' ali začetna zastavica (ang. Opening Flag). Začetno zastavico sestavlja uvodna ničla, neprekinjeno zaporedje šestih enic in zaključna ničla: 01111110. Končna zastavica (ang. Closing Flag) je kar enaka začetni zastavici. Ko sprejemnik sprejme zaporedje šetnih ničel to razume kot začetek novega (ali konec tekočega) okvirja.

Zato se znotraj okvirja tako zaporedje binarnih simbolov ne sme nikoli pojaviti. Kadarkoli v nizu podatkovnih bitov oddajnik odkrije pet ali več zaporednih enic, za peto enico avtomatično vstavi eno 'polnilno' ničlo. Šest zaporednih enic, ki bi jih sprejemnik lahko zamenjal z začetnim vzorcem, se med podatki tako ne more pojaviti. Ko sprejemnik sprejme pet zaporednih enic ve, da mu sledi polnilna ničla, 'vsrka' ničlo in niz podatkovnih bitov dobi prvotni pomen. Izgled okvirja in primer vstavljanja ničle (ang. Bit Stuffing) prikazuje slika 58.



Slika 58: Okvirjenje z začetnim in končnim vzorcem in polnjenjem ničle, če se v nizu podatkovnih bitov pojavi zaporedje enic, ki je daljše od petih bitov. Zgoraj: prvotno zaporedje bitov. V sredini: oddani okvir s primerom polnilne ničle. Spodaj: 'očiščeno' sprejeto zaporedje bitov.

Zadnja možnost okvirjenja je označevanje začetka in konca okvirja s takim vzorcem informacijskega signala, ki se znotraj okvirja ne sme pojaviti, je prepovedan. Na primer, z informacijskim signalom, ki zmore štiri stanja, lahko eno stanje pomeni logično nič, eno stanje pomeni logično ena, ostali dve stanji pa se izkoristita za označevanje začetka in konca okvirja, glej sliko 59. Znotraj bitne celice sta dovoljeni dve stanji, prehod signala z visokega na nizek nivo in prehod signala z nizkega na visok nivo. Stanji signala cel čas nizek nivo ali cel čas visok nivo sta za kodiranje podatkov prepovedani, lahko pa ju izkoristimo za kodiranje začetka in konca okvirja.



Slika 59: Primer signala s štirimi možnimi stanji, prvi dve sta predvideni za kodiranje podatkov (0 in 1). Drugi dve stanji sta za kodiranje podatkov prepovedani, služita pa za označevanje začetka in konca okvirja.

4.2 Nadzor nad napakami in nad pretokom podatkov

V tem podpoglavju se bomo posvetili postopkom za prenos podatkov, ki zagotovljajo, da je tok podatkov med oddajno in sprejemno napravo v vseh okoliščinah čimbolj zanesljiv in kar se da tekoč. S temi postopki rešujemo probleme, ki so posledica nepopolnosti naprav in povezav, končne hitrosti delovanja naprav oziroma razlike v hitrosti med oddajanjem in sprejemanjem, omejene velikosti sprejemnih in oddajnih medpomnilnikov, kasnitve med sprejetim in oddanim signalom in podobno. To vedno vključuje odkrivanje in včasih popravljanje napak.

V glavnem se postopki med seboj razlikujejo glede na stopnjo zanesljivosti, v pogledu prepustnosti, po izkoriščenosti kanala, po kasnitvah med oddajo in sprejemom, in v zahtevnosti izvedbe. Nekateri postopki (protokoli) zagotavljajo boljšo prepustnost na račun daljšega čakalnega časa, drugi spet skrajšajo čakalni čas na račun izkoriščenosti kanala. Vsi pa vnašajo redundanco.

Kadar govorimo o prepustnosti (ang. Throughput), nas ne zanima trenutna hitrost prenašanja podatkov, ampak se zanimamo za efektivno hitrost prenašanja podatkov gledano čez daljše časovno obdobje. *Prepustnost* je po definiciji enaka povprečni vrednosti pretoka koristnih podatkov. Kaj so 'koristni' podatki ni enoznačno določeno. Včasih je to cel okvir (recimo glava in podatki) včasih pa je to samo podatkovni del okvirja.

Izkoriščenost (ang. Efficiency) je razmerje med prepustnostjo in maksimalno možno hitrostjo prenosa (to je hitrostjo oddajanja, ki se običajno enači s kapaciteto kanala). Običajno se izkoriščenost računa kot razmerje časa, ko se prenašajo koristni podatki in časa, ki je v povprečju potreben za prenos teh podatkov.

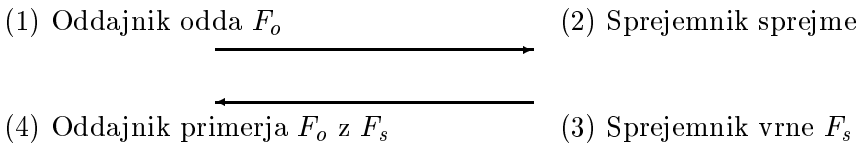
Odzivni čas ali reakcijski čas (ang. Response Time) je tisti čas, ki preteče od zahteve za prenos (za oddajo) do tedaj, ko so podatki (okvir) prenešeni. Je seštevek časa čakanja, da pride okvir na vrsto, časa oddajanja in sprejemanja (procesiranja) in časa širjenja signala od oddajnika do sprejemnika. Hiter odziv je v sistemih stvarnega časa pomembnejši od visoke prepustnosti.

Uporabljajo se trije osnovni koncepti nadzora nad tokom podatkov med oddajno in sprejemno napravo:

- preverjanje odmeva (ang. Echo Checking),
- vnaprejšnje popravljanje napak (ang. Forward Error Correction - FEC) in
- (avtomatska) zahteva za ponovitev prenosa okvirja (ang. Automatic Repeat Request - ARQ).

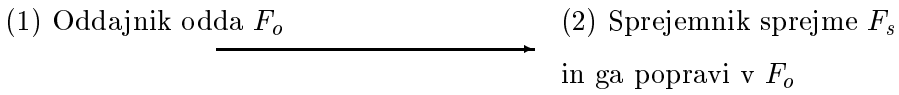
Preverjanje odmeva je najenostavnejši način za odkrivanje napak med

prenosom. Oddajna naprava odda podatek ali zaporedje podatkov in nato čaka na odgovor sprejemne naprave. Sprejemna naprava sprejme podatke in jih vrne oddajniku v nespremenjeni obliki. Oddajna naprava primerja oddane podatke s sprejetimi in če se ne razlikujejo, to pomeni, da je sprejemik podatke pravilno sprejel. Preverjanje pravilnosti prenosa opravlja oddajnik. Takšen način zahteva dupleksni kanal in zasede pol kapacitete kanala za preverjanje pravilnosti prenosa (polovico za prenos podatkov/ informacije v eni smeri in drugo polovico za prenos v drugi smeri).



Slika 60: Preverjanje odmeva. Preverjanje opravlja oddajnik.

Vnaprejšnje popravljanje napak (FEC) se pojavlja skupaj z enim od kodov za popravljanje napak (na primer s Hammingovim kodom, RS kodom, BCH kodom, Golay-evim kodom, ...). Oddajna naprava doda h koristni informaciji dovolj odvečne informacije, da lahko sprejemna naprava odkrije (ali vsaj z zelo veliko verjetnostjo) napako in jo tudi popravi. Slabost vnaprejšnjega popravljanja napak je nizka izkoriščenost kanala zaradi velike odvečnosti v prenašanih podatkih. To je glavni vzrok, da se razmeroma malo uporablja. Prednost tega postopka se pokaže pri simpleksnih prenosnih poteh, pri prenosu na velike razdalje, kjer postane zakasnitev sprejemnika glede na oddajnik odločilnega pomena, in pri prenosu po zelo motenih prenosnih poteh.

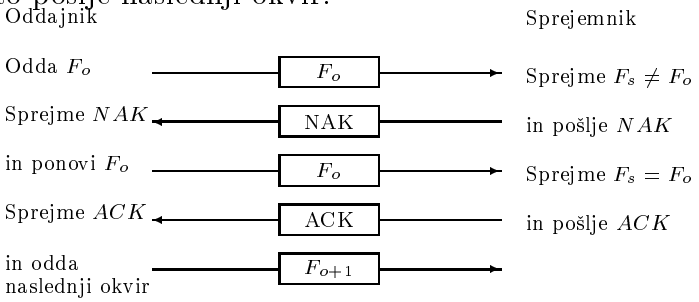


Slika 61: Vnaprejšnje popravljanje napak. Popravljanje opravlja sprejemnik.

Avtomatska *zahteva za ponovitev* (ARQ) se v podatkovnih omrežjih največ uporablja. Eden od vzrokov za to je najbrž tudi v dejstvu, da so sodobne prenosne poti razmeroma zanesljive (napake se redko pojavijo), in da se informacija običajno prenaša v obliki blokov (okvirjev ali paketov). Zahteva za ponoven prenos napačno prenešenega okvirja se uporablja skupaj s kodirnimi postopki za odkrivanje napak (tipično s preverjenjem parnosti ali s cikličnim preverjanjem). Ti vnašajo znatno manj redundance kot postopki za popravljanje napak.

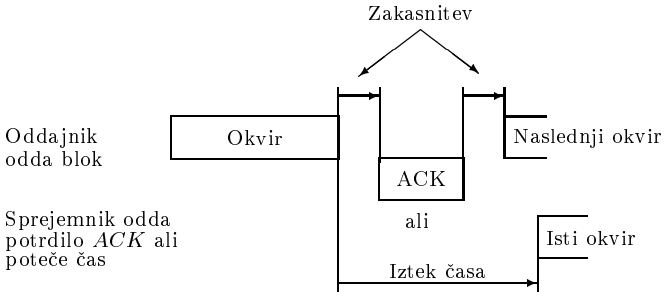
Postopek ARQ vključuje preverjanje okvirjev s strani sprejemne naprave. Zato mora oddajna naprava koristni informaciji pred oddajo v kanal dodati po nekem postopku dovolj odvečne informacije, da je sprejemna naprava zmožna

odkriti napako, ki se mogoče pojavi med prenosom. V primeru, da napako odkrije ($F_s \neq F_o$), zavrže sprejeti okvir ter po povratnem kanalu z negativno potrditvijo, ki se tipično označuje z NAK (ang. Negative Acknowledge), zahteva od oddajne naprave ponovno oddajo napačno prenešenega okvirja. V primeru, da je okvir prenešen brez napake ($F_s = F_o$) ali sprejemnik napake ne odkrije, pošje oddajniku pozitivno potrdilo ACK (ang. Positive Acknowledge) in s tem potrdi pravilen sprejem. Oddajnik nato pošlje naslednji okvir.



Slika 62: Avtomatska zahteva za ponovitev. Preverjanje opravlja sprejemnik.

Pri avtomatski zahtevi za ponovitev se lahko potrjuje tudi samo pravilno sprejete okvirje, nepravilen prenos pa pomeni odsotnost potrdila. Razmere prikazuje slika 63. Oddajnik pošlje okvir in nato čaka na potrdilo. V ta namen nastavi časovni števec in če se ta 'izteče' (ang. Time-Out) predno prispe pozitivno potrdilo od sprejemnika, sledi ponovitev okvirja. Čakalni interval mora biti primerno izbran. Predolgo čakanje po nepotrebnem zmanjša prepustnost. Vendar mora biti čakalni interval vsaj nekoliko daljši od časa za prenos potrdila. Pozitivno in negativno potrjevanje (ACK in NAK) zato običajno zagotavlja boljšo izrabo prenosne poti.

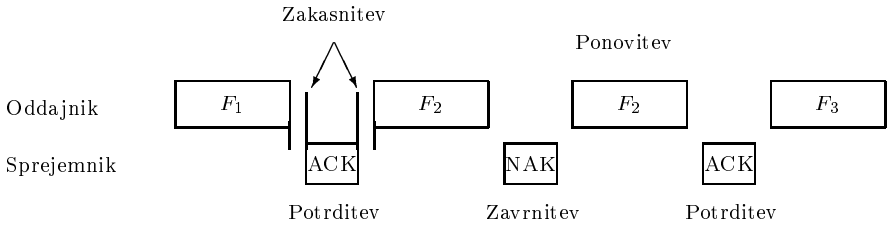


Slika 63: Potrjevanje s čakanjem in z iztekom časa.

Obstajata dve možnosti za realizacijo avtomatske zahteve za ponovitev: potrjevanje s čakanjem (ang. Stop-And-Wait ali tudi Send-And-Wait) in potrjevanje brez čakanja.

4.2.1 Potrjevanje s čakanjem

Pri *potrjevanju s čakanjem* oddajna naprava po oddaji tekočega ovirja in pred oddajo naslednjega vedno počaka na sprejem potrdila, glej sliko 64. Šele nato bodisi odda naslednji okvir bodisi ponovi tekoči okvir. Zaradi čakanja na potrditev in zaradi prenosa potrdila samega pride med dvema zaporednima okvirjema do časovnega presledka (premora). V tem času se ne prenašajo koristni podatki in kanal je neizkoriščen. Premor je še posebno dolg pri polduplekšnih zvezah, ker se za preklop z oddaje na sprejem porabi še nekaj več časa. Posledica pa je slaba izkoriščenost prenosne poti.



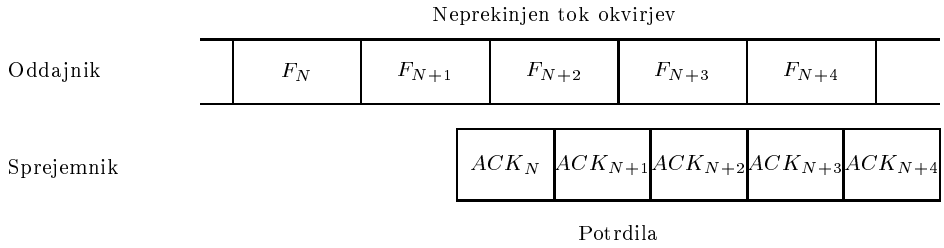
Slika 64: Potrjevanje s čakanjem.

Kaj se zgodi, če se izgubi (ali pokvari) potrdilo sicer pravilno sprejetega okvirja? Oddajna naprava v takem primeru napačno sklepa, da je bil sprejem tekočega okvirja neuspešen in še enkrat pošlje isti okvir. Posledica tega je dvakratno pošiljanje in sprejem istega okvirja. Oziroma, oddajnik ponovi že pravilno sprejeti okvir in sprejemnik sprejme (ohrani in preda naprej) isti okvir dvakrat. Problem podvojenih okvirjev rešimo s številčenjem okvirjev. Vsak okvir je označen z zaporedno številko. Če sprejemnik sprejme dva okvirja z enako številko, enega od obeh enostavno zavrže.

Številčenje okvirjev je nujni sestavni del vsakega podatkovnega protokola. Številčenje okvirjev uporablja tudi protokol XMODEM za prenos podatkov med majhnimi računalniki, ki je bil podlaga za nastanek bolj izpopolnjenih protokolov (YMODEM, ZMODEM). Pri potrjevanju s čakanjem je vedno nepotrjen samo eden (zadnji) okvir, ki ga je potrebno razlikovati od predhodnjega okvirja. Torej je dovolj, da izmenoma označujemo (številčimo) okvirje enkrat z nič in drugič z ena. Eden najbolj znanih protokolov potrjevanja s čakanjem uporablja nič/ena številčenje in je po njem dobil tudi ime ABP protokol (ang Alternating-Bit-Protocol). Sami se prepričajte, da je potrebno ne le številčenje okvirjev, ampak tudi številčenje potrdil.

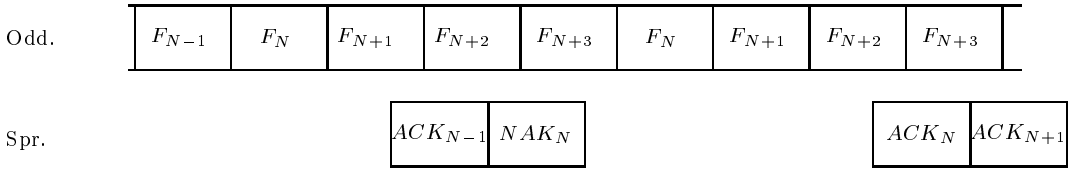
4.2.2 Potrjevanje brez čakanja

Pri *potrjevanju brez čakanja* je tok okvirjev od oddajnika k sprejemniku neprekinjen. Okvirji si sledijo drug za drugim, kot je narisano na sliki 65. Prenosna pot je zato bolje izkoriščena. Ker oddajnik s potrjevanjem okvirjev 'zamuja', je poleg številčenja okvirjev nujno potrebno tudi številčenje potrdil. Iz številke potrdila sprejemnik ve, na kateri okvir se nanaša potrdilo. Takšen način potrjevanja ima prednost pred potrjevanjem s čakanjem posebno na kvalitetnih prenosnih poteh s sicer relativno veliko zakasnitvijo sprejemnika proti oddajniku. Zahteva pa na oddajni strani začasno shranjevanje večjega števila oddanih, a še ne potrjenih okvirjev in s tem večji oddajni medpomnilnik.



Slika 65: Princip potrjevanja okvirjev brez čakanja.

Denimo, da pride med prenosom do napake okvirja z zaporedno številko N. Na voljo sta dve možnosti: vračanje nazaj na okvir s številko N ali s kratico GBN (ang. Go-Back-N) in selektivna ponovitev okvirja ali s kratico SRQ (ang. Selective-Repeat-Request), imenovana tudi SRP (ang. Selective Repeat Protocol). Pri *vračanju nazaj na N* pošlje sprejemnik, ko ugotovi napako okvirja s številko N, oddajniku negativno potrdilo (zavrnitev) s številko N (NAK_N), ali pa negativnega potrdila enostavno ne pošlje in pride do izteka časa. Sprejemnik zavrže N-ti okvir, prezre pa tudi vse kasnejše okvirje, ki jih je že oddal oddajnik (slika 66).

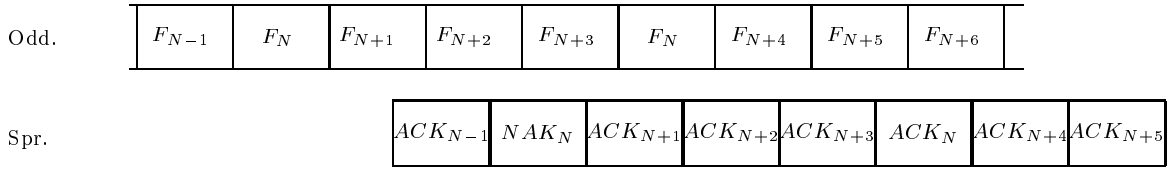


Slika 66: Ponavljanje z vračanjem nazaj na N (GBN).

Oddajnik razume zavrnitev NAK_N kot zahtevo, da se mora oddajanje vrniti nazaj na N-ti okvir. Zato odda najprej pokvarjeni okvir s številko N, za njim pa še vse naslednje okvirje ($N+1, N+2$, i.t.d.) tudi če so bili prenešeni brez napake. Posledica takega načina ponavljanja je razmeroma slaba izkoriščenost manj kvalitetnih prenosnih poti, ker je veliko tudi nepotrebnih ponovitev. Dobra

lastnost GBN pa je, da na sprejemni strani (na podatkovnem sloju) ne zahteva medpomnilnika večjega od velikosti enega okvirja.

Pri *selektivnem ponavljanju* v primeru napake N-tega okvirja oddajna naprava na zahtevo sprejemne naprave ponovi samo pokvarjeni (N-ti) okvir, kasnejših okvirjev $N+1, N+2$, i.t.d. pa ne. Posledica takega načina ponavljanja je, da imajo pravilno sprejeti okvirji drugačen vrstni red kot oddani okvirji. Z drugimi besedami, možno je, da sprejemnik prej dobi okvir, ki je bil oddan kasneje. Možne razmere so narisane na sliki 67. Da je na sprejemni strani možno obnoviti prvoten vrstni red okvirjev, je potrebno preurejanje okvirjev. To zahteva začasno pomnenje sprejetih okvirjev in kajpak razmeroma velik vmesni pomnilnik sprejemne naprave. S selektivnim ponavljanjem blokov torej bolje izkoristimo kanal (ni nepotrebnega ponavljanja), potrebujemo pa dodaten vmesni pomnilnik sprejemnika. Ta pomnilnik mora biti dovolj velik, da hrani vse sprejete in še ne potrjene okvirje. Če je možno veliko število nepotrjenih okvirjev, moramo predvideti velik medpomnilnik. Ker je težko oceniti zadostno velikost medpomnilnika, se ta način razmeroma manj uporablja kot GBN protokol.



Slika 67: Selektivno ponavljanje napačno sprejetega okvirja.

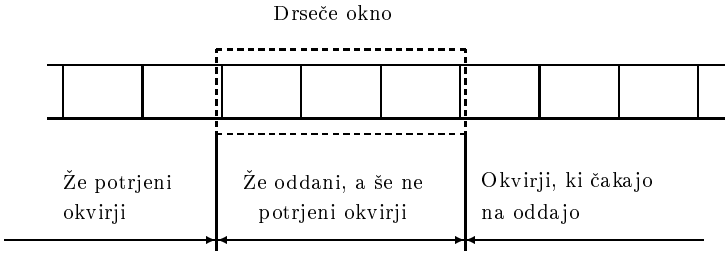
Na osnovi povedanega lahko povzamemo naslednje. V splošnem se pri prenosu podatkovnih okvirjev javljajo tile problemi:

- sprejemnik mora biti sposoben vzdrževati pravilno zaporedje podatkovnih okvirjev,
- vsi že oddani, vendar še ne potrjeni okvirji morajo biti na oddajni strani začasno shranjeni zaradi morebitne potrebe po ponovitvi,
- sprejemnik ima omejeno hitrost sprejemanja in omejeno velikost pomnilnika. Torej je možno, da bi oddajnik oddajal hitreje, kot je sprejemnik sposoben sprejemati.

Iz tretje ugotovitve neposredno sledi, da bo v nekaterih primerih oddajnik moral za določene čas počakati z oddajo. V predhodnih razdelkim pa smo tudi že ugotovili, da je za vzdrževanje pravilnega zaporedja podatkovnih okvirjev potrebno številčenje okvirjev in v večini primerov tudi številčenje potrdil. V primeru, da

prenašamo daljša zaporedja okvirjev lahko postanejo zaporedne številke prevelike. Za zapis velikih števil pa moramo predvideti več bitov. Vendar enoznačno številčenje okvirjev od začetka do konca sporočila sploh ni potrebno. Dejansko mora biti enoznačeno označenih (številčenih) samo toliko okvirjev, kolikor jih sme biti v določenem času že oddanih in še ne potrjenih. Število takih okvirjev imenujemo velikost okna. Mislimo si lahko, da po časovnem zaporedju okvirjev drsi okno velikosti W, okvirje pa potem številčimo z $0, 1, \ldots, N-1 = W$ in nato spet od 0 naprej (glej sliko 68). Torej po modulu N. Če število nepotrjenih okvirjev doseže N, mora oddajnik z oddajo počakati, dokler sprejemnik vsaj nekaj okvirjev ne potrdi. Takšen pristop k nadzoru nad tokom podatkov je znan pod izrazom *drseče okno* (ang. Sliding-Window-Protocol) in se uporablja skupaj s selektivnim ponavljanjem ali z vračanjem nazaj na N. Zadnja rešitev ima za posledico naslednji pridobitvi:

- omejimo velikost zaporedne številke (recimo na 8) in s tem število bitov za zapis številke (v tem primeru 3),
- omejiti smemo velikost sprejemnega in oddajnega pomnilnika.



Slika 68: Koncept drsečega okna.

Poglavje zaključimo z razvrstitvijo tehnik prenašanja, ki spadajo v skupino avtomatske zahteve za ponovitev – ARQ. Razlikujejo se glede na način potrjevanja, in sicer:

- samo s pozitivim potrdilom (ACK) ali
- s pozitivnim in z negativnim potrdilom (ACK/NAK).

Tako prva kot druga oblika je možna pri potrjevanju

- s čakanjem (ABP) ali
- brez čakanja (GBN ali SRP) z drsečim oknom.

4.3 Vrednotenje podatkovnih protokolov

4.3.1 Potrjevanje s čakanjem

Poglejmo, kako potrjevanje s čakanjem vpliva na izkoriščenost in s tem prepustnost kanala. Kadarkoli po kanalu ne prenašamo ničesar ali po kanalu prenašamo kaj drugega kot koristne podatke (nadzorne podatke, kot so številka okvirja, biti za preverjanje pravilnosti prenosa, potrdilo ali kaj drugega), je kanal neizkoriščen. Čim več je takšnih premorov ali nadzornih podatkov, tem manj je kanal izkoriščen, po drugi strani pa je nižja tudi prepustnost kanala. Proučujmo različico potrjevanja s čakanjem in z iztekom časa v primeru napake pri prenosu okvirja, glej sliko 69. Naj bo za naš izračun kapaciteta kanala C enaka 9600 bitov na sekundo, za dolžino okvirja F vzemimo 512 bitov, za dolžino potrdila A izberimo 8 bitov, kasnilni čas T_z naj bo 0.2 milisekund. Verjetnost za napako okvirja p in vrednost za iztek časa T_o bomo določili kasneje. Nadalje predpostavimo, da so vsi okvirji enako dolgi in da vsak okvir predvideva $D = 496$ bitov za koristne podatke in $G = 16$ bitov za potrebe samega protokola, recimo za preverjanje parnosti, $F = D + G$.

Izkoriščenost E definiramo z razmerjem med časom T_D, ko kanal prenaša koristne podatke in celotnim časom, ki je potreben za prenos teh podatkov T_S,

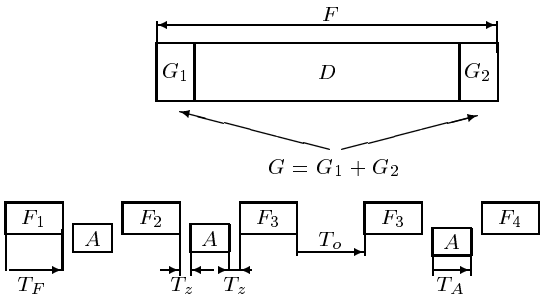
$$E = \frac{T_D}{T_S}.$$

Če napak ne bi bilo, bi bil čas T_S enak seštevku časov trajanja okvirja T_F, trajanja potrdila T_A in trajanja dveh kasnitev T_z, glej sliko 69: $T_S = T_F + T_z + T_A + T_z$. Predpostavljamo, da prenašamo podatke z največjo možno hitrostjo, ki jo dovoljuje kanal. Potem imamo:

$$E = \frac{D/C}{F/C + A/C + 2T_z} = \frac{D}{(D+G) + A + 2 \times T_z \times C}.$$

Za izbrane vrednosti je izkoriščenost kanala $E = \frac{496}{512+8+2\times 0.0002\times 9600} \approx 0.94685$. Izkoriščenost kanala ne bi bila 100% niti tedaj, ko ne bi bilo napak.

V primeru, da se zgodi napaka na okvirju (ali potrdilu), je kanal neizkoriščen čas $T_1 = T_F + T_o$. To drži, če v naslednjem poskusu prenos okvirja uspe. V nasprotnem primeru, ko se ta isti okvir pokvari še drugič, izgubimo čas $T_2 = 2 \times (T_F + T_F)$. Naj bo verjetnost neuspešnega prenosa okvirja (verjetnost napake med prenosom) enaka $p = 10^{-3}$. Z drugimi besedami, od 1000 uspešnih prenosov je eden prenos neuspešen. Verjetnost, da prenašamo isti okvir dvakrat, je enaka produktu $p \times (1-p)$. Torej produktu verjetnosti, da je prvi prenos neuspešen (p) in da je drugi prenos, ki pa je hkrati tudi zadnji, uspešen ($1-p$). Verjetnost, da prenašamo isti okvir k-krat pa je enaka verjetnosti, da po $(k-1)$ ponesrečenih



Slika 69: Izgled okvirja (zgoraj) in primer komunikacije (spodaj).

poskusih prenos končno uspe. Ta verjetnost je $p^{k-1} \times (1-p)$. Povprečno število prenosov na okvir je:

$$\overline{k} = \sum_{k=1}^{\infty} k \times p^{k-1} \times (1-p) = (1-p) \sum_{k=1}^{\infty} k \times p^{k-1} = (1-p)\frac{1}{(1-p)^2} = \frac{1}{1-p}.$$

Povprečno število ponesrečenih prenosov je za ena manjše, $\overline{k} - 1 = \frac{1}{1-p} - 1 = \frac{p}{1-p}$, povprečno trajanje ponavljanja (v bitih) pa je $\frac{p}{1-p} \times (F + T_o \times C)$. Dobljeni rezultat upoštevamo v imenovalcu izraza za izkoristek,

$$E = \frac{D}{(F + A + 2 \times T_z \times C) + \frac{p}{1-p}(F + T_o \times C)}.$$

Ko ni napak, je $p = 0$ in ta izraz je enak prvotnemu izrazu za E. Za iztek časa T_o je smiselno izbrati malenkost daljši čas od časa, ki je skupaj s kasnitvijo potreben za potrdilo, $T_o \times C \approx A + 2 \times T_z \times C$. Izraz za izkoristek kanala dopolnimo z zadnjo ugotovitvijo,

$$E = \frac{D \times (1-p)}{p \times (F + T_o C) + (1-p) \times (F + T_o C)} = \frac{D \times (1-p)}{F + T_o C} = (1-p) \times \frac{D}{D+G} \times \frac{1}{1 + \frac{T_o C}{D+G}}.$$

Prvi člen prispevajo napake in linearno zmanjšuje izkoristek E. Drugi člen je posledica odvečnosti znotraj okvirja. Na G običajno ne moremo kaj dosti vplivati, lahko pa večamo dolžino podatkovnega dela okvirja D (in s tem F) ter tako večamo iskoristek. V zadnjem členu se kaže odvisnost izkoriščenosti kanala od potrjevanja s čakanjem. Za naš primer je:

$$E = (1 - 10^{-3}) \times \frac{496}{512} \times \frac{1}{1 + \frac{11.84}{512}} = 0.999 \times 0.96875 \times 0.9774 = 0.9459.$$

Vidimo, da morebitno ponavljanje okvirjev zaradi prisotnosti napak pri verjetnosti neuspelega prenosa $p = 10^{-3}$ zanemarljivo malo vpliva na zmanjšanje izkoriščenosti.

Sedaj pa poskusimo pri danih lastnostih kanala (pri dani verjetnosti napake bita p_b in pri dani kapaciteti kanala C) določiti optimalno dolžino okvirja, torej

tako dolžino, ki zagotavlja najboljši izkoristek. Gotovo je pri dani verjetnosti napake na bitu verjetnost napake na okvirju podatkov tem večja, čim daljši je okvir. Po drugi strani pa moramo zaradi napak okvirje večkrat ponavljati. Poiščimo najprej zvezo med verjetnostjo napake na okvirju in verjetnostjo napake na bitu. Da je prenos okvirja pravilen, morajo biti pravilno prenešeni prav vsi biti. Verjetnost, da se to zgodi, je $p_u = (1 - p_b)^F$. Ker lahko napaka nastane tudi na potrdilu, je verjetnost uspešnega prenosa enega okvirja enaka

$$(1 - p_b)^F \times (1 - p_b)^A = (1 - p_b)^{(F+A)}.$$

Recimo za naš primer, ko smo privzeli za verjetnost neuspelega prenosa okvirja $p = 10^{-3}$, je verjetnost napake bita

$$1 - 10^{-3} = (1 - p_b)^{520} \rightarrow p_b \approx 1.9 \times 10^{-6},$$

kar je tipična vrednost za verjetnost napake bita na najetem telefonskem vodu. Izkoriščenost kanala v odvisnosti od verjetnosti napake bita in dolžine podatkovnega dela okvirja je:

$$\begin{aligned} E &= (1 - p_b)^{(G+A)} \times (1 - p_b)^D \times \frac{D}{D + G + T_o C} \\ &= (1 - p_b)^{(G+A)} \times (1 - p_b)^D \times \frac{1}{1 + G + T_o C D^{-1}} \\ &= K_1 \times (1 - p_b)^D \times \frac{1}{1 + K_2 D^{-1}} \end{aligned}$$

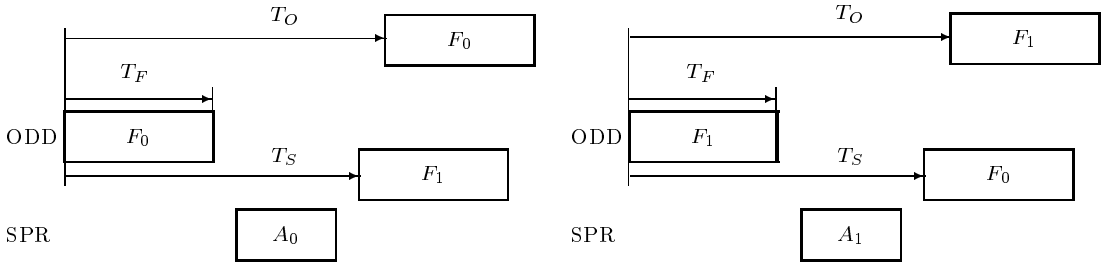
Ta izraz odvajamo po D in izenačimo z nič, $\frac{\partial E}{\partial D} = 0$ ter izračunamo optimalno dolžino podatkovnega dela okvirja v odvisnosti od verjetnosti napake bita. Sami se lahko prepričate, da je

$$D_{opt} \approx \frac{1}{\sqrt{p_b}} \times \sqrt{G + T_o C}.$$

To pomeni, da optimalna dolžina z naraščanjem verjetnosti napake bita hitro upada. Za naš primer je $D_{opt} \approx \sqrt{\frac{16+11.84}{210^{-6}}} \approx 3730$ in optimalna izkoriščenost je $E_{opt} = (1 - 0.0000019)^{3730+16+8} \times \frac{3730}{3730+16+11.84} \approx 0.985$.

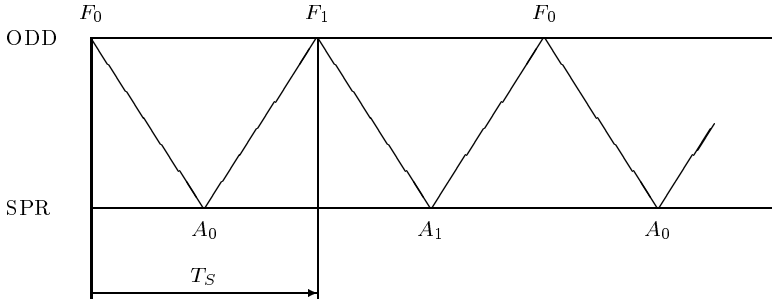
4.3.2 Izkoristek protokola ABP

Izračunajmo izkoristek protokola ABP (Alternating Bit Protocol) s pozitivnim potrjevanjem (pozitivnim odgovorom), ki uporablja nič/ena številčenje okvirjev in potrdil. Delovanje protokola je skicirano na sliki 70.



Slika 70: Protokol ABP (nič/ena številčenje s pozitivnim potrjevanjem).

Obravnava protokola bo podobna predhodni, vendar nas bo zdaj zanimalo samo to, koliko na izkoriščenost kanala vpliva protokol (pozitivno potjevanje s čakanjem). Odvečnosti znotraj okvirja ne bomo upoštevali, ker je ne pripisujemo samo obravnavanemu protokolu (tudi drugi protokoli vnašajo redundanco znotraj okvirja). Model, na katerem bomo zasnovali izračun, prikazuje slika 71.

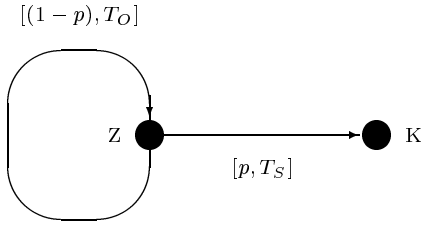


Slika 71: Model za analizo protokola ABP.

Slika prikazuje dogajanje med oddajnikom in sprejemnikom. "Cik-cak" črta ponazarja potovanje okvirjev in potrdil med oddajnikom in sprejemnikom. S T_S je označen čas, ki preteče od oddaje prvega bita okvirja do sprejema zadnjega bita potrdila, skupaj z vmesnimi kasnitvami, ki jih vnašajo kanal, oddajnik in sprejemnik. Po času T_S je možna oddaja naslednjega okvirja. Oddajnik nastavi časovnik na začetku oddaje prvega bita okvirja. Po času T_O se časovnik izteče. Če v tem času oddajnik ne dobi potrdila, ponovi isti okvir. Izkoriščenost definira naslednji obrazec:

$$E_{ABP}(p) = \frac{T_F}{\bar{T}}.$$

Pri tem je T_F čas trajanja okvirja, $\bar{T}$ je čas, ki je v povprečju potreben za prenos enega okvirja in p je verjetnost napake na okvirju (ali potrdilu). Izkoristek ne bomo računali po dolgem postopku, ki smo se ga poslužili v prejšnjem razdelku, ampak bomo skušali računati s čim manj truda. Pri tem nam bo pomagala skica 72.



Slika 72: Verjetnostni model za določitev časa $\bar{T}$, ki je v povprečju potreben za prenos enega okvirja.

Prenos okvirja od oddajnika do sprejemnika smo ponazorili kot problem prehoda iz začetnega stanja Z v končno stanje K. V začetnem stanju smo od trenutka, ko začnemo z oddajanjem okvirja, dokler nam poskus prenosa tega okvirja ne uspe. Z verjetnostjo $(1-p)$ nam prenos uspe in s to verjetnostjo se selimo v končno stanje. To traja čas T_S. Ta zadnji prehod prispeva k povprečnemu času $(1-p) \cdot T_S$. Verjetnost neuspešnega poskusa prenosa je p. Pri vsakem neuspelem poskusu izgubimo čas T_O. Vsak neuspel poskus nam povprečni čas poveča za T_O. To se zgodi z verjetnostjo p. Povprečen čas, da pridemo iz začetnega v končno stanje je:

$$\bar{T} = p \cdot (T_O + \bar{T}) + (1-p) \cdot T_S$$

in ko izrazimo $\bar{T}$, dobimo:

$$\bar{T} = T_S + \frac{p}{1-p} T_O.$$

Izkoristek v odvisnosti od verjetnosti napake je:

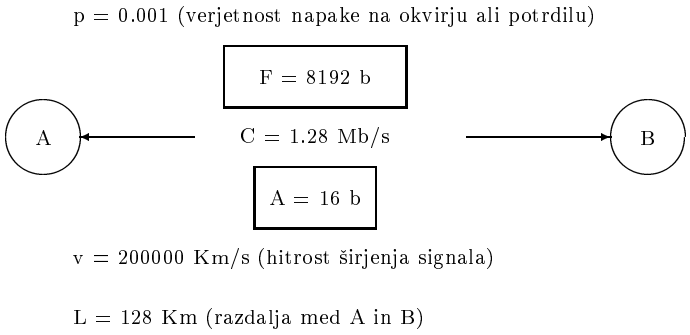
$$E_{ABP}(p) = \frac{T_F}{T_S + \frac{p}{1-p} T_O}.$$

Ker je v praksi $T_O \approx T_S$, je $\bar{T} = \frac{1}{1-p} T_S$ in

$$E_{ABP}(p) = (1-p)\frac{T_F}{T_S},$$

ali če upoštevamo posamezne prispevke k času T_S:

$$E_{ABP}(p) = (1-p)\frac{F}{F + A + 2 \cdot T_z \cdot C}.$$



Slika 73: Podatki za izračun izkoristka protokola ABP.

Izkoristek pada z verjetnostjo napake, s hitrostjo prenosa (oziroma s kapaciteto kanala C) in s kasnilnim časom, ki je odvisen tudi od razdalje med oddajnikom in sprejemnikom, kot smo enkrat že ugotovili.

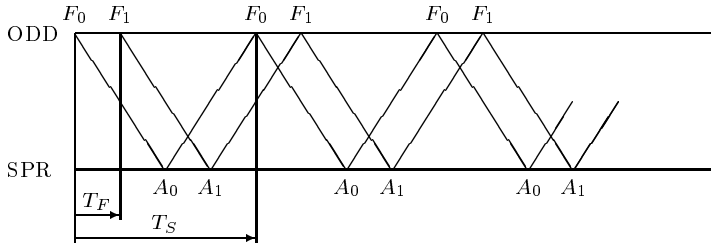
Za primer vzemimo naslednje podatke, glej sliko 73.

Izkoristek je:

$$E_{ABP} = 0.999 \frac{8192}{8192 + 16 + 2 \cdot 64 \cdot 10^{-5} \cdot 1.28 \cdot 10^{6}} = 0.83$$

4.3.3 Analiza protokola GBN

Analizirajmo izkoristek protokola z vračanjem nazaj na N (GBN) in z drsečim oknom. Po tem protokolu oddaja oddajnik brez premorov, če le dobi pravočasno potrdilo za oddani okvir. Razmere, ko ni napak, prikazuje slika 74. Slika prikazuje dogajanja za okno velikosti $w = 2$. Oddajnik odda dva okvirja F_0 in F_1, potem pa mora počakati na potrdilo.



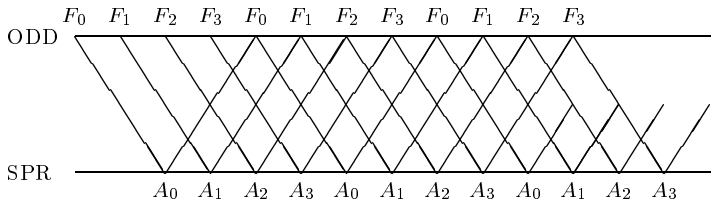
Slika 74: Vračanje na N za $w = 2$.

V času T_S po oddaji prvega bita okvirja F_0 pride potrdilo zanj (A_0) v celoti nazaj do sprejemnika. Po tem času je možna oddaja tretjega okvirja, ki je označen enako kot prvi okvir s F_0 in ker je medtem prišlo tudi potrdilo A_1 za drugi okvir,

lahko takoj po oddaji tretjega okvirja odda še četrtega, ki je ponovno označen s F_1. Zatem sledi spet čakanje na potrdilo. V času T_S je oddajnik oddal $w = 2$ okvirja. Izkoristek je:

$$E_{GBP}(p = 0) = \frac{w \cdot T_F}{T_S}.$$

Na sliki 75 smo predpostavili, da je velikost okna $w = 4$ izbrana tako, da je čas povratka potrdila ravno tak, da pride potrdilo nazaj tik predno bi oddajnik moral prekiniti z oddajanjem, $T_S = w \cdot T_F$. Izkoristek je za tako izbiro okna enak ena.



Slika 75: Vračanje na N za $w = 4$ in $wT_F = T_S$).

Izkoristek bi bil ena tudi v primeru, da bi potrdilo prišlo že prej. Manjši od ena je izkoristek v primeru premajhnega okna. Velja,

$$E_{GBN}(p = 0) = \min\{1, w\frac{T_F}{T_S}\},$$

medtem ko je izkoristek protokola ABP enak

$$E_{ABP}(p = 0) = \frac{T_F}{T_S}.$$

Razmere, ki smo jih skicirali na sliki 75, se ohranjajo, dokler ne pride do prve napake. Slika 76 prikazuje primer, ko se zgodi napaka na okvirju s številko F_0, vendar bi si lahko zamislili napako na kateremkoli okvirju, saj način številčenja do prve napake ne vpliva na izkoristek. Ker po času T_O oddajnik ne dobi potrdila, ponovno oddaja okvir F_0 in tudi vse tiste, ki jih je med tem že oddal. V primeru napake se zato izgubi čas T_O.

Izračunajmo povprečni čas za prenos enega okvirja. V primeru, da ni napake, se za prenos okvirja porabi čas T_F. To se zgodi z verjetnostjo $(1 - p)$. V primeru, da pride do napake, se izgubi čas T_O in okvir še ni prenešen. To se zgodi z verjetnostjo p. V primeru, da bi bila možna na istem okvirju samo enkratna napaka, bi bil povprečni čas za prenos okvirja enak $(1 - p)T_F + p(T_O + T_F)$. Ker pa se lahko na istem okvirju napaka ponovi poljubno mnogokrat, postane izraz za povprečni čas prenosa enega okvirja podoben tistemu pri analizi protokola ABP:

$$\bar{T} = (1 - p)T_F + p(T_O + \bar{T})$$

in

$$\bar{T} = T_F + \frac{p}{1-p} T_O.$$

Končno z upoštevanjem $T_O \approx T_S$ in $wT_F = T_S$, sledi:

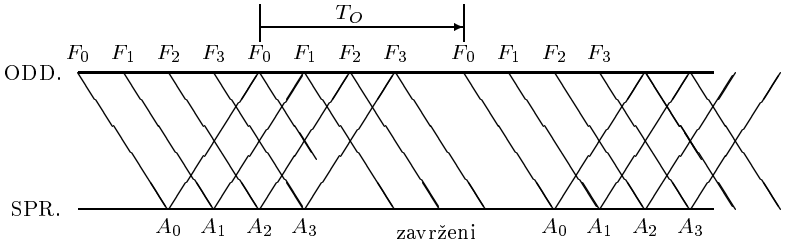
$$E_{GBN}(p) = \frac{T_F}{T_F + \frac{p}{1-p} w T_F} = \frac{1}{1 + \frac{p}{1-p} w}$$

Ker je verjetnost napake p majhna, je $(1-p)$ približno ena,

$$E_{GBN}(p) = \frac{1}{1+pw}$$

Če privzamemo, da je v praksi pw tudi majhen, je

$$E_{GBN}(p) \approx 1 - wp = 1 - p\frac{T_S}{T_F}.$$



Slika 76: Vračanje na N v primeru napak.

Za številčni primer izberimo enake podatke kot za protokol ABP, le hitrost prenašanja naj bo višja, in sicer $C = 128 Mb/s$. Pri tej hitrosti traja okvir $T_F = F/C = 8192/128 \times 10^{-6} = 128\mu s$, čas povratka potrdila pa je $T_S = F/C + A/C + 2 \cdot T_z = 128 + 0.125 + 1280 \approx 1408\mu s$. Izberemo velikost okna $w = 11$ in izkoristek je:

$$E_{GBN} = 1 - 0.001 \times \frac{1408}{128} = 1 - 0.011 = 0.989,$$

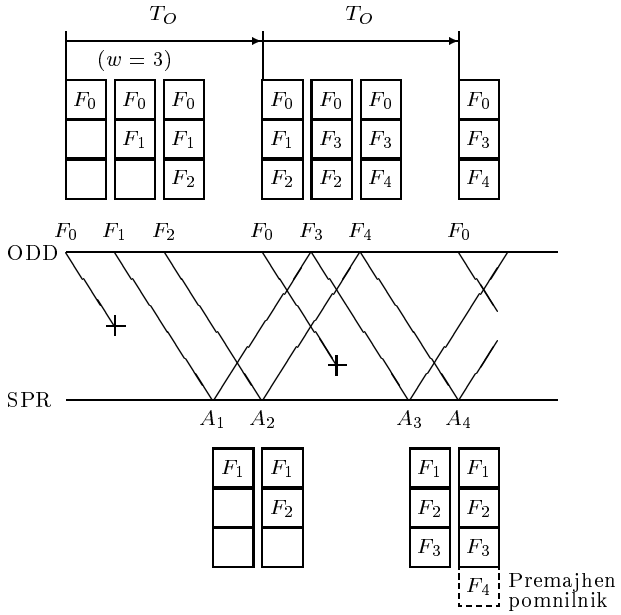
za protokol ABP pa samo:

$$E_{ABP} = 0.999 \frac{128}{1408} = 0.091$$

Protokol ABP je primeren le za nizke hitrosti in majhne razdalje.

4.3.4 Analiza protokola SRP

Videli bomo, da je določanje izkoristka protokola SRP bolj zahtevno. Predno pa se lotimo računanja izkoristka, se prepričajmo, da je za protokol SRP z velikostjo okna w potrebno številčiti okvirje po modulu $2w$ in ne po modulu w kot pri protokolu GBN. Nadalje, velikost okna je v tem primeru definirana z razliko med številko oddanega in številko potrjenega okvirja (številčeno po modulu $2w$), in ne s številom oddanih okvirjev, ki čakajo na potrdilo.



Slika 77: Selektivno ponavljanje in definicija velikosti okna (primer večkratne napake na okvirju F_0).

Najprej dokažimo, da drži zadnja trditev. Dokaz ne bo veliko trpel na splošnosti, če za velikost okna izberemo neko konkretno vrednost. Naj bo $w = 3$. Velikost oddajnega in sprejemnega pomnilnika je enaka velikosti okna. Razmere prikazuje slika 77. Zaenkrat okvirjev ne številčimo po modulu $2w$, kot smo trdili, da bi sicer zadostovalo, ampak si privoščimo številčenje kar od začetka do konca: $0, 1, 2, 3, 4, 5, 6, \ldots$. Zamislimo si, da sprejemnik prekine z oddajanjem takrat, kadar je oddal največ tri okvirje po zadnjem potrjenem potrdilu. Torej ne upošteva zahteve po razliki med številko oddanega okvirja in sprejetega potrdila. To bi moralo povzročiti težave.

Na sliki smo si zamislili napako na okvirju z zaporedno številko nič (F_0). Predno poteče čas čakanja na potrdilo, oddajnik odda še dva okvirja, ker je $w = 3$. Oddajnik hrani oddane in še ne potrjene okvirje v vmesnem pomnilniku, dokler ne dobi potrdil, ker bo morda moral katerega od okvirjev ponoviti. Po času T_O ne dobi potrdila A_0 in ponovno odda okvir (F_0), ki pa že spet ne bo prišel brez napake

do sprejemnika. Med tem je sprejemnik potrdil in tudi shranil okvirja F_1 in F_2. Ko oddajnik za ponovljenim prenosom okvirja F_0 dobi potrdilo za F_1, sprosti njegov medpomnilnik, vanj shrani okvir F_3 in ga odda. (Pozor, številka oddanega in še ne potrjenega okvirja se sedaj že razlikuje za več kot je velikost okna). Ko dobi potrdilo še za okvir F_2, sprosti njegov medpomnilnik, vanj vpiše okvir F_4 in ga odda. Med tem je medpomnilnik sprejemnika že poln, vendar medpomnjenih okvirjev ne more dati naprej, ker še vedno čaka na okvir F_0 in bi bilo zaporedje predanih okvirjev nepravilno. Da bi zagotovil zahtevi po pravilnem vrstnem redu (zaporednosti) okvirjev, bi mu morali povečati medpomnilnik. Teoretično bi moral biti medpomnilnik neskončen, kar je v praksi nemogoče. Končna velikost pa bi zadoščala, če bi upoštevali zahtevo po največji razliki številk okvirjev in potrdil.

Prepričajmo se še v potrebnost (in zadostnost) številčenja okvirjev in potrdil po modulu $2w$. Naj bo N zaporedna številka okvirja (številčeno od začetka do konca sporočila), ki ga je sprejemnik kot zadnjega predal naprej (konkretno mrežnemu sloju). Torej so bili dostavljeni tudi vsi prejšnji okvirji, medtem ko okvirja s številko $N+1$ še ni potrdil, ker bi ga sicer tudi že dal naprej. Oddajnik je med tem oddal največ še $n+w$-ti okvir, ker mora biti razlika med številko oddanega in številko potrjenega okvirja manjša ali enaka w. Oddajnik je med tem moral že dobiti potrdila za vse okvirje vključno z $N-w$-tim okvirjem, ker bi sicer ne oddal okvirja N. Iz tega sledi, da lahko sprejemnik dobi kot naslednji okvir okvir s številko iz množice

$$\{N-w+1, N-w+2, \ldots, N-1, N, N+1, \ldots, N+w-1, N+w\}$$

Pri tem pa tudi ve, da števila

$$\mathrm{STAR} = \{N-w+1, N-w+2, \ldots, N-1, N\}$$

pomenijo številke starih okvirjev, ki jih je že potrdil in tudi dal naprej. Take okvirje zavrže in vseeno potrdi. Če pa dobi okvir s številko

$$\mathrm{NOV} = \{N+1, \ldots, N+w-1, N+w\}$$

je lahko to samo okvir, ki ga je že uspešno sprejel in se zato nahaja v njegovem medpomnilniku ali pa ga še ni sprejel. Če še ni v pomnilniku ga shrani, sicer pa zavrže, v vsakem primeru pa ga potrdi. Vidimo, da je vseh potencialno možnih okvirjev $2 \cdot w$, ti pa bodo različno številčeni, če številčimo po modulu $2 \cdot w$. Sprejemnik ne bo pri takem načinu številčenja zamenjal starega okvirja z novim, kar bi vodilo v podvajanje ali izgubljanje okvirjev. Sprejemnik ne more zgrešiti.

Poglejmo primer za $w = 4$, ko zahtevamo številčenje po modulu $2w = 8$. Naj bo zaporedna številka zadnje predanega okvirja enaka $N = 121$. Pri številčenju po modulu 8 bo številka tega okvirja $121/8 = 5$. Sprejemnik lahko sprejme naslednje stare okvirje

$$\mathrm{STAR} = \{118, 119, 120, 121\}$$

ki so označeni z

$$\{2, 3, 4, 5\}$$

in naslednje nove okvirje

$$\text{NOV} = \{122, 123, 124, 125\}$$

ki so označeni z

$$\{6, 7, 0, 1\}$$

Oznake starih in novih okvirjev se res razlikujejo.

Obstaja še vprašanje, ali lahko tako številčenje zmede oddajnik. Oddajnik je gotovo že dobil vsa potrdila do vključno A_{n-4}, saj je že oddal okvir s številko N, vendar pa zagotovo še ni oddal nobenega okvirja za okvirjem s številko $N+4$. Torej morajo biti potrdila znotraj množice

$$\{N-3, N-2, \ldots, N, N+1, \ldots, N+4\}$$

Tudi ta števila so različna, če za velikost okna $w = 4$ številčimo po modulu osem.

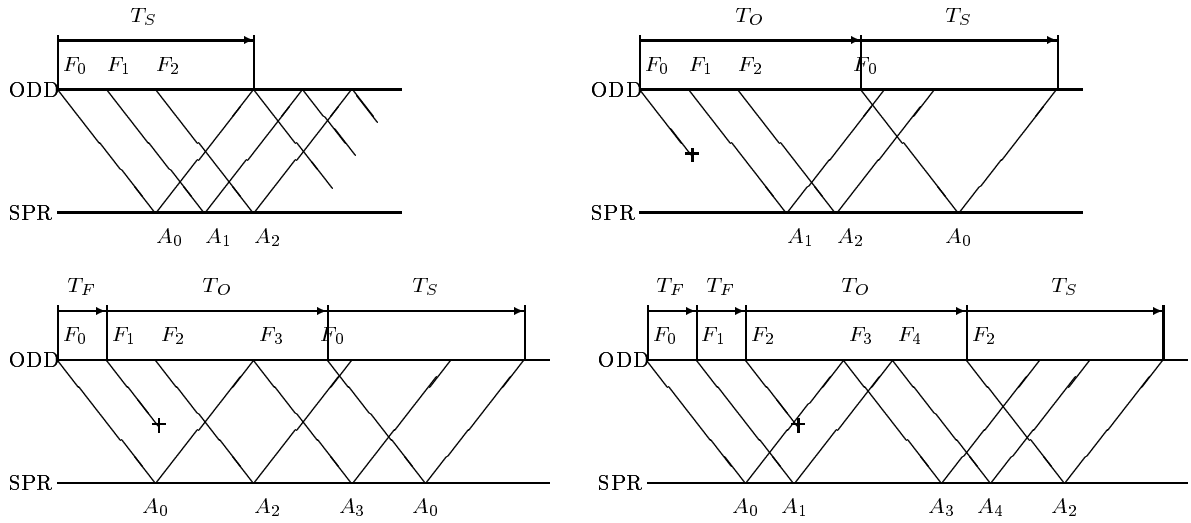
Izkoristek Izračunajmo izkoristek protokola SRP. Pri tem nam bo pomagala slika 78. V slučaju, da ni napak, oddamo v času T_S ravno w okvirjev. Zato je kanal izkoriščen čas $w \cdot T_F$ (v narisanem primeru je izbrana velikost okna $w = 3$). Izkoristek je:

$$E_{SRP}(p = 0) = \min\{1, w \cdot \frac{T_F}{T_S}\},$$

enako kot za protokol GBN. Predno se lotimo splošnega računanja izkoristka z upoštevanjem napak, si problem poenostavimo s predpostavko, da sta medpomnilnika oddajnika in sprejemnika neskončna (tako velika, da ni potrebno zaradi pomankanja pomnilnika nikoli čakati). V tem primeru bi prenašali okvirje drugega za drugim, brez premorov. Zaradi napak bi nekatere okvirje prenašali večkrat. Ponavljanje bi pomenilo izgubo časa. Čbi ne bilo napak, bi vsak okvir prenašali samo enkrat in izkoristek bi bil ena. V primeru, da bi bila verjetnost napačnega prenosa $p = 0.5$, bi v povprečju okvir prenašali dvakrat, povprečno število prenosov istega okvirja je namreč enako $\frac{1}{1-p}$. Do tega pridemo na enak način, kot smo to že naredili v prejšnjih razdelkih. Izkoristek protokola SRP z neskončnim oknom ($w = \infty$) je:

$$E_{SRP,w=\infty}(p) = \frac{T_F}{\frac{1}{1-p}T_F} = 1 - p.$$

To je zgornja meja izkoristka v odvisnosti od verjetnosti napake. Zaradi končne velikosti okna bo izkoristek v stvarnih razmerah manjši. Izračunajmo ga. Da bo problem lažje rešljiv, bomo upoštevali, da je verjetnost ponovne napake istega



Slika 78: Primeri razmer za protokol SRP: brez napake, napaka na prvem, na drugem ali na tretjem okvirju (za $w = 3$).

okvirja zanemarljiva. Podobno, verjetnost napake na dveh okvirjih znotraj okna, naj je tudi zanemarljiva.

Za velikost okna $w = 3$ obstajajo v tem primeru tri možnosti napak, napaka na prvem, na drugem ali na tretjem okvirju, glej sliko 78. V primeru napake na prvem okvirju (F_0), prenesemo v času $t_0 = T_O + T_S$ tri okvirje, $n_0 = 3$. V primeru napake na drugem okvirju prenesemo v času $t_1 = T_F + T_O + T_S$ štiri okvirje, $n_1 = 4$. Podobno prenesemo pri napaki na tretjem okvirju v času $t_2 = 2 \cdot T_F + T_O + T_S$ pet okvirjev, $n_2 = 5$. Splošen izraz za t_i in n_i v odvisnosti od velikosti okna, je:

$$n_i = w + i, \qquad t_i = T_O + T_S + i \cdot T_F \quad (i = 0, 1, 2, \ldots, w - 1),$$

Pri pogoju $T_S \approx T_O$ in $T_S = w \cdot T_F$ je

$$t_i = 2 \cdot w \cdot T_F + i \cdot T_F = T_F \cdot (2 \cdot w + i).$$

Verjetnost napake na enem izmed treh okvirjev je

$$p_i = p \cdot (1 - p)^2 \approx p$$

in je neodvisna od številke okvirja, na katerem se zgodi napaka. Verjetnost, da ni napake, pa je $p_n(1 - p)^3 \approx 1 - 3p$, ali v splošnem $1 - wp$. V tem primeru prenesemo v času $t_w = T_S$ tri okvirje, $n_w = 3$. Če zanemarimo možnost, da se zgodi napaka na dveh okvirjih, so to tudi vsi možni dogodki. Čase t_i in število prenešenih okvirjev n_i lahko obravnavamo kot diskretni naključni spremenljivki T in N, katerih zalogo vrednosti in porazdelitveni zakon smo pravkar ugotovili,

$$T = \begin{pmatrix} t_0, & t_1, & t_2, & \ldots, & t_w \\ p, & p, & p, & \ldots, & 1 - wp \end{pmatrix}.$$

$$N = \begin{pmatrix} n_0, & n_1, & n_2, & \ldots, & n_w \\ p, & p, & p, & \ldots, & 1-wp \end{pmatrix}.$$

Izračunajmo povprečne vrednosti naključnih spremenljivk T in N,

$$\begin{aligned}
\bar{T} &= \sum_{i=0}^{i=w} p_i \cdot t_i \\
&= T_F[\cdot \sum_{i=0}^{i=w-1} p \cdot (2 \cdot w + i) + (1 - wp) \cdot w] \\
&= T_F[p \cdot 2w^2 + p\frac{(w-1) \cdot w}{2} + w - p \cdot w^2] \\
&= w \cdot T_F[1 + \frac{p}{2}(3w - 1)].
\end{aligned}$$

$$\begin{aligned}
\bar{N} &= \sum_{i=0}^{i=w} p_i \cdot n_i \\
&= \sum_{i=0}^{i=w-1} p \cdot (w + i) + (1 - wp)w \\
&= p \cdot w^2 + p\frac{w(w-1)}{2} + w - p \cdot w^2 \\
&= w + p\frac{w(w-1)}{2}.
\end{aligned}$$

Izkoristek pa je:

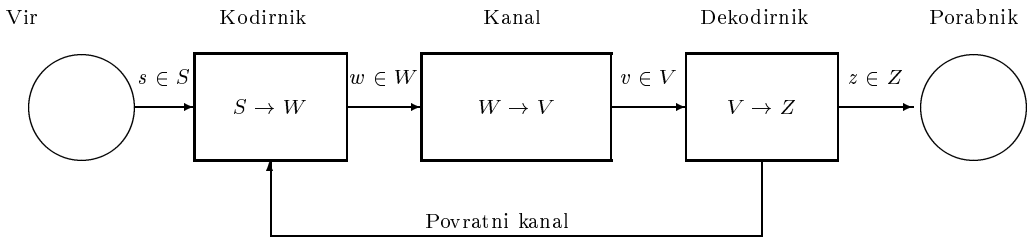
$$E_{SRP}(p) = \frac{T_F \bar{N}}{\bar{T}} = \frac{2 + p(w-1)}{2 + p(3w-1)}.$$

4.4 Odkrivanje in popravljanje napak

Pri prenosu podatkov se *napakam* ne moremo nikoli popolnoma izogniti. Podatki, ki jih sprejme sprejemna naprava, se zaradi nepopolnosti prenosnih naprav ter zunanjih motilnih vplivov, ki se s časom nakljućno spreminjajo, v splošnem razlikujejo od oddanih podatkov. Pravimo, da prenos podatkov ni popolnoma zanesljiv. Zanesljivost prenosa lahko povečamo na več načinov. Na primer tako, da za prenos uporabimo kvalitetnejše (dražje) prenosne poti. Zanesljivost prenosa se da vedno povečati na račun nižje hitrosti. Pri nižji hitrosti prenosa oddani signal dalj časa vstraja v enem od možnih stanj, zato ga sprejemna naprava lažje loči od nakljućnih stanj šuma. Tretjo možnost za povečanje zanesljivosti prenosa daje *kodiranje*. Če povzamemo, zanesljivost prenosa je odvisna od kvalitete prenosne poti, od hitrosti prenosa in načina kodiranja.

V tem podpoglavju si bomo ogledali osnovne postopke kodiranja, ki omogočajo, da v danih pogojih (pri dani prenosni poti) z *odkrivanjem* in redkeje s *popravljanjem* napak povečamo zanesljivost prenosa. Kljub številnim študijam kodov za popravljanje napak se v praksi več uporabljajo postopki za odkrivanje napak, skupaj z enim od mehanizmov (protokolov) za potrjevanje pravilno ali napačno sprejetih podatkov s strani sprejemne naprave.

Zamisel odkrivanja in popravljanja napak bomo razložili na modelu komunikacijskega sistema, ki je narisan na sliki 79.



Slika 79: Model komunikacijskega sistema. Informacija ($z \in Z$), ki prispe do porabnika informacije ni nujno enaka informaciji, ki jo ustvari vir $s \in S$. S primernim načinom kodiranja, prenašanja in dekodiranja pa skušamo doseči, da bi bila ($z = s$).

Informacijski vir ustvarja informacijo, oddaja simbole iz končne zaloge S. To informacijo kodirna naprava (za)kodira in pošlje v *informacijski kanal*. Zaradi napak med prenosom sprejeti podatki $v \in V$ niso vedno enaki oddanim podatkom $w \in W$. Dekodirna naprava sprejete podatke V dekodira in da porabniku informacijo $z \in Z$. Informacija, ki prispe do porabnika informacije, zaradi napak ni nujno enaka informaciji, ki jo ustvari vir. S primernim načinom kodiranja, pošiljanja in dekodiranja pa skušamo to doseči.

Praktično vsi postopki za odkrivanje ali popravljanje napak temeljijo na vnašanju odvečnosti (redundance) v sporočila. Informacijskim simbolov $i_1, i_2, \ldots, i_k$, ki nosijo informacijo vira S, se na oddajni strani po nekem pravilu f_α določi (izračuna) odvečne simbole $o_{k+1}, o_{k+2}, \ldots, o_n$,

$$o_\alpha = f_\alpha(i_1, i_2, \ldots, i_k) \quad (\alpha = k+1, k+2, \ldots, n)$$

nakar se vse skupaj pošlje v kanal,

$$w = i_1, i_2, \ldots, i_k, o_{k+1}, o_{k+2}, \ldots, o_n.$$

Sprejemna naprava sprejme zaporedje simbolov w in po enakem pravilu kot oddajna naprava izračuna odvečne simbole, te pa primerja s sprejetimi. V primeru, da se izračunano zaporedje ne ujema s sprejetim, pomeni to napako oziroma točneje, da je napaka odkrita. Nekateri načini kodiranja omogočajo, da sprejemna naprava na osnovi izračuna določi mesto napake in jo tudi popravi. V primeru, da sprejemna naprava napake ne zna popraviti, pa zahteva ponovitev prenosa.

Osnovni parametri, s katerimi vrednotimo uspešnost kodirnega sistema so koristnost, odvečnost in zanesljivost.

Koristnost koda E je definirana z razmerjem med številom informacijskih simbolov k in številom vseh simbolov n (informacijskih in odvečnih),

$$E = \frac{k}{n}.$$

Odvečnost koda R je podana z razmerjem med številom odvečnih ali kontrolnih simbolov in številom vseh prenašanih simbolov,

$$R = \frac{n-k}{n} = 1 - E.$$

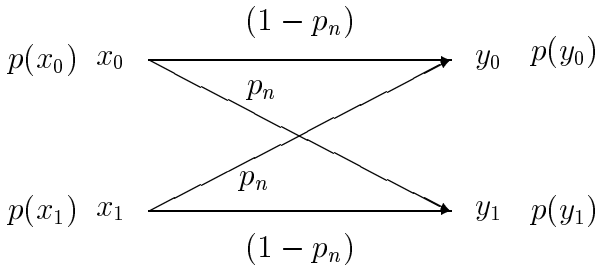
Odvečnost in koristnost sta običajno, a ne nujno, povezana z zmožnostjo koda za odkrivanje in/ali popravljanje napak. Večja odvečnost omogoča s stališča odkrivanja in popravljanja boljše kode.

Zanesljivost prenosa je definirana z razmerjem med številom pravilno prenešenih simbolov in številom vseh simbolov, ki jih prenašamo.

4.4.1 Osnovna zamisel odkrivanja in popravljanja napak

Da bi razložili zamisel odkrivanja se prej spoznajmo z verjetnostnim modelom kanala. Napake so posledica motenj, motnje pa so naključne narave. Na manj

kvalitetnih prenosnih poteh so motnje in s tem napake bolj pogoste ali bolj *verjetne* kot na kvalitetnih prenosnih poteh. Verjetnost napake na simbolu, na skupini simbolov ali na celemu sporočilu, je eden od parametrov, ki podajajo lastnost prenosne poti. Slika 80 prikazuje verjetnostni model binarnega simetričnega kanala. Kanal je v tem primeru določen z verjetnostjo napake med prenosom, ostale značilnosti kanala (prenosni medij, oblika signala, namen uporabe) pa nas ne zanimajo. Na vhodu sta dovoljena dva simbola, to sta x_0 in x_1. Na izhodu sta možna dva simbola, y_0 in y_1. Zato je kanal *binaren*. Verjetnost, da se simbol x_0 zaradi napake spremeni v y_1 je enaka verjetosti, da se simbol x_1 spremeni v y_0. Zato je kanal *simetričen*.



Slika 80: Verjetnostni model binarnega simetričnega kanala z verjetnostjo napake na simbolu p_n. Za idealen kanal je $p_n = 0$, za neuporaben kanal je verjetnost napake 0.5. Na primer, za telefonski vod naj bi bila verjetnost napake približno 10^{-4}.

Sedaj pa si mislimo vir informacije, ki lahko zavzame eno od dveh enakoverjetnih stanj, s_0 in s_1. Priredimo tem dvem stanjem binarna simbola 0 in 1, na primer $s_0 \to w_0 = 0$ in $s_1 \to w_1 = 1$. Vzemimo, da zaporedja tako kodiranih stanj (sporočila) pošiljamo po binarnem simetričnem kanalu do sprejemnika. Naj bo verjetnost napake na kanalu $p_n = 0.1$. Z drugimi besedami, od 1000 oddanih simbolov se bo med prenosom 100 od njih spremenilo iz 1 v 0 ali iz 0 v 1, ne da bi na sprejemni strani za to vedeli. Ker simbol 0 pomeni stanje s_0, simbol 1 pa stanje s_1, bomo napačno razpoznali 100 stanj. Pri takem načinu kodiranja ne bo moč odkriti niti ene same napake.

Sedaj pa priredimo (zakodirajmo) vsakemu od stanj s_0 in s_1 po dva binarna simbola, in sicer: $s_0 \to w_0 = 00$ in $s_1 \to w_1 = 11$. Zaradi napak med prenosom so na izhodu možna vsa zaporedja $v_1 = w_0 = 00$, $v_2 = 01$, $v_3 = 10$, $v_4 = w_1 = 11$. V primeru, da se zgodi napaka na enem samem simbolu, sprejmemo zaporedje v_2 ali v_3. Ker sta na vhodu možni le kombinaciji $w_0 = 00$ in $w_1 = 11$ vemo, da je prišlo do napake. Iz sprejete kombinacije se seveda ne da razbrati, katera od besed w_0 in w_1 je bila oddana. Enako verjetna je napaka na prvem simbolu kot na drugem simbolu. Napake ne znamo popraviti. V primeru napake na dveh

simbolih (dvojne napake), se w_0 spremeni v $v_4 = w_1$ ali w_1 v $v_1 = w_0$. Sprejemnik se take napake ne zaveda. Verjetnost, da se to zgodi je:

$$p_{2n} = p_n \times p_n = 0.1 \times 0.1 = 0.01,$$

verjetnost, da se to ne zgodi pa 0.99. Sprejemnik se odloči za verjetnejšo možnost, ki pa je v 0.01 primerih napačna. To pomeni, da bi od tisoč oddanih stanj bilo deset razpoznanih napačno. Verjetnost, da napaka ni odkrita je za cel velikostni razred manjša kot v prvem primeru. Pri dani hitrosti prenosa podatkov bi v istem času prenesli pol manj informacije. Dejanski informacijski pretok se je zmanjšal na polovico.

Dodajmo še en simbol in zakodirajmo s_0 z $w_0 = 000$ in s_1 z $w_1 = 111$. Zaradi napak med prenosom je na izhodu kanala možno vsako zaporedje treh simbolov. Takih zaporedij je osem, v_i, $(i = 0, 1, \ldots, 7)$. Sprejemnik napake ne odkrije, če se zgodijo tri napake, saj se ena kodna beseda popolnoma spremeni v drugo. Verjetnost, da se to zgodi je

$$p_{3n} = p_n^3 = 0.1^3 = 0.001.$$

To pomeni, da sprejemnik v zaporedju tisočih stanj zagreši samo še eno napako.

Sedaj pa zahtevajmo, da sprejemnik napake ne le odkriva, ampak tudi popravlja. Iz zaporedja treh simbolov se mora sprejemnik odločiti, kaj je oddal oddajnik. Vzemimo, da se odloča takole: če sprejme več ničel kot enic, se odloči za s_0, če sprejme več simbolov 1 pa s_1. V primeru, da je verjetnost napake (spremembe simbola) manjša od 0.5, se zdi tak način odločanja smiseln.

Sprejem	Odločitev	Sprejem	Odločitev
$v_0 = 000$	s_0	$v_1 = 001$	s_0
$v_2 = 010$	s_0	$v_3 = 011$	s_1
$v_4 = 100$	s_0	$v_5 = 101$	s_1
$v_6 = 110$	s_1	$v_7 = 111$	s_1

Pravilno se odločimo vsakič, ko se od treh pokvari samo en simbol. Pravimo, da popravimo vsako enojno napako.

Napaka ni odkrita v primeru, da se na zaporedju treh simbolov zgodita dve ali tri napake, ne glede na način, kako se to zgodi. Izračunajmo verjetnost napačne odločitve. Pričakujemo, da je manjša kot prej, ko je znašala 0.1. Imamo:

$$\begin{aligned} p_{napake} &= 0.1 \times 0.1 \times 0.9 + 0.9 \times 0.1 \times 0.1 + 0.1 \times 0.9 \times 0.1 + \\ &+ 0.1 \times 0.1 \times 0.1 = \\ &= 3 \times 0.1^2 \times 0.9 + 0.1^3 = 0.028. \end{aligned}$$

Vidimo, da se je verjetnost napačne odločitve zmanjšala z 0.1 na 0.028, toda na račun nižje efektivne hitrosti prenosa, ki je trikrat manjša.

V zadnjem primeru smo vsako enojno napako ne le odkrili, temveč tudi popravili. Če bi dolžino kodnih besed še povečali, bi sposobnost odkrivanja in/ali popravljanja napak še povečali. To je razumljivo, saj se daljši kodni besedi bolj razlikujeta in se potemtakem ena beseda težje popolnoma spremeni v drugo. Izgubljamo pa na dejanski hitrosti prenosa.

Število simbolov, na katerih se dve enako dolgi zaporedji binarnih simbolov razlikujeta, imenujemo *Hammingova* razdalja. Na primer, v zadnjem primeru je Hammingova razdalja

$$d(w_0, w_1) = 3.$$

Kod je sposoben odkrivati in popravljati tem večkratne napake, čim večja je Hammingova razdalja med besedami koda. Zato želimo, da bi bila razdalja med besedami čim večja. To se da povečati z dodajanjem simbolov, ki sicer niso potrebni, ne nosijo informacije in so s tega stališča odveč. So pa bistveni za odkrivanje in popravljanje napak.

4.4.2 Hammingova razdalja in zmožnost koda za odkrivanje napak

Vzemimo, da želimo napraviti takšen kod $W = \{w_i\}$, ki je zmožen odkrivati vse napake na enem, na dveh, na e simbolih, ne glede na to, na kakšen način se napake pojavijo. Velja:

$$d(w_i, w_j) \geq e + 1, \tag{20}$$

kjer je $d(w_i, w_j)$ Hammingova razdalja med poljubnima kodnima besedama koda.

V to se prepričajmo na primeru. Vzemimo, da ima kod samo dve kodni besedi dolžine $n = 8$:

$$w_1 = 00000000 \quad in \quad w_2 = 11111111.$$

Razdalja med njima je $d(w_1, w_2) = 8$. Vzemimo, da sprejmemo zaporedje

$$v = 111111110.$$

Zaporedje v je neveljavno - ni kodna beseda. Veljavni sta samo zaporedji w_1 in w_2, sprejeto zaporedje pa ni enako nobeni od besed. Možno je, da je bila oddana beseda w_2, vendar smo zaradi napake na zadnjem simbolu sprejeli v. Manj verjetno, vendar tudi možno pa je, da je bila oddana beseda w_1, zaradi napak na prvih sedmih simbolih ($e = 7$) pa smo sprejeli v.

$$d = 8 = e + 1 \Longrightarrow e = 7.$$

V obeh primerih vemo, da je prišlo med prenosom do napake. V primeru osmih napak bi se ena beseda popolnoma spremenila v drugo, vendar iz sprejetega zaporedja za to ne bi mogli vedeti. V splošnem odkrijemo vsako tako kombinacijo napak, ki ene kodne besede ne spremeni v kakšno drugo kodno besedo. To se ne zgodi, če je najmanjša razdalja med kodnimi besedami vsaj za eno večja od števila napačnih simbolov.

4.4.3 Hammingova razdalja in zmožnost koda za popravljanje napak

Vzemimo, da želimo skonstruirati tak kod $W = \{w_i\}$, ki bi bil sposoben napake ne le odkrivati, ampak tudi popravljati. Velja naslednji izrek. Kod z lastnostjo

$$d(w_i, w_j) \geq 2 \times e + 1 \tag{21}$$

je sposoben odkrivati in popravljati vse e-kratne ali manj-kratne napake. Kod z lastnostjo

$$d(w_i, w_j) \geq 2 \times e \tag{22}$$

pa je sposoben odkrivati in popravljati vse $(e-1)$-kratne ali manj-kratne napake, medtem ko je e-kratne napake sposoben samo odkrivati, popravljati pa ne.

Spet se v to prepričajmo na primeru. Vzemimo, da sta w_i in w_j kodni besedi z najmanjšo razdaljo. Vse druge kodne besede so si bolj oddaljene. Naj bo med njima razdalja $d(w_i, w_j) = 5$. Označimo z v_1, v_2, v_3, v_4 poljubna možna zaporedja, ki se od w_i razlikujejo na enem, na dveh, na treh in na štirih mestih,

$$d(v_1, w_i) = 1,\ d(v_2, w_i) = 2,\ d(v_3, w_i) = 3,\ d(v_4, w_i) = 4.$$

Za razdalje do kodne besede w_j pa naj velja

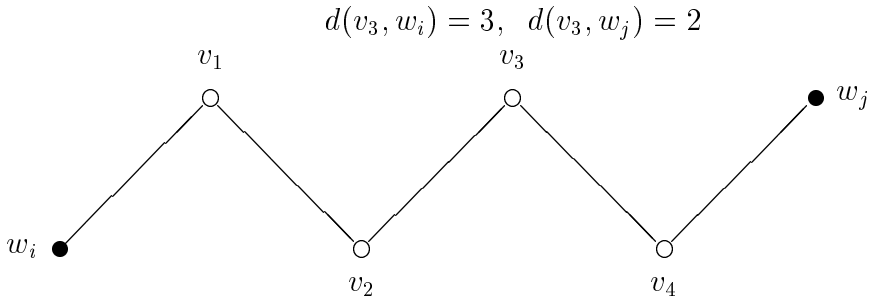
$$d(v_4, w_j) = 1,\ d(v_3, w_j) = 2,\ d(v_2, w_j) = 3,\ d(v_1, w_i) = 4.$$

Razmere so narisane na sliki 81.

Vzemimo, da sprejmemo zaporedje v_3. To zaporedje sprejmemo v primeru, da pošljemo w_j in se zgodita dve napaki, ali da pošljemo w_i in se zgodijo tri napake. Zaporedje v_3 je bližje kodni besedi w_j. Če je verjetnost napake manj od 0.5, je verjetnost treh napak manjša od verjetnosti dveh napak. Zato se odločimo, da je bila oddana kodna beseda, ki je bližja sprejetemu zaporedju. V našem primeru je $d(v_3, w_j) < d(v_3, w_i)$, zato se odločimo za besedo w_j.

V primeru dveh napak ($e = 2$) bi bila ta odločitev pravilna in napako smo uspeli popraviti, kot smo trdili v začetku:

$$d(w_i, w_j) = 5 = 2 \times e + 1 \Longrightarrow e = 2.$$



Slika 81: Primer koda s sposobnostjo popravljanja napak. Najmanjša razdalja med besedami koda je $d(w_i, w_j) = 5$. Kod je sposoben popravljati še vse dvakratne ($e = 2$) napake ($5 = 2e + 1$).

V primeru, ko bi se v resnici zgodile tri napake, bi bila odločitev napačna, napake ne bi popravili.

Če bi bila razdalja med besedama w_i in w_j enaka 4, bi bilo neko zaporedje v enako oddaljeno od obeh besed, $d(v, w_i) = d(v, w_j) = 2$. V primeru, da bi zaradi napak sprejeli zaporedje v, bi sicer vedeli, da so prisotne napake, napako bi odkrili, ne bi pa se znali odločiti, kaj je bilo oddano; od tu neenačba (22).

4.4.4 Preverjanje parnosti

Preverjanje parnosti je eden najpogostejših postopkov za odkrivanje napak. Pri oddaji dodamo h k informacijskim bitom $b_j, \;\; (j = 1, 2, \ldots, k)$ še parnostni bit. Možno je 'sodo' ali 'liho' preverjanje parnosti. Pri sodem preverjanju dodamo parnostni bit (simbol) tako, da je skupno število enic (skupaj s parnostnim bitom) v besedi sodo. Pri lihem preverjanju dodamo parnostni bit tako, da je skupno število enic liho. Od tu obe imeni. Vrednost parnostnega bita za k informacijskih bitov izračunamo pri sodem preverjanju parnosti po enačbi:

$$b_1 + b_2 + \cdots + b_j + \cdots + b_k + p = 0 \;\; (mod\ 2), \tag{23}$$

pri lihem preverjanju pa po enačbi:

$$1 + b_1 + b_2 + \cdots + b_j + \cdots + b_k + p = 0 \;\; (mod\ 2). \tag{24}$$

Sodo preverjanje parnosti je pogostejše od lihega. Če je zaporedje informacijskih bitov na primer:

$$1\ 1\ 0\ 0\ 1\ 1\ 0$$

je pri sodem preverjanju parnostni bit enak $\mathbf{0}$,

$$1 + 1 + 0 + 0 + 1 + 1 + \mathbf{0} = 0 \quad (mod\ 2),$$

odposlano zaporedje pa

$$1\ 1\ 0\ 0\ 1\ 1\ 0\ \mathbf{0}.$$

Ko sprejemna naprava sprejme zaporedje simbolov, najprej preveri pogoj parnosti:

$$c = b_1 + b_2 + \cdots + b_j + \cdots + b_k + p \pmod 2, \tag{25}$$

V primeru, da je c različen od nič, pomeni to napako med prenosom.

Najmanjša Hammingova razdalja koda je 2. Po enačbi (20) je možno odkrivati vse enkratne napake ($2 = e + 1 \Rightarrow e = 1$), zaradi pravila parnosti pa je možno odkriti vsako lihokratno število napak.

4.4.5 Vzdolžno in prečno preverjanje parnosti

Pri vzdolžnem in prečnem preverjanju parnosti dodamo zaporedjem k informacijskih bitov parnostni bit za prečno preverjanje:

$$\begin{array}{lllllllll}
w_\alpha & = b_{\alpha 1} & +b_{\alpha 2} & +\cdots & +b_{\alpha j} & +\cdots & +b_{\alpha k} & +p_\alpha & = 0 \pmod 2 \\
w_\beta & = b_{\beta 1} & +b_{\beta 2} & +\cdots & +b_{\beta j} & +\cdots & +b_{\beta k} & +p_\beta & = 0 \pmod 2 \\
\cdots & = & & \cdots & & & & & \\
w_\delta & = b_{\delta 1} & +b_{\delta 2} & +\cdots & +b_{\delta j} & +\cdots & +b_{\delta k} & +p_\delta & = 0 \pmod 2 \\
\cdots & = & & \cdots & & & & & \\
w_\omega & = b_{\omega 1} & +b_{\omega 2} & +\cdots & +b_{\omega j} & +\cdots & +b_{\omega k} & +p_\omega & = 0 \pmod 2
\end{array} \tag{26}$$

celotnemu zaporedju besed $w_\alpha, w_\beta, \ldots, w_\omega$, to je sporočilu ali delu sporočila pa parnostno besedo $w_p = (p_1, p_2, \ldots, p_j, \ldots, p_{k+1})$ za vzdolžno preverjanje:

$$\begin{array}{lllllllll}
b_{\alpha 1} & +b_{\beta 1} & +\cdots & +b_{\delta 1} & +\cdots & +b_{\omega 1} & +p_1 & = 0 \pmod 2 \\
b_{\alpha 2} & +b_{\beta 2} & +\cdots & +b_{\delta 2} & +\cdots & +b_{\omega 2} & +p_2 & = 0 \pmod 2 \\
 & & \cdots & & & & & & \\
b_{\alpha j} & +b_{\beta j} & +\cdots & +b_{\delta j} & +\cdots & +b_{\omega j} & +p_j & = 0 \pmod 2 \\
 & & \cdots & & & & & & \\
b_{\alpha k} & +b_{\beta k} & +\cdots & +b_{\delta k} & +\cdots & +b_{\omega k} & +p_k & = 0 \pmod 2 \\
p_\alpha & +p_\beta & +\cdots & +p_\delta & +\cdots & +p_\omega & +p_{k+1} & = 0 \pmod 2
\end{array} \tag{27}$$

Poglejmo primer. Naj bo zaporedje informacjskih bitov:

$$1\ 1\ 1\ 0\ 1\ 0\ 1\ 0\ 0.$$

Dodajmo vsakemu zaporedju treh simbolov en parnostni simbol (po vrsticah) in celotnemu zaporedju še parnostno besedo (parnostne simbole po stolpcih):

$$
\begin{array}{lcccccc|cc}
w_\alpha & = & 1 & + & 1 & + & 1 & + & 1 & = 0 \\
 & & + & & + & & + & & + & \\
w_\beta & = & 0 & + & 1 & + & 0 & + & 1 & = 0 \\
 & & + & & + & & + & & + & \\
w_\gamma & = & 1 & + & 0 & + & 0 & + & 1 & = 0 \\
 & & + & & + & & + & & + & \\
\hline
w_p & = & 0 & + & 0 & + & 1 & + & 1 & = 0 \\
 & & \| & & \| & & \| & & \| & \\
 & & 0 & & 0 & & 0 & & 0 &
\end{array}
$$

Pošljemo zaporedje $w_\alpha, w_\beta, w_\gamma, w_p$. Zaradi napak med prenosom v splošnem sprejmemo različno zaporedje $v_\alpha, v_\beta, v_\gamma, v_p$.

Vzemimo, da se med prenosom zgodi napaka in sprejmemo zaporedje simbolov:

$$1\ 1\ 1\ 1\ 0\ 0\ 0\ 1\ 1\ 0\ 0\ 1\ 0\ 0\ 1\ 1.$$

Kje je napaka? Preverimo vzdolžni in prečni pogoj parnosti:

$$
\begin{array}{lcccccc|cc}
v_\alpha & = & 1 & + & 1 & + & 1 & + & 1 & = 0 \\
 & & + & & + & & + & & + & \\
v_\beta & = & 0 & + & 0 & + & 0 & + & 1 & = 1 \\
 & & + & & + & & + & & + & \\
v_\gamma & = & 1 & + & 0 & + & 0 & + & 1 & = 0 \\
 & & + & & + & & + & & + & \\
\hline
v_p & = & 0 & + & 0 & + & 1 & + & 1 & = 0 \\
 & & \| & & \| & & \| & & \| & \\
 & & 0 & & 1 & & 0 & & 0 &
\end{array}
$$

V drugi vrstici in tretjem stolpcu pogoj parnosti ni izpolnjen. Napaka je na presečišču vrstice in stolpca. Ker vemo, kje je napaka, jo lahko popravimo.

Vzdolžno in prečno preverjanje parnosti omogoča odkrivanje in popravljanje vseh enkratnih napak ($e = 1$). Hammingova razdalja med besedami mora biti torej vsaj $2 \times e + 1 = 3$. Kljub temu, da kod omogoča tudi popravljanje napak, pa se v praksi uporablja v glavnem samo za odkrivanje napak.

4.4.6 Hammingov kod

Hammingov kod je poseben primer koda iz velike družine *parnostnih* kodov, ki omogoča ne le odkriti, ampak tudi popraviti vsako enojno napako (napako na enem simbolu). Tudi prej obravnavano preverjanje parnosti ter vzdolžno in prečno preverjanje parnosti sodita v družino parnostnih kodov. V osnovnem preverjanju parnosti smo imeli za skupino informacijskih simbolov en sam parnostni

simbol. Pri preverjanju vzdolžne in prečne parnosti je bilo parnostnih simbolov že več. V splošnem lahko po nekih pravilih dodamo poljubno mnogo parnostnih simbolov. S številom parnostnih simbolov se seveda veča odvečnost koda. Pričakujemo, da se istočasno z večanjem odvečnosti veča tudi zmožnost koda za odkrivanje in/ali popravljanje napak.

Vzemimo kod $W = \{w_i\}$ z dolžino kodnih besed n. Vse kodne besede so torej enako dolge in imajo po n simbolov. Naj bo od teh n simbolov k informacijskih in m parnostnih (odvečnih). Da določimo m parnostnih bitov potrebujemo m pogojev, m enačb oblike (23). Poglejmo primer. Naj bo $n = 6,\ k = 3$ in $m = 3$. Izberimo si simbole b_1, b_2, b_3 za informacijske, simboli b_4, b_5, b_6 pa naj bodo parnostni, kot to določajo naslednji trije pogoji:

$$\begin{array}{ccccc} b_1 & + & b_4 & = 0 & (mod\ 2) \\ b_2 & + & b_5 & = 0 & (mod\ 2) \\ b_3 & + & b_6 & = 0 & (mod\ 2) \end{array}$$

Enačbe lahko zapišemo v matrični obliki:

$$\begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \quad (mod\ \ 2).$$

ali bolj zgoščeno:

$$\mathbf{A}w^T = \mathbf{0}. \tag{28}$$

Matriko $\mathbf{A}$ imenujemo *parnostna* matrika in popolnoma določa kod W - kodne besede w tega koda so rešitve enačbe (28). Kodnih besed je toliko, kolikor je rešitev enačbe. Ko izberemo matriko $\mathbf{A}$ je torej že določena dolžina kodnih besed, število informacijskih simbolov in s tem število kodnih besed ter število parnostnih simbolov. S parnostno matriko je določena tudi sposobnost koda za popravljanje napak.

Hammingovemu kodu pripada naslednja parnostna matrika velikosti $(m \times n) = (3 \times 7)$:

$$\mathbf{A} = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix}$$

Dolžina kodnih besed je $n = 7$. Brez dodatnih omejitev bi bilo možnih $2^n = 2^7 = 128$ kodnih besed. Vendar imamo za Hammingov kod še tri pogoje, tri linearno

neodvisne enačbe, iz katerih se da izračunati tri neznanke - tri parnostne simbole. Za zapis informacije ostanejo samo še štirje simboli ($k = 4$). Te lahko poljubno izberemo. Možno je izbrati $2^k = 2^4 = 16$ različnih kombinacij. Denimo, da so simboli b_1, b_2 in b_4 parnostni, ostali štirje simboli b_3, b_5, b_6 in b_7 pa informacijski.

Kodne besede koda določimo tako, da za vsako kombinacijo informacijskih simbolov izračunamo parnostne:

$$\begin{aligned} w_0 &= b_1\ b_2\ 0\ b_4\ 0\ 0\ 0 \\ w_1 &= b_1\ b_2\ 0\ b_4\ 0\ 0\ 1 \\ w_2 &= b_1\ b_2\ 0\ b_4\ 0\ 1\ 0 \\ w_3 &= b_1\ b_2\ 0\ b_4\ 0\ 1\ 1 \\ w_4 &= b_1\ b_2\ 0\ b_4\ 1\ 0\ 0 \\ \cdots &= \cdots \\ w_{15} &= b_1\ b_2\ 1\ b_4\ 1\ 1\ 1 \end{aligned}$$

Na primer, za kodno besedo w_3 imamo:

$$\begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \\ 0 \\ b_4 \\ 0 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \quad (mod\ \ 2).$$

Iz česar sledi $b_1 = 1$, $b_2 = 0$ in $b_4 = 0$. Kodna beseda pa je: $w_3 = 1\ 0\ 0\ 0\ 0\ 0\ 1\ 1$.

Poglejmo, kako se da odkrivati napake. Denimo, da eno od kodnih besed pošljemo v kanal in sprejmemo zaporedje v. V splošnem $v \neq w$. Recimo, da je $v = w + \delta$, kjer je δ posledica napak in ima enke na mestih napak. V primeru ene napake ima samo eno enico. Na primer: naj pošljemo $w_3 = 1\ 0\ 0\ 0\ 0\ 0\ 1\ 1$ in zaradi napake na šestem mestu sprejmemo $v = 1\ 0\ 0\ 0\ 0\ 0\ 0\ 1$. Imamo:

$$v = w_3 + \delta = [1\ 0\ 0\ 0\ 0\ 0\ 1\ 1] + [0\ 0\ 0\ 0\ 0\ 0\ 1\ 0].$$

Na sprejemni strani preverimo pogoj parnosti, izračunamo vektor c,

$$c = \mathbf{A}v^T = \mathbf{A}(w + \delta)^T = \mathbf{A}w^T + \mathbf{A}\delta^T = \mathbf{A}\delta^T.$$

V primeru, da je napaka ena sama, vsebuje δ eno samo enico in sicer na mestu napake. Vektor c je v tem primeru kar enak ustreznemu stolpcu matrike $\mathbf{A}$. Za

naš primer:

$$c = \begin{bmatrix} c_1 \\ c_2 \\ c_3 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} \quad (mod\ 2).$$

Mesto napake se da v tem primeru določiti iz vektorja c,
Mesto napake $= c_1 \times 2^2 + c_2 \times 2^1 + c_3 \times 2^0$. Za naš primer:
Mesto napake $= 1 \times 4 + 1 \times 2 + 0 \times 1 = 6$.

Popravljanje napak sestavlja naslednje zaporedje opravil:

- sprejmemo v,
- izračunamo c,
- če je vektor c enak nič, napake ni (ali ni odkrita) in v je enak eni od besed. Odločimo se, da je bila oddana beseda $w_i = v$,
- če pa je vektor c različen od nič, iz njega določimo mesto napake in jo popravimo.

Hammingov kod je sposoben popravljati samo napake na enem simbolu. V primeru, da pride do dveh napak, vsebuje δ dve enici in c je kombinacija dveh stolpcev matrike $\mathbf{A}$. Ker je vsaka kombinacija dveh ali več stolpcev spet enaka nekemu drugemu stolpcu, dobimo za c enako vrednost, kot bi jo dobili v primeru ene same napake.

Obstaja več inačic Hammingovega koda, ki se razlikujejo po dolžini (in številu) kodnih besed in po številu parnostnih simbolov, vsi pa popravljajo enojne napake. Na primer, za dolžine kodnih besed $n = 15, 31, 63, \ldots$ imamo $m = 4, 5, 6, \ldots$ parnostnih simbolov, pripadajoča parnostna matrika pa je velikosti (4×15), (5×31), $(6 \times 63), \ldots$. Odvečnost Hammingovega koda pa je

$$R = 1 - \frac{m}{n}.$$

4.5 Ciklično preverjanje

Ciklično kodiranje ali preverjanje (ang. **C**yclic **R**edundancy **C**ode ali **CRC**) se pri prenosu in pri shranjevanju podatkov največ uporablja, preprosto zato, ker je razmeroma enostavno izvedljivo in neprimerno bolj učinkovito od drugih oblik kodiranja. Ciklični kodi spadajo v veliko družino linearnih blokovnih kodov. Ime ciklični izhaja iz dejstva, da je vsak krožni ('ciklični') pomik kodne besede tudi kodna beseda koda. Ta lastnost za naše potrebe razen zaradi poimenovanja postopka ni zanimiva.

Osnovna zamisel cikličnega preverjanja je preverjanje deljivosti zaporedja informacijskih simbolov z zaporedjem *delilnih simbolov*. Delilno zaporedje simbolov mora biti znano tako oddajni kot sprejemni napravi. Na oddajni strani delimo informacijsko zaporedje simbolov z delilnim zaporedjem, ostanek deljenja pripnemo informacijskim simbolom tako, da je skupno zaporedje deljivo z delilnim zaporedjem brez ostanka. ter vse skupaj pošljemo v kanal. Sprejemna naprava deli sprejeto zaporedje z enakim delilnim zaporedjem in če se deljenje **ne** izide, pomeni to napako. Postopek cikličnega preverjanje je tako popolnoma določen z izbiro zaporedja delilnih simbolov.

Ciklični kod imenujemo tudi polinomski kod (ang. Polynomial Code). To drugo ime izhaja iz dejstva, da se da zaporedje binarnih simbolov obravnavati kot koeficente polinoma potrebne stopnje. Zaporedje binarnih simbolov dolžine $k+1$

$$b_k\ b_{k-1}\ b_{k-2}\ \ldots\ b_i\ \ldots\ b_1\ b_0$$

se da predstaviti s polinomom $P_k(x)$ stopnje k:

$$P_k(x) = b_k x^k + b_{k-1}x^{k-1} + \cdots + b_i x^i + \cdots + b_1 x^1 + b_0.$$

Binarni simboli

$$b_i = \begin{cases} 0 \\ 1 \end{cases} \quad (i = k, k-1, \ldots, 1, 0)$$

določajo koeficiente polinoma. Na primer, zaporedju šestih simbolov 1 1 0 0 1 1 ustreza naslednji polinom pete stopnje:

$$P_5(x) = 1x^5 + 1x^4 + 0x^3 + 0x^2 + 1x^1 + 1 = x^5 + x^4 + x^1 + 1.$$

To dejstvo nam bo v pomoč pri razlagi osnovne zamisli ugotavljanja pravilnosti prenosa s cikličnim preverjanjem. Postopek cikličnega preverjanja bomo torej razložili s predstavitvijo binarnih zaporedij simbolov s polinomi.

Informacijsko zaporedje simbolov dolžine $(k+1)$ predstavimo s polinomom $P_k(x)$ stopnje k. Delilno zaporedje simbolov dolžine $r+1$ predstavimo s polinom $G_r(x)$ stopnje r. Temu polinomu pravimo *generatorjev* polinom. Določanje in preverjanje kontrolnih simbolov obsega naslednje zaporedje korakov:

- zaporedju $(k+1)$ informacijskih simbolov pripišemo r simbolov z vrednostjo nič. Dobljenemu zaporedju pripada polinom $P_k(x) \times x^r$.

- Polinom $P_k(x) \times x^r$ delimo (in sicer po modulu 2) z generatorjevim polinomom. Deljenje se v splošnem ne izide. Dobimo rezultat deljenja $Q(x)$ in ostanek $R(x)$, ki je največ stopnje $(r-1)$:

$$\frac{P_k(x) \times x^r}{G_r(x)} = Q(x) + \frac{R(x)}{G_r(x)}. \tag{29}$$

- Odštejemo (po modulu 2) ostanek deljenja $R(x)$, ki ima vedno največ r simbolov od $P_k(x) \times x^r$. Ostanek deljenja lahko tudi prištejemo, ker je odštevanje po modulu 2 enako seštevanju. Dobimo zaporedje

$$T(x) = P_k(x) \times x^r - R(x) = P_k(x) \times x^r + R(x),$$

  ki je deljivo z $G_r(x)$ brez ostanka. V to se prepričamo, če enačbo (29) množimo z $G_r(x)$. Dobimo

$$P_k(x) \times x^r = Q(x) \times G_r(x) + R(x).$$

  Ko nesemo ostanek $R(x)$ na levo stran, imamo

$$P_{k-1}(x) \times x^r + R(x) = Q(x) \times G_r(x).$$

  Desna stran enačbe vsebuje člen $G_r(x)$ in je gotovo deljiva z njim (brez ostanka). Ker je desna stran enaka levi, mora biti tudi ta deljiva z $G_r(x)$.

- Zaporedje $T(x)$ pošljemo v kanal. Sprejemnik sprejme $T(x)'$, ki zaradi napak med prenosom ni nujno enako $T(x)$,

$$T(x)' = T(x) + E(x).$$

  V polinomu $E(x)$ so zajete napake. Ko napak ni, je $E(x)$ identično nič. Vsaka napaka na simbolu prispeva en člen k polinomu $E(x)$. Na primer, v primeru napake na i-tem simbolu (šteto z desne proti levi, začenši z nič), je $E(x) = x^i$.

- Sprejemnik preveri pogoj deljivosti - deli sprejeto zaporedje $T'(x)$ z $G_r(x)$,

$$\frac{T(x)'}{G_r(x)} = \frac{T(x) + E(x)}{G_r(x)}.$$

  Ker je $T(x)$ deljiv z $G_r(x)$, se deljenje izide, če napake ni $(E(x) = 0)$ ali, če je polinom napak $E(x)$ deljiv z $G_r(x)$ brez ostanka. Takih napak se ne da odkriti.

Primer:

Informacijskemu zaporedju desetih simbolov

$$1\ 1\ 0\ 1\ 0\ 1\ 1\ 0\ 1\ 1$$

bomo določili kontrolne simbole. Naj bo delilno zaporedje

$$1\ 0\ 0\ 1\ 1.$$

Torej ustreza informacijskemu zaporedju polinom

$$P_9(x) = x^9 + x^8 + x^6 + x^4 + x^3 + x^1 + 1$$

in generatorjev polinom je

$$G_4 = x^4 + x^1 + 1.$$

Informacijskemu zaporedju dodamo štiri ničle ($r = 4$) in ga delimo z delilnim zaporedjem. Čeprav iščemo samo ostanek deljenja, bomo računali tudi rezultat deljenja. Računamo po modulu dva:

$$\begin{array}{llllllllllllll}
1\ 1\ 0\ 1\ 0\ 1\ 1\ 0\ 1\ 1\ 0\ 0\ 0\ 0 & : & 1\ 0\ 0\ 1\ 1 & = & 1\ 1\ 0\ 0\ 0\ 0\ 1\ 0\ 1\ 0 \\
\underline{1\ 0\ 0\ 1\ 1} & & & & & & & & & \\
0\ 1\ 0\ 0\ 1\ 1 & & & & & & & & & \\
\quad \underline{1\ 0\ 0\ 1\ 1} & & & & & & & & & \\
\quad 0\ 0\ 0\ 0\ 0\ 1\ 0\ 1\ 1\ 0 & & & & & & & & & \\
\qquad\qquad \underline{1\ 0\ 0\ 1\ 1} & & & & & & & & & \\
\qquad\qquad 0\ 0\ 1\ 0\ 1\ 0\ 0 & & & & & & & & & \\
\qquad\qquad\quad \underline{1\ 0\ 0\ 1\ 1} & & & & & & & & & \\
\qquad\qquad\quad 0\ 0\ 1\ 1\ 1\ 0 & & \text{Ostanek } (1\ 1\ 1\ 0) & & & & & & &
\end{array}$$

V kanal pošljemo zaporedje:

$$1\ 1\ 0\ 1\ 0\ 1\ 1\ 0\ 1\ 1\ 1\ 1\ 1\ 0.$$

Sposobnost odkrivanja napak s cilkličnim preverjanjem je odvisna od generatorjevega polinoma. Poglejmo, kakšne lastnosti mora imeti 'dober' generatorjev polinom.

Zmožnost odkrivanja enojnih napak V primeru enojne napake je $E(x) = x^i$, pri čemer i določa, na katerem simbolu je napaka. Če $G(x)$ vsebuje vsaj dva člena, potem $E(x)$ ne bo deljiv z $G(x)$ in vsaka enojna napaka bo odkrita.

Zmožnost odkrivanja dvojnih napak V primeru dveh napak vsebuje polinom napak $E(x)$ dva člena, $E(x) = x^i + x^j \quad (i > j)$. Ekvivalentno, $E(x) = x^j(x^{i-j}+1) = x^j(x^k+1)$ in k pomeni razdaljo med napakama. Če predpostavimo, da $G(x)$ ni deljiv z x, je zadosten pogoj za odkrivanje dvojnih napak, da $G(x)$ ne deli $x^k + 1$ za vsak k do dolžine sporočila. Polinomi razmeroma nizke stopnje s to lastnostjo so znani. Na primer, polinom $x^{15} + x^{14} + 1$ ne deli $x^k + 1$ vse do $k = 32768$.

Zmožnost odkrivanja lihokratnih napak V primeru lihega števila napak vsebuje polinom $E(x)$ liho število členov. Denimo, da je $E(x) = x^7 + x^2 + 1$ v primeru treh napak. Zanimivo je, da noben polinom z lihim številom členov ne vsebuje faktorja $(x+1)$ in se ga zato nikakor ne da zapisati kot $E(x) = (x+1)F(x)$. Torej ni deljiv z $(x+1)$. V to se prepričamo, če za x vstavimo vrednost 1. Vedno kadar polinom vsebuje liho število členov, je njegova vrednost 1. Na primer:

$$E(1) = 1^7 + 1^2 + 1 = 1 + 1 + 1 = 1 \quad (mod\ 2).$$

V primeru, da bi se $E(x)$ dalo zapisati kot $(x + 1)F(x)$, bi imeli:

$$E(1) = (1 + 1) \times F(1) = 0 \times F(1) = 0 \quad (mod\ 2),$$

kar je protislovno. Torej polinom z lihim številom členov ne vsebuje faktorja $(x + 1)$ in tudi ni deljiv z $(x + 1)$. Če hočemo odkrivati vse lihkokratne napake, moramo izbrati tak generatorjev polinom $G(x)$, ki ne deli nobenega polinoma z lihim številom členov. To dosežemo, če v generatorjev polinom vgradimo člen $(x + 1)$. Na primer, polinom (mednarodno priporočilo CCITT V.41)

$$G(x) = x^{16} + x^{12} + x^5 + 1$$

vsebuje faktor $(x + 1)$.

Zmožnost odkrivanja zaporednih napak Pri prenosu podatkov se napake običajno ne pojavljajo kot napake na osamljenih simbolih, temveč si sledijo na zaporedju sosednjih simbolov. Najpogostejši vzrok napak je impulzni šum. Tipično trajanje impulznega šuma je okrog 10 milisekund in je večinoma posledica preklopov stikal ali drugih elektromehanskih naprav, razelektrenj v ozračju, delovanja motorjev i.t.d. V slišnem področju spoznamo impulzni šum po motečem prasketanju, pri prenosu podatkov s hitrostjo 2400 [b/s] pa tak šum uniči $0.01 \times 2400 \approx 24$ zaporednih bitov. Take 'izbruhe' zaporednih napak je dosti težje odkriti, kot pojav osamljenih napak na posameznih simbolih.

Ciklično preverjanje sporočil je, v nasprotju z drugimi postopki, učinkovito tudi pri odkrivanju zaporednih napak. Hitro se prepričamo, da ciklično preverjanje sporočil s polinom stopnje r omogoča odkrivanje vsakega zaporedja napak

na r ali manj simbolih. Izbruh napak na zaporedju k simbolov se da predstaviti kot polinom

$$x^i(x^{k-1} + \cdots + 1),$$

kjer člen x^i določa samo mesto napak vzdolž sporočila. Če generatorjev polinom vsebuje konstantni člen $x^0 = 1$, potem ne bo v nobenem primeru imel x^i za faktor in ostanek deljenja ne bo nič, dokler je stopnja polinoma v oklepajih nižja od stopnje $G(x)$. Zato:

$$k - 1 < r \Rightarrow k < r + 1 \quad ali \quad k \leq r.$$

V primeru izbruha napak na zaporedju $r + 1$ simbolov, bo ostanek deljenja sporočila z $G(x)$ enak nič samo v primeru, da je $E(x) = x^i G(x)$. Po definiciji napake na zaporedju simbolov dolžine $r+1$ morata biti napačna prvi in zadnji bit v zaporedju bitov dolžine $r + 1$, sicer bomo imeli opravka s krajšim zaporedjem. Da se zaporedje napak ujame z zaporedjem bitov za preverjanje $(G(x))$ je odvisno od $r-1$ vmesnih simbolov. Ob predpostavki, da so vse možne kombinacije napak enako verjetne, je verjetnost neodkrite napake $1/2^{r-1}$.

Naslednji trije generatorjevi polinomi so postali mednarodni standard:

$$\begin{aligned} CRC-12 &= x^{12} + x^{11} + x^3 + x^2 + x^1 + 1 \\ CRC-16 &= x^{16} + x^{15} + x^2 + 1 \\ CRC-CCITT &= x^{16} + x^{12} + x^5 + 1. \end{aligned} \tag{30}$$

Vsi trije vsebujejo faktor $(x + 1)$. Prvi polinom se uporablja pri prenosu šestbitnih podatkov, ostala dva pri prenosu 8-bitnih podatkov. Polinom CRC-16 se uporablja v protokolu IBM BSC, polinom CRC-CCITT je IBM najprej uporabljal v protokolu SDLC, zatem pa ge je CCITT standardiziral. Pri zadnjih dveh polonimoh šestnajste stopnje pripnemo k zaporedju informacijskih bitov zaporedje 16 kontrolnih simbolov. Da se pokazati, da odkrivata vsako enojno ali dvojno napako, vsako lihokratno napako, vsak izbruh napak na 16 zaporednih simbolih ali manj in 99.997 % izbruhov napak na 17 simbolih.

Ciklično preverjanje je v pogledu odkrivanja napak neprimerno učinkovitejše kot preverjanje parnosti, sicer pa se oba precej uporabljata. Preverjanje parnosti je v pogledu realizacije manj zahtevno od cikličnega preverjanja. Da se ga razmeroma enostavno realizirati s programom ali vezjem. Izračun kontrolnih simbolov pri cikličnem preverjanju je bolj zahteven, zato se ga skoraj vedno realizira z materialno opremo. Dobro znana so učinkovita vezja za računanje in preverjanje kontrolnih simbolov s preprostimi pomikalnimi registri. V časovno manj zahtevnih primerih realiziramo ciklično preverjanje s programom za deljenje. Znani pa so tudi učinkoviti algoritmi za določanje kontrolnih bitov s pomočjo pregledovalnih tabel [7].

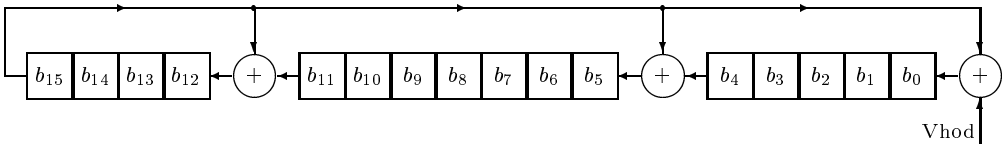
4.5.1 Vezja za ciklično preverjanje

Oglejmo si primer realizacije cikličnega preverjanja s pomikalnim registrom in logičnimi vrati za generatorjev polinom:

$$CRC - CCITT = x^{16} + x^{12} + x^5 + 1.$$

Poenostavljeno vezje je shematično narisano na sliki 82. Naj bodo vrednosti celic pomikalnega registra v začetku postavljene na nič. Seveda bi se lahko odločili za poljubno začetno vrednost pomikalnega registra, važno je le, da je le-ta enaka na oddajni in sprejemni strani. Vezje deluje kot delilnik po pravilih deljenja uporabljenih v prejšnjem primeru 'ročnega' deljenja. Zaporedje bitov (sporočilo) vstopa na sponki 'Vhod' ter se pomika proti levi. Ko prvi simbol z vrednostjo ena prispe v zadnjo celico (b_{15}), se njegova vrednost prenese nazaj in vpliva na vrednosti celic b_{12}, b_5 in b_0 (primerjaj z generatorjevim polinomom). Z drugimi besedami, dokler ne pride enica do konca registra, opravljamo samo pomik in opravimo 'odštevanje', ko pride na konec registra enica. Ko zaporedja informacijskih bitov konča, je potrebnih še šestnajst pomikov z vrednostjo nič na vhodu, da pride ostanek deljenja iz pomikalnega registra.

Za odkrivanje napak je potrebno enako vezje in enak postopek. Na vhodni sponki vstopajo najprej informacijski simboli, za njimi pa še kontrolni simboli. Če je po zadnjem pomiku vsebina vseh celic pomikalnega registra enaka nič, je sporočilo prenešeno brez napake.



Slika 82: Poenostavljena shema vezja za ciklično preverjanje s polinom CRC-CCITT.

Vezja, ki se praktično uporabljajo, so nekoliko drugačna od vezja, na katerem smo razložili delovanje. Ta vezja delujejo sicer po enakem principu, le da pomikanje konča takoj, ko je konec sporočila. Dodatno pomikanje, da dobimo ostanek iz registra, torej ni potrebno.

4.5.2 Ciklično preverjanje s programskim deljenjem

Če želimo realizirati ciklično preverjanje programsko, potem je najlažje napisati podprogram, ki deli dano informacijsko zaporedje z izbranim generatorjem, nekako tako, kot smo pokazali na primeru. Funckijo v jeziku C, ki na oddajni strani generira in na sprejemni strani preverja kontrolne simbole, prikazuje spodnji zapis. Predpostavljamo, da je generator stopnje šestnajst.

```
#include <stdio.h>
#include <stdio.h>
/*-------------------------------------------------------------------------
  Ime:       CrcDeljenje
             Funkcija za deljenje okvirja z generatorjem
  Okvir:     kazalec na polje 8-bitnih podatkov okvirja, vecje za 2 od
             stevila podatkov.
  Generator: bitni vzorec, ki definira generator - brez clena x^16
             npr: CRC-CCITT: 0001 0000 0010 0001 (0x1021).
  Dolzina:   dolzina podatkovnega dela okvirja - brez ostanka (kontrole).
  Vrne:      ostanek deljenja - 16 bitni (2 bajta) CRC ostanek.
             Ostanek doda tudi k okvirju - za podatki.
*/
typedef unsigned short int  ushort;
typedef unsigned char       uchar;

ushort CrcDeljenje( uchar *Okvir, ushort Generator, ushort Dolzina )
{
  ushort TestBit, Ostanek;
  uchar  Podatek;
  int    i, n;

  Ostanek = 0; Okvir[ Dolzina ] = Okvir[ Dolzina + 1 ] = 0;
  for( n = 0; n < Dolzina+2; n++ ){
    Podatek = Okvir[ n ];         /* na vrsti je n-ti bajt okvirja    */
    for( i = 0; i < 8; i++ ){
      TestBit = Ostanek & 0x8000; /* testiramo zgornji bit deljenca   */
      Ostanek <<= 1;              /* predno ga pomaknemo levo         */
      if((Podatek & 0x80)==0x80){ /* dodamo naslednji podatkovni bit  */
        Ostanek |= 1;             /* v primeru, da  je bit = 1        */
      }
      Podatek <<= 1;              /* V vsakem primeru pomik podatka   */
      if( TestBit ){
        Ostanek ^= Generator;     /* ''Odstejemo deljitelj''          */
      }
    }
  }
  Okvir[ Dolzina     ] = Ostanek >> 8;
  Okvir[ Dolzina + 1 ] = Ostanek & 0xff;
  return Ostanek;
} /* Konec CrcDeljenje */
```

4.5.3 Ciklično preverjanje s tabelo ostankov

Bolj učinkoviti algoritmi za ciklično preverjanja temeljijo na pregledovanju (indeksiranju) tabele ostankov (delnega) deljenja. Ostanki deljenja z izbranim generatorjem se najprej in enkrat za vselej izračunajo ter tabelirajo. Tabela in algoritem sta enaka na oddajni in sprejemni strani. V algoritem vstopa sporočilo bajt za bajtom. Za vsak naslednji bajt se v odvisnosti od vrednosti bajta, trenutnega ostanka deljenja in tabele računa nov ostanek deljenja, do konca okvirja.

Zamisel algoritma za ciklično preverjanje s pregledovanjem tabele bomo razložili na podoben način, kot smo razložili osnovno zamisel cikličnega preverjanja.

Naj bo $G_{16}(x)$ generatorjev polinom šestnajste stopnje in $P(x)$ naj tokrat pomeni začetnih nekaj bajtov okvirja. Predpostavimo, da je trenutni ostanek deljenja enak $S(x)$. Torej je

$$P(x)x^{16} = Q(x)G_{16}(x) + S(x).$$

Ostanek $S(x)$ je največ stopnje 15. Dodajmo k $P(x)$ še naslednjih osem bitov – naslednji bajt – okvirja. Podaljšanemu zaporedju potem ustreza polinom $P'(x)$,

$$P'(x) = P(x)x^8 + B(x),$$

pri čemer simbole dodanega bajta upoštevamo s polinomom $B(x)$,

$$B(x) = b_7x^7 + b_6x^6 + b_5x^5 + b_4x^4 + b_3x^3 + b_2x^2 + b_1x^1 + b_0.$$

Naj $S'(x)$ označuje ostanek deljenja podaljšanega polinoma $P'(x)$. Potem je

$$P'(x)x^{16} = Q'(x)G_{16}(x) + S'(x).$$

Z upoštevanjem zveze med $P'(x)$ in $P(x)$ dobimo za levo stran enačbe:

$$\begin{aligned}
[P(x)x^8 + B(x)]x^{16} &= P(x)x^{16}x^8 + B(x)x^{16} = \\
&= [Q(x)G_{16}(x) + S(x)]x^8 + B(x)x^{16} = \\
&= Q(x)G_{16}(x)x^8 + S(x)x^8 + B(x)x^{16} = \\
&= Q(x)G_{16}(x)X^8 + [S(x) + B(x)x^8]x^8.
\end{aligned}$$

Prvi člen je očitno deljiv z $G_{16}(x)$ brez ostanka. Ker nas zanima samo ostanek deljenja, se osredotočimo na zadnji člen. Zapišimo ostanek $S(x)$ kot vsoto dveh polinov sedme stopnje, ki opisujeta zgornji in spodnji bajt ostanka, $S(x) = S_H(x)x^8 + S_L(x)$ in upoštevajmo nove oznake v prejšnjem izrazu. Dobimo:

$$[S(x)] + B(x)x^8]x^8 = S_L(x)x^8 + [S_H(x) + B(x)]x^{16}.$$

Prvi člen je nižje stopnje od generatorja in je zato kar enak ostanku deljenja z generatorjem. Nov ostanek deljenja $S'(x)$ je:

$$S'(x) = S_L(x)x^8 + Ostanek\{[S_H(x) + B(x)]x^{16}]\}$$

Iz zadnjega izraza sledi, da se da izračunati ostanek sporočila podaljšanega za en dodani bajt iz spodnjega bajta prejšnjega ostanka in ostanka deljenja seštevka zgornjega bajta prejšnjega ostanka in dodanega bajta, $V(x)x^{16} = [S_H(x) + B(x)]x^{16}$. Obstaja samo 256 možnih kombinacij člena $V(x)$, za katere se da ostanek vnaprej izračunati in tabelirati. Rezultat je tabela 256 šestnajstbitnih ostankov deljenja, ki je odvisna samo od izbire generatorja.

Algoritem za ciklično preverjanje, ki temelji na pregledovanju tabele ostankov T, je na primer tak:

1. postavi začetni ostanek na nič, $S = 0$,
2. prištej po modulu 2 naslednji bajt sporočila k zgornjemu bajtu ostanka, $V = S_H + B$. Vrednost za V je indeks v tabelo ostankov T, vrednost ostanka je $T[V]$.
3. pomakni S za osem mest levo. S tem se zgornji del S_H izgubi. Rezultat je S_L pomaknjen za osem mest levo.
4. prištej po modulu 2 vrednost ostanka $T[V]$ k S, $S = S + T[V]$.
5. ponavljaj korake 2-4 do konca okvirja.

Naslednja funkcija izračuna tabelo vseh možnih ostankov deljenja z generatorjem CRC-CCITT. Pri deljenju uporablja funkcijo `CrcDeljenje`.

```
/*---------------------------------------------------------------------------
  Ime:       GenCrcTabelo
             Funkcija za generiranje tabele za dan generator. Tabela
             se izpi\ss e tudi na zaslon.
  CrcTabela: kazalec na tabelo ostankov - velikosti 256, ki jih
             izra\cc una (vrne) funkcija.
  Generator: bitni vzorec, ki definira generator - brez clena x^16
             npr: CRC-CCITT: 0001 0000 0010 0001 (0x1021).
*/

void GenCrcTabelo( ushort *CrcTabela, ushort Generator )
{
  int    Index;
  uchar  Temp[5];
  ushort CrcDeljenje( uchar *, ushort, ushort );

  for( Index = 0; Index < 256; Index++ ){
    Temp[ 0 ] = Index;  Temp[ 1 ] = Temp[ 2 ] = 0;
    CrcTabela[ Index ] = CrcDeljenje( Temp, Generator, 1 );
    if( (Index % 16) == 0) printf("\n");
    printf("%4x ", CrcTabela[ Index ] );
  }
  printf("\n");
} /* Konec GenCrcTabelo */
```

Tabela, ki jo izpiše zgornja funkcija, izgleda takole:

```
0000 1021 2042 3063 4084 50a5 60c6 70e7 8108 9129 a14a b16b c18c d1ad e1ce f1ef
1231 0210 3273 2252 52b5 4294 72f7 62d6 9339 8318 b37b a35a d3bd c39c f3ff e3de
2462 3443 0420 1401 64e6 74c7 44a4 5485 a56a b54b 8528 9509 e5ee f5cf c5ac d58d
3653 2672 1611 0630 76d7 66f6 5695 46b4 b75b a77a 9719 8738 f7df e7fe d79d c7bc
48c4 58e5 6886 78a7 0840 1861 2802 3823 c9cc d9ed e98e f9af 8948 9969 a90a b92b
5af5 4ad4 7ab7 6a96 1a71 0a50 3a33 2a12 dbfd cbdc fbbf eb9e 9b79 8b58 bb3b ab1a
6ca6 7c87 4ce4 5cc5 2c22 3c03 0c60 1c41 edae fd8f cdec ddcd ad2a bd0b 8d68 9d49
7e97 6eb6 5ed5 4ef4 3e13 2e32 1e51 0e70 ff9f efbe dfdd cffc bf1b af3a 9f59 8f78
9188 81a9 b1ca a1eb d10c c12d f14e e16f 1080 00a1 30c2 20e3 5004 4025 7046 6067
83b9 9398 a3fb b3da c33d d31c e37f f35e 02b1 1290 22f3 32d2 4235 5214 6277 7256
b5ea a5cb 95a8 8589 f56e e54f d52c c50d 34e2 24c3 14a0 0481 7466 6447 5424 4405
a7db b7fa 8799 97b8 e75f f77e c71d d73c 26d3 36f2 0691 16b0 6657 7676 4615 5634
d94c c96d f90e e92f 99c8 89e9 b98a a9ab 5844 4865 7806 6827 18c0 08e1 3882 28a3
cb7d db5c eb3f fb1e 8bf9 9bd8 abbb bb9a 4a75 5a54 6a37 7a16 0af1 1ad0 2ab3 3a92
fd2e ed0f dd6c cd4d bdaa ad8b 9de8 8dc9 7c26 6c07 5c64 4c45 3ca2 2c83 1ce0 0cc1
ef1f ff3e cf5d df7c af9b bfba 8fd9 9ff8 6e17 7e36 4e55 5e74 2e93 3eb2 0ed1 1ef0
```

Računanje CRC stanka s pregledovanjem prej izračunane tabele prikazuje spodnji program.

```
/*---------------------------------------------------------------------------
  Ime:       CrcPoBajtih
             Funkcija za CRC preverjanje s tabelo

  Okvir:     kazalec na polje 8-bitnih podatkov okvirja, vecje za 2 od
             stevila podatkov (torej Dolzina+2).

  CrcTabela: Kazalec na tabelo 256 ostankov za CRC-CCITT

  Dolzina:   dolzina podatkovnega dela okvirja.
  Vrne:      ostanek deljenja - 16 bitni (2 bajta) CRC ostanek
             Ostanek doda tudi k okvirju - za podatki.
*/

ushort CrcPoBajtih( uchar *Okvir, ushort *CrcTabela, ushort Dolzina)
{
  ushort Ostanek, Podatek;
  int    i;

  for( Ostanek = 0, i = 0; i < Dolzina; i++ ){
    Podatek = (ushort)Okvir[ i ];
    Ostanek = (Ostanek << 8)^CrcTabela[ (Ostanek>>8)^Podatek];
  }
  Okvir[ Dolzina     ] = Ostanek >> 8;
  Okvir[ Dolzina + 1 ] = Ostanek & 0xff;
  return Ostanek;
}
```

Primer preizkusnega programa, ki za dano zaporedje podatkov izračuna CRC ostanek na oba načina (s preprostim deljenjem in s pregledovanjem tabele), je zapisan spodaj.

```
/* Preizkusni program */

int
main( int argc, char **argv )
{
   ushort Polinom = (ushort)0x1021;
   ushort CrcTabela[256];

   ushort Ostanek;
   int    i;

   uchar Okvir[8] = { 1, 2, 3, 4, 5, 6, 0, 0 };

   /* CRC z deljenjem */
   Ostanek = CrcDeljenje( Okvir, Polinom, 6 );
   for( i = 0; i < 8; i++){
     printf("Okvir[ %2d ] = %x\n", i, Okvir[ i ] );
   }
   printf( "Ostanek = %x\n", Ostanek );

   /* CRC s tabelo  */
   GenCrcTabelo( CrcTabela, Polinom );
   Ostanek = CrcPoBajtih( Okvir, CrcTabela, 6 );
   for( i = 0; i < 8; i++){
     printf("Okvir[ %2d ] = %x\n", i, Okvir[ i ] );
   }
   printf( "Ostanek = %x\n", Ostanek );

   return 0;
}
```

4.5.4 Ciklično preverjanje z reducirano tabelo ostankov

S pomočjo tabele ostankov prihranimo na času, ki je potreben za izračun CRC ostanka, na račun pomnilnika, ki je potreben za shranjevanje tabele. Algoritem, ki zmanjša pomnilniški prostor za shranjevanje tabele na račun nekaj več računanja, upošteva naslednjo lastnost ostanka:

$$\begin{aligned} O\{V(x)x^{16}\} &= O(b_7x^{23} + b_6x^{22} + \cdots + b_0x^{16}) \\ &= O(b_7x^{23}) + O(b_6x^{22}) + \cdots + O(b_0x^{16}) \end{aligned} \tag{31}$$

in členi $b_i, (i = 0, 1, \ldots, 7)$ so lahko 0 ali 1. Za vsak člen b_i izračunamo ostanek deljenja z izbranim generatorjem. Dobimo osem 16-bitnih "delnih" ostankov, ki jih tabeliramo. Končni ostanek izračunamo tako, ta seštejemo tiste delne ostanke, za katere je $b_i = 1$. V povprečju so torej potrebna štiri dodatna seštevanja, potrebujemo pa osemkrat manj pomnilnika za shranjevanje tabele.

4.6 Primeri protokolov podatkovnega sloja

4.6.1 BSC

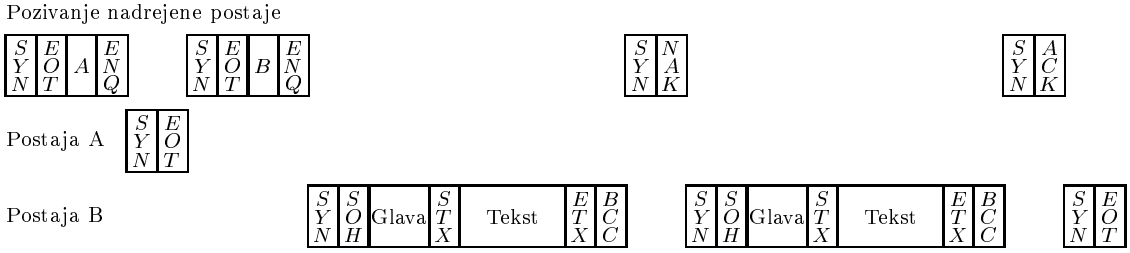
Protokol BSC (Binary Sinchronous Communications) ali z drugim imenom Bysinc je znakovno usmerjen protokol za sinhroni prenos. Predvideva IBM EBCDIC (Extended Binary Coded Decimal Interchange Code) kodiranje znakov, možna pa je tudi uporaba ASCII kodiranja. Protokol so razvili pri IBM-u in je v uporabi že od leta 1968. "BSC kompatibilnost" ali "emulacija" BSC protokola s strani drugih proizvajalcev komunikacijske opreme je bila včasih praktično obvezna. Konec sedemdesetih let so ga začeli izrivati bitno usmerjeni protokoli, recimo SDLC. BSC protokol tu omenjamo predvsem zato, ker se da njegove sledi marsikje opaziti tudi še danes.

Za nadzor komunikacije so izbrani kontrolni znaki: SYN (Synchronization), SOH (Start Of Header), STX (STart of teXt), ETX (End of TeXt), ETB (End of Block), EOT (End Of Transmission), NAK (Negative Acknowledge), DLE (Data Link Escape), ENQ (Inquiry), in še nakaj drugih. BSC okvir je deljen na več polj, od katerih sta glava in podatkovni del spremenljive dolžine. Oblika okvirja je skicirana na sliki 83.

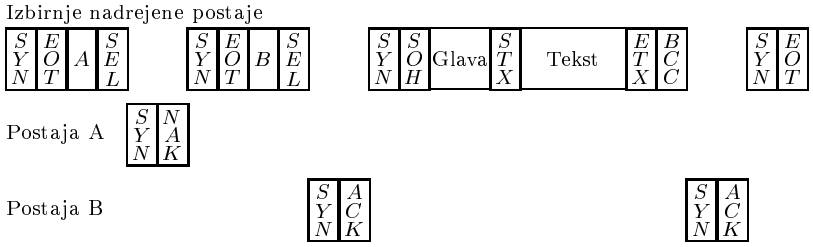
Slika 83: Oblika okvirja protokola BSC.

Okvir začne s sinhronizacijskima znakoma SYN SYN, z neobvezno "glavo" (ang. Header), ki začne s SOH in konča s STX. STX hkrati napove začetek podatkovnega dela okvirja. Podatkovni del je poljubne dolžine in konča z ETB za vmesni ali z ETX za zadnji okvir sporočila. Možno je ali vzdolžno in prečno preverjanje parnosti ali ciklično preverjanje s šestnajstimi biti (polje BCC). Transparenten prenos podatkov se zagotavlja z napovedovanjem kontrolnih znakov. Za napovedni znak je izbran DLE. Pravilen prenos okvirjev se izmenično potrjuje z ACK0 in ACK1. Ponoven prenos se zahteva z NAK. Znak EOT zaključi komunikacijo in postavi protokol v nadzorni način. BSC protokol se lahko nahaja v enem od dveh načinov, v kontrolnem ali v tekstovnem. V tekstovnem načinu se prenašajo podatki, v kontrolnem pa se opravljajo nadzorne funkcije kanala, kot je recimo vzpostavljanje zveze. V večtočkovnem ali zvezdastem omrežju je možno vzpostavljanje zveze s pozivanjem (ang. Polling) ali izbiranjem (ang. Selecting). Slika 84 ponazarja poenostavljen način pozivanja. Pri pozivanju nadrejena postaja krožno poziva oziroma naslavlja podrejene postaje. Poziv pomeni povabilo na oddajo. Če podrejena postaja nima pripravljenih podatkov, odgovori odklonilno z NAK. Sicer začne z oddajo podatkovnega okvirja. Ko odda zadnji

okvir pošlje EOT in s tem je zveza prekinjena. V primeru, da želi nadrejena postaja poslati podatke podrejeni postaji, jo "izbere". Prenos podatkov začne samo, če se izbrana postaja odzove pritrdilno in traja do oddaje EOT. Poenostavljen primer pozivanja je skiciran na sliki 85.



Slika 84: Vzpostavljanje zveze s pozivanjem. Nadrejena postaja najprej pozove postajo A, ki se odzove odklonilno. Za njo pozove postajo B, ki v odgovor pošlje podatke. Ker je prvi prenos okvirja neuspešen, ga nadrejena postaja zavrne z NAK. Ponovljeni prenos je uspešen, kar nadrejena postaja potrdi z ACK. Prenos konča podrejena postaja z EOT.



Slika 85: Vzpostavljanje zveze z izbiranjem. Nadrejena postaja najprej izbere postajo A (ukaz SEL), ki se odzove odklonilno z NAK. Za njo izbere postajo B, ki se odzove pritrdilno z ACK. Sledi oddaja okvirja, ki ga izrana postaja potrdi z ACK. Nadrejena postaja zaključi komunikacijo z oddajo EOT.

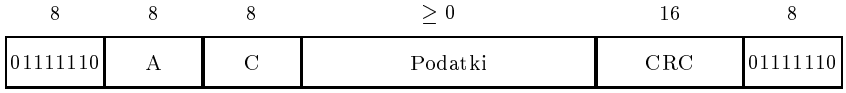
4.6.2 SDLC, HDLC in LAPB

Protokol SDLC (Synchronous Data Link Control) so razvili pri IBM-u v začetku sedemdesetih let (1974) za svojo arhitekturo omrežja SNA (System Network Architecture). IBM je specifikacijo protokola poslal v standardizacijo organizaciji ANSI, da bi ga potrdila kot ameriški standard in organizaciji ISO, da bi ga potrdila za mednarodni standard. Nobena ga ni potrdila brez sprememb. Tako sta brez večjih in bistvenih razlik nastala dva uradna protokola: ANSI ADCCP (Advanced Data Communication Control Procedure) in ISO HDLC (High-Level Data Link Control), ki ga je CCITT potem za svoje potrebe v omrežjih X.25

priredil in izdal pod imenom LAP (Link Access Procedure) ter nato v dopolnjeni obliki pod imenom LAPB (LAP-Balanced) (slika 86).

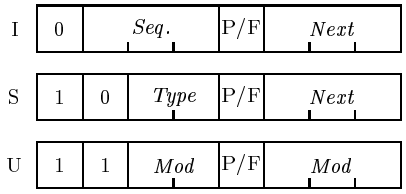
Slika 86: Razvoj protokolov podatkovnega sloja.

Vsi omenjeni protokoli so bitno usmerjeni in uporabljajo enako obliko okvirja, razlikujejo se samo v podrobnostih nadzornega polja, glej skico 87. Najkrajši ovir ne nosi podatkov in obsega 32 bitov (naslov + nadzor + preverjanje). Maksimalna dolžina podatkovnega dela ni predpisana.



Slika 87: Oblika okvirja protokola SDLC in njegovih naslednikov.

Za preverjanje okvirja se koristi polinom CRC-CCITT. Naslovni del (A) pri povezavi dveh vozlišč točka-točka nima pomena, pomemben je predvsem pri pozivanju podrejenih (ang. Slave ali Tributary) vozlišč v večtočkovnih ali zvezdastih omrežjih. Nadzorni del (C) razlikuje podatkovne od nadzornih okvirjev, služi za številčenje podatkovnih okvirjev in potrdil ter drugim nadzornim funkcijam. Oblika nadzornega dela je prikazana na sliki 88. Bit P (Poll/Final) uporablja nadrejeno vozlišče (ang. Master) za pošiljanje ukaza podrejenemu vozlišču, podrejeno vozlišče pa postavi bit F za javljanje odziva na ukaz.



Slika 88: Oblika nadzornega dela okvirja SDLC.

Možno je selektivno ponavljanje (SRP) okvirjev ali vračnaje nazaj na N (GBN) s pozitivnim in z negativnim potrjevanjem ter z drsečim oknom. Številčenje okvirjev in potrdil je po modulu osem s tribitno številko. Protokola HDLC in LAPB omogočata prehod na 7-bitno številčenje.

Obstajajo tri vrste okvirjev: I (information), S (Supervisory) in U (Unnumbered). Podatkovni okvir (I) nosi podatke in *Seq.* je tedaj zaporedna številka

okvirja. S podatkovnimi okvirji se lahko prenaša potrdila podatkovnih okvirjev, ki gredo v nasprotni smeri. V tem primeru je *Next* zaporedna številka potrdila oziroma številka naslednjega okvirja, ki ga pričakuje sprejemnik. Na primer, če se pokvari okvir s številko tri, bo sprejemnik zahteval ponoven prenos okvirja s številko tri (Next = 3). Možni so štirje tipi nadzornih (S) okvirjev: 0,1,2 in 3, ki jih kodira polje *Type*:

- Okvir tipa 0 je pozitivno potrdilo (ang. Receive Ready) in služi za potrjevanje okvirjev, kadar ni podatkovnih okvirjev v nasprotni smeri in *Next* je številka naslednjega pričakovanega okvirja.
- Okvir tipa 1 je negativno potrdilo (ang. Reject) za GBN način in *Next* je številka poškodovanega okvirja.
- Z okvirjem tipa 2 sprejemnik javlja, da sprejemnik ni pripravljen (ang. Receivenot Ready) in prosi za premor, dokler ne pošlje okvir tipa 0 ali 1.
- Tip 3 okvirja je veljaven samo v protokolih HDLC in ADCCP. Služi za selektivno zavrnitev (ang. Selective Reject) in torej potrjevanje SRP.

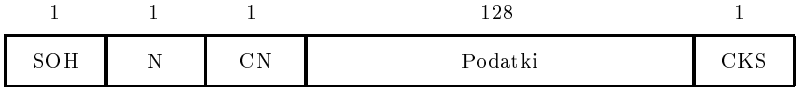
Protokoli se med seboj še najbolj razlikujejo v pogledu neštevilčenih (U) okvirjev. Vsi poznajo ukaz DISC (ang. Disconnect) za javljanje neaktivnosti vozlišča (izven obratovanja), vračanje nazaj v obratovanje SNRM (Set Normal Response Mode), in okvir UA (ang. Unnumbered Acknowledgement) za potrjevanje neštevilčenih okvirjev.

4.6.3 XMODEM

Eden najbolj priljubljenih protokolov za asinhrone prenose podatkov z ali brez modemov med manjšimi računalniki je protokol imenovan XMODEM. Razvil ga je Ward Christensen v poznih sedemdesetih letih, protokol pa je kasneje doživljal spremembe in dopolnila (MODEM2, MODEM7, YMODEM, ZMODEM). Oblika okvirja je skicirana na sliki 89. Protokol je razmeroma primitiven. Uvodni znak je ASCII SOH (binarno 00000001) (Start O Header), njemu sledi osembitna zaporedna številka okvirja in zaradi zanesljivosti njen eniški komplement. Okvirji so na žalost nespremenljive dolžine in nosiljo 128 podatkovnih bajtov. Podatkom sledi osembitna kontrolna vsota podatkovnega dela okvirja. Protokol XMODEM uporablja potrjevanje s čakanjem. Pobuda za prenos pride od sprejemnika, ki periodično vsakih nekaj (deset ali petnajst) sekund oddaja ASCII znak NAK in vabi oddajnik k oddaji. Različica protokola XMODEM-CRC uporablja ciklično preverjenje s polinomom CRC-CCITT. Sprejemnik se v tem primeru javlja z ASCII znakom C. Različica YMODEM pa poleg 128 podatkovnih bajtov dovoljuje tudi okvirje s 1024 podatki. Protokol ZMODEM je ena zadnjih različic protokola

XMODEM, ki nima z njim praktično skoraj nič skupnega, razen da omogoča XMODEM način prenosa.

Oddajnik na povabilo sprejemnika oddaja okvirje dokler sprejemnik ne prekine prenosa z oddajo ASCII znaka CAN (Cancel) ali pa potrdi regularen zaključek. Prenos teče nakako takole. Oddajnik odda okvir. Sprejemnik preveri zaporedno številko. V primeru napake zahteva prekinitev prenašanja z CAN. Nato preveri parnost. Če je pogoj parnosti izpolnjen, pošlje ACK, sicer pa NAK. Ko oddajnik konča z oddajanjem, pošlje EOT in sprejemnik potrdi zaključek z oddajo ACK.

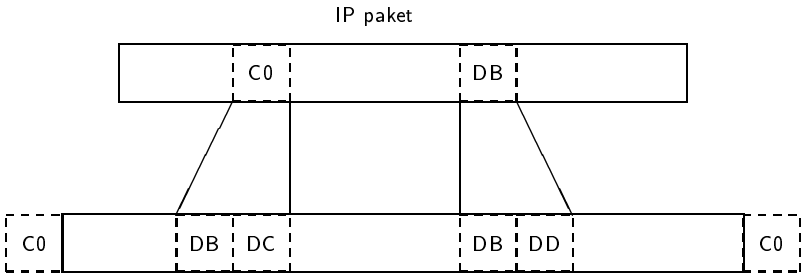


Slika 89: Oblika okvirja protokola XMODEM. Okvir začne z ASCII SOH, zaporedno številko okvirja (N) in njenim eniškim komplementom (CN), vsebuje 128 bajtov podatkov in konča s kontrolno vsoto podatkov.

4.6.4 SLIP

Protokol SLIP (Ang. Serial Link IP) služi za prenos IP paketov po serijskih linijah (točka – točka), na primer za priključitev osebnega računalnika od doma na Internet. V zadnjem času ga je izpodrinil protokol PPP (Point to Point protocol). SLIP je specificiran v dokumentu RFC 1055. Protokol SLIP ne opravlja preverjanja pravilnosti prenosa, to je naloga višjih slojev. Slika 90 prikazuje obliko SLIP okvirja. Okvir konča s posebnim končnim znakom, imenovanim END znak. END znak ima vrednost $C0. Ta isti znak se uporablja tudi na začetku okvirja. Znotaj okvirja se prenaša paket mrežnega sloja, ta je v omrežju Internet IP paket. Da bi sprejemnik podatka $C0 znotraj IP paketa ne tolmačil napačno kot konec okvirja, se znaki s to vrednostjo zamenjajo s kombinacijo SLIP ESC zanka, ki ima vrednost $DB in znaka $DC. Ko sprejemnik med sprejemanjem okvirja naleti na zaporedje $DB $DC ga tolmači kot en sam podatek $C0. Da ne bi prišlo do zamenjave "pravega" para podatkov $DB $DC, se podatek $DB nadomesti s kombinacijo $DB $DD.

Kot vidimo SLIP nima polja, ki bi nosil adreso ali ki bi označeval tip okvirja, niti ne omogoča številčenje okvirjev. Nadzor na nivoju podatkovnega protokola zato ni možen, potreben je torej na višjih slojih. SLIP opravlja zgolj okvirjenje IP paketov in s tem omogoča transparenten prenos IP paketov po serijskih linijah.



Slika 90: Oblika okvirja protokola SLIP.

Literatura

[1] R. Ash, *Information Theory*, John Wiley & Sons, 1965.

[2] R. Cole, *Computer Communications*, Springer-Verlag 1982.

[3] F. da Cruz, B. Catchings, "Kermit: A File-Transfer Protocol for Universities", Part I, Part II, BYTE, Jun. in Jul. 1984.

[4] L. Gyergyek, *Teorija informacij*, ZAFER, Ljubljana 1988.

[5] T. Madron, *Micro-Mainframe Connection*, HOWARD W.SAMS & COMPANY, 1987.

[6] N. Pavešić, *Informacija in kodi*, ZAFER, Ljubljana 1997.

[7] T. Ramabadran, S. Gaitonde, "A Tutorial on CRC Computations", IEEE Micro, Aug. 1988, pp. 62-74.

[8] W. R. Stevens, *TCP/IP Illustrated, Vol. 1*, Addison–Wesley, 1994.

[9] A. Tanenbaum, *Computer Networks*, Prentice-Hall 1989.

[10] J. Walrand, *Computer Communications: A First Course*, Pacific Palisades, CA: Asken Associates, 1991.

5 Dostop do prenosnega sredstva in lokalna omrežja

Lokalna omrežja (Ang. **L**ocal **A**rea **N**etworks - **LAN**) so se pojavila v začetku sedemdesetih iz potrebe po povezovanju manjših računalniških sistemov. Lokalno omrežje spoznamo predvsem po omejenem geografskem področje (od nekaj 10 metrov do nekaj kilometrov), relativno ceneni tehnologij komunikacijskih naprav in prenosnih poti, po razmeroma visoki zanesljivosti prenosa podatkov, ki je vsaj 1000-krat višja od zanesljivosti prenosa v omrežjih velikih geografskih razsežnosti (Ang. **W**ide **A**rea **N**etwork - **WAN**) in po visoki hitrosti prenosa, ki je vsaj 1 Mb/s, lahko pa dosega tudi 100 Mb/s in več. Uporabnik lokalnega omrežja je največkrat tudi lastnik omrežja.

Lokalna omrežja omogočajo delitev sredstev (strojne in programske opreme, procesne moči in podatkovnih zbirk) med večje število uporabnikov neodvisno od krajevne namestitve sredstev, lažje in cenejše vzdrževanje programske in strojne opreme, omogočajo povečanje zanesljivosti delovanja s podvajanjem kritičnih sredstev sistema, dajejo celovit pregled nad zbiranjem, obdelavo, izmenjavo ter shranjevanjem podatkov in nudijo centralen nadzor nad stanjem sistema v celoti. Lokalna računalniška omrežja se med seboj tipično razlikujejo

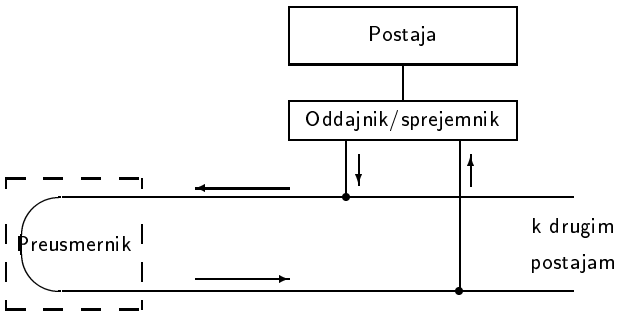
- glede na njihovo fizično ali logično obliko (topologijo),
- glede na način dostopa do skupnega prenosnega medija (ang. Media Access),
- glede na frekvenčni pas prenašanja,
- glede na izvedbo prenosne poti,
- po svoji arhitekturi, po izvedbi, po proizvajalcu ter po upoštevanju standardov in/ali priporočil.

Računalnik ali vsako drugo komunikacijsko napravo, ki je neposredno vezana na omrežje, imenujemo *komunikacijsko vozlišče* (Ang. node). V omrežjih velikih geografskih razsežnosti opravljajo vozlišča samo komunikacijske naloge. Na eno tako vozlišče je možno povezati večje število avtonomnih računalniških sistemov ali postaj (Ang. Host, End-station, Work-station, Terminal-station in podobno). Vozlišča so med seboj povezana v komunikacijski podsistem (Ang. Communication Subnet). V lokalnih računalniških omrežjih pa je postaja hkrati tudi vozlišče. Oziroma, lahko bi rekli, da je vozlišče del postaje same. Lokalna omrežja srečamo v vseh osnovnih oblikah: v topologiji zvezda, obroč ali vodilo ali drevo, možne pa so tudi kombinacije naštetih oblik.

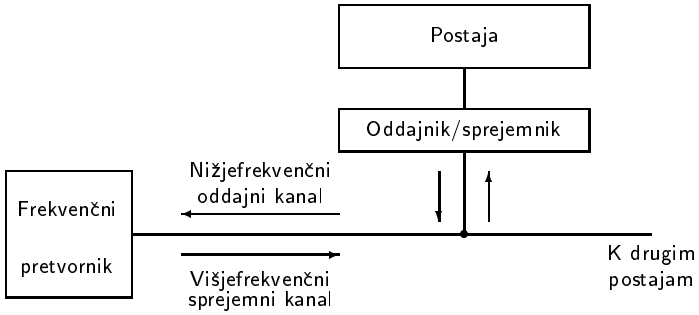
Dostop do komunikacijskega kanala je lahko centraliziran ali decentraliziran. Pri *centraliziranem* (osrednjem) nadzoru dostopa je dodeljevanje skupnega kanala posameznim postajam (ali vzpostavljanje, vzdrževanje in sproščanje zveze) naloga ene od postaj v omrežju. Vzpostavljanje zveze s pozivanjem (ang. Polling) in izbiranjem (ang. Selecting) sta tipična predstavnika centraliziranega nadzora. V sistemih daljinskega vodenja se uporabljata v zvezdastih in večtočkovnih omrežjih. Pri pozivanju osrednja ali nadrejena postaja (ang. Master Station) krožno (drugo za drugo) poziva zunanje ali podrejene postaje (ang. Slave ali Tributary Stations). Če pozvana (naslovljena) postaja ima pripravljene podatke, se odzove pritrdilno ali v odgovor kar takoj pošlje podatke. V primeru, da postaja nima podatkov, se odzove odklonilno in nadzorna postaja pozove naslednjo postajo. Kadar osrednja postaja želi poslati podatke zunanji postaji, jo naslovi-izbere, in če je zunaja postaja pripravljena na sprejem, ji pošlje podatke. V lokalnih omrežjih so znani učinkovitejši mehanizmi dodeljevanja kanala, zato se pozivanje in izbiranje v njih malo uporabljata. Vendar je mešano omrežje (večtočkovno in zvezdasto omrežje) s pozivanjem tipičen primer komunikacije v sistemih daljinskega vodenja. Pri decentraliziranem (porazdeljenem) nadzoru je nadzor dostopa do kanala porazdeljen med vse postaje v omrežju. Prednost porazdeljenega nadzora je manjša občutljivost na okvare (če izpade nadzorna postaja, razpade celotno omrežje) in svobodnost pri spreminjanju omrežja (dodajanju/odvzemanju postaj). Porazdeljen nadzor se v lokalnih omrežjih več uporablja.

Glede na dostop do prenosnega medija (kanala) ločimo omrežja z naključnim dostopom in omrežja s predvidljivim (determinističnim) dostopom. V prvem primeru je trenutek dodelitve kanala neki postaji vnaprej nemogoče zagotovo napovedati. Lahko se zgodi, da neko postajo neprestano prehitevajo druge postaje. Teoretično je možno, čeprav malo verjetno, da postaja z oddajo sploh ne pride na vrsto. Takšen način dodeljevanja je v časovno kritičnih sistemih (nekaterih industrijskih sistemih, ki morajo delovati v stvarnem času), manj primeren ali celo neprimeren. Dobro se obnese v neobremenjenih omrežjih. Lokana omrežja s predvidljivim dostopom do kanala nimajo te slabosti. Dostop do kanala je bodisi urejen centralno s pozivanjem in izbiranjem ali pa je dostop do kanala urejen s posebnim bitnim vzorcem, imenovanim žeton (Ang. token), ki potuje od postaje do postaje. Ko postaja dobi žeton, postane kanal njen in začeti sme z oddajo. Ko konča z oddajo, preda žeton naslednji postaji. Najdaljši čas čakanja na kanal se da vnaprej napovedati za vsako postajo.

Glede na frekvenčni pas kanala ločimo omrežja z informacijskim signalom v osnovnem frekvenčnem pasu (Ang. Baseband) in omrežja z informacijskim signalom v višjem frekvenčnem pasu (Ang. Broadband). V prvem primeru je informacijski signal nemoduliran in informacijski kanal zavzema celotno razpoložljivo frekvenčno širino prenosnega medija (od 0 Hz naprej). Večina lokalnih omrežij je te vrste. Glavni razlog za gradnjo takšnih omrežij je nižja cena izvedbe, spreminjanja in vzdrževanja omrežja. Omrežja z informacijskim signalom v višjem



Slika 91: Dvokabelska izvedba omrežja tipa vodilo s signalom v visokofrekvenčnem pasu.



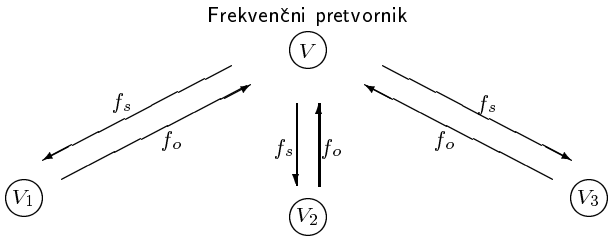
Slika 92: Enokabelska izvedba omrežja tipa vodilo s signalom v visokofrekvenčnem pasu.

frekvenčnem pasu zahtevajo visokofrekvenčne modulatorje in demodulatorje (visokofrekvenčne MoDeme), ki premaknejo informacijski signal v višji frekvenčni pas. To podraži izvedbo in vzdrževanje omrežja. Tehnologija tovrstnih mrež je prevzeta od kabelske televizije (KTV). Cenovno ugodnejša se izkaže tedaj, kadar se isti prenosni medij (kabel) koristi večnamensko (na primer za prenos podatkov, govora in za prenos TV slike). Obstajata dve izvedbi omrežij tipa vodilo, ki za prenos podatkov izkoriščajo višje frekvenčno področje, in sicer: enokabelska izvedba in dvokabelska izvedba. Pri dvokabelski izvedbi so vsi oddajniki postaj vezani na enega od obeh kablov (oddajni kanal), vsi sprejemniki postaj pa na drugi kabel (sprejemni kanal), kot je narisano na sliki 91.

V enokabelski izvedbi imamo namesto dveh kablov en sam kabel, sicer pa so razmere podobne kot v dvokabelski izvedbi. Oddajni in sprejemni kanal sta frekvenčno multipleksirana na skupni kabel, kot je prikazano na sliki 92.

Vse postaje oddajajo v nižjem frekvenčnem pasu (oddajni kanal) in sprejemajo v višjem frekvenčnem pasu (sprejemni kanal). Frekvenčni premik signala iz nižjega v višji frekvenčni pas opravlja posebna naprava na koncu kabla (Ang. Head-End).

Glede na uporabljeni prenosni medij spadajo lokalna omrežja v tri glavne kate-



Slika 93: Omrežje ALOHA. Postaje oddajajo na frekvenci $f_o = 407\ MHz$ in sprejemajo na frekvenci $f_s = 419\ MHz$..

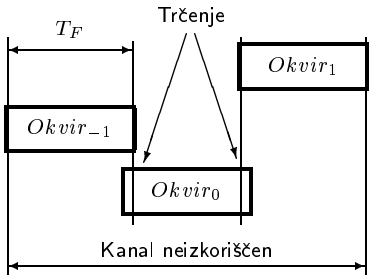
gorije. Omrežja, ki uporabljajo: koaksialni kabel, optična vlakna (Ang. Optical fibers) ali parico (Ang. Twisted pair). Največ se uporablja koaksialni kabel, tako za prenos v osnovnem kot tudi v višjem frekvenčnem področju. Je odporen na zunaje elektromagnetne vplive in ima dobre frekvenčne karakteristike. Najcenejša je običajna telefonska parica. Zunaje vplive se zmanjša s prepletanjem. Ker sta oba vodnika enako izpostavljeni elektomagnetnim vplivom, se vpliv motenj medsebojno uničuje. Parica se uporablja v zvezdastih omrežjih za povezavo postaj točka-točka in tudi za večtočkovno povezovanje. Optična prenosna sredstva vse bolj nadomeščajo kovinske vodnike. Omogoča višje hitrosti prenosa in daljše razdalje. Prenosna zmogljivost optičnega vlakna znaša do 100 GHz · Km. Odlikuje jih nizko prenosno slabljenje (pod 0.5 dB/Km), majhne prečne dimenzije (premer obloge 125 μ m), majhna teža, imunost na zunanje elektromagnetne motnje in sklapljanje (presluh). Zaradi zahtevnega priključevanja so vlakna primerna predvsem za omrežja zvezdaste ali zankaste oblike (povezave točka-točka).

Najbolj značilen za lokalna omrežja je način dostopa do skupnega prenosnega medija. V nadaljevanju se bomo posvetili predvsem načinom dostopa večjega števila postaj do skupnega kanala in mednarodnim standardom s tem v zvezi.

5.1 Protokoli tipa ALOHA

V sedemdesetih letih je Norman Abramson s sodelavci na *University of Hawaii* izdelal nov način dodeljevanja skupnega kanala večjemu številu postaj. Abramson je imenoval svoj sistem ALOHA (slika 93).

V tem sistemu je za komunikacijski kanal sicer izkoristil radijske zveze, vendar je način dodeljevanja kanala primeren za vsak informacijski kanal z množičnim dostopom. Omrežje ALOHA je delovalo s hitrostjo 9600 bitov na sekundo. Vse postaje so oddajle na enem frekvenčnem pasu in sprejemale na drugem frekvenčnem pasu. Frekvenčno pretvorbo je opravljala ena od postaj. Kar je oddala ena postajo so "slišale" vse postaje. Po Abramsonovem vzoru so kasneje razvili številne različice množičenga dostopa do skupnega kanala tipa ALOHA.



Slika 94: Vpliv trčenj na neizkoriščenost kanala v čistem ALOHA.

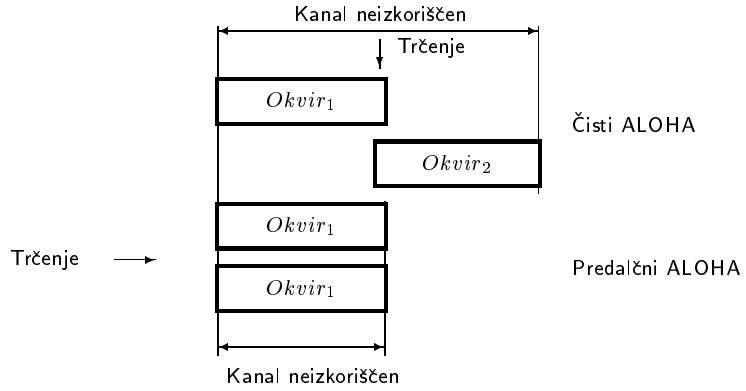
Za vse protokole tipa ALOHA je značilno, da je nadzor dostopa do kanala porazdeljen med vse postaje in da je naključne narave.

Čisti ALOHA Osnovna zamisel sistema ALOHA je enostavna: postaja sme začeti z oddajo okvirja vedno in kadarkoli ima pripravljen okvir, neodvisno od ostalih postaj in neodvisno od zasedenosti kanala. Če v kanal od začetka do konca okvirja oddaja samo ena postaja, ostane okvir nepoškodovan. Prisotnost okvirja zaznajo sicer vse postaje, vendar ga sprejeme samo naslovljena postaja oziroma tista, ki ji je namenjen. Možna je seveda tudi splošna oddaja, to je oddaja na naslov, ki je skupen vsem postajam (ang. Broadcast) ali skupini (ang. Multicast) postaj.

V primeru, da sočasno oddajata vsaj dve postaji, se v skupnem kanalu oba signala pomešata in okvir je uničen. Pojavu sočasnega oddajanja večjega števila postaj v skupen informacijski kanal pravimo trčenje (ang. Collision). Oddajna postaja ima tako kot vse druge postaje možnost sprejemati iz kanala. Torej ima tudi možnost primerjati oddani okvir s sprejetim okvirjem. V primeru, da je prišlo do trčenja, se oddani in sprejeti okvir razlikujeta in potrebna je ponovna oddaja istega okvirja. Čas čakanja pred ponovnim začetkom oddaje mora biti naključen, sicer bodo iste postaje vedno znova trčile.

Kakšna je prepustnost kanala? V primeru, da je malo potreb po oddaji oziroma kanal ni obremenjen, je trčenj malo in praktično vsaka postaja dobi kanal takoj, ko ga potrebuje. Prepustnost in izkoristek kanala sta dobra. Z obremenitvijo kanala se veča število trčenj in prepustnost kanala močno upade. Slika 94 ponazarja problem trčenj.

Srednji okvir se časovno čisto malo prekrije (trči) s predhodnjim okvirjem. Podobno naslednji okvir za malo trči s srednjim okvirjem. Rezultat v narisanih razmerah pa je ta, da so vsi trije okvirji uničeni. Kanal je v tem času neizkoriščen. Naj postaja začne z oddajo okvirja v trenutku t_0 in naj okvirji trajajo čas T_F. Okvir bo 'preživel', če ni nihče pred njim začel z oddajo najmanj čas T_F in če ne



Slika 95: Primerjava čistega in predalčnega ALOHA. Čas neikoriščenosti kanala je pri predalčnem ALOHA polovico krajši kot v čistem ALOHA.

bo nihče začel z oddajo vsaj še čas T_F. Torej sme v intervalu $[t_0 - T_F, t_0 + T_F]$ dolžine $2 \times T_F$ začeti z oddajo samo ena postaja, čeprav bi morda pričakovali, da je dovolj krajši interval T_F.

Predalčni ALOHA Leta 1972 je Roberts objavil metodo, s katero se da podvojiti prepustnost sistema tipa *čisti* ALOHA. Predlagal je, da bi čas delili na enake časovne presledke ali 'predale'. Od tu ime *predalčni* ALOHA (Ang. Sloted ALOHA). Časovno zvezni ALOHA se s tem spremeni v časovno diskretni ALOHA. Trajanje presledka naj bi bilo enako trajanju enega okvirja. Postaje v omrežju tipa predalčni ALOHA ne smejo več začeti z oddajo kadarkoli, denimo v sredini predala, kot pri čistem ALOHA, temveč morajo počakati na naslednji časovni predal. Postaje so s tem sinhronizirane na začetek predala. Sinhronizacijo postaj na začetek predala je na primer možno doseči s kratkim signalom, ki ga periodično oddaja neka naprava. S tem, ko omejimo pravico za začetek oddaje na začetek predala, se 'nered' v kanalu zmanjša in pričakujemo, da se prepustnost zveča.

Kdaj bo oddajanje ovirja uspešno? Oddajanje okvirja uspe, če v intervalu T_F oddaja samo ena postaja. To pa se zgodi, če na začetku predala začne z oddajo samo ena postaja. Namreč, postaji lahko trčita samo na začetku predala ne pa v sredini. V primeru, da začne na začetku predala z oddajo več postaj, pride do trčenja in kanal je v času trajanja predala neizkoriščen. Torej, če postaji trčita, bo pri predalčnem ALOHA kanal neizkoriščen samo čas T_F, in ne (skoraj) dvakrat toliko kolikor je v najslabših razmerah pri čistem ALOHA. Čas neizkoriščenosti kanala zaradi trčenj se skrajša. Vpliv trčenja na neizkoriščenost kanala v obeh izvedbah ALOHA prikazuje slika 95.

5.2 Izkoristek ALOHA

V omrežju ALOHA z velikim številom neodvisnih postaj lahko verjetnost za sočasno oddajanje k postaj aproksimiramo s Poissonovo verjetnostno porazdelitvijo,

$$p_k = \frac{\lambda^k \exp^{-\lambda}}{k!}, \quad (k = 0, 1, 2, \ldots).$$

Pri tem je p_k verjetnost, da v intervalu T_F začne z oddajo natanko k postaj in je λ povprečno število oddajanj (frekvenca) v tem intervalu. Pri predalčnem ALOHA oddaja okvirja uspe, če v časovnem predalu s trajanjem T_F začne z oddajo natanko ena postaja. Kanal je v takem predalu torej izkoriščen. Verjetnost, da se to zgodi je

$$p_1 = \lambda e^{-\lambda}.$$

Kanal ni izkoriščen v tistih predalih, ko ne oddaja nobena postaja ali če začne z oddajo več kot ena postaja. Kakšen naj bo λ, da bo verjetnost p_1 največja. Odvajajmo po λ in izenačimo odvod z nič,

$$\frac{dp_1}{d\lambda} = e^{-\lambda} - \lambda e^{-\lambda} = 0.$$

Verjetnost je maksimalna, če je povprečno število oddaj na predal λ_{opt} enako ena. Maksimalna verjetnost uspeha in s tem maksimalni izkoristek, na katerega lahko upamo v idealnih razmerah, je torej

$$E_{S.ALOHA.max} = \frac{p_{1.max} T_F}{T_F} = 1/e \approx 0.37.$$

To ni veliko, v omrežju tipa čisti ALOHA pa je izkoristek še slabši. Pri čistem ALOHA oddaja okvirja s trajanjem T_F uspe, če v intervalu $2 \times T_F$ začne z oddajo natanko ena postaja. Kanal ni izkoriščen, če v času $2 \times T_F$ ne oddaja nobena postaja ali če začne z oddajo več postaj. Verjetnost, da začne z oddajo v intervalu $2 \times T_F$ k postaj je

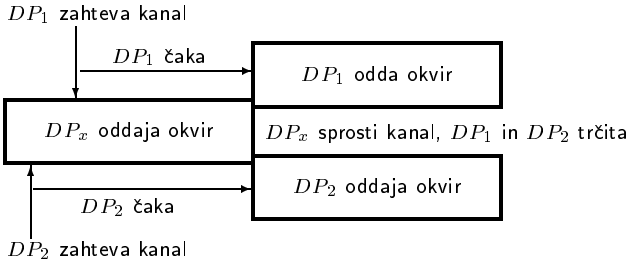
$$p_k = \frac{(2\lambda)^k e^{-2\lambda}}{k!}, \quad (k = 0, 1, 2, \ldots).$$

pri čemer je 2λ povprečno število oddaj v tem intervalu. Verjetnost oddaje enega je:

$$p_1 = 2\lambda e^{-2\lambda}.$$

Ta verjetnost je maksimalna pri $\lambda_{opt} = 1/2$ in znaša $p_{1.max} = 1/e$. V času $2 \times T_F$ je kanal izkoriščen čas T_F. Najboljši izkoristek omrežja čisti ALOHA je zato dvakrat nižji od izkoristka pri predalčnem ALOHA,

$$E_{P.ALOHA.max} = \frac{p_{1.max} T_F}{2T_F} = \frac{1}{2e} \approx 0.18.$$



Slika 96: Primer trčenja v 1-persistentnem CSMA. Čakajoče postaje se sinhronizirajo na koncu okvirja in takoj trčijo.

5.3 Protokoli tipa CSMA

Najboljši izkoristek kanala (prepustnost) predalčnega ALOHA je komaj 37 odstotkov. Torej, od 100 časovnih predalov je samo 37 izkoriščenih. Eden od razlogov za tako slab izkoristek so precej kaotične razmere v kanalu. Pričakujemo, da bi z uvedbo več 'reda' pri dodeljevanju kanala izkoristek kanala narasel.

V lokalnih omrežjih so zakasnitve zaradi širjenja signala v kanalu razmeroma majhne, zato lahko vse postaje kaj kmalu po začetku oddajanja neke postaje zaznajo prisotnost signala na kanalu. Pravzaprav lahko postaje cel čas opazujejo razmere v kanalu in se prilagajajo tem razmeram. Protokole, pri katerih postaje preverjajo prisotnost informacije v kanalu (prisluškujejo) imenujemo prisluškovalni protokoli (ang. Carrier Sense Protocols). V prispodobi bi lahko rekli, da velja pravilo 'poslušaj predno govoriš'.

Eden od prisluškovalnih protokolov je *1-perzistenten CSMA protokol* (Carrier Sense Multiple Access). V tem protokolu vsaka postaja, ki ima pripravljene podatke za oddajo, najprej preveri, če je kanal prost. Če je kanal že zaseden, postaja počaka - zadrži oddajo, dokler ni kanal prost. Ko postane kanal prost, postaja takoj začne z oddajo. V primeru trčenja (istočasnega začetka oddajanja še vsaj ene postaje) postaja počaka naključno dolg čas in nato poskusi znova. Protokol se imenuje 1-perzistenten CSMA zato, ker postaja, ki ima pripravljene podatke, začne oddajanje z verjetnostjo 1 takoj ko ugotovi, da je kanal prost.

V kakšnih okoliščinah pride do trčenja? Predvsem takrat, ko se med oddajanjem neke postaje nabere več postaj, ki čakajo na oddajo. Takoj, ko oddajna postaja sprosti kanal (konec okvirja), se hočejo čakajoče postaje polastiti kanala in zato neizbežno trčijo, glej sliko 96.

Na možnost trčenja bistveno vpliva tudi čas širjenja signala vzdolžkanala. Poglejmo zakaj. Naj bo čas širjenja signala med skrajnima koncema kanala (kabla) enak τ. Denimo, da je na enem koncu postaja A, na drugem koncu postaja B in da obe postaji želita pridobiti kanal. Postaja A preveri kanal in

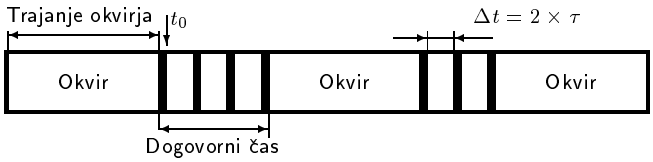
ugotovi, da je kanal prost. Zato začne z oddajo okvirja. Signal se širi vzdolž kanala in pride do postaje B šele po času τ. Tudi postaja B se skuša polastiti kanala. Predno signal pride do postaje B le-ta čuti, da je kanal prost. Torej sme začeti z oddajo in trčenje je neizbežno. Do trčenja lahko pride kadarkoli znotraj intervala τ. Daljši je ta čas, večja je možnost trčenja in slabša je prepustnost kanala.

Naslednji protokol je *neperzistenten CSMA* (0-perzistenten CSMA). V tem protokolu so postaje manj požrešne. Vsaka postaja pred oddajo prisluškuje, tako kot prej. Če je kanal prost (nihče ne oddaja), začne z oddajo. Če pa je kanal zaseden, se postaja ne skuša polastiti kanala takoj, ko ga sprosti oddajna postaja, kar bi se zgodilo pri 1-perzistentnem CSMA. Pri neperzistentnem CSMA postaja počaka naključno dolgo in šele nato poskusi znova z oddajo. Neperzistenten CSMA zagotavlja boljši izkoristek kanala, vendar je čakalni čas postaj daljši.

Zadnji protokol tega tipa je *p-perzistenten* CSMA. Temelji na predalčnem ALOHA in deluje takole. Predno postaja začne z oddajo, preveri stanje kanala. Če je kanal zaseden, postaja počaka na naslednji presledek in ponovi preverjanje kanala. Če pa je kanal prost, začne oddajati z verjetnostjo p. Z verjetnostjo $q = (1 - p)$ počaka z oddajo do naslednjega predala. V primeru, da se postaja odloči za čakanje na naslednji predal, se razmere ponovijo. Torej, če je naslednji predal prost, začne oddajati z verjetnostjo p in ne začne oddajati z verjetnostjo q. Če pa je med tem začela z oddajo druga postaja (naslednji presledek je zaseden), postaja postopa enako, kot če bi trčila. To je, počaka naključno dolgo in tedaj poskusi znova. Pri *p-perzistentnem* CSMA je p parameter z vrednostjo med nič in ena, ki vpliva na prepustnost kanala in tudi na čas čakanja postaj na kanal. V splošnem velja, da za manjši p prepustnost kanala z obremenitvijo počasneje pada, zato je prepustnost kanala pri večji obremenitvi večja. Zakasnitev oddaje posameznih postaj pa narašča.

5.4 Protokol CSMA/CD

x-perzistentni CSMA ($0 \leq x \leq 1$) protokoli so v pogledu prepustnosti kanala nedvomno boljši od čistega ali predalčnega ALOHA. Preprosto zato, ker postaje ne začnejo z oddajo, če je kanal že zaseden. S tem se v večini primerov izognejo trčenju. Povejmo še, da je v vseh do sedaj obravnavanih protokolih potrebna enaka materialna oprema, različen je samo algoritem. Tako pri protokolih CSMA kot pri protokolih ALOHA pa postaje oddajo okvir do konca, tudi če je prišlo do trčenja. Prepustnost kanala se da izboljšati, če postajam omogočimo, da prekinejo z oddajo čimprej po trčenju, saj nadaljevanje oddajanja okvirja, ki je v vsakem primeru že pokvarjen, pomeni s strani kanala čisto izguba časa. Potrebujejo pa postaje dodatno vezje za odkrivanje trčenj in za prekinitev oddajanja, ko se trčenje odkrije. V prispodobi bi mogli imenovati tak način dostopa do



Slika 97: Model dodeljevanja kanala po protokolu CSMA/CD.

kanala 'poslušaj tudi ko govoriš'. Protokole s to lastnostjo označujemo s kratico CSMA/CD (Carrier Sense Multiple Access with Collision Detection). Protokoli CSMA/CD se množično uporabljajo v lokalnih omrežjih. Model CSMA/CD (prisluškovanje z odkrivanjem trčenj) prikazuje slika 97.

V trenutku t_0 je neka postaja končala z oddajo. Vsaka druga postaja, ki ima potrebo po oddaji, lahko tedaj poskusi z oddajo, saj je kanal prost. V primeru, da nobena postaja nima potrebe za oddajo, bo kanal ostal prost - neizkoriščen. Če začne z oddajo ena sama postaja, bo ta pridobila kanal in kanal bo v času trajanja okvirja spet izkoriščen. Če pa se za začetek oddajanja odločita vsaj dve postaji, pride do trčenja in kanal je neizkoriščen. Vsaka postaja bo (skoraj) takoj odkrila trčenje, prekinila z oddajo, počakala naključno dolgo in nato poskusila znova. Kanal bo v tem primeru neizkoriščen toliko časa, kolikor je potrebno za to, da se trčenje odkrije. V kanalu so torej možna tri stanja: v kanalu je okvir in kanal je izkoriščen, kanal je prost in zato neizkoriščen in postaje so trčile in kanal je neizkoriščen. V bistvu se izmenjujeta dve obdobji: obdobje, ko je kanal *izkoriščen* in obdobje, ko je kanal *neizkoriščen*. Trajanje obdobja, ko je kanal neizkoriščen, se naključno spreminja. Trajanje obdobja, ko je kanal izkoriščen, pa je vedno enako in je enako trajanju okvirja. Trajanje obdobja, ko je kanal neizkoriščen, bomo imenovali dogovorni čas. Sestavljen je iz poljubnega števila časovnih presledkov Δt. Časovni presledek Δt je določen s časom, ki je potreben, da postaje odkrijejo trčenje. Poglejmo, koliko časa je potrebno za to. Denimo, da začneta postaji A in B z oddajo natanko v trenutku t_0. Postaja A je na enem skrajnem koncu kanala in postaja B je na drugem skrajnem koncu. Predno pride signal z enega konca kanala na drugi konec, mine čas τ. To je hkrati tudi čas, ki je potreben, da se trčenje odkrije (če zanemarimo čas, ki ga rabi zase nadzorno vezje posameznih postaj). V najslabših razmerah, in prav je, da upoštevamo najslabše razmere, pa je ta čas še enkrat daljši. Vzemimo, da se v trenutku t_0 skuša polastiti kanala postaja A. Predno postaja B za to zve, mine čas τ. Lahko se zgodi, da začne sama z oddajo tik pred tem in trčenje je tu. Postaja A se trčenja zave šele po času τ, ko jo doseže signal postaje B. Da se trčenje odkrije, je v najslabših razmerah potreben čas skoraj $2 \times \tau$. Postaja A je prepričana, da se je polastila kanala šele potem, ko je oddajala čas $2 \times \tau$. Na primer, na kablu dolžine 1000 metrov je $\tau \approx 5\mu s$ in $2 \times \tau$ je približno 10 μs. Čim daljši je kabel, tem večja je možnost trčenja. Poudarimo, da je zaznavanje trčenj analogen in ne digitalen

problem. Postaja mora vsekakor sprejemati med oddajanjem. Če se oddani signal razlikuje od sprejetega, pomeni to trčenje. Informacijski signal mora biti zato tak, da je možno razlikovati med pokvarjenim signalom in nepokvarjenim signalom. Recimo, trčenje dveh signalov z vrednostjo 0 voltov je nemogoče ugotoviti. Zato se v večini primerov uporablja 'Manchester' postopek kodiranja signalov.

5.5 Izkoristek protokola CSMA/CD

Kot smo že ugotovili, se pri dodeljevanju kanala po protokolu CSMA/CD na kanalu izmenjujeta dve obdobji: obdobje s trajanjem T_F, ko je kanal izkoriščen in obdobje s povprečnim trajanjem $\overline{T}$, ko kanal ni izkoriščen. Kanal je izkoriščen, ko se ena od postaj polasti kanala. Trajanje tega obdobja je določeno s trajanjem okvirja T_F. Trajanje obdobja, ko kanal ni izkoriščen se naključno spreminja. Sestavlja ga poljubno število časovnih presledkov Δt, (glej sliko 97). Izkoriščenost kanala $E_{CSMA/CD}$ bomo definirali z razmerjem časov,

$$E_{CSMA/CD} = \frac{T_F}{T_F + \overline{T}}. \tag{32}$$

Izkoriščenost je največja (enaka 1), ko je $\overline{T} = 0$. V izrazu (32) nastopa povprečni čas neizkoriščenosti kanala, ki ga moramo šele določiti. Za ovredotenje protokola CSMA/CD predpostavimo močno in enakomerno obremenjeno omrežje. V teh razmerah je stanje v omrežju najbolj kritično. Namreč, v neobremenjenem omrežju je kanal večinoma prazen in postaje dobijo kanal (skoraj) vedno takoj, ko ga potrebujejo. V obremenjenem omrežju pa obstaja več postaj, ki čakajo na oddajo. Naj bo v omrežju vedno N čakajočih postaj. Nadalje predpostavimo, da je verjetnost, da bo ena od N postaj začela z oddajo enaka za vse postaje in da znaša p. Izračunajmo najprej, kolikšna je verjetnost, da se ena od čakajočih postaj polasti kanala. To se zgodi, če ena od postaj in nobena druga začne z oddajo. Verjetnost, da se to zgodi je

$$p_u = N \times p \times (1-p)^{N-1}.$$

Ko se ena pod postaj polasti kanala, se konča dogovorni interval in začne interval oddajanja. Da bi izračunali povprečni čas trajanja dogovornega intervala, izračunajmo najprej verjetnost, da traja dogovorni interval k, $(k = 0, 1, 2, \ldots, \infty)$ časovnih presledkov Δt:

Brez čakanja $(k = 0)$,	$p_0 = p_u$
Čaknaje Δt $(k = 1)$,	$p_1 = (1 - p_u) \times p_u$
Čakanje $2 \times \Delta t$,	$p_2 = (1 - p_u) \times (1 - p_u) \times p_u$
$\ldots$	$\ldots$
Čakanje $k \times \Delta t$,	$p_k = (1 - p_u)^k \times p_u$
$\ldots$	$\ldots$

Povprečno število časovnih presledkov Δt je:

$$\bar{k} = \sum_{k=0}^{\infty} k \times p_k = \sum_{k=1}^{\infty} k \times (1-p_u)^k \times p_u = p_u \times \sum_{k=1}^{\infty} k \times q^k, \tag{33}$$

kjer je $q = (1 - p_u)$. Ko izračunamo neskončno vsoto (33), dobimo:

$$\overline{k} = p_u \times \frac{q}{(q-1)^2} = \frac{1-p_u}{p_u}$$

in povprečni čas $\overline{T}$, ko kanal ni izkoriščen, je $\overline{T} = \overline{k} \times \Delta t$. Izkoriščenost kanala je:

$$E_{CSMA/CD} = \frac{T_F}{T_F + \frac{1-p_u}{p_u}\Delta t} = \frac{1}{1 + \frac{1-p_u}{p_u}\frac{\Delta t}{T_F}}.$$

Ker p_u pada z naraščanjem obremenitve omrežja (s številom čakajočih postaj N), pada z obremenitvijo tudi izkoriščenost. Poglejmo, kakšen naj bo p, da bo verjetnost uspeha p_u največja. Odvajajmo p_u na p in odvod izenačimo z nič,

$$\frac{dp_u}{dN} = N(1-p)^{N-1} - N(N-1)(1-p)^{N-2} = 0, \rightarrow p = \frac{1}{N}.$$

Verjetnost p_u je največja, če postaje pri oddaji upoštevajo število postaj N tako, da je $p = 1/N$ in

$$p_{u.max} = (\frac{N-1}{N})^{N-1}$$

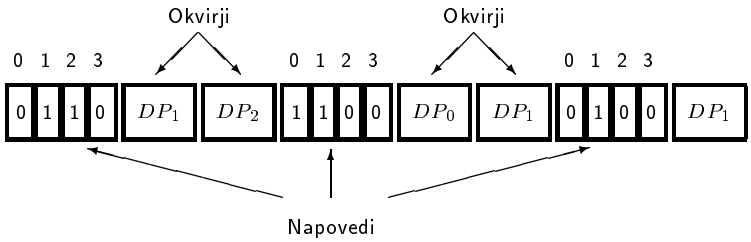
Vrednost tega izraza se v odvisnosti od N kaj malo spreminja (za $N = 4$ je $p_{u.max} = 0.42$, za $N = 8$ je $p_{u.max} = 0.39$, za $N = 16$ je $p_{u.max} = 0.38$, i.t.d.). Za naše izračune naj bo $p_{u.max}$ kar enak 0.4. Rekli smo že, da je časovni presledek Δt približno enak dvojnemu času potovanja signala z enega na drugi konec kabla, $\Delta t = 2 \times \tau$. Z upoštevanjem teh ugotovitev dobi izraz za izkoristek naslednjo obliko,

$$E_{CSMA/CD} = \frac{1}{1 + 3\frac{\tau}{T_F}}.$$

Torej na en uspešen prenos okvirja v povprečju izgubimo 3τ časa, v resnici pa je ta čas še daljši. Na trajanje okvirja vpliva število bitov v okvirju (dolžina okvirja) in hitost prenosa. Predpostavimo, da postaje oddajajo z največjo možno hitrostjo C, ki jo dovoljuje kanal in da okvir vsebuje F bitov. Upoštevajmo še, da je čas širjenja signala po kanalu odvisen od hitrosti širjenja v in dolžine kabla L. Izkoriščenost kanala potem je:

$$E_{CSMA/CD} = \frac{1}{1 + 3\frac{L\,C}{v\,F}}. \tag{34}$$

Izkoriščenost kanala torej raste z dolžino okvirja in pada z dolžino kabla, glej enačbo (34). Čim večja je kapaciteta kanala (kvalitetna in zato draga prenosna pot), tem večji je vpliv dolžine kabla in obremenitve omrežja na upadanje prepustnosti omrežja. Iz tega sklepamo, da v obsežnih omrežjih z visoko kapaciteto kanala (veliko pasovno širino) protokol CSMA/CD ni najbolj primeren.



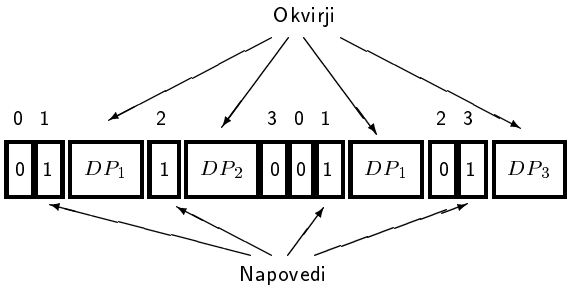
Slika 98: Dodeljevanje kanala z napovedovanjem (Osnovni napovedni protokol). Po napovednem intervalu sledijo napovedani okvirji.

5.6 Protokoli brez nevarnosti trčenja

V protokolih tipa CSMA/CD ne more priti do trčenja, čim se neka postaja polasti kanala. Do trčenja more priti samo v dogovornem času. Vendarle je trčenje možno in pri velikem času kasnenja (dolg kabel) in kratkem času trajanja okvirja (majhna informacijska vsebina in velika hitrost prenosa) lahko tudi to resno ogrozi prepustnost kanala. To pa pomeni, da utegne postati v omrežjih prihodnosti z večjimi razsežnosti in z višjimi hitrosti prenosa danes najbolj priljubljeni protokol CSMA/CD neprimeren. Trčenjem se lahko izognemo na dva načina: z napovedovanjem oddaje pred začetkom oddaje ali z eksplicitnim dodeljevanjem pravice za oddajo. Ta pravica, ki jo običajno imenujemo "žeton" (ang. Token), po nekem pravilu kroži od postaje do postaje in dokler je v omrežju samo en žeton do trčenja ne more priti. V tem podpoglavju bomo opisali dva protokola iz prve skupine, ki popolnoma preprečita možnost trčenja. Oba temeljita na predhodni napovedi oddaje. V kanalu se izmenjujeta dve obdobji: napovedni interval in oddajni interval. V napovednem intervalu imajo postaje možnost napovedati potrebo po oddaji.

V osnovni obliki napovednega protokola (ang. Basic Bit-Map Method) sestavlja napovedni interval N časovnih predalov (bitni vzorec dolžine N bitov). Vsaki postaji na skupnem kanalu je dodeljen en predal (en bit informacije). Postaje so označene z naslovi, denimo $0, 1, \ldots, N-1$. Primer dodeljevanja kanala za štiri postaje ponazarja slika 98.

Če postaja z naslovom i hoče napovedati oddajo, postavi v i-ti predal vrednost ena, sicer pa nič. Po koncu napovednega intervala se v prepisanem zaporedju zvrstijo okvirji postaj, ki so predhodno napovedale oddajo. V primeru, da neka postaja 'zamudi' napovedni interval, mora počakati na naslednjega. Lepa stran takega protokola je določljivost najslabših razmer, to je najdalšega časa čakanja na oddajo za katerokoli postajo.



Slika 99: Dodeljevanje kanala po protokolu BRAM. Postaja dobi kanal takoj po napovedi.

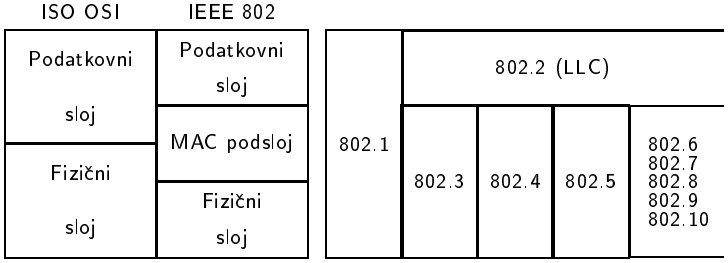
Osnovni napovedni protokol ima vsaj eno pomankljivost. Namreč, v primeru neobremenjenega kanala mora tista postaja, ki postavi zahtevo za oddajo, najprej počakati, da mine cel napovedni interval. Šele nato se lahko polasti kanala. Posledica takega načina dodeljevanja kanala je daljši čas čakanja na kanal kot je v resnici potreben. Leta 1976 sta Chlamtac in Scholl neodvisno drug od drugega in tudi pod drugim imenom predlagala protokol, ki nima te slabosti. Chlamtac je protokol imenoval BRAM (Broadcast Recognition Access Method). Dostop do kanala tudi po tem protokolu ureja napovedni bit, vendar sledi dodelitev kanala takoj po napovedi.

Princip dodeljevanja kanala v primeru štirih postaj je skiciran na sliki 99. Dokler ni nobene zahteve za oddajo, traja napovedni interval. Kakor hitro se hoče ena od postaj polastiti kanala, postavi svoj napovedni bit v stanje ena in s tem prekine napovedni interval. Takoj za tem se že polasti kanala, odda okvir in obmolkne. Napovedni interval se potem nadaljuje na mestu prekinitve. Možnost napovedi in zatem oddaje dobi postaja z naslednjim naslovom.

Protokol BRAM je boljši od osnovnega napovednega protokola, ker postajam zagotavlja krajši čakalni čas na kanal. V pogledu izkoriščenosti kanala sta oba protokola enakovredna. Z naraščanjem obremenitve kanala sta si oba protokola enakovredna tudi v pogledu časa čakanja. V obremenjenem kanalu prispeva h kasnitvi pred oddajo glavni delež prisotnost okvirjev v kanalu, čas čakanja zaradi samega napovedovanja okvirjev pa je zanemarljiv.

5.7 Lokalna omrežja in standardi IEEE 802

Na pobudo velikih proizvajalcev komunikacijske opreme je IEEE izdelal standarde, po katerih naj bi gradili lokalna računalniška omrežja. Standarde IEEE 802 so kasneje postali tudi ANSI in ISO (z oznako ISO 8802). Stadardi z oznako IEEE 802.x (x = 1,2,3,4,5) se v skladu z referenčnim modelom OSI nanašajo na podatkovni in fizični sloj lokalnih omrežij, glej sliko 100. Standard IEEE 802.1 se nanaša na skupne lastnosti ostalih štirih in definira osnovne operacije na obeh vmesnikih podatkovnega sloja ter most (ang. Bridge) - komunikacijsko napravo podatkovnega sloja. Standard IEEE 802.2 opisuje zgornji del podatkovnega sloja ter protokol znan pod kratico **LLC** (**L**ogical **L**ink **C**ontrol). Ostali trije standardi se nanašajo na spodnji del podatkovnega sloja MAC (ang. Media Access Control Sublayer): IEEE 802.3 na CSMA/CD, IEEE 802.4 na vodilo z žetonom (ang. Token Bus), in IEEE 802.5 na obroč z žetonom (ang. Token Ring).



Slika 100: Standardi IEEE 802 in sloji modela OSI.

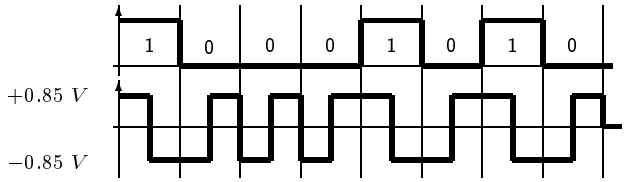
5.7.1 IEEE 802.3 in Ethernet

Standard IEEE 802.3 standardizira 1-perzistenten CSMA/CD način dostopa do skupnega kanala topologije vodilo (ang. Bus) ali drevo (ang. Tree). Ko postaja rabi kanal, najprej prisluhne kanalu. Če je kanal prost, postaja začne z oddajo. Če je kanal že v lasti druge postaje, ta postaja počaka, dokler ga druga postaja ne sprosti. Do trčenja pride v primeru, da začne sočasno z oddajo več kot ena postaja. Postaje, ki trčijo in odkrijejo trčenje najprej oddajo motilni signal, da je trčenje gotovo zaznano, potem prekinejo z oddajo in poskusijo znova po naključno dolgem času čakanja.

Največ zaslug za standardizacijo CSMA/CD gre Xerox-u, DEC-u in Intel-u, korenine za nastanek CSMA/CD pa segajo prav do Abramsonovega sistema ALOHA, ki ga je Metcalfe v svoji disertaciji na MIT-ju izpopolnil z odkrivanjem trčenj. Po zamisli Metcalfea so pri Xerox-u leta 1976 realizirali lokalno omrežje tega tipa (Metcalfe in Boggs 1976) s priljubljenim imenom *Ethernet*. Omrežje, ki je delovalo s hitrostjo 2.94 Mb/s in je povezovalo nad 100 postaj, se je izkazalo za tako uspešno, da so DEC, Intel in Xerox (zato kratica DIX) izdelali standard za

Tabela 5: Omrežja 802.3 glede na prenosni medij, hitrost prenosa, frekvenčni pas, razsežnost in število postaj.

	10BASE5 (Ethernet)	10BASE2 (CheaperNet)	1BASE5 (StarLan)	10BROAD36 (BroadBand)	10BASE-T
Medij	koaksialec 50 Ω 10mm (Debeli Ethernet)	koaksialec 50 Ω 5mm (Tanki Ethernet)	neoklopljena parica	koaksialec 75 Ω	dve simleksni neoklopljeni parici
Max. segment Max. omrežje	500 m 2500 m	185 m 925 m	500 m 2500 m	1800 m 3600 m	100 m 1000 m
Max. postaj na segment	100	30			2
Odkrivanje trčenj	Presežen tok	Presežen tok	Dva aktivna vhoda na spojišču	Oddani sig. različen od sprejetega	Aktivnost oddajnika in sprejemnika

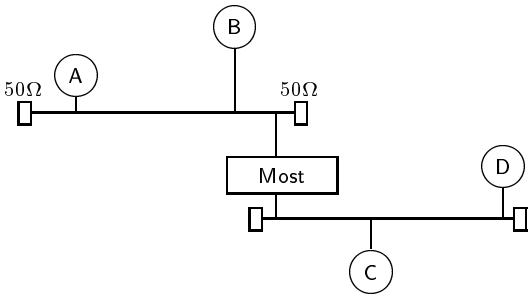


Slika 101: Manchester oblika signala.

10 Mb/s Ethernet, ki je potem služil kot osnova pri izdelavi standarda Ieee 802.3. Standard 802.3 se razlikuje od (de-facto) predhodnika po tem, da opisuje celo družino protokolov tipa 1-perzistenten CSMA/CD, predvideva različne prenosne medije in dovoljuje različne hitrosti, od 1 Mb/s do 10 Mb/s. V navadi je, da imenujemo Ethernet vsako omrežje, ki deluje po principu CSMA/CD. Bolj prav pa je, da Ethernet obravnavamo le kot eno od realizacij principa CSMA/CD. Izraz Ethernet se pojavlja nekako v treh pomenskih zvezah: Ethernet kot prvo omrežje s CSMA/CD dostopom, Ethernet kot prenosni medij v obliki 50 ohmskega koaksialnega kabla in Ethernet II kot de-facto standard oziroma protokol, ki ga je leta 1985 zamenjal IEEE 802.3. Tabela 5 prikazuje nekatere zanimivejše lastnosti omrežij po standardu 802.3.

Iz same oznake se da razbrati hitrost prenosa (v Mb/s), frekvenčni pas in maksimalno dovoljeno dolžina segmenta (v metrih $\times 100$). Tako na primer 10BASE5 pomeni hitrost prenosa 10 Mb/s, prenos v osnovnem frekvenčnem pasu (BASE) in dolžino segmenta 500 metrov. Ethernet in 802.3 10BASE5 sta si zelo podobna z majhno razliko v obliki okvirjev.

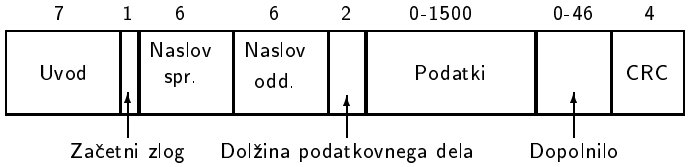
Oglejmo si najprej nekaj zanimivejših določil standarda IEEE 802.3 za lokalno omrežje tipa vodilo, ki za prenosni medij izkorišča koaksialni kabel s signalom v osnovnem frekvenčnem pasu (Ang. Baseband). IEEE 802.3 predpisuje za tak prenos Manchester obliko signala, ker omogoča dobro sinhronizacijo



Slika 102: Most poveže dva segmenta, vendar loči promet na enem segmentu od prometa na drugem segmentu. Postaji A in B lahko komunicirata sočasno s postajama C in D.

sprejemnika z oddajnikom in omogoča odkrivanje trčenj. V kanalu so možna tri stanja: logična nič, kodirana s prehodom signala z nizkega (-0.85 V) na visok nivo (+0.85 V), logična ena (prehod signala z visokega na nizek nivo) ali pa je kanal prazen (signal 0 voltov), kot je skicirano na sliki 101. Enosmerna komponenta signala Manchester je nič . Zahteva večjo (dvojno) širino frekvenčnega pasu kanala, kar pri prenosu signala v osnovnem frekvenčnem pasu ne predstavlja resnejšega problema, saj je na razpolago celotna frekvenčna širina koaksialnega kabla ($\approx 300MHz$). Najpogosteje se za prenosni medij uporabljata dve izvedbi 50-ohmskega koaksialnega kabla: 'debeli Ethernet' (10BASE5) in 'tanki Ethernet' (10BASE2), predvidena pa je tudi uporaba navadne telefonske parice (StarLan1BASE5 in 10BASE-T). Debeli Ethernet je kvalitetnejši (in debelejši) koaksialni kabel, običajno rumene barve z oznakami vsakih 2.5 metra. Na označenih mestih se sme neposredno priključevati enote za dostop do medija. Enota za dostop do medija (Ang. Media Access Unit - MAU) je povezana s postajo z do 50 metrov dolgim kablom, ki združuje pet oklopljenih paric. Največja dopustna dolžina segmenta (neprekinjenega kosa koaksialnega kabla) je 500 metrov. 'Tanki Ethernet' je tanjši, cenejši koaksialni kabel in dovoljuje krajše razdalje (segment do 185 metrov). Postaje se nanj priključujejo s cenejšimi T členi in BNC konektorji. Obe vrsti kabla se v razsežnejših lokalnih omrežjih običajno kombinirata. Debeli kabel služi za 'hrbtenico' omrežja, nanj se preko spojišč (ang. Multiport Repeater) vežejo segmenti tankega Etherneta, na tanki Ethernet pa se preko T členov vežejo neposredno postaje z vgrajenim vmesnikom z visokohmskim priključkom. Kabel mora biti na obeh koncih zaključen s 50 ohmskimi zaključniki.

Za večje razsežnosti omrežja so potrebni ponavljalniki in mostovi. Ponavljalnik (Ang. Repeater) je naprava, ki sprejme signal z enega segmenta, ga obnovi in odda na naslednji segment. Ponavljalnik je naprava prvega (fizičnega) sloja in ne potrebuje programske opreme. *Most* (Ang. Bridge) ali selektivni ponavljalnik je 'pametnejši' - vsebuje programsko opremo. Sodi na drugi, podatkovni sloj. Most preverja okvirje in jih po potrebi spušča naprej na naslednji segment.



Slika 103: Oblika okvirja po standardu Ieee 802.3.

Recimo, če postaja A pošlje okvir postaji C, ga most spusti naprej (slika 102). Če pa postaja A pošilja postaji B, ostane okvir na istem segmentu. Most logično in fizično loči promet na posameznih segmentih, zato lahko istočasno komunicira A z B in C z D, s čimer se poveča prepustnost omrežja. Celotno lokalno omrežje tipa CSMA/CD sme vsebovati več kabelskih segmentov, vendar je razdalja med poljubnima postajama na kanalu omejena na 2500 metrov, signal pa ne sme nikoli prečkati več kot štiri vmesne naprave (ponavljalnike). Večje dolžine kabla omejuje čas širjenja signala, ki neposredno vpliva na pogostost trčenj in s tem na prepustnost omrežja.

Omrežje StarLan 1BASE5 (glej tabelo) je dobilo ime po svoji obliki "zvezde". Postaje so z do 500 metrov dolgo parico vezane na skupno spojišče. Spojišče sprejeti signal enostavno razpošlje vsem postajam, tako da je dostop do kanala tudi tu CSMA/CD. Hitrejše omrežje 10BASE-T (T za Twisted-Pair) uporablja enako kofiguracijo omrežje, le da so postaje vezane na spojišče z dvema paricama, ena je za prenos v eno smer, druga je za prenos v drugo smer.

Obliko okvirja po standardu Ieee 802.3 prikazuje slika 103. Med dvema zaporednima okvirjema mora biti časovna praznina v trajanju vsaj 96 bitov oziroma 9.6 mikrosekund. Okvir začenja z uvodom sedmih zlogov za sinhronizacijo (Bitni vzorec 10101010). Ker je signal tipa Manchester, predstavlja uvodni del val pravokotnih impulzov s frekvenco 10 MHZ v skupnem trajanju 5.6 μs. Uvodnemu delu sledi začetni zlog (10101011) in za njim dve polji naslovov: naslov pošiljatelja (oddajnika) in naslov namembne postaje (sprejmnika). Vsak naslov obsega dva ali šest zlogov. Za naslovom pošiljatelja sledi polje (dva zloga), ki vsebuje število zlogov v podatkovnem delu okvirja. V podatkovnem delu okvirja se prenaša LLC okvir podatkovnega protokola, ki obsega najmanj 3 bajte. Število podatkovnih zlogov gre torej od 3 do največ 1500 bajtov. Za podatkovnim delom so še štirje zlogi za ciklično preverjanje okvirja s polinomom stopnje 32,

$$X^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1.$$

Okvir mora vsebovati najmanj 64 zlogov (od začetka do konca) in traja $8 \times 64 \times 0.1\mu s = 51.2\mu s$. Če je podatkovni del krajši od 46 zlogov, se ga dopolni z dopolnilnimi zlogi na dolžino 46 zlogov. Omejitev dolžine okvirja navzdol je potrebna zato, da oddajno vozlišče ne konča z oddajo še predno bi prvi bit okvirja prišel do najbolj oddaljene postaje in nazadnje "trčil". Poleg tega pa se s tem postajam olajša razlikovati zaradi trčenja "odrezane" okvirje od kratkih koristnih okvirjev. Postaja mora biti sposobna odkriti trčenje najkasneje v času trajanja

450 bitov ali 45 mikrosekundah. Motilni signal, ki ga potem odda obsega od 32 do 48 bitov. Trajanje odrezanega okvirja tako ne preseže 49.5 mikrosekund, kar je manj od tarjanja najkrajšega veljavnega okvirja.

V primeru, da postaja trči, sme poskusiti znova po premoru naključne dolžine. Trajanje premora je mnogokratnik trajanja 512 bitov. Postaja, ki trči poskuša znova z enako verjetnosti po premoru dolžine 0×512 bitov ali 1×512 bitov ali 2×512 i.t.d. Verjetnost oddaje okvirja v k-tem trenutku upada s številom trčenj. Po 16 zaporednih neuspešnih poskusih postaja prekine z oddajo. Trenutek ponovne oddaje se določa takole. V primeru trčenja postaja generira naključno število $k \in \{0, 1, 2, 3, ..., 2^m - 1\}$, kjer je $m = min\{n, 10\}$ in je n število trčenj ter poskusi znova po premoru v trajanju $k \times 512$ bitov. Po prvem trčenju torej postaja poskuša znova bodisi v trenutku 0×512 ali 1×512 in sicer z enako verjetnostjo. Po drugem trčenju postaja poskuši z enako verjetnostjo po $0, 1, 2, 3$ časovnih presledkih dolžine 512 bitov, i.t.d. Ponovni poskusi oddaje postaj, ki trčijo, so na ta način časovno razpršeni. Verjetnost ponovnega trčenja s številom trčenj upada, čakalni čas pa narašča. Po več kot desetih trčenjih postaja zadrži oddajo za največ 1023×512 bitov ali pri hitrosti oddajanja 10 Mb/s za 52.4 milisekund. Angleško ime opisanega algoritma čakanja je Truncated Binary Exponential Backoff, mi pa bi ga lahko prevedli binarno eksponentno čakanje.

Denimo, da trčita dve postaji. Verjetnost ponovnega trčenja je $1/2 \times 1/2 + 1/2 \times 1/2 = 1/2$. Verjetnost, da trčita še tretjič je enaka produktu verjetnosti, da trčita drugič in da trčita tudi tretjič: $1/2 \times 1/4 = 1/8$. Verjetnost ponovnega trčenja hitro pada.

Ko se postaja polasti kanala in odda okvir, je za njo delo le na pol opravljeno. Oddani okvir se lahko zaradi nepravilnosti v kanalu tudi pokvari in potreben bo ponoven prenos. Torej mora naslovljena naprava pozitivno ali negativno potrditi sprejem okvirja. Če naj naslovljena naprava potrdi sprejem okvirja, mora seveda priti do kanala in oddati potrdilo, enako, kot pri oddaji podatkovnega okvirja. Potrdilo okvirja lahko zato precej zamuja. Kako zagotoviti hitro potrjevanje okvirjev? Ena od možnih rešitev tega problema je rezervacija prvega prostega časovnega presledka po oddanem okvirju za naslovljeno postajo.

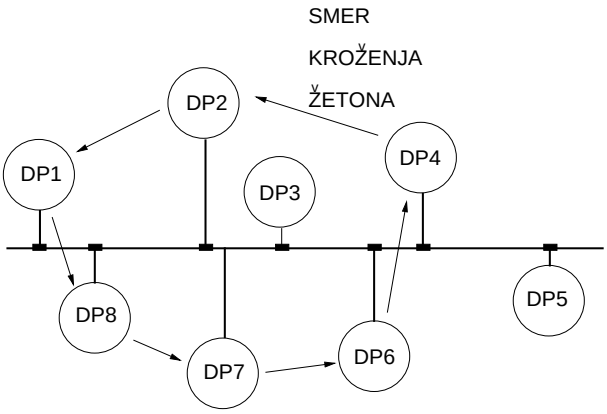
5.7.2 Vodilo z žetonom in IEEE 802.4

Lokalno omrežje tipa vodilo po standardih ETHERNET II in IEEE 8002.3 so se od vsega začetka množično uporabljala v raznih okoljih. Vendar pa so bili razvijalci pri GM-u,[21] (General Motors) Boing-u in IBM-u mnenja, da v industrijskem okolju *naključni dostop* do kanala ne pride v poštev, kajti lahko se zgodi, pa čeprav z zelo majhno verjetnostjo, da postaja čaka na kanal zelo zelo dolgo, dlje, kot to dovoljuje industrijski proces. Čas čakanja na kanal v omrežju tipa CSMA/CD ni navzgor omejen. Druga kritika omrežja tipa CSMA/CD gre na račun nezmožnosti izvedbe *prioritetnega sistema* dodeljevanja kanala. Namreč, manj pomembni okvirji lahko brez razloga prehitevajo bolj pomembne okvirje, kar je v sistemih, ki morajo delovati v stvarnem času, nedopustno. V mrežah s topologijo obroč dostop do kanala ni naključen, čas čakanja na kanal v najslabših razmerah pa je za vsako postajo navzgor omejen in vnaprej znan. Namreč, v omrežju z N postajami, bo poljubna postaja čakala na kanal največ $(N-1) \times T$, kjer je T čas trajanja okvirja oziroma najdaljši čas oddajanja posamezne postaje. Razvijalcem zavzetih za avtomatizacijo proizvodnje je bil všeč koncept dodeljevanja kanala v sistemih z obročem, ni pa jim bila všeč *fizična izvedba obroča*, saj v primeru napake-prekinitve kanala, izpade celotno omrežje. Omrežja z vodilom se odlikuje po robustnosti, pa tudi njihova oblika bolj ustreza industrijskim okoljem (na primer tekočemu traku). Zato so iskali rešitev v kombinaciji obeh oblik omrežij. Predlagali so omrežje tipa *vodilo z žetonom*[22] (ang. Token Bus). Fizično je *vodilo z žetonom* vodilo (kabel), ki gre od enega konca do drugega konca ali pa ima obliko razvejanega drevesa (slika 104), medtem ko je način dostopa do kanala podoben tistemu v omrežjih tipa obroč.

V omrežju tipa vodilo z žetonom je dostop do kanala urejen z žetonom. Žeton pomeni pravico do oddaje. Samo postaja, ki ima v lasti žeton, sme začeti z oddajo okvirja. Ker je v omrežju en sam žeton, do trčenja ne more priti. Žeton je poseben bitni vzorec oziroma ustrezno označen kratek okvir, ki "kroži" od postaje do postaje. Ko postaja odda podatkovni okvir ali zaporedje okvirjev, nazadnje odda še žeton in zraven pove komu je namenjen. V omrežju po obliki vodilo zaznajo žeton sicer vse postaje, sprejme pa ga le naslovljena postaja (tista, ki ji je žeton namenjen). Vsaka postaja ima na vodilu enoznačno določen naslov, po katerem je znana drugim postajam. Vsaka postaja na vodilu pozna naslov predhodne postaje, od katere bo dobila žeton in naslov nasledne postaje, ki ji bo prepustila žeton. Na ta način se pravica do oddaje seli od postaje do postaje v predvidenem zaporedju. Zadnja postaja preda žeton spet prvi postaji in logični obroč je sklenjen. Vrstni red, v katerem so postaje fizično priključene na kanal, je nepomemben. Možno je tudi, da so v logičnem obroču samo nekatere od fizično

[21] GM je razvil omrežje MAP (Manufacturing Automation Protocol), ki temelji na vodilu z žetonom

[22] tudi vodilo s podajanjem žetona (ang. Token-Passing Bus)



Slika 104: Vodilo z žetonom. V logičnem obroču so samo postaje 8, 7, 6, 4, 2, 1. Postaji 3 in 5 nista v obroču.

priključenih postaj, kot je narisano na sliki 104. Na kanalu se vrstijo okvirji, ki jih poslušajo vse postaje, ki so v logični zanki, okvir pa sprejme samo tista postaja, ki ji je okvir namenjen.

V začetku obratovanja omrežja logičen obroč še ne obstaja in potrebno ga je šele vzpostaviti. Vzdrževanje obroča pa je potrebno tudi med obratovanjem omrežja. Razmere v omrežju se s časom spreminjajo, nekatere postaje postajajo aktivne in želijo "vstopiti" v obroč, druge obroč zapuščajo. Možno je tudi, da postaje izpadejo zaradi okvare ali pa se enostavno izklopijo. S tem se logični obroč prekine. V primeru, da izpade postaja, ki ima v lasti žeton, pa se izgubi tudi žeton. Skratka, omrežje vodilo z žetonom mora imeti izdelane mehanizme, ki omogočajo vzpostavitev in vzdrževanje obroča, vstopanje v obroč in izstopanje iz obroča ter nadzor nad podvojenimi in izgubljenimi žetoni.

Omrežje *vodilo z žetonom* je standardizirano s standardom IEEE 802.4. Za fizični nivo omrežja je izbran 75 ohmski koaksialni kabel, podobno kot v sistemih kabelske televizije. Informacijski signal je z modulacijo premaknjen v višji frekvenčni pas (ang. Broadband Transmission). Možne so različne vrste frekvenčnih, faznih in tudi amplitudnih modulacij. Dovoljene so hitrosti prenosa 1, 5 in 10 Mbitov na sekundo. Na kanalu je možnih šest stanj signala, od katerih tri predstavljajo logično nič, logično ena in prazen kanal, tri stanja pa so predvidena za nadzor nad kanalom.

Standardizirani so štirje prioritetni razredi, 0, 2, 4 in 6. Razred 6 ima najvišjo prioriteto. Lahko si mislimo, da je vsaka postaja navznoter (logično) razdeljena na štiri postaje s padajočimi prioritetami. Ko postaja dobi žeton, ga interno preda 'postaji' s prioritetoa šest. Če ta 'postaja' ima pripravljene podatke, jih oddaja, dokler ne odda vseh podatkov ali dokler se ji ne izteče dodeljeni čas oddajanja. Nato (interno) preda žeton 'postaji' z naslednjo nižjo prioriteto. Žeton resnično zapusti postajo šele, ko konča z oddajo 'postaja' s

prioriteto nič. Postaja preda žeton (naprej po kablu) sosednji postaji z nižjim naslovom. S primernim vzdrževanjem časovnikov, je možno dodeliti večji delež kapacitete kanala pomembnejšim podatkov ('postajam' z višjo prioriteto).

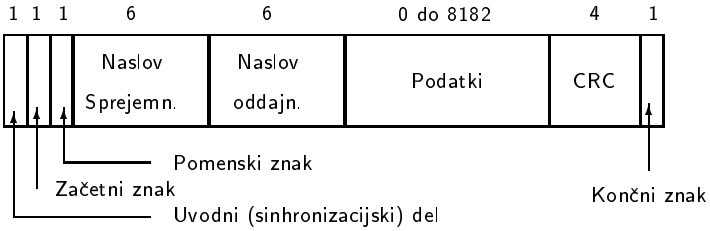
Oblika okvirja v mreži tipa vodilo z žetonom po standardu Ieee 802.4 je podobna okvirju v mreži po standardu 802.3, okvirja pa sta si dovolj različna, da omrežji nista direktno združljivi. Slika 105 prikazuje splošno obliko okvirja. Uvodni del obsega vsaj en zlog in služi sinhronizaciji. Začetni in končni znak (osem bitov) sta drugače kodirana kot podatki, zato se ne moreta pojaviti znotraj okvirja (med podatki). Dolžine okvirja ni potrebno prenašati. Za naslov oddajne in za naslov sprejemne postaje je predvidenih po šest zlogov, enako kot po standardu 802.3. Podatkovni del obsega od nič do 8182 zlogov, za podatkovnim delom pa so še štirje zlogi za ciklično preverjanje okvirja z enakim polinomom kot po standardu 802.3.

Pomenski del okvirja (osem bitov) razlikuje nadzorne okvirje od podatkovnih okvirjev. Pri podatkovnih okvirjih vsebuje pomenski zlog še prioriteto okvirja. Če pa je potrebna potrditev okvirja s strani sprejemne postaje, je v pomenskem delu tudi zahteva za potrditev okvirja. To pomeni, da oddajna postaja prepusti kanal sprejemni postaji - ji začasno preda žeton, da sprejemna postaja lahko odda potrdilo. Potrditev okvirja v splošnem ni vedno potrebna.

Obstaja sedem vrst nadzornih okvirjev. Nadzorni okvir je lahko žeton sam, dovoljenje za vstop nove postaje v logično zanko ali slovo postaje od nje. Z nadzornimi okvirji in standardiziranimi postopki se rešujejo tudi problemi kot so: izgubljen žeton, podvojen žeton, okvare postaj, i.t.d..

Začetna ali ponovna vzpostavitev obroča V mrežo vstopajo najprej postaje z višjimi naslovi, nato z nižjimi naslovi. V tem vrstnem redu (smeri) kroži tudi žeton (od postaj z višjimi naslovi k postajam z nižjimi naslovi). Vsakič, ko postaja ima žeton, sme oddajati vnaprej predpisan čas, nakar mora predati žeton sosednji postaji z nižjim naslovom. Možno je tudi, da v tem času odda zaporedje krajših okvirjev. Če pa nima podatkov, preda žeton naprej takoj.

V začetku obratovanja omrežja ali v primeru, da se izgubi žeton, nima žetona nobena postaja. Ker z oddajo okvirja ne sme sama od sebe začeti nobena postaja, je kanal neaktiven oziroma prazen. Vendar postaje spremljajo dogajanja v omrežju ter ugotovijo, da v kanalu še ni aktivnosti ali pa so zaradi izgube žetona zamrle. Zato ena ali več postaj odda nadzorni okvir (ang. $CLAIM\_TOKEN$), enako kot v omrežju CSMA/CD, s katerim druge postaje obveščajo, da "jamčijo" za žeton. Postaja, ki se polasti kanala in uspešno odda nadzorni okvir z njim obvesti druge postaje, da jamči za žeton. Potem odda podatkovni okvir ali nadzorni okvir, ki pomeni povabilo nove postaje v obroč.

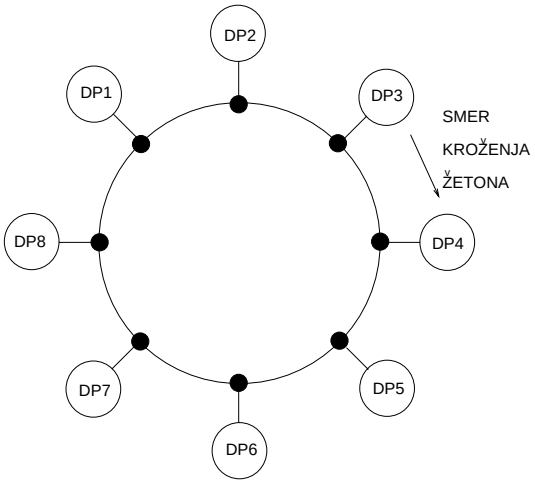


Slika 105: Oblika okvirja v mreži *vodilo z žetonom* po standardu Ieee 802.4.

Vstop nove postaje v obroč Lastnih žetona občasno ponudi postajam, ki niso v obroču, da se vključijo v obroč. V ta namen pošlje nadzorni okvir ustreznega tipa (ang. $SOLICIT\_SUCCESSOR$), v njem pa svoj naslov in naslov naslednika v obroču, to je naslov sosednje postaje z nižjim naslovom. Postaja, ki je v območju med obema naslovoma in želi vstopiti v obroč, sme vstopiti. Če je ena sama postaja s tako željo, potem je vse v redu. Povabljena postaja vstopi v obroč, prevzame njenega naslednika za svojega naslednika, sama pa postane njen nov naslednik in lastnik žetona. Če pa trči več postaj (zahtevo za vstop izrazi več postaj), lastnik žetona sproži postopek razsodbe, katera postaja naj vstopi prej in kdo naj naslednji dobi žeton.

Izstop iz obroča Izstop iz obroča je enostaven. Iztopi lahko le postaja, ki ima žeton. V ta namen pošlje svojemu predhodniku ustrezen nadzorni okvir (ang. $SET\_SUCCESSOR$) in v njem naslov svojega naslednika. Z njim predhodniku prepusti žeton in od njega zahteva, naj vzame njenega naslednika za svojega novega naslednika. Po tem dejanju postaja ni več v logičnem obroču.

Kaj se zgodi, če postaja izstopi iz obroča na neregularen način - brez obvestila? Logični obroč se s tem prekine in preti, da omrežje razpade. Da se to ne zgodi, vsaka postaja, ko odda žeton nasledniku, opazuje aktivnost kanala. Če aktivnosti zamrejo, poskusi ponovno s predajo žetona. Če tudi to ne uspe, pošlje nadzorni okvir (ang. $WHO\_FOLLOWS$), s katerim poizveduje za naslednika svojega naslednika. Naslovljena postaja odgovori z nadzornim okvirjem (ang. $SET\_SUCCESOR$), s katerim lastniku žetona sporoča, da postaja ona nov naslednik. S tem poskrbi za iztop postaje, ki je nepravilno izstopila ali je v okvari. Če pa je tudi sama izstopila ali je v okvari, lastnik žetona vstraja in z ustreznim nadzornim okvirjem nadaljuje poizvedovanje, če je sploh še kdo aktiven.



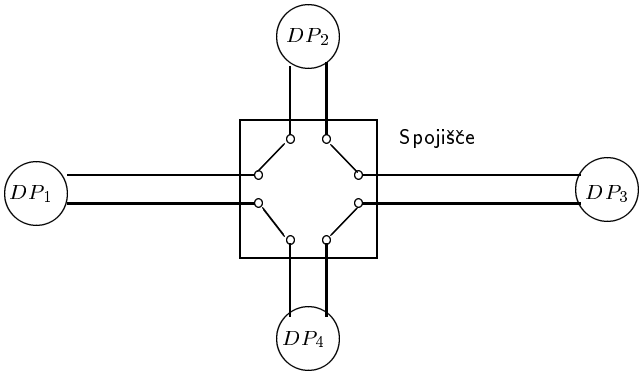
Slika 106: Mreža topologije obroč.

5.7.3 Obroč z žetonom in IEEE 802.5

Lokalna omrežja *obroč z žetonom*[23] je po obliki obroč. Obroč je pogosta oblika omrežja tudi v omrežjih velikih geografskih razsežnosti (ang. Wide Area Netoworks - WAN). Obstaja več omrežij tipa obroč, lokalno omrežje obroč z žetonom je eno od njih. Za omrežje topologije obroč je značilno, da so postaje oziroma vmesniki postaj krožno vezani eden z drugim s povezavo točka-točka (ang. Point-To-Point). Postaje je možno povezati z običajnimi telefonskimi ali oklopljenimi paricami, koaksialnimi ali optičnimi kabli. Tehnologija obroča je skoraj popolnoma digitalna, medtem ko zahteva CSMA/CD precej analognih kompenent, recimo za odkrivanje trčenj. Obroč je nepristranski do postaj in jim omogoča popolnoma deterministično dostop do kanala. Zaradi teh razlogov je IBM izbral obroč za svojo lokalno omrežje, IEEE pa ga je standardardiziral v standardu z oznako 802.5. Na sliki 106 je shematično prikazana mreža, ki je po obliki obroč. Postaje so na omrežje vezane preko vmesnikov. Vmesniki postaj pa so vezani v obroč. Vsak vmesnik ima medpomnilnik enega bita. Vsak bit informacije, ki pride do vmesnika, se prepiše v njegov medpomnilnik. Ko je bit v medpomnilniku, ga postaja lahko preveri (sprejme) ali spremeni in potem pošlje nazaj v obroč (odda). Vmesnik torej vnaša kasnitev enega bita.

Slaba stran vsakega obroča je, da mreža razpade, če se kabel kjerkoli prekine. Poleg tega obroč po obliki ni najbolj primeren za industrijska okolja. Problem se da rešiti s skupnim spojiščem, kot je narisano na sliki 107. Logično je omrežje še vedno obroč, fizično pa je zvezda (Ang. Star-shaped-ring). Večina omrežij tudi v resnici uporablja središčno spojišče. Prednosti takega povezovanja postaj v omrežje se pokažejo posebno tedaj, ko imamo v omrežju več takšnih spojišč.

[23]tudi obroč s podajanjem žetona (ang. Token-Passing Ring)



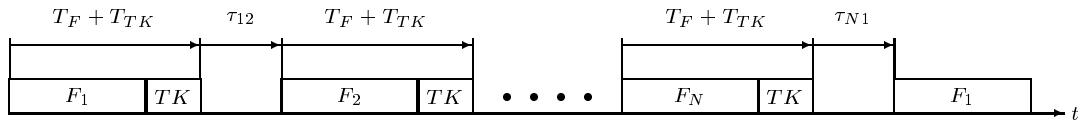
Slika 107: Omrežje obroč z žetonom oblikovano v zvezdo.

V obroču z žetonom kroži v obroču poseben bitni vzorec, ki mu pravimo žeton. Ko nobena postaja nima želje po oddaji, je obroč prost in po obroču enakomerno kroži žeton. Če postaja želi oddajati, se mora prej polastiti žetona. Postaja počaka, da žeton pride do nje, vzame žeton iz obroča oziroma spremeni žeton v začetek okvirja in prične z oddajo vsebine okvirja. Ker je v obroču samo eden žeton, ima pravico do oddaje v nekem obdobju samo ena postaja - tista, ki ima v lasti žeton. Trčenje dveh ali več postaj je zato nemogoče. Zaradi fizične izvedbe kanala pride oddana informacija po zakasnilnem času naokrog tudi do oddajne postaje. Oddajna postaja ima takrat možnost, da primerja oddano informacijo s sprejeto informacijo, lahko pa jo preprosto zavrže. To omogoča sprejemni postaji razmeroma enostavno potrjevanje pravilno sprejetih okvirjev in oddajni postaji preverjanje pravilnosti prenosa. Ko oddajna postaja odda zadnji bit informacije (konec okvirja), obnovi žeton in se preklopi na sprejemanje. Da se kasnitev v obroču ne spreminja, mora biti v obeh načinih delovanja vmesnika (sprejem/oddaja) enaka kasnitev.

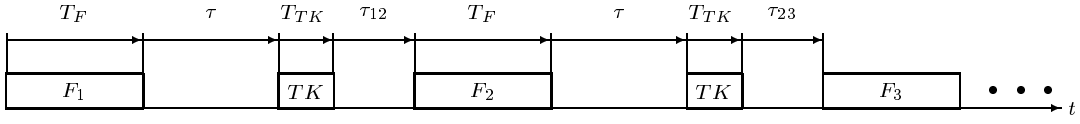
Kadar omrežje ni obremenjeno, se večino časa v obroču nahaja žeton. Ko pa obremenitev narašča in postaje čakajo na oddajo, se žetona polasti naslednja (sosednja) postaja takoj, ko prejšnja postaja konča z oddajo in obnovi žeton. V močno obremenjeni mreži zato pravica do oddaje enakomerno kroži od postaje do postaje in kanal je maksimalno izkoriščen. To je važna lastnost in prednost mreže obroč z žetonom. Kot vemo, so pri mreži tipa CSMA/CD razmere ravno obratne: prepustnost mreže z obremenitvijo zaradi trčenj hitro upada.

Eden od parametrov obroča z žetonom je maksimalni držalni čas žetona (ang. Token Holding Time). V tem času lahko lastnik žetona odda okvir ali zaporedje krajših okvirjev, po tem času pa je obvezen sprostiti žeton. Z daljšanjem držalnega časa žetona se izkoristek omrežja sicer veča, čakalni čas na oddajo pa narašča.

Obstajata dva načina sproščanja žetona: sprostitev žetona takoj po oddaji



Slika 108: Sproščanje žetona takoj po oddaji okvirja.



Slika 109: Sproščanje žetona po sprejemu.

okvirja (ang. Release After Transmission) ali s kratico RAT in sprostitev žetona po sprejemu (ang. Release After Reception) ali s kratico RAR. Časovne razmere v omrežju s sprostitvio žetona po oddaji okvirja prikazuje slika 108. Postaja DP_1 odda okvir F_1 in takoj za njim žeton TK. Po kasnilnem času τ_{12} pride žeton do naslednje postaje v obroču, ki odda okvir F_2 in nato žeton. Zadnja postaja DP_N preda žeton spet prvi postaji. Prva postaja dobi ponovno žeton po času $T_s = N \times T_F + N \times T_{TK} + \tau_{12} + \tau_{23} + \cdots$. V času T_S porabimo $N \times T_F$ časa za prenos koristnih okvirjev. Vsota kasnilnih časov je enaka skupnemu kasnilnemu času obroča τ. Izkoristek obroča s takim sproščanjem žetona zato je:

$$E_{TB.RAT} = \frac{N \times T_F}{N \times T_F + N \times T_{TK} + \tau} = \frac{1}{1 + \frac{T_{TK}}{T_F} + \frac{1}{N}\frac{\tau}{T_F}} \approx \frac{1}{1 + \frac{1}{N}\frac{\tau}{T_F}},$$

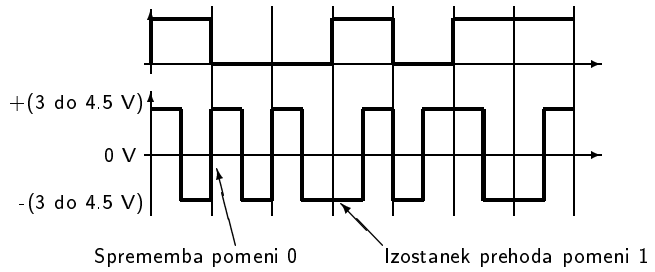
kjer smo v aproksimaciji upoštevali, da je $T_{TK} << T_F$. Čim daljši je okvir oziroma držalni čas žetona, tem boljši je izkoristek.

Pri načinu sproščanja žetona po sprejemu lastnik žetona po oddaji okvirja počaka, da pride oddani okvir naokrog po obroču v celoti nazaj, šele nato pošlje žeton. Časovne razmere so skicirane nas slike 109. Postaja dobi ponovno žeton po času $T_S = N \times (T_F + \tau) + N \times T_{TK} + \tau_{12} + \tau_{23} + \cdots$. Od tega traja prenašanje okvirjev čas $N \times T_F$ in izkoristek je

$$E_{TB.RAR} = \frac{N \times T_F}{N \times (T_F + \tau) + N \times T_{TK} + \tau} \approx \frac{1}{1 + \frac{N+1}{N}\frac{\tau}{T_F}}$$

kar za velike kasnitve (obsežna omrežja) lahko postane bistveno manj od $E_{TB.RAT}$. Vendar pa se da v tem primeru elegantno realizirani potrjevanje okvirjev.

Standard Ieee 802.5 določa za prenosni medij oklopljeno parico, za hitrost prenosa 1 do 4 Mb/s, ali s 16 Mb/s po parici ali optičnem kablu. Za obliko signala je izbran diferencialni Manchester (slika 110).

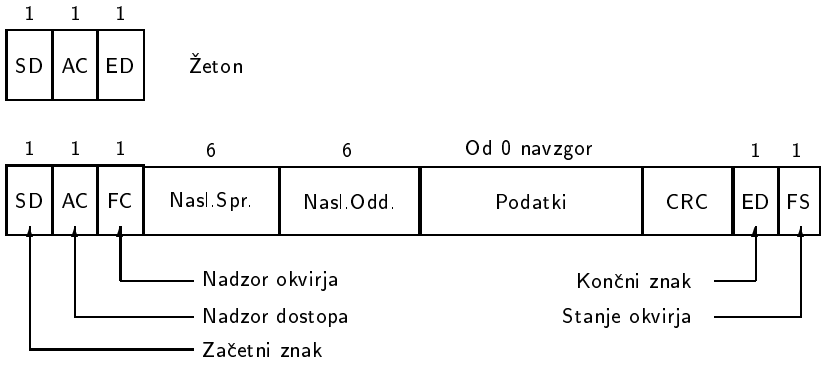


Slika 110: Oblika signala 'diferencialni Manchester'.

Za dostop do kanala skrbi bitni vzorec (žeton) dolžine $3 \times 8 = 24$ bitov. Ko je kanal prazen, žeton kroži po obroču in čaka, da se ga polasti katera od postaj. Postaja, ki želi oddajati, se polasti žetona tako, da komplementira enega od bitov v drugem (AC) zlogu žetona. To dejanje spremeni prva dva zloga žetona v napoved okvirja. Postaja nato odda ostanek okvirja, kot je predočeno na sliki 111. Začetni znak SD (Ang. Starting Delimiter) in končni znak ED (ang. Ending Delimiter) sta kodirana drugače od podatkovnih bitov. Zato ni nevarnosti, da bi podatkovni zlog zamenjali s katerim od teh dveh znakov. Polje, ki bi označevalo dolžino okvirja, ni potrebno. Okvirji po standardu IEEE 802.5 niso združljivi z okvirji po standardih IEEE 802.3 in 802.4, združljivost pa velja vsaj za obe polji naslova in za ciklično preverjanje okvirjev (CRC).

Standard 802.5 ima dobro premišljen prioritetni sistem postaj. Žeton vsebuje v polju AC označeno prioriteto žetona. Samo postaja, ki želi poslati okvir z enako ali višjo prioriteto od prioritete žetona, se lahko polasti žetona. Ko gre podatkovni okvir skozi vmesnik postaje, ima postaja možnost napovedati zahtevo za oddajo okvirja tako, da v napovedni del okvirja (polje AC) vpiše zahtevano prioriteto. Vendar sme postaja to storiti (rezervirati žeton) samo, če napovedno polje še ne vsebuje napovedi višje prioritete. Ko je tekoči okvir končan, lastnik žetona generira žeton z rezervirano prioriteto in žetona se slej ko prej lahko polasti tisti, ki ga je rezerviral. Da ne pride do zastoja, je vsaka postaja, ki je dvignila prioriteto žetona, dolžna prioriteto spustiti na prvotno vrednost. Skratka, po oddaji svojega okvirja obnovi žeton s prvotno prioriteto.

V normalnih pogojih pride prvi bit okvirja naokrog po obroču do oddajne postaje še preden je okvir v celoti oddan. Oddajna postaja 'posrka' sprejete bite in oddaja naprej do konca okvirja. Postaja sme zadržati žeton v svoji lasti največ za 10 milisekund. Če en okvir traja manj časa, sme oddati tudi več okvirjev zapored. Po tem času mora postaja prepustiti žeton drugi postaji. Ko konča z oddajanjem ali ji poteče dodeljeni čas in sprejme še zadnji bit zadnjega okvirja (torej RAR), odda še žeton.



Slika 111: Žeton (zgoraj) in okvir (spodaj) v obroču z žetonom Ieee 802.5

Zanimiv je pomen zadnjega zloga FS (Ang. Frame Status). V njem sta med drugimi dva bita z oznako A in C. Ko gre okvir skozi vmesnik namembne (sprejemne) postaje, postaja postavi bit A v stanje ena. Če postaja ne obratuje, se to ne zgodi in oddajna postaja je o tem obveščena, ko jo doseže še zadnji zlog okvirja naokrog po obroču. Če namembna postaja uspešno prepiše okvir iz svojega vmesnika (sprejme okvir), postavi bit C, ki ga preveri oddajna postaja, podobno kot bit A. Ker bajt FS ni zajet v ciklično preverjanje, sta oba bita v nadzornem zlogu zaradi zanesljivosti podvojena. Končni zlog (ED) vsebuje bit E, ki je postavljen, če katerikoli vmesnik odkrije napako). Vsebuje tudi bit, ki lahko označuje zadnji okvir v daljšem sporočilu.

Vzdrževanje obroča je za razliko od vodila z žetonom *centralizirano*. Vzdrževanje obroča v danem obdobju opravlja samo ena od postaj imenovana *monitor*. Vsaka postaja je sposobna postati monitor. Če monitor izpade, ga zamenja ena od postaj. Za vzdrževanje obroča se monitor poslužuje nadzornih okvirji. Obstaja šest vrst nadzornih okvirjev, ki so kodirani s poljem FC. Nadzorni okvirji služijo za inicializacijo obroča, ugotavljanje prekinitev obroča, ugotavljanje prisotnosti postaj z enakim naslovom in izbiranje monitorja. V polju FC okvirja je predviden tudi bit, ki ga monitor ob prehodu skozi njegov vmesnik periodično postavlja (ang. Active Monitor Present). Če se to ne zgodi, udeleženci obroča to kmalu ugotovijo in skušajo postati monitor. Ko hoče katera od postaj postati monitor, odda ustrezen nadzorni okvir (ang. $CLAIM\_TOKEN$). Čim pride oddani okvir po obroču nazaj, postane postaja aktivni monitor. Naloga monitorja je tudi izločanje opuščenih okvirjev (okvirjev, ki nimajo aktivne niti oddajne niti sprejemne postaje).

5.7.4 Predalčni obroč

Obroč z žetonom ni edina možnost za izvedbo obroča. V lokalnih omrežjih je priljubljena vsaj še ena izvedba obroča imenovana *predalčni obroč* (Ang. Sloted

Ring). Omrežje predalčni obroč je dobilo tako ime zato, ker so vsi okvirji v obroču enake in stalne dolžine. V obroču se lahko istočasno nahaja več okvirjev. Zato so v obroču namenoma povečane kasnitve. Kasnitve se da brez težav povečati z vgradnjo pomikalnih registrov v vmesnike postaj. Namesto kasnitve enega bita, kot v obroču z žetonom, so v predalčem obroču prisotne daljše kasnitve. Vsak okvir v obroču ima poseben bit, ki označuje ali je okvir poln ali prazen (prost). Ko postaja želi oddati okvir, mora počakati, da pride prazen okvir naokrog do nje. Tak okvir postaja označi kot poln in vstavi podatke. Ko okvir prispe do namembne postaje, ta prevzame podatke in obrne bit oznake - sprosti okvir. Tudi v predalčnem obroču je možno prav elegantno realizirati nadzor nad tokom podatkov med oddajno in sprejemno postajo (preverjanaje napak, potrjevanje okvirjev, i.t.d.).

5.7.5 Primerjava lokalnih omrežij IEEE 802

Primerjati omrežja IEEE 802.x (x=3,4,5) med seboj je nehvaležno delo. Vedno se da najti takšne okoliščine in kriterije, ob katerih se eno od omrežij bolje obnese od ostalih. Postavitev, vzdrževanje, spreminjanje omrežja tipa IEEE 802.3 je lažje od ostalih dveh. Omrežjem IEEE 802.3 (in Ethernet) se navadno očita naključen dostop in nevarnost nepredvidljivo dolgega čakanja. Zato naj bi bila manj primerna za časovno-kritične primere uporabe. Nekaj je gotovo: močno obremenjeno omrežje IEEE 802.3 se ne obnese. Tudi za zelo visoke hitrosti prenosa ni priporočljivo. Takšna okolja torej niso za CSMA/CD in primernejša so omrežja z žetonom. Najdaljši čakalni čas je v omrežjih z žetonom vnaprej znan, z naraščanjem obremenitve pa prepustnost celo raste do največje možne vrednosti. Čisto drugače pa je v malo obremenjenih omrežjih. V neobremenjenih omrežjih z žetonom morajo postaje po nepotrebnem čakati na žeton, čeprav bi bila možna takojšnja oddaja. V omrežju CSMA/CD je dostop do kanala tedaj praktično takojšen. Naslednja vprašanja v omrežjih z žetonom so povezana z (ponovno) vzpostavitvijo obroča, izgubo žetona in podobnimi okoliščinami, ki se običajno javljajo prav v najbolj kritičnih trenutkih. Analize kažejo, da so razmere bolj nepredvidljive, kot so zagovorniki žetona pripravljeni priznati.

Lokalna omrežja v industrijskih okoljih so izrazito nizko obremenjena. Meritve obremenitve v takšnih omrežij so pokazale [5], da se povprečna obremenitev giblje v razredu nekaj odstotkov (3 % do 5 %), vršna obremenitev pa redko preseže 12 %. Rezultat je torej v prid omrežjem CSMA/CD. Še več,izčrpne analize in simulacije časovno kritičnih primerov uporabe so pokazale, da se CSMA/CD v povprečju bolje obnese od ostalih dveh tudi pri bistveno višjih obremenitvah (do 40 %). Kasnilni čas v omrežju CSMA/CD je krajši vse do obremenitve 60 %, pri konstantni obremenitvi do 55 % pa CSMA/CD zagotavlja boljši odziv. V omrežjih, kjer je enako pomemben maksimalni odzivni čas in variabilnost odzivnh

časov, pa so se vsa omrežja izkazala za ekvivalentna.

5.7.6 Most in lokalna omrežja

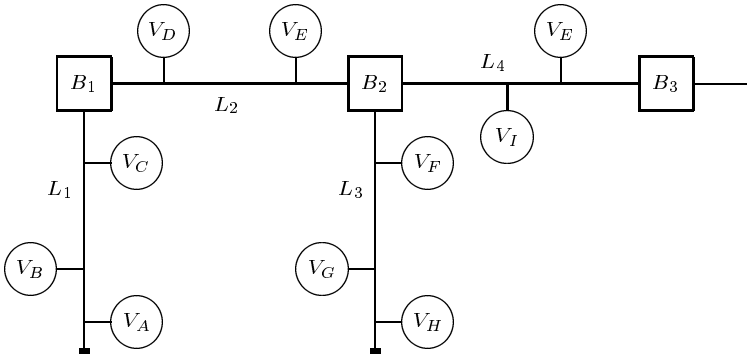
Most (ang. Bridge) je komunikacijska naprava, ki v omrežju služi kot posrednik informacije. Za razliko od ponavljalnika (ang. Repeater), ki je naprava fizičnega sloja in deluje na nivoju digitalnih signalov, pa je most naprava drugega, to je podatkovnega sloja in deluje na nivoju okvirjev. Most sprejme in začasno shrani okvir, mu po potrebi spremeni obliko in pošlje naprej, če je to potrebno. Pravimo, da opravlja funkcijo 'filtriranja' okvirjev. Pri svojem delovanju most upošteva obliko in notranjost okvirja, vendar ga podatkovni del okvirja ne zanima. Podatkovni del okvirja ostane pri prehodu skozi most nedotaknjen.

Tipični razlogi za vgradnjo mostu ali več mostov v omrežje so: povezovanje obstoječih, samostojno zgrajenih omrežij, povečanje razsežnosti omrežja, povečanje števila postaj, povečanje prepustnosti omrežja, povečanje zanesljivosti delovanja, povečanje varnosti in tajnosti. Lokalna omrežja so razmeroma cenena in ne redko nastajajo ločeno in neodvisno eno od drugega, pa čeprav v istem podjetju ali celo v isti zgradbi. Razumljivo se s časom pojavi potreba po povezovanju takšnih omrežij v skupno omrežje. Omrežja, ki jih povezujemo z mostovi, so od vključno tretjega sloja navzgor običajno popolnoma identična (enaki protokoli na vseh slojih). Z mostom lahko povezujemo omrežja, ki so na spodnjih dveh slojih enakega ali različnega tipa. Povezovanje omrežij enakega tipa, recimo omrežje po standardu IEEE 802.3 z enakim omrežjem, je neprimerno manj zahtevno od povezovanja omrežij različnih tipov. V prvem primeru se oblika okvirja ne spremeni in most okvirje resnično samo filtrira. Pri prehodu iz omrežja IEEE 802.3 v omrežje IEEE 802.4 ali podobno pa se mora spremeniti tudi oblika okvirja, velikost okvirja, hitrost prenosa, frekvenčni pas signala, nekaterih funkcij pa se enostavno ne da realizirati (npr. prioritetnega sistema).

Most je večkrat potreben tudi zaradi širjenja istega omrežja. Lastnik lokalnega omrežja je največkrat hkrati tudi uporabnik in ko s časom ugotovi, da omrežje več ne ustreza potrebam, ga hoče razširiti. Če obseg omrežja presega dovoljeno dolžino enega segmenta ali dovoljeno število priključkov na segment, je potreben most.

Most je primeren tudi tedaj, kadar obremenitev omrežja tako naraste, da se začne manjšati njegova prepustnost. Za lokalna omrežja so značilne razmeroma intenzine komunikacije znotraj delovnin skupin, navzven pa je komunikacij bistveno manj. S primernim nameščanjem mostov se da omejiti širjenje okvirjev samo na del omrežja, tako da lokalne komunikacije po nepotrebnem ne obremenjujejo celega omrežja.

Z vgrajevanjem mostov se veča tudi zanesljivost delovanja omrežja. Če se



Slika 112: Lokalno omrežje z več segmenti in mostovi B_1, B_2 in B_3.

prekine segment, ki je od ostalega omrežja ločen z mostom, to ne moti obratovanja ostalega dela omrežja. Z mostovi se da tudi omeji širjenje okvirjev z zaupno informacijo, ki bi jih lahko sicer prestregel vsiljivec.

Uporabljata se dve vrsti mostov, prozorni most (ang. Transparent Bridge) in most z usmerjanjem izvora (ang. Source Routing Bridge). Prozorni most se je uveljavil v omrežjih po obliki vodilo (CSMA/CD - IEEE 802.3 in vodilo z žetonom - IEEE 802.4), most z usmerjanjem izvora pa v obroču IEEE 802.5. Poglejmo najprej, kako deluje prozorni most.

Prozorni most Prozoren most se sam prilagaja spremembam v omrežju. Tako ime ima zato, ker je za uporabnike praktično neviden. Delovanje mostu bomo razložili s pomočjo slike 112.

Mislimo si, da postaja V_A pošilja okvir postaji V_B. Ker sta V_A in V_B na istem segmentu L_1, ga most ne pošlje na naslednji segment. Sedaj pa vzemimo, da postaja V_A pošilja okvir postaji V_F. Ker V_F ni na istem segmentu kot V_A, most B_1 pošlje okvir na segment L_2. Pri tem most B_1 ne zanima, ali se V_F zares nahaja na segmentu L_2 ali ne. Posredovanje okvirja naprej na segment L_3, kjer se nahaja postaja V_F, je naloga mostu B_2. Seveda mora most B_2 vedeti, da V_F ni na segmentu L_2 in tudi, na katerem segmentu je.

Zastavlja se vprašanje, kako mostovi ugotovijo in sicer brez posredovanja uporabnika, kdaj naj okvir zadržijo in kdaj naj ga pošljejo naprej ter tudi v katero smer. V ta namen most vzdržuje usmerjevalno tabelo. V začetku obratovanja mostu je tabela prazna. Most ne ve niti kje so postaje niti kje so drugi mostovi. To mora šele ugotoviti. Ko pride okvir za neznano postajo do mostu, ga most pošlje naprej v vse smeri. Temu rečemo preplavljanje. Z obratovanjem pa se most uči in začne pošiljati okvirje naprej in samo tja, kamor je potrebno. Delovanje mostu lahko povzamemo v naslednjem algoritmu.

če je naslovljenec na istem segmentu kot pošiljatelj **potem**
    zavrži okvir
sicer če je segment naslovljenca znan **potem**
    pošlji v to smer
sicer
    pošlji v vse smeri.

Most se uči iz prišlih okvirjev. Algoritmu učenja pravimo "povratno učenje" (ang. Backward Learning). Ko na primer most B_1 sprejme okvir od postaje V_A, ki je na segmentu L_1, preveri naslov izvora (pošiljatelja) in naslov ponora (naslovljenca). Če naslovljenca ne pozna, pošlje okvir naprej v vse smeri. Če pa ne pozna pošiljatelja (torej V_A), vpiše v svojo usmerjevalno tabelo, da se V_A nahaja na segmentu L_1. Ko v prihodnosti dobi okvir za postajo V_A, pošlje okvir na segment L_1 ali pa ga zadrži na tem segmentu. Po nekem začetnem času učenja si vsi mostovi zgradijo usmerjevalne tabele. Da tabele zaradi morebitnih sprememb v omrežju s časom ne zastarajo, se vodi časovni nadzor. Po predvidenem času obratovanja most ponovi postopek učenja.

Most z usmerjanjem izvora Kot pove ime, sodeluje pri tej obliki usmerjanja pošiljatelj, most pa sledi dogovorjenim pravilom. Vsak segment in vsak most v omrežju ima svojo oznako oziroma naslov. Kadar postaja odpošlje okvir, vpiše v okvir ne le naslov končne postaje, ki ji je okvir namenjen, ampak celotno pot. Pot je zaporedje naslovov, v katerem si izmenoma sledijo naslovi segmentov in mostov. Na primer pot od postaje V_A do postaje V_F je (slika 112):

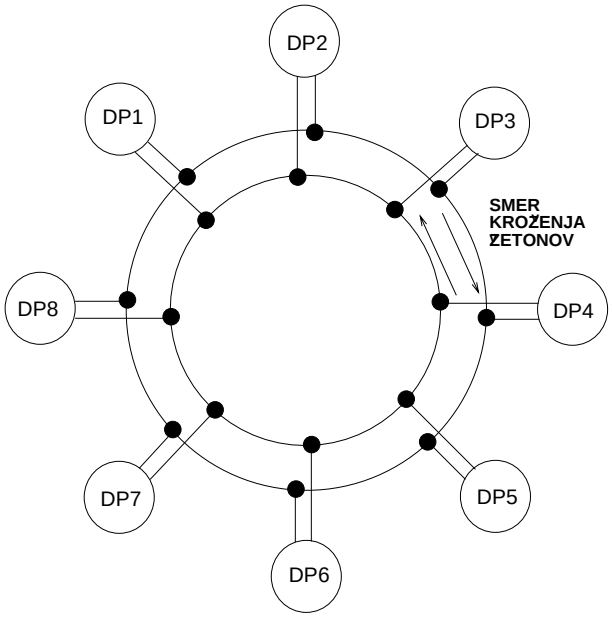
$$L_1 \;\; B_1 \;\; L_2 \;\; B_2 \;\; L_3$$

Mostove zanimajo samo okvirji s postavljenim domenjenim naslovnim bitom. Ko most dobi tak okvir preveri pot. Če v poti najde svoj naslov za naslovom segmenta, s katerega prihaja okvir, pošlje okvir na segment z naslovom, ki sledi njegovemu naslovu. Na primer, v našem primeru naslov u mosta B_1 sledi naslov segmenta L_1. Če most B_1 dobi okvir s segmenta L_1, ga pošlje naprej na segment L_2, sicer pa ga zavrže.

Ponovno se zastavlja vprašanje, kako naj postaje ugotovijo poti do ostalih postaj v omrežju. Temu služijo poizvedovalni okvirji. Če izvor (pošiljatelj) ne pozna poti do ponora (naslovljenca), pošlje poizvedovalni okvir. Ko most sprejme tak okvir, vanj vpiše svoj naslov in ga s pošlje naprej v vse smeri. Tak okvir se tako "s preplavljanjem" širi povsod po omrežju. Končno pride do iskane postaje. Pravzaprav pride do iskane postaje po različnih poteh več kopij poizvedovalnega okvirja. Pot potovanja poizvedovalnega okvirja je razvidna iz vsebine okvirja in postaja se lahko odloči za najprimernejšo pot in jo vrne izvoru po izbrani poti.

5.8 FDDI

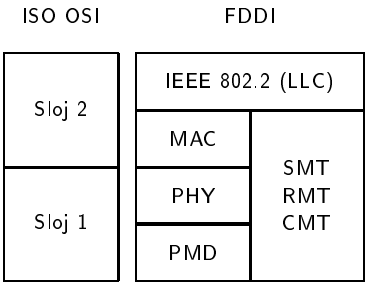
FDDI (Fiber Distributed Data Interface) je omrežje srednjih razsežnosti (MAN). Namenjeno je predvsem za povezovanje manjših, krajevo ločenih, lokalnih omrežij (LAN). FDDI je bil najprej ameriški standard (ANSI X3.T9), dokler ga ni kasneje sprejel ISO (ISO 9314). FDDI-II poleg običajnega paketnega načina prenosa podatkov obvladuje tudi isohroni promet, ki je vezan na stvarni čas (npr. zvok in slika).



Slika 113: Dvojni obroč omrežja FDDI. Žetona krožita v nasprotnih smereh.

Omrežje FDDI ima obliko dvojnega obroča, kot je prikazano na sliki 113. V primeru napake (prekinitve ene od povezav) je možen prehod na delovanje v enojnem obroču. Dostop do kanala ureja žeton. S tega stališča je omrežje podobno omrežju IEEE 802.5, vendar omogoča je zaradi višje hitrosti prenosa podatkov prepustnost bistveno večja. Ker postaje sproščajo postaje takoj po oddaji (princip RAT), se lahko po obroču sočasno prenaša več okvirjev. Obhodni čas žetona je navzgor omejen, kar prioritetnim podatkom jamči hitrejši odziv. Omrežje deluje s hitrostjo 100 Mb/s in dopušča za posamezen obroč obseg do 100 Km. V omrežju je lahko do 500 postaj, razdalja med sosednjimi vozlišči pa ne sme presegati 2 Km. Za prenosno sredstvo je uporabljeno optično vlako, možna pa je tudi uporaba parice. Zahtevano hitrost in razdaljo se da doseči z večrodovnimi vlakni (Multi-mode fiber). Ta so cenejša od kvalitetnejših enorodovnih vlaken (Single-mode fiber), zato se multirodovna vlakna 62.5/125 (62.5 je premer sredice in 125 je premer obloge v mikronih) z valovno dolžino svetlobe 1300 nm tudi največ uporabljajo.

Omrežje FDDI pozna dva tipa vozlišč: tip A in tip B. Vozlišče tipa A



Slika 114: Primerjava arhitekturne zgradbe omrežja FDDI z modelom OSI.

dopušča dvojno priključitev oziroma je vezano na oba obroča. Vozlišče za enojno priključitev (tip B) je vezano samo na enega izmed obročev. Vozlišče, ki je vezano v enega ali v oba obroča, polega tega pa je nanj možno priključiti vsaj še eno dodatno vozlišče, imenujemo koncentrator. Druga vozlišča so končna vozlišča oziroma postaje.

Standard FDDI določa štiri sestavine omrežja, glej sliko 114:

- MAC (Media Access Control) določa dostop do medija, vključno z obliko okvirjev, nadzorovanjem žetona, naslavljanjem, CRC preverjanjem in ukrepanjem v primeru napak,
- PHY (Physical Layer Protocol), ki določa obliko signalov (kodiranje signala), okvirjenje in sinhronizacijo,
- PMD (Physical Layer Medium Dependent), ki predpisuje zahtevane lastnosti in vrsto prenosnega sredstva ter konektorjev,
- SMT (Station Management) za upravljanje postaj, obroča (RMT - Ring Management) ter povezav (CMT - Connection Management), vključno z vstopanjem in izstopanjem postaj iz obroča, inicializacijo obroča in zbiranjem statističnih podatkov.

Okvir in žeton sta po obliki in tudi pomenu skoraj enaka kot v omrežju IEEE 802.5, drugačno pa je kodiranje signala. Za kodiranje signala je izbrana NRZI oblika. NRZI signal se v primeru daljših neprekinjenih zaporedij ničel malo spreminja, to pa zmanjša sposobnost sinhroniziranja sprejemnika z oddajnikom. Da ne pride do izpada sinhronizacije, se NRZI signal izpopolni s 4B/5B kodiranjem tako, da se 4 informacijski binarni znaki nadomestijo s petimi, pri čemer so 5-bitne kombinacije izbrane tako, da si v nobenem primeru zaporedoma ne sledijo več kot tri ničle. Peterko binarnih znakov imenujemo simbol in predstavlja osnovno enoto za prenos v FDDI omrežju. Pred dodatnih omejitev je s

petimi biti možnih 32 kombinacij. Od teh se jih 16 rabi za kodiranje koristnih (informacijskih) bitov, osem kombinacij služi za nadzor, ostale pa so neveljavne.

FDDI loči dve vrsti podatkov: sinhrone podatke (okvirje) in asinhrone podatke. Za podatke asinhronega značaja kasnilni čas ni kritičen, medtem ko moramo podatkom sinhronega značaja zagotovljati, da pridejo pri oddaji pravočasno na vrsto. To se doseže s časovnim nadzorom žetona. V času inicializacije obroča se vozlišča dogovorijo za vrednost parametra TTRT (Target Token Rotation Time), ki ima naslednji pomen. Če vozlišče sprosti žeton v trenutku t, potem naslednjič, ko dobi žeton, po času $t + TTRT$ ne sme začeti oddajati asinhronih podatkov, lahko pa odda sinhrone podatke. Parameter $TTRT$ je izbran tako, da je seštevek intervalov, ki jih postaje dobijo za oddajo sinhronih podatkov, manj kot $TTRT$. Na ta način FDDI jamči, in to bomo tudi pokzali, da dejanski obhodni čas žetona RTRT (Real Token Rotation Time) nikoli ne preseže $2 \times TTRT$. Pred tem pa se prepričajmo, da je obhodni čas žetona ko ni sinhronih podatkov nikoli ne preseže $TTRT + T_F + \tau$, kjer je čas T_F čas prenašanja najdaljšega okvirja in τ je čas obhoda žetona v praznem obroču (kasnilni čas žetona). Ob prvem obhodu žetona skozi vozlišče se v vsakem vozlišču nastavi časovnik na nič. Ko pride žeton po času τ naokrog, lahko vozlišče začne z oddajo. Naj vozlišče V_0 dobi žeton ob času t_0, oddaja čas T_{F0}, nakar odda žeton vozlišču V_1. Vozlišče V_1 (ponovno) dobi žeton po času $t_1 = T_{F0} + \tau$. Denimo, da je $t_1 < TTRT$ in V_1 lahko začne z oddajo. Naj oddaja čas T_{F1} in potem odda žeton. Žeton pride do vozlišča V_2 po času $t_2 = t_1 + T_{F1}$. V primeru, da je $t_2 < TTRT$ lahko začne z oddajo. Naj bo t_2 za malenkost manjši od $TTRT$, zato V_2 lahko začne z oddajo okvirja v trajanju $T_{F2} = T_F$ in ga odda do konca. Zato pride žeton do vozlišča V_3 kasnjen za čas $t_3 = TTRT + T_F$, ki ga mora po dogovoru predati naprej brez oddaje. Interval med dvema zaporednima prihodoma žetona do katerega koli vozlišča zato ne more biti daljši od $TTRT + T_F + \tau$.

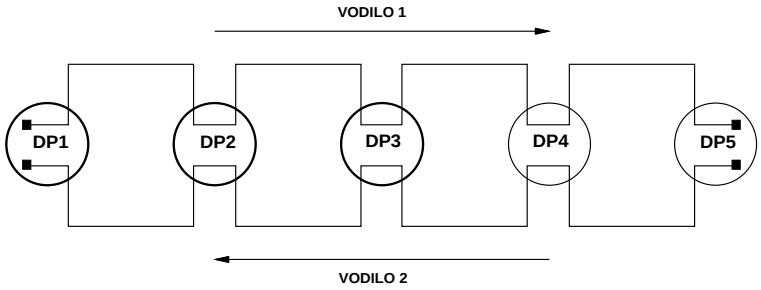
V primeru, da prenašamo samo sinhrne podatke (ni asinhronih podatkov), bo obhodni čas žetona (po definiciji) krajši od $TTRT$. Ko pa prenašamo sinhrone in asinhrone podatke, bo obhodni čas žetona seštevek obeh. Če sinhroni podatki v seštevku trajajo $TTRT - T_F - \tau$), bo skupen kasnilni čas $2 \times TTRT$.

Izkoristek omrežja FDDI bomo ocenili na naslednji način. Denimo, da je obroč polno obremenjen in pride žeton naokrog v času $2 \times TTRT$. V tem času se večinoma prenašajo koristni podatki, nekaj časa pa se izgubi za prenos žetona, ki traja čas T_{TK}, in nekaj k izgubi prispeva kasnitev obroča. Naj bo kasnitev obroča τ in naj bo v omrežju N postaj. Izkoristek potem je:

$$E_{FDDI} = \frac{2 \times TTRT - N \times T_{TK} - \tau}{2 \times TTRT}.$$

5.9 DQDB in IEEE 802.6

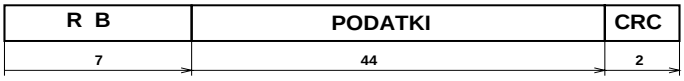
Omrežje po standardu IEEE 802.6 ali DQDB (Distributed Queue Dual Bus) spada med omrežja srednjih razsežnosti (MAN). Topologijo omrežja prikazuje slika 115. Kot pove ime, ima omrežje izgled dvojnega vodila. Po vsakem izmed dveh vodil se prenašajo podatki samo v eni smeri, po enem vodilu v eni smeri, po drugem vodilu v nasprotni smeri. Sam izraz "vodilo" ni čisto dosleden, saj so vozlišča v resnici vezana po principu točka-točka. Vsako vozlišče je vezano na obe vodili.



Slika 115: Dvojno vodilo omrežja DQDB. Okvirji se prenašajo samo v označeni smeri.

Standard DQDB določa samo delovanje MAC podsloja. Dostop do kanala je praktično popolnoma decentraliziran. Način streženja zahtev za oddajo zelo dobro aproksimira načelo "prvi pride prvi trežen". Videti je namreč, kot bi se zahteve za oddajo druga za drugo nabirale v strežni vrsti tipa FIFO ter se stregle v enakem zaporedju kot so prihajale. V resnici taka vrsta ne obstaja, ker so zahteve porazdeljene (saj jih dajejo postaje) po celem omrežju. Od tu tudi del imena "porazdeljena vrsta" (Ang. Distributed Queue).

Podatki se prenašajo v okvirjih razmeroma majhne dolžine 53 bajtov. Oblika okvirja je razvidna na sliki 116. Poleg 44 bajtov koristnih podatkov vsebuje okvir še 7 začetnih ("čelo") in 2 končna bajta. Znotraj čela sta bita B in R, ki se uporabljata za dostop. Bit B (Busy) v stanju ena označuje, da je okvir poln



Slika 116: Okvir v omrežju DQDB. Bit B (Busy) v glavi okvirja označuje, da je okvir poln. Bir R (Reservation) služi za rezervacijo okvirja.

in nosi koristne podatke ene od postaj. V nasprotnem primeru je okvir prazen in postaje smejo vanj vstaviti svoje podatke. Bit R (Reservation) bit postavi (če še ni postavljen) postaja, ki bi želela oddajati.

Način dostopa bomo opisali za vodilo 1 (slika 115). Dostop na drugem vodilu je simetričen. Najbolj leva postaja na vodilu 1 generira prazne okvirje dolžine 53 bajtov. Kadar ima postaja pripravljene podatke za oddajo in želi oddajati na vodilu 1, to javi na vodilu 2 tako, da postavi bit R v prvem okvirju, ki pride do nje z nasprotne smeri in še nima postavljnega tega bita. Poleg tega vsaka postaja beleži število postaj desno od nje, ki imajo podatke pripravljene za oddajo, pa še niso prišle na vrsto. V ta namen vzdržuje stanje dveh števcev, števca CD (Count Down) in števca RC (Request Counter), ki jih spreminja po sledečem algoritmu. Vedno, kadar pride po vodilu 2 z desne strani do nje okvir s postavljenim R bitom, poveča števec RC za ena. Vedno, kadar pride po vodilu 1 z leve strani do nje prazen okvir, zmanjša ta isti števec za ena, kajti ena od postaj na njeni desni bo lahko oddala v ta okvir. Na ta način prikazuje stanje števca RC število postaj na njeni desni, ki čakajo na oddajo.

Denimo, da v dotični postaji v opazovanem trenutku, ko je stanje njenega števca RC enako N, nastopi zahteva za oddajo. Če naj se zahteve v omrežju strežejo po principu prvi pride prvi strežen, mora N postaj priti na vrsto pred njo. Zato dotična postaja ob nastopu zahteve prepiše vrednost števca RC v števec CD. Ta števec nato zmanjša za ena ob vsakem prehodu praznega okvirja na vodilu 1. Ko pade vrednost števca na nič, odda okvir.

Literatura

[1] U. Black, *Data Communications and Distributed Networks*, Prentice-Hall 1993.

[2] R. Metcalfe, D. Boggs, "Ethernet: Distributed Packet Switching for Local Computer Networks", *Comm. of the ACM*, 19(7), Jul. 1976, pp.395-403.

[3] A. Tanenbaum, *Computer Networks*, third edition, Prentice-Hall 1996.

[4] J. Walrand, *Computer Communications: A First Course*, Pacific Palisades, CA: Asken Associates, 1991.

[5] *Digital Industrial Networks Guidebook*, Digital, Jun. 1988.

6 Industrijska omrežja

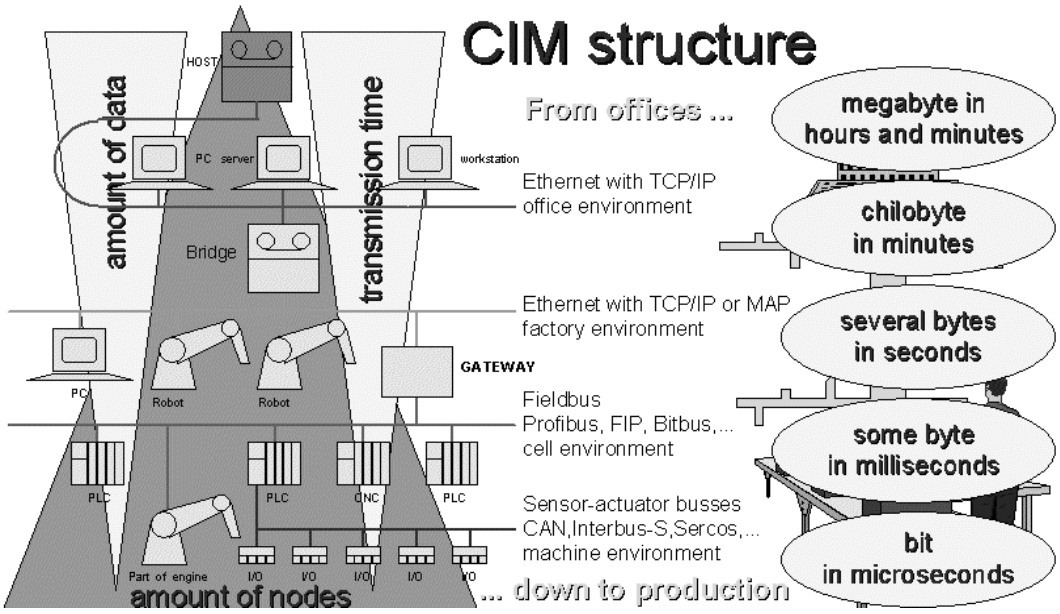
Industrijska omrežja so prilagojena potrebam, ki jih narekuje industrijsko okolje. Največkrat imajo obliko vodila, ali pa jim je vodilo kot način povezovanja vsaj za osnovo. Taka omrežja imajo precej dobrih lastosti, na primer omogočajo splošno ali skupinsko naslavljanje, in nudijo ekonomičen način povezovanja velikega števila razmeroma enostavnih naprav. Skupno ime za omrežja takega tipa je v angleškem jeziku Fieldbus, v nemškem Feldbus, mi pa jih bomo imenovali področna vodila. (Procesna) področna vodila so začela pridobivati na pomenu z razvojem mikroelektronske tehnologije, ki je omogočila gradnjo vse bolj sposobnih ("inteligentnih") in zanesljivih procesnih naprav. Inteligentni senzorji, aktuatorji, krmilniki in podobne naprave lahko prevzamejo nase številne funkcije v zvezi z obdelavo signalov, ki bi se sicer morale opravljati v centralnem računalniku. Sposobni pa so tudi preverjanja pravilnosti lastnega delovanja, parametriranja (umerjanja) na daljavo, in pododbno. Analogne veličine se ne prenašajo analogno (npr. z 4-20 mA merilno zanko), kot na primer v telemetrijskih sistemih, temveč se takoj digitalizirajo, po potrebi predobdelajo in digitalno tudi prenašajo. Porazdeljeno procesiranje skupaj z digitalnim načinom prenosa zagotavlja večjo zanesljivost in prilagodljivost ob manjših stroških.

Procesna področna vodila so v marsičem zelo podobna lokalnim omrežjem, od njih pa se značilno razlikujejo v pomenu informacije, ki jo prenašajo, in v okoljih, kjer se uporabljajo. V tej podobnosti in različnosti z omrežji tipa LAN gre najbrž iskati razlog, da je eden od proizvajalcev omrežij za procesne namene (EcheLon) skoval kratico LON (ang. Local operating Networks). Med omrežja tipa LON bi lahko šteli omrežja LAN, ki so bila prirejena za delovanje v stvarnem času in omrežja, ki so bila razvita nalašč za potrebe industrijske informatizacije. Če je v omrežjih LAN pomembnejša prepustnost od odzivnosti, pa se v omrežjih LON poudarja kratek odzivni čas in zmožnost prioritetnega obravnavanja prenašanih podatkov.

Mrežnih arhitektur in tehnologij, ki jih lahko uvrstimo med procesna področna vodila, je veliko. Tipični predstavniki področnih vodil so Profibus, WorldFIP, P-Net, Interbus-S, CAN, Bitbus, ControlNet, DeviceNet, SDS, Modbus Plus, i.t.d., od katerih ima vsak še podizvedbe, prilagojene specifičnim potrebam določene industrije. Čeprav je pestrost ponudbe in izbire zaželena, pa je po drugi strani področnih vodil tudi preveč, še posebno ko se pokaže potreba po povezovanju naprav in podsistemov različnih proizvajalcev. V praksi se izkaže, da tako imenovana odprtost sistemov največkrat sama po sebi še ni dovolj.

Slika 117 prikazuje, kako se komunikacijska tehnologija vklaplja v model računalniško integrirane proizvodnje (CIM - Computer Integrated Manufacturing). Na sezorskem nivoju imamo razmeroma malo podatkov (signalov), vednar zahtevajo kratek odzivni čas. Proti vrhu piramide količina podatkov narašča,

vendar ti podatki niso časovno kritični (dopuščajo daljše odzivne čase). Visoka prepostnost omrežja je zaželena, a ni obvezna.



Slika 117: Komunikacijska tehnologija v računalniško integrirani proizvodnji.

Vsa omrežja, ki se uporabljajo za prenos procesnih podatkov (precesnih v širšem smislu besede), niso po obliki vodilo. Vendarle pa omrežja te oblike prevladujejo ali pa imajo z vodilom nekaj skupnega. V tabeli 118 vidimo načelno razvrstitev vidnejših predstavnikov mrežnih tehnologij na spodnjih nivojih industrijske informatizacije. Spodnji nivo je podrobneje razslojen na senzorski nivo, nivo naprav in na področni nivo. Vidimo, da nekatere tehnologije, kot na primer LonWorks segajo preko več podnivojev.

Sensor-Actuator Bus (Bit-level)	Device Bus (Byte-level)	Fieldbus (Block-level)
CAN	CAN	IEC 1158/ISA SP50.02
AS-Interface	DeviceNet	FOUNDATION Fieldbus
InterBus Sensor Loop	SDS	Profibus-PA
Seriplex	CAL/CANopen	Profibus-FM
SERCOS	CAN Kingdom	WorldFIP
Sensoplex	InterBus-S	P-NET
	Device WorldFIP (DWF)	Measurement Bus
	FIP IO	Bitbus
	Profibus-DP	
	I/O Lightbus	
	SERCOS	
	MIL-STD-1553	
LonWorks		

Slika 118: Razvrstitev mrežnih tehnologij po namenu uporabe.

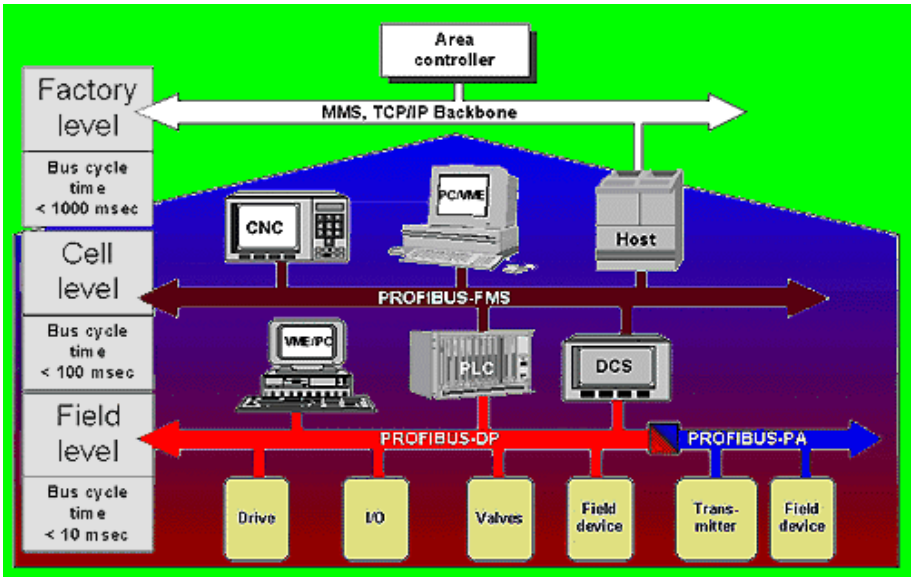
Naslednji seznam zaobsega nekaj najbolj razširjenih omrežnih tehnologij in izvedbe področnih in podobnih omrežij:

- IEC/ISA SP50, mednarodni Fieldbus standard, ki obsega štiri sloje: fizični, podatkovni, aplikacijski, uporabniški (PL, DLL, AL, UL) in upravljanje omrežja.
- Fundation Fieldbus (FF) je področno vodilo fundacije: Fieldbus Foundation.
- WorldFIP: World Factory Instrumentation Protocol, ki je nastal na podlagi francoskega nacionalnega standarda FIP.
- Profibus: PROcess FIeldBUS (verzije FMS, DP, PA) so razvili v Nemčiji in razširjen po vsem svetu, posebno po zaslugi firme Siemens. Od leta 1996 evropski standard EN 50 170.
- P-NET: Process automation NET je omrežje danskega porekla (PROCES-DATA Silkeborg ApS). Od leta 1996 evropski standard EN 50 170.
- INTERBUS-S je razvila firma Phoenix Contact za povezovanje na nivoju senzorjev in aktuatorjev.
- CAN: Controller Area Network so razvili v Nemčiji v glavnem pri firmi Rober Bosch Gmbh, sprva za potrebe avtomobilske industrije. Je tudi mednarodni standard ISO 11898 in ISO 11519-2.
- DeviceNet je razvila firma Allen-Bradley; temelji na CAN. Podpira ga ODVA (Open DeviceNet Vendor Association), Inc.
- SDS: Smart Distributed System so razvili pri Honeywell's MICRO SWITCH Division, temelji na CAN.
- Bitbus je razvil Intel leta 1984. Kasneje je bil dopolnjen in standardiziran z dokumentom IEEE 1118.
- LonWorks: Local Operating Network so razvili pri Echelon-u. Vsebuje vse sloje ISO/OSI modela.
- Modbus, Modbus Plus sta omrežji, ki jih je razvila firma MODICON.
- ASI: Actuator Sensor Interface, razvit v Nemčiji s strani konzorcija dobaviteljev procesne opreme, za njim stoji v glavnem Siemens.
- M-Bus: Meter Bus, sistem hišne elektronike (HES) za daljinsko branje gospodinjskih števcev.

- CEBus: Consumer Electronic Bus, razvit s strani konzorcija proizvajalcev s pomočjo EIA (Electronic Industry Association).
- SERCOS (SErial Real-time COmmunication System) mednarodni standard IEC 1491 za komunikacijo med digitalnimi pogoni in kontrolnimi enotami numerično krmiljenih strojev, razvit v Nemčiji VDW (German Machine Tool Builders Association) in ZVEI (German Electrical Standards Association),
- ARCNET: Attached Resourse Computer Network, razvit pri Datapoint in podprt s strani ARCNET Trade Association.
- HART (Highway Addressable Remote Transducer), razvit pri Rosemount in podprt od HCF (HART Communication Fundation). Ni pravo področno vodilo, v glavnem DDL (Device Description Language).

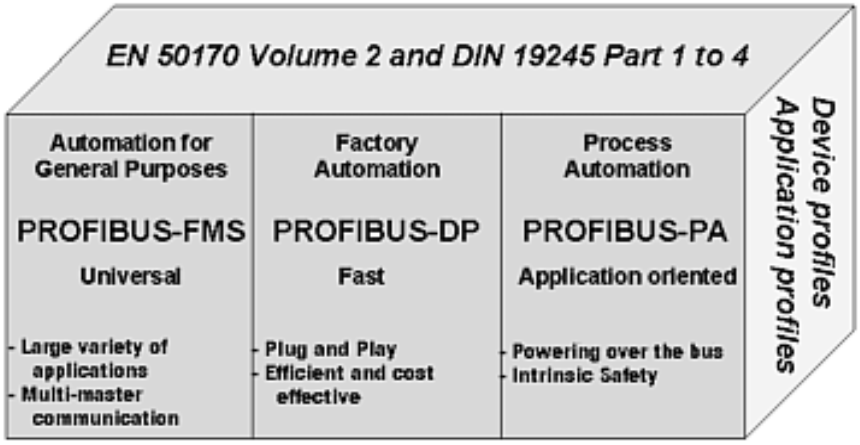
6.1 Profibus

Procesno področno vodilo Profibus (Process Field Bus) je nastalo konec 80 let v Nemčiji. Profibus med področnimi vodili za enkrat vodi z največ inštalacijami na svetu. Glavni zagovornik omrežja Profibus je Siemens. Profibus je zasnovan na podlagi obstoječih nemških in mednarodnih standardov. Upošteva arhitekturni model ISO OSI, vendar definira samo prvi, drugi in sedmi sloj. Lahko bi dodali, da nad sedmim slojem nadgradi od namena uporabe odvisen sloj.



Slika 119: Profibus v sistemih integrirane proizvodnje (Povzeto po: M. Volz Profibus Technical Overview)

Slika 119 prikazuje družino omrežij Profibus in kako se omrežja vklapljajo v hierarhijo računalniško integrirane proizvodnje, slika 120 pa podaja njihove glavne značilnost z namenom uporabe in oznako ustreznega standarda. Obstajajo tri različice omrežje Profibus, in sicer: Profibus-FMS, Profibus-DP in Profibus-PA.



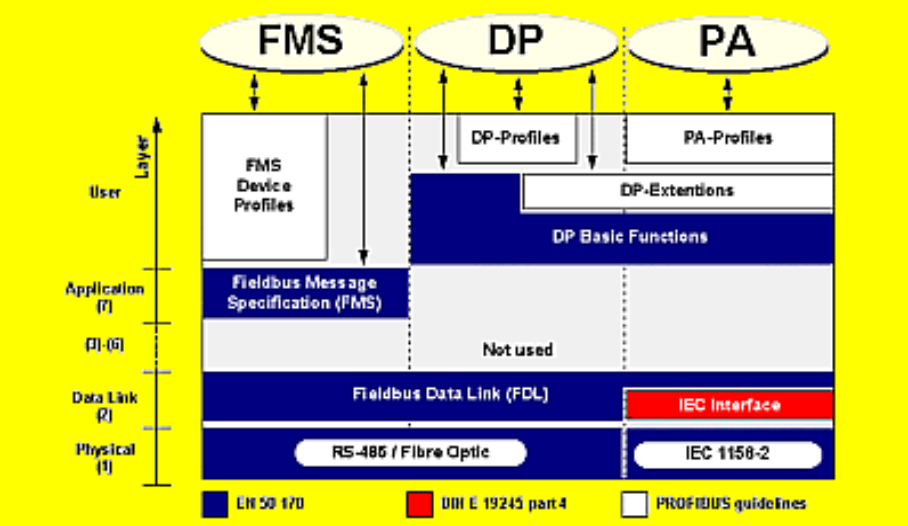
Slika 120: Družina Profibus, njene glavne značilnosti z namenom uporabe in ustreznim standardom. (Povzeto po: M. Volz Profibus Technical Overview)

Profibus-FMS je splošno namensko omrežje na celičnem nivoju. Odlikuje ga velika prilagodljivost za potrebe konkretne aplikacije. Definira fizični in podatkovni sloj ter sloj aplikacije. Sloji 3,4,5 in 6 niso izraženi. Funkcije teh slojev so zajete v podsloju LLI (ang. Lower Layer Interface), ki je del sloja aplikacije. FMS (ang. Fieldbus Message Specification) definira protokol aplikacije in realizira storitve, ki jih ta sloj opravlja za uporabnika. FMS definira tudi vmesnik z uporabnikom. Soritve FMS so definirane kot podmnožica funkcij MMS (Manufacturing Message Specification - ISO 9506) omrežja MAP (Manufacturing Automation Protocol), optimiranih za potrebe področnih vodil.

PROFIBUS-PA je zasnovan posebej za potrebe procesne avtomatizacije v nevarnem okolju (kemična, petro kemična industrija). Omogoča prenos podakov in napajanja po skupnem dvožilnem kablu v skladu z mednarodnim standardom IEC 1158-2.

Omrežje Profibus-DP je optimirano za povezovanje decentarlizirane periferije (DP) s centralnim krmilnikom. Definira samo spodnja dva sloja, medtem ko sloji 3-7 niso izraženi. Za preslikavo funkcij uporabniškega vmesnika na drugi sloj poskrbi DDLM (Direct Data Link Mapper). DDLM je specificiran v dokumentu DIN 19245 Teil3.

Kako se arhitektura Profibus vklaplja v ISO OSI arhitekturni model in njeno nadgradnjo na nivoju uporabne prikazuje slika 121.



Slika 121: Arhitektura Profibus in model OSI. (Povzeto po: M. Volz Profibus Technical Overview)

6.1.1 Fizični sloj

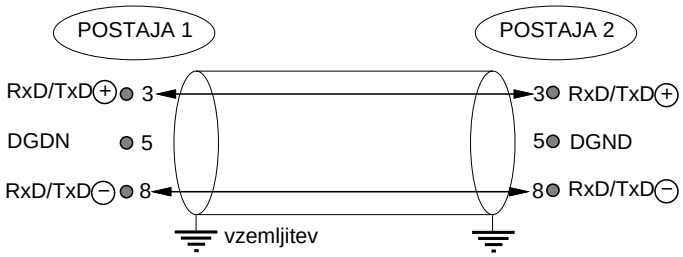
Podlaga za osnovno izvedbo fizičnega sloja je ameriški standard RS485, obstajajo pa tudi druge možnosti, kot IEC 1158-2 in optična vlakna. Za prenosno sredstvo služi oklopljena parica s karakteristično upornostjo 100 do 130 ohmov. Najvišja dovoljena hitrost prenosa pada z dolžino segmenta, vendar je dopustna dolžina odvisna tudi od kvalitete sredstva (kabla). Za dolžino segmenta 1200 metrov sme biti hitrost samo še 93.75 Kb/s. Najvišja dovoljena hitrost je 12 Mb/s in sicer pri razdalji 100 metrov.

Hitrost [Kb/s]	9.6	19.2	93.75	187.5	500	1500	12000
Razdalja [m]	1200	1200	1200	1000	400	200	100

Na en segment je lahko priključenih največ 32 postaj. Omrežje se lahko razširi s pomočjo ponavljalnikov do 127 postaj. Med dvema poljubnima postajama ne smejo biti več kot 3 ponavljalniki. Za priključitev postaj na vodilo se uporablja 9-polni konektor DB9, glej sliko 122.

6.1.2 Dostop do prenosnega sredstva

Dostop do vodila (MAC) združuje pozivanje s podajanjem žetona. Obstajata dva tipa postaj: aktivne in pasivne. Aktivne postaje ali gospodarji si v dogovorjenem zaporedju podajajo žeton. Na ta način tvorijo logični obroč, kot prikazuje slika 123. Postaje, ki niso v logičnem obroču, so podrejene oziroma pasivne postaje. Ko aktivna postaja dobi žeton, postane gospodar vodila, dokler ji ne poteče



Slika 122: Vezava postaj na omrežje PROFIBUS.

dodeljeni čas. Nato preda žeton naslednji aktivni postaji. Postaja, ki ima žeton, lahko s pozivanjem dobi ali pošlje podatke kateri koli pasivni ali aktivni postaji. Pasivna postaja sama od sebe ne sme začeti z oddajo.

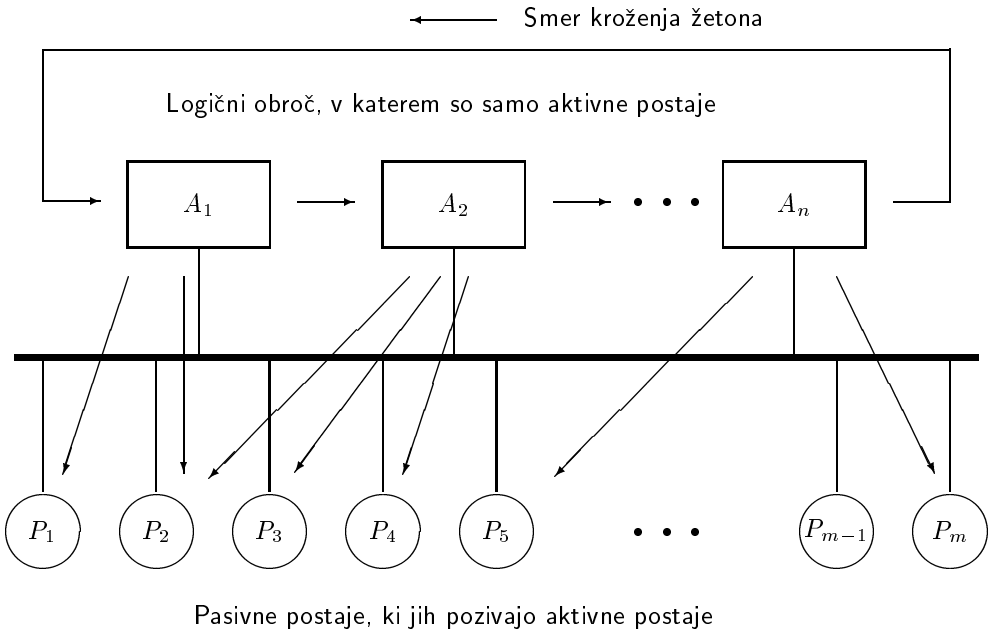
Slika 123 prikazuje splošno konfiguracijo omrežja. Možno je tudi čisto vodilo z žetonom brez pasivnih postaj ali pa čisto pozivanje z eno samo aktivno in več pasivnimi postajami, ki jih poziva aktivna postaja. Običajno je v omrežju relativno malo aktivnih postaj. Na ta način se skrajša obhodni čas žetona. Kratek obhodni čas žetona, skupaj s prioritetnim obravnavanjem pomembnejših podatkov, pa zagotavlja kratek odzivni čas.

Profibus-DP v načelu predvideva samo eno aktivno postajo, ki krožno poziva več pasivnih postaj (Monomaster sistem – tip aktivne postaje DPM1). V enem pozivnem ciklu vsem postajam pošlje in od njih sprejme sveže podatke. Če je v omrežju več aktivnih postaj (tip postaje DPM2), te opravljajo upravljalske funkcije. V primeru, da je v omrežju več aktivnih postaj tipa DPM1, pa smejo pasivne postaje sprejemati podatke samo od ene aktivne postaje (definirano s konfiguracijo), bere pa jih lahko tudi več aktivnih postaj.

6.1.3 Prioritete

Profibus nudi na podatkovnem sloju dva prioritetna nivoja okvirjev ali telegramov: višjo in nižjo. Kadar dobi aktivna postaja žeton, ima vedno možnost oddati vsaj en okvir višje prioritete, okvirje z nižjo prioriteto pa sme oddajati samo v primeru, če ji po oddaji prioritetnih okvirjev prej ne poteče razpoložljivi čas imenovan držalni čas žetona (T_{TH}). Prioriteto sporočil določa uporabnik. Zato, da se doseže želeni učinek (hiter odziv za pomembne podatke), uporabnik ne sme pretiravati s številom prioritetnih sporočil.

V procesu kroženja žetona sta pomembna naslednja časovna parametra: ciljni obhodni čas žetona T_{TR} (Target Token Rotation Time) in dejanski obhodni



Slika 123: Topologija omrežja Profibus in način dostopa do prenosnega sredstva (vodila).

čas žetona T_{RR} (Real Token Rotation Time). T_{TR} je parameter, ki je odvisen od zahtev in konfiguracije omrežja. Ko postaja dobi žeton, postavi časovnik T_{RR} na nič, ki nato prosto teče. Ko postaja ponovno dobi žeton, sme vedno, brez omejitev, oddati eno sporočilo višje prioritete. Potem primerja dejanski obhodni čas T_{RR} z dovoljenim T_{TR}. V primeru, da je držalni čas žetona (Token Holding Time) $T_{TH} = T_{TR} - T_{RR} > 0$, sme postaja čas T_{TH} oddajati druge okvirje, potem pa mora brezpogojno predati žeton. Okvir, ki je v procesu oddajanja, bo oddan do konca, vključno z morebitnimi ponovitvami oddaje tudi v primeru, da T_{RR} preseže T_{TR}, vendar se presežek direktno upošteva za skrajšanje držalnega časa ob naslednjem prejemu žetona.

6.1.4 Minimalni ciljni obhodni čas žetona

Najkrajši ciljni obhodni čas žetona T_{TRmin} je odvisen od števila aktivnih n_a udeležencev, od trajanja žetona T_{TC}, od trajanja okvirja z visoko prioriteto T_{MC-h}, od trajanja okvirja z nizko prioriteto T_{MC-l} in števila teh okvirjev na en obhod žetona n_p ter od (pričakovanega) časa potrebnega za ponovitev sporočil na en obhod žetona T_{MC-r}. T_{TRmin} ocenimo po sledeči formuli:

$$T_{TRmin} = n_a \times (T_{TC} + T_{MC-h}) + n_p \times T_{MC-l} + T_{MC-r}$$

Pri tem smo predpostavili enako dolžino prioritetnih okvirjev, enako dolžino okvirjev nižje prioritete in da ima vsaka aktivna postaja možnost oddati en visoko

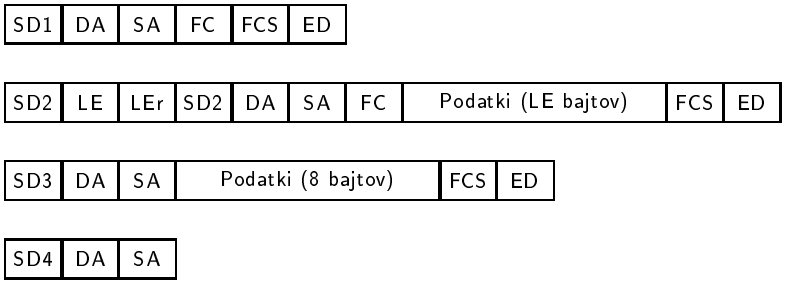
prioritetni okvir na obhod žetona. V slučaju, da ni ponovitev, določa prvi člen maksimalni odzivni čas za okvirje z visoko prioriteto. Drugi člen daje možnost tudi okvirjem z nižjo prioriteto. V splošnem si želimo, da bi bil čas T_{TR} čim krajši. Zato naj bi bilo aktivnih postaj kot tudi prioritetnih okvirjev malo, prioritetni okvirji pa kratki (npr. največ 20 bajtov podatkov).

6.1.5 Upravljanje obroča

Žeton kroži med aktivnimi udeleženci od postaje z nižjim proti postajam z višjimi naslovi. Postaja z najvišjim naslovom (ta naslov je znan vsem postajam) pošlje žeton postaji z najnižjim naslovom in obroč je sklenjen. Vsaka postaja ima svojega predhodnika z nižjim naslovom, od katerega pričakuje žeton in naslednika z višjim naslovom, ki mu preda žeton. Aktivni udeleženci so zabeleženi v seznamu aktivnih postaj (LAS - List of Active Stations), s katerim razpolaga vsaka aktivna postaja. V kolikor neka postaja zazna na njo naslovljen žeton, ki ga ne pošilja njen predhodnik, predpostavlja, da gre za napako in žetona ne upošteva. Šele po ponovnem sprejemu žetona od iste postaje razume to kot spremembo obroča, zato prevzame žeton in obnovi LAS.

Ko postaja preda žeton nasledniku, istočasno opazuje aktivnost na vodilu (s tem preveri delovanje tudi lastnega oddajnika). V kolikor v predvidenem časovnem intervalu pride do nepričakovane aktivnosti na vodilu (slediti mora okvir z izvornim naslovom naslednika) interpretira to kot napako. Zato se umakne iz obrača in gre v začetni način delovanja, ki zahteva ponovno vabilo v obroč ali celo ponovno vzpostavitev obroča. V kolikor v predvidenem času ne pride do pričakovane aktivnosti (vodilo je prosto), poskusi ponovno s predajo žetona. Če tudi tokrat predaja ne uspe, poskusi s predajo nasledniku njegovega naslednika. Če niti ta poskus ne uspe, postaja predpostavi, da je obroč razpadel in prevzame nalogo ponovne vzpostavitve obroča. V ta namen pošlje žeton dvakrat zapored na samo sebe in s tem obvesti ostale udeležence, da začenja ponovno vzpostavitev obroča.

Aktivne postaje smejo svobodno vstopiti v obroč ali iz njega izstopiti. Aktivni udeleženec, ki je v obroču, je pooblaščen za povabilo za vstop v obroč tistih postaj , ki so v naslovnem območju med njim in njegovim naslednikom. To naslovno področje se imenuje naslovna reža. Osvežitev seznama naslovne reže (npr. vabilo za vstop) se opravlja ciklično, za kar skrbi časovnik časa T_{GAP} (GAP Update Time). Postaja, ki ima žeton in se ji po oddaji sporočil še ni iztekel držalni čas žetona, poizveduje zapored po stanju postaj znotraj adresne reže. V kolikor se pasivna postaja ne javi niti na drugi poziv, se izvzame iz seznama. V kolikor aktivna postaja odgovori, da je pripravljena na vstop v obroč, mu postaja preda žeton in s tem je njeno delo opravljeno.



Slika 124: Oblike okvirjev podatkovnega sloja Profibus.

Vzpostavitev ali ponovna vzpostavitev obroča je potrebna tedaj, kadar v obroču ni žetona (torej obroč niti ne obstaja). Vzpostavitev obroča je pravzaprav poseben primer osvežitve seznamov aktivnih udeležencev in seznama adresne reže. Vzpostavitev obroča začne postaja, ki odda žeton dvakrat zapored na samo sebe. Nato začne s povpraševanjem postaj od nižjih proti višjim naslovom. V kolikor se postaja predstavi kot pasivna, se vpiše v GAPL. V seznam GAPL se vpiše tudi aktivna postaja, ki še ni pripravljena na vstop v obroč. Postaja nadaljuje s poizvedovanjem dokler ne naleti na prvo aktivno postajo, ki je pripravljena na vstop v obroč in ji preda žeton.

6.1.6 Oblike okvirjev ("telegramov")

Profibus pozna na podatkovnem sloju naslednje oblike okvirjev:

- okvirji s konstantno dolžino brez podatkov: zahteva, potrdilo, in kratko potrdilo,
- okvirji s konstantno dolžino s podatki: zahteva s podatki ali odziv s podatki, in
- okvirji s spremenljivo dolžino: zahteva s podatki in potrdilo s podatki.

Naslednje slike prikazujejo oblike naštetih okvirjev.

Med zaporednima okvirjema je potreben premov v trajanju najmanj 33 bitov. Okvir začne z začetnim znakom SD1, SD2, SD3 ali SD4 (Starting Delimiter), ki napove začetek okvirja in hkrati kodira njegovo obliko. SD1 ($10) je oznaka okvirja brez podatkov, SD3 ($A2) je oznaka okvirja s podatki, SD2 ($68) je oznaka okvirja spremenljjive dolžine. Žeton ima oznako SD4 ($DC). Znak ED (Ending Delimiter) (koda $16) označuje konec okvirja. Polje FC (Frame Control)

kodira pomen okvirja (na primer pozitivno ali negativno potrdilo, tip postaje: aktivna ali pasivna, tip storitve, i.t.d.). DA (Source Address) in DA (Destination Address) sta naslova izvorne in ponorne postaje. FCS (Frame Check Sequence) je za preverjanje pravilnosti prenosa, ki zagotavlj Hammingovo razdaljo HD = 4. V okvirjih spremenljive dolžine je LE (Lenght) dolžina – število podatkov v okvirju (do 246 bajtov) in LEr je ponovljena dolžina zaradi zanesljivosti. Potrditev okvirja je možna tudi z enim samim znakom SC, ki ima vrednost $ E5. To daje možnost hitri potrditvi, kadar ni potrebe za prenos podatkov v nasprotni smeri.

6.1.7 Storitve linijskega sloja

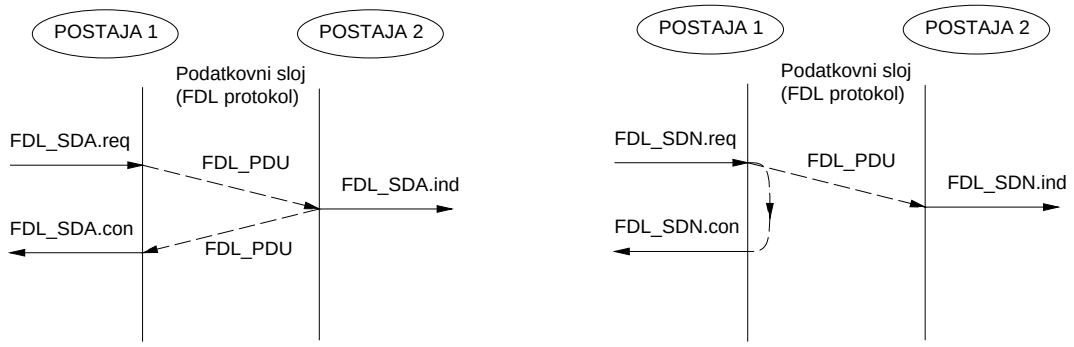
Podatkovni/linijski sloj nudi na podlagi protokola FDL (ang. Fieldbus Data Link) sloju aplikacije naslednje storitve:

- SDA (Send Data with Acknowledge) - pošlji podatke in zahtevaj potrditev,
- SDN (Send Data with no Acknowledge) - pošlji podatke brez zahteve za potrditev,
- SRD (Send and Request Data with Reply) - pošlji podatke in zahtevaj odgovor s podatki,
- CSRD (Cyclic Send and Request Data with ciklično pošiljaj podatke in zahtevaj odgovor s podatki.

Časovni potek storitev SDA in SDN je skiciran na sliki 125. SDA je torej potrjena storitev in se uporablja za pošiljanje podatkov, pri čemer mora sprejemna stran sprejem podatkov obvezno tudi potrditi. Oddajna stran v vsakem primeru dobi potrdilo - pozitiven ali negativen odgovor. Za razliko od osnovnega modela potrejene storitve, SDA storitev nima odziva. Odziv pride od aktivnosti linijskega sloja oddaljene strani. Ob tej priliki ponovno povejmo, da je storitev eno, protokol pa drugo. Potrdilo storitve (ang. Confirmation) ne smemo zamenjevati s potrdilom (ang. Acknowledgment) kot sestavine protokola. Popolnoma jasno je, da je za prenos protokolovne podatkovne enote (PDU) v splošnem lahko potrebnih več poskusov, kar vključuje ± potrditve.

Storitev SDN ni potrjena. Uporablja se največ pri skupinskem in splošnem naslavljanju, to je pošiljanju istih podatkov na več naslovov (ang. Broadcast in Multicast). Oddajna stran dobi samo lokalno potrditev, kar ji seveda ne zagotavlja, da je oddaljena stran podatke v resnici tudi dobila.

Storitev SRD je najsplošnejši način za pošiljanje in istočasno zbiranje podatkov. Možna sta oba podprimera, to je, ko se podatki samo pošiljajo ali samo



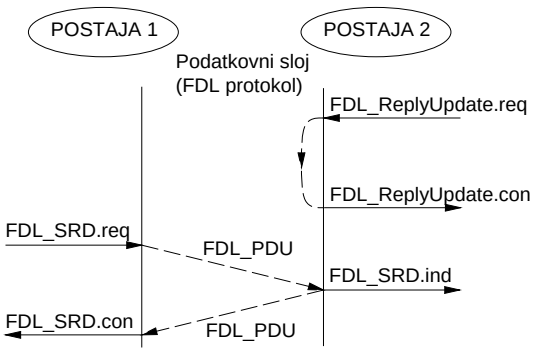
Slika 125: Storitvi SDA in SDN podatkovnega sloja omrežja Profibus. SDA (levo) je potrjena storitev; potrditev generira podatkovni sloj na oddaljeni strani (Postaja 2). SDN (desno) je nepotrjena storitev; potrditev, ki jo dobi jemalec storitve, generira podatkovni sloj na lokalni strani (Postaja 1). Potrditev je zgolj lokalna.

zbirajo. V tem primeru je podatkovno polje okvirja (podatkovna enota storitve) v eni ali drugi strani prazno. Potek storitve je skiciran na sliki 126.

Storitev CSRD se uporablja za ciklično pošiljanje podatkov na skupino naslovov in po potrebi istočasno zbiranje podatkov. S to storitvijo je na nivoju podatkovnega sloja realizirano krožno pozivanje postaj (Polling), z namenom, da se podatki tem postajam pošljejo ali od njih zberejo. Sloj aplikacije na lokalni strani zahteva z FDL_CSRD.req od podatkovnega sloja ciklično storitev ter mu skupaj z zahtevo preda seznam naslovov postaj na katere želi poslati ali od njih dobiti podatke. Nato mu zapored predaja podatke za posamezno postajo in zahteva oddajo. Če lokalna stran kot odgovor oddaljene strani dobi podatke (kot pri storitvi SRD), jih takoj preda sloju aplikacije ter nadaljuje s pozivanjem (nadaljuje CSRD storitev), dokler se zahteva eksplicitno ne razveljavi.

6.1.8 FMS – sloj aplikacije za Profibus-FMS

Protokole in storitve sloja aplikacije v omrežju Profibus-FMS pokrivata FMS (Fieldbus Message Specification) in LLI (Lower Layer Interface). LLI skrbi za prilagoditev sloja aplikacije na podatkovni sloj oziroma pravkar obravnavane storitve, ki jih ta nudi. FMS ima za specifikacijo MMS (Manufacturing Message Specification), ki je kot mednarodni standard opredeljen v dokumentih ISO/IEC 9506-1 in ISO/IEC 9506-2. ISO/IEC 9506-1 opisuje storitve aplikacijskega vmesnika in ISO/IEC 9506-2 definira protokol aplikacijskega sloja. FMS realizira podmnožico funkcij FMS, optimiranih za področje uporabe Profibus-FMS (predvsem povezovanje PLC). Bistvenega pomena je gotovo koncept odje-



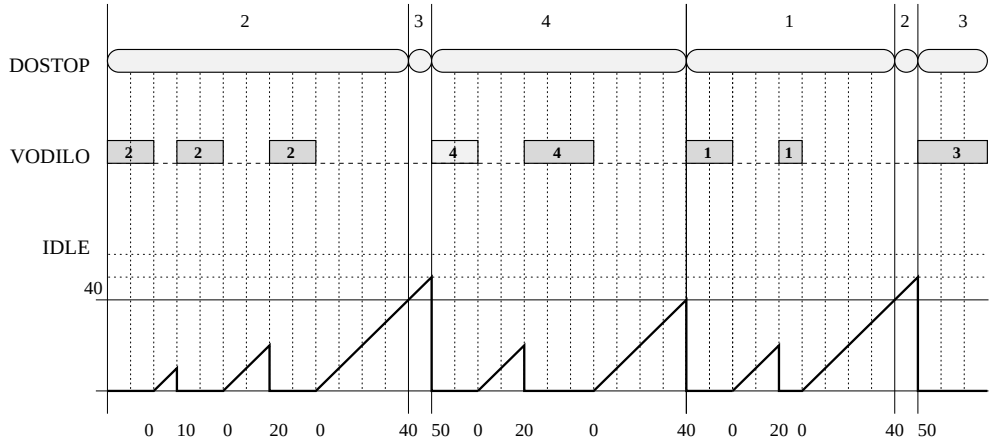
Slika 126: Storitev SRD podatkovnega sloja omrežja Profibus. Aktivnost podatkovnega sloja na oddaljeni strani (Postaja 2) pošlje v odgovor sveže podatke. Če podatkov nima, pošlje samo potrdilo. Sveže podatke dobi podatkovni sloj od sloja aplikacije preko storitve FDL_ReplyUpdate.

malec/strežnik (ang. Cient/Server Model) skupaj s konceptom navidezne (virtualne) področne naprave VFD (ang. Virtual Field Device) oziroma v sklopu MMS proizvodne naprave VMD (Virtual Manufacturing Device). Strežnik je naprava, ki daje svoja sredstva (dejavnosti, podatke) na voljo drugim, tipično oddaljenim procesom, to je odjemalcem. Odjemalec (npr. uporabnikov proces) postavi zahtevo za izvedbo potrebne storitve, ta zahteva se po omrežju prenese do strežnika (strežnega procesa). Strežnik poskrbi za izvedbo storitve in se preko omrežja odzove odjemalcu, ki je potem obveščen o realizaciji (uspešni ali neuspešni) storitve. Po konceptu FMS navidezno napravo realizira strežnik. Navidezna naprava daje izgled resnične naprave z vidika ča krajevno porazdeljene aplikacije. Daje izgled resnične naprave (npr. ventila, merilnika tlaka) s stališča medprocesne komunikacije. Na ta način se doseže enotno obravnavanje različnih naprav, neodvisno od njihove izvedbe, če so s stališča zahtevane funkcije ekvivalentne.

6.2 P-Net

P-Net so razvili na Danskem v začetku osemdesitih let (1983), za povezovanje industrijskih naprav na področnem nivoju. Od marca 1996 je tudi evropski standard EN 50170 (Part 1). Dostop do vodila v tem omrežju ima nekaj zanimivih lastnosti, ki nas bodo zanimale. Še prej pa poglejmo nekaj splošnih podatkov. Omrežje P-Net ima obliko vodila in temelji na standardu RS485. Na en segment gre do 125 postaj. Standardna hitrost prenosa je 76800 b/s. Podobno kot Profibus sta v omrežju P-Net dva tima postaj: nadrejene (ang. Master) in podrejene (ang. Slave). Na enem segmentu je lahko do 32 nadrejenih postaj. V nekem obdobju

ima lastništvo nad vodilom samo ena od nadrejenih postaj. Ta postaja lahko tedaj poziva druge postaje – upravlja vodilo. Rečemo, da ima žeton. Nadrejene postaje druga za drugo prevzemajo vodilo in na ta način tvorijo obroč. V tem pogledu je omrežje nepristransko. Logični obroč je zaznovan na podajanju **virtualnega žetnona**; virtualnega žetona zato, ker v resnici kot okvir ne obstaja. Kroženje virtualnega žetona je urejeno z ugotavljanjem dolžine časovnega intervala, ko je vodilo prosto. V ta namen ima vsaka nadrejena postaja dva števca: PROSTO in DOSTOP. Števec PROSTO se povečuje za ena s hitrostjo oddajanja/sprejemanja. Števec začne šteti vedno od nič navzgor, vsakič ko postane vodilo prosto. Če ostane vodilo prosto vsaj 40 bitnih celic, se poveča števec DISTOP za ena. Števec DOSTOP se poveča za ena vsakič, ko števec PROSTO doseže vrednost 40, 50, 60 in tako naprej. Ta števec teče od nič do števila nadrejenih postaj na vodilu (najvišje addrese). Lahko bi ga imenovali kar adresni števec. Postaja lahko prevzame vodilo (začne z oddajo) tedaj, ko se števec DOSTOP izenači z njenim naslovom. To pomeni, da je postaja dobila žeton. Ni pa nujno, da začne z oddajo. V tem primeru ostane vodilo prosto in števec PROSTO teče naprej. Vsakič, ko se poveča za deset, se adresni števec poveča za ena in pravico za oddajo dobi naslednja postaja. Ko doseže vrednost najvišje prisotne adrese, začne šteti spet od začetka. Možne razmere za štiri nadrejene postaje prikazuje slika 127. Adrese postaj so 1, 2, 3, in 4. Najprej ima vodilo postaja 2, adresni števec postaj ima vrednost 2. Postaja 2 komunicira z drugimi postajami. Vsakič, ko postane vodilo prosto, steče števec PROSTO. Ko ostane vodilo prvič prosto za več kot 40 enot, se poveča števec DOSTOP za ena in postaja 2 izgubi lastništvo vodila. Vodila bi se lahko polastila postaja 3, vendar se v narisanih razmerah to ne zgodi. Postaja 3 se ne polasti žetona. V naslednjih 10 enotah se števec DOSTOP poveča na 4. Tokrat se postaja 4 polasti vodila in vodi komunikacijo, ki se zaključi s premorom. Adresni števec se spet poveča in ker so v omrežju samo štiri nadrejene postaje, preplavi na 1, vendar ostane postaja 1 neaktivna. Adresni števec se poveča na 2 in vodila se polasti postaja 2.



Slika 127: Načelo virtualnega žetona v omrežju P-Net.

6.3 CAN

CAN (Controller Area Network) so razvili v sedemdesetih letih v nemški firmi Bosch za potrebe avtomobilske industrije. Glavni namen razvoja omrežja CAN je bil zmanjšanje števila povezav v vozilih nove dobe. Taka vozila vsebujejo vseveč elektronskih komponent (senzorjev, aktuatorjev), ki povečujejo varnost in udobnost vožnje in jih je potrebno povezati med seboj. Tovrstna omrežja morajo zadostiti številnim ostrim zahtevam, morda najvažnejša pa je hiter odziv, ki ga zagotavlja CAN. Kratek in predvsem predvidljiv odzivni čas je bistvenega pomene v vseh sistemih, ki zahtevajo delovanje v stvarnem času (prepozno delovanje zavor je lahko prav tako usodno kot odpoved), zato se je CAN hitro uveljavil tudi na drugih področjih in bil tudi standardiziran (ISO 11898 in ISO 11519-1). Poleg determinističnega odziva je v omrežjih tipa CAN učinkovito rešen sistem prioritet; sporočilo z višjo prioriteto pride vedno prej na vrsto kot sporočila z nižjo prioriteto.

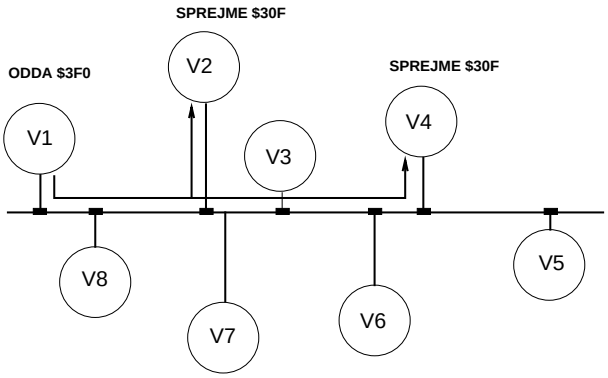
CAN kot tak se pretežno nanaša na na drugi sloj oziroma spodnja dva sloja modela OSI. Definira način dostopa do skupnega prenosnega medija, medija samega pa ne definira. Drugi, višji sloji, so trenutno v razvojni fazi. Na primer, omrežja DeviceNet (Allen-Bradly) in SDS (Smart Distributed Sensors firme Honeywell) definirata sloj uporabe, a koristita CAN na sponjih dveh slojih.

Omrežje CAN je po obliki vodilo. Število postaj na vodilu teoretično ni navzgor omejeno, omejena pa je dolžina vodila. Z dolžino vodilo narašča čas širjenja signala od konca do konca, ta pa je za pravilno delovanje omrežja odločilnega pomena. Največja dopustna dolžina vodila pada z naraščanjem hitrosti prenosa. Pri hitrosti prenosa 1 Mb/s je le-ta največ 50 m, pri hitrosti 500 Kb/s ne sme biti več kot 100 m, in tako naprej.

Vsak okvir (ali paket) začenja z 11 bitno razpoznavno številko. CAN sicer ne uporablja naslovov vozlišč, vendar je z razpoznavno številko direktno določeno, komu je okvir namenjen. Za primer je na sliki 128 skiciran primer, ko vozlišče V1 (npr. merilnik hitrosti) odda okvir s številko 11100001111 (30F šestnajstiško), ki je namenjeno vozliščema V2 (npr. prikazovalniku hitrosti na armaturni plošči) in vozlišču V4 (npr. sistemu za regulacijo motorja). Ostala vozlišča okvir zavržejo. Ta odnos se določi v času konfiguriranja sistema.

6.3.1 Dostop do kanala v omrežju CAN

Sedaj pa bomo razložili princip, po katerem vozlišča v omrežju CAN dostopajo do prenosnega medija (sredstva). Ta je za CAN bistven in najbolj značilen. CAN realizira prioritetni CSMA/CA (Carrier Sense Multiple Access with Collision Avoidance) oziroma CSMA s preprečevanjem trčenj (nedestruktivni CSMA).



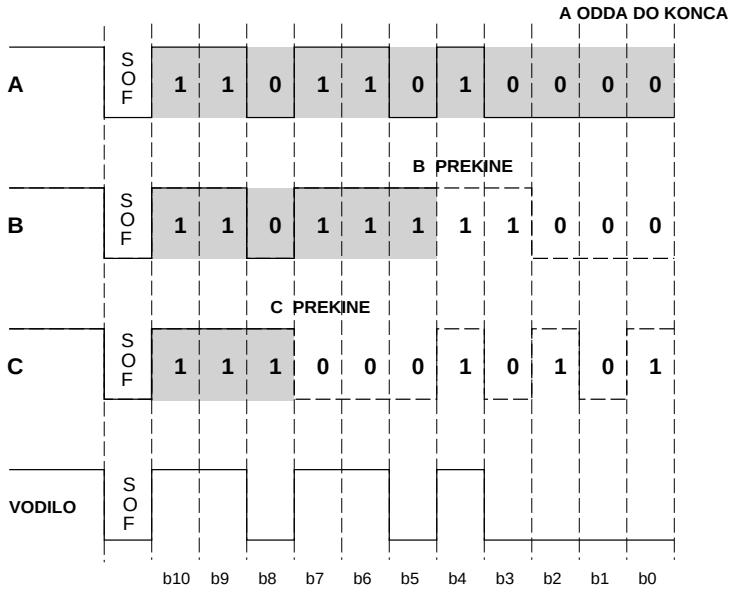
Slika 128: Topologija omrežja CAN in princip naslavljanja. Okvir je določen z razpoznavno številko. Vsa vozlišča, ki ki jih zanimajo podatki okvirja, lahko istočasno sprejmejo podatke.

Razpoznavna številka okvirja pomeni hkrati tudi njegovo prioriteto. Manjša vrednost (kodirana binarno, najpomembnejši bit se prvi odda) pomeni višjo prioriteto. Ker so kombinacije 1111111xxxx, kjer je x 0 ali 1, zaradi kompatibilnosti prepovedane, je tako možnih 2032 prioritetnih nivojev. Vodilo (največkrat RS485) deluje kot krajevno porazdeljena logična vrata IN. V mirovnem stanju je na vodilu logična enica (visok nivo). V primeru, da začne sočasno z oddajo več vozlišč, bo na vodilu stanje ena samo v primeru, da vsa vozlišča oddajo enico. Če pa vsaj eno vozlišče odda ničlo, ničla prevlada in postavi vodilo v stanje nič. V tem smislu je logična ničla "prevladujoča" ali "dominantna", enica pa "popustljiva" ali "recesivna". Na tem je zasnovano preprečevanje trčenj. V primeru, da je vodilo prosto, lahko vsako vozlišče, ki ima željo po oddaji, začne takoj z oddajo. Če je takih vozlišč več, pride do trčenja. Vozlišča, ki začnejo sočasno z oddajo, so s tem sinhronizirana na začetek okvirja. Druga vozlišča, pri katerih pride do potrebe po oddaji kasneje, čutijo, da je vodilo zasedeno in počakajo z oddajo.

Poglejmo konkreten primer. Ne pozabimo, da postaje med oddajanjem tudi sprejemajo. Naj pride do sočasne oddaje naslednjih okvirjev:

	b_{10}	b_9	b_8	b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0
Vozlišče A:	1	1	0	1	1	0	1	0	0	0	0
Vozlišče B:	1	1	0	1	1	1	1	1	0	0	0
Vozlišče C:	1	1	1	0	0	0	1	0	1	0	1

Po dogovoru ima najvišjo prioriteto okvir, ki ga odda vozlišče A, torej mora njen okvir "preživeti". Slika 129 prikazuje razmere na vodilu. Vse tri postaje začnejo z oddajo dominantne ničle, ki označuje začetek okvirja. Ker so vse tri postaje začele sočasno z oddajo, so njihovi okvirji sinhronizirani. Nato vse tri postaje oddajo prvi prioritetni bit, ki je recesivna enica, zato je vodilo v stanju



Slika 129: Primer razrešitve problema sočasne oddaje. Vodilo dobi okvir z najvišjo prioriteto.

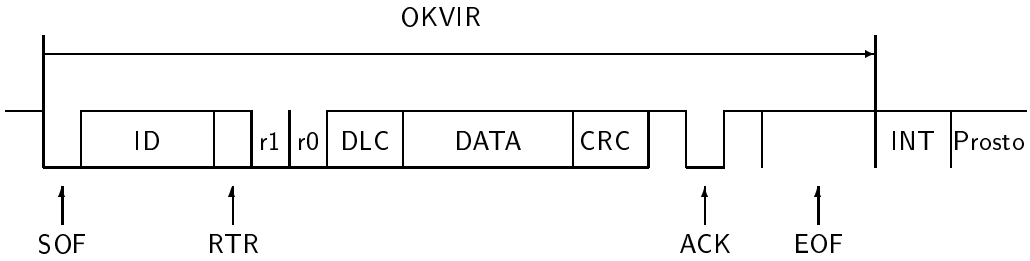
ena. Vsaka izmed postaj primerja sprejeto vrednost z oddano vrednostjo. Ker se pri vseh postajah sprejeta vrednost ujema z oddano, nadaljujejo z oddajo. Enako je po oddaji naslednjega bita. Pri oddaji tretjega bita prevlada ničla. Postaji A in B ponovno sprejmeta enako vrednost, kot sta oddali, zato nadaljujeta z oddajo. Postaja C, ki je enako kot A in B sprejela ničlo vendar oddala enico, pa takoj prekine z oddajo in se preklopi na sprejem. Od tu naprej tekmujeta za vodilo samo še postaji A in B, dokler ne oddata šestega bita. Postaja A tedaj odda dominantni bit, s čimer dokončno ostane edina oddajna postaja in odda vsebino celega okvirja brez prekinitve.

CAN uporablja NRZ obliko signala. V primeru daljšega zaporedja simbolov enake vrednosti se tak signal malo spreminja in možen je izpad sinhronizacije. Da do tega ne pride, se na vsakih 5 simbolov enake polaritete, na oddajni strani vrine polnilni simbol obratne polaritete, ki se na sprejemni strani avtomatsko izločijo.

6.3.2 Oblika okvirja v omrežju CAN

CAN pozna dve obliki okvirjev: standardni okvir (CAN verzija 2.0A) in "razširjeni" ali "podaljšani" okvir (CAN verzija 2.0B). Edina razlika je v dolžini identifikacijske številke. CAN 2.0B uporablja 29 bitno razpoznavno številko (11 bitna identifikacija in 18 bitni podaljšek), tako da se zagotovi kompatibilnost "za nazaj". Večinoma za identifikacijo zadošča 11 bitov, zato se CAN 2.0A več uporablja. Obliko standardnega okvirja prikazuje slika 130.

Začetek okvirja (SOF - Start Of Frame) označuje dominantna ničla, t.j. prehod signala na vodilu iz mirovnega (recesivna enica) v aktivno stanje (domi-



Slika 130: Oblika okvirja protokola CAN.

nantna ničla). SOF služi za vzpostavitev začetne sinhronizacije, tako imenovane trde sinhronizacije, sprejemnih vozlišč. Ponovna sinhronizacija je možna med sprejemanjem okvirja v smislu skrajšanja ali podaljšanja trajana signala enega bita. Polje za arbitražo vodila obsega 12 bitov, od teh je prvih 11 bitov identifikacijskih oziroma prioritetnih (ID), zadnji pa je RTR bit (ang. Remote Transmission Request). S pomočjo tega bita lahko neko vozlišče zahteva od drugega (oddaljenega) vozlišča oddajo podatkovnega okvirja z enako razpoznavno številko kot je številka zahteve. Okvir z bitom RTR = 0 je običajen podatkovni okvir, medtem ko okvir z RTR = 1 ne vsebuje podatkov, temveč pomeni zahtevo za oddajo podatkovnega okvirja. Polje dolžine okvirja je v tem primeru nepomembno.

Sledi šestbitno kontrolno polje, pri čemer štirje biti (DLC) označujejo dolžino okvirja (število podatkov v okvirju), prva dva bita (r1, r0) pa sta dominantna in predvidena za kasnejše razširitve. DATA je podatkovno polje in vsebuje od 0 do 8 bajtov podatkov. Polje CRC (Cyclic Redundancy Check) vsebuje 15 bitov za preverjanje pravilnosti prenosa in en ločilni recesivni bit. Za preverjanje je izbran generatorjev polinom stopnje 15,

$$x^{15} + x^{14} + x^{10} + x^8 + x^7 + x^4 + x^3 + 1$$

CRC ostanek se računa od vključno SOF bita. Polje za potrditev sestavljata dva bita. Prvi je oddan kot recesiven, vendar ga kasneje vsa vozlišča, ki pravilno sprejmejo okvir, postavijo v dominantno stanje. Ta bit se imenuje potrdilna reža (ACK Slot). Drugi bit je ločilna recesivna enica. Na ta način se potrdilni bit nahaja med dvema recesivnima bitoma, kar poveča zanesljivost njegove detekcije. Konec okvirja (EOF - End Of Frame) sestavlja sedem recesivnih enic. Za koncem okvirja sledi časovni premor treh recesivnih enic (INT), potem pa začne nov okvir ali pa vodilo preide za poljubno dolgo v mirovno stanje.

6.3.3 Časovna analiza protokola CAN

V tem podpoglavju bomo ocenili odzivnost (reakcijske čase) za sporočila v omrežju CAN. Zanimalo nas bo, kako dostop do kanala po načelu CAN vpliva na odzivni čas, to je na čas, ki poteče od trenutka ko je dana zahteva za prenos nekega sporočila do tedaj, ko je sporočilo v celoti prenešeno. Protokol CAN predvideva fiksne prioritete sporočil. Zato ima sporočilo z višjo prioriteto vedno prednost pred sporočili z nižjo prioriteto in pride pri prenosu prej na vrsto. Pri prioritetnem načinu časovnega razvrščanja s fiksnimi prioritetami se običajno privzame, da je zahtevano odzivnost možno zagotoviti samo sporočilom z najvišjo prioriteto. Ken Tindell [9] pa je pokazal, da se da na osnovi poznanih parametrov omrežja določiti zgornjo mejo odzivnega časa za vsako sporočilo, tudi za tisto z najnižjo prioriteto. V nadaljevanju bomo opisali Tindellov pristop.

Predpostavimo, da je v omrežju končno mnogo sporočil S_m in da je število vseh sporočil v omrežju znano vnaprej in enako M,

$$\text{množica sporočil } = \{S_1, S_2, \ldots, S_m, \ldots, S_M\}.$$

Sporočila, ki se potegujejo za prenos, so monotono urejena po podajočih prioritetah. Sporočila so lako bodisi periodična bodisi sporadična (vezana na dodgodke, ki se zgodijo nepredvidljivo), vendar je čas med zaporednima nastankoma istega sporočila navzdol omejen in poznan vnaprej. Torej privzamemo, da za vsako sporočilo poznamo ponovitveni cikel T_m. Poleg tega imamo za vsako sporočilo podan skrajni rok (ang. Deadline) D_m, v katerem mora biti dostavljeno (prenešeno do cilja), seveda pa je lahko prenešeno že prej. Naša naloga je, da določimo odzivni čas R_m za vsako posamezno sporočilo, in sicer v najbolj neugodnih okoliščinah. Če naj omrežje CAN zadosti zahtevam časovno kritičnih sistemov, mora zagotoviti skrajni rok dostave za vsako sporočilo, tudi za tisto z najnižjo prioriteto. Pogoj torej je:

$$R_m \leq D_m, \quad (m = 1, 2, \ldots, M).$$

Če to v danih okoliščinah ni mogoče doseči, CAN omrežje ne izpolnjuje pogojev za delovanje v stvarnem času.

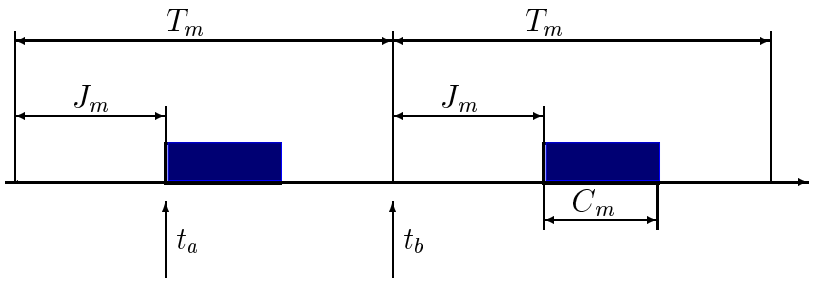
Slika 131 prikazuje model, s katerim si bomo pomagali pri analizi protokola CAN. Predpostavimo, da od trenutka, ko sporočilo S_j nastane, do trenutka, ko se pripravi (razvrsti) za prenos, mine največ čas J_m. S tem zajamemo variabilnost časovne razvrstitve za vsako posamezno sporočilo, ki se sicer s časom spreminja in je včasih lahko tudi nič. Imenujemo jo negotovost razvrstive. Kot rečeno, za vsako sporočilo poznamo ponovitveni cikel T_m. Sporočilo S_m, ki čaka na oddajo, mora biti oddano pred prihodom naslednjega, ker se v nasprotnem primeru tekoče sporočilo v oddajnem medpomnilniku "prekrije" z novim sporočilom. Zaradi nedoločenosti razvrstitve je najkrajši interval med dvema zaporednima zahtevama za prenos enak $t_b - t_a$. V tem času mora biti sporočilo v najneugodnejših razmerah

oddano, sicer se izgubi. Iz tega sledi naslednji dodatni pogoj:

$$R_m \leq (T_m - J_m), \quad (m = 1, 2, \ldots, M)$$

ali v splošnem:

$$R_m \leq \min\{(T_m - J_m), D_m\}, \quad (m = 1, 2, \ldots, M).$$



Slika 131: Model za analizo protokola CAN.

Sedaj bomo določili odzivne čase R_m za najneugodnejše razmere. Tedaj je odzivni čas sporočila S_m je seštevek časa prenašanja sporočila C_m in časa čakanja na oddajo w_m. K temu moramo prišteti še čas potreben za razvrstitev J_m,

$$R_m = J_m + w_m + C_m. \tag{35}$$

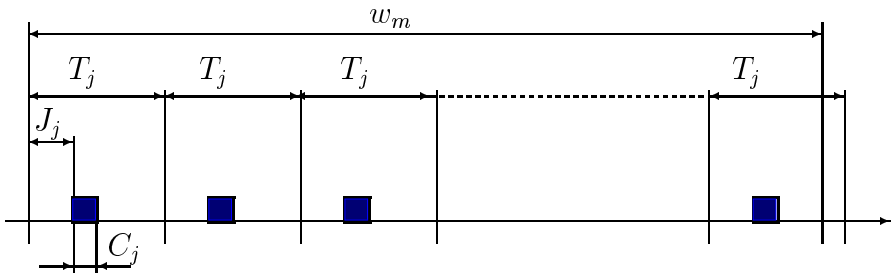
Za dano hitrost prenosa oziroma za dan čas trajanja enega bita τ je čas prenosa odvisen od dolžine – števila bitov v sporočilu. Dolžina sporočila CAN v bitih je $47 + 8 \times s_m$, pri čem 47 bitov prispevajo uvodni in zaključni biti in s_m je število podatkovnih bajtov znotraj okvirja, ki znaša največ osem. Ker CAN na vsakih 5 enakih simbolov (ničel ali enic) vrine simbol z drugačno vrednostjo (med enice ničlo, med ničle enico), bo v najslabšem primeru v sporočilo dodal še $\lfloor \frac{34+8\times s_m}{5} \rfloor$ (47-34 = 13 je izvzetih iz polnjenja). Zato je

$$C_m = (\lfloor \frac{34 + 8 \times s_m}{5} \rfloor + 47 + 8 \times s_m) \times \tau. \tag{36}$$

Določiti moramo še čas čakanja na oddajo w_m. Ta je seštevek čakanja na konec prenosa tekočega sporočila, ki je v najslabšem primeru enak trajanju najdaljšega sporočila nižje prioritete B_m (čas blokade) in skupnega časa prenašanja sporočil z višjo prioriteto:

$$w_m = B_m + \sum_{\forall j \in hp(m)} n_j \times C_j, \tag{37}$$

pri čemer je $hp(m)$ množica sporočil z višjo prioriteto od sporočila S_m, C_j čas prenašanja sporočila S_j, z n_j pa smo označili število sporočil S_j, ki nastanejo med



Slika 132: Ponazoritev števila sporočil S_j v času w_m.

tem, ko sporočilo S_m čaka, da pride na vrsto. Razmere so ponazorjene na sliki 132.

Ker je ponovitveni cikel sporočila S_j enak T_j, se v času w_m zvrsti $\lceil \frac{w_m}{T_j} \rceil$ zahtev za prenos sporočila S_j. V resnici lahko zaradi razvrstitve negotovosti J_j v času w_m nastane celo $n_j = \lceil \frac{w_m + T_j}{T_j} \rceil$ zahtev za prenos. Vseh n_j sporočil mora biti prenešenih pred oddajo sporočila S_m. Če to upoštevamo v enačbi (37), dobimo:

$$w_m = B_m + \sum_{\forall j \in hp(m)} \lceil \frac{w_m + J_j}{T_j} \rceil \times C_j, \tag{38}$$

pri čemer je $hp(m)$ množica sporočil z višjo prioriteto od S_m. Neznanka w_m nastopa na levi in na desni strani enačbe in se je ne da izraziti eksplicitno. Zato enačbo rešimo numerično z iterativno metodo. Iteracijska enačba je:

$$w_m^{(k+1)} = B_m + \sum_{\forall j \in hp(m)} \lceil \frac{w_m^{(k)} + J_j}{T_j} \rceil \times C_j, \quad (k = 0, 1, 2, \ldots), \tag{39}$$

pri čemer za začetni približek $w_m^{(0)}$ vzamemo na primer nič. Po enačbah (39) in (35) izračunamo čakalne in odzivne čase za vsa sporočila od najvišje do najnižje prioritete. Ker je čakalni čas w_m za sporočilo z nižjo prioriteto daljši od čakalnega časa sporočila z neposredno višjo prioriteto w_{m-1}, lahko v iteracijski enačbi za začetni približek vzamemo kar $w_m^{(0)} = w_{m-1}, \quad (m = 1, 2, \ldots, M-1)$.

6.4 Interbus

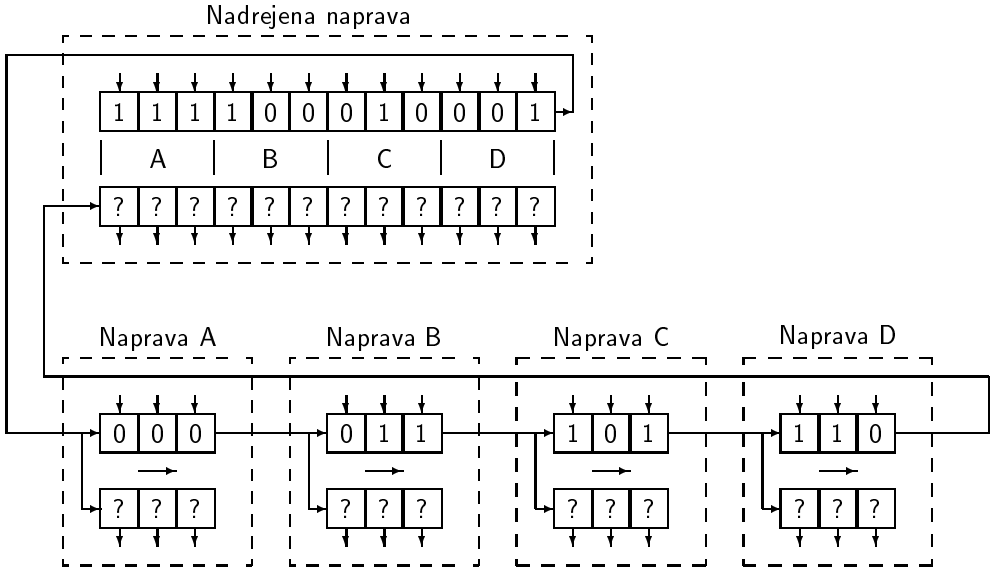
Interbus-S (IBS) je razvila firma Phoenix Contact za potrebe procesno merilne tehnike. Navzven je to omrežje sicer res videti kot vodilo, vendar je s topološkega vidika Interbus-S v resnici obroč. V obroču je vedno ena nadrejena naprava (ang. Master) in več podrejenih (ang. Slave). Nadrejena naprava (gospodar) skrbi za pravilno delovanje omrežja, t.j. poskrbi za vzpostavitev (inicializacijo)

in obratovanje omrežja. Podrejene naprave opravljajo vhodno izhodne funkcije: sprejemajo podatke iz procesa in krmilijo proces. Naprave so vezane zaporedno druga za drugo v verigo: nadrejena naprava je vezana na prvo podrejeno napravo, ta je vezana na naslednjo podrejeno napravo, i.t.d. Zadnja podrejena naprava je vezana ponovno na nadrejeno napravo. Delovanje omrežja si lahko ponazorimo z velikanskim pomikalnim registrom, ki je porazdeljen med naprave. V tem pomikalnem registru se podatki krožno pomikajo oziroma prenašajo odnaprave do naprave. Vsaka naprava ima svoj sprejemni in svoj oddajni pomikalni register: v oddajni register vpiše podatke, ki jih zajame iz procesa, po drugi strani pa poskrbi, da se podatki iz sprejemnega registra pravilno prenesejo na ustrezne krmilne izhode.

Kroženje (osveževanje) podatkov v obroču se ponavlja cel čas obratovanja omrežja. Časovnemu intervalu, v katerem so sprejeti in oddani podatki vseh naprav, pravimo cikel. Podatke, ki so oddani in sprejeti v enem ciklu, imenujemo okvir. Oddajno sprejemni (osvežitveni) cikel začne in konča nadrejena naprava. Nadrejena naprava javi podrejenim napravam začetek osveževalnega cikla, ki zato prepišejo zajete podatke iz procesa v svoje oddajne registre. Podobno nadrejena naprava prepiše oddajne podatke iz svojega internega pomnilnika v oddajni pomikalni register in začne z oddajanjem. Podrejene naprave pošiljajo podatke iz svojega oddajnega pomikalnega registra naprej naslednji napravi. Tako pridejo ti podatki preko pomikalnih registrov drugih naprav končno v sprejemni register nadrejene naprave. Po drugi strani pa se podaki, ki jih pošilja nadrejena naprava pomikajo preko pomikalnih registrov podrejenih postaj do postaje, ki so ji namenjeni. Ko pridejo podatki po obroču do postaje, ki so ji namenjeni, jih ta prevzame (prepiše iz sprejemnega pomikalnega registra). Oddajanje in sprejemanje poteka sočasno. Ko je osveževalni cikel (in s tem okvir) končan, so v sprejemnih registrih vseh podrejenih naprav sveži podatki, v sprejemnem registru nadrejene naprave pa so podatki, ki so jih oddale podrejene naprave.

Princip delovanja omrežja bomo razložili na sliki 133. Slika prikazuje poenostavljeno vezavo naprav. Za vsako od štirih podrejenih naprav smo si zamislili 3 bite sprejemnih in 3 bite oddajnih podatkov. Toliko je tudi dolžina njihovih pomikalnih registrov. Dolžina pomikalnih registrov nadrejenih naprav je enaka vsoti dolžin registrov podrejenih naprav, v tem primeru torej 12. Toliko je tudi dolžina podatkovenega dela okvirja. Nadrejena naprava hrani oddajne in sprejemne podatke za vse podrejene naprave v svojem pomnilniku. Oddajne podatke vpiše v oddajni register. Ko je oddajno sprejemni cikel končan (oz. okvir oddan), se v sprejemnem pomikalnem registru nahajajo sprejeti podatki in nadrejena naprava jih prepiše v svoj pomnilnik.

Slika 133 prikazuje vsebine registrov tik predno začne (naslednji) osveževalni cikel. V naslednji tabeli so zbrani podatki, ki jih želi nadrejena naprava oddati podrejenim napravam, medtem ko imajo podrejene naprave tudi pripravljene

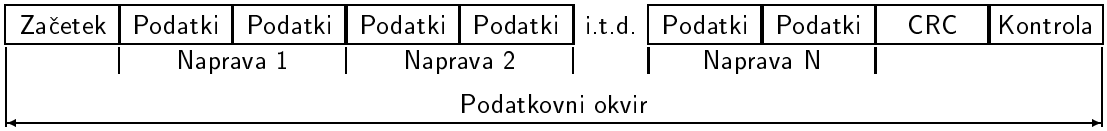


Slika 133: Delovanje omrežja Interbus.

podatke.

Postaja	Od nadrejene k podrejeni postaji	Od podrejene k nadrejeni postaji
A	111	000
B	100	011
C	010	101
D	001	110

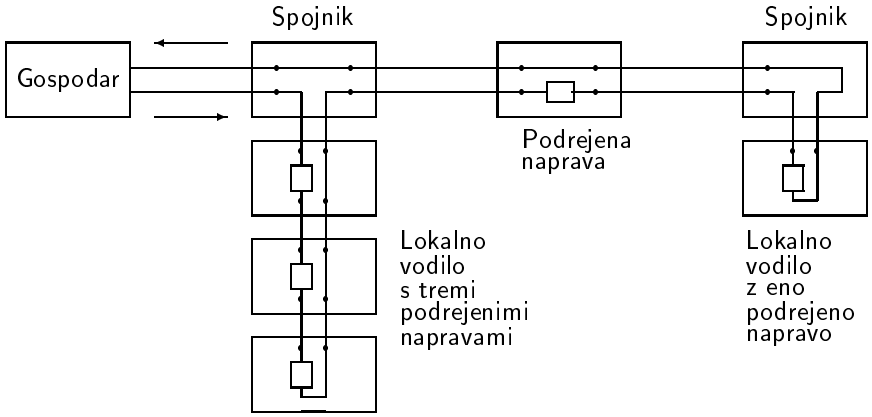
Vidimo, da je vsebina in pomen vsebine okvirja odvisna od vrstnega reda povezovanja postaj v obroč. Slika 134 prikazuje načelno obliko podatkovnega okvirja. Po najavi konca prejšnjega in začetka naslednjega okvirja so podatki za najbolj oddaljeno napravo, t.j. Napravo 1. Na tem mestu so podatki, ki jih ta naprava odda, pa tudi sprejme. Sledijo podatki drugih naprav v enakem zaporedju, kot so naprave vezane v obroč. Za podatki sta še CRC beseda za preverjanje pravilnosti beseda in kontrolna beseda (5 bitov). CRC računanje in preverjanje opravljajo vse naprave. Če ugotovijo napako, podatkov ne upoštevajo.



Slika 134: Izgled okvirja v omrežju Interbus-S

V začetku obratovanja nadrejena naprava ne pozna vrstnega reda naprav, zato začne identifikacijski cikel. S tem ciklom mora gospodar ugotoviti tipe posameznih naprav, njihov vrstni red kot tudi število aktivnih naprav v omrežju. V ta namen pošlje telegram, ki postavi vse podrejene naprave v ID stanje. Ta telegram sprejmejo hkrati vse naprave, kajti oddajni registri podrejenih naprav so v začetku premoščeni. Zato podrejene naprave pripravijo svojo razpoznavno številko (ID). Nato gospodar odda besedo za začetek okvirja (ang. Loop Back Word) in s tem začne ID cikel. V ID ciklu naprave vpisujejo (oddajajo) v okvir svoj ID, ki jih gospodar sprejema, dokler končno ne pride znak za začetek okvirja po obroču nazaj na okrog. S tem mu postane topologija obroča znana.

Za izvedbo omrežja Interbus-S sta predvidena dva tipa vodila: tako imenovano "daljinsko" vodilo in lokalno vodilo (slika135). Obe vodili prenašata signale z enakim pomenom, nivoji signalov pa so različni. Hitrost prenosa je na obeh vodilih 500 Kb/s. Daljinsko vodilo povezuje postaje na daljavo do 400 m, signali so v skladu s standardom RS485. Lokalno vodilo mora biti kratko (povezovanje modulov v enem ohišju). Služi istočasno za prenašanje podatkov in napajanje naprav. Lokalno vodilo se priključi na daljinsko vodilo preko prilagodilnih enot.



Slika 135: Izvedba omrežja Interbus-S. Na daljinsko vodilo vežemo lokalna vodila. Podrejene naprave vežemo direktno na daljinsko vodilo ali na lokalna vodila.

Literatura

[1] James J. Pinto, "Fieldbus: A Neutral Instrumentation Vendor's Perspective", Action Instruments, Inc. San Diego, CA 92123, 1996.

[2] J. Coutinho, S. Martin, G. Samata, S. Tapley, D. Wilkin, "Fieldbus tutorial", maj 1995.

[3] Klaus Bender, "PROFIBUS Der Feldbus für die Automation", Carl Hanser Verlag München Wien, 1990.

[4] Michael Volz, *PROFIBUS*, PROFIBUS Nutzerorganisation, Technische Druckschrift, 1994.

[5] DIN19245 Teil 1, PROFIBUS, Process Field Bus, Übertragungstechnik, Buszugriffs- und Übertragungsprotokoll, Dienstschnittstelle zur Anwendung-Schicht, Management, Beuth Verlag Gmbh, Berlin 1991.

[6] K. Furman, *Analiza industrijskega procesnega vodila PROFIBUS*, Diplomsko delo, Fakulteta za elektrotehniko in računalništvo, Ljubljana 1992.

[7] Anka Ščerk, *Industrijsko področno vodilo PROFIBUS*, Zaključna naloga, Fakulteta za elektrotehniko, Ljubljana 1996.

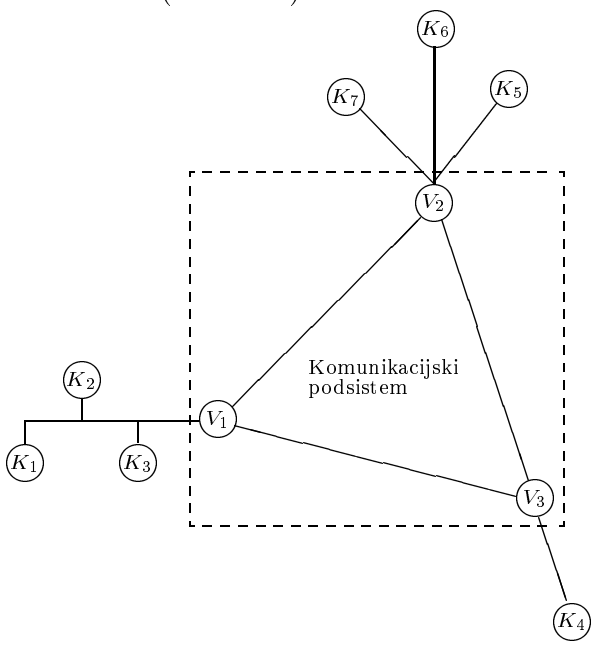
[8] -, *CAN Specification, Version 2.0*, Robert Bosch GmbH, Stuttgart, September 1991.

[9] K. Tindell, A. Burns, "Guaranteeing Message Latencies on Controller Area Network (CAN)", Proceedings 1st International CAN Conference, Mainz 1994.

[10] K. Tindell, "Analysis of Hard Real-Time Communications", YCS-94-222, Department of Computer Science, University of York, England, 1994.

7 Elementi mrežnega sloja

Komunikacijska omrežja sodijo s svojim bistvom med obsežne sisteme. V sebi združujejo veliko število vmesnih in končnih vozlišč ter še več povezav. Običajno obravnavamo vmesna vozlišča ločeno od končnih vozlišč. Pravimo, da vmesna vozlišča sestavljajo *komunikacijski podstistem*, na katerega se po potrebi priključujejo končna vozlišča (slika 136).



Slika 136: Komunikacijski podsistem vmesnih vozlišč (V_1, V_2, V_3) in končna vozlišča.

Za pravilno in učinkovito delovanje komunikacijskega podsistema skrbi mrežni sloj. Temeljna naloga komunkacijskega podsistema in s tem mrežnega sloja je povezovanje krajevno porazdeljenih končnih vozlišč. Poskrbeti mora, da pridejo *paketi* od enega končnega vozlišča skozi omrežje do drugega končnega vozlišča. Za razliko od podatkovnega sloja, ki skrbi za zanesljiv prenos podatkov med sosednjimi vozlišči v omrežju, pa mora mrežni sloj zagotoviti ustrezno *prenosno pot* skozi omrežje "od konca do konca" med poljubnima končnima vozliščema v omrežju.

V nekaterih primerih ostane prenosna pot v času komuniciranja dveh oddaljenih procesov (končnih vozliščih) nespremenjena cel čas komuniciranja. Paketi gredo drug za drugim po isti prenosni poti, ki je bila določena še predno je začel dejanski prenos. Videti je, kot da sta vozlišči neposredno povezani drugo z drugim, čeprav gredo podatki preko vmesnih vozlišč. Pravimo, da med vozliščema obstaja (navidezna) *zveza*. Prenosna pot pa se lahko v času komunikacije tudi spreminja za vsak posamezni paket. Tedaj je paket samostojna podatkovna enota

in do zveze med končnima vozliščema v pravem pomenu besede sploh ne pride.

Pri določanju prenosne poti skozi omrežje sodelujejo vmesna vozlišča. Ko paket pride do takega vozlišča, ga mora le-to poslati naprej v predvideni smeri. Pravimo, da opravlja funkcijo povezovanja oziroma *usmerjanja* (ang. routing). Usmerjanje je dejavnost, ki jo opravljajo vmesna vozlišča na nivoju omrežja pri določanju prenosne poti paketov. Za opravljanje funkcije usmerjanja morajo vmesna vozlišča poznati statične in dinamične lastnosti omrežja, denimo topologijo omrežja in s tem v zvezi neposredno povezanost vozlišč, efektivno hitrost prenosa na posameznih povezavah, kvaliteto in ceno posameznih prenosnih poti, in nenazadnje, obremenjenost omrežja ali dela omrežja. Obremenjenost omrežja je odvisna od usmerjanja, torej mora biti način usmerjanja odvisen od obremenjenosti omrežja.

Naslednje pomembno vprašanje v zvezi z mrežnim slojem je *naslavljanje*. Če naj bo omrežje sposobno dostaviti paket naslovljenemu vozlišču, mora poznati njegovo lokacijo oziroma njeno priključno mesto. Uporabnik (izvor) običajno navede naslov ponora (oddaljene postaje), ki ne pove ničesar o položaju postaje v omrežju. Torej mora biti omrežje sposobno preslikati naslov vozlišča v njegov položaj.

Zadnje in ne najmanj pomembno vprašanje z mrežnim slojem pa je povezovanje omrežij med seboj v *omrežje omrežij*. Omogočati mora povezovanje ne le enakih, ampak tudi povezovanje podobnih in celo čisto različnih omrežij. To nalogo opravljajo posebna vozlišča, ki jim rečemo *prehod* (ang. Gateway).

7.1 Omrežje in storitve

Po modelu ISO OSI daje mrežni sloj storitve naslednjemu višjemu - prenosnemu sloju. S stališča prenosnega sloja je najvažnejše, da pridejo njegove podatkovne enote od oddajnega vozlišča skozi omrežje do sprejemnega vozlišča. Oddajno in sprejemno vozlišče sta končni vozlišči, ki sta vezani na komunikacijski podsistem, za katerega skrbi mrežni sloj. Torej komunikacijski podsistem nudi storitve končnim vozliščem. Zato vmesnik med prenosnim in mrežnim slojem modela ISO OSI sovpada z vmesnikom med končnimi vozlišči in omrežjem. S tem storitve mrežnega sloja v resnici definirajo vmesnik med končnimi vozlišči in komunikacijskim podsistemom, s tem pa tudi pravila, ki jih morajo končna vozlišča upoštevati.

V majhnih omrežjih je uporabnik omrežja tudi njegov lastnik. V velikih omrežjih pa lastnik, ki upravlja omrežje, praviloma ni hkrati tudi neposreden uporabnik omrežja. Tedaj je vmesnik med prenosnim in mrežnim slojem, ki je hkrati tudi vmesnik med končnim vozliščem in komunikacijskim omrežjem, tudi

vmesnik med uporabnikom in lastnikom omrežja.

Zastavlja se vprašanje, kaj mora dajalec storitve v osnovi nuditi oziroma zagotoviti uporabniku oziroma kje naj se opravlja glavnina komunikacijskih dejavnosti. Za uporabnika omrežja je po eni strani najugodnejše, če mu omrežje daje možnost zanesljive komunikacije s poljubnim vozliščem v omrežju. Po drugi strani pa se visoke stopnje zanesljivosti omrežja ne da vedno upravičiti. Eden od razlogov je ta, da visoka stopnja zanesljivosti ni vedno nujna; lahko je pomemnejša pravočasna dostava. Drugi razlog pa je, da se preverjanje in nadzor nad pretokom podatkov večkrat opravlja (in s tem podvaja) tudi v končnih vozliščih.

Obstajata dve obliki storitev, ki se pojavljata tudi na drugih slojih, vendar prideta na mrežnem sloju najbolj do izraza, in sicer:

- povezana storitev in
- nepovezana storitev.

V prvem primeru je s strani dajalca storitve zagotovljeno, da gredo podatki zanesljivo, zaporedno in brez podvajanja na drugo stran, in da je jemalec storitve o tem po potrebi tudi obveščen. Pri nepovezani storitvi vtis povezanosti sprejemnika z odajnikom ne obstaja, ni zagotovljen zanesljiv prenos podatkov, možno je podvajanje ali izgubljanje podatkov, in jemalec storitve o tem praviloma ni obveščen.

7.2 ARP in RARP

Kot rečeno, je ena od nalog mrežnega sloja je preslikava mrežnega naslova postaje v njen fizični naslov. Če želi neka postaja poslati paket oddaljeni postaji, mora poznati njen omrežni naslov. Za dostavo paketa mora poskrbeti omrežje. Če naj bo omrežje zmožno dostaviti paket naslovljeni postaji, mora poznati njeno priključno mesto oziroma tako imenovani fizični naslov. Preslikava vsakega mrežnega naslova v pripadajoči fizični naslov bi bila lahko dana v obliki tabele in dana na uporabo vsem zainteresiranim v omrežju. Čeprav bi bila takšna rešitev možna, pa je neprikladna. Bolj prilagodljivo rešitev daje protokol. Ogledali si bomo kako je za to poskrbljeno v omrežju Internet, ki uporablja na nižjih slojih Ethernet.

V omrežju TCP/IP opravljata preslikavo med mrežnimi, to je IP naslovi in fizičnimi, v našem primeru Ethernet naslovi, protokola z imenom ARP (Address Resolution Protokol) in RARP (Reverse ARP). Potek preslikave bomo ponazorili na primeru.

Denimo, da želi postaja z naslovom IP1 = 193.2.72.151 poslati paket postaji z naslovom 193.2.72.169. Zato od svojega linijskega sloja (gonilnika mrežne kartice) zahteva, naj pošlje okvir na ta naslov. Če naj linijski sloj pošlje okvir postaji z naslovom IP2 = 193.2.72.169, mora poznati njen Ethernet naslov. Ker linijski sloj Ethernet naslova zaenkrat ne pozna, pošlje ARP poizvedovalni okvir, vanj vpiše IP naslov iskane postaje in zahtevo, naj se postaja z naslovom IP2 odzove s svojim Ethernet naslovom E2. Če iskana postaja obstaja, se odzove in komunikacija med obema postajama lahko steče preko Ethernet naslovov. Postaja IP1 od tedaj naprej hrani Ethernet naslov postaje IP2 v svojem lokalnem pomnilniku, tako da kasneje poizvedovanje ni več potrebno. V bistvu je odgovor na poizvedovanje dostopen vsem postajam, na istem segmentu. Tudi ostale postaje lahko vpišejo zvezo med naslovoma IP2 in E2 v svoj pomnilnik in njihovo poizvedovanje ni potrebno. Še več. Iz poizvedovalnega okvirja je razvidna zveza med naslovoma IP1 in E1. Ker je poizvedovani okvir dosegljiv vsem postajam na segmentu, eksplicitno poizvedovanje po naslovu E1 ni potrebno. Ostaja še vprašanje, kako poteka iskanje fizičnega naslova za postaje, ki niso na istem segmentu. Če so segmenti povezani s ponavljalniki ali mostovi, je enako kot za iskanje postaj na istem segmentu. V primeru, da je med segmentoma usmerjevalnik ali več usmerjevalnikov, pa opravi poizvedovanje usmerjevalnik. Usmerjevalnik, ki zazna poizvedovanje po postaji, ki se ne nahaja na istem segmentu, ponovi poizvedovanje na drugem segmentu. Ko dobi odgovor na poizvedovalni okvir, javi poizvedovalni postaji namesto Ethernet naslova iskane postaje svoj lastne Ethernet naslov. Z drugimi besedami, poizvedovalni postaji se predstavi kot da je on sam iskana postaja in komunikacija z iskano postajo steče preko njega – usmerjevalnika.

7.3 Notranja zgradba mrežnega sloja

Glede na notranji ustroj mrežnega sloja obstajata dva tipa omrežij, in sicer omrežje tipa

- *navidezni tokokrog* in omrežje tipa
- *datagram.*

Z enim ali drugim tipom omrežij so možne povezane in nepovezane storitve (storitve z vzpostavitvijo zveze kot tudi brez vzpostavitve zveze). Res pa je, da je v omrežju tipa navidezni tokokrog bolj naravna storitev z vzpostavitvijo zveze, v omrežju tipa datagram pa storitev brez predhodne vzpostavitve zveze.

V omrežju tipa navidezna zveza se prenosna pot med vozliščema določi predno začne dejanski prenos podatkov. Od tedaj naprej med njima na videz obstaja direktna zveza. Ko je zveza vzpostavljanja, začne prenos paketov, ki gredo

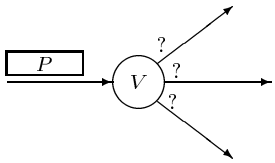
drug za drugim po isti prenosni poti, dokler obstaja za to potreba. Vmesna vozlišča na prenosni poti usmerjajo pakete iste zveze vedno v isto smer. Komunikacijo sestavljajo tri faze: vzpostavitev zveze, vzdrževanje zveze in sproščanje zveze.

Omrežje tipa datagram omogoča komunikacijo med končnimi vozlišči brez da bi kadarkoli prišlo do vzpostavitve zveze med njima. Pot prenosa se ne določi vnaprej, ampak se določa sproti za vsak posamezen paket posebej, neodvisno od ostalih paketov. Vsak paket se posebej usmerja. Funkcija usmerjevalnih vozlišč je v tem tipu omrežja bistveno drugačna.

V obeh primerih je osnovni problem usmerjanje ali določanje prenosne poti paketom. Vendar pa se le-ta v enem primeru določa na nivolju zveze, v drugem primeru pa na nivolju paketa. Problematika usmerjanja je v obeh primerih seveda različna in s tem v zvezi tudi opravljanje funkcije usmerjanja vmesnih vozlišč.

7.4 Usmerjanje

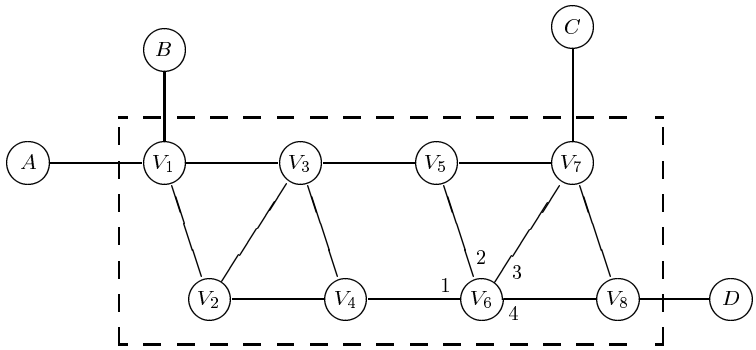
Bistvena naloga mrežnega sloja je določanje prenosne poti paketov. Paketom se določi pot po enem izmed dveh osnovnih načinov: datagram način ali navidezni tokokrog (povezava, zveza). Pri določanju prenosne poti sodelujejo vmesna vozlišča. Ko pride paket do takega vozlišča, ga mora vozlišče čimprej poslati naprej, vprašanje pa je, v kateri smeri. Pri tem se poslužuje usmerjevalnih tabel. Organizacija tabel, pa tudi način usmerjanja je bistveno odvisen od delovanja komunikacijskega podsistema - načina prenašanja paketov.



Slika 137: Osnovna naloga vmesnih vozlišč je usmerjanje paketov.

7.4.1 Datagram

V omrežju tipa datagram predstavljajo paketi na nivoju mrežnega sloja samostojne podatkovne enote, ki gredo preko vmesnih vozlišč od oddajnega do sprejemnega končnega vozlišča. Denimo, da obstaja prenos paketov med procesoma na končnih vozliščih A in D (slika 138. Ko pride denimo paket od vozlišča A do vmesnega vozlišča V_6, ki je namenjen vozlišču D, ga mora to vozlišče poslati naprej. To pomeni, da se mora odločiti, v kateri smeri naj pošlje paket.



Slika 138: Usmerjanje v omrežju tipa datagram.

V ta namen vzdržuje usmerjevalno tabelo. Primer usmerjevalne tabele za prihodno smer 1 smo skicirali spodaj.

Končno vozlišče	Smer pošiljanja	Delež paketov
C	2	0.1
	3	0.2
	4	0.7
D	2	0.3
	3	0.5
	4	0.2

Če pride paket do vozlišča V_6, ki je namenjen vozlišču D, ga V_6 pošlje naprej v smeri $2, 3$ ali 4. Za smer se odloči tako, da je delež paketov v predvideni smeri tak, kot je zabeleženo v usmerjevalni tabeli. Pomni, da v tabeli ni zabeležena smer prihoda.

Podobna pravila kot za obravnavano vozlišče V_6, veljajo tudi za druga vmesna vozlišča. S pomočjo usmerjevalnih tabel vozlišča uravnavajo odhodni promet. Usmerjevalne tabele torej določajo delež paketov, ki jih vozlišče odda v neki smeri. V našem primeru vozlišče V_6 pošilja večinski delež paketov za vozlišče D v smeri 3, manjši delež pa v smereh 2 in 4. Tako so določene glavne in alternativne prenosne poti. Tabele se ocenjujejo na osnovi prometa (dinamično) in topologije, kako pa nas za enkrat ne zanima.

7.4.2 Navidezni tokokrog

V omrežju tipa navidezni tokokrog ali navidezna zveza so oddaljeni procesi na končnih vozliščih v času komuniciranja med seboj na videz povezani. V splošnem obstaja v nekem obdobju v omrežju več navideznih povezav, vendar vsak paket v omrežju pripada samo eni taki navidezni povezavi. Paketi, ki pripadajo eni navidezni povezavi, se morajo zato razlikovati od paketov, ki pripadajo neki drugi

navidezni povezavi. To pomeni, da morajo biti paketi v takem omrežju še dodatno označeni.

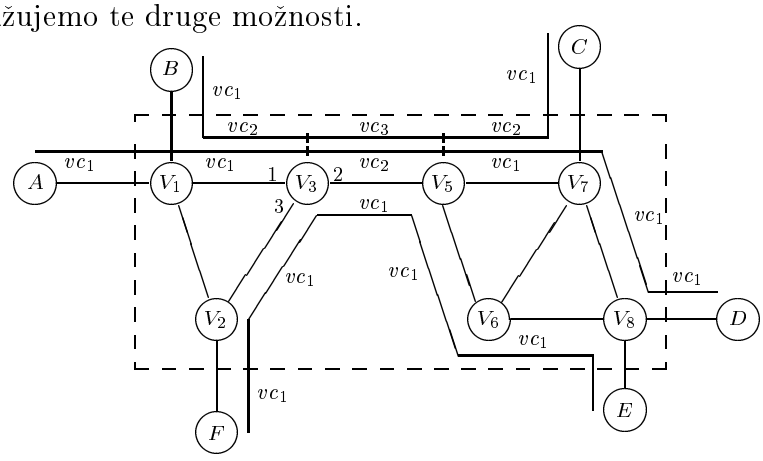
Tako kot v omrežju tipa datagram moramo tudi v omrežju tipa navidezni tokokrog med seboj razlikovati posamezne fizične povezave do vozlišča. Ker pa na isti fizični povezavi lahko sočasno obstaja več logičnih (navideznih) povezav, moramo dodatno označiti še logične povezave (glej 139).

Vsaka logična povezava, ki obstaja na neki fizični povezavi, je določena z dvema oznakama: oznako fizične povezave in oznako logične povezave. Bolj konkretno, vsak posamezen paket pripada eni fizični in eni logični povezavi. To velja za vsak paket v omrežju, tako tiste, ki pridejo do usmerjevalnega vozlišča, kot tiste, ki vozlišče zapustijo. Naloga vmesnega vozlišča je torej, da pri prehodu paketa skozi vozlišče usmeri paket na ustrezno fizično povezavo in mu zraven priredi enoznačno oznako logične povezave. Skratka, naloga vmesnega vozlišča je, da par fizične in logične povezave prihodnega paketa preslika v oznako fizične in logične oznake odhonega paketa.

$$(f, v)_{prihod} \rightarrow (f, v)_{odhod}$$

V ta nemen vzdržujejo ustrezno organizirane usmerjevalne tabele. Primer usmerjevalne tabele za vozlišče V_3 smo skicirali spodaj.

Zastavlja se vprašanje, zakaj ne številčimo iste navidezne povezave od začetka do konca z enako oznako. To bi bilo vsekakor možno, vendar pa ni potrebno. Dovolj je, da razlikujemo različne navidezne povezave na, ki gredo po isti fizični povezavi, za preslikavo oznak logičnih povezav pa poskrbijo usmerjevalne tabele. Ker je lažje je uporabljati različne oznake za isto logično povezavo na se poslužujemo te druge možnosti.

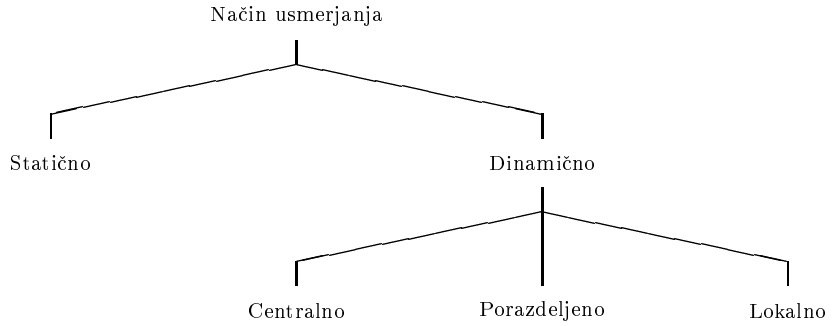


Slika 139: Omrežje tipa navidezni tokokrog. Na isti fizični povezavi sočasno obstaja več logičnih ali navideznih povezav.

Prihod		Odhod	
smer	vc	smer	vc
1	vc1	2	vc2
1	vc2	2	vc3
3	vc1	2	vc1

7.5 Algoritmi usmerjanja

Algotitme za usmerjanje delimo v dve družini. V prvo sodijo algoritmi usmerjanja, pri katerih se pravila za usmerjanje (usmerjevalne tabele) določijo enkrat za vselej in se kasneje ne spreminjajo ali pa samo izjemma, vendar ne med samim obratovanjem omrežja. Pravilo usmerjanja se torej ne določa na osnovi ocenjevanja dejanskih razmer v omrežju. Tem algoritmom usmerjanja rečemo neadaptivni algoritmi ali tudi statični. V drugo skupino spadaja adaptivni algritmi. Ti se prilagajajo spremembam v omrežju (spremembam v topologiji ali v prometu) in jim zato rečemo tudi dinamični. Glede na strategijo usmerjanja jih naprej delimo na centralne, lokalne, in porazdeljene. Pri centralnem načinu usmerjanja se določajo usmerjevalne tabele na osnovi globalne informacije o omrežju, ki se zbere na enem (centralnem) mestu. Centralno vozlišče skuša na osnovi informacije o celotnem omrežju priti v nekem smislu do optimalne rešitve. Na osnovi te informacije se določijo pravila usmerjanja za vsa vozlišča v omrežju, ki se po predvidenem protokolu iz centralnega vozlišča občasno distribuirajo drugim vozliščem. Lokalni algoritmi se pri usmerjanju zanašajo na informacijo, ki je prisotna v usmerjevalnem vozlišču samem. Vozlišča si eksplicitno ne izmenjujejo informacije, ki bi jo koristili za usmerjanje. V porazdeljenem ali distribuiranem načinu usmerjanja koristi usmerjevalno vozlišcc e poleg svoje lastne informacije za gradnjo usmerjevalne tabele še informacijo iz neposredno sosednjih vozlišcc.



Slika 140: Klasifikacija algoritmov usmerjanja.

Splošni kriteriji za snovanje in izbiro algoritmov usmerjanja se načelno ne razlikujejo od algoritmov za druge namene. Biti morajo robustni, stabilni, nepristranski in v nekem smislu optimalni.

Robustnost algoritma pomeni njegovo odpornost na okvare in v splošnem na spreminjajoče se razmere v omrežju. Z drugimi besedami to pomeni, da zaradi spremenjenih okoliščin (izpad ene ali več povezav ali dela omrežja) ne sme odpovedati celotno omrežja.

Stabilnost algoritma pomeni, da ima majhna sprememba v obratovalnih pogojih omrežja za posledico majhno spremembo v načinu usmerjanja.

Nepristranskost algoritma usmerjanja je vedno zaželena. To pomeni, da bi morala imeti enakovredna vozlišča imeti enakovredne pogoje.

Optimalnost zahteva, da je v danih pogojih (omejitvah) usmerjanje tako, da maksimizira (ali minimizira) izbrano kriterijsko funkcijo (npr. maksimizira prepustnost omrežja, minimizira povprečni kasnilni čas, minimizira stroške, zagotavlja zaupnost, in podobno). Največkrat nepristranskost in optimalnost ne gresta skupaj.

7.5.1 Izolirano – lokalno usmerjanje

Pri lokalnem usmerjanju vozlišče gradi usmerjevalno tabelo na podlagi svoje (lokalne) informacije. Z drugimi vozlišči si eksplicitno ne izmenjuje informacije, ki bi jo uporabil za usmerjanju (na primer tabel ali delov tabel). Lokalna informacija, ki je lahko podlaga za usmerjanje je na primer: število aktivnih povezav preko vozlišča, dolžine čakalnih vrst v dani smeri, frekvenca prihodov, frekvenca odhodov, kapacitete direktnih povezav, in podobno.

Preprost način usmerjanja, ki ga je predlagal Baran (1964), temelji na predpostavki, da bo prepustnost omrežja velika, če bo čas bivanja paketov v vozliščih kratek. Zato se skuša usmerjevalno vozlišče vsakega prispelega paketa čimprej znebiti (ang. "Hot patato"). To doseže tako, da za določen sprejeti paket izbere odhodno smer, za katero je čas čakanja na oddajo najkrajši – izbere smer z najkrajšo čakalno vrsto. Lahko pa vzdržuje eno samo čakalno vrsto in pošlje paket, ki pride na vrsto v tisto smer, ki postane najprej prosta. Oba primera dobro poznamo iz čakalnih vrst v banki.

Naslednji način lokalnega usmerjanja temelji na povratnem učenju (ang. Backward Learning). Usmerjevalna vozlišča gradijo usmerjevalne tabele na podlagi prispelih paketov. Usmerjevalna tabela vsebuje za vsako znano vozlišče smer, v kateri naj vozlišče pošlje paket naprej in oceno kvalitete te smeri. Za omrežje na sliki 141 bi usmerjevalna tabela vozlišča V_3 lahko izgledala tako, kot je prikazano v tabeli 6(a).

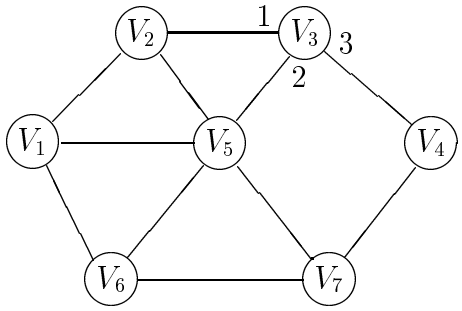
Na podlagi te vsebine tabele pošilja vozlišče V_3 pakete za vozlišče V_1 v smeri 2. Ocena smeri lahko na primer pomeni število vmesnih vozlišč na poti do danega vozlišča v predvideni smeri. Denimo, da vozlišče V_1 pošilja paket vozlišču

V_4 preko vozlišča V_3. Zato v glavo paketa vpiše svoj naslov (V_1), naslov vozlišča V_4 ter števec prehodov skozi vozlišča, ki ga postavi na nič:

V_4	V_1	$N = 0$	Vsebina

Vsakič, ko paket prečka vozlišče, se števec poveča za ena. Naj paket pride do vozlišča V_3 iz smeri 1 in naj bo vrednost števca $N = 1$. Na primer, paket je prišel preko vozlišča V_2, ki poveča števec na ena. Vozlišče V_3 preveri oceno, ki jo ima v usmerjevalni tabeli za vozlišče V_1. Ker je nova ocena smeri na podlagi prispelega paketa boljša od že zabeležene, privzame novo smer in novo oceno. Obnovljena (izboljšana) vsebina tabele je prikazana v tabeli 6(b). Ko vozlišče V_3 dobi paket, ki je namenjen vozlišču V_1, ga pošlje v smeri 1 in ne več v smeri 2.

Po določenem času vsa usmerjevalna vozlišča razpolagajo z najugodnejšimi smermi pošiljanja. Ker vozlišča popravljajo odhodne smeri vedno samo na podlagi boljše ocene, se lahko zgodi, da čez čas vsebina tabele ne odraža pravih razmer v omrežju oziroma zastara. Da se to ne zgodi, se vsebine tabel občasno na novo formirajo.



Slika 141: Primer omrežja in usmerjanje s povratnim (vzvratnim) učenjem.

Najbolj preprost način delovanja vozlišča je "usmerjanje brez usmerjanja". Vsak prispeli paket vozlišče pošlje naprej v vse možne smeri (razen v smer, iz katere prihaja paket). Tak način imenujemo preplavljanje. Preplavljanje ustvarja veliko število podvojenih paketov. Podvojeni paketi predstavljajo dodatno breme, ki zmanjša prepustnost omrežja. Zato se preplavljanje kot edina oblika usmerjanja redko uporablja, obstaja pa kot ena od možnih alternativ. Dobra lastnost preplavljanja je, da gredo paketi do naslovljenca po več poteh. Možnost, da dosežejo cilj, se poveča. Teoretično bi prišli do naslovljena paketi po vseh možnih poteh in potemtakem tudi po najkrajši (najhitrejši). Ko pride prvi paket uspešno do cilja, naslovljenec vse kasnejše dvojnike enostavno zavrže. Da se število podvojenih paketov zadrži v obvladljivih mejah, se vozlišča štejejo število prehodov paketa skozi vozlišča. Če števec preseže dovoljeno vrednost, se paket zavrže.

Vozlišče	Ocena smeri	Smer
V_1	3	2
V_2	0	1
V_3	-	-
V_4	0	3
V_5	2	1
V_6	3	2
V_7	2	3

(a)

Vozlišče	Ocena smeri	Smer
V_1	**1**	**1**
V_2	0	1
V_3	-	-
V_4	0	3
V_5	2	1
V_6	3	2
V_7	2	3

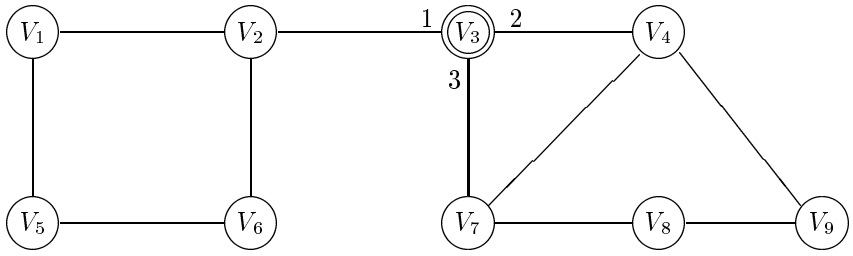
(b)

Tabela 6: Možen izgled usmerjevalne tabele za vozlišče V_3 pred (a) in po (b) prihodu paketa iz ugodnejše smeri za V_1.

7.5.2 Porazdeljeno usmerjanje

Pri porazdeljenem načinu usmerjanja si usmerjevalne tabele izmenjujejo samo neposredno povezana vozlišča. Vsako usmerjevalno vozlišče občasno pridobi vsebine tabel od neposredno sosednjih vozlišč in jih upošteva v svoji tabeli. Vsako usmerjevalno vozlišče torej občasno pošlje svojo usmerjevalno tabelo (ali samo del tabele) svojim neposrednim sosedom in ta jo upoštevajo pri gradnji svojih tabel. Sčasoma se vsebine tabel z izmenjavo preko neposrednih sosedov razširijo po vsem omrežju.

Slika 142 prikazuje primer omrežja na katerem si bomo ogledali potek gradnje usmerjevalne tabele v vozlišču V_3.



Slika 142: Primer porazdeljenega usmerjanja.

Vozlišče V_3 ima tri neposredne sosede, $S = \{V_2, V_4, V_7\}$. Vozlišče razpolaga tudi z ocenami kvalitete neposrednih povezav (lahko jih občasno oceni s pošiljanjem testnih paketov), $L = \{L_1 = (V_3, V_2), L_2 = (V_3, V_4), L_3 = (V_3, V_7)\} = \{5, 10, 15\}$. Denimo, da je V_3 od sosednjih vozlišč pridobilo vsebine tabel R_2, R_4 in R_7. Deli tabel za sosednja vozlišča in vozlišče V_3 pred in po popravku tabele prikazuje tabela 7. V tabeli R_3 je za vsako znano vozlišce v omrežju zabeležena najugodnejša smer in kvaliteta poti do danega vozlišča v tej smeri.

Novo vsebino svoje tabele (ocene in smeri) določi po naslednjem pravilu:

$$C(3,i) = \min\{L_1 + C(2,i), L_2 + C(4,i), L_3 + C(7,i)\} \quad (i = 1, 2, \ldots)$$

R_2		R_4		R_7		R_3	Prej	Smer	R_3	Po	Smer
V_i	$C(2,i)$	V_i	$C(4,i)$	V_i	$C(7,i)$	V_i	$C(3,i)$		V_i	$C(3,i)$	
V_1	15	V_1	30	V_1	40	V_1	65	3	V_1	20	1
V_2	0	V_2	15	V_2	20	V_2	5	1	V_2	5	1
V_3	5	V_3	10	V_3	15	V_3	0	-	V_3	0	-
V_4	15	V_4	0	V_4	15	V_4	10	2	V_4	10	2
...	...	...	...	...	...	...	...		...	...	

Tabela 7: Del vsebine usmerjevalnih tabel sosednjih vozlišč (R_2, R_4, R_7) in vozlišča V_3 pred in po upoštevanju tabel sosednjih vozlišč.

Primer porazdeljenega usmerjanja je usmerjanje v omrežju Internet po protokolu RIP (Routing Information Protocol). Usmerjevalniki, ki med seboj komunicirajo po protokolu RIP, pošljajo svojo usmerjevalno tabelo (ali del tabele) ali zahtevajo sosednjih vozlišč pošiljanje njihovih usmerjevalnih tabel. Uporabljena metrika (kvaliteta) je število prehodov vmesnih vozliščpo določeni poti.

Slabost porazdeljenega usmerjanja je razmeroma nestabilo delovanje. V omrežju se sicer dokaj hitro vzpostavijo optimalne poti, vendar se v primeru spremembe v omrežju (izpada nekega usmerjevalnika) ta sprememba precej pozno upošteva. V omrežju Internet je RIP zamenjal protokol OSPF (Open Shortest Path First).

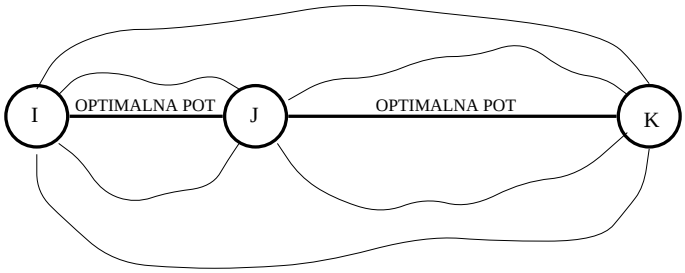
7.5.3 Globalno – centralizirano usmerjanje

Pri centraliziranem ali globalnem načinu usmerjanja se upošteva za določitev vsebine usmerjevalne tabele vsakega usmerjevalnega vozlišča stanje celotnega omrežja. Običajno eno od vozlišč(usmerjevalni center) od drugih vozlišč pridobi informacijo, na podlagi katere razbere topologijo omrežja in kvaliteto neposrednih povezav. Merilo kvalitete povezav je na primer pasovna širina povezave, prepustnost, odzivnost, geografska razdalja, cena, ali kombinacija teh. Pogosto merilo kvalitete poti je kar število vmesnih vozlišč.

Ko se na enem mestu ustvari pregled o razmerah v celotnem omrežju (ali delu omrežja), se na podlagi tega določi najugodnejše (optimalne) poti in po potrebi tudi alternativne poti med poljubnimi pari vozlišč. Usmerjevalni center potem razpošlje usmerjevalne tabele drugim vozliščem. Možno je tudi, da vsako vozlišče zase na osnovi globalne (celotne) informacije določi lastno usmerjevalno tabelo - distribucija tabel ni potrebna.

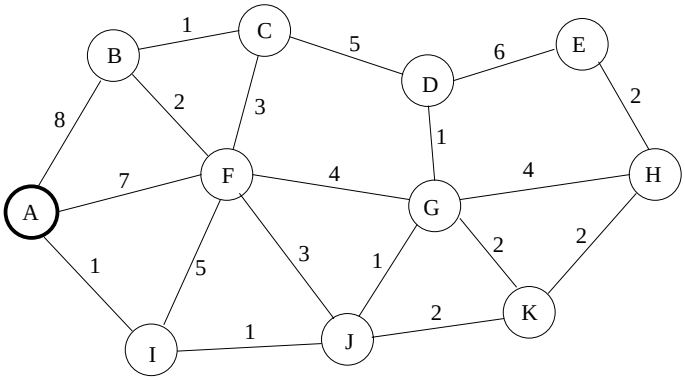
Obstaja več algoritmov za določanje optimalnih oziroma v izbrani metriki najkrajših poti. Vsi algoritmi na načelu optimalnosti (Slika 143), ki pravi: če je

vozlišče J na optimalni poti med vozliščema I in K, potem je tudi pot od vozlišča J do K optimalna. Enako velja za pot od I do J.



Slika 143: Princip optimalnosti. Če leži vozlišče J na optimalni poti med vozliščema I in K, potem je tudi pot od J do K optimalna.

V nadaljevanju bomo razložili algoritem za iskanje optimalne poti, ki ga je predlagal Dijkstra. Delovanje algoritma si najlažje razložimo na konkretnem primeru. Zato si poglejmo, kako alogiritem deluje na omrežu s topologijo in oceno kvalitete (razdalje) direktnih povezav, kot je prikazano na sliki 144.



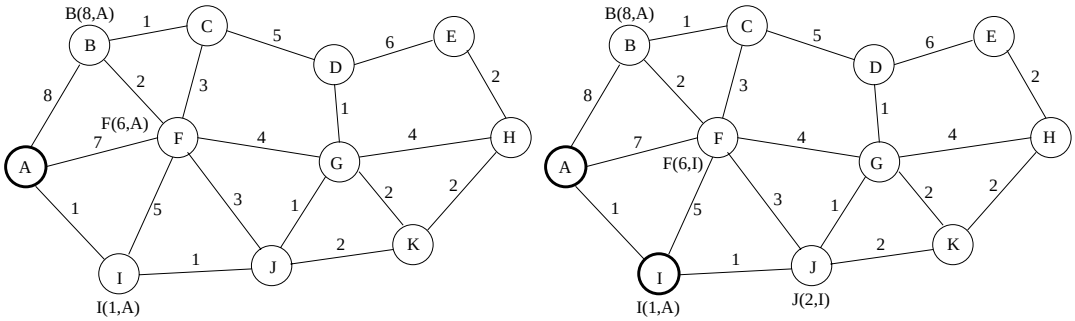
Slika 144: Iskanje optimalne poti od A do H. Prikazana je topologija omrežja in razdalje med sosednjimi vozlišči. Kot permanentno je označeno začetno vozlišče, to je A.

Naša naloga je, da poiščemo najkrajšo pot med vozliščema A in H. Začnemo v vozlišču A, ki mu sedaj rečemo "tekoče" vozlišče in napredujemo proti končnemu vozlišču. Vozlišče A je začetek poti in ga zato označimo kot permanentno, ker je gotovo na poti, ki jo iščemo. Med napredovanjem algoritma se pomikamo od vozlišča do vozlišča ter jih označujemo z najkrajšo do tedaj znano potjo od začetnega vozlišča. Med napredovanjem se oznaka vozlišča lahko popravlja samo na bolje. Sprva ni poznana nobena pot in vozlišča so označena z nedefinirano vrednostjo ali z dovolj veliko vrednostjo (v našem primeru niso označena).

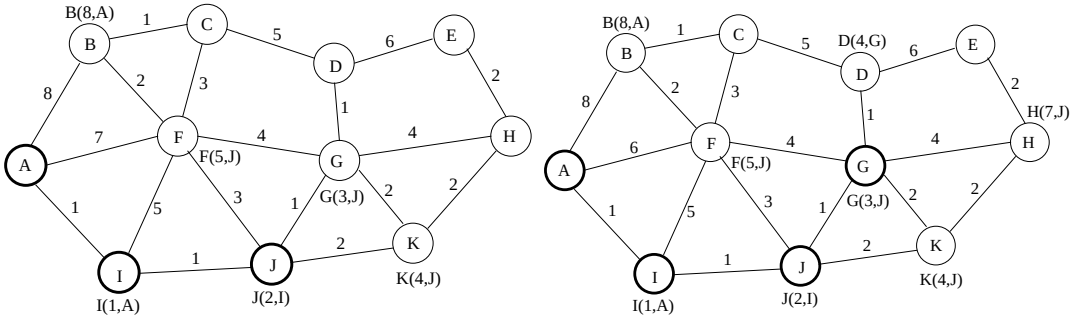
Ker je na začetku vozlišče A tekoče vozlišče, preiskusimo vsa njemu sosednja vozlišča - izračunamo razdalje do sosednjih vozlišč in ta vozlišča na novo označimo.

Oznake imenujemo “poskusne”, ker je možno, da se kasneje spremenijo. Vedno, kadar na novo označimo kako vozlišče, mu pripišemo smer, od kjer smo prišli. Označevanje smeri je potrebno zato, da bomo tedaj, ko bomo na koncu poti, znali najti pot nazaj do začetnega vozlišča. Ko označimo sosede tekočega vozlišča, preiskusimo vsa označena vozlišča v omrežju. Med njimi izberemo kot naslednje permanentno tisto vozlišče, ki ima najkrajšo razdaljo do začetnega vozlišča. To vozlišče postane novo tekoče vozlišče. med napredovanjem algoritma se oznaka tega vozlišča večne spreminja.

Po prvem koraku ima vozlišče I najugodnejšo oznako. Zato ga označimo kot permanentnega in izberemo za tekočega (Slika 145). Nadaljujemo iz vozlišča I in preizkusimo njemu sosednja vozlišča. Najugodnejša je izbira vozlišča J, ki ga označimo za permanentno in nadaljujemo iz njega (Slika 146). Na enak način nadaljujemo, dokler ne označimo končnega vozlišča. S pomočjo oznak smeri najdemo pot nazaj do začetnega vozlišča (glej slike 144 do 148 in tabelo 8). Na podoben način poiščemo najkrajše poti med ostalimi pari vozlišči.



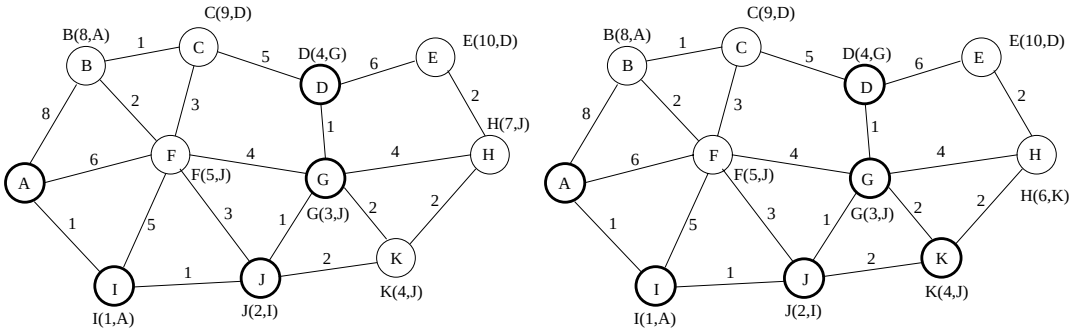
Slika 145: Določanje optimalne poti od A do H, koraka 1 in 2.



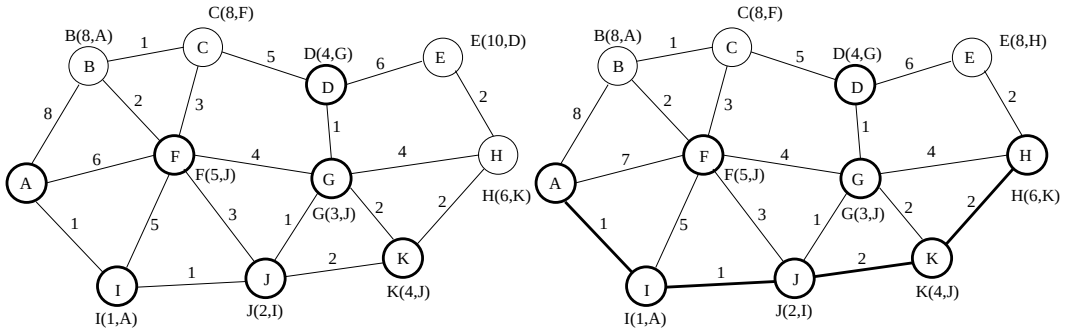
Slika 146: Določanje optimalne poti od A do H, koraka 3 in 4.

7.5.4 Usmerjanje v omrežju Internet

Omrežje Internet deluje po načelu datagram. Osnova za delovanje omrežja je IP protokol. Z IP paketi se prenaša ves omrežni promet. Usmerjanje poteka



Slika 147: Določanje optimalne poti od A do H, koraka 5 in 6.



Slika 148: Koraka 7 in 8 ter optimalna pot od A do H.

od usmerjevalnega do naslednjega usmerjevalnega vozlišča. Na podlagi ciljnega naslova (naslova končnega vozlišča) usmerjevalna vozlišča izbirajo smer, v katero se pošlje dospeli paket. Poti, po kateri poteka prenos paketa ne poznajo, pa je za usmerjanje tudi ne potrebujejo.

V pogledu usmerjanja je delovanje končnih in usmerjevalnih vozlišč zelo podobno. Glavna razlika je v tem, da končna vozlišča nikoli ne pošiljajo prispelih paketov naprej, medtem ko usmerjevalniki po potrebi pošiljajo (posredujejo) pakete naprej - imajo več mrežnih priključkov (komunikacijskih vmesnikov). Vozlišča vzdržujejo usmerjevalne tabele, ki izgledajo nekako tako, kot je prikazano v tabeli 9. Tabela je bila dobljena z UNIX ukazom

`netstat -rn`

na končnem vozlišču z imenom `luz.fe.uni-lj.si`, čigar IP naslov je 193.2.72.140.

V usmerjevalni tabeli je za vsako znano ciljno vozlišče v omrežju ali skupino vozlišč (omrežju ali delu omrežja) zabeleženo, ali je to ciljno vozlišče z njim direktno povezano ali ne. Če ni, potem je v tabeli zabeležen naslov usmerjevalnika, ki bo poskrbel, da se paket pošlje naprej v smer, ki vodi do ciljnega vozlišča.

Stolpci tabele imajo naslednji pomen:

- IP ciljni naslov (ang. Destination IP address): to je poln naslov vozlišča ali

Št. Koraka	Tekoče vozlišče	Poskusne oznake	Permanentno vozlišče
1	A	B(3,A), F(7,A), I(1,A)	I(1,A)
2	I(1,A)	J(2,I), F(6,I)	J(2,I)
3	J(2,I)	K(4,J), G(3,J), F(5,I)	G(3,J)
4	G(3,J)	H(7,G), D(4,G)	D(4,G)
5	D(4,G)	E(10,D), C(9,D)	K(4,J)
6	K(4,J)	H(6,K)	F(5,J)
7	F(5,J)	C(8,F)	H(6,K)
8	H(6,K)		

Tabela 8: Zaporedje korakov za določitev optimalne poti od A do H. Na podlagi oznak sledimo pot nazaj od končnega proti začetnemu vozlišču.

Destination	Gateway	Flags	Refs	Use	Interface	Pmtu	PmtuTime
127.0.0.1	127.0.0.1	UH	0	320455	lo0	4608	
193.2.72.140	127.0.0.1	UH	27	3201312	lo0	4608	
193.2.72	193.2.72.140	U	9	5245675	lan0	1500	
default	193.2.72.65	UG	55	3004572	lan0	1500	

Tabela 9: Primer usmerjevalne tabele končnega vozlišča v omrežju Internet.

naslov omrežja (zgornji del, t.j. omrežni del IP naslova).

- IP naslov usmerjevalnika (ang. IP address of a next-hop-router) ali naslov direktno vezanega omrežja (ang. IP address of a directly connected network). Usmerjevalnik prevzema pakete, ki jih (lokalna) vozlišča preko njega pošiljajo vozliščem na drugi strani.

- Zastavice (ang. Flags): Zastavica U pomeni, da je povezava aktivna (Up). Zastavica H (Host) pove ali je ciljni naslov naslov omrežja ali vozlišča. Če je H postavljen je to naslov vozlišča. Zastavica G (Gateway) pove ali je v stolpcy usmerjevalnik res naslov usmerjevalnika ali pa gre za direktno vezano vozlišče. V primeru, da G ni postavljen, gre za direktno povezano vozlišče.

- Ime vmesnika, na katerega se pošlje paket.

- Ostala določila (število aktivnih povezav, št. prenešenih paketov, ...).

Tabela 9 vsebuje v prvi vrstici pravilo usmerjanja za pakete, ki jih vozlišče pošlje samo sebi (lokalne pakete). V tem primeru gre za komunicikacijo procesov, ki obstajajo na tem vozlišču in komunicirajo po TCT/IP protokolu. To je t.i. "loopback interface". Za ta namen rezerviran naslov je 127.0.0.1. Podobno je z drugo vrstico. Paketi, ki so poslani na naslov 193.2.72.140, to pa je naslov

vozlišča samega, se zaključijo direktno - lokalno v okviru dejavnosti mrežnega sloja na tem vozlišču in ne obremenjujejo omrežja. Tretja vrstica vsebuje kot ciljni naslov naslov omrežja (193.2.72) in naslov usmerjevalnika, ki je zadolžen za dostavo paketov naslovljenih na to omrežje. Zadnja vrstica je vsebuje naslov privzetega usmerjevalnika (193.2.72.65) za vse ostale ciljne naslove (ang. "default router"). Samo v zadnji vrstici gre za "pravo" usmerjanje naprej preko usmerjevalnika z naslovom 193.2.72.65 (postavljen je G), ostali naslovi so naslovi direktno vezanih vozlišč. V prvih dveh vrsticah je ciljni naslov polni naslov vozlišča (H je postavljen), v ostalih dveh primerih gre za naslov omrežja.

Vsebina usmerjevalne tabele se običajno določi ob začetnem zagonu vozlišča. Za formiranje (dodajanje) vrstic tabele služi UNIX ukaz `route`. Na primer:

```
route add default ags-fer.fer.uni-lj.si 1
```

doda vrstico s ciljnimi naslovi (default) in ime usmerjevalnika, ki je zadolžen za usmerjanje. To je "pravi" usmerjevalnik. Zadnja enka namreč pomeni število vmesnih vozlišč do ciljnega (default) naslova - to pa je navedeni (vsaj razume se tako) usmerjevalnik. Nasprotno bi ničla bi pomenila direkten priključek.

V pomoč sledenju paketov obstaja UNIX ukaz `traceroute`. Izvršitev ukaza:

```
traceroute razor.fri.uni-lj.si
```

na vozlišču

```
luz.fe.uni-lj.si
```

da naslednji rezultat:

```
traceroute to razor.fri.uni-lj.si (193.2.76.37), 30 hops max, 20 byte packets
 1 ags-fer.fer.uni-lj.si (193.2.72.65)              1 ms    1 ms    1 ms
 2 razor.fer.uni-lj.si (193.2.76.37)                2 ms    2 ms    2 ms
```

Algoritem usmerjanja pa je naslednji:

- Ciljni naslov paketa se primerja s ciljnimi naslovi tabele. V primeru zadetka, se postopa takole. Če je vozlišče s ciljnim naslovom direktno priključeno, se pošlje na ta naslov. V nasprotnem primeru se pošlje usmerjevalniku, ki je naveden v pripadajoči vrstici tabele.
- V primeru, da naslova vozlišča ni v tabeli, se v tabeli išče ujemenje naslova ciljnega omrežja paketa, naprej se postopa enako.
- Če se naslova v tabeli ne najde, se uporabi privzeti usmerjevalnik.

V primeru, da se ciljnega vozlišča ne najde, sledi sporočilo napake “host unreachable” ali “network unreachable”.

Literatura

[1] W. R. Stevens, *TCP/IP Illustrated, Vol. 1*, Addison–Wesley, 1994.

[2] A. Tanenbaum, *Computer Networks*, 3rd ed. Prentice-Hall 1996.

8 Elementi višjih slojev

V tem poglavju pobo obravnavali nekatere elemente višjih slojev arhitekturnega modela. Začeli bomo z zgoščevanjem in nadaljevali s šifriranjem. Oba sta navadno, a ne nujno, del sloja predstavitve.

8.1 Zgoščevanje podatkov

8.1.1 Uvod

Kodiranje je postopek, s katerim prvoten zapis podatkov spremenimo v nov zapis podatkov. Pravilo, ki določa, kako podatek ali zaporedje podatkov (za)kodiramo, imenujemo *kod*. S kodiranjem dobimo kodiran zapis podatkov, ki ga z obratnim postopkom *dekodiramo* ('odkodiramo'), da dobimo prvoten zapis. Običajno pri tem želimo, da ostane informacijska vsebina nespremenjena. Kodiramo zaradi različnih ciljev. Kodiramo na primer zato, da bi povečali zanesljivost prenosa ali shranjevanja podatkov. Kodiramo pa tudi zato, da bi informacijsko vsebino sporočila čim krajše zapisati (*zgostiti podatke*) in s tem informacijski pretok ali zmanjšali potreben pomnilni prostor za shranjevanje.

8.1.2 Osnovna zamisel

Problem kodiranja zastavimo takole: imamo množico simbolov $\mathcal{S}$ z N simboli. Dane so tudi verjetnosti simbolov (porazdelitveni zakon).

$$S = \begin{pmatrix} s_1, & s_2, & \ldots, & s_i, & \ldots, & s_N \\ p_1, & p_2, & \ldots, & p_i, & \ldots, & p_N \end{pmatrix}. \tag{40}$$

Simbolom pripada entropija $H(S)$,

$$H(S) = -\sum_{i=1}^{N} p_i \, log_2 \, p_i \quad [bitov]. \tag{41}$$

Simbolom $\mathcal{S}$ bomo priredili zaporedja dvojiških (binarnih) simbolov. Označimo dvojiška *kodna simbola*, kot je v navadi, z 0 in z 1. Simbolom

$$\mathcal{S} = \{s_i\}, \qquad (i = 1, 2, \ldots, N) \tag{42}$$

torej priredimo zaporedja kodnih simbolov $b \in \{0, 1\}$. Dobljenim zaporedjem kodnih simbolov rečemo *kodne besede* in jih označimo z w_i,

$$w_i = (b_{1_i} \; b_{2_i} \; \ldots b_{k_i} \ldots b_{n_i}) \qquad (i = 1, 2, \ldots, N) \quad (k = 1, 2, \ldots, n_i) \tag{43}$$

Množici vseh kodnih besed w_i pravimo *kod* in jo označimo z W,

$$W = \{w_i\}, \qquad (i = 1, 2, \ldots, N). \tag{44}$$

Kod W popolnoma določa, kako simbole $\mathcal{S}$ preslikamo v zaporedja kodnih simbolov, $\mathcal{S} \Rightarrow W$. Kodne besede so v splošnem poljubno in različno dolge, imajo različne dolžine n_i. Naš osnovni cilj je, da izberemo takšne dolžine kodnih besed, da bi v povprečju dosegli čim krajšo dolžino kodnih besed $\bar{n}$:

$$\bar{n} = \sum_{i=1}^{N} n_i p_i. \tag{45}$$

Takoj se prepričajmo, da je povprečna dolžina besed $\bar{n}$ vedno večja ali kvečjemu enaka $H(S)$,

$$\bar{n} \geq H(S). \tag{46}$$

Vzemimo, da je verjetnost simbola s_i enaka p_i. Tej pripada entropija (informacijska vsebina) $log_2(1/p_i)$ bitov, ki se jo da vedno zapisati (zakodirati) z $n_i = \lceil log_2(1/p_i) \rceil$ binarnimi simboli, kjer $\lceil x \rceil$ pomeni najmanjše celo število večje ali enako x:

$$log_2 \frac{1}{p_i} \leq n_i < log_2 \frac{1}{p_i} + 1. \tag{47}$$

Množimo izraz s p_i:

$$p_i \; log_2 \; \frac{1}{p_i} \;\; \leq \;\; p_i \; n_i \;\; < \;\; p_i \; log_2 \; \frac{1}{p_i} + p_i \; 1 \tag{48}$$

in opravimo seštevanje po vseh i, $(i = 1, 2, \ldots, N)$. Dobimo

$$\sum_{i=1}^{N} p_i log_2 \frac{1}{p_i} \leq \sum_{i=1}^{N} p_i n_i < \sum_{i=1}^{N} p_i log_2 \frac{1}{p_i} + \sum_{i=1}^{N} p_i 1. \tag{49}$$

Končno:

$$H(S) \leq \bar{n} < H(S) + 1, \tag{50}$$

kar smo hoteli pokazati. Povprečna dolžina kodnih besed je kvečjemu enaka entropiji $H(S)$ in ne more biti manjša od nje, kakorkoli že kodiramo. Ni pa treba, da bi bila večja od $H(S) + 1$.

Osnovna parametra, s katerima lahko ovrednotimo uspešnost kodiranja, sta koristnost in odvečnost. *Koristnost* E je definirana z razmerjem med informacijsko vsebino signala in celotnim številom binarnih simbolov, s katerim zakodiramo signal. Za naš primer:

$$E = \frac{H(S)}{\bar{n}} \quad (0 \leq E \leq 1). \tag{51}$$

Odvečnost R je določena z razmerjem med številom *odvečnih* binarnih simbolov in številom vseh simbolov:

$$R = \frac{\bar{n} - H(S)}{\bar{n}} = 1 - \frac{H(S)}{\bar{n}} = 1 - E \tag{52}$$

Ko je $\bar{n} = H(S)$ je koristnost 1 in odvečnost je nič.

Poleg tega, da hočemo doseči v povprečju čim krajše kodne besede ($R \to 1$), upoštevamo pri kodiranju še naslednje zahteve:

- kod mora biti *regularen*, kar pomeni, da različnim vrednostim s_i pripadajo različne kodne besede,
- kod mora biti *enoznačen*, kar pomeni, da mora biti možno vsako sporočilo čitati (dekodirati) na en sam način,
- kod naj bo *trenuten* ali sproti čitljiv. To pomeni, da se da vsaka kodna beseda, takoj ko je v celoti sprejeta, tudi takoj dekodirati.

Da razjasnimo regularnost, enoznačnost in trenutnost koda, poglejmo naslednji primer. Zakodirajmo stanja signala $\mathcal{S} = \{s_1, s_2, s_3\}$ na štiri načine:

$\mathcal{S}$	Kod W_1	Kod W_2	Kod W_3	Kod W_4
s_1	0	0	0	0
s_2	0	1	001	10
s_3	1	01	101	11

Kod W_1 očitno ni regularen, saj smo dvema različnima stanjema (s_1 in s_2) priredili enako kodno besedo (0).

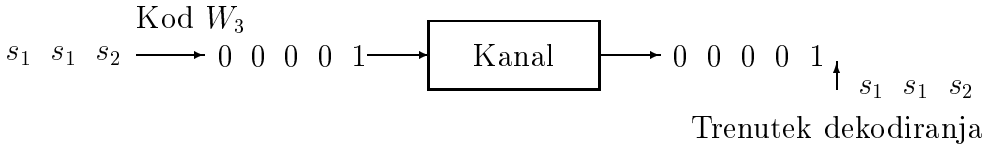
Kod W_2 ni enoznačen. Vzemimo zaporedje stanj $s_1 s_2 s_3$. Ko ga zakodiramo, dobimo 0101. Če skušamo to zaporedje binarnih simbolov dekodirati, naletimo na težave. Zaporedje 0101 bi lahko pomenilo $s_1 s_2 s_1 s_2$ ali $s_3 s_3$ ali $s_1 s_2 s_3$.

$$s_1\ s_2\ s_3\ \to 0\ 1\ 0\ 1\ \to \begin{cases} s_1\ s_2\ s_1\ s_2 \\ s_3\ s_3 \\ s_1\ s_2\ s_3 \end{cases}$$

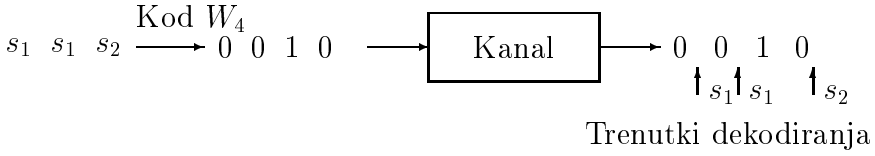
Zadnje od teh zaporedij je sicer pravilno, vendar se iz zakodiranega sporočila tega ne da ugotoviti, razen če vpeljemo še dodaten ločilni znak (dodatno kodno besedo) in s tem povečamo odvečnost.

Kod W_3 je regularen in enoznačen, ni pa trenuten. Da bi se v to prepričali, si mislimo zaporedje $s_1 s_1 s_2$ in njemu prirejeno zaporedje kodnih simbolov 00001.

Denimo, da sprejmemo prvi kodni simbol, ki je 0. Dokler ne sprejmemo naslednjega simbola, lahko ta simbol pomeni stanje s_1 ali začetek kodne besede 001, ki pomeni stanje s_2. Nič bolje ni, ko sprejmemo še drugi simbol (0). Simbola 00 lahko pomenita dve stanji s_1 ali sta začetek kodne besede 001. V našem primeru znamo zaporedje kodnih simbolov (kodirano sporočilo) dekodirati šele, ko ga sprejmemo v celoti.



Če bi bil kod trenuten, bi se vsako sporočilo dalo sproti dekodirati, kar si želimo. Kod W_4 je trenuten, pa tudi regularen in enoznačen. Vsako zaporedje simbolov se da sproti čitati, kot je narisano na sliki.



8.1.3 Optimalno kodiranje - Huffmanov kod

S kratkim razmislekom se prepričajmo v naslednje lastnosti optimalnega kodiranja, na katerih temelji Huffmanov postopek kodiranja:

- Stanjem z večjo verjetnostjo prirejamo krajše kodne besede:

  $$p_i > p_j \longrightarrow n_i \leq n_j.$$

  To je logično. Da bo sporočilo vsebovalo manj kodnih simbolov, pogostejša stanja (tista, ki jih je v sporočilu več) zakodiramo s krajšimi kodnimi besedami.

- Stanji z najmanšo verjetnostjo imata enako dolgi kodni besedi.

- Najdaljši kodni besedi se razlikujeta samo na enem, in sicer na zadnjem simbolu.

V zadnji dve trditvi se prepričamo takole. Ker sta stanji najmanj verjetni, sta njuni kodni besedi daljši od vseh ostalih in se od njih tudi gotovo razlikujeta. Razlikovati se morata tudi besedi sami. Dovolj je, da se razlikujeta na enem simbolu. Torej, če bi se razlikovali na več kot enem simbolu in ne bi bili enako

dolgi, bi te simbole smeli preprosto odstraniti, vse do zadnjega različnega simbola. Dobili bi kodni besedi, ki sta enako dolgi in kod z manjšo popvrečno dolžino kodnih besed.

Huffmanov postopek je zasnovan na sledeči predpostavki. Vzemimo, da želimo optimalno zakodirati signal S_N, ki ima N možnih stanj. Vzemimo, da tega ne znamo storiti, znamo pa optimalno zakodirati signal z $N-1$ stanji. No, signal S_N se da vedno obravnavati kot signal z $N-1$ stanji, če dve stanji združimo v eno stanje. Odločimo se, da združimo stanji z najmanjšo verjetnostjo. Označimo to stanje s $s_{N-1,N}$. Verjetnost stanja $s_{N-1,N}$ je enaka vsoti verjetnosti stanj s_{N-1} in s_N. Dobljeni signal S_{N-1} zakodiramo. Predpostavili smo, da ga znamo zakodirati optimalno. Recimo, da je dobljeni kod

$$W_{N-1} = \{w_1, w_2, \ldots, w_i, \ldots, w_{N-2}, w_{N-1,N}\},$$

pri čemer smo združenemu stanju $s_{N-1,N}$ priredili kodno besedo $w_{N-1,N}$. Ko imamo kod W_{N-1} se da z upoštevanjem tretjega dejstva priti do koda W_N za prvotni signal S. Poglejmo kako. Združeno stanje $s_{N-1,N}$ razdružimo v prvotni stanji, ki imata (kot smo se domenili) najmanjši verjetnosti, obe prvotni stanji s_{N-1} in s_N pa zakodiramo tako, da kodni besedi $w_{N-1,N}$ pripišemo enkrat 0 in drugič 1. Dobimo enako dolgi kodni besedi

$$w_{N-1} = w_{N-1,N}\,0 \qquad in \qquad w_N = w_{N-1,N}\,1,$$

ki se razlikujeta samo na zadnjem mestu. Ker sta stanji najmanj verjetni, sta kodni besedi tudi najdaljši. Torej je dobljeni kod W_N,

$$W_N = \{w_1, w_2, \ldots, w_i, \ldots, w_{N-2}, w_{N-1,N}0, w_{N-1,N}1\},$$

ki zakodira prvotni signal, optimalen, če je kod W_{N-1} optimalen. Ostane vprašanje, kako optimalno zakodirati signal S_{N-1}. No, na signalu S_{N-1} uporabimo prvotno predpostavko, to je, da znamo optimalno zakodirati signal, ki ima eno stanj manj. Zato spet združimo stanji z najmanjšo verjetnostjo. Dobimo signal S_{N-2}, ki ima še eno stanje manj. Ta signal zakodiramo s kodom W_{N-2}, iz koda W_{N-2} pa pridemo do koda W_{N-1} tako kot smo prej prišli iz koda W_{N-1} do koda W_N. Postopek zdrževanja nadaljujemo, dokler ne ostaneta samo dve stanji, ki ju znamo (optimalno) zakodirati na dva enakovredna načina, bodisi z 0 in 1 bodisi z 1 in 0. Če si zapomnimo zaporedje združevanja, znamo z razdruževanjem stanj nazaj do prvotnega signala in hkrati do optimalnega koda za prvotni signal.

Opišimo postopek kodiranja v celoti:

Vhod : Signal $S_N = (\mathcal{S}_N, \mathcal{P}_N)$ z N stanji
Izhod : Kod W_N z N kodnimi besedami

Uredi(S_N, S_N^u)

$k = N$

dokler je $k > 1$ **ponavljaj** {

    Združi(S_k^u, S_{k-1})

    Uredi(S_{k-1}, S_{k-1}^u)

    $k = k - 1$

}

Pojasnilo: Začnemo s kodom signala S_2^u

$W_2 = \{0, 1\}$

$k = 2$

Dokler je $k < N$ **ponavljaj** {

    Razdruži(S_k^u, S_{k+1}^u)

    Kodiraj(W_k, S_{k+1}^u, W_{k+1})

    $k = k + 1$

}

Združi(S_k^u, S_{k+1}) je operacija, ki združi stanji z najmanjšo verjetnostjo:

Združi(S_k^u, S_{k-1})

Za vse i **od** 1 **do** $k - 2$ {

    $p_i = p_i^u$

}

$p_{k-1} = p_{k-1}^u + p_k^u$

$s_{k-1} = [s_{k-1}^u, s_k^u]$

Operacija **Uredi**(S_k, S_{k+1}^u) preprosto uredi stanja signala S_k po padajočih verjetnostih, rezultat pa je urejen signal S_k^u. Operacija **Razdruži** je obratna operacijama **Uredi** in **Združi**. Končno, operacija **Kodiraj**(W_x, S_y^u, W_y) zakodira stanja signala S_y^u s kodom W_y, ki ga izpelje iz koda W_x ($y = x + 1$),

Kodiraj(W_x, S_y^u, W_y)

Za vse i **od** 1 **do** $x - 1$ {

    $w_{i_y} = w_{i_x}$

}

$w_{x_y} = w_{x_x}0$

$w_{y_y} = w_{x_x}1$

}

Pstopek kodiranja bo bolj jasen na preprostem primeru. Vzemimo signal S,

$$S = \begin{pmatrix} s_1, & s_2, & s_3 & s_4, & s_5 \\ 0.30 & 0.25 & 0.20 & 0.15 & 0.10 \end{pmatrix}.$$

Stanja so že urejena, prvo urejanje zato odpade. Na sliki 149 je narisan postopek združevanja stanj in preurejanja. Postopek se konča v treh korakih, ko ostaneta samo še dve stanji. Slika 150 prikazuje obratno pot, kjer stanjem $s_{\alpha,\beta}$ prirejamo kodne besede $w_{\alpha,\beta}$. Vedno, ko stanje $s_{\alpha,\beta}$ razdružimo na stanji s_α in s_β , dodamo kodni besedi $w_{\alpha,\beta}$ en simbol. Dobimo kodni besedi $w_\alpha = w_{\alpha,\beta}0$ in $w_\beta = w_{\alpha,\beta}1$.

Izračunajmo še povprečno dolžino kodnih besed:

$$\bar{n} = 2 \times 0.35 + 2 \times 0.25 + 2 \times 0.20 + 3 \times 0.15 + 3 \times 0.10 = 2.25$$

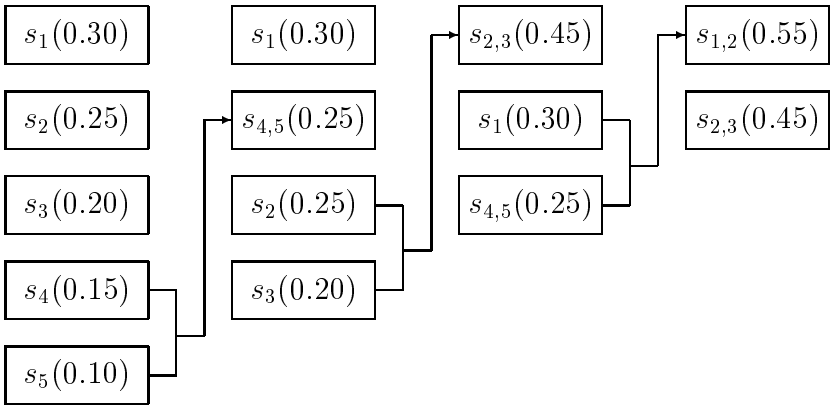
in entropijo:

$$\begin{aligned} H(S) = - \quad (\quad & 0.35\, log_2\, 0.35 + 0.25\, log_2\, 0.25 + 0.20\, log_2\, 0.20 \\ + \quad & 0.15\, log_2\, 0.15 + 0.10\, log_2\, 0.10) = 2.24. \end{aligned}$$

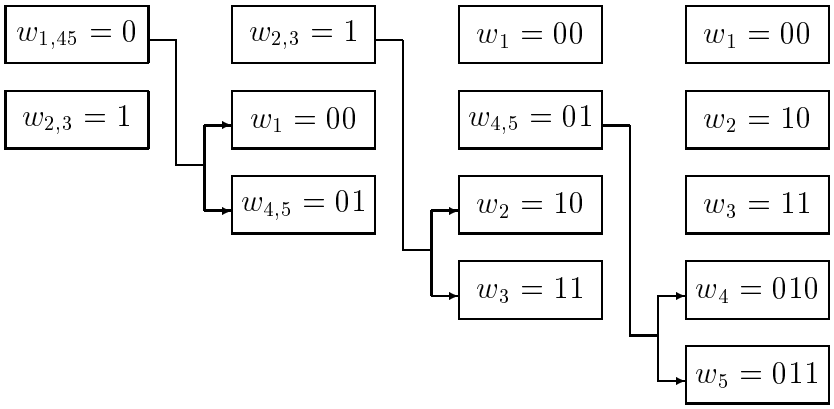
Vidimo, da je povprečna dolžina $\bar{n}$ sicer večja od entropije $H(S)$, vendar sta si $\bar{n}$ in $H(S)$ prav blizu. Odvečnost koda je:

$$R = \frac{\bar{n} - H(S)}{\bar{n}} = \frac{2.25 - 2.24}{2.25} = 0.004^{\cdot}.$$

Če bi petim stanjem signala S priredili enako dolge kodne besede, bi potrebovali 3 kodne simbole. Dva simbola sta premalo, trije pa so že preveč ($2^2 = 4 < 5 < 2^3$). Da bi zakodirali zaporedje stanj signala S dolžine $G = 100$ rabimo v tem primeru $3 \times 100 = 300$ binarnih simbolov. Pri kodiranju s Huffmanovim postopkom pa zadostuje samo $2.25 \times 100 = 225$ simbolov, kar je samo en simbol več od teoretične spodnje meje 224 simbolov.



Slika 149: Postopek združevanja in preurejanja stanj.

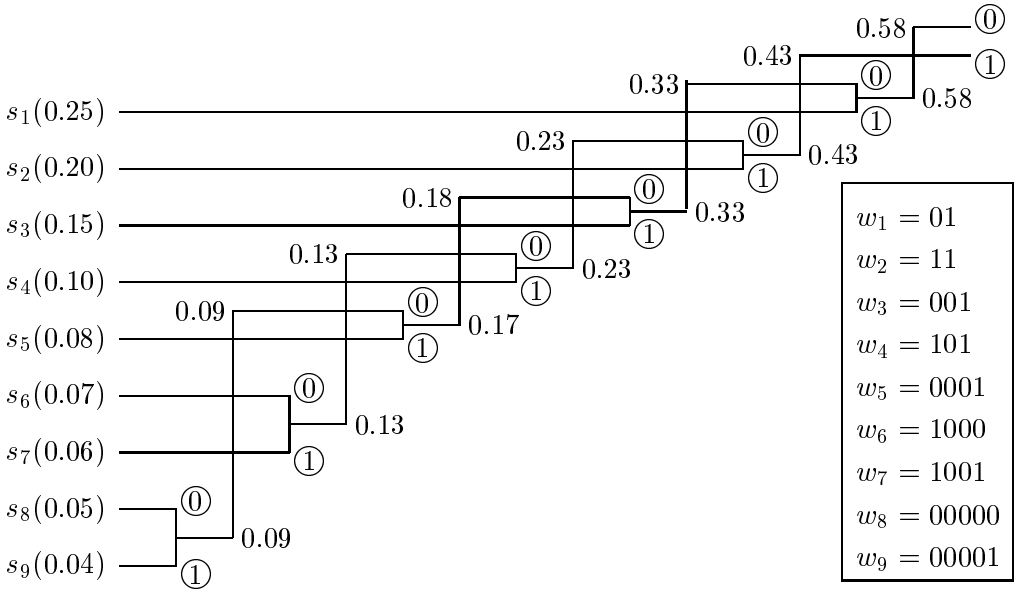


Slika 150: Postopek razdruževanja in kodiranja stanj.

Naredimo še en primer. Zakodirajmo po Huffmanovem postopku naslednji signal:

$$S = \begin{pmatrix} s_1, & s_2, & s_3, & s_4, & s_5, & s_6, & s_7 & s_8, & s_9 \\ 0.25, & 0.20, & 0.15, & 0.10, & 0.08, & 0.07, & 0.06, & 0.05, & 0.04 \end{pmatrix}.$$

Postopek ponazorimo v skrčeni obliki na sliki 151. Bralec naj sam izračuna entropijo signala in povprečno dolžino besed koda.



Slika 151: Ponazoritev kodiranja po Huffmanovemu postopku. Simbole združujemo in preurejamo. Ko končamo, označimo razvejišča z 0 in z 1. Kodno besedo w_i dobimo tako, da se pomikamo od stanja s_i po začrtani poti na desno do konca poti in beležimo simbole, ki jih pri tem obiščemo. Simbole zapisujemo povrsti od desne proti levi.

Za konec poglejmo dva primera kodiranja signala, ki ga opazujemo čez daljše časovno obdobje. V prvem primeru bomo kodirali sproti, stanje za stanjem. V drugem primeru bomo počakali, da se realizirata dve stanji signala in nato zakodirali obe stanji s skupno kodno besedo. Denimo, da se signal S neprimerno verjetneje nahaja v stanju s_1 kot v stanju s_2,

$$S = \begin{pmatrix} s_1, & s_2 \\ 0.9, & 0.1 \end{pmatrix}$$

Entropija signala je $H(0.9, 0.1) = 0.469$. Zakodirajmo signal binarno in sproti. Možna sta dva enakovredna načina kodiranja, $W = \{0, 1\}$ ali $W = \{1, 0\}$. V

vsakem primeru je 'povprečna' dolžina kodnih besed enaka 1. Na primer, zaporedju dvajsetih stanj

$$s_1s_1s_1s_2s_1s_1s_1s_1s_1s_1s_1s_2s_1s_1s_1s_1s_1s_1s_1s_1$$

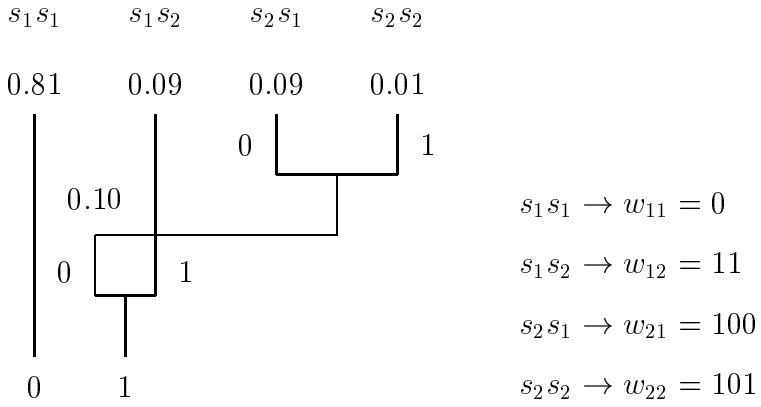
ustreza naslednje zaporedje dvajsetih kodnih simbolov,

$$00010000000100000000.$$

Sedaj pa poskusimo kodirati po dve zaporedni stanji hkrati. Možne so štiri kombinacije stanj. Lahko tudi rečemo, da kodiramo signal, ki ima štiri stanja,

$$S_2 = \begin{pmatrix} s_1s_1, & s_1s_2, & s_2s_1, & s_2s_2 \\ 0.81, & 0.09, & 0.09 & 0.01 \end{pmatrix}.$$

Verjetnosti sestavljenih stanj so enake produktom verjetnosti posameznih stanj, ker so stanja signala med seboj neodvisna. Entropija signala S_2 je $H_2(0.81, 0.09, 0.09, 0.01) = 2 \times H(0.9, 0.1) = 0.938$. Zakodirajmo signal S_2 (po Huffmanu):



Povprečna dolžina kodnih besed koda $W_2 = \{w_{11}, w_{12}, w_{21}, w_{22}\}$ je

$$\bar{n_2} = 1 \times 0.81 + 2 \times 0.09 + 3 \times 0.09 + 3 \times 0.01 = 1.29$$

vendar z vsako besedo zakodiramo dve stanji signala S. Na eno stanje signala S tako odpade v povprečju $1.29/2 = 0.645$ kodnih simbolov. Lahko rečemo, da smo pri sprotnem kodiranju rabili po en binarni simbol na stanje signala, v drugem primeru pa shajamo v povprečju že z 0.545 simbola. Z drugimi besedami, če zakodiramo zaporedje tisočih stanj signala S, rabimo v prvem primeru tisoč binarnih simbolov, a v drugem primeru 545 binarnih simbolov. Za naše zaporedje dvajsetih stanj pa rabimo 12 simbolov:

$$010000100000.$$

Če bi jemali skupaj še več kot dve stanji, bi shajali še z manj simboli. Vendar, kakorkoli že kodiramo, ne moremo doseči manj simbolov na eno stanje signala od entropije $H(S) = 0.469$.

8.1.4 Algoritem LZW

Glavno vprašanje pri kodiranju po Huffmanu je kako čimbolje oceniti statistično zakonitost sporočila, ki ga kodiramo, t.j. koko oceniti verjetnosti simbolov. Algoritem, ki sta ga predlagala Lempel in Ziv, Welch pa ga je izpopolnil, reši tudi ta problem. Ime algoritma sestavljajo začetnice avtorjevih priimkov - LZW. Algoritem analizira vhodno sporočilo in različno dolga zaporedja vhodnih simbolov nadomešča s kodnimi besedami enake dolžine. Pri tem uporablja kodno tabelo. Sprva so v tabeli kodne besede za posamezne vhodne simbole, med kodiranjem pa kodno tabelo sproti izpopolnjuje. Če v vhodnem zaporedju naleti na zaporedje simbolov, za katerega že ima v kodni tabeli nadomestno kodno bsesedo, ga nadomesti s to kodno besedo. Dolge nize vhodnih simbolov tako nadomešča (kodira) s krajšimi nizi kodnih simbolov. Da bi dekodirnik znal dekodirati sprejeto sporočilo, bi načeloma potreboval kodno tabelo. Vendar kodirnik gradi tabelo na ta način, da jo je možno zgraditi na podlagi sprejetega kodiranega sporočila. Dekodirnik si zgradi dekodirno tabelo sam.

Algoritem deluje takole. Vedno, kadar kodirnik naleti na zaporedje vhodnih simbolov za katerega v tabeli že ima kodno besedo, skuša zaporedje podaljšati še za en simbol. To ponavlja toliko časa, dokler več ne najde pripadajoče kodne besede. To zaporedje je za en simbol daljše od najdaljšega dotlej znanega. Temu zaporedju dodeli novo kodno besedo, pošlje pa kodno besedo za en vhodni simbol krajšega zaporedja. Novo kodno besedo bo uporabil prvič, ko bo na vhodu naletel na ravno tako zaporedje.

Algoritem je tak.

```
Inicializiraj kodno tabelo tako, da vsebuje
kodne besede za posamezne vhodne simbole
Beri prvi simbol, s
P = s
Ponavljaj do konca vhodnega niza {
      beri naslednji simbol, s
      E = s
      če je PE v kodni tabeli {
            P = PE
      sicer
            oddaj kodno besedo od P, w(P)
            dodaj PE v kodno tabelo, priredi w(PE)
            P = E
      }
}
```

Pa si poglejmo postopek na konkretnem primeru. Naj bo vhodno zaporedje sestavljeno iz treh različnih simbolov, a, b, in c. Kodne besede naj bodo kar desetiška števila. Sprva je kodna tabela taka:

Vhodni simbol	Kodna beseda
a	1
b	2
c	3

Naj bo vhodno zaporedje simbolov naslednje:

aaabbcaab (in tako naprej)

Kodirnik bere vhodni niz. V njem najde a, ki mu skuša dodati naslednji simbol, ki je tudi a. Zaenkrat niza aa nima v tabeli, zato odda besedo za a, ki je 1, v kodno tabelo pa vpiše novo kodno besedo za aa ter inicializira začetek iskanja novega niza.

Vhodni simbol	Kodna beseda
a	1
b	2
c	3
aa	4

Kodirnik prebere novi simbol, ki je spet a in ga pripne k na zadnje prebranemu simbolu, ta je bil tudi a. V tabeli sedaj že ima kodno besedo niza aa, zato nadaljuje in prebere še en simbol, tako dobi vhodni niz aab. Ta niz v tabeli nima kodne besede, zato odda kodo za aa, ki je 4, doda v tabelo kodno besedo za aab ter nadaljuje z analizo. Oddano zaporedje je sedaj videti takole:

1 4

Na podoben način kodirnik nadaljuje do konca vhodnega niza.

8.2 Prikrivanje podatkov

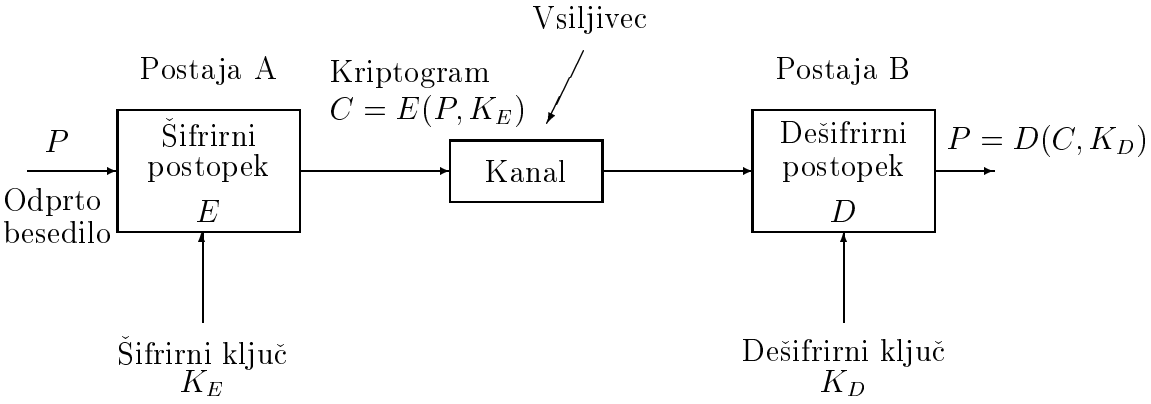
V tem podpoglavju bomo spoznali osnovne pojme v zvezi s prikrivanjem podatkov in razložili nekatere važnejše pristope, ki prispevajo k tajnosti in verodostojnosti (avtentičnosti) prenosa ali shranjevanja informacije.

8.2.1 Model sistema za prikrivanje podatkov

Osnovni namen postopkov za *prikrivanje podatkov* je prikrivanje informacije (ali zaščita) pred vsiljivcem. *Vsiljivec* je v tem primeru oseba ali naprava, ki bi hotela informacijo zlorabiti. Pod zlorabo informacije razumemo vsako dejanje, ki je z moralnega, pravnega ali tehničnega stališča nedopustno.

Model klasičnega sistema za prikrivanje podatkov je narisan na sliki 152. Postaja A mora po komunikacijskem kanalu poslati neko važno informacijo - *odprto besedilo* P postaji B. Besedilo P je take narave, da ga sme prebrati samo postaja B. V kanalu lahko njuni komunikaciji prisluškujejo tudi druge postaje. Kanal torej ne zagotavlja tajnosti komunikacije.

Predno postaja A pošlje besedilo P v kanal, ga 'zapre' (*prikrije ali šifrira*) s šifrirnim postopkom E in s šifrirnim ključem K_E. Šifrirni postopek in ključ drži v tajnosti.



Slika 152: Model klasičnega šifrirnega sistema.

S šifriranjem odprtega besedila P nastane *kriptogram* C, ki je odvisen od šifrirnega postopka E in šifrirnega ključa K_E,

$$C = E(P, K_E).$$

Kriptogram (skritopis) je sporočilo, ki ga zna praviloma 'razvozljati' oziroma *dešifrirati* samo tisti, ki pozna dešifrirni postopek D in dešifrirni ključ K_D,

$$P = D(C, K_D).$$

To je tisti, ki mu je sporočilo namenjeno. V našem primeru je to postaja B. Področje, ki se ukvarja z izdelavo kriptogramov, imenujemo *kriptografija* (šifriranje ali skritopisje). Da šifrirni sistem zagotavlja tajno komunikacijo med postajama A in B, morata biti tudi dešifrirni postopek in dešifrirni ključ tajna.

Oddajna postaja pošlje kriptogram C sprejemni postaji po informacijskem kanalu. V kanalu ga lahko opazuje ali prestreže *vsiljivec*. Možni sta dve obliki vsiljivca. *Pasivni* vsiljivec skuša iz kriptograma priti do odprtega besedila. *Aktivni* vsiljivec pa kriptograme tudi prestreže, zakasni, spreminja obstoječe kriptograme ali celo vnaša svoje (lažne) kriptograme. Vsiljivec v splošnem ne pozna niti šifrirnega niti dešifrirnega ključa. Tudi postopka šifriranja in dešifriranja se običajno držita v tajnosti. Kljub temu mora dober šifrirni sistem 'zdržati napad' tudi v primeru, da sta oba postopka znana vsiljivcu, zato predpostavljamo da sta postopka javna. V vsakem primeru morata biti tajna *oba* ključa, tako šifrirni kot dešifrirni ključ. To je glavna značilnost klasičnih kriptografskih sistemov. Razlog temu je samo eden. Namreč, iz šifrirnega ključa se da enostavno ugotoviti dešifrirni ključ, velja pa tudi obratno. Področje, ki se ukvarja z razbijanjem kriptografskih sistemov, imenujemo *kriptoanaliza*. *Kriptologija* (kriptoslovje) pa je širši pojm. Kriptologija je veda, ki obsega kriptografijo in kriptoanalizo.

8.2.2 Klasični šifrirni postopki

Šifrirne postopke delimo v naslednji dve veliki skupini:

- šifriranje z *nadomeščanjem* (zamenjavo) ali substitucijo in
- šifriranje s *premeščanjem* ali transpozicijo.

Možne so tudi poljubne kombinacije obeh postopkov.

Šifriranje z nadomeščanjem

Pri šifriranju z nadomeščanjem se vsaka črka (simbol) ali zaporedje črk zamenja z neko drugo črko ali zaporedjem črk. Torej, simbole s_i iz abecede S po nekem pravilu preslikamo v simbole r_i iz abecede R,

$$s_i \to r_i, \quad (i = 0, 1, 2, \ldots, N-1),$$

Najstarejši postopek šifriranja pripisujejo Juliju Cezarju. Po Cezarjevem postopku zamenjamo **a** z **D**, **b** zamenjamo **E**, **c** z **F**, i.t.d.. Imamo:

$S = \{$a,b,c,č,d,e,f,g,h,i,j,k,l,m,n,o,p,r,s,š,t,u,v,z,ž$\}$,

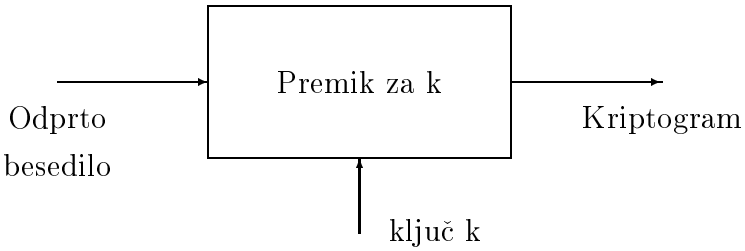
$R = \{$d,e,f,g,h,i,j,k,l,m,n,o,p,r,s,š,t,u,v,z,ž,a,b,c,č$\}$.

Na primer, kriptogram za odprto besedilo **sdv** je **UGA**.

Majhna posplošitev Cezarjevega postopka šifriranja dopušča premik abecede za poljubno število, denimo k črk. Simbole $s_i \in S$ preslikamo v simbole $r_i \in R$ tako, da je

$$r_i = s_{(i+k) \ (mod\ N)} \quad (i = 0, 1, 2, \ldots, N-1)$$

in je k ključ, ki parameterizirana šifrirni postopek.



Še splošnejše šifriranje te vrste preslika vsako črko odprtega besedila v neko drugo črko. Torej, za vsako črko posebej je določena nadomestna črka. 'Ključ' je zato kar cela abeceda. Možnih je (za slovenski jezik, če štejemo tudi presledek) $26! = 4 \times 10^{26}$ različnih ključev. To je dosti preveč, da bi kriptoanalitik mogel s slepim poskušanjem zlomiti sistem - odkriti ključ. Kot bomo videli, se da kljub tako številnim možnostim že iz majhne zaloge kriptogramov prav lahko zlomiti tudi ta sistem.

Tipičen napad na šifrirni postopek temelji na statistični naravi besedila. Na primer, samoglasniki se v splošnem pogosteje pojavljajo kot soglasniki. Če izvzamemo presledek, je najpogostejša črka v slovenščini **e**, najredkeje pa naletimo na črko **f**. Nekatere dvočrkovne, tročrkovne i.t.d. kombinacije so zelo pogoste, druge pa so čisto nemogoče. Verjetnosti črk in nekaterih najpogostejših dvočrkovnih kombinacij so zbrane v tabelah 10 in 11.

Kriptoanalitik, ki napada šifirni sistem, bi lahko začel z izračunom pogostosti pojavljanja posameznih črk v kriptogramu. Zatem bi najpogostejšo črko zamenjal z **e**, naslednjo manj pogosto z **a**, i.t.d.. Nato bi upošteval soodvisnost sosednjih črk. Recimo, črki **e** najverjetneje sledi črka **m**, pred njo pa je najpogosteje **j**. Podobno bi si lahko pomagal z najpogostejšimi besedami in frazami. Še lažje delo bi imel, če bi poznal (ali uganil) tematiko šifriranega besedila.

A	0.0770	B	0.0194	C	0.0055	Č	0.0123
D	0.0265	E	0.0937	F	0.0002	G	0.0126
H	0.0097	I	0.0762	J	0.0362	K	0.0292
L	0.0499	M	0.0316	N	0.0504	O	0.0705
P	0.0230	R	0.0365	S	0.0469	Š	0.0084
T	0.0315	U	0.0146	V	0.0270	Z	0.0172
Ž	0.0051	ločila	0.1890				

Tabela 10: Verjetnosti črk v slovenskem besedilu. V znaku *ločila* je zajet tudi presledek. Če izvzamemo ločila, je črka **e** najpogostejša. Tabela je rezultat obdelave okrog 120000 črkovnih znakov (Ivan Cankar: Moje življenje).

je	0.0182	se	0.0135	il	0.0116	ni	0.0102
na	0.0095	la	0.0091	al	0.0091	po	0.0091
ko	0.0088	in	0.0087	ne	0.0086	st	0.0086
li	0.0082	em	0.0081	re	0.0077	en	0.0077
bi	0.0077	ra	0.0076	da	0.0075	el	0.0074

Tabela 11: Najpogostejše kombinacije parov črk, če izvzamemo kombinacije črke z ločilom.

Da bi kriptoanalitiku onemogočili takšen pristop, je potrebno v kriptogramu zabrisati prvotno statistično vsebino besedila. Ena od možnosti je šifriranje ne le z eno abecedo (R), ampak z več abecedami $R_1, R_2, \ldots$. Primer takega šifrirnega sistema je Vigenerjev sistem. Sestavlja ga (za slovenščino) matrika velikosti 25×25. Vsaka vrstica začne z drugačno črko in vsebuje eno šifrirno abecedo. Torej imamo 25 šifrirnih abeced, ki so zapisane v preglednici 12. Ključ pri tem postopku je poljubno zaporedje črk in določa abecedo (vrstico), po kateri šifriramo dano črko v besedilu.

a	b	c	č	d	e	f	g	h	i	j	k	l	m	n	o	p	r	s	š	t	u	v	z	ž
b	c	č	d	e	f	g	h	i	j	k	l	m	n	o	p	r	s	š	t	u	v	z	ž	a
c	č	d	e	f	g	h	i	j	k	l	m	n	o	p	r	s	š	t	u	v	z	ž	a	b
č	d	e	f	g	h	i	j	k	l	m	n	o	p	r	s	š	t	u	v	z	ž	a	b	c
d	e	f	g	h	i	j	k	l	m	n	o	p	r	s	š	t	u	v	z	ž	a	b	c	č
.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
s	š	t	u	v	z	ž	a	b	c	č	d	e	f	g	h	i	j	k	l	m	n	o	p	r
.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
v	z	ž	a	b	c	č	d	e	f	g	h	i	j	k	l	m	n	o	p	r	s	š	t	u
z	ž	a	b	c	č	d	e	f	g	h	i	j	k	l	m	n	o	p	r	s	š	t	u	v
ž	a	b	c	č	d	e	f	g	h	i	j	k	l	m	n	o	p	r	s	š	t	u	v	z

Tabela 12: Šifrirne abecede Vigenerjevega sistema za slovensko besedilo

Izberimo si ključ **sdv** in poskusimo zašifrirati besedilo

danesnebopredavanj.

Zapišimo ključ nad odprto besedilo:

s	**d**	**v**	**s**	**d**	**v**	**s**	**d**	**v**	**s**	**d**	**v**	**s**	**d**	**v**	**s**	**d**	**v**
d	**a**	**n**	**e**	**s**	**n**	**e**	**b**	**o**	**p**	**r**	**e**	**d**	**a**	**v**	**a**	**n**	**j**

Poglejmo, kako zašifriramo prvi dve črki besedila. Črka **s** ključa nad prvo črko besedila (**d**) določa, da pri šifriranju črke **d** izberemo abecedo, ki začne s črko **s**. Torej jo nadomestimo s črko **v**. Naslednjo črko **a** šifriramo z abecedo, ki začne z **d**. Zato **a** nadomestimo z **d**. Po tem pravilu nadomeščanja črk nastane iz zgornjega odprtega besedila naslednji kriptogram:

vdkzvkzellucvdšssg.

Ker eno in isto črko nadomestimo po več različnih pravilih (abecedah), odvisno od položaja črke v besedilu, se iz porazdelitve črk v kriptogramu ne da sklepati na verjetnostno porazdelitev črk v odprtem besedilu. Na primer, črko **e** smo dvakrat nadomestili z **z** in enkrat s **c**. Napad na šifrirni sistem bi tu zahteval najprej določitev (s poskušanjem) dolžine ključa. Predpostavimo, da je dolžina ključa k. Kriptogram bi uredili v vrstice, k črk na vrstico. Ko uganemo dolžina ključa, so črke ene vrstice šifrirane z isto abecedo in porazdelitveni zakon črk se v

vsaki vrstici ujema s porazdelitvenim zakonom odprtega besedila. Ker poznamo porazdelitveni zakon črk odprtega besedila, poskušamo z različnimi dolžinami ključa tako dolgo, dokler se izračunane verjetnosti črk ne ujemajo z napovedanimi (Za slovenski jezik vemo, da je na primer verjetnost črke **e** okrog 0.094).

Šifriranje s premeščanjem

Pri šifriranju z nadomeščanjem črke v odprtem besedilu zamenjamo (nadomestimo) z drugimi črkami, vrstni red črk pa se ohrani. Pri šifriranju (samo) s premeščanjem črk v odprtem besedilu ne zamenjamo z drugimi črkami, ampak jim samo premestimo, zamenjamo jim vrstni red. Tudi pri šifrirnanju s premeščanjem je kriptogram odvisen še od ključa. V naslednjem šifrirnem postopku sme biti ključ poljubo zaporedje črk, le da se vsaka črka v ključu pojavi samo enkrat. Naj bo ključ **vaje** in odprto besedilo

danesbodopredavanja.

Zapišimo odprto besedilo v več vrstic, štiri črke v vsako vrstico (štiri je dolžina ključa).

v	a	j	e	a	e	j	v
4	1	3	2	1	2	3	4
d	a	n	e	a	e	n	d
s	b	o	d	b	d	o	s
o	p	r	e	p	e	r	o
d	a	v	a	a	a	v	d
n	j	a	x	j	x	a	n

Ključ uporabimo za številčenje stolpcev. Ker je črka **a** v abecedi pred črkami **e**, **j** in **v**, ima drugi stolpec številko ena. Naslednja v abecedi je črka **e**, zato ima četrti stolpec številko dva. Tretji stolpec ima številko tri in prvi stolpec oštevilčimo s štiri. Stolpce preuredimo tako, kot zahtevajo številke stolpcev in zapišemo kriptogram, ki je za naš primer:

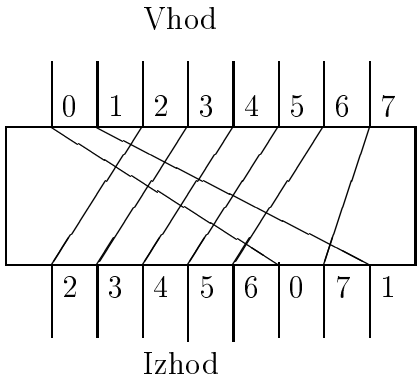
aendbdosperoaavdjxan.

Če hoče kriptoanalitik priti do odprtega besedila, mora najti dolžino ključa, odkriti ključ in preurediti stolpce. Ker so črke samo premeščene, je verjetnostna porazdelitev črk v kriptogramu taka kot v odprtem besedilu, kar kriptoanalitiku še olajša delo.

8.2.3 Šifrirni sistem DES

Čeprav novejša kriptografija uporablja iste prijeme kot tradicionalna kriptografija, namreč nadomeščanje in premeščanje črk, se v nečem bistveno razlikujeta. V preteklosti so kriptografi uporabljali razmeroma enostavne postopke šifriranja in se zanašali na dolge ključe. Sedaj je ravno obratno: pudarek je na sila zapletenem šifrirnem postopku in razmeroma kratkem ključu. Razlog je jasen. Danes obstajajo učinkovite naprave za izvedbo postopka šifriranja.

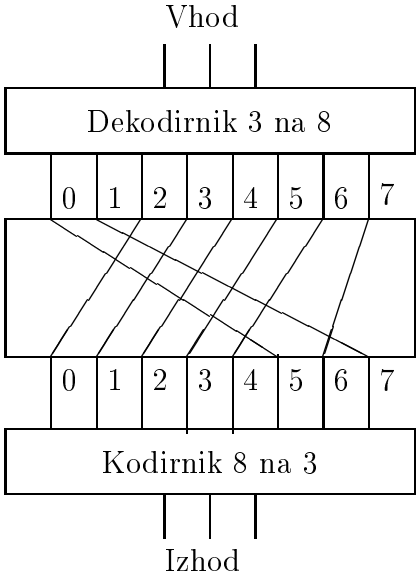
Osnovni operaciji šifriranja, substitucijo (zamenjavo) in transpozicijo (premeščanje), se da izvesti z razmeroma enostavnimi vezji. Slika 153 shematično ponazarja primer P-bloka, ki izvede transpozicijo (P pomeni permutacijo) osmih binarnih simbolov.



Slika 153: P-blok za izvedbo transpozicije (premeščanja) osmih bitov.

P-blok na sliki ima osem vhodov in osem izhodov ter izvede transpozicijo na osem-bitnem vhodu. Če oštevilčimo vhode po vrsti od leve na desno s števili od 0 do 7, dobimo na izhodu zaporedje 23456071. Z ustrezno povezavo vhodov z izhodi se da doseči poljubno transpozijo.

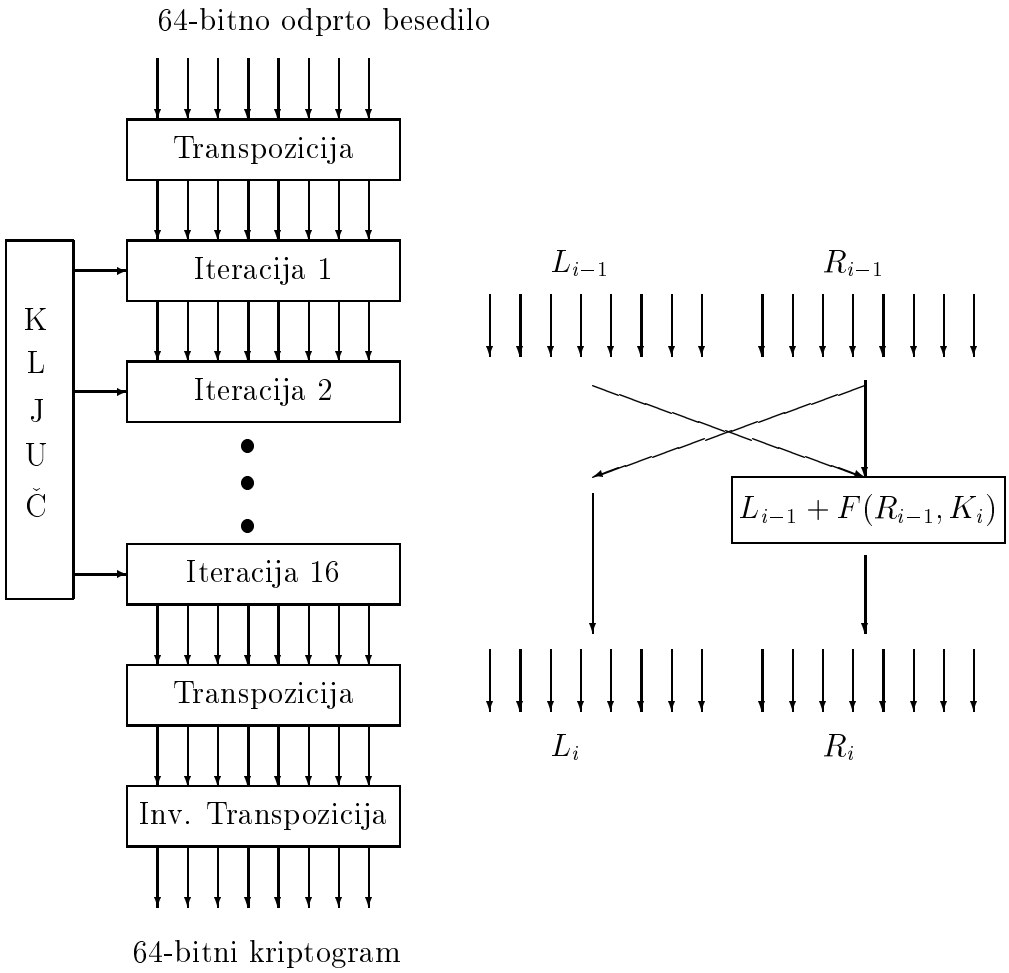
Substitucijo (nadomeščanje) izvede S-blok. Primer S-bloka je narisan na sliki 154. V narisanem primeru se tri-bitna informacija na vhodu zamenja s tri-bitno informacijo na izhodu. Na primer, če bi imeli na vhodu osmiški simbol 4, bi se ta na izhodu nadomestil s simbolom 2. Oziroma, za vhodno zaporedje (osmiških) simbolov 01234567, bi imeli na izhodu zaporedje 57012346. S kombinacijo P-blokov in S-blokov se da sestaviti poljubno bolj zapletene šifrirne strukture.



Slika 154: S-blok za izvedbo substitucije (nadomeščanja) tri-bitne informacije (osmiških cifer).

Leta 1977 je ameriška vlada privzela šifrirni sistem, ki so ga tedaj razvili pri IBM-u. Standardizacija sistema s strani NBS (National Bureau of Standards) je spodbudila številne proizvajalce, da so začeli izdelovati materialno opremo za izvedbo tega postopka. Hitra in cenena materialna oprema pa je povzročila, da so uporabniki množično sprejeli ta šifrirni sistem, ki je danes znan pod kratico DES (Data Enctyption Standard). Šifrirni algoritem (v grobem) shematično ponazarja slika 155(levo). Algoritem sestavlja 19 korakov in je parameteriziran s 64-bitnim ključem. Šifrirni ključ je enak dešifrirnemu ključu, postopek dešifriranja pa je ravno obraten postopku šifriranja.

Odprto besedilo se pred šifriranjem deli na 64 bitne bloke in nato šifrira. Prvi korak je transpozicija bitov, ki je neodvisna od ključa. Zadnji korak je obratna transpozicija k tej začetni transpoziciji. Predzadnji korak preprosto zamenja zgornjih in spodnjih 32 bitov. Ostalih 16 korakov (iteracij) je parameteriziranih s ključem. Operacija ene od teh iteracij je prikazana na sliki 155(desno). Vhod v vsako tako stopnjo sta dve skupini po 32 bitov. Tudi izhod sta dve skupini po 32 bitov. Desni vhod je preprosto prepisan na levi izhod. Desni izhod je odvisen od levega vhoda, desnega vhoda, od ključa K_i in od funkcije F, v kateri je skrita vsa kompleksnost postopka. Funkcijo F sestavljajo štirje koraki, ki so lahko izvedeni samo s P-bloki in s S-bloki. V vsaki od teh 16 iteracij je uporabljen drugačen (transformiran) 56 bitni ključ K_i.



Slika 155: Šifrirni algoritem DES (levo) in podrobnejši prikaz ena izmed šestnajstih iteracij (desno).

8.2.4 Šifriranjem z javnim ključem

Vsi do sedaj obravnavani postopki temeljijo na predpostavki, da sta in tudi morata biti šifrirni in dešifrirni ključ tajna. Če se spomnimo obravnavanih postopkov šifriranja z nadomeščanjem in s premeščanjem, se nam to zdi tudi popolnoma razumljivo. Namreč, če poznamo šifrirni ključ, se da z lahkoto najti tudi dešifrirni ključ, če že nista kar enaka. Leta 1976 sta Diffie in Hellman bistveno spremenila miselnost, ki je vladala vse do tedaj pri izdelavi sedaj že klasičnih šifrirnih postopkov.

Diffie in Hellman sta prva predlagala uporabo takih šifrirnih postopkov/ključev ter ustreznih dešifrirnih postopkov/ključev, za katere velja, da se dešifrirnega postopka/ključa ne da določiti (vsaj ne enostavno ali v sprejemljivem

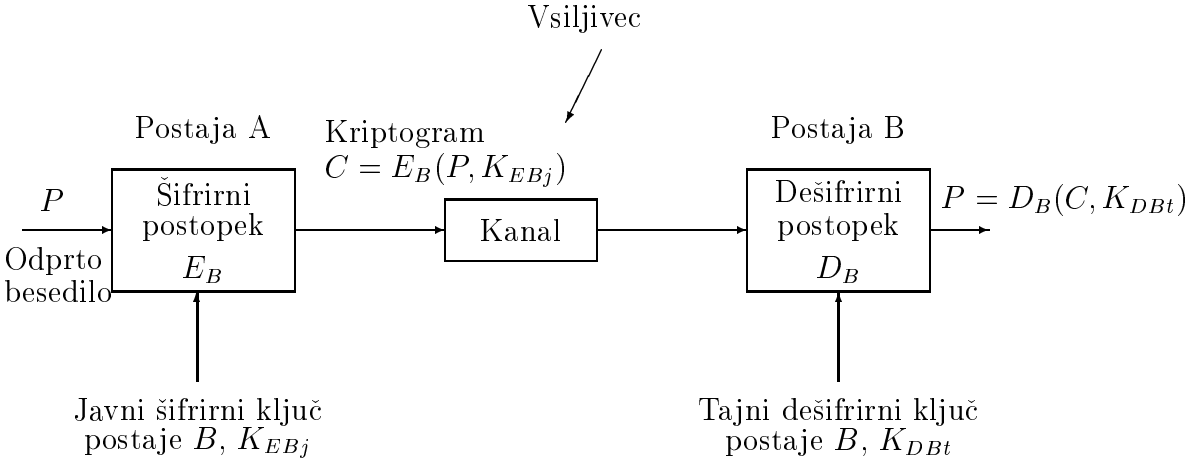
času) niti, če je šifrirni postopek/ključ popolnoma znan. Če je izpolnjen ta pogoj, potem ni nobenega razloga več, da bi šifrirni postopek/ključ ne smel biti javen - znan vsem. Od tu tudi ime šifriranje z javnim ključem. S tem, ko postane šifrirni ključ javen, problema tajne distribucije ključev ni več (problem: kaj je bilo prej - kokoš ali jajce). Ta problem je prisoten v sistemih s tajnimi ključi. Namreč, kako poslati tajno informacijo (v tem primeru ključ) tistim, s katerim hočemo tajno komunicirati.

Slika 156 prikazuje model sistema šifriranja z javnim ključem. Vsaka postaja, ki želi tajno komunicirati z ostalimi postajami, objavi svoj šifrirni ključ, dešifrirni ključ pa drži v tajnosti. Vzemimo, da hoče postaja A poslati postaji B odprto besedilo P. Zato odprto besedilo zašifrira s šifrirnim postopkom E_B in javnim šifrirnim ključem K_{EBj} postaje B. Postaja A odpošlje kriptogram C,

$$C = E_B(P, K_{EBj}),$$

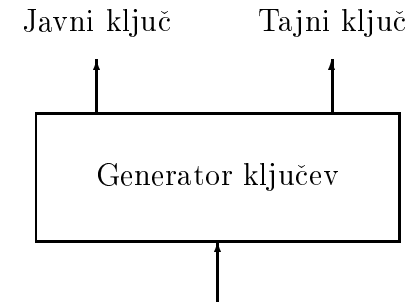
ki ga je možno dešifrirati samo z dešifrirnim postopkom D_B in dešifrirnim ključem K_{DBt}, ki ju postaja B drži v tajnosti,

$$P = D_B(C, K_{DKt}) = D_B(E_B(P, K_{EBj}), K_{DBt}).$$



Slika 156: Model šifrirnega sistema z javnim ključem.

Model šifrirnega sistema običajno dopolnimo še s podsistemom za generiranje ključev, ki je v narisanem modelu last postaje B.



Odprto je še vprašanje, kako določiti šifrirni in dešifrirni postopek oziroma ključ. Rivest, Shamir in Adleman so leta 1978 objavili naslednji postopek za izvedbo šifrirnega sistema (RSA algoritem):

- izberi dve veliki praštevili p in q, vsako večje od 10^{100},
- izračunaj $n = p \times q$ in $z = (p-1) \times (q-1)$,
- določi število d tako, da nima skupnega deljitelja s številom z,
- poišči tako število e, da velja $e \times d = 1 \pmod{z}$.

Tako so določeni parametri e, n in d, ki parameterizirajo šifrirni sistem. Za šifriranje sta potrebna e in n. Par (e, n) predstavlja šifrirni ključ. Ta je javen in postaja ga 'objavi', recimo vpiše v skupen seznam javnih ključev. Ze dešifriranje rabimo par števil (d, n), ki predstavlja dešifrirni ključ. Število d postaja zadrži v tajnosti. Postopek RSA temelji na dejstvu, da je velika števila težko faktorizirati, iz n je težko najti p in q, ki posredno določata e in d.

Odprto besedilo, ki ga neka druga postaja pošilja tej postaji, pred šifriranjem deli na manjše enote ali bloke P. Blok P mora biti krajši od n. Oddajna postaja zašifrira blok P z naslednjim šifrirnim pravilom:

$$C = P^e \pmod{n}.$$

Izračunani kriptogram C naša postaja sprejme in dešifrira, izračuna

$$P = C^d \pmod{n}$$

Rezultat je prvotno besedilo P. Pravili šifriranja in dešifriranja sta javni.

Poglejmo preprost primer določitve ključev.

- Naj bosta zaradi nazornosti p in q majhna, denimo $p = 3$ in $q = 11$.

- Izračunamo $n = 3 \times 11 = 33$ in $z = 2 \times 10 = 20$.
- Za d vzemimo 7, ker z 20 nima skupnega delitelja.
- Izberemo e tako, da bo $e \times 7 = 1 \pmod{20}$. Sledi, da je $e = 3$.

Šifrirni ključ, ki je javen, je $K_{Ej} = (3, 33)$, tajni dešifrirni ključ pa je $K_{Dt} = (7, 33)$. Kriptogram za besedilo P je $C = P^3 \pmod{33}$ in ga dešifriramo po pravilu $P = C^7 \pmod{33}$. Ker sta p in q majhna, je tudi n majhen in besedilo P mora biti krajše od 33. Ker ima abeceda 25 črk (recimo številčena od 1 do 25), moramo šifrirati vsako črko posebej (posamezno).

Šifrirajmo odprto besedilo **SDV**. Zaporedne številke črk so 19, 5 in 23. Podrobnosti šifriranja prikazuje naslednja preglednica:

Črka	Številka	P^3	$P^3 (mod\ 33)$
S	19	6859	28
D	5	125	26
V	23	12167	23

Pošljemo zaporedje 28, 26, 23. Podrobnosti dešifriranja prikazuje naslednja preglednica:

Številka	P^7	$P^7 (mod\ 33)$	Črka
28	13492928512	19	S
26	8031810176	5	D
23	3404825447	23	V

8.2.5 Overovitev pošiljatelja in digitalni podpis

Eden od glavnih problemov šifrirnih sistemov z javnim ključem je možnost ponarejanja sporočil. Ker je šifrirni ključ javen, lahko vsaka postaja pošlje sporočilo dotični postaji, vendar pa sprejemna postaja iz sporočila ne more ugotoviti, kdo ji je v resnici poslal sporočilo. To pomeni,

- da se lahko neka postaja predstavi kot kakšna druga postaja, ali
- da neka postaja utaji, da je kdajkoli poslala sporočilo.

Obe možnosti sta nedopustni. Prvi problem se da rešiti z overovitvijo sporočila. Denimo, da komunicirata postaji A in B. Postaja B pričakuje od postaje A sporočilo. Da ne bi sporočilo ponaredila kakšna druga postaja, zahteva od postaje

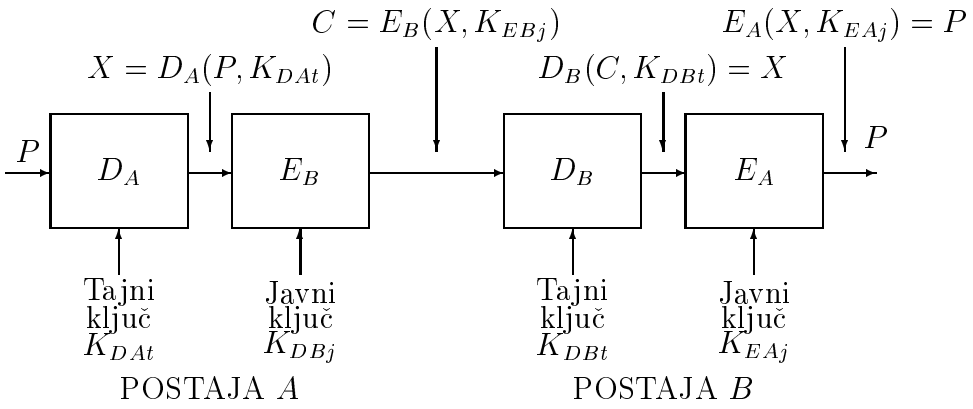
A, da se najprej predstavi. Ker pozna njen javni šifrirni ključ, ji pošlje naključno generirano sporočilo in zahteva, da ji ga vrne dešifriranega. Ker ima ustrezni dešifrirni ključ samo postaja A, je ta edina, ki zna sporočilo dešifrirati. Tako dobi postaja B potrdilo, da komunicira s postajo A. Poleg tega pa mora postaja A v vsako kasnejše sporočilo dodati še geslo ali kaj podobnega, kar drugim postajam preprečuje ponarejanje.

Overovitev pošiljatelja še ne prepreči možnoti, da pošiljatelj odposlano sporočilo kasneje utaji. To preprečuje *digitalni podpis*. Model šifrirnega sistema z digitalnim podpisom in z javnim ključem prikazuje slika 157.

Postaja A, ki pošilja odprto besedilo P postaji B, najprej zašifrira besedilo P s svojim postopkom D_A in s tajnim ključem K_{DAt},

$$X = D_A(P, K_{ADt}).$$

Postopek in ključ (pravzaprav samo ključ) pozna le postaja A, kar je zadosten dokaz, da besedilo pošilja ta postaja. Tako 'podpiše' sporočilo.



Slika 157: Model šifrirnega sistema z digitalnim podpisom in z javnim ključem.

Postaja A zatem X šifrira s postopkom E_B in z javnim ključem K_{EBj} tiste postaje, ki ji je namenjeno sporočilo. To postajo smo označili z B. V kanal pošlje kriptogram C,

$$C = E_B(X, K_{EBt}) = E_B(D_A(P, K_{DAt}), K_{EBt}).$$

Na sprejemni strani teče obraten postopek. Postaja B najprej dešifrira kriptogram s svojim postopkom D_B in s tajnim ključem K_{DBt}, da dobi

$$X = D_B(C, K_{DBt}) = D_B(E_B(D_A(P, K_{DAt}), K_{EBt}) = D_B(E_B(X, K_{EBt}),$$

zatem pa še s postopkom E_A in z javnim ključem K_{EAj} postaje A. Rezultat je odprto besedilo P,

$$P = E_A(X, K_{EAj}) = E_A(D_B(E_B(D_A(P, K_{DAt}), K_{EBt}).$$

Za šifrirne postopke in ključe mora očitno veljati

$$D(E(Y, K_E)) = Y$$

tako kot prej pri šifriranju z javnim ključem, da je možna izvedba digitalnega podpisa pa mora dodatno veljati tudi:

$$E(D(Y, K_D)) = Y.$$

Torej, če opustimo pisanje ključev, velja $D(E(Y)) = Y$ in $E(D(Y)) = Y$ za vsak šifrirni par.

A Verjetnostni račun

Osnovna pojma verjetnostnega računa sta *poskus* in *dogodek*. Dogodek je realizacija poskusa ali eksperimenta. Poskus običajno označimo z malo, dogodek pa z veliko črko z začetka abecede. Dogodek je lahko gotov (G), naključen ali nemogoč (N). Dogodek je sestavljen iz elementarnih dogodkov. Elementaren dogodek imenujemo izid.

Z dogodki računamo. Vsota dogodkov $A + B$ je dogodek, ko se zgodi eden ali drugi ali oba. Produkt dogodkov AB je dogodek, ko se zgodita oba. Če je $AB = N$, sta dogodka A in B *nezdružljiva*. Dogodka A in B sta *nasprotna*, če se realizira natanko eden od obeh. Tedaj je $A + B = G$ in $AB = N$ oziroma B je negacija A-ja, $B = \bar{A}$.

Paroma nezdružljivi dogodki $A_i, \quad (i = 1, 2, \ldots, n)$ sestavljajo *popoln sistem* dogodkov, če se lahko v poskusu realizira natanko eden od njih:

$$A_i \neq N, \;\; A_i.A_j = N \quad (i, j = 1, 2, \ldots, n) \;\text{ in }\; (i \neq j), \;\; \sum_{i=1}^{n} A_i = G.$$

Verjetnost dogodkov Po matematični definiciji je verjetnost dogodka A enaka

$$P(A) = \frac{m}{n} = \frac{\text{število ugodnih izidov}}{\text{število vseh možnih izidov}}$$

Statistična definicija verjetnosti dogodka A temelji na ponavljanju poskusa pod enakimi pogoji, neodvisno enega od drugega,

$$P(A) \approx f(A) = \frac{m}{n} = \frac{\text{število realizacij A-ja}}{\text{število ponovitev poskusa}}$$

Za nezdružljiva dogodka ($AB = N$) z verjetnostima $P(A)$ in $P(B)$ je verjetnost vsote enaka vsoti verjetnosti,

$$P(A + B) = P(A) + P(B).$$

Verjetnost produkta nezdružljivih dogodkov je enaka nič,

$$P(AB) = 0.$$

Denimo, da sta dogodka A in B iz poskusa **a** združljiva ($AB \neq N$). Naj ima poskus n izidov, od teh n_A ugodnih za A in n_B ugodnih za B ter n_{AB} izidov ugodnih za A in za B. Verjetnost dogodka A, verjetnost dogodka B ter verjetnost njunega produkta so zaporedoma:

$$P(A) = \frac{m_A}{n}, \quad P(B) = \frac{m_B}{n}, \quad P(AB) = \frac{m_{AB}}{n}.$$

Verjetnost vsote združljivih dogodkov pa je

$$P(A+B) = \frac{m_A + m_B - m_{AB}}{n} = P(A) + P(B) - P(AB)$$

Verjetnosti produkta dogodkov $P(AB)$ pravimo tudi *vezana verjetnost* dogodkov A in B. *Pogojna verjetnost* $P(A/B)$ dogodka A glede na dogodek B je verjetnost dogodka A pri pogoju, da se je (ali bi se) zgodil dogodek B. Verjetnosti dogodka $P(A)$ pravimo (v primeru dvoumnosti) tudi *lastna verjetnost* dogodka A.

Dogodka A in B sta *neodvisna*, če se z realizacijo enega ne spremeni verjetnost za realizacijo drugega. Velja:

$$P(A/B) = P(A) \quad \text{ali} \quad P(B/A) = P(B).$$

Naj bo v poskusu **a** n možnih izidov, od teh m_A ugodnih za A in m_B ugodnih za B ter m_{AB} ugodnih za oba. Potem je:

$$P(A) = \frac{m_A}{n}, \quad P(B) = \frac{m_B}{n}, \quad P(AB) = \frac{m_{AB}}{n}.$$

Pogojna verjetnost dogodka B glede na dogodek A je:

$$P(B/A) = \frac{\text{št. realizacij } B\text{-ja skupaj z } A}{\text{št. realizacij } A} = \frac{m_{AB}}{m_A} = \frac{m_{AB}/n}{m_A/n} = \frac{P(AB)}{P(A)}$$

Torej je:

$$P(AB) = P(A)P(B/A) = P(B)P(A/B)$$

Za neodvisna dogodka je $P(B/A) = P(B), P(AB) = P(A)P(B)$. Za nezdružljiva dogodka je $P(B/A) = 0$. Koliko pa je verjetnost vsote neodvisnih dogodkov?

Popolna verejetnost Naj dogodki $H_i, \quad (i = 1, 2, \ldots, n)$ sestavljajo popoln sistem. Zato je $\sum_i H_i = G$ in $H_i H_j = N \ \ (i \neq j)$. Naj bo A dogodek, ki se lahko zgodi samo hkrati z enim toda katerimkoli od dogodkov H_i. Poznamo $P(H_i)$ in $P(A/H_i), \quad (i = 1, 2, \ldots, n)$ in iščemo popolno (totalno) verjetnost dogodka A, $P(A)$.

$$A = AG = A\sum_i H_i = AH_1 + AH_2 + \cdots + AH_n$$

Dogodki AH_i so paroma nezdružljivi. Zato je

$$P(A) = P(AH_1) + P(AH_2) + \cdots + P(AH_n).$$

Nadalje,

$$P(A) = P(H_1)P(A/H_1 + P(H_2)P(A/H_2) + \cdots + P(H_n)P(A/H_n)$$

in formula za popolno verjetnost je:

$$P(A) = \sum_{i=1}^{n} P(H_i)P(A/H_i).$$

Bayesova formula Bayesova formula je kombinacija formule za popolno verjetnost,

$$P(A) = \sum_{1=1}^{n} P(H_i)P(A/H_i)$$

in formule za vezano verjetnost,

$$P(AH_i) = P(H_i)P(A/H_i) = P(A)P(H_i/A).$$

Po njej izračunamo, kakšna je verjetnost, da se je dogodek A zgodil skupaj prav z dogodkom H_i,

$$P(H_i/A) = \frac{P(H_i)P(A/H_i)}{\sum_i P(H_i)P(A/H_i)}.$$

Primer 1 Vsota in produkt dogodkov

Poskus a:	met kocke
Dogodek A:	pade sodo število pik, $A = \{2, 4, 6\}$
Dogodek B:	pade število pik deljivo s 3, $B = \{3, 6\}$

Vsota dogodkov je dogodek $A + B = \{2, 3, 4, 6\}$. Produkt dogodkov je dogodek $AB = \{6\}$.

Primer 2 Verjetnost dogodka - izbiranje kart

Izračunajmo verjetnost, da iz svežnja 32 igralnih kart izvlečemo dva asa. Predpostavljamo, da so v svežnju štirje asi. Verjetnost dogodka A (izvlečemo dva asa) je:

$$P(A) = \frac{\text{št. ugodnih izidov}}{\text{št. vseh možnih izidov}} = \frac{\binom{4}{2}}{\binom{32}{2}} = \frac{3}{248}.$$

Primer 3 Verjetnost dogodka - izbiranje kroglice

V posodi so tri bele in štiri črne kroglice. Izračunajmo verjetnost dogodka A, da izberemo belo kroglico:

$$P(A) = \frac{\text{št. ugodnih izidov}}{\text{št. vseh možnih izidov}} = \frac{3}{7}.$$

Primer 4 Verjetnost dogodka - izbiranje kroglic

V posodi so štiri bele in štiri črne kroglice. Opravimo poskus **a**. Sezimo v posodo in izvlečimo dve kroglici. Ko izvlečemo dve beli kroglici, se zgodi dogodek A.

Verjetnost, da se to zgodi, je

$$P(A) = \frac{\text{št. ugodnih izidov}}{\text{št. vseh možnih izidov}} = \frac{\binom{4}{2}}{\binom{8}{2}} = \frac{3}{14}.$$

Primer 5 Verjetnost dogodka - zaporedno izbiranje kroglic

V posodi so štiri bele in štiri črne kroglice. Tokrat opravimo poskus **b**. Dvakrat (drug za drugim) sezimo v posodo in vsakič izvlecimo po eno kroglico. Kroglic ne vrnačajmo v posodo. Označimo z A dogodek, ko izvlečemo dve beli kroglici. Verjetnost dogodka A je

$$P(A) = P(\text{prvič izvlečemo belo kroglico}) \times P(\text{drugič izvlečemo belo kroglico}) = \frac{4}{8} \times \frac{3}{7} = \frac{3}{14},$$

kar je toliko, kolikor v prejšnjem primeru.

Primer 6 Lastna, pogojna in vezana verjetnost - štetje črk

V slovenskem besedilu, ki obsega 40000, smo našteli 1492 črk V, 3309 črk E, 291 parov črk VE in 3 zaporedja črk VEM. Izračunajmo sledeče relativne frekvence (ocene verjetnosti): lastne verjetnosti črk V in E, verjetnost para črk VE, verjetnost, da črki V sledi črka E, in verjetnost, da paru črk VE sledi črka M.

Imamo:

$$P(V) = \tfrac{1492}{40000} = 0.037,\ P(E) = \tfrac{3309}{40000} = 0.083,\ P(VE) = \tfrac{291}{40000} = 0.007,$$

$$P(E/V) = \tfrac{291}{1492} = 0.196,\ P(M/VE) = \tfrac{3}{291} = 0.001.$$

Primer 7 Računanje popolne verjetnosti

Dve tovarni serijsko izdelujeta isti izdelek. Prva tovarna izdela 10 % prvovrstnih izdelkov. Druga tovarna izdela 15 % prvovrstnih izdelkov. Trgovina ima na zalogi 200 izdelkov iz prve tovarne in 100 izdelkov iz druge tovarne. Kupec pride v trgovino in na slepo izbere enega od izdelkov. Kolikšna je verjetnost, da bo kupil izdelek, ki je prvovrsten? Verjetnost, da je izdelek v trgovini iz prve tovarne, je $P(H_1) = \frac{200}{300} = \frac{2}{3}$

Verjetnost, da je izdelek v trgovini iz druge tovarne, je $P(H_2) = \frac{100}{300} = \frac{1}{3}$

Verjetnost, da je izdelek prvovrsten, če je iz prve tovarne, je $P(A/H_1) = 0.10$

Verjetnost, da je izdelek prvovrsten, če je iz druge tovarne, je $P(A/H_2) = 0.15$

Verjetnost, da je izdelek prvovrsten (ne glede na to, iz katere tovarne je), je

$$P(A) = \frac{2}{3} \times 0.10 + \frac{1}{3} \times 0.15 = \frac{7}{60}.$$

Lahko bi računali še drugače. Trgovina ima na zalogi 300 izdelkov, od teh je 35 prvovrstnih. Torej je verjetnost, da kupec kupi prvovrsten izdelek $\frac{35}{300} = \frac{7}{60}$.

Primer 8 Računanje pogojne verjetnosti

Vprašajmo se, kakšna je verjetnost, da je kupec iz prejšnje naloge kupil prvovrsten izdelek, ki ga je izdelala prva tovarna,

$P(H_1/A) = P$(prvovrsten izdelek je iz prve tovarne / kupec kupi prvovrsten izdelek).

$$P(H_1/A) = \frac{\frac{2}{3} \times 0.10}{\frac{2}{3} \times 0.10 + \frac{1}{3} \times 0.15} = \frac{4}{5}.$$

B Naključne spremenljivke

Naključna spremenljivka je količina, katere vrednost je odvisna od naključja. Določata jo *zaloga vrednosti* in *porazdelitveni zakon*. Glede na zalogo vrednosti ločimo *zvezne* naključne spremenljivke in *diskretne* naključne spremenljivke, ki nas tokrat bolj zanimajo. Naključne spremenljivke bomo označevali z velikimi črkami s konca abecede. Običajno zapišemo zalogo vrednosti diskretne naključne spremenljivke $X = \{x_1, x_2, \ldots, x_i, \ldots x_n\}$ in njen porazdelitveni zakon $P(X) = \{p_1, p_2, \ldots, p_i, \ldots, p_n\}$ skupaj v *verjetnostni shemi*:

$$X = \begin{pmatrix} x_1, & x_2, & \ldots, & x_i, & \ldots, & x_n \\ p_1, & p_2, & \ldots, & p_i, & \ldots, & p_n \end{pmatrix}.$$

Dejstvo, da naključna spremenljivka zavzame prav določeno vrednost x_i, $(i = 1, 2, \ldots, n)$ iz svoje zaloge vrednosti, je dogodek $(X = x_i)$ z verjetnostjo $p_i = P(X = x_i)$. Naključna spremenljivka zmore zavzeti eno samo vrednost; eno pa gotovo zavzame. Dogodki $(X = x_i)$ so torej paroma nezdružljivi in tvorijo popoln sistem. Zato je $\sum_i p_i = 1$.

Naključni spremenljivki X in Y sta *neodvisni*, če se z realizacijo ene ne spremeni porazdelitveni zakon druge. Če naključni spremenljivki nista neodvisni, sta *odvisni*.

Pogojna porazdelitev naključne spremenljivke je porazdelitev naključne spremenljivke glede na nek dogodek. Če sta naključni spremenljivki X in Y odvisni, je porazdelitev ene spremenljivke odvisna od realizacije druge. Recimo,

$$P(Y/X = x_i)$$

je pogojna porazdelitev naključne spremenljivke Y v primeru, da naključna spremenljivka X zavzame vrednost x_i. Torej, če se zgodi dogodek $(X = x_i)$.

Primer 9 Signal kot naključna spremenljivka

Signalu, ki zmore v nekem trenutku zavzeti eno od osmih enakoverjetnih nivojev (stanj) $0, 1, 2, 3, 4, 5, 6, 7$ priredimo naključno spremenljivko X,

$$X = \begin{pmatrix} 0, & 1, & 2, & 3, & 4, & 5, & 6, & 7 \\ \frac{1}{8}, & \frac{1}{8}, & \frac{1}{8}, & \frac{1}{8}, & \frac{1}{8}, & \frac{1}{8}, & \frac{1}{8}, & \frac{1}{8} \end{pmatrix}$$

Primer 10 Neodvisni naključni spremenljivki

Mečemo kocko in nato kovanec. Privzemimo, da je kocka poštena. Z enako verjetnostjo pade 1,2,3,4,5 ali 6 pik. Metu kocke priredimo naključno spremenljivko X,

$$X = \begin{pmatrix} 1, & 2, & 3, & 4, & 5, & 6 \\ \frac{1}{6}, & \frac{1}{6}, & \frac{1}{6}, & \frac{1}{6}, & \frac{1}{6}, & \frac{1}{6} \end{pmatrix}.$$

Naj bo tudi kovanec pošten - z enako verjetnostjo pade grb ali cifra. Metu priredimo naključno spremenljivko Y,

$$Y = \begin{pmatrix} grb, & cifra \\ \frac{1}{2}, & \frac{1}{2} \end{pmatrix}$$

Met kocke prav nič ne vpliva na izid meta kovanca. Verjetnost, da pade grb oziroma cifra ni odvisna od izida meta kocke. Povejmo še drugače. Met kovanca nam o izidu meta kocke ne pove ničesar. Velja tudi obratno - iz meta kocke ne moremo sklepati na izid meta kovanca. Naključni spremenljivki, ki ju priredimo obema poskusoma, sta neodvisni: iz realizacije ene ne moremo sklepati na realizacijo druge.

Primer 11 Odvisni naključni spremenljivki

V prvi posodi so tri črne in tri bele kroglice. V drugi posodi so štiri črne in tri bele kroglice. Sežemo v prvo posodo, izvlečemo kroglico in jo predanemo v drugo posodo. Potem iz druge posode izvlečemo kroglico. Izbiranju kroglice iz prve posode priredimo naključno spremenljivko, ki zavzame vrednost x_1, če izberemo belo kroglico in vrednost x_2, če izberemo črno kroglico,

$$X = \begin{pmatrix} x_1, & x_2 \\ \frac{1}{2}, & \frac{1}{2} \end{pmatrix}.$$

Izbiranju kroglice iz druge posode priredimo naključno spremenljivko Y, ki zavzame vrednost y_1, ko izvlečemo belo kroglico, sicer pa y_2,

$$Y = \begin{pmatrix} y_1, & y_2 \\ p(y_1), & p(y_2) \end{pmatrix}.$$

Spremenljivki X in Y sta očitno odvisni. Poglejmo, kakšna je pogojna porazdelitev spremenljivke Y, če spremenljivka X zavzame vrednost x_1 (izvlečemo belo kroglico),

$$P(Y/X = x_1) = \{P(Y = y_1/X = x_1), P(Y = y_2/X = x_1)\} = \{\frac{1}{2}, \frac{1}{2}\}.$$

Če iz prve posode izvlečemo črno kroglico ($X = x_2$), je pogojna porazdelitev naključne spremenljivke Y drugačna,

$$P(Y/X = x_2) = \{P(Y = y_1/X = x_2), P(Y = y_2/X = x_2)\} = \{\frac{3}{8}, \frac{5}{8}\}.$$

Porazdelitev $P(Y) = \{p(y_1), p(y_2)\}$ naključne spremenljivke Y ne glede na izid prvega poskusa (spremenljivke X), pa je:

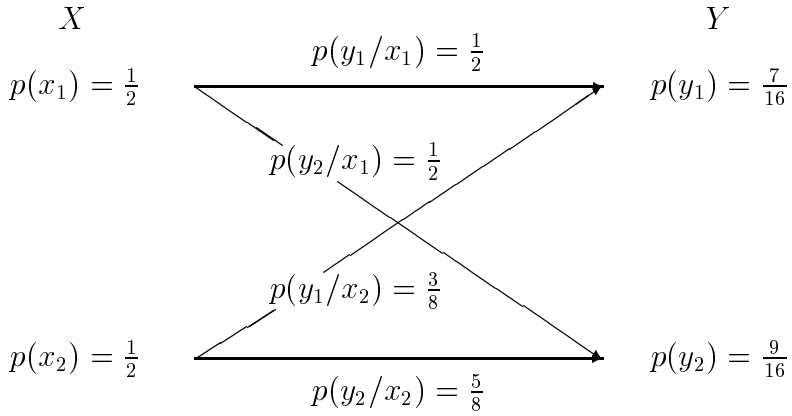
$$p(y_1) = p(x_1) \times p(y_1/x_1) + p(x_2) \times p(y_1/x_2) = \frac{1}{2} \times \frac{1}{2} + \frac{1}{2} \times \frac{3}{8} = \frac{7}{16},$$

$$p(y_2) = p(x_1) \times p(y_2/x_1) + p(x_2) \times p(y_2/x_2) = \frac{1}{2} \times \frac{1}{2} + \frac{1}{2} \times \frac{5}{8} = \frac{9}{16},$$

in

$$Y = \begin{pmatrix} y_1, & y_2 \\ \frac{7}{16}, & \frac{9}{16} \end{pmatrix}.$$

Odnos med naključnima spremenljivkama včasih predočimo grafično. Za naš primer:



Matematično upanje in varianca naključne spremenljivke *Matematično upanje*, srednja ali povprečna vrednost naključne spremenljivke

$$X = \begin{pmatrix} x_1, & x_2, & \ldots, & x_i, & \ldots, & x_n \\ p_1, & p_2, & \ldots, & p_i, & \ldots, & p_n \end{pmatrix}$$

je po definiciji

$$\overline{x} = \sum_{i=1}^{n} x_i \times p_i.$$

Matematično upanje je merilo centralne tendence naključne spremenljvke. Pogosto ga označujemo tudi z $E(X)$ ali z $M(X)$.

Varianca naključne spremenljivke X je po definiciji srednja vrednost kvadratov odstopanj od srednje vrednosti naključne spremenljivke,

$$\sigma^2 = \sum_{i=1}^{n} (x_i - \bar{x})^2 \times p_i$$

Standardni odklon je definiran kot pozitivni kvadratni koren variance,

$$\sigma = +\sqrt{\sigma^2}.$$

Varianco označujemo tudi z $D(X)$. Varianca je merilo za razpršenost naključne spremenljivke. Varianco se včasih lažje izračuna po naslednji formuli,

$$\sigma^2 = \sum_{i=1}^{n} x_i^2 \times p_i - \left(\sum_{i=1}^{n} x_i \times p_i\right)^2 = E(X^2) - (E(X))^2.$$

Sami se prepričajte v veljavnost formule.

Primer 12 Matematično upanje in varianca

Izračunajmo matematično upanje in varianco naključne spremenljivke, ki jo priredimo metu kocke,

$$X = \begin{pmatrix} 1, & 2, & 3, & 4, & 5, & 6 \\ \frac{1}{6}, & \frac{1}{6} & \frac{1}{6}, & \frac{1}{6}, & \frac{1}{6}, & \frac{1}{6} \end{pmatrix}$$

$$\overline{x} = \sum_{i=1}^{6} p_i \times x_i = \frac{1}{6} \times 1 + \frac{1}{6} \times 2 + \frac{1}{6} \times 3 + \frac{1}{6} \times 4 + \frac{1}{6} \times 5 + \frac{1}{6} \times 6 = \frac{7}{2}.$$

$$\sigma_x^2 = \frac{1}{6} \times 1^2 + \frac{1}{6} \times 2^2 + \frac{1}{6} \times 3^2 + \frac{1}{6} \times 4^2 + \frac{1}{6} \times 5^2 + \frac{1}{6} \times 6^2 - \left(\frac{7}{2}\right)^2 = \frac{35}{12}.$$

Važnejše porazdelitve

Enakomerna porazdelitev Naključna spremenljivka $X = \{x_i\}$ $(i = 1, 2, \ldots, n)$ zavzame vsako vrednost z enako verjetnostjo, $P(X) = \{P(X = x_i)\} = \{\frac{1}{n}\}$ $(i = 1, 2, \ldots, n)$. Njeno matematično upanje je (aritmetična sredina):

$$\overline{x} = \frac{1}{n} \sum_{i=1}^{n} x_i.$$

Binomska porazdelitev Naključna spremenljivka z zalogo vrednosti $\{0, 1, 2, \ldots, k, \ldots, n\}$ ima porazdelitev

$$p_k = \binom{n}{k} p^k (1-p)^{n-k} \quad (0 < p < 1)$$

Porazdelitev ima dva parametra, n in p. Označujemo jo tudi z $b(n, p)$. Binomska porazdelitev nam pride prav pri ponavljanju poskusa, v katerem sta možna dva dogodka, A in $\bar{A}$. Takemu zaporedju poskusov pravimo Bernoullijevo zaporedje.

Ponavljajmo poskus n-krat. Vprašajmo se, kakšna je verjetnost, da se dogodek A z verjetnostjo p zgodi ravno k-krat, ne večkrat ne manjkrat. Torej se dogodek $\bar{A}$ zgodi $(n-k)$-krat. To se lahko zgodi na $\binom{n}{k}$ načinov. Zato je verjetnost, da se v n ponovitvah A zgodi natanko k-krat enaka

$$p_k = \binom{n}{k} p^k \times (1-p)^{k-n}$$

Matematično upanje naključne spremenljivke, ki je porazdeljena po binomskem zakonu, je np in njena varianca je $np(1-p)$.

Poissonova porazdelitev Naključna spremenljivka z neskončno zalogo vrednosti $X = \{0, 1, \ldots, k, \ldots\}$ je porazdeljena po Poissonovem zakonu, če je

$$p_k = \frac{\lambda^k e^{-\lambda}}{k!}, \quad \lambda > 0.$$

Porazdelitev je odvisna od parametra λ. Parameter λ je hkrati matematično upanje in standardni odklon naključne spremenljivke, ki je porazdeljena po Poissonovem zakonu. Pri majhnem p in velikem n je binomska porazdelitev skoraj taka kot Poissonova za $\lambda = np$.

Geometrijska porazdelitev Naključna spremenljivka z zalogo vrednosti $k = 1, 2, \ldots$ je porazdeljena po geometrijskem zakonu, če je:

$$p_k = p \times (1-p)^{k-1}.$$

Pride nam prav pri ponavljanju poskusa, v katerem sta možna dogodka A in $\bar{A}$ z verjetnostima $(1-p)$ in p, dokler se ne zgodi dogodek A.

Denimo, da ponovimo poskus k-krat. Naj se najprej $(k-1)$-krat zgodi dogodek $\bar{A}$ z verjetnostjo $(1-p)$, in nazadnje dogodek A z verjetnostjo p. Verjetnost za to je enaka

$$(1-p)^{k-1} \times p.$$

Primer 13 Geometrijska porazdelitev

Oddajnik oddaja sporočila v kanal. Verjetnost napake na sporočilu je $p_n = 0.1$. Oddajnik ponavlja sporočilo, dokler sprejemnik ne sprejme sporočila brez napake. Najmanj kolikokrat je potrebno prenašati sporočilo, da je verjetnost pravilnega prenosa vsaj 0.999.

samo enkrat - prvič brez napake	$(1-p) = 0.9$
dvakrat - prvič z napako in drugič brez	$p \times (1-p) = 0.1 \times 0.9$
trikrat	$p^2 \times (1-p)$
...	...
k-krat	$p^{k-1} \times (1-p)$
...	...

$$0.999 \leq (1-p) + p \times (1-p) + p^2 \times (1-p) + \cdots = (1-p)(1+p+p^2+\cdots)$$

$$0.99 \leq 0.9 \times (1 + 0.1 + 0.01) = 0.999.$$

Oddajnik mora ponoviti sporočilo vsaj trikrat.

Primer 14 Povprečno število ponovitev

Izračunajmo še povprečno število prenašanj $\overline{k}$ (matematično upanje naključne spremenljivke, ki je porazdeljena po geometrijskem zakonu):

$$\begin{aligned}
\overline{k} &= \sum_{k=1}^{\infty} kp^{k-1}(1-p) = (1-p)\sum_{k=1}^{\infty} kp^{k-1} \\
&= (1-p)\frac{d}{dp}\sum_{k=1}^{\infty} p^k = (1-p)\frac{d}{dp}(\frac{1}{1-p} - 1) \\
&= (1-p)\frac{1}{(1-p)^2} = \frac{1}{1-p} = \frac{10}{9}.
\end{aligned}$$

Torej je v povprečju približno 10 odstotkov prenosov odveč.

Primer 15 Binomska porazdelitev

Binarni vir $V = \{0, 1\}$ oddaja simbole v simetrični binarni informacijski kanal. Verjetnost napake na simbolu (sprememba $0 \to 1$ ali $1 \to 0$) je $p_n = 0.1$. Vsak simbol vira na oddajni strani trikrat ponovimo (prenašamo trikrat),

$$0 \to x_0 = 000$$

$$1 \to x_1 = 111$$

Zaradi morebitnih napak med prenosom, dobimo na sprejemni strani poljubno zaporedje treh binarnih simbolov.

Sprejemnik se odloča po principu večje pogojne verjetnosti $p(x_i/y_j)$ $(i = 0, 1)$. Na primer, če sprejme zaporedje $y_1 = 001$, se odloči, da je oddano zaporedje $x_0 = 000$ in ne $x_1 = 111$, ker je pogojna verjetnost $p(x_0/y_1) = p(000/001)$ večja od pogojne verjetnosti $p(x_1/y_1) = p(111/001)$. Pogojna verjetnost, da sprejmemo $y_1 = 001$ če oddamo $x_0 = 000$, je:

$$p(y_1/x_0) = p(001/000) = 0.9 \times 0.9 \times 0.1 = 0.081.$$

Pogojna verjetnost, da sprejmemo $y_1 = 001$ če oddamo $x_1 = 111$, je manjša:

$$p(y_1/x_1) = p(001/111) = 0.1 \times 0.1 \times 0.9 = 0.009,$$

ker je verjetnost napake manjša od $\frac{1}{2}$. Pogojna verjetnost, da je bilo oddano zaporedje $x_0 = 000$, če sprejmemo $y_1 = 001$, pa sledi iz formule za vezano verjetnost:

$$p(000) \times p(001/000) = p(001) \times p(000/001).$$

Dobimo:

$$p(x_0/y_1) = p(000/001) = \frac{p(000) \times p(001/000)}{p(001)}$$

Podobno izračunamo verjetnost, da je oddano zaporedje $x_1 = 111$, če sprejmemo $y_1 = 001$,

$$p(x_1/y_1) = p(111/001) = \frac{p(111) \times p(001/111)}{p(001)}$$

Če sta zaporedji $x_0 = 000$ in $x_1 = 111$ enakoverjetni $(p(x_0) = p(x_1) = \frac{1}{2})$, je odločanje na osnovi večje pogojne verjetnosti $p(x_i/y_j)$ $(i = 0, 1)$ enakovredno odločanju na osnovi večje pogojne verjetnosti $p(y_j/x_i)$ $(i = 0, 1)$. To je hkrati večinski princip: sprejemnik se odloči, da je bil oddan tisti simbol, ki se v sprejetem zaporedju večkrat pojavi.

Vprašanje:

Kakšna je verjetnost, da se bo sprejemnik napačno odločil (kakšna je verjetnost napake).

Sprejemnik se odloči napačno, če se zgodita dve napaki ali tri napake. Po binomskem porazdelitvenem zakonu je verjetnost dveh napak enaka:

$$p_2 = \binom{3}{2} \times 0.1^2 \times 0.9^1 = 3 \times 0.009 = 0.027.$$

Verjetnost treh napak je:

$$p_3 = \binom{3}{3} \times 0.1^3 \times 0.9^0 = 0.1^3 = 0.001.$$

Verjetnost napačne odločitve, je

$$p_{2,3} = p_2 + p_3 = 0.027 + 0.001 = 0.028,$$

kar je nekajkrat manj, kot če bi prenašali vsak simbol vira samo enkrat.

Vprašanje:

Denimo, da binarni simbol, ki ga odda vir, prenašamo trikrat, torej enako kot prej, le da sprejemnik tokrat zavrže vsako zaporedje treh neenakih simbolov. Kolikšna je verjetnost napačne odločitve, torej primera, ko se sprejemnik (napačno) odloči, da je bil oddan simbol nič, če je bil v resnici oddan simbol ena ali obratno.

Verjetnost, da do tega pride, je enaka verjetnosti treh napak, torej

$$0.1 \times 0.1 \times 0.1 = 0.001,$$

kar je neprimerno manj kot prej.

C Entropija

Lastna entopija Lastna entropija ali kar samo *entropija* naključne spremenljivke X, ki zmore zavzeti eno od n diskretnih vrednosti ali stanj,

$$X = \begin{pmatrix} x_1, & x_2, & \ldots, & x_i, & \ldots, & x_n \\ p(x_1), & p(x_2), & \ldots, & p(x_j), & \ldots, & p(x_n) \end{pmatrix}.$$

je po definiciji (C.E. Shannon, 1948) enaka

$$H(X) = -\sum_{i=1}^{n} p(x_i)\, log_2\, p(x_i) \quad \text{bitov na stanje.}$$

Entropija je svojska lastnost naključne spremenljivke, podobno kot na primer njeno matematično upanje.

Posamezni vrednosti (stanju) naključne spremenljivke X priredimo nedoločenost $h(p_i)$, ki je sorazmerna logaritmu obratne vrednosti verjetnosti stanja,

$$h(p_i) = log\frac{1}{p_i} \quad (i = 1, 2, \ldots, n).$$

Čim manjša je verjetnost stanja x_i, tem bolj je to stanje nedoločeno in težje uganemo, da je naključna spremenljivka zavzela prav to stanje.

Entropija naključne spremenljivke je (po definiciji) enaka *povprečni nedoločenosti* na stanje,

$$H(X) = \sum_{i=1}^{n} p(x_i)\ h(p_i).$$

Večkrat zapišemo entropijo naključne spremenljivke tako, da zraven zapišemo njen porazdelitveni zakon,

$$H(X) = H(p_1, p_2, \ldots, p_i, \ldots, p_n).$$

V primeru, da so vsa stanja naključne spremenljivke X enako verjetna, je

$$p_i = \frac{1}{n} \quad (i = 1, 2, \ldots, n)$$

in njena entropija je

$$H(X) = \sum_{i=1}^{n} p_i log p_i = \sum_{i=1}^{n} \frac{1}{n} log n = log n$$

Entropija naključne spremenljivke z dvema enakoverjetnima stanjema je enaka 1 bit.

Primer 16 Met kovanca

Priredimo metu kovanca naključno spremenljivko X,

$$X = \begin{pmatrix} grb, & cifra \\ 0.5 & 0.5 \end{pmatrix}$$

Njena entropija je:

$$H(X) = H(\frac{1}{2}, \frac{1}{2}) = log_2 2 = 1\ \ bit/stanje$$

Primer 17 Izbiranje karte

Priredimo izbiranju karte iz svežnja 32 kart naključno spremenljivko Y z zalogo vrednosti $\{y_i\} \quad (i = 1, 2, \ldots, 32)$. Ker imajo vse karte enako možnost, da so izbrane, imamo za entropijo:

$$H(Y) = H(\underbrace{\frac{1}{32}, \frac{1}{32}, \cdots, \frac{1}{32}}_{32}) = log_2 32 = 5 \quad \text{bitov / stanje}$$

Primer 18 Dež

Verjetnost, da bo ob sončnem vremenu deževalo je vsaj nekaj velikostnih razredov manjša od verjetnosti, da dežja ne bo. Opišimo to dejstvo z naključno spremenljivko Z,

$$Z = \begin{pmatrix} \text{dež}, & \text{sonce} \\ 0.001, & 0.999 \end{pmatrix}$$

in njena entropija je:

$H(Z) = H(0.001, 0.999) = -0.001 log_2 0.001 - 0.999 log_2 0.999 = 0.001$ bitov / stanje

Primer 19 Entropija vira

Izračunajmo entropijo binarnega informacijskega vira V, ki odda simbol v_1 z verjetnostjo $p_1 = \frac{1}{4}$, simbol v_2 pa z verjetnostjo $p_2 = \frac{3}{4}$,

$$V = \begin{pmatrix} v_1, & v_2 \\ \frac{1}{4}, & \frac{3}{4} \end{pmatrix}.$$

Njegova entropija je:

$$H(V) = H(\frac{1}{4}, \frac{3}{4}) = -\frac{1}{4} log_2 \frac{1}{4} - \frac{3}{4} log_2 \frac{3}{4} = 0.5 + 0.311 = 0.811 \ \textit{bitov/na simbol.}$$

Primer 20 Entropija sporočila

Izračunajmo entropijo sporočila vira V dolžine $G = 3$. Zanima nas torej entropija zaporedja treh neodvisnih in enakih naključnih spremenljivk v trenutkih $t, t+1$ in $t+2$,

$$V_G = V^t \, V^{t+1} \, V^{t+2}.$$

Naključna spremenljivka V_G ima $2^3 = 8$ različnih možnih sporočil. Njena verjetnostna shema je:

$$V_G = \begin{pmatrix} v_1 v_1 v_1, & v_1 v_1 v_2, & v_1 v_2 v_1, & v_1 v_2 v_2, & v_2 v_1 v_1, & v_2 v_1 v_2, & v_2 v_2 v_1, & v_2 v_2 v_2, \\ \frac{1}{64}, & \frac{3}{64}, & \frac{3}{64}, & \frac{9}{64}, & \frac{3}{64}, & \frac{9}{64}, & \frac{9}{64}, & \frac{27}{64} \end{pmatrix}.$$

Za entropijo imamo:

$$H(V_G) = H(\frac{1}{64}, \frac{3}{64}, \frac{3}{64}, \frac{9}{64}, \frac{3}{64}, \frac{9}{64}, \frac{9}{64}, \frac{27}{64}) = 2.43 \ \text{bitov / sporočilo.}$$

V splošnem:

$$\begin{aligned} H(V_G) &= -\sum_{i=1}^{2}\sum_{j=1}^{2}\sum_{k=1}^{2} p_i p_j p_k log_2 p_i p_j p_k \\ &= -\sum_{i=1}^{2}\sum_{j=1}^{2}\sum_{k=1}^{2} p_i p_j p_k (log_2 p_i + log_2 p_j + log_2 p_k) \\ &= -\sum_{i=1}^{2}\sum_{j=1}^{2}\sum_{k=1}^{2} p_i p_j p_k log_2 p_i - \sum_{i=1}^{2}\sum_{j=1}^{2}\sum_{k=1}^{2} p_i p_j p_k log_2 p_j - \sum_{i=1}^{2}\sum_{j=1}^{2}\sum_{k=1}^{2} p_i p_j p_k log_2 p_k \\ &= -\sum_{i=1}^{2} p_i log_2 p_i \sum_{j=1}^{2} p_j \sum_{k=1}^{2} p_k - \sum_{j=1}^{2} p_j log_2 p_j \sum_{i=1}^{2} p_i \sum_{k=1}^{2} p_k - \sum_{k=1}^{2} p_k log_2 p_k \sum_{i=1}^{2} p_i \sum_{j=1}^{2} p_j \\ &= H(V) + H(V) + H(V) = 3 \times H(V) \end{aligned}$$

kot bi tudi pričakovali. Torej,

$$H(V_G) = G \times H(V) = 3 \times 0.811 = 2.43 \ \text{bitov/sporočilo}$$

Sestavljene entropije, množina informacije Informacija nastane, ko se nedoločenost naključne spremenljivke zmanjša. Od naključne spremenljivke dobimo toliko informacije, kolikor se zmanjša njena nedoločenost.

Naj ima naključna spremenljivka X entropijo $H(X)$. Ko naključno spremenljivko spoznamo (vemo, kakšno vrednost je zavzela), pade njena nedoločenost na nič. Če bi opazovali daljše zaporedje realizacij naključne spremenljivke, bi z vsako realizacijo od nje dobili (v povprečju)

$$I(X) = H(X) - 0 = H(X)$$

informacije.

Vezana entropija $H(XY)$ naključnih spremenljivk X in Y je entropija spremenljivke, ki je enaka produktu teh spremenljivk,

$$H(XY) = -\sum_{i=1}^{n}\sum_{j=1}^{m} p(x_i, y_j) log p(x_i, y_j),$$

kjer je $p(x_i, y_j)$ verjetnost, da naključna spremenljivka X zavzame vrednost x_i, naključna spremenljivka Y pa vrednost y_j,

$$p(x_i, y_j) = P(X = x_i, Y = y_j).$$

Če sta naključni spremenljivki medsebojno odvisni, se z realizacijo ene spremeni porazdelitveni zakon druge in s tem njena entropija. Naj bo porazdelitveni zakon spremenljivke X v primeru, da naključna spremenljivka Y zavzame vrednost y_j, enak

$$P(X/Y = y_j) = \{p(x_1/y_j), p(x_2/y_j, \ldots, p(x_i/y_j, \ldots, p(x_n/y_m)\},$$

pri čemer je $p(x_i/y_j)$ pogojna verjetnost, da naključna spremenljivka X zavzame vrednost x_i, če je naključna spremenljivka Y zavzela vrednost y_j,

$$p(x_i/y_j) = P(X = x_i/Y = y_j).$$

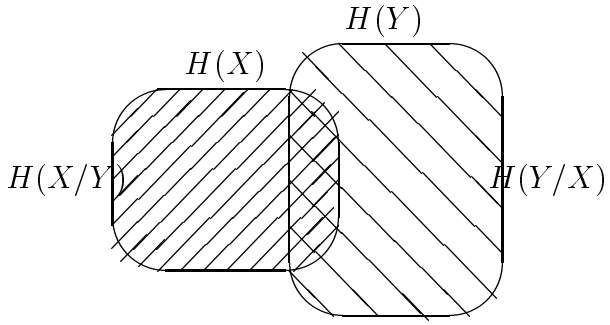
Pogojna entropija naključne spremenljivke X, ko naključna spremenljivka Y zavzame vrednost y_j, je po definiciji enaka

$$H(X/Y = y_j) = -\sum_{i=1}^{n} p(x_i/y_j) log p(x_i/y_j).$$

Pogojna entropija $H(X/Y)$ naključne spremenljivke X glede na naključno spremenljivko Y (ne glede na to, kakšno vrednost zavzame naključna spremenljivka Y), je enaka matematičnemu upanju pogojnih entropij $H(X/Y = y_j)$,

$$H(X/Y) = \sum_{j=1}^{m} p(y_j) H_j = \sum_{j=1}^{m} p(y_j) H(X/Y = y_j).$$

Zvezo med lastnima entropijama $H(X)$ in $H(Y)$, pogojnima entropijama $H(X/Y)$ in $H(Y/X)$ ter vezano entropijo $H(X,Y)$ se da ponazoriti tudi grafično.



Z ovaloma predočimo lastni entropiji $H(X)$ in $H(Y)$. Enkratno šrafirani področji predočata pogojni entropiji $H(X/Y)$ in $H(Y/X)$. Vezani entropiji $H(X,Y)$ ustreza področje obeh ovalov (obe šrafirani področji in dvojno šrafirano področje),

$$H(X,Y) = H(X) + H(Y/X) = H(Y) + H(X/Y).$$

Medsebojna ali prevajana informacija Medsebojna informacija naključnih spremenljivk X in Y je definirana kot zmanjšanje povprečne nedoločenosti (entropije) ene spremenljivke ko spoznamo realizacijo druge spremenljivke.

Naj ima naključna spremenljivka X entropijo $H(X)$. Entropija naključne spremenljivke Y naj bo $H(Y)$. Ko naključno spremenljivko Y spoznamo, se njena nedoločenost zmanjša na nič. Možno je, da se z opazovanjem naključne spremenljivke Y zmanjša tudi nedoločenost naključne spremenljivke X, vendar v splošnem ne na nič. Nedoločenost (entropijo) spremenljivke X, ki še ostane, smo imenovali pogojna entropija $H(X/Y)$. Razlika entropij $H(X)$ in $H(X/Y)$ je enaka medsebojni informaciji med naključnima spremenljivkama X in Y,

$$I(X,Y) = H(X) - H(X/Y).$$

Medsebojna (prevajana) informacija je merilo za to, koliko nam realizacija ene naključne spremenljivka pove o stanju (realizaciji) druge naključne spremenljivke.

Dvojno šrafirano področje na sliki ponazarja *medsebojno* ali *prevajano informacijo*,

$$I(X,Y) = H(X) - H(X/Y) = H(Y) - H(Y/X) = H(X) + H(Y) - H(X,Y).$$

Funkcija za medsebojno informacijo je simetrična,

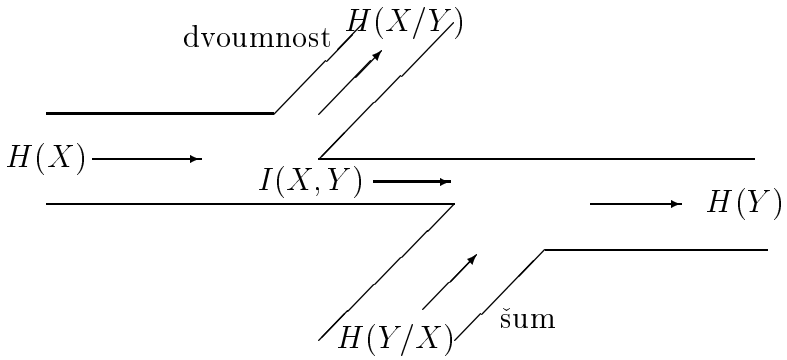
$$I(X,Y) = I(Y,X).$$

Če sta naključni spremenljivki X in Y neodvisni (ovala se ne prekrivata), veljajo sledeče enakosti:

$$H(X,Y) = H(X) + H(Y)$$

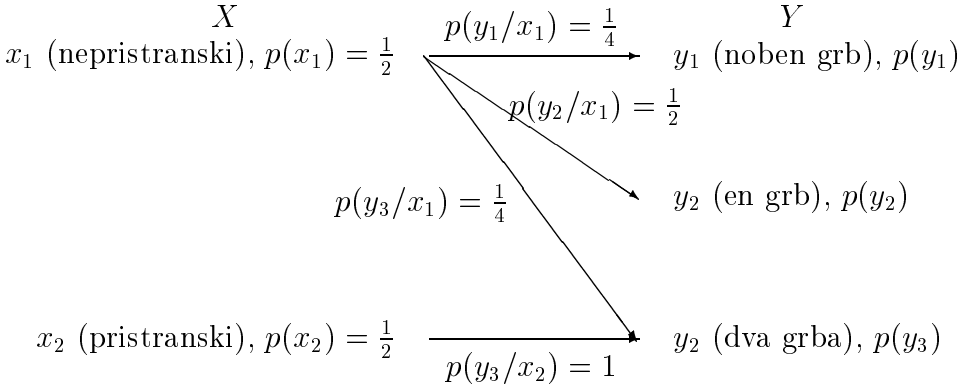
$$H(X/Y) = H(X)$$
$$H(Y/X) = H(Y)$$

Medsebojna informacija $I(X,Y)$ neodvisnih naključnih spremenljivk je enaka nič. Medsebojno ali prevajano infoprmacijo lahko grafično ponazorimo tudi tako, kot prikazuje naslednja skica.



Primer 21 Računanje medsebojne informacije
Imejmo dva kovanca, *pristranskega* in *nepristranskega*. Pristranski vedno pade kot grb, neprostranski pade z enako verjetnostjo na eno ali drugo plat. Naključno izberemo enega od kovancev in ga dvakrat mečemo. Zanima nas, koliko nam dvakratni met kovanca pove o tem, katerega od kovancev smo izbrali. Torej, koliko se da iz izida dvakratnega meta kovanca sklepati na izid izbiranja kovanca.

Priredimo poskusu izbiranja kovanca naključno spremenljivko X, dvakratnemu metu kovanca pa naključno spremenljivko $H(Y)$. Tisto, kar nas zanima, je medsebojna informacija $I(X,Y)$ med naključnima spremenljivkama X in Y. Predočimo zvezo med spremenljivkama X in Y grafično.



Entropija naključne spremenljivke X, ki je prirejena izbiranju kovanca, je enaka

$$H(X) = H(p(x_1), p(x_2)) = H(\frac{1}{2}, \frac{1}{2}) = 1\text{bit}.$$

Sprva torej, ko izberemo enega od kovancev, o naključni spremenljivki X ne vemo ničesar - ne vemo, katerega od kovancev smo izbrali. Ko spoznamo izid dvakratnega meta izbranega kovanca (naključno spremenljivko Y), se nedoločenost naključne spremenljivke X zmanjša, vendar s tem naključne spremenljivke X popolnoma ne spoznamo, njena nedoločenost je še vedno $H(X/Y)$. Izračunajmo jo.

Po formuli za popolno verjetnost izračunajmo najprej porazdelitveni zakon naključne spremenljivke Y,

$$\begin{aligned} p(y_1) &= \frac{1}{2} \times \frac{1}{4} + \frac{1}{2} \times 0 = \frac{1}{8}, \\ p(y_2) &= \frac{1}{2} \times \frac{1}{2} + \frac{1}{2} \times 0 = \frac{1}{4}, \\ p(y_3) &= \frac{1}{2} \times \frac{1}{4} + \frac{1}{2} \times 1 = \frac{5}{8}. \end{aligned}$$

Da izračunamo pogojno entropijo $H(X/Y)$, moramo prej izračunati pogojne verjetnosti $p(x_i/y_j), \quad (i = 1, 2;\ j = 1, 2, 3)$. Za $j = 1$ dobimo:

$$\begin{aligned} p(x_1/y_1) &= \frac{p(x_1) \times p(y_1/x_1)}{p(y_1)} = \frac{1/2 \times 1/4}{1/8} = 1, \\ p(x_2/y_1) &= 0, \end{aligned}$$

kar je s slike očitno. Podobno velja za $j = 2$:

$$\begin{aligned} p(x_1/y_2) &= \frac{p(x_1) \times p(y_2/x_1)}{p(y_2)} = \frac{1/2 \times 1/2}{1/4} = 1, \\ p(x_2/y_2) &= 0. \end{aligned}$$

Izračunajmo pogojni verjetnosti še za $j = 3$:

$$\begin{aligned} p(x_1/y_3) &= \frac{p(x_1) \times p(y_3/x_1)}{p(y_3)} = \frac{1/2}{\times}\frac{1/4}{5/8} = \frac{1}{5} \\ p(x_2/y_3) &= \frac{4}{5}. \end{aligned}$$

Pogojne entropije $H(X/Y = y_j) \quad (j = 1, 2, 3)$ so:

$$\begin{aligned} H_1 &= H(p(x_1/y_1), p(x_2/y_1)) = H(1, 0) = 0, \\ H_2 &= H(p(x_1/y_2), p(x_2, y_2)) = H(1, 0) = 0, \\ H_3 &= H(p(x_1/y_3), p(x_2/y_3)) = H(\frac{1}{5}, \frac{4}{5}) = 0.72. \end{aligned}$$

Pogojna entropija $H(X/Y)$ pa je:

$$H(X/Y) = p(y_1)H_1 + p(y_2)H_2 + p(y_3)H_3 = \frac{1}{8} \times 0 + \frac{1}{4} \times 0 + \frac{5}{8} \times 0.72 = 0.45$$

Z dvakratnim metom izbranega kovanca (Y) smo o izbranem kovancu (X) dobili nekaj informacije, in sicer prav toliko, kolikor se zmanjša njena nedoločenost,

$$I(X,Y) = H(X) - H(X/Y) = 1 - 0.45 = 0.55.$$

Do istega rezultata bi prišli z nekaj manj računanja po drugi poti,

$$I(Y,X) = H(Y) - H(Y/X).$$

Lastna entropija naključne spremenljivke Y je:

$$H(Y) = H(p(y_1), p(y_2), p(y_3)) = H(\frac{1}{8}, \frac{2}{8}, \frac{5}{8}) = 1.3.$$

Pogojna entropija $H(Y/X)$ je:

$$H(Y/X) = p(x_1) \times H_1 + p(x_2) \times H_2 = \frac{1}{2} \times 1.5 + \frac{1}{2} \times 0 = 0.75$$

ker je

$$\begin{aligned} H_1 &= H(p(y_1/x_1), p(y_2/x_1), p(y_3/x_1)) = H(\frac{1}{4}, \frac{1}{2}, \frac{1}{4}) = 1.5 \\ H_2 &= H(p(y_1/x_2), p(y_2/x_2), p(y_3/x_2)) = H(0,0,1) = 0 \end{aligned}$$

in medsebojna informacija je

$$I(X,Y) = H(Y) - H(Y/X) = 1.3 - 0.75 = 0.55.$$

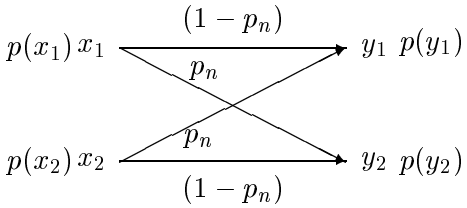
Dobljeni rezultat za $I(X,Y)$ bi lahko razlagali takole. Denimo, da ponovimo poskus izbiranja kovanca (pristranski, nepristranski) 10000-krat. Realizacije poskusa ne poznamo in na nas je, da jo ugotovimo. Da ugotovimo realizacijo enega poskusa, rabimo en bit informacije, za 10000 (neodvisnih) realizacij torej rabimo 10000 bitov. Če bi nam nekdo dal o teh poskusih toliko informacije, bi bili sposobni povedati, kakšna je realizacija vsake ponovitve poskusa. Obrnjeno, ko zvemo realizacijo zaporedja poskusov, dobimo 10000 bitov informacije.

Sedaj pa vzemimo, da moramo uganiti realizacijo poskusa izbiranja kovanca (pristranski, nepristranski), vendar nam je tokrat dana možnost, da izbrani kovanec pred ugibanjem dvakrat vržemo. Ker poznamo realizacijo dvakratnega meta, bi moralo biti ugibanje lažje. Spet ponovimo poskus izbiranja kovanca 10000-krat. Iz zgornjega izračuna vidimo, da nam realizacija dvakratnega meta v povprečju da 0.55 bitov informacije in 10000 ponovitev nam da tolikokrat več, torej 5500 bitov. Da bi uganili izid izbiranja kovanca (pristranski, nepristranski)

rabimo samo še 4500 bitov informacije, toliko, kolikor je preostala nedoločenost teh 10000 poskusov.

Primer 22 Prevajana informacija po binarnem simetričnem kanalu
Izračunajmo prevajano informacijo po binarnem simetričnem kanalu (BSK) za verjetnost napake $p_n = 0, \frac{1}{4}$ in $\frac{1}{2}$, če na vhod kanala deluje inofrmacijski vir X_1, X_2 ali X_3,

$$X_1 = \begin{pmatrix} x_1, x_2 \\ 0, 1 \end{pmatrix}, \quad X_2 = \begin{pmatrix} x_1, x_2 \\ \frac{1}{4}, \frac{3}{4} \end{pmatrix}, \quad X_3 = \begin{pmatrix} x_1, x_2 \\ \frac{1}{2}, \frac{1}{2} \end{pmatrix}.$$



Računajmo po formuli

$$\begin{aligned} I(X,Y) &= H(Y) - H(Y/X) = H(Y) - [p(x_1)H(p_n, 1-p_n) + p(x_2)H(1-p_n, p_n)] \\ &= H(p(y_1), p(y_2)) - H(p_n, 1-p_n). \end{aligned}$$

Zadnji člen (pogojna entropija $H(Y/X)$) je odvisen samo od kanala, prvi člen (entropija $H(Y)$), je odvisna od vira in od kanala,

$$H(Y) = H(p(x_1)p_n + p(x_2)(1-p_n), p(x_1)(1-p_n) + p(x_2)p_n).$$

Za $p_n = 0$, imamo:

$$I(X,Y) = H(Y) = H(p(x_1), p(x_2)) = H(X).$$

Prevajana informacija za X_1, X_2 in X_3 je:

$$\begin{aligned} I(X_1, Y_1) &= H(0,1) = 0 \\ I(X_2, Y_2) &= H(\frac{1}{4}, \frac{3}{4}) = 0.811 \\ I(X_3, Y_3) &= H(\frac{1}{2}, \frac{1}{2}) = 1 \end{aligned}$$

Za $p_n = \frac{1}{4}$, imamo:

$$I(X,Y) = H(p(x_1)\frac{1}{4} + p(x_2)\frac{3}{4}, p(x_1)\frac{3}{4} + p(x_2)\frac{1}{4}) - H(\frac{1}{4}, \frac{3}{4}).$$

Prevajana informacija za X_1, X_2 in X_3 je:

$$\begin{aligned} I(X_1, Y_1) &= H(\frac{1}{4}, \frac{3}{4}) - H(\frac{1}{4}, \frac{3}{4}) = 0 \\ I(X_2, Y_2) &= H(\frac{5}{8}, \frac{3}{8}) - H(\frac{1}{4}, \frac{3}{4}) = 0.954 - 0.811 = 0.143 \\ I(X_3, Y_3) &= H(\frac{1}{2}, \frac{1}{2}) - H(\frac{1}{4}, \frac{3}{4}) = 1 - 0.811 = 0.169 \end{aligned}$$

Za $p_n = \frac{1}{2}$, imamo:

$$I(X,Y) = H(p(x_1)\frac{1}{2}+p(x_2)\frac{1}{2}, p(x_1)\frac{1}{2}+p(x_2)\frac{1}{2}) - H(\frac{1}{2}, \frac{1}{2}) = H(\frac{1}{2}, \frac{1}{2}) - H(\frac{1}{2}, \frac{1}{2}) = 0,$$

neodvisno od vira.

Izračunane vrednosti so zbrane v tabeli.

	$X_1 = \begin{pmatrix} x_1, x_2 \\ 0, 1 \end{pmatrix}$,	$X_2 = \begin{pmatrix} x_1, x_2 \\ \frac{1}{4}, \frac{3}{4} \end{pmatrix}$,	$X_3 = \begin{pmatrix} x_1, x_2 \\ \frac{1}{2}, \frac{1}{2} \end{pmatrix}$.
$p_n = 0$	0	0.811	1.000
$p_n = \frac{1}{4}$	0	0.143	0.189
$p_n = \frac{1}{2}$	0	0	0

Ker je kanal simetričen, je prevajana informacija simetrična glede na $p(x_1) = \frac{1}{2}$.