1 UVOD

1.1 Potreba po protokolih in standardih

Ljudje uporabljamo za sporazumevanje med sabo jezike, ki jim rečemo tudi **naravni jeziki**. Jezike uporabljamo tudi za sporazumevanje s stroji (računalniki); najbolj znani tovrstni jeziki so **programski jeziki**.

Predstavljajmo si situacijo, kakršna nastane, če se srečata predstavnika dveh različnih narodov, npr. Zulujec in Norvežan, vsak od njiju pa govori le svoj jezik. Kljub svoji najboljši volji in naravni inteligenci, ki jo premore **homo sapiens**, se bosta le težko sporazumela, saj ne govorita istega jezika, pri pripadnikih tako različnih in tako oddaljenih narodov pa imajo lahko tudi kretnje različen pomen. Če naj se torej dva ali več ljudi razume med seboj, morajo govoriti isti jezik. Na svetu pa so tudi jeziki (npr. kitajščina), ki imajo veliko dialektov, ti pa se lahko med seboj razlikujejo do te mere, da se ljudje, ki govorijo različne dialekte sicer istega jezika, sploh ne razumejo med seboj*. Rešitev lahko najdejo v uporabi knjižnega (torej v nekem smislu standardnega) jezika. Tudi v mednarodnem sporazumevanju so se uveljavili nekateri jeziki kot standardni; nekoč je bila to francoščina, danes angleščina (zato bomo tudi v tem besedilu navajali angleške izraze za nekatere najpomembnejše strokovne pojme!).

Če pa med seboj komunicirajo stroji (ti stroji so v današnjem času v veliki večini kar računalniki), je problem seveda še veliko bolj pereč, saj ti razlike v uporabljenih jezikih ne morejo nadomestiti z inteligenco. Računalnik lahko sicer komunicira v več različnih jezikih, vendar pa mora pred začetkom neke konkretne komunikacije vedeti, v katerem jeziku bo komuniciral. Za komunikacijo med računalniki, ki poteka preko telekomunikacijskega omrežja, uporabljamo posebne jezike, ki jih lahko za začetek imenujemo kar **protokoli** oziroma **komunikacijski protokoli** oziroma **telekomunikacijski protokoli** (uporabljajo se vsi trije izrazi, mi pa bomo v nadaljevanju največkrat uporabljali kar prvega, ki je najkrajši), kasneje pa bomo pojem protokola še natančneje opredelili. Izraz protokol (gr. **protos** prvi, **kollan** lepiti) se je sprva uporabljal v francoščini za razne podatke, nalepljene na prvo stran pravnih listin, danes pa ta izraz pomeni uradni zapisnik o izjavi ali sklepih konference oziroma pravilnik o načinu

* Pred kratkim sem srečal dva Kitajca, od katerih je eden govoril mandarinski, drugi pa tajvanski dialekt; med sabo sta se pogovarjala v angleščini!

diplomatskega vedenja ali medsebojnega komuniciranja (diplomatski protokol, telekomunikacijski protokol).

Če smo rekli, da so protokoli neke vrste jeziki za komuniciranje, moramo računalnike teh jezikov na tak ali drugačen način naučiti, oziroma jim jih vgraditi. Sestavni deli računalnikov, ki znajo komunicirati po določenem protokolu, so bodisi posebna (temu namenjena) elektronska vezja (**strojna oprema**), ali pa posebni komunikacijski programi (**programska oprema**). Da pa bi lahko med seboj komunicirali tudi računalniki s pripadajočimi programskimi opremami, ki smo jih kupili od različnih proizvajalcev, morajo biti protokoli **standardizirani**. S standardizacijo protokolov in tudi drugih izdelkov, procesov in postopkov se ukvarjajo različne državne in mednarodne institucije.

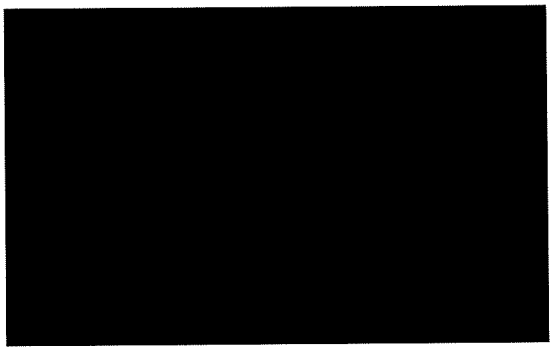
Najbrž je odveč utemeljevati na dolgo in široko, zakaj je študij protokolov sploh potreben v času, ko so telekomunikacije ena izmed tistih panog, ki z medsebojnim povezovanjem strojev in ljudi najbolj spreminjajo svet. Prav z lahkim srcem lahko izrečemo trditev, da današnji svet brez telekomunikacij ne bi bil to, kar je, telekomunikacije pa brez protokolov tudi ne bi bile mogoče, še zlasti ne v takšni obliki in v tolikšnem obsegu. Pojma telekomunikacij in telekomunikacijskih protokolov sta namreč med seboj tesno povezana. Protokoli določajo način komuniciranja, prispevajo pa tudi k določenosti vsebine prenešene informacije. Prav tako, kot telekomunikacijski sistemi postajajo vse bolj kompleksni, pa postajajo vse bolj kompleksni tudi protokoli; njihovo poznavanje torej zahteva temeljit študij.

1.2 Kratka zgodovina podatkovnih telekomunikacij

Zgodovina razvoja telekomunikacij in z njimi vred telekomunikacijskih protokolov je veliko daljša, kot bi si lahko na prvi pogled predstavljali. Prve tehnologije komuniciranja na daljavo so bile osnovane na optičnih signalih. Starogrški dramatik **Ajshil** je že v 5. stoletju pred našim štetjem poročal o prižiganju ognja kot o načinu sporazumevanja na daljavo (ta način so uporabljali tudi naši predniki pred nekaj stoletji, ko so na ta način signalizirali vdore Turkov v deželo). Kasneje (vsaj v 2. stol. pr. n. št.) pa so Grki sistem telekomunikacij še izpopolnili z uporabo desetih bakel v dveh skupinah po pet. Ta protokol pa je poleg samega prenosa koristne informacije predvidel še znak za začetek sporazumevanja (danes bi temu rekli nadzorno sporočilo za vzpostavljanje zveze). Grški zgodovinar **Polybius**, ki je poročal o tej iznajdbi, je že tedaj opozoril na nekatere še danes

aktualne probleme pri načrtovanju protokolov: »še zlasti pa je potrebno posvetiti takojšnjo pozornost nepričakovanim pojavom«. Danes pravimo, da je protokol, ki je odporen na takšne nepričakovane pojave, robusten.

Enega izmed vidnejših in tudi v praksi uporabljenih korakov v razvoju telekomunikacij je v času francoske revolucije napravil francoski duhovnik **Claude Chappe** (1763-1805), ki je iznašel sistem gibljivih mehanskih konstrukcij za signaliziranje sporočil; medtem ko je operater na oddajni strani spreminjal položaj sestavnih delov oddajne naprave, je operater na sprejemni strani ta položaj vizualno odčitaval in ga po potrebi z enako napravo oddal naslednji postaji. Tako je bilo tik pred iznajdbo električnega telegrafa po vsej Franciji 556 takih naprav, ki so omogočale prenos sporočil po vsej državi (glej sliko **Error! Not a valid link.**). Obstajale pa so tudi prenosne telegrafske postaje, kar nam dokazuje, da mobilne telekomunikacije niso iznajdba 20. stoletja (slika **Error! Not a valid link.**)! Model Chappeove naprave lahko danes občudujemo na grobu iznajditelja (ki ga nekateri imenujejo očeta telekomunikacij) na pokopališču Père Lachaise v Parizu.



Slika 1-1. Chappe-ov telegraf

Slika 1-2. Chappe-ov telegraf v krimski vojni, montiran na šotoru

O Chapppovem telegrafu lahko beremo tudi v romanu Grof Monte-Cristo francoskega pisatelja Alexandra Dumasa. V tem romanu, ki je nastal v času, ko je bil Chappe-ov telegraf na vrhuncu popularnosti, zlahka opazimo vsaj dve šibki točki delovanja tovrstnega telekomunikacijskega sistema, ki sta bili očitno znani že tedanjim uporabnikom: odvisnost delovanja sistema od okolja (telegrafist v romanu se je veselil meglenih dni, saj je takrat imel prost dan) in občutljivost sistema na zlonamerne vdore (junak romana je zaradi lastnih interesov podkupil telegrafista, da je le-ta spremenil posredovano sporočilo, in tako postal eden prvih znanih »hackerjev«).

Nova era v telekomunikacijah se je začela v prvi polovici 19. stoletja z iznajdbo elektromagnetnega telegrafa. Za iznajditelja tega telegrafa (1837) štejejo ameriškega umetnika in iznajditelja **Samuela Morseja**, ki je nekoč kot slikar v Parizu opazoval in naslikal Chappeov telegraf. Eden prvih uporabnikov tega sistema so bile železnice, ki so telegraf uporabljale med drugim tudi za zaščito posebej nevarnih odsekov prog, npr. predorov. Pri tem pa protokoli za sporazumevanje med progovnimi čuvaji še niso bili dovolj dodelani, zato je občasno prihajalo celo do železniških nesreč. Še posebej je znano trčenje vlakov, do katerega je prišlo leta 1861 v predoru Clayton v Angliji in je zahtevalo 21 življenj, 176 oseb pa je bilo ranjenih. Nesreča se je zgodila po tem, ko je prišlo do situacije, ki je protokol sporazumevanja med progovnima čuvajema na obeh straneh predora ni predvideval. V predoru sta se zaradi zatajitve signalizacije naenkrat znašla dva vlaka, protokol pa čuvaju, ki je to vedel, ni dajal možnosti, da bi to sporočil drugemu čuvaju, ki tega ni vedel; drugi čuvaj pa tudi ni imel možnosti sporočiti prvemu, koliko vlakov je pripeljalo iz predora. Danes bi rekli, da je bil protokol premalo robusten.

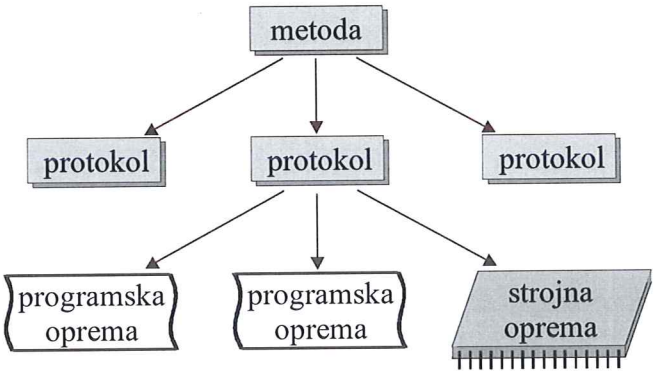
Komunikacijam med posameznimi uporabniki je v 20. stoletju sledilo povezovanje velikega števila uporabnikov v telekomunikacijska omrežja, ki so vedno bolj rastla, še zlasti v zadnjih nekaj desetletjih tisočletja, na kar je in še vedno vpliva razvoj vedno novih telekomunikacijskih in računalniških tehnologij. Med seboj so začeli povezovati tudi celotna sorodna ali raznorodna omrežja (npr. Internet) ter v omrežjih integrirati prenos govora, večpredstavnostne informacije in podatkov. Poleg fiksnih so se pojavila tudi omrežja mobilnih uporabnikov.

Z vsem tem pa se je kompleksnost načrtovanja in preverjanja pravilnosti protokolov seveda le še povečevala. Zato je danes to zelo obsežno strokovno in zelo aktivno

raziskovalno področje. V tem delu pa bomo seveda lahko opozorili le na nekaj najpomembnejših in najpreprostejših problemov in podali le nekaj najosnovnejših principov in metod.

1.3 Hierarhija snovanja telekomunikacijskih protokolov in naprav

Namen telekomunikacijskih protokolov je reševanje določenih problemov pri prenosu informacije. V ta namen obstajajo različne **metode**, ki so dovolj splošne, da dovoljujejo uporabo različnih variant v praksi. Način komuniciranja, ki je tako natančno določen, da ga ni moč nápak ali dvoumno razumeti, običajno pa je tudi na nek način standardiziran, imenujemo **protokol**. Nek konkreten protokol je mogoče **izvesti (implementirati)** na različne načine, bodisi v obliki strojne (ang. hardware) ali programske (ang. software) opreme. Trdimo lahko torej, da protokol vedno uporablja neko metodo komuniciranja, izvedemo pa ga v obliki elektronskega vezja (strojne opreme) ali programa (programske opreme). Tovrstno hierarhijo snovanja komunikacijskih naprav prikazuje slika 1 3. Ker je torej protokol vedno vezan na metodo komuniciranja, vsaka metoda določa skupino protokolov. Zato izrazov »metoda« in »protokoli« ne bomo vedno strogo ločevali. Tako bomo npr. govorili o protokolih ARQ, pri tem pa ne bomo mislili na nek konkreten protokol, ampak na metodo popravljanja napak s ponavljanjem (Automatic Repeat reQuest).



Slika 1-3. Hierarhija snovanja telekomunikacijskih naprav

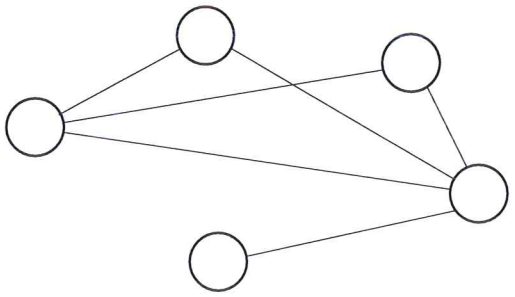
S hitrim razvojem telekomunikacij se seveda tudi protokoli izredno hitro razvijajo oziroma spreminjajo, vsekakor hitreje kot splošnejše metode komuniciranja. Kot značilen primer naj navedemo internetska protokola IP in TCP, ki sta 20 let po svojem nastanku (to pa je le polovica običajne inženirjeve delovne dobe) že prava klasika med protokoli.

Vendar pa gre tu za dva temeljna protokola internetskega komuniciranja. Življenjska doba manj pomembnih protokolov je dandanes lahko precej krajša (prva generacija mobilnih telefonskih omrežij in njim pripadajočih protokolov je npr. živela le dobrih 10 let!). Poznavanje in razumevanje metod je torej trajnejša dobrina sodobnega telekomunikacijskega inženirja kot podrobno poznavanje konkretnih protokolov. Zategadelj se bomo v tem delu (kljub naslovu knjige) osredotočili na metode komuniciranja, nekatere konkretne protokole pa bomo podrobneje opisali kot zglede, ki naj predvsem utrdijo razumevanje tvarine, hkrati pa dajo bralcu nekaj občutka za življenjsko stvarnost.

2 OSNOVNI PRINCIPI PODATKOVNIH KOMUNIKACIJ

V tem poglavju bomo definirali in opredelili nekaj sicer že znanih osnovnih pojmov, ki jih bomo potrebovali pri nadaljnji obravnavi telekomunikacijskih protokolov. V nadaljevanju teksta pa bomo nekatere izmed teh pojmov tudi še podrobneje opisali.

Izraz telekomunikacije pomeni v tehničnem smislu izmenjavo informacij na daljavo (**tele** pomeni v grščini daleč, **communicatio** pa v latinščini sporočilo). Za uspešno komuniciranje so potrebni **komunikacijski osebki** ali **komunikacijske naprave**, ki med seboj izmenjujejo informacijo, in **komunikacijsko sredstvo** (**medij**), ki povezuje komunikacijske osebke med seboj in ga pogosto imenujemo tudi **kanal**. Takšno strukturo prikazuje slika 2-1, v kateri so komunikacijski osebki prikazani s krogi, komunikacijski kanali pa s črtami. Komunikacijski osebek pogosto imenujemo tudi **entiteta** (ang. **entity**). Izmenjevanje informacije med osebki preko kanalov imenujemo **komuniciranje**, dogajanje v komunikacijskem osebku ali v celotnem sistemu pa je **telekomunikacijski proces**. Kot smo že ugotovili v uvodu, mora med komunikacijskimi osebki obstajati vsaj načelen dogovor o načinu medsebojnega komuniciranja. Sistem, ki ga sestavljajo komunikacijski osebki in kanali, imenujemo **telekomunikacijsko omrežje**. Komunikacijske osebke včasih imenujemo tudi **vozli**. Tiste vozle, ki predstavljajo izvor ali ponor informacije, ki se prenaša skozi omrežje, imenujemo tudi **terminali omrežja**, notranji vozli (ki usmerjajo informacijo v samem omrežju) pa so **posredovalni** vozli (in so lahko **komutacijski** ali **usmerjevalni** vozli).



Slika 2-2. Struktura telekomunikacijskega omrežja

Načeloma so komunikacijski osebki lahko ljudje ali stroji, med slednjimi so to seveda najpogosteje računalniki oziroma procesi, ki tečejo v njih. Komunikacijski medij oziroma kanal pa je kakršno koli sredstvo, ki omogoča prenos informacije od enega osebka do drugega. Takšna definicija komunikacijskega medija se sicer sliši precej abstraktno, kar

pa je popolnoma upravičeno. Simetrični dvovod, mikrovalovna usmerjena zveza ali optični kabel so sicer posebni primeri komunikacijskih kanalov, vendar pa je komunikacijski kanal tudi povezava med dvema telefonoma ali navidezna zveza med dvema računalnikoma, priključenima na Internet, pri čemer pa nas prav nič ne zanima, kako ta povezava fizično izgleda oziroma kakšne naprave (npr. telefonske centrale ali internetni usmerjevalniki) se nahajajo med subjektoma, ki komunicirata. V primeru, ko bo kanal predstavljala neka fizično otipljiva prenosna pot, bomo govorili o **fizičnem (realnem) kanalu**, v nasprotnem primeru pa o **navideznem (virtualnem) kanalu**. Bistveno pri tem pa je, da je fizična dolžina kanala, ki povezuje dva ali več osebkov, končna – to je od nič različna. Ena izmed temeljnih fizikalnih zakonitosti je namreč ta, da lahko informacijo prenesemo z enega mesta na drugo vedno le s končno **hitrostjo** v (katere zgornja meja je svetlobna hitrost, 3×10^8 m/s); to pomeni, da je za prenos informacije z enega kraja na drugega potreben nek končen, od nič različen čas t_p, ki ga bomo imenovali **čas propagacije**. Zaradi tega se spremembe stanja v različnih delih telekomunikacijskega sistema ne dogajajo sočasno. Pravimo, da je telekomunikacijski sistem **porazdeljen (distribuiran)** sistem, ki ga lahko opazujemo le v odvisnosti od kraja in časa. S tem pa je povezan tudi **temeljni telekomunikacijski problem**, ki ga rešujejo telekomunikacijski protokoli: ker informacija o stanju sistema ni shranjena na enem mestu, ampak je porazdeljena po sistemu, nobeden izmed osebkov, ki med seboj komunicirajo, tega stanja ne pozna v celoti. Pozna le stanje, v katerem se nahaja sam. O stanju ostalih delov porazdeljenega sistema lahko sklepa le iz informacije, ki jo v teku procesa komunikacije sprejme, seveda pa se je med prenosom te informacije lahko stanje ostalih delov sistema že spremenilo. Eden temeljnih ciljev telekomunikacijskih protokolov je omogočiti komunikacijskim osebkom čim boljše in zanesljivejše poznavanje stanja, v katerem se porazdeljeni sistem nahaja, s tem pa tudi tem lažje in pravilnejše odločanje o nadaljnjem delovanju.

Osebki, ki med seboj komunicirajo, kanali, preko katerih komunicirajo, prav tako pa tudi telekomunikacijski procesi imajo seveda določene lastnosti, ki jih je treba pri načrtovanju ali analizi telekomunikacijskega sistema upoštevati. Nekatere izmed osnovnih **značilnosti komunikacijskega kanala**, ki pa med seboj niso vse neodvisne, so naslednje. **Kapaciteta kanala** je maksimalna hitrost prenosa informacije v b/s, pri kateri

je še mogoča zanesljiva komunikacija. Za kanal je značilno tudi **število različnih vrednosti**, ki jih je mogoče prenašati skozenj. Če je teh vrednosti neskončno mnogo, govorimo o **analognem** prenosu, če pa je vrednosti končno mnogo, je prenos **digitalen**. V tem delu se bomo ukvarjali le z digitalnim prenosom, saj pri analognem prenosu protokoli skorajda niso potrebni oziroma so skrajno poenostavljeni. Digitalni prenos z dvema možnima prenašanima vrednostima imenujemo **binarni prenos**. Pri digitalnih prenosih prenašamo informacijo kot zaporedje diskretnih (najpogosteje binarnih) vrednosti, te vrednosti pa združujemo v **sporočila**. Dolžino sporočil merimo v **bitih**, to je s številom binarnih vrednosti v sporočilu, lahko pa tudi v zlogih (angleško **byte**) oziroma **oktetih**; zlog oziroma oktet je zaporedje osmih binarnih vrednosti. Enoto bit označujemo s kratico b, enoto zlog (byte) pa s kratico B. Če pa predstavlja sporočilo zaporedje alfanumeričnih in ostalih znakov (znakovno orientirano sporočilo), lahko podamo dolžino sporočila tudi s številom znakov. **Pasovna širina kanala** je širina frekvenčnega pasu, ki ga je možno prenašati skozi kanal; neposredno od pasovne širine je odvisna kapaciteta kanala. Medtem ko predstavlja kapaciteta kanala teoretično zgornjo mejo hitrosti prenosa informacije, je **nazivna hitrost prenosa** R in še celo dejanska **hitrost prenosa** r skozi kanal manjša od kapacitete. V splošnem drži, da je pri analognem prenosu primernejši pojem pasovne širine, pri digitalnem pa pojem hitrosti prenosa informacije*. Hitrost prenosa merimo v bitih na sekundo – b/s (v anglosaški literaturi se uporablja kratica bps, bits per second), lahko pa tudi v zlogih na sekundo – B/s. Hitrost prenosa pomeni število binarnih vrednosti – bitov (ali, če se tako dogovorimo, zlogov), ki jih v časovni enoti oddamo v kanal, oziroma iz njega sprejmemo. Iz teorije informacij vemo, da ena binarna vrednost ne nosi vedno informacije enega bita, ampak lahko tudi manj. Zato je **hitrost prenosa informacije** (**informacijski pretok**) strogo vzeto običajno manjša od hitrosti prenosa. Vendar pa se v tem besedilu v tovrstne podrobnosti ne bomo spuščali in bomo z obema terminoma mislili na hitrost prenosa. Včasih, ko gre za digitalni prenos preko fizičnega kanala, prenašati pa je možno več kot dve različni vrednosti (prenos ni binaren), lahko za hitrost prenosa uporabljamo tudi starejšo enoto **baud**, ki pomeni število

* Resnici na ljubo je treba povedati, da se je izraz pasovna širina uveljavil tudi na področju digitalnega prenosa, vendar tam običajno pomeni hitrost prenosa.

vrednosti na sekundo; se pa ta enota vedno manj uporablja. Pretvorba med hitrostjo r_b v baudih in hitrostjo r v b/s je podana z enačbo

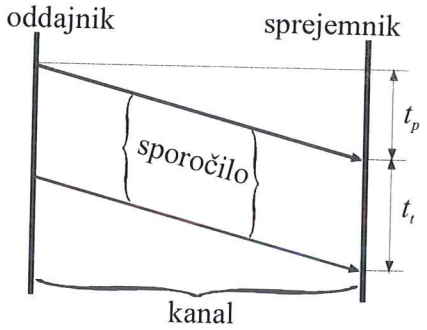
$$r = r_b \cdot \log_2 N, \tag{1}$$

kjer je N število možnih prenašanih vrednosti.

Hitrost prenosa r določa povezavo med dolžino sporočila L in časom t_t, ki je potreben za oddajanje oziroma sprejem sporočila. Če imamo torej sporočilo z dolžino L in kanal s hitrostjo prenosa informacije r, bo oddajanje sporočila trajalo čas t_t, sprejemnik pa ga bo sprejemal prav toliko časa:

$$t_t = \frac{L}{r}. \tag{2}$$

Omenili smo že, da informacija pri prenosu skozi kanal utrpi zakasnitev t_p. Potovanje sporočila z dolžino L bitov po kanalu s hitrostjo prenosa informacije r in s propagacijskim časom t_p lahko torej skiciramo tako, kot vidimo v sliki 2-3. Iz te slike vidimo, da je od začetka oddajanja sporočila na oddajni strani kanala pa do dokončnega sprejema sporočila na sprejemni strani kanala potreben čas $t_p + t_t$, kjer je čas oddajanja sporočila t_t podan z enačbo (2).



Slika 2-4. Zakasnitev sporočila

Zakasnitev sporočila v kanalu (propagacijski čas) je lahko **deterministična**, kar pomeni, da je pri prehodu istega sporočila skozi kanal vedno enaka, ali pa je **nedeterministična**, kar pomeni, da je zakasnitev pri različnih prehodih istega sporočila skozi isti kanal lahko različna; v tem primeru ima zakasnitev naključno naravo in jo lahko le statistično obravnavamo: ima določeno statistično porazdelitev, srednjo vrednost, standardni odklon itd. V splošnem lahko trdimo, da bo zakasnitev deterministična v fizičnem kanalu; ta zakasnitev je podana z enačbo

$$t_d = \frac{D}{v}, \tag{3}$$

kjer je D fizična dolžina kanala in v hitrost elektromagnetnega vala v tem kanalu; seveda pa moramo tej minimalni zakasnitvi prišteti še zakasnitve v morebitnih vmesnih napravah (npr. ojačevalnikih ali regeneratorjih). Nedeterministična pa bo zakasnitev v navideznih kanalih, saj moramo v tem primeru minimalni zakasnitvi po enačbi (3) prišteti še procesiranje sporočila in še zlasti zakasnitve v vmesnih čakalnih vrstah:

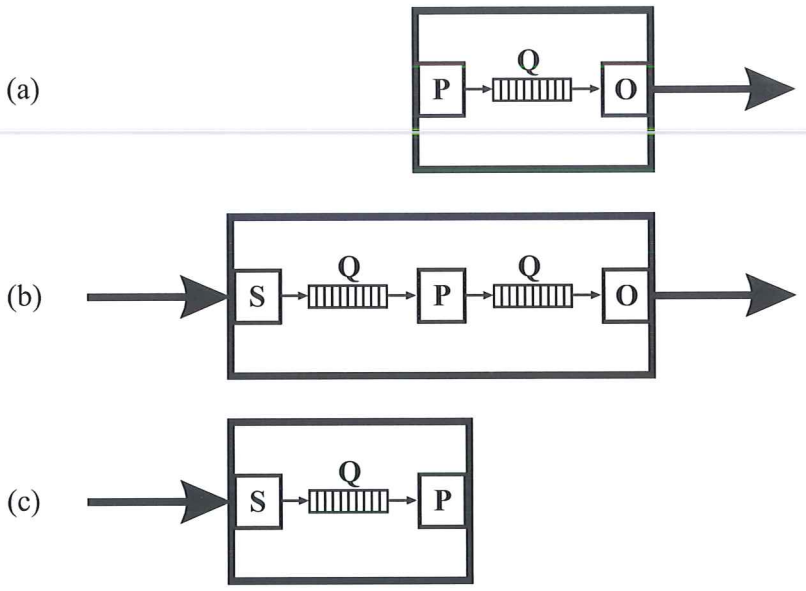
$$t_d = \frac{D}{v} + t_{proc} + t_q, \tag{4}$$

Ena izmed značilnosti kanala je tudi smer prenosa informacije. Poznamo **enosmerni (simpleksni)** kanal, **dvosmerni (dupleksni)** kanal ali **izmenični enosmerni (poldupleksni)** kanal. Medtem ko se lahko po dupleksnem kanalu prenaša informacija v obe smeri hkrati, se lahko po poldupleksnem prenaša v določenem časovnem obdobju le v eno smer, v naslednjem obdobju pa v obratno smer. Načeloma lahko dupleksni kanal vidimo kot dvojico simpleksnih kanalov. Pri digitalnem prenosu je to tudi običajna implementacija dupleksnega kanala; pri tem je pomembna izjema kanal, ki povezuje ISDN naročnika z ISDN centralo. Podobno, čeprav z določeno razliko v pomenu, pa velja tudi za procese komuniciranja med napravami. Simpleksna komunikacija je seveda mogoča tako skozi simpleksni (seveda v pravi smeri!), kot tudi skozi poldupleksni ali dupleksni kanal; slednji pa pri tem seveda ni tako izkoriščen, kot bi lahko bil. Poldupleksna komunikacija seveda ni mogoča preko simpleksnega kanala, za dupleksno komunikacijo pa je potreben dupleksni kanal.

Kot zadnjo, nikakor pa tudi ne najmanj pomembno lastnost kanala bomo omenili vernost oziroma zanesljivost prenosa. Medtem ko tu pri analognih prenosih mislimo na šum in razne vrste popačenj, se bomo pri digitalnih prenosih ukvarjali s problemi drugačne vrste. Sporočilo se v kanalu lahko pokvari, lahko se preprosto izgubi, lahko pa se celo podvoji. Razmerje pokvarjenih sporočil proti vsem oddanim sporočilom imenujemo **pogostnost napak**, ki predstavlja merilo za kvaliteto kanala. Razmerje pokvarjenih binarnih vrednosti proti vsem oddanim binarnim vrednostim imenujemo **pogostnost bitnih napak**; pogosto jo označujemo s kratico *ber* (angleško bit error rate), predstavlja pa merilo za kvaliteto fizičnega kanala. Napake se seveda pojavljajo naključno.

Naštejmo sedaj še najpomembnejše lastnosti komunikacijskih osebkov. Komunikacijska naprava lahko deluje kot **oddajnik informacije** ali kot **sprejemnik informacije**, največkrat pa igra obe vlogi hkrati. Informacijo lahko tudi oblikuje, obdeluje ali predeluje. Informacijo lahko bodisi **posreduje** iz enega kanala v drugega, ali pa predstavlja **izvor** oziroma **ponor** informacije; v zadnjih dveh primerih govorimo o ***terminalu***. Če ima posredovalna naprava en sam vhodni in en sam izhodni kanal, je to **relejna naprava**; pri več vhodnih in enem izhodnem kanalu govorimo o **multipleksiranju** (včasih tudi o **koncentraciji**), pri enem vhodnem in več izhodnih kanalih o **demultipleksiranju**, pri več vhodnih in več izhodnih kanalih pa o **usmerjanju** informacije.

Tudi za komunikacijsko napravo je značilna hitrost oddajanja oziroma sprejemanja informacije; ta hitrost mora biti usklajena med oddajno oziroma sprejemno napravo ter kanalom (sinhronizacija hitrosti). Ker pa mora osebek sporočila največkrat tudi obdelati, je ena izmed njegovih značilnosti tudi **hitrost procesiranja informacije**; seveda pa mora biti tudi ta hitrost usklajena s hitrostjo oddajanja oziroma sprejemanja. Ker te hitrosti same po sebi pogosto niso usklajene, potrebuje komunikacijska naprava še spomin, kamor lahko shranjuje sporočila, ki čakajo bodisi na procesiranje ali pa na oddajanje. Ta spomin omogoča neodvisno delovanje sprejemnika, procesorja in oddajnika. Organiziran je kot **čakalna vrsta** (ang. **queue**). Slika 2 5 podaja shematični prikaz (a) oddajne naprave, (b) relejne naprave in (c) sprejemne naprave. V tej sliki pomenijo oznake S, P, O in Q sprejemnik, procesor, oddajnik oziroma čakalno vrsto. Pri demultipleksorjih in usmerjevalnih napravah ima seveda vsak izhodni kanal svoj oddajnik, vsak oddajnik pa lahko svojo čakalno vrsto; prav tako ima lahko pri multipleksorjih in usmerjevalnih napravah vsak vhodni kanal svoj sprejemnik, vsak sprejemnik pa lahko svojo čakalno vrsto. Čakalna vrsta je implementirana s spominom, zato je razumljivo, da je njena maksimalna velikost omejena; to pa je tudi ena izmed značilnosti komunikacijske naprave. Število in razporeditev čakalnih vrst (de)multipleksorske ali usmerjevalne naprave sta odvisna od arhitekture (zasnove) te naprave.



Slika 2-5. Oddajna, relejna in sprejemna naprava

Povedali smo že, da se bomo v tem delu ukvarjali le z digitalnim prenosom, torej s prenosom sporočil. Sporočila potujejo po kanalu eno za drugim, sprejemnik pa lahko ta sporočila sprejema le, če ugotovi, kdaj se katero sporočilo začenja; to je problem **sinhronizacije sporočil**. Pri tem vsekakor ni nepomembno, ali prihajajo sporočila v enakomernih časovnih presledkih, to je sinhrono z neko uro (oziroma ob pričakovanih časih), ali pa ob bolj ali manj naključnih časih; v prvem primeru gre za **sinhroni prenos**, v drugem pa za **asinhroni prenos** sporočil.

Komunikacijski proces pomeni prenos zaporedja sporočil med dvema ali več komunikacijskimi osebki skozi telekomunikacijsko omrežje; ker pa ni nujno, da so ti osebki neposredno povezani med seboj, lahko pri taki komunikaciji v resnici sodeluje tudi še več vmesnih posredovalnih naprav. Ločimo dva različna načina komuniciranja: povezavno usmerjeno in nepovezavno usmerjeno komuniciranje. **Povezavno usmerjena komunikacija** (ang. **connection-oriented communication**) poteka tako, da se med osebkoma najprej vzpostavi **zveza** (ta je lahko realna ali navidezna), sporočila se nato pošiljajo preko te zveze, nazadnje pa se zveza še sprosti; zveza je lahko **stalna** (**permanentna**), kar pomeni, da je ni treba vzpostavljati in sproščati oziroma jo vzpostavi in sprosti operater za daljši čas. Nasprotno pa je **komutirana** zveza takšna, ki jo omrežje vzpostavi oziroma sprosti na zahtevo uporabnika. Za povezavno usmerjeno komunikacijo je značilno, da vsa sporočila, ki pripadajo isti zvezi, potujejo po isti poti skozi omrežje in

torej prihajajo do naslovnika v istem vrstnem redu, kot so bila poslana. Pri **nepovezavno usmerjeni komunikaciji** (ang. **connectionless communication**) pa vsako sporočilo povsem samostojno išče pot in potuje skozi omrežje, kar pomeni, da lahko nekatera sporočila celo prehitijo druga; vrstni red sprejetih sporočil je torej lahko različen od vrstnega reda oddanih sporočil. Sporočila, ki ne pripadajo zvezi, imenujemo **datagrami**. Pomembna je tudi povezava med povezavnim oziroma nepovezavnim načinom komuniciranja ter usmerjanjem. Pravo **usmerjanje** (ang. **routing**) je iskanje poti skozi omrežje, ki mora upoštevati topologijo in ostale lastnosti celotnega omrežja in se izvaja pri nepovezavnem komuniciranju za vsak datagram posebej, pri povezavnem načinu komuniciranja pa se usmerjanje izvede le ob vzpostavitvi zveze, med samim potekom zveze pa je postopek bistveno enostavnejši in ga imenujemo **komutacija** (ang. **switching**), izvaja pa se v vsakem vozlu posebej.

3 STANDARDI V TELEKOMUNIKACIJAH

Že v uvodu smo razmišljali o tem, da mora obstajati med subjekti, ki med seboj komunicirajo, trden dogovor o načinu in postopku komuniciranja, kakor tudi o oblikah informacijskih sporočil, ki si jih ti subjekti med seboj izmenjujejo. Temeljni namen takega dogovora je torej, da se doseže kompatibilnost med različnimi udeleženci v telekomunikacijskem procesu, oziroma kompatibilnost med različnimi napravami v telekomunikacijskem sistemu (omrežju). Če obstaja tak dogovor v pisni obliki predpisa, ki je javnosti dostopen in vsaj do neke mere ali za nekatere udeležence obvezen, ga imenujemo **standard**. Standard, ki ima pravno veljavo, imenujemo **de iure** (t.j. pravno veljavni) standard. Če pa se je nek standard uveljavil v praksi, ne da bi imel pri tem tudi pravno veljavo (vsaj ne v okolju, kjer se je uveljavil), pravimo, da je to **de facto** (t.j. dejanski) standard. Nekatere organizacije namesto standardov izdajajo **priporočila**, ki sama po sebi sicer nimajo pravne veljave v smislu dokumenta, ki ga je obvezno treba upoštevati, vendar pa se lahko kljub temu uveljavijo in uporabljajo kot standardi, pogosto pa kasneje tudi dobijo pravno veljavo. Medtem ko imajo nekatere izmed organizacij, ki se ukvarjajo s standardizacijo, za to mednarodna ali državna pooblastila, je naloga ostalih organizacij ta, da predlagajo svoje standarde v sprejem organizacijam iz prvih dveh skupin.

3.1 Vrste standardov in organizacije za standardizacijo

Glede na institucijo, ki nek standard sprejme oziroma mu da pravno veljavo, ločimo

- mednarodne standarde
- regionalne standarde
- nacionalne standarde
- strokovne standarde
- interne standarde

Mednarodne standarde sprejemajo mednarodne organizacije za standardizacijo, zato imajo standardi, ki jih izdajajo, mednarodno veljavo. Tabela 1 3 našteva štiri najpomembnejše mednarodne organizacije za standardizacijo, ki pokrivajo tudi področje telekomunikacij, za lažje iskanje dodatnih informacij pa so dodani tudi naslovi njihovih spletnih strani.

Tabela 1. Mednarodne organizacije za standardizacijo

organizacija	kratica	www naslov
International Organisation for Standardisation	ISO	http://www.iso.ch/
International Telecommunication Union	ITU	http://www.itu.ch/
International Electrotechnical Commission	IEC	http://www.iec.ch/
Internet Society	ISOC	http://www.isoc.org/

Regionalni standardi imajo veljavo v določeni regiji oziroma delu sveta (glej tabelo 2 4).

Tabela 2. Regionalne organizacije za standardizacijo

regija	organizacija	kratica	www naslov
Evropa	Comité Européen de Normalisation	CEN	http://www.cenorm.be/
Evropa	Comité Européen de Normalisation Electrotechnique	CENELEC	http://www.cenelec.be/
Evropa	European Telecommunications Standards Institute	ETSI	http://www.etsi.fr/
Amerika	Comicion Panamericana de Normas Tecnicas	COPANT	http://www.copant.org
Azija in Pacifik	ASEAN Consultative Committee for Standards and Quality	ACCSQ	http://www.aseansec.org/accsq/sqmain.htm

Nacionalni standardi so tisti, ki naj bi imeli nacionalni pomen za neko državo, njihova veljavnost pa je omejena na to državo. Razvija in sprejema jih lahko ena ali več organizacij, ki imajo bodisi ustrezno pooblastilo državnih organov za tako dejavnost, ali pa imajo v okviru države dovolj moči in ugleda. Stvar državne zakonodaje pa je, kakšen status imajo ti standardi v posamezni državi; lahko so obvezni, priporočeni, ali pa je njihova uporaba prostovoljna. Seveda pa imajo produkti, ki se skladajo s standardi, še zlasti s takimi, ki so v stroki širše sprejeti, določene prednosti tržne narave. V mnogih državah (npr. v mnogih evropskih) obstaja ena organizacija za standardizacijo, ki jo nadzira ali celo tudi financira država. Spet v drugih državah pa je takih organizacij več (npr. v ZDA). V tabeli 2 5 najdemo nekaj pomembnejših nacionalnih organizacij za standardizacijo, skupaj z ustaljenimi kraticami in naslovi njihovih spletnih strani.

Tabela 3. Nacionalne organizacije za standardizacijo

država	organizacija	kratica	www naslov
SI	Slovenski inštitut za standardizacijo	SIST	http://www.sist.si/
CA	Canadian Standards Association	CSA	http:www.csa.ca
CH	Association Suisse de Normalisation	SNV	http://www.snv.ch/
D	Deutsches Institut für Normung e.V.	DIN	http://www.din.de/
F	Association française de normalisation	AFNOR	http://www.afnor.fr/
GB	British Standards Institution	BSI	http://www.bsi.org.uk/
JP	Japanise Industrial Standards Committee	JISC	http://www.jisc.org/
USA	American National Standards Institute	ANSI	http://web.ansi.org/

Strokovne standarde razvijajo in sprejemajo večja strokovna združenja in institucije, ki so največkrat vezane na ožje strokovno področje. Namen sprejemanja takih standardov ni le njihova omejena uporaba v okviru članov združenja, ampak predvsem tudi zastopanje interesov članov združenja pri razvoju in sprejemanju nacionalnih in še zlasti mednarodnih standardov. Nacionalni in mednarodni standardi namreč pogosto nastanejo kar s prirejanjem ali celo povzemanjem strokovnih standardov. Nekaj tovrstnih organizacij je naštetih v tabeli 3.

Tabela 4. Strokovna združenja za standardizacijo

organizacija	kratica	www naslov
ATM Forum	-	http://www.atmforum.com/
Electronic Industries Association	EIA	http://www.eia.org/
Standardizing Information and Communication Systems (European Computer Manufacturers Association)	ECMA	http://www.ecma.ch/
Frame Relay Forum	-	http://www.frforum.com/
Institute of Electrical and Electronics Engineers	IEEE	http://www.ieee.org/
3rd Generation Partnership Project	3GPP	http://www.3gpp.org
World Wide Web Consortium	W3C	http://www.w3.org/

Večje industrijske družbe pogosto za svoje potrebe razvijajo standarde, ki imajo omejeno veljavo in omogočajo kompatibilnost le med produkti te družbe. To so **interni standardi**. Na področju telekomunikacij so znani in imajo precejšen ugled standardi ameriške družbe Bell Communications Research (Bellcore).

3.2 Razvoj in sprejemanje standardov

Pri razvoju in sprejemanju mednarodnih, nacionalnih in industrijskih standardov obstaja več možnih postopkov.

Mednarodne standarde sprejemajo mednarodne organizacije, katerih člani prihajajo iz različnih držav, ki so različno razvite in imajo različne interese. Ker je lahko mednarodni

standard uspešen le, če ga sprejme in upošteva čim več udeležencev v svetu telekomunikacij, mora biti sprejet, če že ne s popolnim soglasjem, pa vsaj z veliko večino sodelujočih članov organizacije. Ker pa prihaja pri tem pogosto do navzkrižnih interesov, se lahko postopek sprejemanja standarda zavleče na več let. Žal pa so posledice iskanja soglasij vidne tudi na kvaliteti standardov, saj ti večkrat niso le plod razumne inženirske presoje, ampak so tudi plod raznih kompromisov. Včasih se mednarodna organizacija za standardizacijo loti razvoja povsem novih standardov. Večkrat pa služi kot osnova za sprejem mednarodnega standarda tudi že obstoječi nacionalni ali industrijski standard, če se je ta že uspel uveljaviti v mednarodnem prostoru kot de facto standard, ali pa ga nekateri subjekti, ki so dovolj močni, uspejo uveljaviti kot mednarodni standard.

Razvoj standarda lahko izhaja iz teoretičnih spoznanj stroke, seveda pa mora biti standard pred sprejetjem dodobra preverjen in preizkušen. Lahko pa standard tudi izhaja iz že delujoče implementacije, ki jo na ta način le še "uzakonijo". Medtem ko je prvi način v precejšnji meri značilen za področje klasičnih telekomunikacij (to je za področje javnih telekomunikacijskih omrežij) in ga na področju mednarodne standardizacije uporabljajo v organizaciji ITU, drugo možnost velikokrat uporabljajo pri standardizaciji računalniških komunikacij, zlasti v organizaciji za standardizacijo Interneta ISOC. Prvi način da rezultate, ki so bolj nevtralni glede možnih implementacij, drugi način sprejemanja standardov pa je hitrejši.

V nadaljnjih razdelkih si bomo podrobneje ogledali nekaj najpomembnejših organizacij za standardizacijo.

3.3 International Organisation for Standardisation - ISO

Mednarodna organizacija za standardizacijo ISO (**International Organisation for Standardisation, Organisation Internationale de Normalisation**) je bila ustanovljena leta 1947 kot svetovna zveza nacionalnih organizacij za standardizacijo. Vsako državo zastopa po ena organizacija. Izdaja **mednarodne standarde (International Standards)**, ki urejajo praktično vsa področja tehnike, od vijakov pa do kakovosti proizvodov. Za področje telekomunikacij skrbi skupaj z Mednarodno elektrotehniško komisijo IEC. Njen sedež je v Ženevi, uradni jeziki pa so angleščina, francoščina in ruščina. Ime ISO razlagajo kot izpeljanko iz grške besede **ΙΣΟΣ** (isos=enak).

Za razvoj in sprejemanje standardov skrbijo komiteji, podkomiteji in delovne skupine. Standard najprej definirajo kot "Delovni osnutek", ki nato napreduje v "Osnutek komiteja". Ko članice glede tega osnutka dosežejo potrebno stopnjo soglasja, izdajo "Osnutek mednarodnega standarda". Ta pa po končnem glasovanju postane "Mednarodni standard", če si pridobi vsaj dve tretjini glasov aktivnih članic in če ni več kot ena četrtina vseh članic argumentirano proti.

3.4 International Telecommunication Union - ITU

To organizacijo so ustanovili že daljnega leta 1865 pod imenom International Telegraph Union. Sedanje ime je dobila leta 1934, od leta 1947 deluje pod okriljem Združenih narodov, zadnjo večjo reformo pa je doživela leta 1993. Tudi ta ima sedež v Ženevi. Članice Mednarodne zveze za telekomunikacije so lahko države ali pa tudi druge organizacije, kot npr. večji operaterji telekomunikacijskih omrežij. ITU je torej nekakšna razširjena medvladna organizacija.

Do leta 1993 je imela Mednarodna zveza za telekomunikacije dva posvetovalna komiteja. Mednarodni posvetovalni komite za telegrafijo in telefonijo (**Comité Consultatif International Télégraphique et Téléphonique - CCITT**) je skrbel za standardizacijo telekomunikacij po vodih, Mednarodni posvetovalni komite za radio (**Comité Consultatif International Radio - CCIR**) pa je skrbel za standardizacijo brezžičnih telekomunikacij. Po reformi je delo prvega in precejšen del dela drugega komiteja prevzel Sektor za telekomunikacije ITU (**International Telecommunication Union - Telecommunications Sector, ITU-T**), z upravljanjem radijskega spektra pa se ukvarja Sektor za radiokomunikacije ITU (**International Telecommunication Union - Radiocommunications Sector, ITU-R**).

ITU-T ne izdaja standardov, temveč **priporočila** (ang. **recommendation**). Vendar pa imajo ta priporočila tolikšen vpliv in ugled, da se jih proizvajalci v veliki meri držijo, veliko teh priporočil pa je tudi uzakonjenih bodisi v obliki mednarodnih standardov ISO ali nacionalnih standardov. Mnogo teh priporočil je še znanih pod imenom CCITT.

Delo ITU-T poteka v konferencah, študijskih skupinah in delovnih telesih. Priporočila so razdeljena v serije, ki so označene s črkami angleške abecede (razen W), vsaka serija pa je posvečena določenemu področju telekomunikacij. Če kakšno priporočilo smiselno spada v dve različni seriji, ima lahko dva naziva. Naziv priporočila ima obliko ITU-T

S.nnn, kjer je *S* oznaka serije (velika črka), *nnn* pa oznaka priporočila v seriji (eno-, dvo- ali tromestno število). Primer naziva priporočila je ITU-T X.25 (po starem CCITT X.25). Tabela 2 podaja oznake in naslove serij (v originalu - angleščini).

Tabela 5. Serije priporočil ITU-T

serija	vsebina serije
A	Organization of the work of the ITU-T
B	Means of expression: definitions, symbols, classification
C	General telecommunication statistics
D	General tariff principles
E	Overall network operation, telephone service, service operation and human factors
F	Non-telephone telecommunication services
G	Transmission systems and media, digital systems and networks
H	Audiovisual and multimedia systems
I	Integrated services digital network
J	Transmission of television, sound programme and other multimedia signals
K	Protection against interference
L	Construction, installation and protection of cables and other elements of outside plant
M	Maintenance: international transmission systems, telephone circuits, telegraphy, facsimile and leased circuits
N	Maintenance: international sound programme and television transmission circuits
O	Specifications of measuring equipment
P	Telephone transmission quality, telephone installations, local line networks
Q	Switching and signalling
R	Telegraph transmission
S	Telegraph services terminal equipment
T	Terminals for telematic services
U	Telegraph switching
V	Data communication over the telephone network
X	Data networks and open system communication
Y	Global information infrastructure
Z	Programming languages

V seriji je lahko več skupin priporočil, priporočila v skupini pa so med seboj vsebinsko povezana. Organiziranje priporočil v skupine ni formalno in se zato tudi ne kaže v nazivu priporočil. Tabela 6 kaže skupine priporočil v seriji Z.

Tabela 6. Skupine priporočil ITU-T serije Z

Priporočila v skupini	Vsebina skupine
Z.100 - Z.106	Specification and Description Language (SDL)
Z.110 - Z.110	Applicability of formal Description Techniques
Z.120 - Z.120	Message Sequence Chart
Z.200 - Z.200	ITU-T High Level Language (CHILL)
Z.301 - Z.302	General principles
Z.311 - Z.317	Basic syntax and dialogue procedures
Z.321 - Z.323	Extended MML for visual display terminals
Z.331 - Z.360	Specification of the man-machine interface

Z.400 - Z.500	Miscellaneous

V tabeli 7 pa najdemo priporočila v prvi od skupin serije Z, Specification and Description Language (SDL).

Tabela 7. Priporočila ITU-T v prvi skupini serije Z

Priporočilo	Vsebina priporočila
Z.100	CCITT Specification and description language (SDL)
Z.105	SDL combined with ASN.1 (SDL/ASN.1)
Z.106	Common interchange format for SDL

3.5 Internet Society - ISOC

Proces standardizacije je bil na področju računalništva in računalniških komunikacij že po tradiciji manj formalističen kot na področju klasičnih telekomunikacij. To se kaže in se je zlasti še kazalo v zgodovini razvoja Interneta. Krovna organizacija, ki sedaj skrbi za standardizacijo Interneta, je Internet Society, s tehničnimi vprašanji pa se zlasti ukvarja njena problemska skupina za Internet - **IETF (Internet Engineering Task Force)**, ki ima za preučevanje raznih problemov celo vrsto delovnih skupin, razdeljenih po funkcionalnih področjih. Nobeno od teh teles nima strogo formalnega članstva, v njih lahko sodeluje kdorkoli, ki ga problematika zanima; dokumenti so na vpogled komurkoli, vsakdo lahko daje nanje tudi pripombe. Druga bistvena značilnost razvoja standardov za Internet pa je, da mora biti vsak predlog, preden je sprejet, implementiran in praktično preizkušen. Tudi predlog standarda za Internet gre skozi več faz: najprej je to "predlagani standard", nato "osnutek standarda" in nazadnje "standard". Če nek standard postane zastarel in ni več potreben, postane "zgodovinski standard". Internetski standardi nosijo oznako **RFC (Request For Comment)** - saj njihovi avtorji in IETF v fazi razvoja standarda pričakujejo komentarje strokovne javnosti.

Potek standardizacije je torej v ISOC nekoliko drugačen kot v ISO in ITU-T; po eni strani je manj formalističen, po drugi strani pa povezan in preizkušen z implementacijo. Je tudi precej hitrejši. Seveda pa ISOC v današnjih časih, ko postaja meja med klasičnimi in računalniškimi telekomunikacijami vedno bolj zabrisana, tudi sodeluje z ISO, ITU-T in drugimi organizacijami za standardizacijo.

3.6 Nekaj ostalih organizacij za standardizacijo

S standardizacijo se ukvarjajo številna strokovna, profesionalna in industrijska združenja z namenom, da pospešujejo razvoj stroke, uvajajo nove standarde s svojih ožjih področij in pospešijo sicer počasno delo velikih in okornih mednarodnih organizacij. Standarde teh organizacij pogosto sprejmejo tudi za nacionalne ali mednarodne standarde, bodisi take, kot so, ali pa z manjšimi spremembami. Pri tem prihaja pogosto tudi do neskladja med raznimi variantami standardov. V tem razdelku bomo omenili le nekaj izmed teh organizacij, ki igrajo na področju telekomunikacij pomembno vlogo.

ATM Forum in **Frame Relay Forum** sta organizaciji, ki se ukvarjata z razvojem, usklajevanjem in pospeševanjem standardizacije na področjih prenosa vrste ATM oziroma Frame Relay. **Electronic Industries Association (EIA)** je ameriško združenje, ki se ukvarja tudi s standardizacijo elektronske opreme. Brez dvoma je najbolj znan in kjub častitljivi starosti preko 40 let še vedno uporabljan standard te organizacije RS-232C, ki mu je skoraj identično tudi mednarodno priporočilo ITU-T V.24, širše pa je znan kot serijski vmesnik (npr. osebnega računalnika). **ECMA** je evropsko združenje za standardizacijo informacijskih in telekomunikacijskih sistemov. Razvilo je precej standardov, ki so jih kasneje povzele evropske in mednarodne organizacije. Kratica ECMA izhaja iz starega imena **European Computer Manufacturers Association**. **European Telecommunications Standards Insitute (ETSI)** je ustanovila Evropska unija zato, da skrbi za standardizacijo na področju telekomunikacij v Evropi. V ETSI sodelujejo vlade, operaterji, industrija, raziskovalci in uporabniki. Trenutno je ETSI morda še posebej zanimiv zaradi standardizacije mobilnih telekomunikacijskih omrežij GSM. Standardizacijo mobilnih omrežij 2,5. in 3. generacije (GPRS oziroma UMTS pa je prevzela organizacija **3GPP (3rd Generation Partnetship Project)**. **Institute of Electrical and Electronics Engineers (IEEE)** je strokovna organizacija, ki se poleg izdajanja strokovnih revij in knjig ukvarja tudi s standardizacijo. Na področju standardizacije je zlasti znana po seriji standardov 802 za lokalna omrežja, ki jih je privzela tudi ISO. **National Institute of Standards and Technology (NIST)**, prej National Bureau of Standards (NBS), se ukvarja s standardizacijo za potrebe ameriške vlade, del dejavnosti pa posveča tudi področju telekomunikacij. **World Wide Web Consortium (W3C)** je združenje, ki se ukvarja z razvojem in standardizacijo Svetovnega

spleta (World Wide Web - WWW), vključno s standardizacijo jezikov za opis spletnih dokumentov.

3.7 Slovenski inštitut za standardizacijo - SIST

SIST je pristojen za področja standardizacije v republiki Sloveniji.

Inštitut je član vseh pomembnih mednarodnih in evropskih organizacij za standardizacijo. Njegove naloge so:

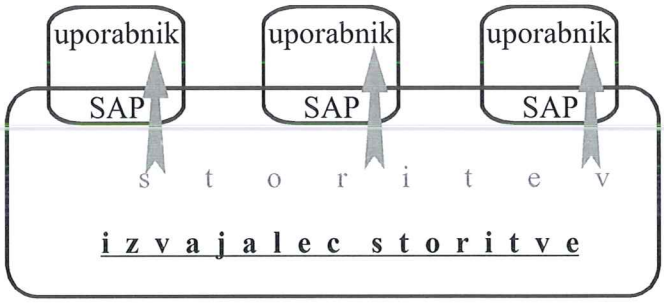
- vzpostavitev in delovanje nacionalnih služb za standardizacijo
- posredovanje mednarodnih, evropskih in ostalih standardov
- prevzemanje in izdajanje mednarodnih standardov kot slovenskih dokumentov
- izdajanje slovenskih standardov
- zastopanje interesov Slovenije v mednarodnih in regionalnih organizacijah za standardizacijo
- sodelovanje z drugimi organizacijami za standardizacijo

4 STORITVE IN PROTOKOLI

V tem poglavju si bomo ogledali nekaj osnovnih pojmov, ki so povezani tako s teorijo, kot tudi s prakso protokolov. To bosta najprej storitev in protokol, ki v nekem smislu predstavljata osnovna elementa opisa delovanja telekomunikacijskega sistema, z njima pa bomo v naslednjem poglavju zgradili še protokolni sklad, ki predstavlja strukturo delovanja telekomunikacijskega sistema.

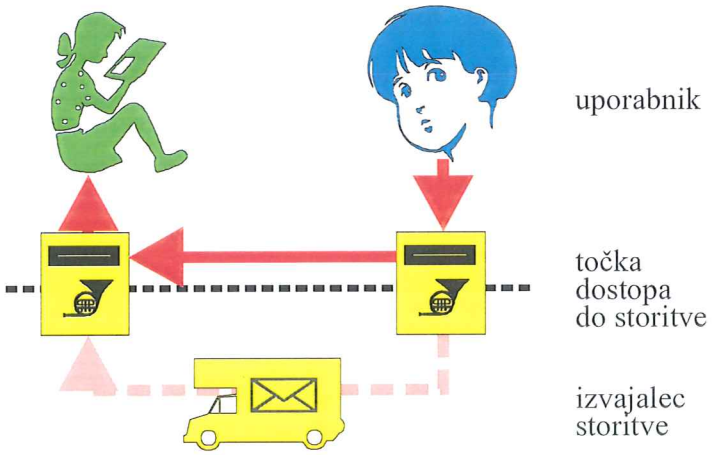
4.1 Storitev

Telekomunikacijski sistem seveda ni sam sebi namen; to, kar dela, vedno dela za nekega **uporabnika**, za katerega pomeni opravljeno delo **storitev** (angleško **service**). Uporabnik izvršitev take storitve zahteva od sistema in mu jo običajno tudi na tak ali drugačen način plača. Seveda pa mora uporabnik, preden določeno storitev naroči, vedeti, kaj bo pri tem dobil; storitev mora torej imeti svojo **specifikacijo**. Uporabnika zanima le specifikacija storitve, ki mu jo telekomunikacijski sistem ali njegov sestavni del nudi, ne pa tudi, na kakšen način bo sistem to storitev izvršil. Delovanje telekomunikacijskega sistema si torej lahko abstraktno predstavljamo kot storitev in množico (vsaj dveh) uporabnikov, ki imajo do te storitve dostop. Pri takšnem prikazu si seveda lahko tudi predstavljamo, da ima uporabnik dostop do storitve v neki točki, ki jo bomo imenovali **točka dostopa do storitve**; ker se za točko dostopa do storitve v angleščini uporablja izraz **Service Access Point**, jo pogosto označimo kar s kratico **SAP**. Z drugimi besedami bi lahko tudi rekli, da predstavlja točka dostopa do storitve stik med uporabnikom in storitvijo. Sistem, ki uporabniku nudi storitev, imenujemo **izvajalec storitve** ali tudi **ponudnik storitve** (angleško **service provider**). Slika 1 3 nam predstavo, ki smo si jo pravkar ustvarili, podaja še na grafičen način. Vendar pa se je treba zavedati, da je točka dostopa do storitve povsem konceptualen pojem, ki mu v realnem komunikacijskem sistemu pogosto ne moremo najti nekega fizičnega ekvivalenta. Tako je na primer izvajalec storitve lahko izveden kot nek proces, ki teče v komunikacijskem računalniku, točko dostopa do storitve pa predstavlja klic ustrezne procedure.



Slika 4-1. Uporabnik in izvajalec storitve

Odnos med uporabnikom in storitvijo si bomo morda lažje predstavljali z zgledom iz vsakdanjega življenja, ki ga ilustriramo na sliki 2 4. Vzemimo, da želi oseba A poslati osebi B pismo. Ustrezno storitev nudi Pošta, osebi A in B sta torej uporabnika poštne storitve, dostopna točka do te storitve (v tem primeru fizično otipljiva) pa je pri uporabniku A okence poštnega urada, pri uporabniku B pa njegov poštni nabiralnik. Za uporabnika A, ki je storitev zahteval, je pomembno, da bo uporabnik B pismo dobil, ni pa njegova stvar, ali bo to pismo potovalo z vlakom, tovornjakom ali poštno kočijo. Lahko pa seveda specifikacija poštne storitve predpisuje, da mora dobiti naslovnik pismo v roku 24 ur, da gre za pospešeno pošto, da naslovniku pismo osebno vročijo, ali pa da je pošiljatelj obveščen, kdaj je naslovnik pismo dobil. Skratka, Pošta lahko uporabnikom nudi storitve z različnimi lastnostmi in torej tudi z različnimi specifikacijami. (Storitev »letalska pošta«, ki nam jo nudi Pošta Slovenije in tudi marsikatera druga poštna organizacija, žal ni specificirana v tem duhu, njeno nesmiselnost pa kaže tudi dejstvo, da pride navadna pošta iz Ljubljane na Dunaj prej kot letalska!)



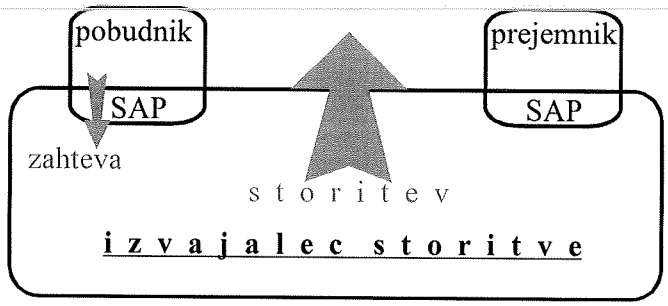
Slika 4-2. Storitev pošte

Izvajalec telekomunikacijskih storitev lahko torej nudi uporabnikom različne vrste storitev, ki se med seboj razlikujejo po svojih splošnih in posebnih lastnostih. Najosnovnejše, kar storitev nudi uporabniku, je seveda **prenos sporočil(a)** drugemu uporabniku (ali drugim uporabnikom). S stališča izvajalca storitve takšno sporočilo pogosto imenujemo **storitvena podatkovna enota** in ga označimo s kratico **SDU** (angleško **Service Data Unit**), lahko pa mu preprosteje rečemo kar **uporabniško sporočilo**. Ta osnovna storitev pa ima lahko različne posebne lastnosti. Ponudnik storitve lahko garantira, da bodo sporočila **zagotovo prišla** do naslovnika in to **brez napak** (kljub temu, da na realnih prenosnih poteh prihaja do napak) in da bodo sporočila prihajala k naslovniku v istem **vrstnem redu**, kot jih je pošiljatelj oddal; (vse to pa lahko garantirajo le tiste storitve, ki temeljijo na povezavno orientiranem prenosu). Nekatere storitve prenosa podatkov tudi omogočajo razlikovanje med sporočili različnih **prednosti** (**prioritet**), kar pomeni, da imajo uporabniška sporočila z višjo prednostjo možnost priti hitreje do naslovnika kot uporabniška sporočila z nižjo prednostjo. Zelo pomembna lastnost storitve prenosa podatkov pa je tudi zagotavljanje varnosti prenosa.

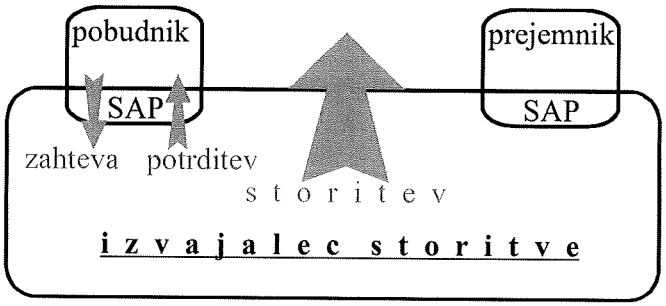
Storitve, ki temeljijo na povezavno orientirani komunikaciji, nudijo še možnost **upravljanja navidezne zveze**. Sem vsekakor najprej spadata **vzpostavitev** oziroma **sprostitev zveze**. Poleg tega pa je lahko možno še **spreminjanje parametrov zveze**, **ponovno vzpostavljanje** (**resetiranje**) **zveze**, do česar lahko pride ob večjih, nepopravljivih napakah, in medsebojno **obveščanje o stanju uporabnikov**.

Izvajanje telekomunikacijske storitve vedno sproži eden izmed uporabnikov. Tega uporabnika običajno imenujemo **pobudnik** (ang. **initiator**) **storitve**, njegov komunikacijski partner pa je **prejemnik** (ang. **recipient**) **storitve**. Pobudnik storitve je torej tisti uporabnik, ki izvajalcu storitve poda zahtevo po storitvi. Ker pa je telekomunikacijski sistem porazdeljen sistem, v katerem, kot smo že ugotovili, nobeden izmed udeležencev ne more popolnoma zanesljivo poznati stanja celotnega sistema (ve le to, o čemer je eksplicitno obveščen), v fizikalnem sistemu pa lahko tudi prihaja do napak, se ne razume samo po sebi, da bo storitev, ki jo je uporabnik zahteval, tudi uspešno izvedena. Pomembna možnost, ki jo lahko storitev nudi uporabniku, običajno pobudniku storitve, je torej potrdilo o (uspešno ali neuspešno) opravljeni storitvi; storitev, ki to možnost nudi, imenujemo **storitev s potrditvijo** (angleško **confirmed service**), tisto, ki

te možnosti ne nudi, pa **storitev brez potrditve** (angleško **unconfirmed service**). Razliko med storitvijo brez potrditve in tisto s potrditvijo ilustrirata sliki 4 3 in 1 3.



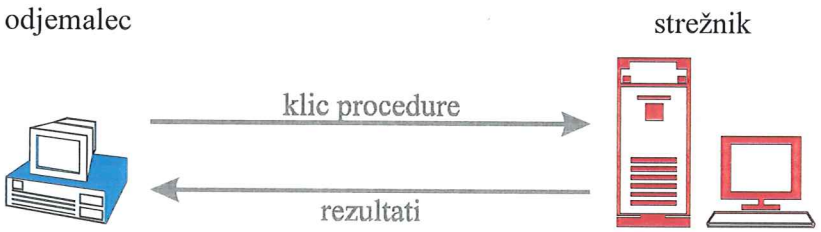
Slika 4-3. Storitev brez potrditve



Slika 4-4. Storitev s potrditvijo

Medtem ko nekatere storitve omogočajo **simetričen (uravnotežen) način komuniciranja** med uporabnikoma, kar pomeni, da sta oba udeležena subjekta enakopravna med seboj in da torej imata enake pravice in dolžnosti, nam druge storitve nudijo **asimetričen (neuravnotežen) način komuniciranja**. Poseben razred asimetričnih storitev omogoča tudi komunikacijo med uporabnikoma, od katerih je eden **odjemalec** (ang. **client**), drugi pa **strežnik** (ang. **server**), kar pomeni, da drugi nudi s posredovanjem telekomunikacijskega sistema prvemu določene storitve. **Komunikacija tipa odjemalec-strežnik** je v nekaterih vrstah omrežij (zlasti še računalniških) zelo pomembna. Še bolj specializirana storitev vrste odjemalec-strežnik pa je storitev **izvajanja operacij na oddaljenem mestu** (angleško **remote procedure call**, **RPC**), kjer odjemalec preko telekomunikacijskega omrežja zahteva izvajanje neke operacije na oddaljenem strežniku, le-ta pa mu po končani operaciji pošlje rezultate ali poročilo o izvedeni proceduri. Sliki 4 5 in 4 6 ponazarjata oba načina nesimetričnega komuniciranja.

Slika 4-5. Komunikacija odjemalec - strežnik



Slika 4-6. Izvajanje operacije na oddaljenem mestu

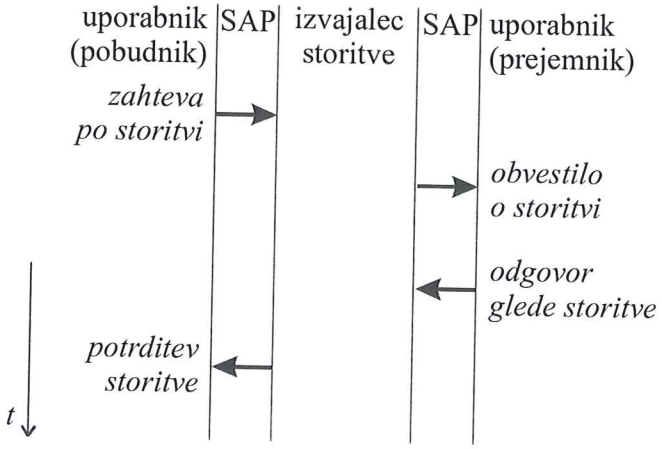
V splošnem ima lahko do iste storitve dostop več uporabnikov. Drugo vprašanje pa je, komunikacijo med kolikimi uporabniki storitev omogoča. Najpreprostejši primer je ta, da lahko med seboj komunicirata točno dva uporabnika; storitev, ki to omogoča, imenujemo **storitev individualne komunikacije** ali tudi storitev tipa **točka-točka** (ang. **point-to-point service**). Takšna je lahko npr. komunikacija med dvema telefonskima naročnikoma. Drugo možnost predstavlja storitev tipa **eden-vsem**; to je storitev **razpršenega prenosa** oziroma storitev **univerzalne komunikacije** (ang. **broadcast**); tu oddajo enega izmed uporabnikov sprejemajo vsi. Primer take komunikacije je pošiljanje obvestil nadzorne postaje v lokalnem omrežju vsem ostalim postajam. Naslednja možnost pa je storitev tipa **eden-mnogim**; imenujemo jo storitev **razdeljevanja sporočil** oziroma storitev **skupinske komunikacije** (ang. **multicast**), kjer oddaja en uporabnik, sprejemnike pa predstavlja neka podmnožica množice ostalih uporabnikov. Primer je pošiljanje obvestil nadzorne postaje le določeni podskupini ostalih postaj. S temi tremi tipi storitev so seveda povezane tri vrste naslavljanja. Poleg **individualnih** (**unicast**) naslovov se lahko uporabljajo tudi **univerzalni** (**broadcast**) in **skupinski** (**multicast**) naslovi.

4.2 Komunikacija med uporabniki, interakcija med uporabniki in ponudnikom storitve

Kanal med obema uporabnikoma je navidezen, kar pomeni, da uporabnika ne komunicirata med seboj neposredno, ampak s posredovanjem izvajalca storitve. Če želi uporabnik - pobudnik zahtevati neko storitev, v katero naj bo vpleten tudi drugi uporabnik - prejemnik storitve (običajno sta pri vsaki storitvi telekomunikacijskega omrežja udeležena vsaj dva uporabnika), mora torej priti med obema uporabnikoma in izvajalcem storitve SP do določene interakcije. Najprej mora pobudnik od izvajalca storitev zahtevati; pri tem mora seveda specificirati, kakšno storitev sploh zahteva (npr. prenos sporočila) ter podati morebitne parametre storitve (npr. samo sporočilo, ki s tem postane uporabniško sporočilo oziroma storitvena podatkovna enota SDU, in ime oziroma naslov uporabnika, ki mu je sporočilo namenjeno); ta del interakcije poteka preko točke dostopa do storitve za uporabnika - pobudnika. Nato bo izvajalec storitve obvestil uporabnika - prejemnika o zahtevani oziroma predlagani storitvi. Načeloma lahko prejemnik to storitev sprejme ali pa jo zavrne; v prvem primeru je storitev uspela, v drugem pa ne. Ta odgovor bo prejemnik sporočil izvajalcu storitve. Vzrok za neuspeh storitve pa lahko leži tudi v samem izvajanju storitve – v tem primeru obvestilo do prejemnika morda sploh ne bo prišlo. Če gre za storitev s potrditvijo, bo izvajalec uporabnika - pobudnika o izidu storitve obvestil s potrditvijo (ki je kajpak lahko tudi negativna).

Vsi ti dogodki potekajo v določenem časovnem zaporedju; tega lahko grafično predstavimo v **časovni shemi**, ki jo kaže slika 2 4. V tej shemi horizontala predstavlja prostor, vertikala pa čas. Takšna predstava pravzaprav ni čisto točna; medtem ko izvajalec storitve dejansko ima krajevno razsežnost, je točka dostopa do storitve nima. Zaenkrat bomo torej predpostavili, da interakcija med uporabnikom in izvajalcem storitve nima časovne razsežnosti (se zgodi v trenutku), zaradi česar smo v sliki 2 4 to interakcijo predstavili z vodoravno puščico. Časovnih shem pogosto ne rišemo v pravem časovnem merilu, saj so razmerja časovnih presledkov pogosto prevelika ali premajhna za praktično risanje. Pogosto pa nas v časovni shemi zanima predvsem zaporedje dogodkov, ne pa časovni razmaki med njimi. Med zahtevo po storitvi in obvestilom o storitvi, prav tako pa tudi med odgovorom glede storitve in potrditvijo storitve, pride očitno do neke

zakasnitve. Zavedati se je namreč treba, da potrebuje izvajalec storitve za svoje delo nek od nič različen čas; med drugim sta oba uporabnika med seboj fizično oddaljena, zato je sestavni del izvajalca storitve tudi kanal z od nič različno dolžino, v katerem pride do zakasnitve. Do zakasnitve pa pride tudi med obvestilom o storitvi in odgovorom glede storitve zaradi obdelave oziroma odločanja, ki je potrebno pri samem uporabniku.



Slika 4-7. Časovna shema interakcije med uporabnikoma in izvajalcem storitve

Štirje tipi interakcije med uporabnikom in izvajalcem storitve, ki smo jih pravkar našteli, zadoščajo za opis vseh možnih interakcij med uporabniki in izvajalci storitev. Zato te štiri tipe imenujemo **osnovni tipi interakcije** oziroma **primitivi** (ang. **primitives**). Zanje so se uveljavili standardni izrazi, običajno uporabljamo kar angleške. Primitivi so torej:

- *zahteva – request* (`req`)
- *obvestilo – indication* (`ind`)
- *odgovor – response* (`resp`)
- *potrditev – confirm* (`conf`)

Običajno uporabljamo primitive v povezavi s specifikacijo, kaj zahtevamo, o čem obveščamo, glede česa odgovarjamo oziroma kaj potrjujemo; zato bomo primitive pisali v obliki *spec.primitiv*(*parametri*), kjer bo *spec* specifikacija storitve, ki jo zahtevamo, o kateri obveščamo, na katero odgovarjamo oziroma ki jo potrjujemo, *primitiv* vrsta primitiva (`req`, `ind`, `resp` ali `conf`), *parametri* pa dodatno pojasnjujejo ali dopolnjujejo primitiv, saj nosijo s seboj podatke, ki so za izvajanje storitve nujno potrebni (npr. naslov prejemnika, podrobnejša specifikacija storitve, uporabniški

podatki...). Imamo lahko različne vrste zahtev: `data.req` je zahteva po prenosu sporočila, `connect.req` ali `establish.req` zahteva po vzpostavitvi zveze, `disconnect.req` ali `release.req` zahteva po njeni sprostitvi itd.

Časovno gledano se primitivi izvajajo v takšnem vrstnem redu, v kakršnem smo jih omenili. Primitiva `req` in `resp` vedno generira uporabnik in sta namenjena izvajalcu storitve, medtem ko primitiva `ind` in `conf` generira izvajalec storitve, namenjena pa sta uporabniku. Uporabnik, ki s primitivom *request* storitev zahteva, je **pobudnik (iniciator)** storitve, drugi uporabnik, ki je o storitvi obveščen s primitivom *indication*, pa je **prejemnik (recipient)** storitve. Na strani pobudnika se torej uporabljata primitiva `req` in `conf`, na strani prejemnika pa primitiva `ind` in `resp`.

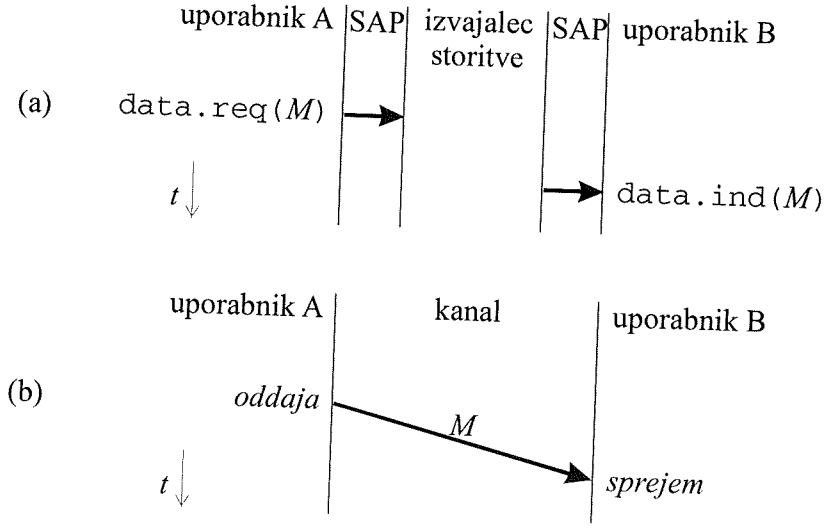
Ni pa potrebno, da bi v vsakem primeru uporabili vse štiri primitive; lahko se izvedejo le nekateri izmed njih, kar je odvisno od storitve. Tako bomo npr. potrditev `conf` uporabili le pri storitvah s potrditvijo, pri storitvah brez potrditve pa ne. Najpogostejše kombinacije so `req - ind - resp - conf`, `req - ind - conf` in `req – ind`; v nekaterih primerih pa lahko uporabimo tudi en sam primitiv, kot npr. `ind` (tak primer pa je pogosto žal slabo znamenje, da gre pri izvajalcu storitve nekaj narobe!).

Specifikacija storitve mora poleg tega, kaj storitev uporabniku nudi, natanko določati tudi interakcijo med uporabnikom in ponudnikom storitve, torej, katere primitive in ob kakšnih priložnostih ter s kakšnimi parametri se sme in mora uporabiti. S tem pa je določen jezik, v katerem se »pogovarjata« uporabnik in izvajalec storitve.

Na komunikacijo med uporabnikoma pa lahko gledamo tudi nekoliko bolj abstraktno (se pravi, poenostavljeno). Namesto, da pripovedujemo zgodbo: »Uporabnik A je preko svoje točke dostopa do storitve podal zahtevo `data.req` izvajalcu storitve, naj pošlje uporabniško sporočilo M uporabniku B; nato je izvajalec storitve preko točke dostopa do storitve za uporabnika B z obvestilom `data.ind` temu uporabniku izročil uporabniško sporočilo M.«, lahko povemo bolj na kratko: »Uporabnik A je poslal uporabniško sporočilo M uporabniku B preko kanala, ki ju povezuje.«! Na storitev skupaj s točkama dostopa do storitve pri obeh uporabnikih lahko torej gledamo kot na navidezni kanal, ki neposredno povezuje oba uporabnika. Tak abstrakten kanal oziroma sama storitev nam torej skriva marsikatero podrobnost komunikacijskega procesa, ki jo pozna le izvajalec storitve in jo skriva pred uporabniki (zato je tudi tisti del slike 2 4, ki pripada izvajalcu

storitve, prazen). Za oba uporabnika je to velika prednost, saj se jima ni treba ukvarjati s podrobnostmi, ki ju sploh ne zanimajo; za njiju je pomembno, da sporočila prihajajo od enega do drugega, kako se to dogaja, pa ju prav nič ne zanima.

Na sliki 2 5 vidimo primerjavo med obema pravkar opisanima načinoma gledanja na komunikacijski proces, npr. na pošiljanje uporabniškega sporočila *M* uporabnika A uporabniku B. Slika (a) kaže komunikacijo med uporabnikoma z uporabo storitve in primitivov (z uporabniškim sporočilom kot parametrom obeh primitivov) ter izvajalca storitve, slika (b) pa prenos sporočila neposredno preko navideznega kanala. V tej sliki nam posebej pade v oči dejstvo, da sta primitiva označena z vodoravnima puščicama, sporočilo v kanalu pa s poševno puščico. Primitiv se namreč pojavi v točki dostopa ob nekem času, nima pa v skladu z našimi predpostavkami trajanja – zgodi se v trenutku, pa tudi točka dostopa do storitve nima krajevne razsežnosti; nasprotno pa potovanje sporočila skozi kanal traja nek končno dolg čas. V nadaljevanju tega teksta bomo sliko vrste (a) uporabljali pri specifikaciji storitev, sliko vrste (b) takrat, ko bomo uporabili preprost komunikacijski model oddajnik – kanal – sprejemnik, kombinacijo obeh pa tedaj, ko bomo želeli v isti sliki prikazati storitev in njeno implementacijo. Ob zadnji sliki še enkrat poudarimo, da igra sporočilo *M* v sliki 2 5(a) vlogo storitvene podatkovne enote SDU, kot jo vidi izvajalec storitve.



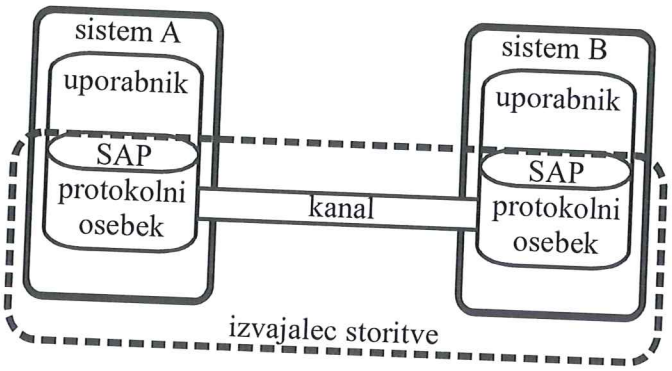
Slika 4-8. Konkretni (a) in abstraktni (b) model prenosa sporočila

4.3 Implementacija storitve, protokol

Izvajalec storitve mora torej poskrbeti za to, da se storitev, ki jo je uporabnik zahteval, dejansko izvede, in sicer v skladu s specifikacijo. Pri tem pa se je treba zavedati, da sta oba uporabnika (lahko jih je tudi več) prostorsko ločena med seboj (npr. vsak v svojem računalniku); s tem pa sta oziroma so prostorsko ločene tudi točke dostopa do storitve. Izvajalec storitve mora torej premagovati neko fizično razdaljo (prenosna funkcija), hkrati pa mora biti sposoben izvajati interakcijo z uporabnikoma na obeh straneh in na obeh straneh tudi sprejemati odločitve (procesorska funkcija). Zato mora biti izvajalec storitve sestavljen iz dveh ali več **osebkov - entitet** (ang. **entity**) in iz kanala, ki ta (te) osebke povezuje. Osebka pri izvajanju storitve sodelujeta med seboj tako, da si med seboj preko kanala izmenjujeta sporočila. Da se osebka med seboj sploh lahko razumeta, morajo biti tako formati (oblike) teh sporočil, kot tudi njihove vsebine vnaprej predpisani; prav tako pa morajo biti vnaprej predpisana pravila, ki določajo, v kakšnem primeru in ob kakšnih pogojih mora in sme en osebek drugemu poslati kakšno sporočilo. Množico pravil, ki določajo medsebojno izmenjavo sporočil ter njihove vsebine in formate, imenujemo **protokol**. Oba osebka torej med seboj komunicirata po nekem protokolu. Protokol določa, na kak način se neka storitev dejansko izvrši, torej protokol predstavlja **implementacijo storitve**. Osebek, ki je sestavni del izvajalca storitve, imenujemo **protokolni osebek** ali **protokolna entiteta**. Protokolna osebka, ki po protokolu med seboj komunicirata, lahko imenujemo tudi **partnerska osebka** oziroma **partnerski entiteti** (ang. **peer entities***) ali preprosto kar **partnerja**, sporočila, ki jih izmenjujeta, pa so **protokolne podatkovne enote**; za protokolne podatkovne enote pogosto uporabljamo kratico **PDU** (ang. **Protocol Data Unit**), bolj slovensko in predvsem preprosteje in razumljiveje pa jim lahko rečemo **protokolna sporočila**, saj jih generirajo protokolni osebki, medtem ko uporabniška sporočila generirajo uporabniki.

Na podlagi novih spoznanj lahko sedaj nekoliko izpopolnimo naš model komunikacije med dvema uporabnikoma, ki v ta namen uporabljata določeno storitev. Slika 3 prikazuje oba uporabnika ter ponudnika storitev, ki je sestavljen iz dveh protokolnih osebkov in kanala. Polni debeli črti označujeta dva sistema (npr. dva računalnika), sistem A in sistem

B, ki se nahajata vsak na svojem mestu in sta med seboj fizično oddaljena; črtkana debela črta pa predstavlja izvajalca storitve. Očitno je izvajalec storitve porazdeljen sistem, problemi komunikacije med osebki porazdeljenega sistema, ki smo jih omenili v uvodu, pa s tem postanejo problemi komunikacijskih protokolov. V tej sliki vidimo, da vsakemu uporabniku pripada v izvajalcu storitve protokolni osebek in da interakcija med uporabnikom in izvajalcem storitve v resnici poteka med uporabnikom in protokolnim osebkom.



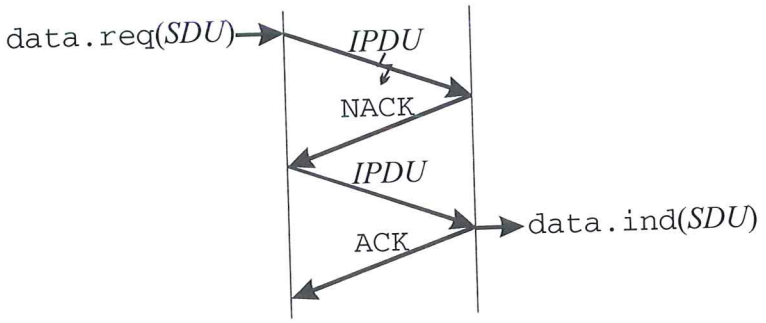
Slika 4-9. Model izvajalca storitve z dvema protokolnima osebkoma

Dogajanja v protokolnih osebkih in kanalu, skratka, v izvajalcu storitve, uporabnika storitve seveda ne vidita; onadva imata le možnost, da z ustreznimi primitivi storitev zahtevata in sta o njenem izidu obveščena; kaj se med posameznimi primitivi dogaja v izvajalcu storitve, ni njuna stvar. Seveda pa lahko pri študiju ali opisu protokola v isti časovni shemi podamo tako primitive, ki storitev zahtevajo, o njej obveščajo itd., kot tudi potek komunikacije med protokolnima osebkoma skozi kanal izvajalca storitve po protokolu, ki storitev implementira, in na ta način relacijo med storitvijo in protokolom tudi časovno opredelimo. V takem primeru zaradi večje preprostosti slike, predvsem pa zaradi korektnosti, točki dostopa do storitve (ki nima fizične dimenzije) ne bomo odmerili posebnega prostora, ampak jo bomo ponazorili le z razmejitveno črto med uporabnikom, kjer se generirajo ali sprejemajo primitivi, in izvajalcem storitve, kjer poteka komunikacija po protokolu, ki storitev implementira. Prav tako ne bomo posebej označevali protokolnih osebkov.

---

* Angleški izraz je tu prav primeren. Medtem ko med uporabnikom in izvajalcem storitve vlada odnos gospodar - podložnik, sta oba uporabnika ali oba protokolna osebka izvajalca storitve med seboj enakovredna - torej nekako na isti višini, skratka, partnerja.

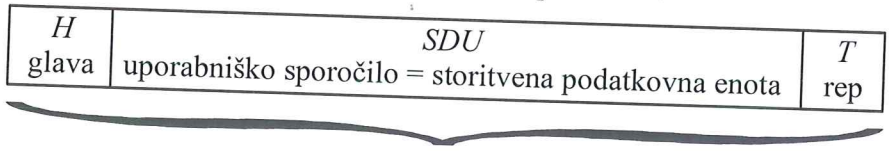
Kot zgled hkratnega prikaza interakcije uporabnik – izvajalec storitve in komunikacije med protokolnima osebkoma po danem protokolu si na sliki 2 oglejmo primer preprostega protokola (o takih bomo še govorili v nadaljevanju), ki nam omogoča zanesljiv prenos sporočila od uporabnika do uporabnika, čeprav se lahko protokolno sporočilo pri prenosu skozi kanal pokvari (ne pa tudi izgubi ali podvoji!); sprejemni protokolni osebek po sprejemu protokolnega sporočila *IPDU* pošlje oddajnemu osebku pozitivno potrditev (v sliki označeno z ACK*) v primeru uspešnega sprejema oziroma negativno potrditev (v sliki označeno z NACK) v primeru neuspešnega sprejema. Oddajni osebek oddaja isti *IPDU* toliko časa, dokler ne prejme pozitivne potrditve; sprejemni osebek pa obvesti svojega uporabnika o prejemu uporabniškega sporočila *SDU* šele tedaj, ko uspešno (to pomeni brez napake) sprejme *IPDU*. Ker storitev že sama po sebi uporabniku zagotavlja zanesljiv prenos uporabniškega sporočila, potrditev o opravljeni storitvi ni potrebna; zato uporabimo le primitiva `data.req` in `data.ind`. Gre torej za storitev brez potrditve! Pri tem ne bo odveč ponovno poudariti, da se lastnost storitve z ali brez potrditve nanaša na dejstvo, ali pobudnik storitve prejme potrditev v obliki primitiva *confirm* od izvajalca storitve, kar pa se v našem primeru ne zgodi; potrditve ACK, ki jih pošilja sprejemna protokolna entiteta, so del implementacije storitve in jih torej uporabnik ne vidi. (V narisanem primeru se je protokolna podatkovna enota pri prvem prenosu skozi kanal očitno pokvarila, kar v sliki nakazuje puščica v obliki strele, drugič pa je bila uspešno prenešena).



Slika 4-10. Primer prikaza komunikacije med osebkoma in interakcije z uporabnikoma v časovni shemi

* V celotnem besedilu pišemo imena sporočil, ki so nespremenljiva in se vedno pojavijo enako, v pisavi z enako širino črk (npr. ACK), s poševno pisavo (npr. *IPDU*) pa označujemo pojme, ki jih v dejanskih situacijah zamenjamo z ustreznim nizom oziroma vsebino.

Pri ogledu slike 2 bi utegnili imeti vtis, da sta uporabniško sporočilo (storitvena podatkovna enota) in protokolno sporočilo (protokolna podatkovna enota) eno in isto. Vendar pa temu ni tako! Protokolna podatkovna enota je namreč sporočilo, ki ga en protokolni osebek pošilja drugemu in torej ni namenjena uporabniku. Lahko pa protokolno sporočilo nosi s seboj uporabniško sporočilo in sicer tako, da je uporabniško sporočilo sestavni del protokolnega sporočila. Protokolni osebek torej uporabniškemu sporočilu, ki ga je prejel od uporabnika, doda še nekaj svoje informacije, ki pa je namenjena partnerskemu osebku in ne uporabniku. V zadnjem zgledu bi takšno dodatno informacijo lahko predstavljal paritetni bit, ki bi sprejemnemu protokolnemu osebku omogočal ugotoviti, ali je prišlo pri prenosu protokolnega sporočila do napake. Prav ta dodatna informacija je bistvena za delovanje protokola, saj je njena naloga, da nadzoruje pravilen potek izmenjave protokolnih sporočil, s tem pa tudi s specifikacijo storitve skladen prenos uporabniških sporočil. Zato to dodano informacijo imenujemo **nadzorna protokolna informacija** in včasih označimo s kratico **PCI** (ang. **Protocol Control Information**). Načeloma lahko protokolni osebek dodaja svojo nadzorno informacijo pred uporabniško sporočilo (ta dodatek imenujemo **glava** ali ang. **header**) in za uporabniško sporočilo (to je **rep** ali ang. **tail**). Splošno protokolno sporočilo je torej sestavljeno iz treh polj (*H*, *SDU*, *T*) in izgleda tako, kot prikazuje slika 4 11. Glava in rep skupaj predstavljata nadzorno protokolno informacijo. V posebnem primeru lahko protokolno sporočilo ne vsebuje nikakršnega uporabniškega sporočila, kar pomeni, da je dolžina polja *SDU* enaka nič. Protokolno sporočilo, ki vsebuje uporabniško sporočilo, bomo imenovali **informacijsko protokolno sporočilo** oziroma **informacijska protokolna podatkovna enota**, saj nosi s seboj uporabniško informacijo; protokolno sporočilo, ki pa uporabniškega sporočila ne vsebuje, bomo imenovali **nadzorno protokolno sporočilo** oziroma **nadzorna protokolna podatkovna enota**, saj taka sporočila služijo le nadzoru nad pravilnim potekom protokola.



Slika 4-11. Sestava protokolnega sporočila

Uporabniško sporočilo je torej v kanalu izvajalca storitve le del protokolnega sporočila. Ker pa je za oba protokolna osebka pomembna predvsem nadzorna protokolna informacija PCI in se največkrat za vsebino uporabniškega sporočila prav nič ne brigata, nadzorna protokolna informacija na ta način ovije in nekako skrije uporabniško informacijo*. Postopku rečemo **ovijanje** ali **inkapsulacija** (ang. **encapsulation**).

Ker protokolna nadzorna informacija ni namenjena sprejemnemu uporabniku, ampak protokolnemu osebku, jo le-ta interpretira in obdrži zase, sprejemnemu uporabniku pa preda le uporabniško sporočilo. Temu postopku rečemo **izluščenje** ali **dekapsulacija** (ang. **decapsulation**). Ovijanje in izluščenje sta torej nasprotni operaciji in vedno nastopata v paru.

Sedaj lahko še nekoliko pokomentiramo zgled, ki ga prikazuje slika 2. Vsa sporočila, ki si jih med seboj izmenjata protokolna osebka, so seveda protokolna sporočila (abeceda našega preprostega protokola torej vsebuje sporočila *IPDU*, `ACK` in `NACK`). Vendar pa je sporočilo *IPDU** informacijsko protokolno sporočilo, `ACK` in `NACK` pa sta nadzorni protokolni sporočili.

Nadzorna protokolna informacija pogosto vsebuje:

- identifikacijo tipa protokolne podatkovne enote;
- sinhronizacijsko sekvenco, ki sprejemniku omogoča, da prepozna začetek protokolne podatkovne enote;
- oznako (zaporedno številko) protokolne podatkovne enote;
- informacijo o dolžini protokolne podatkovne enote;
- naslov pošiljatelja in/ali prejemnika protokolne podatkovne enote;
- dodatno (redundantno) informacijo, ki sprejemniku omogoča ugotoviti, ali protokolna podatkovna enota vsebuje napako.

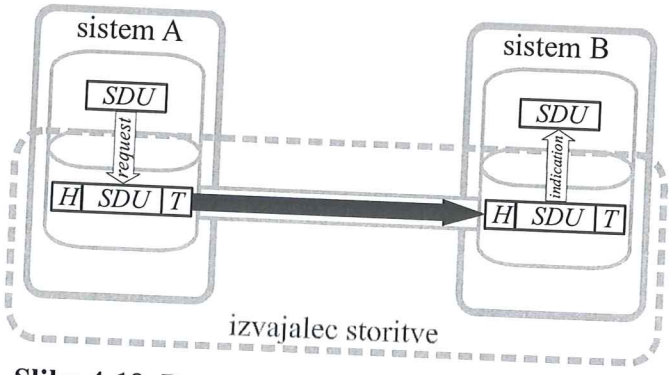
Prenos uporabniškega sporočila od enega do drugega uporabnika torej izgleda nekako takole: en uporabnik preda osebku izvajalca storitve zahtevo po prenosu uporabniškega sporočila skupaj s tem sporočilom kot parametrom; ta osebek doda uporabniškemu

---

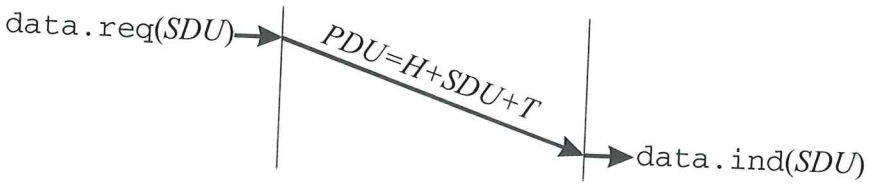
* Zelo podobno, kot kuverta skrije vsebino pisma

* Vsebina tega sporočila se bo od primera do primera spreminjala, saj bodo ta protokolna sporočila prenašala različna uporabniška sporočila, prav tako pa se bo spreminjala tudi nadzorna protokolna informacija, ki je odvisna od uporabniškega sporočila; v tem primeru je torej *IPDU* tip protokolnega sporočila.

sporočilu (ki se s stališča izvajalca storitve imenuje SDU) glavo in rep ter tako dobi protokolno sporočilo *PDU=H+SDU+T* ter pošlje to protokolno sporočilo partnerskemu osebku; le-ta glavo in rep, ki sta namenjena njemu, obdrži, uporabniško sporočilo *SDU* pa skupaj z obvestilom o prispelem sporočilu (*indication*) preda uporabniku. Tak postopek prikazuje slika 6, ustrezno časovno shemo pa slika 7.



Slika 4-12. Prenos sporočila med uporabnikoma



Slika 4-13. Prenos sporočila med uporabnikoma

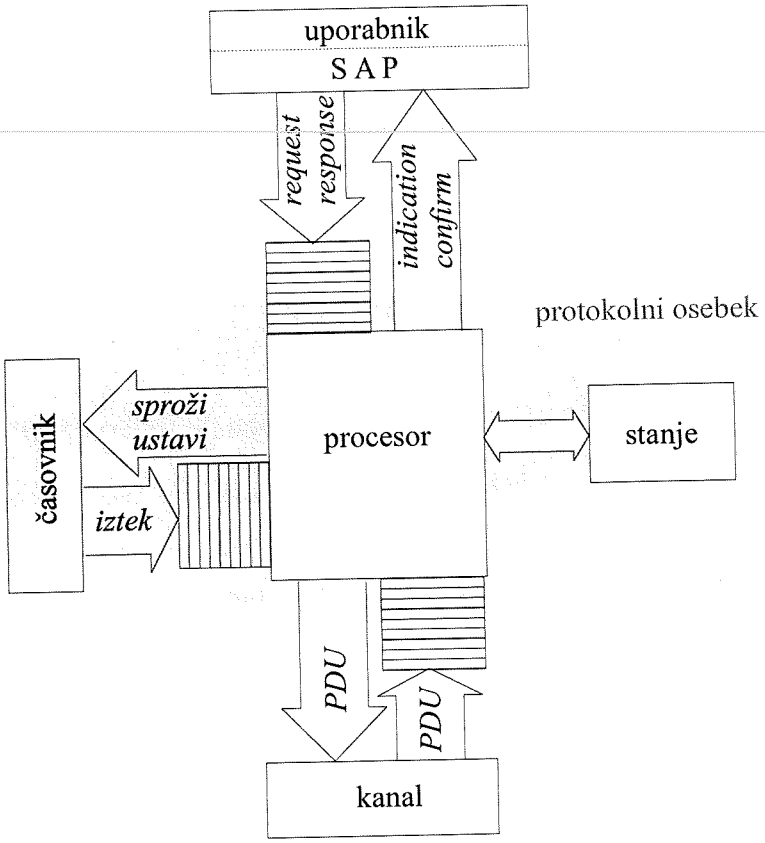
Nadzorna protokolna informacija PCI (torej glava in rep skupaj) predstavlja ***režijo*** protokola (ang. **overhead**), saj je to nekakšna administrativna informacija, ki je sicer potrebna, ne nosi pa uporabniške informacije; nujno zlo torej. Po drugi strani pa uporabniško sporočilo predstavlja **uporabniško informacijo** oziroma koristno breme; zanjo se v angleščini uporablja tudi izraz **payload**, tista obremenitev torej, ki se plačuje. Medtem ko uporabnika zanima le prenos uporabniških sporočil SDU, se po kanalu, ki je sestavni del izvajalca storitve, prenašajo protokolna sporočila PDU, ki so za režijo daljša od uporabniških sporočil; seveda pa se po kanalu izvajalca storitve prenašajo tudi nadzorna protokolna sporočila, ki vsebujejo samo režijo. Uporabnik zato vidi navidezen (virtualen) kanal, katerega hitrost prenosa informacije je manjša od hitrosti prenosa informacije po kanalu izvajalca storitve, saj se del kapacitete kanala izvajalca storitve porabi za izvajanje samega protokola oziroma za prenos režije.

Naprava, ki deluje po pravilih, ki jih določa protokol, je protokolni osebek. Torej protokolni osebek predstavlja **implementacijo (izvedbo) protokola**. Zaradi lažje predstave si sedaj še na kratko poglejmo, kaj protokolni osebek sploh je. To je lahko bodisi elektronsko vezje ali pa nek programski modul. Poznamo torej **strojno** ali **programsko** izvedbo protokola; v obeh primerih imamo proces, ki se odvija v neki komunikacijski napravi. Gre za proces, ki deluje *v* **diskretnem času**, kar pomeni, da se na vhodne dogodke odziva z internimi (notranjimi) ali izhodnimi **dogodki**; tovrstne sisteme imenujemo **dogodkovni sistemi** oziroma **sistemi diskretnih dogodkov** (ang. **discrete event systems**). Ker pa se tak proces mora na vhodne dogodke odzivati sproti oziroma z neko navzgor omejeno zakasnitvijo (torej dovolj hitro), pravimo, da gre za proces, ki deluje v **realnem času**. Vsak sistem, ki v osnovi deluje tako, da se odziva na dogajanje na svojem vhodu, imenujemo **reakcijski sistem**, to pa očitno velja tudi za protokolni osebek. V obeh primerih implementacije potrebujemo nekakšen **procesor**, ki je sposoben sprejemati odločitve glede na vhodne dogodke in trenutno stanje procesa ter v skladu s protokolom, in pa **spomin**, ki hrani stanje procesa. V vsakem primeru gre torej za nek računalnik - bodisi realen, ali pa navidezen. Če pogledamo sedaj celoten sistem, ki ga sestavljajo uporabniki in protokolni osebki, vidimo, da prihaja v tem sistemu do izmenjave informacije med uporabnikom in protokolnim osebkom ter med protokolnima osebkoma. Ker pa lahko vsi ti osebki delujejo z različnimi hitrostmi, informacija pa se lahko v sistemu poraja tudi naključno, bo vsak izmed teh osebkov na svojem vhodu vsaj načeloma potreboval še neko čakalno vrsto, ki bo skrbela za to, da se informacija ne bo izgubila tudi v primeru, če je osebek zaseden v trenutku, ko na njegov vhod pride nova informacija. Tako bo potrebna čakalna vrsta v točki dostopa do storitve za primitive vrste *request* in *response* v smeri proti izvajalcu storitve in za primitive vrste *indication* in *confirm* v smeri proti uporabniku, včasih pa je potrebna tudi čakalna vrsta na vhodu v protokolni osebek s strani kanala.

Protokolni osebek pri svojem delovanju pogosto čaka na nek vhodni dogodek, ki mu omogoči nadaljevanje normalnega delovanja. Vendar pa realni sistemi večkrat žal ne delujejo čisto tako, kot bi od njih najraje pričakovali. Tako se pogosto pripeti, da dogodka, ki ga nestrpno pričakujemo, preprosto ni. Človeku v takšnem primeru pač poide potrpljenje in poišče nek izhod v sili, podoben varnostni mehanizem pa mora imeti

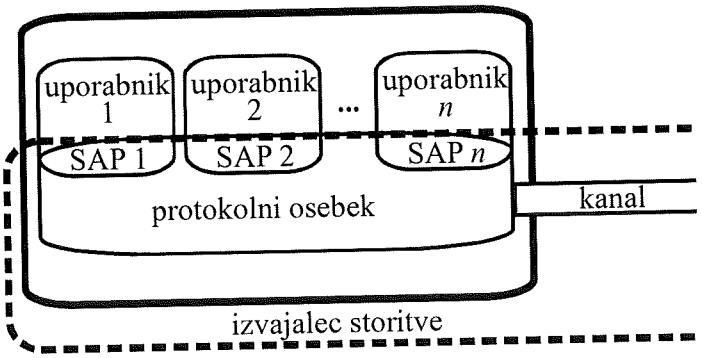
vgrajen tudi stroj. Tak varnostni mehanizem imenujemo **časovnik** (ang. **timer**). Časovnik je nekakšna ura, ki teče, potem ko smo jo pognali, le nek vnaprej določen čas, nakar se ustavi in sproži nekakšen alarm. V našem primeru bomo temu alarmu rekli **iztek časovnika** (ang. **timeout**), čas teka časovnika pa bomo imenovali **čas izteka** ali **časovna omejitev**. Po preteku časa izteka se torej časovnik izteče, seveda pa lahko protokolni osebek časovnik ustavi tudi pred njegovim iztekom (običajno takrat, ko se zgodi dogodek, na katerega smo čakali).

Protokolni osebek si lahko torej nekoliko abstraktno predstavljamo kot nek procesor, ki ima svoje stanje, vhode in izhode. Na vhodih se pojavljajo vhodni dogodki, procesor pa se na vhodne dogodke odziva z izhodnimi dogodki. Vhoda protokolnega osebka sta točka dostopa do storitve (SAP) s primitivoma *request* in *response* kot možnima vhodnima dogodkoma ter kanal s sprejetimi protokolnimi sporočili kot vhodnimi dogodki, možen vhodni dogodek pa je tudi iztek časovnika. Izhoda protokolne entitete sta dostopna točka do storitve (SAP) s primitivoma *indication* in *confirm* kot možnima izhodnima dogodkoma ter kanal z oddanimi protokolnimi sporočili kot izhodnimi dogodki, možna izhodna dogodka pa sta tudi sprožitev in zaustavitev časovnika. Medtem ko vhodni dogodki predstavljajo vzbujanje osebka, so izhodni dogodki njegov odziv. Takšno gledanje na protokolni osebek ilustrira slika 4 14.



Slika 4-14. Model protokolnega osebka

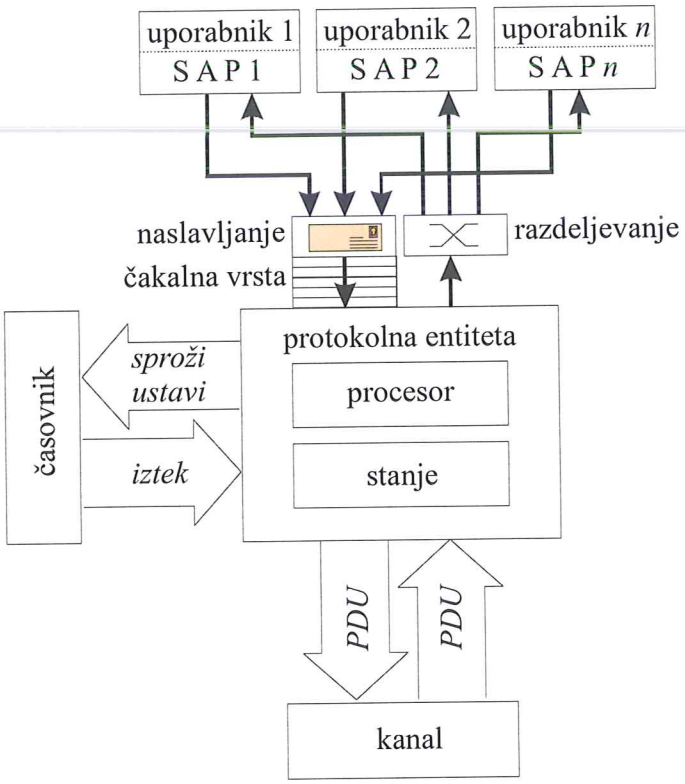
Do storitev, ki jih nudi nek protokolni osebek, ima lahko dostop tudi več uporabnikov hkrati. Ker pa točka dostopa do storitve, če jo gledamo s strani izvajalca storitve, igra tudi vlogo naslova uporabnika, mora vsak uporabnik imeti svojo točko dostopa do storitve, da jih lahko izvajalec loči med seboj. Na sliki 4 15 vidimo primer večih uporabnikov, vsakega s svojo točko dostopa do storitev, ki jih izvaja isti protokolni osebek. Slika 4 16 pa nam kaže pravkar opisano dvojno vlogo točke dostopa do storitve SAP.



Slika 4-15. Protokolni osebek z več uporabniki

Slika 4-16. Dvojna vloga točke dostopa do storitve SAP

Ker ima protokolni osebek običajno le en sam procesor (takšnega smo tudi prikazali v sliki 4 14), lahko v nekem trenutku v resnici opravlja le eno delo. Hkratna obdelava različnih zahtev ali odgovorov je torej le navidezna. V resnici morajo primitivi različnih uporabnikov vstopati v protokolni osebek izvajalca storitve skozi čakalno vrsto. Poleg tega ima lahko izvajalec storitve en sam kanal (za vsako možno destinacijo sporočil). Protokolni osebek lahko torej na eni strani **multipleksira** sporočila različnih uporabnikov v isti kanal in jih na drugi strani **demultipleksira**. Da je to mogoče, morajo biti protokolna sporočila opremljena z naslovi uporabnikov oziroma točk dostopa do storitev. Ti naslovi so običajno zapisani v glavah protokolnih sporočil. Zato je treba sporočila različnih uporabnikov opremiti z naslovi (pogosto pošiljatelja in prejemnika) in združevati v isto čakalno vrsto ter naprej preko procesorja v isti kanal. Po drugi strani pa mora protokolni osebek za vsako sprejeto protokolno sporočilo ugotoviti, na katerega uporabnika se nanaša, sprejeta sporočila morajo torej skozi element za razdeljevanje (distribucijo), ki naslove kot del režije odstrani. Slika 4 17 kaže model protokolnega osebka, ki izvaja multipleksiranje in demultipleksiranje.



Slika 4-17. Protokolni osebek z multipleksiranjem/demultipleksiranjem

4.4 Definicija protokola

Kot smo nakazali že v uvodu, predstavlja protokol jezik, v katerem se pogovarjata oba partnerja. Kot vsak jezik, naravni ali umetni, ima tudi protokol naslednje sestavine.

- **Abeceda (alfabet)** je množica vseh možnih tipov protokolnih sporočil (protokolnih podatkovnih enot).
- **Format** protokolnih sporočil določa njihovo notranjo zgradbo (strukturo)
- **Skladnja (sintaksa)** je množica vseh veljavnih oziroma dovoljenih zaporedij protokolnih podatkovnih enot.
- **Semantika** podaja pomen, učinek oziroma interpretacijo protokolnih sporočil in njihovih zaporedij.

Abecedo in skladnjo skupaj imenujemo slovnica (**gramatika**).

Pri tem je treba poudariti, da je jezik, ki ga predstavlja protokol, daleč manj preprost, kot so npr. jeziki za pisanje običajnih (to je sekvenčnih) računalniških programov. Protokol mora namreč določati ne le množico zaporedij protokolnih sporočil enega osebka, ampak možna zaporedja protokolnih sporočil, ki jih pošiljata dva osebka (ali celo

več), ki pri tem niti točno ne poznata stanja celotnega sistema, sporočila pa potrebujejo tudi določen čas, da pridejo od enega osebka do drugega. Protokol mora torej skrbeti za usklajeno delovanje dveh ali več subjektov v porazdeljenem sistemu. Možnih stanj v takem sistemu pa je veliko več kot v strnjenem sistemu, saj lahko protokolni osebek prejme od drugih osebkov v porazdeljenem sistemu zelo veliko število različnih zaporedij sporočil. Nekoliko drugače bi lahko rekli tudi takole: kompleksnost definicije protokola nam otežuje dejstvo, da lahko protokolni osebek sporočila ne le oddaja, ampak tudi sprejema, oddajanje sporočil pa je vsekakor zelo odvisno od sprejemanja sporočil.

Poskusimo sedaj nekoliko podrobneje predstaviti posamezne elemente definicije protokola.

Nabor sporočil (abeceda) predstavlja množico vseh različnih tipov protokolnih podatkovnih enot, ki jih en protokolni osebek lahko pošlje drugemu. Pri tem vsebina uporabniškega sporočila, če je le-to vsebovano v protokolnem sporočilu, ni pomembna*; dveh protokolnih sporočil istega tipa, ki se razlikujeta le po vsebovanem uporabniškem sporočilu, torej v tem smislu ne bomo šteli za različni. Protokolno sporočilo določenega tipa lahko poleg uporabniškega sporočila nosi s seboj tudi še druge parametre z vrednostmi, ki so značilne za tak tip sporočila. Pogosti parametri so številka paketa, ki ga nosi PDU, naslov, kamor je namenjen, ipd. Protokolno sporočilo lahko na abstrakten način predstavimo v obliki

$$PDUType\ (p_1, p_2, \ldots)\ ,$$

kjer predstavlja *PDUType* tip protokolnega sporočila, p_i pa so parametri protokolnega sporočila. Takšno predstavitev protokolnega sporočila imenujemo abstraktna sintaksa protokolnega sporočila.

Format protokolnih podatkovnih enot predstavlja njihovo implementacijo v smislu njihove notranje strukture. Protokolno sporočilo je namreč lahko sestavljeno iz več sestavnih delov, format pa določa, na kakšen način te sestavne dele sestavimo v celoto. Pogosti sestavni deli protokolnega sporočila so poleg uporabniškega sporočila in parametrov protokolne podatkovne enote še identifikacija tipa PDU-ja, sinhronizacijska

* Načeloma se vsebina uporabniškega sporočila protokolnega osebka sploh ne tiče! (Kje so že časi, ko so poštni uradniki iz samega dolgega časa brali pisma!?)

sekvenca, zaščitna koda, ki omogoča popravljanje napak, zaporedna številka protokolnega sporočila ipd.

Zbirka protokolnih pravil opisuje sintakso in deloma tudi semantiko protokola ter podaja podrobna navodila o tem, kdaj in v kakšnih okoliščinah sme in mora protokolni osebek poslati protokolno sporočilo določenega tipa in z določenimi parametri, pa tudi, kdaj in v kakšnih okoliščinah se spreminja stanje osebka. Pri tem ne smemo pozabiti, da morajo ta pravila specificirati tudi začetek delovanja (inicializacijo) osebka. Protokolna pravila torej določajo množico dovoljenih zaporedij vhodnih in izhodnih dogodkov protokolnega osebka ter smisel in pomen teh zaporedij. Takšna specifikacija protokolnih pravil, ki se nanaša le na delovanje enega protokolnega osebka, predstavlja osnovo za implementacijo protokolnega osebka, saj le-ta med svojim delovanjem pozna izključno le svoje stanje in dogajanje na svojih vhodih in izhodih, ne pozna pa stanja in dogodkov svojih partnerjev, s katerimi sodeluje v komunikacijskem procesu; na stanje partnerjev lahko sklepa le iz sporočil, ki jih od njih dobi. Seveda pa bomo pri našem študiju protokolov pogosto skušali preučiti in razumeti tudi celotno dogajanje v komunikacijskem sistemu, zato bomo skušali s svojim pogledom zaobjeti delovanje obeh (oziroma vseh) partnerjev naenkrat. V ta namen bomo uporabljali časovno shemo, ki smo si jo že ogledali v podpoglavju "Implementacija storitve, protokol". Ni pa taka shema preveč primerna za specifikacijo protokola.

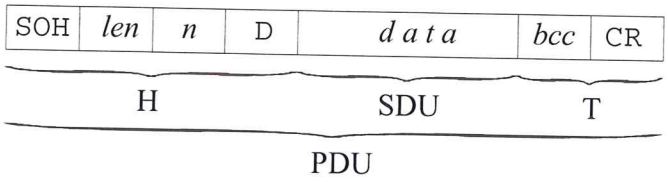
Tem elementom specifikacije protokola, ki smo jih pravkar razložili, pa je treba dodati še dva: **specifikacijo storitev**, ki jih protokol nudi uporabniku, in **specifikacijo kanala**, ki ga protokolna osebka uporabljata za izmenjavo protokolnih sporočil. Ker uporabnik vidi storitve protokola kot nekak abstraktnejši navidezni kanal, lahko rečemo, da uporaba protokola preslika lastnosti konkretnejšega (realnejšega) kanala, ki ga protokolna osebka uporabljata, v lastnosti abstraktnejšega (navideznega) kanala, ki ga vidita oziroma uporabljata uporabnika.

Zgled: preprost protokol za prenos datoteke (Kermit) - Tu si bomo ogledali kot zgled zelo preprost protokol za zanesljiv prenos zaporedja uporabniških sporočil preko kanala, v katerem se sporočila lahko pokvarijo ali izgubijo. Tak mehanizem prenosa, vključno s formatom protokolnih sporočil, je uporabljal star in preprost protokol kermit, ki je služil za prenos datotek med dvema računalnikoma. Seveda smo protokol kermit za naš zgled

precej poenostavili. Tako na primer nič ne bomo izvedeli o tem, kako oddajni osebek razdrobi celotno datoteko na več segmentov, in tudi ne o tem, kako sprejemni osebek ugotovi, kateri segment je zadnji. Abstraktna sintaksa informacijskega protokolnega sporočila je

`FileData` (*n, data*) ;

to sporočilo prenese uporabniške podatke (del datoteteke) od enega protokolnega osebka do drugega; *n* je zaporedna številka informacijske protokolne podatkovne enote in *data* uporabniško sporočilo v obliki zaporedja zlogov. Takšno protokolno sporočilo implementiramo z okvirjem, ki ima format (konkretno sintakso), prikazan na sliki 4 18.



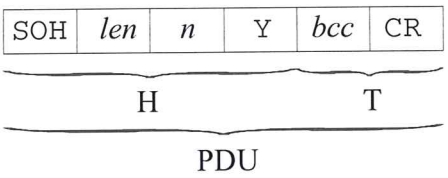
Slika 4-18. Format informacijskega protokolnega sporočila `FileData`

Očitno je ta okvir sestavljen iz več polj; dolžina vseh polj razen polja *data* je 1 zlog, dolžina polja *data* pa se lahko od okvirja do okvirja spreminja. `SOH` je ASCII nadzorni znak "Start Of Header" (z binarno vrednostjo 0000001) in označuje začetek glave protokolne podatkovne enote. *len* je dolžina podatkovnega polja *data* v zlogih, povečana za 35 in zapisana v binarni obliki. Polje *n* dobimo tako, da parametru *n* oziroma zaporedni številki informacijskega protokolnega sporočila (katere vrednost je lahko v razponu med 0 in 63) prištejemo 32 in vsoto zapišemo z binarnim zaporedjem. `D` je ASCII znak "D" (binarno zaporedje 1000100), njegova naloga pa je, da identificira `FileData` kot tip protokolne podatkovne enote. Polje *data* predstavlja storitveno podatkovno enoto (uporabniško sporočilo), ki ga en uporabnik želi poslati drugemu. Vsebino zloga *bcc* določimo na oddajni strani glede na vsebino protokolnega sporočila tako, da nam na sprejemni strani omogoča odkrivanje morebitnih napak pri prenosu. Polje `CR` vsebuje ASCII znak "Carriage Return" (z binarno vrednostjo 0001101) in predstavlja zaključek okvirja. Kaj več o pomenu vseh teh polj bomo spoznali v nadaljevanju, za sedaj pa lahko le ugotovimo, da predstavljajo (kot lahko tudi vidimo v sliki 4 18) polja `SOH`, *len*, *n* in `D` skupaj glavo protokolne podatkovne enote, polje *data* storitveno podatkovno enoto, polji *bcc* in `CR` pa rep protokolne podatkovne enote.

Poglejmo si sedaj še primer abstraktne sintakse nadzornega protokolnega sporočila, ki ne vsebuje uporabniškega sporočila. S sporočilom

`Ack (n)`

lahko protokolni osebek svojemu partnerju potrdi pravilen sprejem informacijskega protokolnega sporočila z oznako (številko) *n*. Format tega sporočila je podan na sliki 4 19 in je podoben formatu sporočila `FileData`, le da je oznaka tipa okvirja tokrat ASCII znak "Y" (z binarno vrednostjo 1011001), uporabniškega sporočila ni, polje *len* pa zatorej vsebuje decimalno vrednost 35, ki ustreza dolžini polja *data* 0. Ta protokolna podatkovna enota torej vsebuje le glavo in rep.



Slika 4-19. Format nadzornega protokolnega sporočila ACK

Za ta protokol bi lahko zapisali še nekaj preprostih pravil za delovanje protokolnega osebka:

1. ko dobiš od uporabnika zahtevo po prenosu datoteke, le-to razdeli v primerno velike kose (fragmente), oddaj protokolno sporočilo `FileData (0, data)` s prvim fragmentom datoteke in sproži časovnik; od partnerja na začetku pričakuj segment številka 0;
2. ko sprejmeš protokolno sporočilo `FileData (n, data)` brez napake, oddaj protokolno sporočilo `Ack (n)` tako, da bosta imela parametra *n* obeh sporočil isto vrednost; če si sprejel pričakovani fragment, dodaj prejeti fragment že sestavljeni datoteki in pričakuj naslednji fragment;
3. ko sprejmeš protokolno sporočilo `Ack (n)` (pri čemer je vrednost parametra *n* enaka vrednosti parametra *n* nazadnje poslanega sporočila `FileData`), ustavi časovnik, oddaj protokolno sporočilo `FileData (n+1, data)` tako, da bo *data* vseboval segment prenašane datoteke, ki sledi segmentu, poslanemu v prejšnjem sporočilu tipa `FileData`, in ponovno sproži časovnik;
4. če se izteče časovnik, ponovno oddaj protokolno sporočilo, ki si ga nazadnje oddal, in spet sproži časovnik.

5. korake 2÷4 ponavljaj toliko časa, dokler ne prejmeš potrditve zadnjega fragmenta prenašane datoteke; če si sprejel zadnji fragment prenašane datoteke, le-to dokončno sestavi in jo predaj uporabniku.

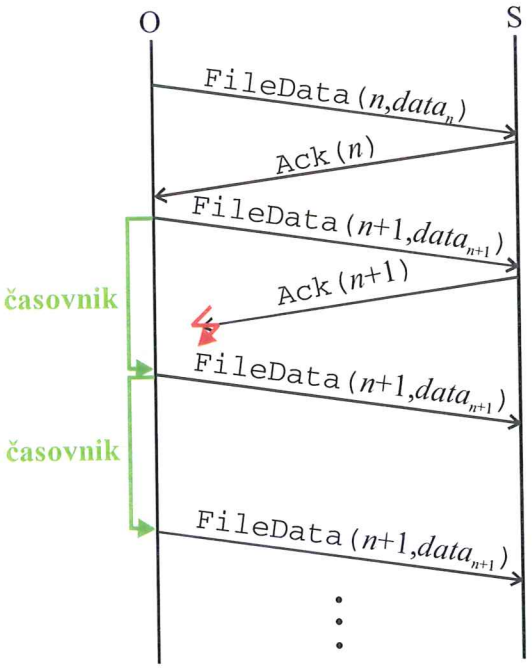
Kakšna je torej specifikacija storitve, ki jo nudi pravkar opisani protokol svojemu uporabniku, in kakšne morajo biti lastnosti kanala, ki ga uporabljata protokolna osebka? Protokol (Kermit) nudi uporabniku prenos datoteke. Za pravilno delovanje pa potrebuje kanal, v katerem se protokolna sporočila sicer lahko pokvarijo ali izgubijo, sprejeta pa so v istem vrstnem redu, kot so bila oddana. ♦

4.5 Lastnosti protokola

Izmed vseh elementov protokola je najtežje specificirati in implementirati zbirko protokolnih pravil. Ta pravila so namreč najbolj podvržena napakam, ki lahko potem vodijo do nepravilnega delovanja komunikacijskega sistema. Da bi se takim težavam izognili, mora imeti pravilno načrtovan protokol določene lastnosti, ki jih bomo sedaj našteli in na kratko opisali.

Logična pravilnost protokolnih pravil pomeni, da ta pravila ne vsebujejo napak, ki bi vodile k napačnemu delovanju komunikacijskega sistema. Z drugimi besedami pa to pomeni, da delovanje komunikacijskega sistema vodi k zastavljenemu cilju, da komunikacijski proces ne zaide v **mrtvo točko** (angleško **deadlock**), iz katere ni izhoda, ali v **brezizhodno zanko** (angleško **livelock**), ki se ponavlja v neskončnost, ne da bi proces pri tem dosegal kakršenkoli napredek. (Pojma mrtve točke in brezizhodne zanke lahko zelo preprosto pojasnimo na zgledu iz prejšnjega razdelka, protokolu za prenos datoteke. Če protokolni osebek časovnika ne bi uporabljal, bi v primeru, ko bi se bodisi informacijsko protokolno sporočilo ali njegova potrditev pokvarila ali izgubila, potrditve ne dobil; sprejemni osebek bi čakal na informacijsko protokolno sporočilo, oddajni pa na njegovo potrditev; komunikacijski proces bi se ustavil v mrtvi točki. Morda pa je bralca v tem zgledu zmotilo dejstvo, da mora protokolni osebek potrditi sprejem tudi takega informacijskega protokolnega sporočila, ki ga ne pričakuje; vzemimo primer, da bi osebek pravilno sprejel pričakovano informacijsko sporočilo in poslal partnerju njegovo potrditev, ki pa bi se izgubila; po izteku časovnika bi oddajni osebek isto informacijsko sporočilo ponovno poslal; če sprejemnik le-tega ne bi ponovno potrdil, bi se oddajniku ponovno iztekel časovnik in osebka bi se znašla v brezizhodni zanki; tak "defekten"

primer si lahko ogledamo v časovni shemi na sliki 4 20; bralec pa se lahko sam prepriča, da se tej zanki izognemo, če sprejemni osebek potrdi vsa pravilno sprejeta, ne pa nujno tudi pričakovana informacijska protokolna sporočila.)



Slika 4-20. Pojav brezizhodne zanke

Popolnost, nedvoumnost in skladnost specifikacije protokolnih pravil pomenijo, da obstaja protokolno pravilo za kakršnokoli situacijo, v kateri se lahko znajde protokolna entiteta, da si teh pravil ni mogoče razlagati na dvoumen način in da si pravila niso v medsebojnem nasprotju.

Pravičnost protokola pomeni, da daje protokol vsem udeležencem v komunikacijskem procesu (to je, vsem partnerjem) enake možnosti, da odpošljejo svoja protokolna sporočila. Obstajajo tudi protokoli, pri katerih ima eden izmed osebkov (**nadrejeni** osebek) a priori večje pravice (in tudi dolžnosti) kot ostali (**podrejeni**) osebki; v tem primeru pravičnost pomeni, da morajo imeti vsi podrejeni osebki med seboj enake možnosti. Obstajajo pa tudi protokoli, kjer imajo lahko različni osebki ali pa celo različna protokolna sporočila, ki jih ti želijo poslati, različne prednosti (prioritete). V tem primeru pa pomeni pravičnost, da imajo enake možnosti vsi osebki oziroma vsa sporočila z enako prednostjo, osebek oziroma sporočilo z višjo prednostjo pa ne sme imeti slabših možnosti kot osebek oziroma sporočilo z nižjo prednostjo.

Robustnost protokola pomeni, da protokol predvidi tudi možnost nastopa situacij, do katerih pri normalnem delovanju sicer ne pride, lahko pa se pojavijo ob kakšni izjemni priliki, kot npr. ob zasedenosti virov ali ob okvari enega izmed partnerjev, in sicer v tem smislu, da protokolna osebka sama najdeta izhod iz takih situacij. Do železniške nesreče v predoru Clayton, ki smo jo omenili v Uvodu, ne bi prišlo, če bi bil protokol bolj robusten. Lastnost sistema, da se je sam sposoben vrniti iz nenormalnih stanj, v katera je zašel zaradi napak ali napada, nazaj v normalna stanja, včasih imenujemu tudi **konvergenca sistema**.

Učinkovitost protokola pomeni, da protokol čimbolj izkoristi dane kapacitete kanala in protokolnih osebkov, kar se kaže v čim večji kapaciteti navideznega kanala med obema uporabnikoma.

Standardiziranost protokola je pogoj za njegovo širšo uporabo, saj zagotavlja, da lahko po tem protokolu skladno med seboj komunicirajo naprave, ki so izdeki različnih proizvajalcev in v lasti oziroma v upravljanju različnih operaterjev.

4.6 Problemi, ki jih protokoli rešujejo

V telekomunikacijskih sistemih nastopa cela vrsta problemov, ki jih morajo reševati telekomunikacijski protokoli. Ti problemi izhajajo bodisi iz lastnosti kanala, po katerem si protokolna osebka izmenjujeta protokolna sporočila, iz informacijskih lastnosti sporočil in iz lastnosti telekomunikacijskega omrežja, del katerega je kanal. Naštejmo sedaj v kratkem nekatere izmed teh problemov.

- **Sinhronizacija** - Ko sprejemnik sprejema zaporedje sporočil skozi kanal, mu mora biti pri vsakem od njih popolnoma jasno, kje se to sporočilo začne in kje konča.
- **Oblikovanje (formatiranje)** sporočil - Vsako sporočilo, ki se prenese skozi kanal, mora biti primerno oblikovano, tako da so v njem jasno razvidni njegovi sestavni deli in jih je na sprejemni strani tudi mogoče ločiti.
- **Detekcija in popravljanje napak** - Telekomunikacijski kanali niso idealni, zato se v njih sporočila lahko tudi pokvarijo ali izgubijo. Protokol lahko skrbi tudi za to, da odkrije in nadomesti manjkajoča ter popravi pokvarjena sporočila, torej, da vsa sporočila pridejo na cilj brez napake.

- **Ohranjanje vrstnega reda sporočil** - V splošnem je zaželeno, da uporabnik sprejme sporočila v istem vrstnem redu, kot mu jih je poslal njegov partner; pogosto je tudi to naloga protokola.
- **Krmiljenje pretoka** - Različne naprave v telekomunikacijskem omrežju lahko delujejo z različnimi hitrostmi, poleg tega pa se v samem omrežju informacijski pretoki ponekod bolj, ponekod manj zgoščajo oziroma redčijo. Do neke mere lahko različne hitrosti in zgostitve prometa sicer usklajujemo s čakalnimi vrstami, vendar pa to največkrat ni dovolj, saj bi se brez dodatnih ukrepov lahko čakalne vrste hitro napolnile in bi prišlo do izgub. Protokoli morajo torej skrbeti tudi za usklajevanje pretoka podatkov skozi različne dele omrežja ali med izvorom in ponorom informacije, včasih pa tudi uravnavajo dotok informacije v samo omrežje.
- **Zagotavljanje kakovosti storitev** – del specifikacije storitve je tudi specifikacija njene kakovosti, le-to pa mora zagotavljati protokol, ki to storitev implementira.
- **Naslavljanje, multipleksiranje in usmerjanje informacije** - Sporočila nosijo s seboj podatke o tem, od kod prihajajo in kam so namenjena. Glede na te podatke omrežje sporočila multipleksira, demultipleksira ter usmerja in komutira od izvora k ponoru, kar je tudi naloga nekaterih protokolov.
- **Usklajevanje stanja komunikacijskega sistema** - Glede na to, da je komunikacijski sistem porazdeljen sistem, so za vzpostavitev nekega stanja v takem sistemu potrebni posebni postopki, ki so tudi naloga protokolov. Tipični primeri usklajevanja stanja so vzpostavljanje, vzdrževanje in sproščanje zvez, pa tudi usklajevanje podatkov v porazdeljeni podatkovni bazi.
- **Dogovarjanje o posebnih postopkih pri prenosu informacije** – Včasih so pri prenosu informacije potrebni posebni postopki, o začetku, načinu in koncu njihovega izvajanja pa se morajo protokolni osebki posebej dogovoriti. Značilen primer takih postopkov je lahko komprimiranje ali šifriranje prenašanih podatkov oziroma pogajanje o verziji in parametrih uporabljenega protokola.
- **Varnost** - Nič novega ni, da vsi subjekti v omrežju nimajo vedno poštenih namenov. Zato obstajajo posebni protokoli, ki udeležence komunikacijskega procesa in podatke ščitijo proti namernemu potvarjanju, spreminjanju, brisanju ali prisluškovanju.

- **Podpora aplikacijam v porazdeljenih sistemih** - Kot smo že ugotovili, je sistem, sestavljen iz dveh ali več računalnikov, ki med seboj komunicirajo, porazdeljen sistem. Medtem ko nekatere aplikacije (programi) delujejo le na enem računalniku ali na vsakem računalniku posebej, delujejo druge aplikacije hkrati na več sistemih, pri čemer delne aplikacije na različnih sistemih med seboj sodelujejo; to so **porazdeljene aplikacije** (ang. **distributed applications**). Nekateri komunikacijski protokoli podpirajo delovanje porazdeljenih aplikacij. Dobro znan primer porazdeljene aplikacije je brskanje po svetovnem spletu, kjer z enega mesta dostopamo do informacije, shranjene na različnih mestih v navideznem omrežju, imenovanem svetovni splet (World Wide Web – WWW). To porazdeljeno aplikacijo omogoča protokol HTTP (Hyper Text Transfer Protocol).

5 PROTOKOLNI SKLAD

5.1 Razslojevanje telekomunikacijskih sistemov

Telekomunikacijski sistemi postajajo vse obširnejši in vse kompleksnejši, s tem pa se postavljajo vse obširnejše in kompleksnejše naloge tudi pred telekomunikacijske protokole oziroma pred njihove načrtovalce, razvijalce in vzdrževalce. Glede na to, da je načrtovanje protokola že samo po sebi težavna naloga z mnogimi pastmi (spomnimo se samo na mrtve točke in neskončne zanke ter nepopolne ali dvoumne specifikacije!), lahko pričakujemo, da bo za človeški um praktično nemogoče brez napak zasnovati in realizirati tako obsežno nalogo v enem kosu, torej en sam protokol, ki bo obvladoval vse naloge tako kompleksnega sistema. Sama po sebi se torej ponuja možnost, da obširno nalogo razdelimo na več manjših in na ta način obvladamo problem po načelu rimskih imperatorjev "divide et impera" (deli in vladaj)! Vendar pa moramo pri tem paziti, da z razdeljevanjem ene velike naloge na več manjših kompleksnosti sistema celo ne povečamo! Delitev mora potekati tako, da bomo lahko sestavne dele naloge oziroma komunikacijskega sistema obravnavali čimbolj samostojno, stiki oziroma vmesniki med njimi pa bodo čim bolj preprosti in čim bolje definirani, pa tudi čim manj jih bo. Celoten sistem moramo torej zelo pazljivo strukturirati. V zadnjih desetletjih se je pri obravnavi najrazličnejših kompleksnih sistemov uveljavila **hierarhična struktura** kot tista stuktura, ki najbolj učinkovito zmanjšuje kompleksnost obravnavanih sistemov.

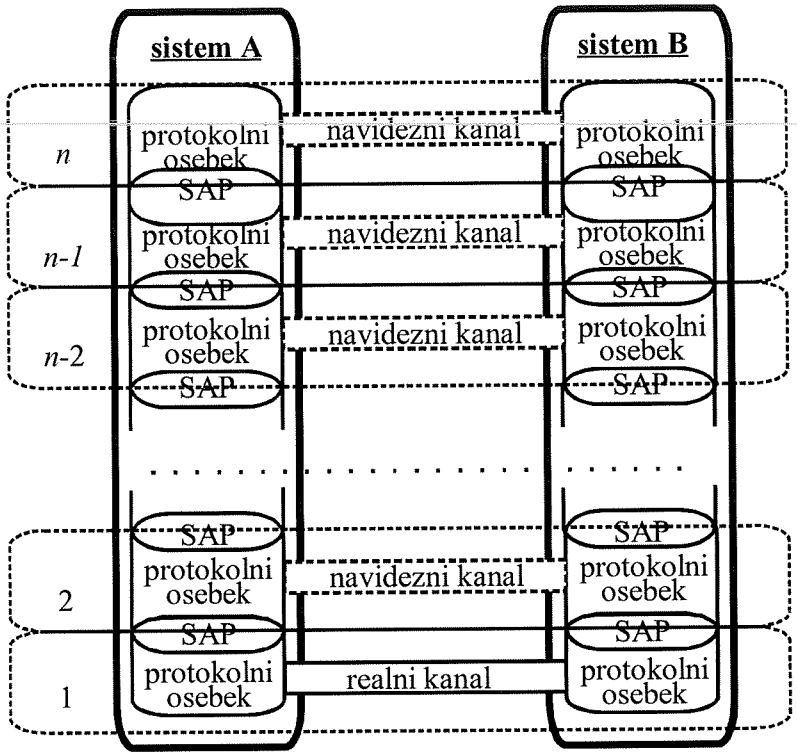
Pravzaprav ne bomo v tem poglavju povedali kaj prida novega. Izhajali bomo iz odnosa med uporabnikom in ponudnikom storitve oziroma iz odnosa med navideznim kanalom, ki povezuje dva uporabnika, in konkretnejšim kanalom, po katerem po določenem protokolu komunicirata partnerska protokolna osebka, vsi trije pa na ta način implementirajo navidezni kanal. V tem poglavju bo novo le to, da bomo že iz prejšnjega poglavja znani princip večkrat zaporedoma uporabili.

Imejmo torej dva uporabnika, ki želita med seboj komunicirati. Abstraktno si to komunikacijo lahko predstavljamo kot nek navidezen kanal, po katerem si uporabnika izmenjujeta uporabniška sporočila. V resnici pa tako komunikacijo realiziramo z dvema protokolnima osebkoma in s kanalom, po katerem ta dva osebka v skladu z določenim protokolom med seboj izmenjujeta protokolna sporočila in na ta način realizirata storitev,

ki jo oba protokolna osebka in kanal skupaj (kot izvajalec storitve) nudijo uporabnikoma. Prav izvajanje te storitve torej realizira komunikacijo po navideznem kanalu.

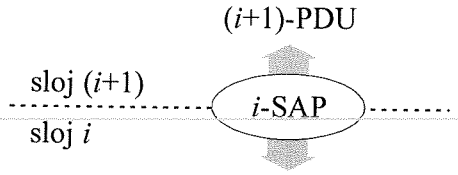
Ker pa smo ugotovili, da je zadana naloga za en sam protokol prekompleksna in pretežavna, naj ta protokol poskrbi le za izvajanje enega dela naloge, za preostali del pa naj uporabi kanal, ki ta del že sam po sebi ponuja kot storitev. Ker takih kanalov največkrat nimamo kar na razpolago, ga bomo spet implementirali z dvema osebkoma in kanalom med njima (torej z neko novo storitvijo oziroma ponudnikom, ki jo ponuja), ta dva osebka pa naj si izmenjujeta sporočila po nekem drugem protokolu, ki naj torej opravi preostalo nalogo. Kaj pa, če je še ta naloga pretežavna? Spet bomo uporabili novo storitev oziroma nov par protokolnih osebkov, ki bosta po novem kanalu, v skladu z novim protokolom prevzela del naloge. Ta postopek ponavljamo toliko časa, dokler ne pridemo do naloge, ki je dovolj preprosta, da jo je moč realizirati s preprostim protokolom, ki se izvaja med protokolnima osebkoma po nekem realnem kanalu, ki dejansko obstaja. Kanali med vsemi prejšnjimi pari partnerskih osebkov so bili namreč le navidezni.

Na ta način smo dobili vrsto parov partnerskih osebkov. Partnerska osebka med seboj komunicirata po navideznem kanalu v skladu s protokolom, ki je značilen za ta par in ga lahko imenujemo **protokol para** oziroma **partnerski protokol** (ang. **peer-to-peer protocol**). Ta protokol po eni strani predstavlja implementacijo navideznega kanala med partnerskima osebkoma na **višjem nivoju**, ki jima na ta način nudi storitev, po drugi strani pa uporablja storitev, ki mu jo nudi spet nek par osebkov s kanalom na **nižjem nivoju**. Vsakemu protokolnemu osebku na višjem nivoju mora pripadati protokolni osebek na nižjem nivoju. Med vsakim uporabnikom (protokolnim osebkom) na višjem nivoju in pripadajočim protokolnim osebkom izvajalca storitve na nižjem nivoju je seveda točka dostopa do storitve, SAP. Oglejmo si sedaj ilustracijo takega sistema na sliki **Error! Not a valid link.**.



Slika 5-1. Protokolni sklad

Iz slike je razvidno, da smo komunikacijski sistem razdelili na *n* **slojev** (ang. **layers**), ki so nekako naloženi eden na drugega, zato bomo taki strukturi rekli **protokolni sklad** (ang. **protocol stack**, **protocol suite**), včasih pa tudi **arhitektura protokolov**. Očitno gre namreč pri vsem tem za **organizacijo** telekomunikacijskega sistema. Sloje v skladu smo oštevilčili od spodaj navzgor, torej od 1 do *n*. V vsakem sloju imamo dva (ali več) protokolna osebka (partnerja) in kanal, ki ju (jih) povezuje. Kanal najnižjega sloja je realen, fizičen kanal, kanali vseh višjih slojev pa so navidezni (virtualni). Partnerja v sloju komunicirata po protokolu, ki je značilen za ta sloj, zato ga imenujemo **protokol sloja** ali tudi **protokol partnerjev** (ang. **peer-to-peer protocol**). *i*-ti sloj nudi določeno storitev (ki jo implementira protokol tega sloja) skozi točko dostopa do storitve tega sloja *i*-SAP sloju *i*+1, sloj *i*+1 je torej uporabnik storitve sloja *i*. V smislu te terminologije lahko sedaj tudi rečemo, da postane protokolno sporočilo (*i*+1)-tega sloja (*i*+1)-PDU uporabniško sporočilo *i*-tega sloja *i*-SDU, ki skupaj z nadzorno protokolno informacijo *i*-tega sloja *i*-PCI tvori protokolno sporočilo *i*-tega sloja *i*-PDU, kar lahko vidimo na sliki **Error! Not a valid link.**.

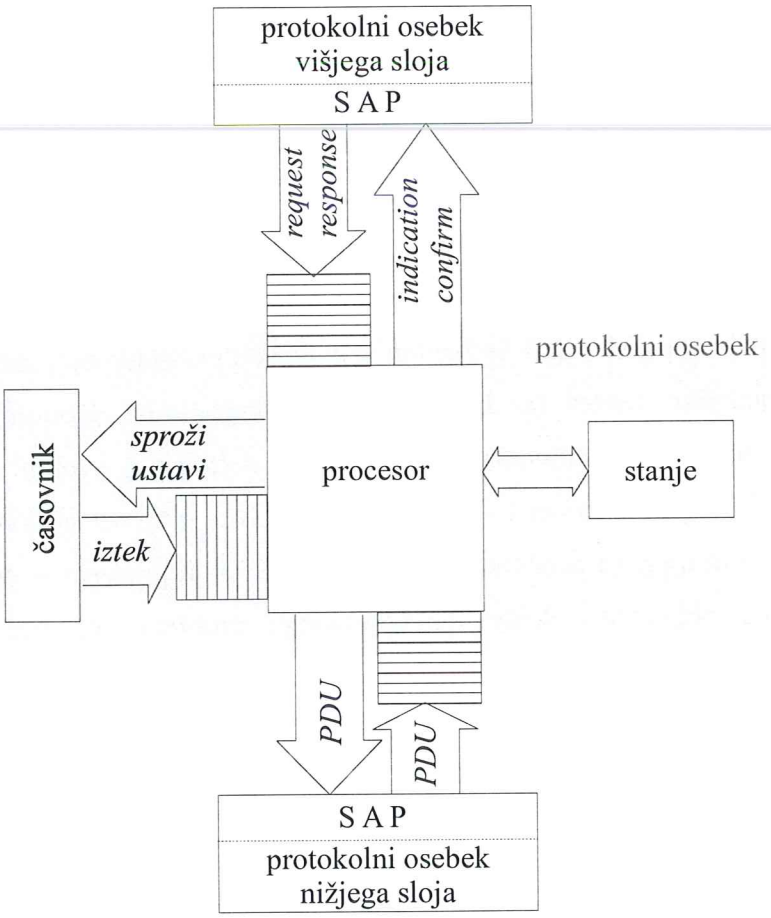


i-PDU = i-PCI + i-SDU

i-SDU = (i+1)-PDU

Slika 5-2. Sporočila dveh sosednjih slojev v protokolnem skladu

Model protokolnega osebka, ki smo si ga že ogledali v poglavju "Implementacija storitve, protokol", lahko za protokolni osebek v enem izmed slojev protokolnega sklada sedaj še nekoliko priredimo njegovemu novemu položaju. Protokolni osebek nekega (razen seveda najnižjega) sloja namesto kanala v resnici vidi točko dostopa do storitev nižjega sloja, za katero pa se skriva protokolni osebek nižjega sloja. Tak spremenjeni model protokolnega osebka v sloju protokolnega sklada vidimo na sliki **Error! Not a valid link.**.

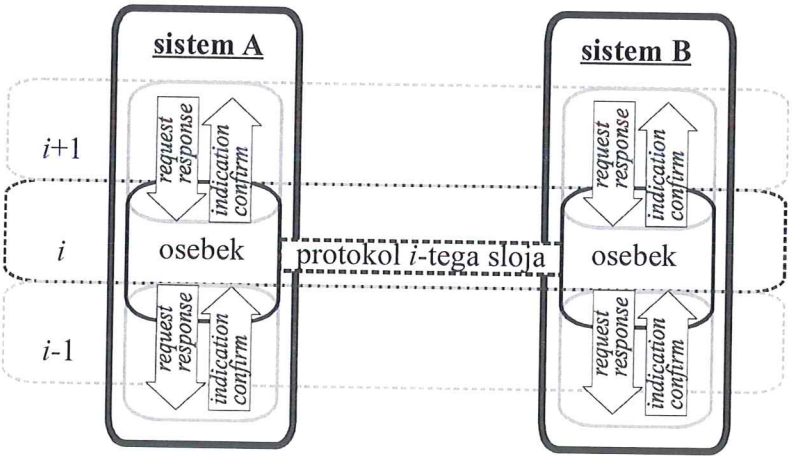


Slika 5-3. Model protokolnega osebka v sloju protokolnega sklada

Če sedaj ponovno pogledamo celoten komunikacijski sistem na sliki **Error! Not a valid link.**, lahko ugotovimo, da imata najvišji in najnižji sloj v nekem smislu poseben položaj. Protokol najvišjega sloja izvaja storitev, ki je storitev celotnega telekomunikacijskega sistema. Uporabnik te storitve je **aplikacija**, ki ni več del telekomunikacijskega sistema, ampak zunanji uporabnik tega sistema. Zato bomo najvišji sloj imenovali **aplikacijski sloj**. V okviru telekomunikacijskega sistema je ta sloj le uporabnik, ne pa tudi ponudnik storitve. Nasprotno pa je najnižji sloj le ponudnik storitve, ne pa tudi uporabnik, saj si protokolna osebka tega sloja neposredno izmenjujeta protokolna sporočila med seboj po fizičnem kanalu; zato bomo najnižji sloj imenovali **fizični sloj**.

Celotno nalogo, ki mu je zaupana, torej opravi protokol sloja *i* le delno sam, preostali del bremena pa poveri sloju *i*-1 tako, da uporabi njegovo storitev. Seveda tudi sloj *i*-1 ne opravi vse svoje naloge sam. Dejansko sodelujejo pri izvajanju storitve sloja *i*-1 vsi sloji,

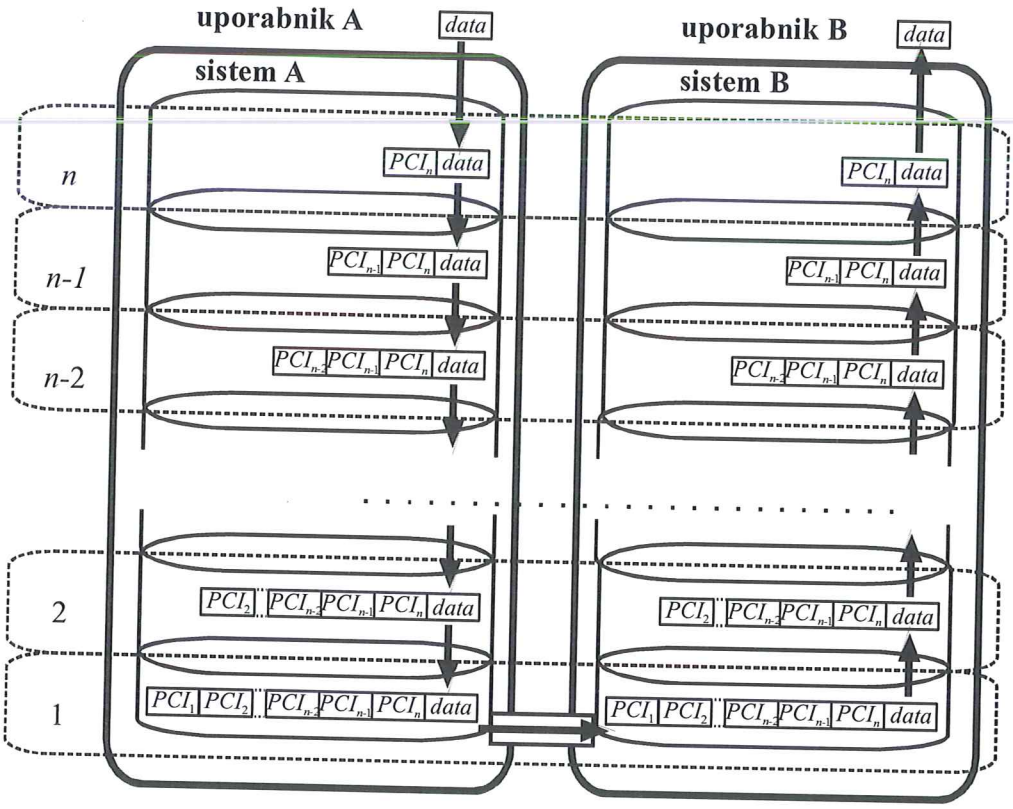
ki so oštevilčeni od 1 do i-1; vendar pa sloj i tega ne ve, saj vidi in pozna le storitev sloja i-1 in nič drugega. Na ta način se naloga celotnega komunikacijskega sistema razdrobi na n delnih nalog; vsakega od teh n delov postori eden izmed n protokolov, ki tako hkrati tečejo v celotnem sistemu. Vendar pa je delovanje vsakega izmed protokolnih osebkov, če ga opazujemo posebej, razmeroma preprosto. Od svojega "delodajalca" (uporabnika) dobiva navodila (v obliki primitivov *request* in *response*) in mu tudi poroča o opravljenem delu (s primitivoma *indication* in *confirm*); prav tako tudi sam daje navodila svojemu ponudniku storitev (v obliki primitivov *request* in *response*) in od njega sprejema poročila (s primitivoma *indication* in *confirm*); s svojim partnerjem pa komunicira po protokolu svojega sloja. Prav ta relativna preprostost komunikacijskega procesa v sloju in njegova neodvisnost od komunikacijskih procesov v drugih slojih pa nam omogoča učinkovito delitev kompleksnega problema na več preprostejših podproblemov. Razslojitev komunikacijskega sistema nam torej poenostavi problem načrtovanja, preizkušanja pravilnosti in vzdrževanja takega sistema. Poleg tega nam omogoča, da med seboj kombiniramo različne protokole v različnih slojih, seveda z omejitvijo, da nižji sloj (npr. i-1) implementira prav tisto storitev, ki jo potrebuje višji sloj (i). Komunikacija med partnerskima osebkoma i-tega sloja ter njuna interakcija z osebkoma sosednjih slojev so prikazani v sliki **Error! Not a valid link.**.



Slika 5-4. Partnerska osebka v sloju protokolnega sklada

Vprašajmo se sedaj še, kako pravzaprav v resnici potuje sporočilo od enega uporabnika do drugega! Vzemimo, da imamo opravka z dvema uporabnikoma in n-slojnim telekomunikacijskim sistemom. Oddajni uporabnik generira uporabniško

sporočilo in ga z ustrezno zahtevo preda protokolnemu osebku v najvišjem (n-tem) sloju oddajnega sistema; le-ta mu doda svojo protokolno nadzorno informacijo in tako oblikuje protokolno sporočilo n-tega sloja, ki ga skupaj z ustrezno zahtevo preda protokolnemu osebku v sloju n-1 istega sistema. Ta mu doda svojo protokolno nadzorno informacijo in tako nastane protokolno sporočilo sloja n-1. Ko dobi to sporočilo oddajni protokolni osebek sloja n-2, mu spet doda svojo protokolno nadzorno informacijo in nastane protokolno sporočilo sloja n-2. Tako originalno uporabniško sporočilo potuje od sloja do sloja navzdol, od aplikacijskega pa vse do fizičnega sloja, v vsakem sloju pa dobi še dodatek - protokolno nadzorno informacijo tega sloja. V fizičnem sloju se uporabniško sporočilo z dodatki vseh slojev vred prenese v sprejemni sistem. Tam pa začne sporočilo potovati navzgor proti aplikacijskemu sloju sprejemnega sistema. Pri tem protokolni osebek v vsakem sloju sprejemnega sistema odvzame protokolno nadzorno informacijo, ki je namenjena njemu, dokler navsezadnje ne prispe originalno uporabniško sporočilo brez dodatkov do sprejemnega uporabnika. Na kratko bi lahko rekli, da vsak sloj na oddajni strani ovije (inkapsulira) protokolno sporočilo višjega sloja v lastno protokolno sporočilo, na sprejemni strani pa protokolno sporočilo višjega sloja izlušči (dekapsulira) iz lastnega protokolnega sporočila. Postopek inkapsulacije in dekapsulacije je torej bistvena sestavina komuniciranja v protokolnem skladu. Opisani način prenosa uporabniškega sporočila ilustrira slika **Error! Not a valid link.**, kjer smo uporabniško sporočilo (storitveno podatkovno enoto sloja n) označili kot *data*.



Slika 5-5. Potovanje sporočila skozi razslojeni telekomunikacijski sistem

Ugotovili smo že, da vsak sloj v protokolnem skladu opravlja svojo nalogo. Lahko bi rekli, da protokol nekega sloja prilagaja storitve nižjega sloja storitvam svojega sloja. Seveda bo ta naloga tem kompleksnejša, čim večja bo razlika med storitvami obeh sosednjih slojev. V primerih, ko je ta naloga še posebej kompleksna, lahko sloj razdelimo na dva ali celo več **podslojev** (ang. **sublayers**); naloga nekaterih izmed njih je poskrbeti za konvergenco enega sloja v drugega. Nekateri podsloji so potem odvisni od nižjega sloja, drugi pa ne.

Postopek (de)multipleksiranja je v povezavi z usmerjanjem informacije seveda vedno vezan na naslavljanje posameznih komunikacijskih osebkov. Povedali smo že, da protokolni osebek, ki informacijo (de)multipleksira, v ta namen kot naslove svojih uporabnikov uporablja oznake (naslove) točk, preko katerih le-ti dostopajo do njegovih storitev. Nek osebek bo torej v celotnem telekomunikacijskem omrežju mogoče naslavljati z zaporedjem naslovov točk dostopa do storitev vseh nižje ležečih slojev, ali pa vsaj z zaporedjem naslovov točk dostopa do storitev tistih slojev, ki multipleksirajo informacijo različnih uporabnikov. Razume se, da mora biti takšno zaporedje naslovov

enoumno znotraj celotnega omrežja. To enoumnost morajo v omrežju uporabljeni protokoli pač zagotavljati.

Organiziranje protokolov v protokolni sklad pa ima še druge prednosti. V skladu namreč lahko uporabimo različne kombinacije protokolov; pri tem pa mora biti izpolnjen pogoj, da se storitve enega protokola ujemajo z zahtevami protokola nad njim. Še več! Ugotovili smo že, da lahko nek izvajalec storitve ponuja storitve več uporabnikom hkrati. Torej lahko v nekem sloju protokolnega sklada poteka komunikacija hkrati med različnimi subjekti po različnih protokolih, vsi ti protokoli pa so lahko uporabniki istega protokola v nižjem sloju. Seveda pa morajo različni protokoli uporabljati različne točke dostopa do storitev protokola v nižjem sloju; protokolni osebek nižjega sloja torej vidi različne protokolne osebke v višjem sloju na različnih naslovih, na katerih nudi svoje storitve. Tako se pogosto zgodi, da lahko v istem telekomunikacijskem omrežju uporabljamo različne kombinacije protokolov hkrati, torej v istem telekomunikacijskem omrežju hkrati uporabljamo več različnih protokolnih skladov. Pa tudi v omrežjih, ki so si po svoji naravi različna, lahko uporabljamo protokolni sklad, v katerem so protokoli višjih slojev enaki, protokoli nižjih slojev pa različni.

Kadar v istem omrežju hkrati uporabljamo več različnih protokolnih skladov za različne naloge, ki pa služijo nekemu skupnemu namenu, govorimo o različnih **ravninah** (ang. **plane**) komuniciranja. Tako v sodobnih omrežjih pogosto uporabljamo **uporabniško ravnino** (ang. **user plane**), v kateri se prenaša uporabniška informacija, **krmilno ravnino** (ang. **control plane**), v kateri se prenaša signalizacija (informacija, potrebna za krmiljenje prenosa uporabniške informacije), in **upravljalsko ravnino** (ang. **management plane**) za upravljanje telekomunikacijskega omrežja samega. Protokolni osebki v protokolnih skladih različnih ravnin takih sistemov lahko med seboj celo do neke mere sodelujejo. Medtem ko s sloji protokolnega sklada razdelimo arhitekturo komunikacijskega sistema v horizontalne plasti, ravnine poskrbijo za vertikalno razdelitev celotnega sistema.

V zgodovini telekomunikacijskih protokolov je bilo definiranih, realiziranih in uporabljanih že kar nekaj protokolnih skladov. V tem poglavju bomo podrobneje opisali dva: to bosta referenčni model za arhitekturo odprtih sistemov OSI in »internetski«

protokolni sklad TCP/IP. Preden pa podrobneje opišemo vsakega izmed njiju, na kratko povzemimo njune poglavitne značilnosti!

- Referenčni model OSI pravzaprav ni protokolni sklad, ampak model za snovanje protokolnih skladov. Zasnovan je bil v organizaciji za standardizacijo ISO predvsem z upoštevanjem dotedanjega razvoja in vedenja o telekomunikacijskih sistemih, omrežjih in protokolih. Zato so teoretične osnove tega sistema čiste, jasne in elegantne. Osnovni pojmi, ki smo jih doslej spoznali o storitvah, protokolih in protokolnih skladih, se naslanjajo predvsem na model OSI in na njegovo terminologijo. Sklad OSI je torej najprej nastal na papirju kot splošen model snovanja in organiziranja telekomunikacijskih protokolov, zato ni bil pisan za konkretne protokole. Šele ko je bil model definiran, so mu začeli prilagajati in v skladu z njim definirati protokole.
- Medtem ko je model OSI nastal v srenji klasičnih telekomunikacij, se je model TCP/IP porodil v računalniških krogih. Po tradiciji so se klasična telekomunikacijska in računalniška omrežja razlikovala po tem, da so dajala prva prednost povezavno orientiranim telekomunikacijam, druga pa nepovezavno orientiranim komunikacijam. Protokolni sklad TCP/IP se je naslonil na konkretne protokole, ki so bili pred tem že praktično preizkušeni oziroma v uporabi in s katerimi je še danes tesno povezan. To so protokoli, ki so jih razvili za omrežje Internet in jih v tem omrežju tudi uporabljamo. Zato je ta sklad bolj praktičen in teoretično manj čist. V primeru sklada TCP/IP so torej bili najprej protokoli, šele nato so jih povezali v sklad.

5.2 Referenčni model OSI

Mednarodna organizacija za standardizacijo ISO je ob koncu sedemdesetih in v začetku osemdesetih let 20. stoletja definirala referenčni model za protokolni sklad ter ga poimenovala **Open Systems Interconnection** ali krajše **OSI**. Referenčni model je bil mišljen kot nekakšen okvir za definiranje in kombiniranje protokolov in naj bi udejanil možnosti za komuniciranje med kompleksnimi sistemi, ki so kombinacija izdelkov različnih proizvajalcev in v upravljanju različnih operaterjev; če takšni sistemi lahko med seboj komunicirajo, jih imenujemo **odprti sistemi**. Edini pogoj, da lahko dva odprta sistema med seboj komunicirata, je torej, da uporabljata iste protokole v vseh slojih, v katerih komunicirata med seboj neposredno (torej po istem partnerskem protokolu).

Nikakor pa seveda ni potrebno, da bi bili ti protokoli implementirani na enak način. Referenčni model OSI ne predpisuje konkretnih protokolov, ki naj se v protokolnem skladu uporabljajo, zato je možno v okviru modela uporabiti različne kombinacije protokolov. Bistvo referenčnega modela je v tem, da predpisuje le storitve, ki naj jih opravljajo protokoli posameznih slojev. Je pa ISO v kasnejših letih definiral tudi nekaj protokolov, ki izvajajo storitve posameznih slojev OSI modela.

Model OSI ima sedem slojev. Spodnji štirje sloji (1÷4) se ukvarjajo s čisto telekomunikacijskimi nalogami, to je s prenosom sporočil med telekomunikacijskimi napravami oziroma uporabniki, in sicer ne glede na to, ali so ti uporabniki povezani med seboj neposredno, skozi omrežje, ali pa celo skozi več med seboj povezanih omrežij. Zgornji trije sloji (5÷7) pa nudijo podporo porazdeljenim aplikacijam, v katerih sodelujejo odprti sistemi. Ti sloji skrbijo za primerno povezovanje in dialog med uporabniki, za varnost, formatiranje in preformatiranje sporočil ter za nudenje nekaterih osnovnih telekomunikacijskih storitev, ko je npr. dostop do oddaljenih računalnikov ali prenos datotek.

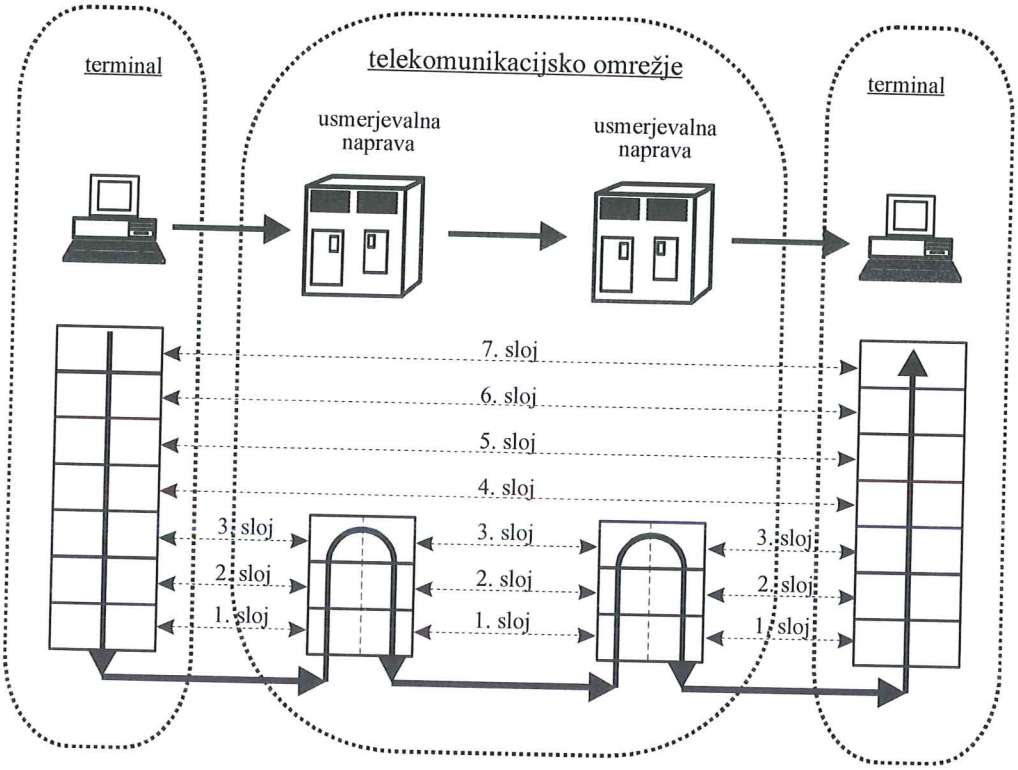
Angleški in slovenski nazivi posameznih slojev ter njihove običajne oznake so podani v tabeli 1.

Tabela 8. Sloji v OSI modelu

sloj	oznaka	angleški naziv	slovenski naziv
7	A	Application layer	Aplikacijski sloj
6	P	Presentation layer	Prikazni sloj
5	S	Session layer	Sejni (Konferenčni) sloj
4	T	Transport layer	Transportni sloj
3	N	Network layer	Omrežni sloj
2	DL	Data-Link layer	Kanalni (Povezavni) sloj
1	Phy	Physical layer	Fizični sloj

Največkrat telekomunikacijski sistem predstavlja telekomunikacijsko omrežje, ki ga sestavljajo posredovalne (usmerjevalne ali komutacijske) naprave in prenosne poti, nanj pa so priključene terminalne in ostale telekomunikacijske naprave. Sporočila torej potujejo od izvornega terminala skozi omrežje (od ene posredovalne naprave do druge) do ponornega terminala. Pri tem povejmo, da se izraz komutacija običajno uporablja za posredovanje v kanalnem (2.) sloju, izraz usmerjanje pa se uporablja za posredovanje v

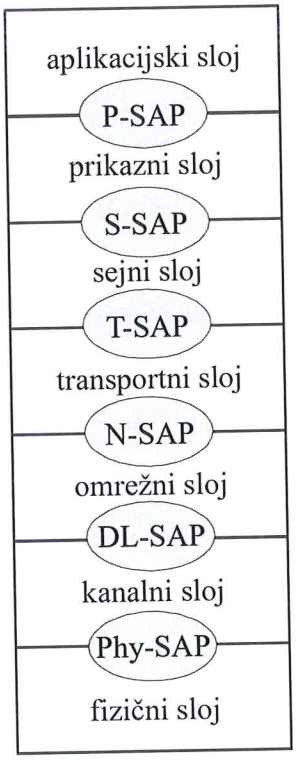
omrežnem (3.) sloju po modelu OSI. Protokoli slojev 1, 2 in 3 so odvisni od vrste telekomunikacijskega omrežja in definirajo komunikacijo med dvema sosednjima telekomunikacijskima napravama, torej takima, ki ju neposredno povezuje med seboj neka fizična prenosna pot, brez vmesnih usmerjevalnih naprav. Pri tem nikakor ni nujno, da bi bili vsi protokoli istega sloja, ki sodelujejo v nekem telekomunikacijskem procesu, isti. Tako lahko med enim terminalom in omrežjem uporabljamo en protokol, med drugim terminalom in istim omrežjem v istem sloju drug protokol, med komunikacijskimi napravami znotraj samega omrežja pa spet v istem sloju še več različnih protokolov. Protokoli v 4., 5., 6. in 7. sloju so neodvisni od vrste omrežja in definirajo neposredno komunikacijo med obema končnima terminaloma. Razmere prikazuje slika **Error! Not a valid link.**.



Slika 5-6. Sloji OSI modela v telekomunikacijskem omrežju

Iz prejšnjega odstavka sledi, da lahko dva odprta sistema komunicirata med seboj, če v zgornjih štirih slojih (4÷7) uporabljata iste protokole, v spodnjih treh slojih pa vsak izmed njiju uporablja protokole, ki so potrebni za komunikacijo z omrežjem, ki ga povezuje z drugim odprtim sistemom.

Med dvema sosednjima slojema je točka dostopa do storitve nižjega sloja, po katerem jo tudi imenujemo, kot lahko vidimo na sliki **Error! Not a valid link.**.



Slika 5-7. Sloji referenčnega modela OSI in točke dostopa do njihovih storitev

V nadaljevanju tega razdelka bomo na kratko opisali posamezne sloje referenčnega modela OSI.

1. **Fizični sloj (Physical Layer)** - Naloga fizičnega sloja je prenos zaporedja binarnih vrednosti s pomočjo elektromagnetnega ali optičnega signala med dvema telekomunikacijskima napravama, ki sta med seboj neposredno povezani. Specifikacija protokola fizičnega sloja določa mehanske lastnosti stičišča med protokolnim osebkom in prenosnim sistemom (obliko in velikost konektorja), elektromagnetne oziroma optične lastnosti signala (implementacijo binarnih vrednosti, modulacijo in linijsko kodo ter prenosno hitrost), funkcionalno specifikacijo posameznih fizičnih delov prenosne poti (npr. funkcijo posameznih kontaktov pri konektorju) ter proceduro (postopek), ki je potrebna za vzpostavitev fizične povezave in za prenos zaporedja binarnih vrednosti. Sem sodi tudi sinhronizacija sprejemnika na sprejem posameznih binarnih vrednosti. Skratka,

fizični sloj nudi višjemu (kanalnemu) sloju storitev prenosa zaporedja binarnih vrednosti.

2. **Kanalni sloj, povezavni sloj (Data Link Layer)** – Protokol kanalnega sloja nudi višje ležečemu omrežnemu sloju storitev prenosa paketov med dvema sosednjima omrežnima napravama. Poskrbi za formiranje protokolne podatkovne enote kanalnega sloja, ki jo imenujemo **okvir**. Protokol drugega sloja skrbi tudi za sinhronizacijo oziroma izločanje okvirjev iz sprejetega zaporedja binarnih vrednosti pri sprejemu. Zaradi šuma in motenj je prenos binarnih vrednosti, ki ga nudi fizični sloj, nezanesljiv. Osnovna naloga kanalnega sloja je zato detekcija napak v okvirjih, do katerih pride v fizičnem sloju. V primeru povezavno orientiranega protokola v kanalnem sloju le-ta napake tudi popravlja. V slednjem primeru kanalni sloj jamči, da bodo prišli paketi omrežnega sloja brez napak in v pravem vrstnem redu na cilj (v realnem svetu sicer ni stvari, ki bi bile 100% zanesljive, vendar pa je prenos sporočil skozi kanalni sloj za precej velikostnih redov zanesljivejši od prenosa skozi fizični sloj). Kanalni sloj lahko torej pred omrežnim slojem skrije poglavitno pomanjkljivost fizičnega sloja. Protokol kanalnega sloja tudi vzpostavlja navidezne zveze skozi kanal 2. sloja, skrbi za (de)multipleksiranje, če je v tem kanalu več navideznih zvez, in izvaja krmiljenje pretoka podatkov med sosednjima telekomunikacijskima napravama.
3. **Omrežni sloj (Network Layer)** – Omrežni sloj ponuja višje ležečemu transportnemu sloju storitev vzpostavitve in vzdrževanja zveze med terminalnima napravama (v primeru povezavno usmerjene komunikacije) oziroma dostave datagrama naslovniku (v primeru nepovezavno usmerjene komunikacije). Za vsako navidezno zvezo med uporabnikoma pri povezavno orientiranih komunikacijah oziroma za vsak datagram pri nepovezavno orientiranih komunikacijah je namreč treba najti ustrezno pot med terminaloma skozi omrežje. Postopek iskanja poti skozi omrežje imenujemo usmerjanje (ang. **routing**), ki pa je po referenčnem modelu OSI temeljna naloga protokolov omrežnega sloja. V primeru, ko sta terminala priključena na različni omrežji, ki sta med seboj neposredno ali posredno povezani, pa mora omrežni sloj poskrbeti še za iskanje poti skozi povezave med omrežji. Problem, ki je neposredno povezan z usmerjanjem, je **naslavljanje** terminalov ter omrežij in njihovih elementov

oziroma opremljanje sporočil z naslovi. Pomembna naloga protokolov tega sloja pa je tudi krmiljenje pretoka podatkov, ki naj v sodelovanju z upravljalskim sistemom omrežja preprečuje zgostitve (ang. **congestion**) sporočil v nekaterih delih omrežja. Protokol omrežnega sloja mora biti prisoten tako v terminalih, kot tudi v vseh komutacijskih napravah. Omrežni sloj skrije pred protokoli transportnega sloja posebnosti in probleme omrežja ali omrežij, skozi katera komunikacija dejansko poteka. Transportni sloj ima tako vtis, da sta oba terminala neposredno povezana med seboj.

4. **Transportni sloj (Transport Layer)** - Transportni sloj je najvišji sloj, ki se ukvarja s klasično telekomunikacijsko funkcijo – s prenosom sporočil med osebki višjega sloja. Tako transportni sloj nudi višje ležečemu konferenčnemu sloju storitev prenosa sporočil med končnimi napravami ne glede na to, kakšno omrežje za ta prenos uporablja. Transportni sloj je najnižji sloj, katerega protokoli se ukvarjajo z izmenjavo sporočil le med dvema končnima uporabnikoma (terminaloma) telekomunikacijskega sistema (ang. **end-to-end**). Čeprav pogosto že protokoli 2. sloja poskrbijo, da na posameznih delih komunikacijske poti ne prihaja do izgub sporočil, lahko pride do izgub v omrežju, bodisi zaradi polnih čakalnih vrst ali pa iz kakšnih drugih vzrokov. Zato je lahko ena izmed nalog transportnega sloja, da skrbi za to, da pridejo na cilj vsa sporočila, in to v pravem vrstnem redu. Skrbi tudi za krmiljenje pretoka podatkov. Posamezne vrste telekomunikacijskih omrežij iz raznih razlogov omejujejo dolžino sporočil. Zato bo protokol transportnega sloja na oddajni strani sporočilo, ki je daljše, kot ga dovoljuje omrežje, razdelil na več delov, vsakega izmed njih posebej predal omrežnemu sloju za prenos k naslovniku, transportni osebek na sprejemni strani pa bo te dele sporočila spet sestavil nazaj v prvotno sporočilo. Tovrstni postopek imenujemo **drobljenje** (**fragmentiranje**, **segmentiranje**) sporočil. Tako skrbi transportni sloj za nekakšno dolžinsko prilagoditev sporočil med višjimi sloji protokolnega sklada in nižjimi sloji, ki so značilni za uporabljeno telekomunikacijsko omrežje. Po potrebi skrbi protokol transportnega sloja tudi za (de)multipleksiranje sporočil različnih uporabnikov (udeležencev različnih sej v konferenčnem sloju) v isti transportni kanal. Skratka, transportni sloj skriva posebnosti uporabljenega telekomunikacijskega omrežja pred aplikacijskimi procesi,

ki to omrežje posredno uporabljajo. Zato transportni sloj potrebuje tem bolj kompleksen protokol, čim manj kvaliteten in zanesljiv je prenos sporočil skozi uporabljeno telekomunikacijsko omrežje. Z izbiro primernega protokola v transportnem sloju lahko torej načrtovalec celotnega sistema (arhitekt sistema) poskrbi za ustrezno kvaliteto telekomunikacijskih storitev ne glede na kvaliteto storitev, ki jo nudi samo omrežje. Čim manjšo kvaliteto storitev nudi telekomunikacijsko omrežje, tem kompleksnejši bo protokol transportnega sloja!

5. **Sejni sloj (Session Layer)** - Sejni sloj je prvi sloj, ki se ukvarja bolj s porazdeljeno aplikacijo, kot pa s komunikacijskim problemom. Protokoli konferenčnega sloja se ukvarjajo z organizacijo dialoga med aplikacijami v terminalnih sistemih in z nadzorom nad tem dialogom. Torej protokol tega sloja poskrbi za vzpostavitev, vzdrževanje in zaključek dialoga med aplikacijskimi osebki. V okviru tega dialoga lahko protokoli konferenčnega sloja vsklajujejo izmenično enosmerno izmenjavo transakcijskih enot med aplikacijskimi osebki. Konferenčni sloj lahko tudi občasno registrira stanje komunikacijskega procesa in s tem ustvari pogoje za definirano obnovitev in nadaljevanje porazdeljene aplikacije v primeru težav (npr. izpada nekega sistema). Tovrstnemu definiranemu stanju porazdeljene aplikacije rečemo sinhronizacijska točka. Ne navsezadnje pa lahko sejni sloj tudi poroča višjima slojema o morebitnih izjemnih stanjih (npr. nepopravljivih napakah), do katerih lahko pride med dialogom aplikacijskih osebkov.
6. **Prikazni sloj (Presentation Layer)** - Prikazni sloj skrbi za format podatkov, ki si jih izmenjujeta aplikaciji in za pretvorbo med posameznimi formati. Porazdeljena aplikacija namreč teče na več različnih računalnikih hkrati, le-ti pa razne vrste podatkov shranjujejo v različnih internih formatih. Podatke lahko sicer na abstrakten način predstavimo z njihovo **abstraktno sintakso**, ki je podobna definiciji podatkovnih tipov, kakršne uporabljamo v visokih programskih jezikih (npr. C, C++, Java, Pascal...) – v takih jezikih pa so aplikacije tudi napisane. Zaradi različnih internih formatov pa se kljub dobro znani abstraktni sintaksi prenešenih podatkov lahko zgodi, da le-teh obe aplikaciji ne bosta interpretirali na enak način. Znan primer različnih internih formatov je označevanje konca vrstice v tekstovnih datotekah na sistemih z operacijskim sistemom Unix (z znakom "newline") ali Windows (z

znakoma "Carriage Return" in "Line Feed"). Drugi takšen primer je vrstni red zapisa zaporedja zlogov, ki tvorijo pomensko celoto (npr. celoštevilsko vrednost), v spomin (nekateri računalniki, tako imenovani "big endians", zapišejo zlog z največjo težo na najnižji naslov v spominu in zlog z najmanjšo težo na najvišji naslov v spominu, računalniki, znani tudi kot "little endians", pa ravno obratno; tako bi 32-bitno vrednost, zapisano v šestnajstiškem sistemu kot 89ABCDEF, zapisali na naslove 0101, 0102, 0103 in 0104 v vrstnem redu 89, AB, CD, EF v računalniku "big endian" oziroma v vrstnem redu EF,CD,AB,89 v računalniku "little endian"[*]). Zato je za abstraktno sintakso podatkov, ki jo uporabljata aplikaciji, treba definirati še **prenosno** oziroma **konkretno sintakso**, ki jo uporabljamo pri prenosu teh podatkov preko omrežja. Pri prenosu en protokolni osebek prikaznega sloja po potrebi pretvarja interni format računalnika, na katerem deluje, v prenosno sintakso, drugi prikazni osebek pa prenosno sintakso v interni format drugega računalnika. Prikazni sloj se lahko ukvarja tudi s kompresijo in šifriranjem prenašanih podatkov.

7. **Aplikacijski sloj (Application Layer)** - Aplikacijski sloj nudi porazdeljenim aplikacijam, ki vključujejo transakcije in operacije na oddaljenih sistemih, le-te pa potekajo preko telekomunikacijskega sistema, nekatere splošno uporabne in/ali specifične storitve, kot so npr. prenos datotek in upravljanje z njimi, proženje aplikacij na drugih sistemih (RPC), overovitev sogovornika, usklajevanje in osveževanje podatkov v porazdeljeni podatkovni bazi, sinhronizacija dostopa do skupnih virov, elektronska pošta ali terminalski dostop do oddaljenega računalnika. Tako ima aplikacija do neke storitve dostop, ki ni odvisen od tega, ali ji to storitev nudi proces na lokalnem ali oddaljenem računalniku. Protokole aplikacijskega sloja imenujemo tudi **aplikacijski storitveni elementi** (ang. **Application Service Elements, ASE**). Včasih jih delimo na splošne aplikacijske storitvene elemente (ang. common application service elements) in posebne aplikacijske storitvene elemente (ang. application-specific service elements).

Ob referenčnem modelu OSI se moramo še posebej pomuditi pri **lokalnih omrežjih (Local Area Network - LAN)** in **metropolitanskih omrežjih (Metropolitan Area**

[*] Pri poimenovanju obeh vrst računalnikov so se zgledovali pri imenih dveh političnih strank v deželi Liliput [Jonathan Swift, *Gulliver's Travels*], pripadniki katerih so razbijali jajca na različnih koncih.

Network - MAN). Posebnost teh omrežij v klasični izvedbi* je namreč v tem, da so vsi komunicirajoči sistemi povezani z eno samo prenosno potjo, skupnim prenosnim medijem. Pri tej vrsti omrežij torej obstaja še en dodaten problem - to je problem koordinacije dostopa do skupnega medija. Zato pri omrežjih LAN/MAN razdelijo kanalni sloj na dva podsloja. Oba sta definirana v standardih serije IEEE 802.*x*, ki jih je povzela tudi Mednarodna organizacija za standardizacijo ISO.

A. **Podsloj za dostop do medija** (**Medium Access Control - MAC**) - Ta podsloj nadzoruje dostop do medija, zlasti pri oddajanju. Protokol tega sloja tudi poskrbi za konvergenco med fizičnim slojem in klasičnim kanalnim slojem ter je, tako kot fizični sloj, specifičen za vsako vrsto lokalnega omrežja posebej.

B. **Podsloj za krmiljenje logičnega kanala** (**Logical Link Control - LLC**) - Ta podsloj opravlja klasične funkcije kanalnega sloja, ne da se bi mu bilo pri tem treba ozirati na metodo dostopa do medija. LLC podsloj je torej neodvisen od dostopa do medija in je enak pri vseh vrstah lokalnih omrežij. Lahko pa omrežnemu sloju nudi tri različne tipe storitev: povezavno orientirani prenos, nepovezavno orientirani prenos s potrditvami ali nepovezavno orientirani prenos brez potrditev. Za eno izmed teh storitev se odločimo glede na protokole, ki jih uporabljamo v nižjih in višjih slojih.

V protokolnem skladu, ki ga oblikujemo po referenčnem modelu OSI, lahko uporabljamo multipleksiranje v več slojih. Naslov aplikacije v omrežju zato podaja **stik** (**konkatenacija**) naslovov točk dostopa do storitev vseh teh slojev. Torej morajo biti ti naslovi zapisani v nadzornih protokolnih informacijah protokolov, ki pripadajo tistim slojem, v katerih izvajamo multipleksiranje. Sloji modela OSI, v katerih najpogosteje izvajamo multipleksiranje, so kanalni, omrežni in transportni sloj.

Referenčni model OSI je kompleksen sistem. Kot pa smo že omenili, ni potrebno, da bi nek protokol moral implementirati vse storitve, ki jih sloj, v katerega ta protokol spada, definira. Celo ni potrebno, da bi nek komunikacijski sistem uporabil vse sloje modela. Mnogi kritiki OSI modela navajajo, da se nekatere funkcije ponovijo v več slojih, kar je delno tudi res. Sloja, ki ju največkrat izpuščajo, sta sejni in prikazni, saj probleme, ki naj bi jih reševala ta dva sloja, pogosto rešuje aplikacijski sloj ali pa celo kar aplikacije same. Če gre za komunikacijo med dvema uporabnikoma, ki nista povezana skozi omrežje,

* Danes namreč tudi v tovrstna omrežja že vpeljujejo postopke komutacije.

ampak neposredno, potem seveda ni potreben omrežni sloj. Pri nekaterih novejših telekomunikacijskih sistemih, ki temeljijo na tehnologiji optičnih vlaken, pa včasih celo združijo funkcije omrežnega in kanalnega sloja, saj je pri prenosu informacije z optičnimi signali možno doseči za vsaj dva velikostna razreda manjšo pogostnost napak kot pri prenosu s klasičnimi elektromagnetnimi signali.

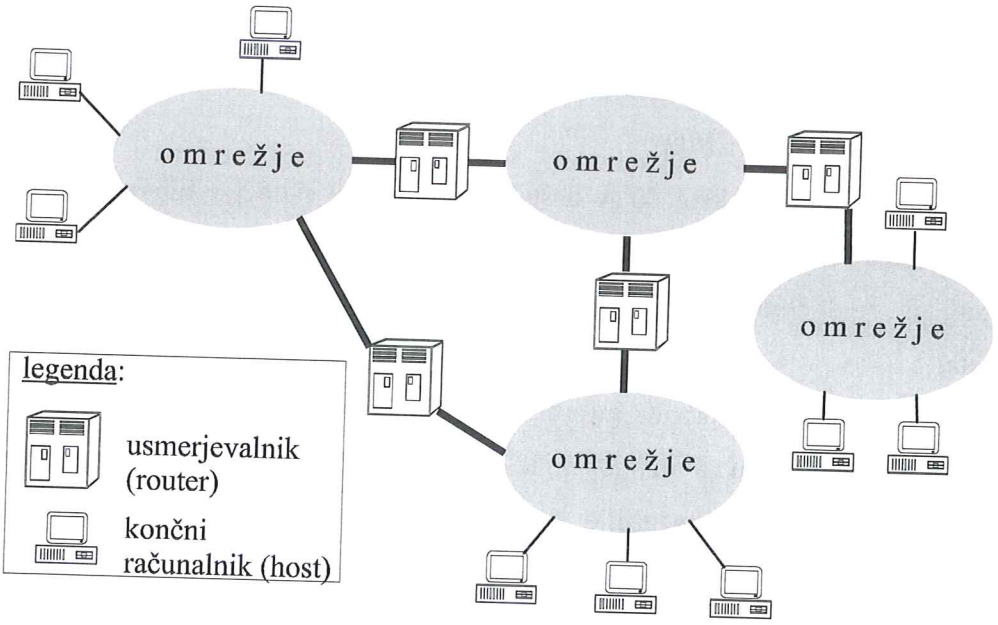
5.3 Protokolni sklad TCP/IP

TCP/IP je protokolni sklad, ki je nastal in se razvijal skupaj z Internetom. Zato so značilnosti tega protokolnega sklada soodvisne z značilnostmi omrežja Internet. Torej bo umestno, da pred obravnavo protokolnega sklada TCP/IP opišemo nekaj značilnosti Interneta.

Internet je omrežje za prenos informacije na paketni način, ki povezuje skoraj vse dežele sveta in je donedavna omogočalo le podatkovne komunikacije, v zadnjem času pa tudi prenos ostalih vrst informacije. Sprva so ga uporabljale predvsem raziskovalne organizacije, sedaj pa vse bolj tudi komercialne. Prvi zametek tega omrežja je predstavljalo omrežje ARPANET, ki je služilo za povezavo raziskovalnih organizacij, vključenih v obrambne projekte Agencije za napredne raziskovalne projekte ameriškega obrambnega ministrstva DARPA (Department of defense Advanced Research Projects Agency). Kasneje se je med seboj povezovalo vedno več različnih omrežij, delujočih na osnovi različnih protokolov; kljub temu pa lahko uporabniki, priključeni na različna omrežja, komunicirajo med seboj na osnovi posebnih, za Internet značilnih protokolov.

Za Internet lahko torej rečemo, da je omrežje omrežij. Ta različna omrežja povezujejo med seboj omrežni elementi (računalniki), katerih naloga je ne le povezovanje različnih omrežij in prevajanje med protokoli, značilnimi za posamezna omrežja (gre za protokole nižjih slojev), ampak predvsem za usmerjanje paketov od izvornega končnega uporabnika prek večjega ali manjšega števila teh elementov k ponornemu končnemu uporabniku. Te omrežne elemente imenujemo **usmerjevalniki** (ang. **routers**). **Končni računalniki**, to so računalniki uporabnikov Interneta ali ponudnikov storitev prek

interneta* (ang. **host computers** ali krajše **hosts**), pa so priključeni na posamezna omrežja. Primer take topologije kaže slika 5-8.



Slika 5-8. Primer topologije Interneta

Protokolni sklad, ki se uporablja v omrežju Internet, se razlikuje od OSI referenčnega modela in ima definirane le štiri sloje. Ti so naslednji.

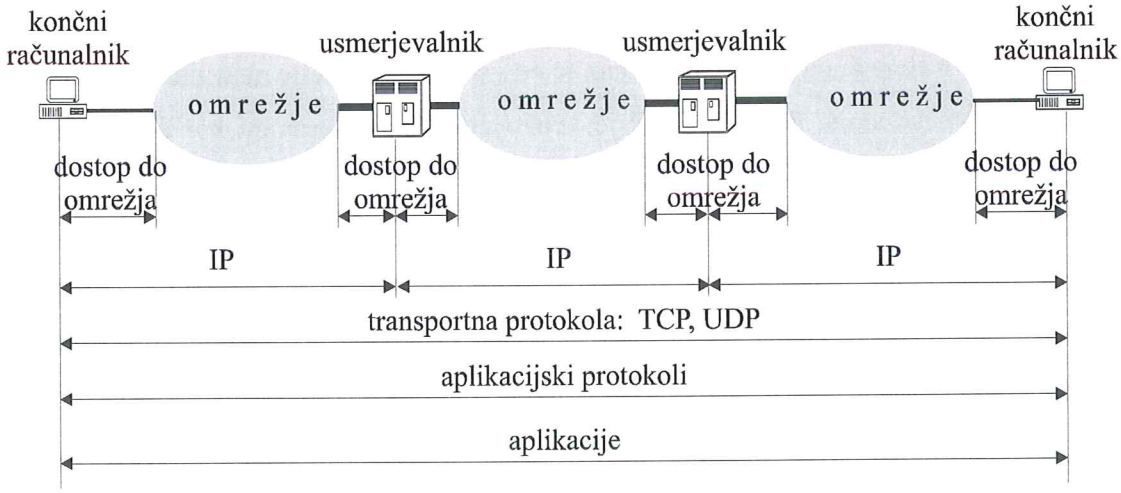
1. **Sloj za dostop do omrežja** (**Network access layer**) omogoča dostop končnega računalnika ali usmerjevalnika do omrežja. S stališča internetske arhitekture gre tu za sloj protokolov, ki jih internetski standardi zvečine ne definirajo, v resnici pa to ni en sloj, ampak več nižjih slojev (od fizičnega do omrežnega po modelu OSI), ki delujejo v skladu z ustreznimi sloji konkretnega omrežja. Protokoli v tem sloju so torej odvisni od vrste omrežja; pogosto uporabljajo lokalno/metropolitansko omrežje – LAN/MAN, javno podatkovno omrežje X.25, javno telefonsko omrežje z nekaterimi dodatnimi protokoli 2. in delno 3. sloja po modelu OSI (npr. PPP), ISDN, omrežje z blokovnim posredovanjem (Frame Relay), ATM itd. Ti protokoli so torej neodvisni od ostalih internetnih protokolov.

* Pri tem ne mislimo na storitve, ki jih en sloj nudi višjemu sloju, ampak na tiste, ki jih strežniki nudijo svojim odjemalcem v smislu komunikacije odjemalec-strežnik, pri čemer sta tako odjemalec kot strežnik aplikaciji, ki uporabljata Internet le kot medij za medsebojno komunikacijo.

2. **Internetni sloj (Internet layer)** skrbi za usmerjanje informacije skozi Internet od enega končnega računalnika do drugega glede na naslov izvora in naslov ponora informacije, torej za usmerjanje informacije med omrežji. V tem sloju se uporablja protokol **IP (Internet Protocol)**, ki deluje med končnim računalnikom in usmerjevalnikom oziroma med dvema usmerjevalnikoma v Internetu. IP je datagramski protokol. Zato je omrežje Internet precej neobčutljivo na okvare oziroma izpade posameznih omrežij in usmerjevalnikov (kar je bil tudi namen prvotnih snovalcev ARPANET-a, ki je bil v bistvu vojaško orientirano omrežje), po drugi strani pa IP sloj ne nudi višjemu (transportnemu) sloju storitve zanesljivega prenosa informacije. Protokol IP definira tudi naslavljanje; tako je vsakemu priključku na omrežje Internet dodeljen **IP naslov** oziroma tako imenovana **IP številka**. Če ima nek računalnik več priključkov na Internet (to velja npr. za usmerjevalnike), ima za vsak priključek tudi svojo IP številko. V internetnem sloju deluje tudi protokol **ICMP (Internet Control Message Protocol)**, čeprav se ICMP datagrami prenašajo kot uporabniška sporočila v IP datagramih. Naloga protokola ICMP je krmiljenje in nadzor nad protokolom IP.
3. **Transportni sloj (Transport Layer)** skrbi za prenos informacije neposredno med dvema končnima računalnikoma, ki med seboj komunicirata prek Interneta. V tem sloju se alternativno uporabljata dva protokola: **TCP (Transport Control Protocol)** je povezavno orientiran, **UDP (User Datagram Protocol)** pa nepovezavno orientiran protokol. Katerega od obeh uporabimo, je odvisno od aplikacije oziroma od protokola aplikacijskega sloja. Protokol TCP je veliko pogosteje v uporabi kot UDP, zato se tudi protokolni sklad Interneta imenuje TCP/IP. Bralec se bo podrobneje seznanil s protokolom TCP v enem od kasnejših poglavij. V transportnem sloju lahko multipleksiramo prenos podatkov med različnimi končnimi računalniki in procesi v njih. Naslov točke dostopa do storitve transportnega sloja v Internetu, pa naj si bo uporabljen protokol TCP ali UDP, imenujemo **vrata** (ang. **port**).
4. V **aplikacijskem sloju (Application layer)** se uporablja vrsta protokolov, ki nudijo neposredno podporo porazdeljenim aplikacijam na Internetu. Tu bomo omenili le nekaj izmed njih. Aplikacijski protokol **telnet** omogoča uporabniku enega računalnika, da prek Interneta dela na nekem drugem računalniku tako, kot da bi bil

terminal uporabnika neposredno povezan z oddaljenim računalnikom. Delo poteka v alfanumeričnem načinu, uporabnik in operacijski sistem oddaljenega računalnika torej komunicirata med seboj s pomočjo nizov znakov (črk, številk, ločil in posebnih znakov). **FTP (File Transfer Protocol)** je protokol, ki omogoča manipuliranje z datotekami na oddaljenem in lokalnem računalniku ter prenos datotek v obeh smereh, seveda pri pogoju, da ima uporabnik za to potrebna dovoljenja. Protokol **TFTP (Trivial File Transfer Protocol)** omogoča prenos datotek z oddaljenega na lokalni računalnik ali obratno na preprost način in z minimalno obremenitvijo virov (procesorja in spomina). Elektronska pošta je sistem, ki uporablja protokol aplikacijskega sloja **SMTP (Simple Mail Transfer Protocol)** za pošiljanje elektronskih pisem v poštni predal in protokol aplikacijskega sloja **POP3 (Post Office Protocol**, version **3)** za dostop do poštnega predala. Najbrž pa je (vsaj poleg elektronske pošte) trenutno najbolj znana in priljubljena aplikacija v Internetu brskanje po omrežju informacijskih virov **svetovnem spletu (World Wide Web - WWW)**; prenos spletnih dokumentov omogoča protokol aplikacijskega sloja **HTTP (HyperText Transfer Protocol)**. Vsi našteti aplikacijski protokoli delujejo na principu odjemalec - strežnik (client - server).

Slika **Error! Not a valid link.** ponazarja področje delovanja protokolov posameznih slojev v skladu TCP/IP.



Slika 5-9. Področje delovanja protokolov v protokolnem skladu TCP/IP

Vsak aplikacijski protokolni osebek in z njim vred pripadajoči aplikacijski proces v Internetu lahko povsem nedvoumno označimo z dvojico naslovov IP številka – številka

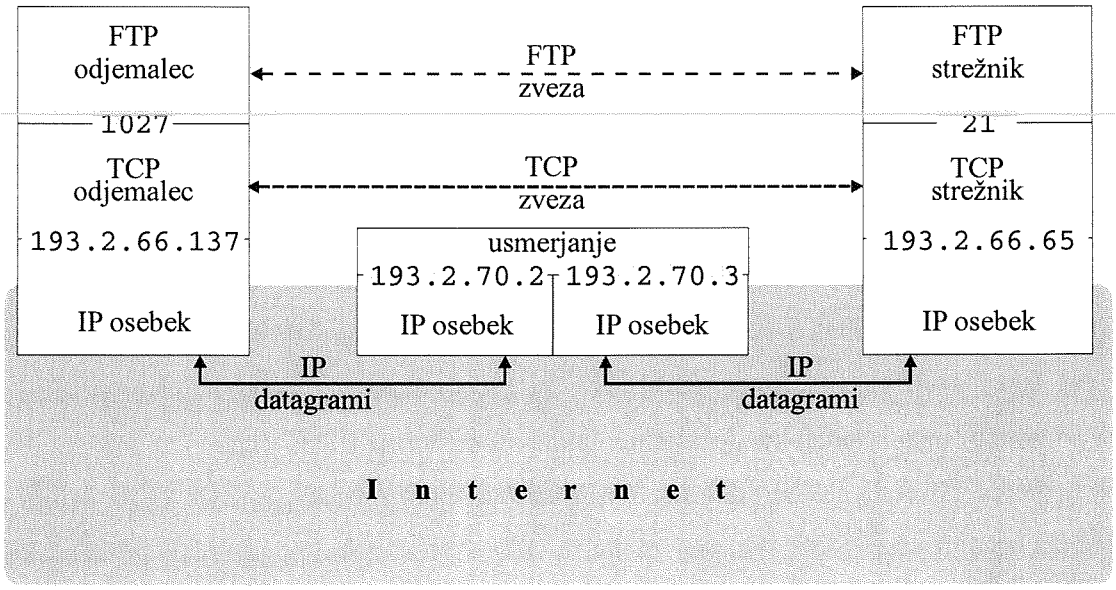
vrat. To dvojico imenujemo **vtičnica** (ang. **socket**). Komunikacija med dvema aplikacijskima procesoma (ki lahko poteka bodisi preko vzpostavljene TCP zveze ali z izmenjavo UDP datagramov) je torej natančno določena z dvema vtičnicama oziroma paroma IP številka – številka vrat na obeh računalnikih. Pri tem ni nepomembno dejstvo, da velika večina internetskih komunikacij poteka po principu odjemalec – strežnik, vsako komunikacijo pa praviloma vedno začne odjemalec, ki mora torej ob tej priliki poznati vtičnico strežnika. Zato strežniki uporabljajo tako imenovane **vnaprej znane številke vrat** (ang. **well known port numbers**), ki pa so različne za različne aplikacijske protokole in imajo dodeljene decimalne vrednosti med 1 in 1023. Vnaprej znane številke vrat določa posebna služba (Internet Assigned Numbers Authority - IANA), ki deluje v okviru organizacije ISOC. Vnaprej znane številke vrat za nekaj pogostejših aplikacijskih protokolov podaja tabela 2.

Tabela 9. Vnaprej znane številke vrat

vnaprej znana številka vrat	*aplikacijski protokol*	*transportni protokol*
21	FTP	TCP
23	telnet	TCP
25	SMTP	TCP
69	TFTP	UDP
80	HTTP	TCP
110	POP3	TCP

Aplikacijski protokolni osebek klienta, ki želi začeti komunicirati s strežnikom, mora v ta namen najprej od transportnega osebka pridobiti začasno vtičnico in z njo vred začasno številko vrat (ephemeral port number). Vtičnica odjemalca je začasna, ker je dodeljena le za čas uporabe, medtem ko vtičnica strežnika obstaja toliko časa, kolikor časa strežnik obratuje*. Na sliki **Error! Not a valid link.** lahko vidimo primer komunikacije med FTP odjemalcem (z IP številko 193.2.66.137 in začasno številko vrat 1027) ter FTP strežnikom (z IP številko 193.2.66.65 in vnaprej znano številko vrat 21).

* Strežniki običajno obratujejo neprekinjeno – 24 ur na dan, 365 dni v navadnem letu in 366 v prestopnem, razen seveda, kadar so v okvari.



Slika 5-10. Primer komunikacije med FTP strežnikom in FTP odjemalcem

5.4 Primerjava referenčnega modela OSI in protokolnega sklada TCP/IP ter njuna kritična presoja

Čeprav sta referenčni model OSI in protokolni sklad TCP/IP na prvi pogled morda precej različna, lahko vendarle najdemo med njima veliko pomembnih podobnosti. To dejstvo po eni strani kaže na kljub razmeroma neodvisnemu razvoju obeh modelov nujno potrebnost razslojevanja protokolnih arhitektur in na povsem naravno izbiro nekaterih slojev, ki se pojavljajo v obeh modelih, po drugi strani pa so te podobnosti tudi posledica medsebojnih vplivov, do katerih je kljub različni filozofiji prihajalo pri sicer sočasnem razvoju obeh modelov. Prva izmed podobnosti je prav princip razslojevanja komunikacijskega sistema, ki nam da večplastno arhitekturo protokolov. Pa tudi v sami razslojenosti lahko najdemo pomembno skupno značilnost. Pri obeh arhitekturah lahko identificiramo tri pomembne skupine slojev: sloje za komunikacijo skozi omrežje oziroma z omrežjem, ki so od tega omrežja pomembno odvisni (sloji 1÷3 v OSI in sloj dostopa do omrežja v TCP/IP), sloj za komunikacijo med končnima uporabnikoma (transportni sloj) in sloj(e), ki podpira(jo) porazdeljene aplikacije uporabnikov (sloji 5÷7 v OSI in aplikacijski sloj v TCP/IP). Vsem tem slojem se v skladu TCP/IP pridruži še dodatni sloj za povezovanje omrežij, ki je specifičen za Internet kot omrežje omrežij

(internetski sloj). Slika **Error! Not a valid link.** prikazuje tovrstno primerjavo med modeloma OSI in TCP/IP.

Slika 5-11. Primerjava referenčnega modela OSI in protokolnega sklada TCP/IP

6 ZMOGLJIVOST PROTOKOLOV

6.1 Uvod v teorijo prometa

Prav tako, kot ima ponudnik storitve svoje uporabnike, ima svoje uporabnike telekomunikacijski kanal ali kakršna koli druga naprava, uporabnike pa ima tudi telekomunikacijsko omrežje kot celota. Ugotovili smo že, da je osnovna naloga telekomunikacijskih naprav in omrežij prenos informacije. Nekateri uporabniki to informacijo porajajo, drugi jo sprejemajo. Pri podatkovnih omrežjih, ki nas v tem delu zanimajo, informacijo prenašamo v obliki **sporočil** (**paketov**), ki se prenašajo skozi telekomunikacijske naprave in omrežje, omrežne naprave pa jih lahko tudi obdelujejo. V tem delu nas bo zvečine zanimal asinhroni prenos podatkov. Torej so paketi lahko različno dolgi, med paketi, ki se prenašajo po istih prenosnih poteh ali se obdelujejo v istih napravah, pa so lahko različno dolgi časovni presledki. Neka telekomunikacijska naprava torej pri paketnem načinu prenosa ni ves čas zasedena; to pa pomeni, da je lahko bolj ali manj izkoriščena. Če gledamo celotno omrežje, lahko tudi zanj rečemo, da je bolj ali manj izkoriščeno. Uporabniki lahko na celotno telekomunikacijsko omrežje ali na njegov del gledajo kot na abstrakten kanal, v katerega lahko oddajajo ali iz katerega lahko sprejemajo informacijo s hitrostjo, ki je navzgor omejena; omrežje torej nudi uporabniku neko hitrost prenosa informacije. Po drugi strani pa tudi vemo, da vsak paket potrebuje določen čas, da se prenese skozi komunikacijsko napravo ali omrežje; naprava oziroma omrežje torej informacijo zakasni. Kot smo v tem delu že videli, fizikalne zakonitosti sistema določajo nek minimalen čas, ki je potreben za prenos paketa oziroma informacije skozi omrežje. Drugi parametri sistema, kot so lastnosti uporabljenih naprav in protokolov ali obremenitev omrežja, pa določajo, koliko večje so dejanske zakasnitve od minimalnih.

Uporabnik bo torej kvaliteto omrežja ocenjeval glede na to, kakšno hitrost prenosa mu le-to nudi in kakšne so zakasnitve informacije v njem*. Po drugi strani pa bo upravljalec omrežja tem bolj zadovoljen, čim bolj bo omrežje izkoriščeno. Pogosto pa so si prizadevanja za čim boljšo kvaliteto omrežja s stališča uporabnikov oziroma za čim večjo

* V tem delu se naslanjamo na v klasičnih podatkovnih omrežjih običajno uporabljena merila za kvaliteto storitev. V sodobnih omrežjih, ki so sposobna na isti telekomunikacijski infrastrukturi integrirati prenos najrazličnejših tipov informacije, pa dobiva pojem kvalitete storitev (ang. Quality of Service) vse širši pomen, uveljavljajo pa se tudi nova merila za njeno vrednotenje.

izkoriščenost omrežja s stališča upravljalca omrežja v medsebojnem nasprotju. Tako kvaliteta prenosa informacije kot tudi izkoriščenost omrežja sta odvisna tudi od dejanskega pretoka informacije, ki ga generirajo uporabniki, in od virov, s katerimi omrežje razpolaga. Da pa bi lahko načrtovalec omrežja predvidel prenosne lastnosti pri različnih obremenitvah omrežja, potrebuje modele, ki omogočajo vrednotenje zmogljivosti oziroma prometnih lastnosti omrežja in njegovih elementov gleda na dejanski pretok informacije in ostale okoliščine. S takimi modeli se ukvarjata **teorija prometa** in **teorija čakalnih vrst**.

Povedali smo že, da so lahko paketi različno dolgi, različno dolgi pa so lahko tudi presledki med njimi. Kakšne so torej te dolžine? Pakete generirajo uporabniki. Vzemimo, da vsak uporabnik zase ve, kdaj bo generiral naslednji paket in kako dolg bo (pogosto tudi to ne drži). Če je uporabnikov omrežja mnogo, pa bo omrežje porajanje vseh teh paketov kljub temu videlo kot nek **naključen** (**stohastičen**) **proces**. To pomeni, da iz zgodovine porajanja paketov ne bo mogoče zanesljivo napovedati niti tega, kdaj se bo pojavil naslednji paket, niti tega, koliko paketov se bo porodilo v naslednji časovni enoti, pa tudi ne, kako dolgi bodo. Preteklo dogajanje je pri takem procesu mogoče le statistično ovrednotiti, na podlagi tega pa je potem mogoče sklepati na verjetnost določenih dogodkov v prihodnosti. Osnovni matematični orodji teorije prometa in teorije čakalnih vrst sta torej **statistika** in **teorija verjetnosti**. Pri naključni spremenljivki (kot je npr. dolžina paketa ali njegova zakasnitev) nas bo zanimala njena povprečna vrednost in porazdelitev (to je gostota verjetnosti različnih odmikov dejanskih vrednosti od povprečne vrednosti. Glede na čase pojavljanja zaporednih dogodkov (npr. generacije in trajanje zaporednih paketov) lahko ločimo različne vrste naključnih procesov. Eno skrajnost predstavlja deterministične proces, kjer je mogoče bodoče dogajanje napovedati; ta proces torej ni naključen. Na drugi strani pa imamo popolnoma naključen proces, kjer je nek dogodek povsem neodvisen od ostalih dogodkov, časi med zaporednimi dogodki pa so eksponentno porazdeljeni. Tak proces imenujemo Poissonov proces*.

* Bolj formalno lahko definiramo Poissonov proces kot proces dogodkov tako, da so dogodki med seboj popolnoma neodvisni, dva dogodka se ne moreta zgoditi hkrati, verjetnost, da se bo dogodek zgodil znotraj časovnega intervala Δt pa je sorazmerna dolžini tega intervala.

Zmogljivost omrežja in zakasnitve v njem pa niso odvisne le od lastnosti kanalov in naprav ter obremenitve omrežja, ampak tudi od protokolnega sklada in komunikacijskih protokolov v njem. Poglavitni namen tega poglavja je študij vpliva protokolov in njihovih lastnosti na zmogljivosti komunikacijskih sistemov.

6.2 Komunikacijski kanali

Vzemimo, da imamo komunikacijski kanal, skozi katerega se prenese povprečno λ paketov v sekundi, povprečna dolžina teh paketov pa naj bo L (bitov). Naj bo nazivna hitrost kanala R (b/s), kar pomeni, da oddajnik, kadar ne miruje, oddaja informacijo s hitrostjo R, prav tako pa tudi sprejemnik sprejema informacijo s hitrostjo R, kadar pač sprejema. Če bi se skozi kanal prenašala informacija neprestano (brez presledkov med paketi), bi se prenašala z dejansko hitrostjo R; če pa se prenese v povprečju λ paketov v sekundi, to pomeni **povprečno (efektivno) hitrost**

$$r = \lambda \cdot L \ . \tag{5}$$

Če to povprečno hitrost normiramo z nazivno hitrostjo kanala, dobimo merilo za izkoriščenost kanala, ki ga imenujemo **prometni pretok** (angleško **traffic intensity**)

$$y = \frac{r}{R} = \frac{\lambda \cdot L}{R} \ . \tag{6}$$

V splošnem torej lahko prometni pretok definiramo kot razmerje dejanske proti nazivni hitrosti prenosa informacije skozi kanal. Ker pa je trajanje povprečno dolgega paketa (to je čas, potreben za oddajo oziroma sprejem paketa) enako

$$t_t = \frac{L}{R} \ , \tag{7}$$

iz definicije pogostnosti paketov λ pa sledi, da se paket pojavi v povprečju vsakih

$$T = \frac{1}{\lambda} \tag{8}$$

sekund, vidimo, da prometni pretok ni le merilo za izkoriščenost prenosne hitrosti kanala, ampak tudi za izkoriščenost časa v kanalu:

$$y = \frac{\lambda \cdot L}{R} = \frac{t_t}{T} \ . \tag{9}$$

Očitno je prometni pretok veličina brez enote, saj predstavlja razmerje dveh hitrosti ali pa razmerje dveh časov. Kljub temu pa v klasični teoriji prometa uporabljamo za prometni pretok navidezno enoto Erlang (erl); le-ta se tako imenuje po danskem inženirju Erlangu,

ki se je med prvimi (že pred 100 leti) ukvarjal s teorijo prometa. Prometni pretok kanala je lahko največ 1.

Kaj pa, če se nekateri izmed paketov, ki jih oddajnik odda v kanal, izgubijo in ne pridejo do sprejemnika? Do takih izgub dejansko prihaja, v fizičnih kanalih zaradi vpliva šuma, v navideznih kanalih pa zaradi preobremenjenosti omrežja ali njegovih naprav in posledično polnih čakalnih vrst. Vzemimo, da oddajnik v kanal z nazivno hitrostjo R odda λ_g paketov s povprečno dolžino L bitov v sekundi, sprejemnik pa sprejme λ_s v povprečju enako dolgih paketov v sekundi*. Oddajnik torej oddaja informacijo s povprečno hitrostjo

$$r_g = \lambda_g \cdot L \ , \tag{10}$$

sprejemnik pa v povprečju sprejema informacijo s hitrostjo

$$r_s = \lambda_s \cdot L \ . \tag{11}$$

Očitno sta potem različna tudi oba prometna pretoka. Prometni pretok oddajnika imenujemo **ponujani prometni pretok**

$$y_g = \frac{r_g}{R} = \frac{\lambda_g \cdot L}{R} \ , \tag{12}$$

prometni pretok sprejemnika pa **opravljeni prometni pretok**

$$y_s = \frac{r_s}{R} = \frac{\lambda_s \cdot L}{R} \ . \tag{13}$$

Razmerje opravljenega proti ponujanemu prometnemu pretoku je lahko kvečjemu enako ena in pomeni verjetnost, da bo oddani paket tudi sprejet

$$p_s = \frac{y_s}{y_g} = \frac{r_s}{r_g} = \frac{\lambda_s}{\lambda_g} \ , \tag{14}$$

verjetnost, da oddani paket ne bo sprejet, pa bomo imenovali **izgube**

$$Z = 1 - p_s = \frac{y_g - y_s}{y_g} \ . \tag{15}$$

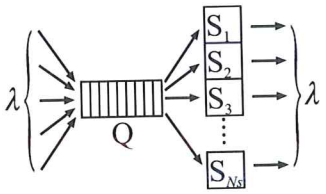
Med ponujanim in opravljenim prometnim pretokom torej obstaja zveza

$$y_s = y_g \cdot p_s = y_g \cdot (1 - Z) \ . \tag{16}$$

* Ta predpostavka ne drži vedno, saj je pogosto verjetnost izgube daljših paketov večja od verjetnosti izgube krajših paketov.

6.3 Strežniki

V prejšnjem razdelku smo raziskali prometne lastnosti komunikacijskega kanala kot sredstva za prenos informacije s kraja na kraj. V telekomunikacijskih omrežjih pa, kot smo omenili že na začetku, poleg kanalov uporabljamo tudi naprave, ki informacijo obdelujejo (procesirajo). V kontekstu teorije prometa in čakalnih vrst tovrstne naprave največkrat imenujemo kar **strežniki**. V uvodu smo prav tako že povedali, da vsaka naprava, ki informacijo obdeluje, potrebuje na svojem vhodu tudi čakalno vrsto. V tem razdelku se bomo torej posvetili obravnavi prometnih lastnosti strežnikov s čakalnimi vrstami. Po eni strani igra čakalna vrsta vlogo nekakšnega amortizerja med napravami, ki lahko delujejo z različnimi hitrostmi. Po drugi strani pa lahko deluje čakalna vrsta kot nekakšno zbirališče (čakalnica) za pakete, ki prihajajo v sistem (strežnik) z različnih vhodov, se tu uredijo, nato pa so lepo po vrsti obdelani. Ker pa paketi v čakalni vrsti čakajo, ima prisotnost čakalnih vrst v telekomunikacijskem sistemu bistven vpliv na zmogljivost tega sistema. Osnovni model naprave s čakalno vrsto brez izgub sestavljajo paketi, ki vstopajo v sistem s povprečno hitrostjo λ paketov v sekundi, čakalna vrsta, eden ali več strežnikov in paketi, ki izstopajo iz sistema prav tako s povprečno hitrostjo λ paketov v sekundi. Tak model kaže Slika 6-1.



Slika 6-1. Sistem s čakalno vrsto brez izgub

Ker se v sistemu s čakalno vrsto brez izgub paketi ne morejo izgubiti, v sistemu pa se tudi ne morejo kopičiti, je torej povprečna hitrost oddajanja paketov enaka povprečni pogostnosti sprejemanja paketov,

$\lambda = \lambda_s = \lambda_g$.

Paketi vstopajo v čakalno vrsto, od koder jih strežniki jemljejo v obdelavo. Poznamo več disciplin jemanja paketov iz čakalne vrste. Najpogostejša je disciplina obdelave po vrsti vstopanja (**FIFO - First In, First Out**). Če imajo paketi v čakalni vrsti različne prednosti (prioritete), pa bodo strežniki najprej obdelovali pakete z višjo prednostjo in šele nato pakete z nižjo prednostjo.

Vzemimo, da imamo N_s strežnikov, ki obdelujejo pakete iz ene same čakalne vrste, vsak izmed njih pa naj bo sposoben obdelovati pakete z nazivno hitrostjo μ paketov v sekundi. To hkrati tudi pomeni, da je povprečni čas obdelave enega paketa v strežniku enak

$$t_{proc} = \frac{1}{\mu} . \tag{17}$$

Seveda lahko dejanske (efektivne) in nazivne hitrosti pretoka (to je prihajanja in odhajanja) ter obdelave paketov pretvorimo tudi v hitrosti pretoka in obdelave informacije v b/s, če λ oziroma μ pomnožimo s povprečno dolžino paketa v bitih L:

$$r = \lambda \cdot L , \tag{18}$$

$$R = \mu \cdot L . \tag{19}$$

Hitrosti dejanskega in nazivnega pretoka oziroma obdelave informacije nismo po naključju označili z istima oznakama kot v prejšnjem razdelku dejansko in nazivno hitrost prenosa informacije skozi kanal. Tudi strežnik s čakalno vrsto si namreč lahko predstavljamo kot nekakšen kanal z določeno kapaciteto obdelave in z določeno dejansko obremenitvijo. Tudi v primeru strežnika lahko govorimo o prometnem pretoku kot razmerju med dejansko in nazivno hitrostjo pretoka paketov oziroma hitrostjo obdelave strežnika:

$$y = \frac{r}{R} = \frac{\lambda}{\mu} , \tag{20}$$

ki nam prav tako pove izkoriščenost naprave (v tem primeru strežnika). Seveda pa je v tem primeru teoretično največji možni prometni pretok enak številu strežnikov N_s.

V nečem pa se strežnik s čakalno vrsto vendarle bistveno razlikuje od fizičnega kanala. Medtem ko ima fizični kanal svojo od nič različno fizično dimenzijo in zato zakasni paket za čas, ki je premo sorazmeren tej dimenziji, štejemo napravo s čakalno vrsto za koncentriran element, v katerem se paketi zakasnijo zaradi obdelave v strežniku za povprečni čas

$$t_{proc} = \frac{1}{\mu} = \frac{L}{R} , \tag{21}$$

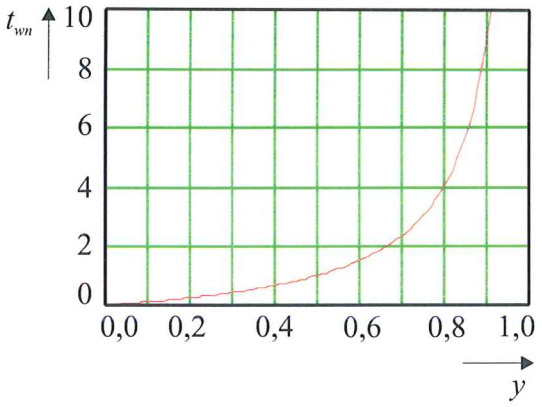
v čakalni vrsti pa še za povprečni čas t_w. Skupni povprečni čas zadrževanja paketa v napravi je torej

$$T = t_w + t_{proc} \ . \tag{22}$$

Čas čakanja v čakalni vrsti je prav lahko tudi precej večji od časa obdelave, zato je pomembno, če ga lahko za dane razmere vnaprej predvidimo. To pa pogosto ni prav lahko, saj je ta čas odvisen od hitrosti pretoka in obdelave paketov, procesa vstopanja paketov v napravo, od dolžine in porazdelitve dolžin paketov, števila strežnikov in discipline čakalne vrste. Preprosta (moč jo je tudi analitično izpeljati) je enačba za naprave s čakalno vrsto FIFO in enim samim strežnikom v primeru Poissonovega procesa pritoka paketov, katerih dolžine so eksponentno porazdeljene. Enačba 23 podaja normiran povprečni čas čakanja v čakalni vrsti takega sistema (čas čakanja smo normirali s časom obdelave v strežniku):

$$t_{wn} = \frac{t_w}{t_t} = \frac{y}{1-y} = \frac{\lambda}{\mu - \lambda} \ . \tag{24}$$

Odvisnost normiranega povprečnega čakalnega časa od prometnega pretoka po tej enačbi kaže tudi Slika 6-2.



Slika 6-2. Odvisnost normiranega čakalnega časa od prometnega pretoka

Tako iz enačbe 23 kot tudi iz slike Slika 6-2 lahko vidimo, da povprečni čakalni čas narašča proti neskončnosti, če narašča prometni pretok proti 1 (torej, če narašča hitrost pretoka paketov proti nazivni hitrosti obdelave paketov v strežniku). To je razumljivo, saj je maksimalni prometni pretok naprave vendar 1, proces prihajanja paketov pa je naključen, kar pomeni, da so med prihodi zaporednih paketov vedno krajši ali daljši presledki. Če bi imeli N_s strežnikov, bi bil maksimalni prometni pretok naprave N_s in čakalni čas bi naraščal proti neskončnosti pri $y \rightarrow N_s$. Pri pogoju $y = N_s$ pa bi lahko naprava

obdelovala pakete brez čakanja v čakalni vrsti le, če bi paketi prihajali deterministično (torej ne naključno) eden za drugim, med njimi pa ne bi bilo presledkov.

Logični razmislek nam pove, da bo v primeru, ko bodo povprečni čakalni časi naraščali, naraščalo tudi povprečno število paketov v čakalni vrsti - naraščala bo torej povprečna dolžina čakalne vrste. Res se da dokazati, da velja za sistem s čakalno vrsto, skozi katero se v povprečju prenaša λ paketov v časovni enoti, takale povezava med povprečnim številom paketov v sistemu N in povprečnim časom zadrževanja paketa v sistemu T:

$$N = \lambda \cdot T \ . \tag{25}$$

Enačbo imenujemo **Little-ov teorem**. Ta rezultat pa nam pove, da gre v primeru, ko narašča povprečni čas čakanja proti neskončnosti, tudi povprečno število paketov v čakalni vrsti proti neskončnosti. Čakalno vrsto implementiramo s spominom, ki pa ga imamo lahko vedno le končno mnogo na voljo. Dejanska dolžina čakalne vrste lahko torej zavzame le vrednosti, ki so manjše ali enake neki maksimalni dolžini. Če pa pride paket do čakalne vrste v trenutku, ko je ta polna (ima maksimalno dolžino), bo naprava paket zavrgla - tak paket bo izgubljen. V resnici torej naprave s čakalnimi vrstami niso idealne, v njih prihaja do izgub paketov. To se dogaja tudi tedaj, ko povprečna hitrost pritoka paketov ni zelo blizu maksimalni hitrosti obdelave. Trenutna hitrost pritoka paketov, z njo vred pa tudi dolžina čakalne vrste, namreč ves čas niha, enkrat bolj in drugič manj, okrog svoje povprečne vrednosti. Tudi pri zmernem pritoku paketov v čakalno vrsto (torej takem, ki je precej pod kapaciteto strežnika) se lahko občasno zgodi, da se bo čakalna vrsta napolnila in bo prišlo do izgub. Brez izgub bo lahko torej le sistem z neomejeno dolžino čakalne vrste, kakršnih pa v praksi ne premoremo. Pri realni napravi s čakalno vrsto torej povprečni hitrosti vstopanja paketov v napravo λ_g in izstopanja paketov iz nje λ_s ne bosta nujno enaki. Vedno pa bosta veljala pogoja

$$\lambda_s \leq \lambda_g \tag{26}$$

in

$$\lambda_s \leq N_s \cdot \mu \ . \tag{27}$$

Prav tako kot v primeru kanala z izgubami, bi lahko tudi v primeru čakalne vrste z izgubami definirali verjetnost uspešne obdelave paketa p_s oziroma verjetnost izgube paketa Z:

$$p_s = \frac{\lambda_s}{\lambda_g}, \tag{28}$$

$$Z = 1 - p_s = \frac{\lambda_g - \lambda_s}{\lambda_g}. \tag{29}$$

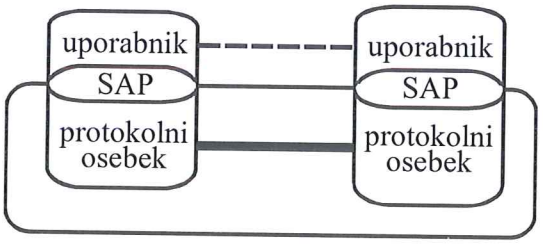
6.4 Telekomunikacijsko omrežje

Telekomunikacijsko omrežje, ki povezuje dva ali več uporabnikov med seboj, je pogosto kompleksen skupek takšnih in drugačnih kanalov ter strežnikov s čakalnimi vrstami. To pa seveda prav tako velja tudi za navidezen komunikacijski kanal. Analitična obravnava tako kompleksnega problema je v splošnem nemogoča, pogosto se pri določevanju lastnosti navideznega kanala, ki ga uporabnika vidita med seboj, zatekamo k meritvam ali k simulaciji.

6.5 Zmogljivost protokola, storitve, protokolnega sklada

Ob obravnavi protokolov se bomo seveda tudi vpraševali po njihovih lastnostih, povezanih s telekomunikacijskim prometom. Pri tem nas bodo predvsem zanimali informacijski pretok in izkoriščenost telekomunikacijskega sistema ter zakasnitve, ki jih v sistemu utrpi uporabniška informacija. Vse to je namreč odvisno od uporabljenega protokola, njegovih parametrov in od ostalih lastnosti sistema.

Oglejmo si najprej problem, ki ga kaže slika 6-3.



Slika 6-3. Implementacija navideznega kanala med uporabnikoma

Med dvema uporabnikoma naj bo navidezen kanal (označen na sliki s tanjšo črtkano črto). Ta navidezni kanal je realiziran z dvema protokolnima osebkoma (s po enim pri vsakem uporabniku) in s kanalom (debelejša polna črta), ki povezuje ta dva osebka in po katerem poteka komunikacija med partnerskima osebkoma po protokolu P. Nazivna

hitrost prenosa informacije skozi kanal ponudnika storitve (to je nazivna hitrost oddajanja informacije v kanal oziroma sprejemanja iz njega) naj bo R_p. Vzemimo, da pri izvajanju storitve prenosa podatkov oddajni protokolni osebek pošlje sprejemnemu λ_g informacijskih protokolnih sporočil v sekundi, povprečna dolžina teh sporočil pa je L_p bitov. Hitrost oddajanja informacije v kanal ponudnika je torej

$$r_g = \lambda_g \cdot L_p \quad \text{bit/s} \quad , \tag{30}$$

kar imenujemo **ponujani pretok informacije** skozi kanal. To je dejanska hitrost oddajanja informacije ponudnika storitve. Če pa to hitrost normiramo z nazivno hitrostjo prenosa informacije skozi kanal R_p, dobimo **ponujani prometni pretok** (angl. **offered load**) skozi kanal G,

$$G = \frac{r_g}{R_p} \quad . \tag{31}$$

Ker oddajnik ne more oddajati s hitrostjo, ki bi bila večja od R_p, je seveda ponujani prometni pretok v kanal lahko le manjši ali enak 1, $G \leq 1$. V resnici pa tudi enačaj ne drži, saj protokolni osebki poleg informacijskih protokolnih sporočil skoraj vedno oddajajo tudi nadzorna protokolna sporočila. Sedaj pa definirajmo še **opravljeni pretok informacije** r_s kot hitrost sprejemanja le tistih informacijskih protokolnih sporočil, ki so s stališča protokola uspešna, kar pomeni, da prispevajo k napredovanju komunikacijskega procesa. Sem ne štejejo protokolna sporočila, ki so se med prenosom skozi kanal kakorkoli pokvarila ali izgubila ali jih je sprejemni protokolni osebek iz kakršnega koli drugega razloga zavrgel, pa tudi ne nadzorna protokolna sporočila. Če sprejemni protokolni osebek uspešno sprejme λ_s protokolnih sporočil v sekundi, je hitrost sprejemanja informacije uspešnih protokolnih sporočil enaka

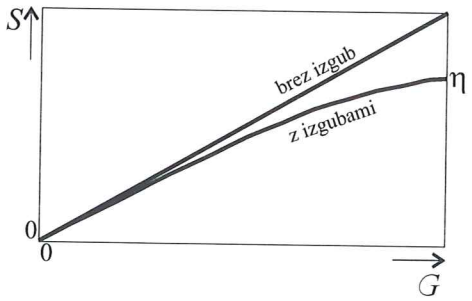
$$r_s = \lambda_s \cdot L_p \quad , \tag{32}$$

z normiranjem pa dobimo še **opravljeni prometni pretok** ali krajše **pretok** (ang. **throughput**)

$$S = \frac{r_s}{R_p} \quad . \tag{33}$$

Če je kanal brez izgub, kar pomeni, da se v njem nobeno sporočilo ne pokvari ali izgubi oziroma so vsa sprejeta sporočila koristna s stališča napredovanja komunikacijskega

sistema, potem sta ponujani in opravljeni prometni pretok enaka, $S=G$. Če pa izgube v kanalu so, velja $S<G$. Slika 6-4 kaže tipično odvisnost funkcije $S(G)$ v primeru brezizgubnega in izgubnega kanala (v tej sliki je predpostavljena odvisnost izgub od prometnega pretoka, zato je odvisnost opravljenega od ponujanega pretoka nelinearna).



Slika 6-4. Tipična odvisnost opravljenega od ponujanega prometnega pretoka

V sliki Slika 6-4 smo s črko η označili maksimalno vrednost pretoka S, torej pretok skozi kanal, ko je ta polno obremenjen, kolikor pač to dopušča protokol. Z drugimi besedami to pomeni, da uporabnik preko točke dostopa do storitve oddajni protokolni osebek ves čas zalaga z uporabniškimi sporočili oziroma da čakalna vrsta med uporabnikom in oddajnim protokolnim osebkom ni nikoli prazna. Maksimalno vrednost pretoka η bomo imenovali **izkoristek protokola** ali **učinkovitost protokola** (angleško **efficiency**). Izkoristek protokola je torej največji možni opravljeni prometni pretok oziroma največja možna hitrost uspešnega prenosa informacije v informacijskih protokolnih sporočilih skozi kanal, ki povezuje oba protokolna osebka, normirana z nazivno hitrostjo tega kanala

$$\eta = \frac{r_{s_{\max}}}{R} \ . \tag{34}$$

Ker pa smo v tem poglavju že ugotovili, da prometni protok ni le delež koristno uporabljene nazivne hitrosti, ampak tudi delež koristno uporabljenega časa, lahko izkoristek protokola zapišemo tudi v obliki

$$\eta = \frac{t_{i_{\max}}}{T} \ , \tag{35}$$

kjer je $t_{i_{\max}}$ maksimalni možni čas oddajanja/sprejemanja uspešnih informacijskih protokolnih sporočil v celotnem času T.

Izkoristek protokola ni odvisen le od protokola, ampak tudi od lastnosti kanala in celo od nekaterih značilnosti protokolnih sporočil.

Žal pa uporabnik storitve ne bo videl niti opravljenega pretoka informacije r_s, niti pretoka S, pa tudi ne učinkovitosti η. Informacijska protokolna sporočila namreč poleg uporabniških sporočil, ki edinole zanimajo uporabnika, vsebujejo tudi režijo. Zato je vse te veličine s stališča uporabnika treba množiti s **faktorjem režije**

$$k_r = \frac{L_u}{L_p} = \frac{L_u}{L_u + L_r} \quad , \tag{36}$$

kjer je L_u povprečna dolžina uporabniškega sporočila (*SDU*), vsebovanega v protokolnih sporočilih, L_r dolžina režije, $L_p = L_u + L_r$ pa dolžina protokolnega sporočila. Dolžine uporabniških sporočil so večinoma lahko različne in se naključno spreminjajo od sporočila do sporočila, zato v enačbi (35) vzamemo povprečno dolžino; dolžina režije v protokolnem sporočilu pa se pri večini protokolov ne spreminja. Tako je **nazivna hitrost prenosa informacije** skozi navidezni kanal med uporabnikoma enaka

$$R_u = \frac{L_u}{L_u + L_r} \cdot \eta \cdot R_p \quad . \tag{37}$$

Enačba (36) torej podaja največjo možna hitrost prenosa informacije med uporabnikoma storitve, ki je seveda manjša od hitrosti prenosa informacije skozi kanal med protokolnima osebkoma.

Iz enačbe (36) bi lahko sklepali, da bo prenosna hitrost, ki jo vidita uporabnika, tem večja, čim daljša bodo (pri dani dolžini režije) uporabniška sporočila. Vendar pa to ne drži vedno, saj se v določenih pogojih, ki jih bomo še srečali, lahko z večanjem dolžine uporabniških sporočil manjša učinkovitost protokola η. Tudi iz tega razloga nekateri protokoli dolžino uporabniških sporočil omejujejo. V razponu dovoljenih dolžin pa je seveda dejanska povprečna dolžina uporabniških sporočil odvisna od uporabnikov.

Enega izmed vzrokov za zmanjšanje izkoristka protokola predstavljajo tudi nadzorna protokolna sporočila. Ta so npr. potrebna za vzpostavljanje in sproščanje zvez; zato bi se nam morda utegnilo na prvi pogled zazdeti, da predstavlja vzpostavljanje zveze samo nepotrebno potrato kapacitete kanala. Vendar pa bo poraba kapacitete kanala za vzpostavljanje, vzdrževanje in sproščanje zveze porabila pomemben del kapacitete kanala le tedaj, ko je treba med dvema uporabnikoma prenesti le malo množino informacije (npr.

nekaj paketov); kadar pa je te informacije veliko, ima upravljanje zveze zanemarljiv vpliv na učinkovitost protokola.

Če imamo torej protokolni sklad z n sloji in hitrostjo prenosa informacije R skozi fizični kanal najnižjega sloja, bo maksimalna hitrost prenosa med uporabnikoma tega sistema enaka

$$R_u = k_{r_n} \cdot \eta_n \cdot k_{r_{n-1}} \cdot \eta_{n-1} \cdot \dots \cdot k_{r_2} \cdot \eta_2 \cdot k_{r_1} \cdot \eta_1 \cdot R = R \cdot \prod_{i=1}^{n} \left(k_{r_i} \cdot \eta_i \right) \ , \tag{38}$$

kar je lahko precej manj od hitrosti fizičnega kanala R. Če pa enačbo (37) delimo z nazivno hitrostjo fizičnega kanala, dobimo učinkovitost (izkoristek) celotnega protokolnega sklada oziroma telekomunikacijskega sistema

$$\eta = \frac{R_u}{R} = k_{r_n} \cdot \eta_n \cdot k_{r_{n-1}} \cdot \eta_{n-1} \cdot \dots \cdot k_{r_2} \cdot \eta_2 \cdot k_{r_1} \cdot \eta_1 = \prod_{i=1}^{n} \left(k_{r_i} \cdot \eta_i \right) \ . \tag{39}$$

Protokol vsakega izmed slojev torej zahteva zase določen del kapacitete kanala, v zameno pa nudi opravljanje določenega dela - reševanje svojega deleža komunikacijskega problema. Celotna režija sistema bi bila lahko seveda manjša, če bi imeli enovit, nerazslojen protokol, vendar pa bi bil tak protokol gotovo manj zanesljiv in kvaliteten, predvsem pa bi ga bilo precej težje načrtati, implementirati, testirati in vzdrževati.

6.6 Zakasnitve

Druga mera kvalitete delovanja komunikacijskega sistema so zakasnitve sporočil na poti od enega uporabnika do drugega.

Če opazujemo izmenjavo informacijskih protokolnih sporočil med protokolnima osebkoma izvajalca storitve, lahko **zakasnitev** T_d definiramo kot povprečno vrednost časa, ki preteče od trenutka, ko oddajni osebek začne sporočilo prvič oddajati (v primeru, ko se to sporočilo pokvari ali izgubi, ga je treba večkrat poslati), pa vse do trenutka, ko to sporočilo v celoti uspešno prispe do sprejemnega osebka in ta uporabniško sporočilo preda svojemu uporabniku (glej primer na sliki **Error! Not a valid link.**). V skladu z osnovnimi pojmi o protokolnem skladu bi lahko tudi rekli, da je to čas ki mine med primitivoma `Data.Req` in `Data.Ind`, ki se nanašata na prenos istega informacijskega protokolneg sporočila. Pogosto pa se uporablja tudi **normirano zakasnitev** T_{dn}, ki jo

dobimo tako, da zakasnitev T_d delimo s časom T_t, potrebnim za oddajanje povprečno dolgega protokolnega sporočila v kanal pri nazivni hitrosti tega kanala:

$$T_{dn} = \frac{T_d}{T_t} = \frac{T_d \cdot R_p}{(L_u + L_r)} \quad , \tag{40}$$

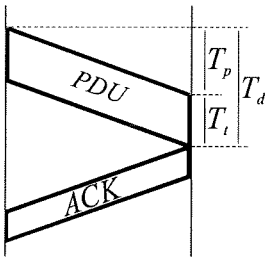
kjer so R_p nazivna hitrost kanala, L_u povprečna dolžina uporabniškega sporočila in L_r dolžina režije. Minimalna zakasnitev je vsota časa oddajanja sporočila T_t in časa propagacije elektromagnetnega signala skozi kanal T_p :

$$T_{d\min} = T_t + T_p = \frac{(L_u + L_r)}{R_p} + T_p \quad . \tag{41}$$

V primeru, da gre za fizični kanal, je čas propagacije T_p enak

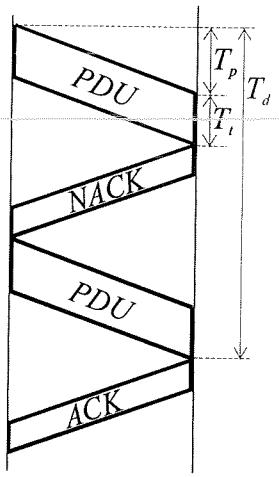
$$T_p = \frac{D}{v} \quad , \tag{42}$$

kjer je D fizična razdalja med subjektoma, ki komunicirata, v pa hitrost širjenja elektromagnetnega signala skozi prenosno pot. Minimalno zakasnitev dosežemo le v primeru, ko kanal nima izgub:



Slika 6-5. Minimalna zakasnitev pri prenosu sporočila

Pri tem seveda čas, ki je potreben za morebitno potrditev sprejema protokolne podatkovne enote, če protokol tako potrditev predpisuje, ne šteje v zakasnitev. Če pa se sporočilo pri prenosu skozi kanal pokvari in ga je treba ponovno pošiljati, bo zakasnitev seveda temu primerno večja, kot vidimo na sliki 6-6. Ta slika zaradi preprostosti in nazornosti prikazuje le prenos enega sporočila. V resnici se pokvarijo le nekatera izmed mnogih sporočil, zakasnitev pa predstavlja povprečno vrednost.



Slika 6-6. Zakasnitev sporočila pri okvari sporočila v kanalu

Čeprav smo v zakasnitvi informacijskega protokolnega sporočila že upoštevali tudi režijo, pa bo uporabnik storitve vseeno videl nekoliko daljšo zakasnitev T_u. En vzrok za to tiči v času, ki je potreben za obdelavo v obeh protokolnih osebkih in ga bomo imenovali **čas procesiranja** T_{proc}, poleg tega pa sama zahteva uporabnika po storitvi prenosa uporabniškega sporočila (`data.req`) lahko tudi čaka čas T_w v čakalni vrsti v točki dostopa do storitve. Torej,

$$T_u = T_w + T_{proc} + T_p \quad . \tag{43}$$

Vse veličine v tej enačbi so seveda povprečne vrednosti.

Vidimo torej, da bo tudi pri zakasnitvi signala vsak sloj v protokolnem skladu prispeval svoj delež, nekaj zaradi obdelave v protokolnih osebkih, nekaj zaradi čakanja v čakalnih vrstah, seveda pa ima tu pomemben delež tudi izvajanje raznih nalog protokolov, kot npr. popravljanje napak s ponovnim pošiljanjem itd. Višje kot je nek sloj v protokolnem skladu, večja bo zakasnitev. Vendar pa lahko spet ugotavljamo, da je povečanje zakasnitve le davek, ki ga plačamo za večjo kvaliteto prenosa.

Definicije prometnih lastnosti protokolov, kot smo jih navedli v tem razdelku, niso splošne in univerzalne. Pri obravnavi različnih protokolov, ki delujejo v različnih omrežjih oziroma v različnih slojih protokolnega sklada, ali pa imajo različne cilje, lahko uporabimo različne definicije, pa tudi pri obravnavi istega protokola se definicije različnih avtorjev lahko nekoliko razlikujejo. Pa to niti ni tako pomembno. Važno je, da pri primerjavi različnih protokolov ali vsaj različnih variant istega protokola, ki služijo

istemu namenu, uporabimo enake definicije prometnih lastnosti, sicer primerjava ne bi bila korektna. Vsekakor pa so vedno poglavitne veličine, s katerimi ocenjujemo kvaliteto protokola, pretok, učinkovitost in zakasnitev. Seveda te veličine med seboj niso neodvisne. Čim večja sta pretok (pri danem ponujanem pretoku) in učinkovitost in čim manjša je zakasnitev, tem kvalitetnejši je protokol.

6.7 Izbira parametrov protokola

Učinkovitost protokola in zakasnitve so odvisni od marsičesa. Lastnosti kanala so tu seveda na prvem mestu, vendar na te lastnosti protokol nima vpliva. Pomemben je predvsem protokol sam. Protokol je nek porazdeljen algoritem, ki se izvaja v obeh (oziroma vseh, če jih je več) protokolnih osebkih hkrati. Seveda je izbira samega algoritma za dano nalogo ključnega pomena. To pa še ni vse. Obnašanje protokola poleg samega algoritma določajo tudi parametri, katerih vrednosti imajo lahko na lastnosti protokola velik vpliv. Primera takih parametrov sta dolžina uporabniškega sporočila in nastavitve časovnikov. Optimalne vrednosti parametrov nekega protokola so lahko močno odvisne od lastnosti kanala in od ostalih razmer v telekomunikacijskem sistemu (omrežju). Lahko pa nastopajo optimumi različnih mer kvalitete sistema pri različnih kombinacijah parametrov, zato se je pogosto treba odločiti za nek kompromis. Tako sta si pogosto težnji po čim manjši zakasnitvi sporočil in čim večjem izkoristku kanala pogosto v medsebojnem nasprotju. Implementacije protokolov so večinoma tako fleksibilne, da je mogoče vrednosti parametrov spreminjati od instalacije do instalacije, ali pa celo med obratovanjem. To nalogo opravlja **upravljalec** komunikacijskega sistema. Upravljalec je lahko človek (**administrator**), lahko pa je to poseben sistem programske opreme, ki ugotavlja stanje v omrežju in glede na to določa parametre sistema oziroma protokolov. Ker pa je omrežje porazdeljen sistem, mora biti **sistem upravljanja omrežja** tudi komunikacijski sistem s svojimi posebnimi **upravljalskimi protokoli** (angleško **management protocols**). Ti protokoli pogosto delujejo v posebni komunikacijski ravnini, ki jo imenujemo **upravljalska ravnina** (ang. **management plane**).

6.8 Analiza in simulacija protokolov

Seveda je ugodno, če imamo možnost primerjati lastnosti dveh ali več protokolov v nekem komunikacijskem okolju, še preden se lotimo gradnje in instalacije takega sistema. Prav tako je pogosto bolje, če lahko preučujemo vpliv parametrov protokola na

njegove lastnosti na nekem modelu, ne pa na že delujočem sistemu. Obstajata dve vrsti modelov protokolov : **analitični model** in **simulacijski model**.

Analitični model predstavlja enačba ali sistem enačb, ki podajajo odvisnost lastnosti protokola od parametrov telekomunikacijskega sistema in samega protokola. Poglavitni matematični orodji, ki nam pomagata pri izpeljavi takih enačb, sta **teorija verjetnosti** in **teorija čakalnih vrst**. Kljub temu, da sta to uveljavljeni in dobro razviti matematični disciplini, pa je analitična obravnava ne ravno najpreprostejših protokolov zelo zahtevna. Zato performančne enačbe pogosto izpeljujejo pri določenih predpostavkah, ki so v realnih sistemih le delno, ali pa sploh niso izpolnjene. Primeri takih predpostavk so določena porazdelitev vhodnega telekomunikacijskega prometa, poenostavljene lastnosti kanala ipd. Pri uporabi takih enačb se moramo torej vedno vprašati, ali oziroma pod kakšnimi pogoji so te enačbe v kontekstu sistema, ki nas zanima, sploh veljavne.

Simulacijski model je računalniški program, ki posnema delovanje telekomunikacijskega sistema oziroma protokola. Poganjanje takega programa imenujemo simulacija. Pri načrtovanju simulacijskega modela imamo vsaj tri možnosti. Prva je ta, da uporabimo kupljen simulacijski program, ki ima že vgrajene modele raznih protokolov, mi pa določimo le še parametre teh modelov. Druga možnost je pisanje simulacijskega programa v programskem jeziku, ki je namenjen prav pisanju simulacijskih programov. Tak jezik ima že vgrajene operatorje in podatkovne strukture, ki so primerni za gradnjo simulacijskih modelov. Tretja možnost pa je pisanje simulacijskega programa v splošnem visokem programskem jeziku, kakršni so npr. Pascal, C, C++, Java ipd. Za te tri možnosti lahko ugotovimo, da vsaka izmed njih zahteva več inženirskega dela pri razvoju simulacijskega programa kot prejšnja in torej tudi dražji razvoj, zato pa je programska oprema, ki je za tak razvoj potrebna, precej cenejša in bolj splošno uporabna, izdelan program pa je lahko bolj pisan na kožo danemu problemu in zato tudi hitrejši. Za delovanje protokolov oziroma telekomunikacijskega sistema je značilno zaporedje dogodkov, ki se dogajajo le ob določenih diskretnih trenutkih, ki pa nastopajo asinhrono (torej ne v enakih razmakih). Za simulacijo takih sistemov je primerna simulacijska metoda, ki se imenuje **simulacija diskretnih dogodkov** (ang. **discrete event simulation**).

7 OPIS IN SPECIFIKACIJA PROTOKOLOV

V dosedanji obravnavi smo že uvideli potrebo po podrobnem opisu oziroma specifikaciji protokolov v standardih ali priporočilih. **Specifikacija protokola** mora biti natančna, razumljiva, popolna in nedvoumna, saj taka specifikacija služi kot podlaga za **implementacijo protokola**. Implementacijo protokola predstavlja bodisi elektronsko (integrirano) vezje ali računalniški program, lahko pa tudi kombinacija obojega. Če specifikacija protokola ne izpolnjuje gornjih zahtev, se lahko zgodi, da različne implementacije istega protokola niso med seboj kompatibilne in torej protokolni osebki, ki so implementirani na nekompatibilne načine, ne morejo uspešno komunicirati med seboj. S tem pa seveda standardizacija izgubi svoj pomen.

Naravni jezik ni ravno najbolj primeren za natančno in nedvoumno specifikacijo, saj je za take namene veliko prebogat; vsebuje mnogo sinonimov, veliko besed pa ima lahko tudi dva ali več povsem različnih pomenov, ki so v vsakdanjem govoru ali pisavi razvidni iz konteksta. Medtem ko lahko te lastnosti literarni tekst le bogatijo, strogo tehniškemu tekstu škodujejo. Zato se je za opis protokolov v standardih razvil suhoparni, pol tehniški in pol pravniški jezik, ki ga je težko brati in še težje razumeti*. So pa prav za opis protokolov razvili tudi več **formalnih jezikov**, ki so precej podobni programskim jezikom, omogočajo pa enoumno specifikacijo protokola. Tak opis je dovolj preprost za branje (če seveda bralec ta jezik pozna in obvlada), hkrati pa lahko opis protokola v formalnem jeziku tudi nudi osnovo za avtomatizirano preverjanje pravilnosti ali implementacijo protokola. V ta namen potrebujemo prevajalnik, ki prevede specifikacijo protokola, napisano (ali tudi narisano, če gre za grafični jezik) v formalnem jeziku, v implementacijo, napisano, če gre za programsko izvedbo, v enem izmed programskih jezikov (npr. C, C++, zbirnik ali celo strojni jezik).

V enem od prejšnjih poglavij smo spoznali, da specifikacijo protokola sestavljajo specifikacije tipov protokolnih sporočil (njihove abstraktne sintakse), njihovi formati ter pravila, ki določajo, kdaj sme oziroma mora protokolna entiteta poslati kakšno protokolno sporočilo. Formalni jezik za opis protokola mora torej nuditi mehanizme za opis vseh teh treh elementov specifikacije protokola. Delovanje protokolnih osebkov pa je tudi časovno pogojeno, kar se kaže v njihovem odzivanju na dogodke in v uporabi

časovnikov. Zato mora formalni jezik za specifikacijo protokolov nuditi ustrezne časovne mehanizme.

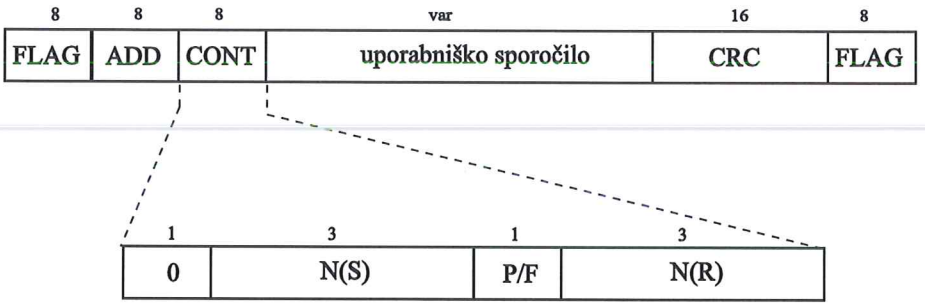
7.1 Specifikacija protokolnih sporočil

V poglavju **Error! Not a valid link.**4 smo že pokazali, kako lahko na preprost in razumljiv način podamo abstraktno sintakso protokolnih sporočil preprostih protokolov, pri čemer navedemo tip protokolnega sporočila in parametre, ki jih le-to nosi s seboj. Videli smo tudi, da ima protokolna podatkovna enota (protokolno sporočilo) neko svojo notranjo strukturo. Pa ne le, da je sestavljena iz glave, uporabniškega sporočila in repa, tudi glava in rep imata lahko neko podrobnejšo notranjo strukturo, ki jo moramo poznati, in vsak element v tej strukturi ima svoj pomen, ki ga moramo razumeti. Element v strukturi protokolnega sporočila po navadi imenujemo **polje** (ang. **field**). Polje samo pa je tudi lahko sestavljeno iz več (pod)polj. Samo zaporedje bitov v protokolni podatkovni enoti nam ne pove ničesar, če teh bitov ne znamo pravilno grupirati in če ne razumemo pomena posameznih binarnih zaporedij. Bistveni del specifikacije protokola je torej tudi specifikacija notranje strukture protokolne podatkovne enote in interpretacija vsebin njenih sestavnih delov - polj.

Ena možnost za specifikacijo formata protokolne podatkovne enote je grafična predstavitev njene strukture z dolžinami posameznih polj in podrobnejšim opisom implementacije njihovih vsebin. V nižjih slojih protokolnega sklada običajno uporabljamo v osnovi linearni opis strukture, ki predstavlja (od leve na desno) zaporedje polj z njihovimi dolžinami in vsebinami; lahko pa je tak opis tudi do neke mere hierarhično organiziran na ta način, da se polja delijo v podpolja itd. Preproste primere linearnega grafičnega opisa formatov protokolnih sporočil smo podali že pri opisu protokola TFTP. Kot primer hierarhično organiziranega opisa pa v sliki 1 3 podajamo strukturo informacijske protokolne podatkovne enote po protokolu HDLC. Številke nad posameznimi polji oziroma podpolji v tej sliki pomenijo njihove dolžine v bitih. Seveda pa k tej sliki spada še podrobnejši opis formata in interpretacije vsebine posameznih polj.

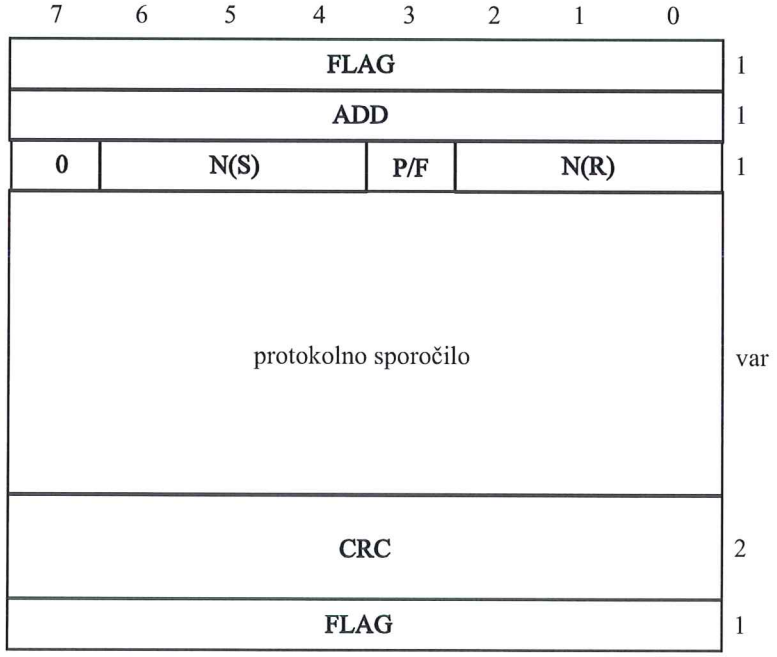
---

* Profesor Tanenbaum iz Amsterdama je tak jezik duhovito poimenoval "beaurocratspeaklanguage".



Slika 7-1. Format informacijskega protokolnega sporočila po protokolu HDLC

Zlasti v višjih slojih protokolnih skladov, kjer je vsebina najobičajneje organizirana po oktetih (zlogih), vsebino protokolnih sporočil pogosto prikazujemo v dvodimenzionalni obliki tako, da nam vertikalna dimenzija (od zgoraj navzdol) podaja oktete ali še celo daljše podatkovne enote v sporočilu, horizontalna dimenzija pa (od leve na desno) bite ali oktete v vrstici. Tak način prikaza nam še zlasti pride prav, kadar bi nam linearni način prikazovanja dal predolgo sliko. Enak PDU kot v sliki 1 3 bi lahko prikazali tudi dvodimenzionalno, kot to vidimo v sliki 2 4, v kateri so zgoraj označeni biti v oktetu, na desni strani pa dolžine polj (ali skupine polj) v oktetih. Seveda je tak način možen le, če je dolžina celotnega protokolnega sporočila enaka naravnemu številu oktetov.



Slika 7-2. Format informacijskega protokolnega sporočila po protokolu HDLC

Take preproste specifikacije protokolnih sporočil so uporabne le, kadar so ta sporočila razmeroma preprosta. V primeru kompleksnih sporočil, zlasti še v najvišjih slojih protokolnega sklada, pa pogosto raje uporabimo njihovo formalno specifikacijo.

7.2 Abstract Syntax Notation 1 (ASN.1)

Opis, ki ga podamo napol z risbo in napol s prostimi besedami, očitno ni preveč formalen, čeprav je dokaj preprost in zato tudi lahko razumljiv. V višjih slojih protokolnega sklada pa so opisi protokolnih podatkovnih enot vse bolj kompleksni, saj imajo lahko podatki, s katerimi tam operirajo protokolne entitete, zelo kompleksno strukturo in abstrakten pomen. Implementacije protokolov v najvišjih slojih so programi, ki so napisani v enem od visokih programskih jezikov (npr. Pascal, C, C++, Java). Programerju, ki piše v takem jeziku, se zdi popolnoma normalna in naravna uporaba bogatih možnosti, ki jih ti jeziki nudijo za strukturiranje podatkov; seveda pa je tako strukturiranje podatkov tudi popolnoma v skladu s sodobnim načinom strukturiranega in objektno orientiranega programiranja. Zato je vsekakor zelo primerno, če lahko pri specifikaciji protokolov v višjih slojih protokolnega sklada za opis protokolnih podatkovnih enot uporabimo podatkovne tipe, ki so podobni onim iz programskih jezikov. To nam omogočajo posebni jeziki za opis protokolnih podatkovnih enot.

Eden od najbolj znanih takih jezikov se imenuje Abstract Syntax Notation 1 ali krajše ASN.1, ki ga je standardizirala ISO. Jezik omogoča deklaracije podatkovnih tipov na podoben način, kot se to dela v večini visokih programskih jezikov, le da je implementacija bolj fleksibilna. Tako lahko npr. za tip INTEGER (celoštevilska vrednost) določimo dolžino polja. Po drugi strani pa je implementacija teh podatkovnih tipov natančno določena ne glede na vrsto računalnika, na katerem je protokolna entiteta implementirana.

Med osnovnimi (to je nestrukturiranimi) tipi so poleg tipa INTEGER še tipi BOOLEAN (logični vrednosti), BITSTRING (zaporedje binarnih vrednosti), OCTETSTRING (zaporedje oktetov), REAL (realna števila), ENUMERATED (naštevalni tip), IA5String (zaporedje znakov IA5 - ti so skorajda ekvivalentni znakom ASCII), GraphString (zaporedje grafičnih znakov), NULL (podatek brez tipa), ANY (tip podatka je definiran drugje). Za strukturiranje podatkov pa služijo SEQUENCE (zaporedje podpolj različnih tipov), SEQUENCEOF (zaporedje podpolj istega tipa), SET

(množica podpolj različnih tipov), SETOF (množica podpolj istega tipa), CHOICE (izbira med različnimi polji).

Opis protokolnih podatkovnih enot v jeziku ASN.1 torej podaja sintakso teh enot na abstrakten način. Tej sintaksi torej rečemo abstraktna sintaksa. Specifikacijo PDU-ja v jeziku ASN.1 pa je mogoče prevesti v standardno implementacijo, ki jo predstavlja zaporedje bitov. Definiciji teh zaporedij rečemo konkretna sintaksa oziroma prenosna sintaksa, saj definira čisto konkretna zaporedja binarnih vrednosti, ki jih prenašamo skozi kanal. Jezik ASN.1 torej definira pretvorbo med običajnimi podatkovnimi tipi visokih programskih jezikov in med implementacijo teh tipov v obliki protokolne podatkovne enote.

Tako pretvorbo je mogoče napraviti tudi avtomatsko s posebnim programom, ki je sestavni del implementacije oddajnega protokolnega osebka, medtem ko na sprejemni strani poteka pretvorba v obratni smeri.

7.3 Neformalni opis telekomunikacijskih sistemov

Sodobni telekomunikacijski sistemi so modularno zgrajeni, kar velja tudi za telekomunikacijske protokole. Za modularno zgrajen sistem je značilno, da ima neko strukturo, ki jo predstavljajo moduli in njihove medsebojne povezave, za posamezne module v tej strukturi pa je značilna funkcionalnost (to je odvisnost njihovih izhodnih signalov od vhodnih signalov in stanja).

Za neformalni opis strukture sistemov so se uveljavili tako imenovani blok diagrami, ki smo jih v tem delu že uporabili. Za opis komunikacije med protokolnimi osebki (torej njihove funkcionalnosti) lahko uporabljamo časovne sheme, ki smo jih tudi že prikazali v tem delu. Čeprav nam lahko časovne sheme nazorno pokažejo potek dogajanja v sistemu (še zlasti, če je sistem preprost in gre za komunikacijo le med dvema protokolnima osebkoma in za njuno interakcijo z osebkoma v višjem sloju), lahko v tak diagram narišemo le en ali nekaj bolj ali manj značilnih scenarijev komunikacijskega procesa, nikakor pa ne moremo zaobjeti vseh možnih zaporedij dogodkov. To pa pomeni, da tak opis vsekakor ni popoln! Tak opis tudi ne more biti neposredna osnova za implementacijo protokolnega osebka, saj implementiramo vsak osebek posebej, ne pa sistema kot celote. Torej je najpomembnejša naloga formalnega jezika za opis protokolov

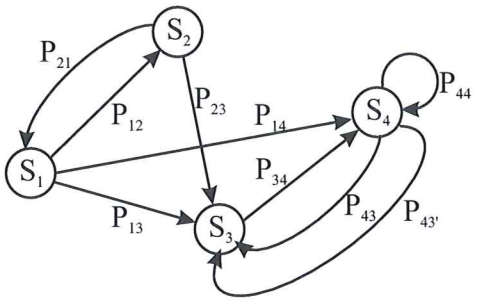
prav specifikacija funkcionalnosti elementa v sistemu (protokolnega osebka). Prav to pa je tudi najtežje specificirati.

7.4 Modeliranje funkcionalnosti s končnimi avtomati

Za opis funkcionalnosti (obnašanja) sistema ali njegovega elementa, ki deluje v diskretnem času (kar pomeni, da oddaja signale in tudi spreminja stanje le ob diskretnih časovnih trenutkih), so v preteklosti razvili več metod. Eden od najstarejših in v principu najpreprostejših modelov takega sistema je **končni avtomat** (ang. **Finite State Machine - FSM**). Ta model vsebuje poleg vhodnih in izhodnih signalov še svoje stanje. Možnih stanj je končno mnogo, odtod tudi izhaja ime avtomata. Avtomat deluje le tedaj, ko je za to izzvan od zunaj. Ob vsaki taki priliki lahko stori dvoje: odda izhodni signal in spremeni svoje stanje. Kaj to pomeni za nek konkreten avtomat, je odvisno od njegovega starega stanja in sprejetega vhodnega signala. To pa je v bistvu model, ki smo ga že uporabili za prikaz delovanja protokolnega osebka. Dogajanje v avtomatu torej predstavljajo prehodi iz stanja v stanje, ki jih vzbujajo zunanji dražljaji (vhodni signali), pri čemer lahko avtomat vsakič tudi odda izhodni signal. V posebnem primeru je lahko novo stanje tudi enako staremu. Delovanje takega avtomata lahko prikažemo s tabelo prehodov med stanji, v kateri za katerikoli vhodni dogodek, ki se je zgodil v katerem koli stanju avtomata, specificiramo ustrezno akcijo avtomata (na primer sprožen izhodni dogodek), pa tudi, v katero stanje bo avtomat pri tem prešel. Primer take tabele vidimo v tabeli .

staro stanje	*vhodni dogodek*	*akcija*	*novo stanje*
S1	P21	?	
S2	P12	?	
S3	P13	?	
S3	P23	?	
S3	P43	?	
S3	P43'	?	
S4	P14	?	
S4	P34	?	
S4	P44	?	

Lahko pa to delovanje ponazorimo tudi z usmerjenim grafom, kakršnega vidimo na naslednji sliki. Stanja predstavljajo krogi, ki imajo svoja imena, da lahko ločimo med njimi. Te kroge povezujejo puščice, ki predstavljajo prehode med stanji; te puščice so opremljene s pogoji, ki morajo biti izpolnjeni, da pride do prehodov. Pogoji pa so seveda dogodki na vhodnih signalih. Seveda ni nujno, da bi bili možni prehodi med poljubnima dvema stanjema. Prehode med istima stanjema, a v različni smeri, označimo posebej. Lahko pa imamo med istima stanjema v isti smeri tudi po več prehodov.



Pri opisu kompleksnega sistema lahko seveda število vseh možnih stanj zelo naraste, kar je tudi poglavitna šibka stran modeliranja s končnimi avtomati. Pri tem si lahko pomagamo tako, da stanje avtomata zapišemo z vrednostmi primerno izbranih spremenljivk, ki so definirane na dovolj abstrakten način. Tako lahko npr. naravno število n, $0 \leq n \leq 255$, zapišemo kot naravno število, ne pa kot zaporedje 8 binarnih vrednosti, kot bi to vrednost verjetno realizirali na površini silicija. Seveda se lahko med prehodom enega stanja v drugega spremeni vrednost večih spremenljivk. Ločili bomo le tista stanja, ki so med seboj ločena z zunanjimi dražljaji. Vse to nam število vseh možnih stanj precej zmanjša, vpelje pa potrebo po uporabi spremenljivk različnih tipov (kot jih poznamo tudi v visokih programskih jezikih). Z dodatkom te in še drugih možnosti pri modeliranju dobimo razširjeni končni avtomat.

7.5 Specifikacija in opis telekomunikacijskih sistemov in protokolov, SDL

Za potrebe formalne specifikacije telekomunikacijskih sistemov na sploh in še posebej telekomunikacijskih protokolov je ITU-T razvila in sprejela priporočilo za jezik, ki omogoča formalen opis strukture sistemov in njihovih sestavnih delov, prav tako pa tudi specifikacijo njihove funkcionalnosti. Jezik se imenuje Specification and Description Language ali krajše SDL. Definiran je v priporočilu ITU-T Z.100. Skladnja (sintaksa) ima dve obliki, ki pa sta si med seboj ekvivalentni: grafično (SDL/GR) in tekstovno (SDL/PR). V prvem primeru opisujemo sistem in njegovo funkcionalnost z grafičnimi

simboli, v drugem primeru pa s tekstom, ki je podoben klasičnim računalniškim programom. Zaradi večje nazornosti bomo v tem delu uporabljali grafično inačico. Opis sistema v jeziku SDL je lahko popolnoma formalen, kar pomeni, da ga je mogoče strojno interpretirati in obdelati, lahko pa so med čisto formalne konstrukte pomešana neformalna besedila, ki pojasnjujejo določeno dogajanje v obliki komentarja. Izbira med popolnoma formalnim in polformalnim opisom je odvisna od namena in uporabe tega opisa. Če zapisa ne nameravamo strojno interprertirati (z nekim računalniškim programom), lahko strogo formalni zapis uporabljamo le na tistih mestih, ki to zahtevajo zaradi nedvoumnosti specifikacije, drugod pa zaradi lažje berljivosti uporabimo neformalen jezik. Če pa je opis namenjen strojnemu branju in interpretaciji, mora biti seveda popolnoma formalen. Obstajajo namreč programski paketi, ki na osnovi opisa sistema v SDL omogočajo implementacijo in/ali simulacijo tega sistema.

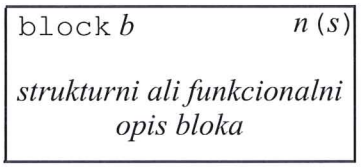
Kot je običaj v večini programskih jezikov, tudi v SDL ločimo med tipom objekta, ki določa lastnosti množice objektov tega tipa, in dejanskim objektom, ki ima lastnosti, določene s tipom tega objekta. Medtem ko moramo tip objekta definirati (to je določiti njegove lastnosti), sam objekt uporabimo v definiciji nekega drugega tipa objekta. Deklaracije nekaterih objektov ali njihovih tipov so lahko le tekstovne, tudi če sicer uporabljamo grafično obliko jezika SDL. Take deklaracije podajamo v posebnih okvirjih za tekstovne deklaracije. Tak okvir vidimo na naslednji sliki.

7.5.1 Struktura sistema

Osnovna strukturna gradnika sistema, ki ga želimo opisati v SDL, sta blok in kanal. Jezik SDL omogoča opis sistema na hierarhičen način. Blok na najvišjem nivoju te hierarhije je sistem. Strukturo sistema opišemo z bloki, ki jih med seboj povezujejo kanali. Le-ti so lahko enosmerni ali dvosmerni, pač glede na to, ali se lahko po kanalu prenaša informacija le v eno smer, ali pa v obe smeri. Za vsak kanal moramo za vsako smer posebej definirati, kateri signali se lahko po njem prenašajo. Blok lahko spet opišemo z njegovo notranjo strukturo, torej tako, da povemo, iz katerih podblokov je sestavljen in kateri kanali te podbloke povezujejo med seboj. Takšen hierarhičen način

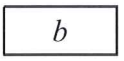
opisovanja lahko nadaljujemo toliko časa, dokler se nam to zdi potrebno. Hierarhičen način opisovanja strukture sistema je v navadi pri obravnavi vseh kompleksnejših sistemov in torej predstavlja metodologijo, ki omogoča obvladovanje kompleksnosti. Če nekega bloka ne opišemo z njegovo notranjo strukturo, moramo definirati njegovo funkcionalnost, o čemer bomo govorili kasneje.

Blok definiramo v posebnem okvirju, ki vsebuje tekstovne deklaracije in definicijo strukture ali funkcionalnosti bloka. Kombinacija strukturnega in funkcionalnega opisa ni dovoljena. Če definiramo blok v grafični obliki na papirju, ena sama stran največkrat ne zadostuje. Potrebujemo torej več okvirjev, vsakega pa moramo označiti z zaporedno številko okvirja. Na naslednji sliki vidimo *n*-tega izmed *s* okvirjev, ki definirajo blok tipa *b*.

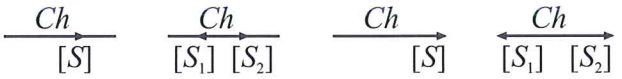


V primeru, da gre za blok na najvišjem nivoju hierarhije, torej za sistem, bomo namesto besede `block` zapisali `system`. Tudi sistem ima svoje ime.

Notranjo strukturo bloka torej podamo kot množico podblokov, ki jih povezujejo kanali. Za blok tipa *b* kot gradnik uporabljamo simbol



Za kanale uporabljamo naslednje simbole:



Pri tem je *Ch* ime kanala, *S* pa seznam imen signalov, ki jih lahko kanal prenaša, ločenih z vejicami. Če ima simbol puščici v obe smeri, gre za dupleksni kanal, seznam signalov pa je definiran za vsako smer posebej; sicer se signali prenašajo le v smeri puščice. Če je puščica oziroma sta puščici označeni na koncu kanala, gre za kanal brez zakasnitve, sicer pa ima signal nedeterministično zakasnitev (torej tako z naključno vrednostjo). Kanal običajno povezuje dva bloka med seboj. Če pa se en konec kanala dotika okvirja bloka, katerega strukturo definiramo, to predstavlja vrata tega bloka (povezavo med notranjostjo in zunanjostjo bloka).

7.5.2 Signali in podatkovni tipi

Bistvo funkcionalnosti komunikacijskega sistema predstavljata obdelava in prenos informacije. Informacija je po svoji naravi lahko statična ali dinamična. Osnovo za opis statične informacije predstavlja vrednost, osnovo za opis dinamične informacije pa dogodek. Množico vrednosti, ki se lahko pojavijo v nekem kontekstu, imenujemo podatkovni tip.

Spoznali smo že, da so nosilci informacije med bloki signali. Signal predstavlja sporočilo, ki ga en blok odda, drugi pa sprejme. Sprejem ali oddaja signala predstavlja dogodek v sistemu. Poleg tega, da prenos takega sporočila že sam po sebi pomeni prenos neke informacije (vrste dogodka), lahko signal s sabo nosi tudi eno ali več vrednosti. Ob deklaraciji signala moramo za vsako izmed teh vrednosti specificirati množico vrednosti, iz katere lahko to vrednost vzamemo, torej podatkovni tip. Signal bomo torej takole deklarirali:

`signal` *ime_signala* ;

`signal` *ime_signala* (*podatkovni_tip*) ;

`signal` *ime_signala* (*podatkovni_tip1*, *podatkovni_tip2...*) ;

Pri tem predstavlja *ime_signala* ime tipa signala, ki ga deklariramo, *podatkovni_tip* pa specifikacijo podatkovnega tipa vrednosti. V prvem primeru imamo signal, ki s seboj ne nosi nobene vrednosti, v drugem primeru nosi signal s seboj eno vrednost, v tretjem primeru pa dve ali več (odvisno od števila naštetih podatkovnih tipov) vrednosti.
Deklariramo pa lahko tudi seznam signalov, in sicer s specifikacijo

`signallist` *ime_seznama_signalov s1*, *s2*, *s3*, ... ;

kjer so *s1*, *s2*, *s3*... signali, specificirani v specifikacijah `signal`.

Sedaj lahko še nekoliko podrobneje pojasnimo, kako specificiramo signale, ki se lahko prenašajo po nekem konkretnem kanalu; v prejšnjem razdelku smo to označili z `[`*S*`]`. V resnici pa je ta *S* lahko signal ali seznam signalov. Dane so torej naslednje možnosti:

[*ime_signala*]

[*ime_signala*, *ime_signala*...]

[(*ime_seznama_signalov*)]

Jezik SDL definira precej podatkovnih tipov, poleg tega pa daje uporabniku možnost, da sam definira nove podatkovne tipe. Tu bomo navedli le nekatere izmed osnovnih tipov in možnosti strukturiranja tipov:

- `Boolean` (množica dveh logičnih vrednost - `True` in `False`)
- `Character` (množica ASCII znakov)
- `Charstring` (množica zaporedij ASCII znakov, torej množica besedil)
- `Integer` (množica celih števil)
- `Real` (množica realnih števil)
- `Duration` (množica časovnih razlik)
- `Time` (množica časov)
- `Array` (množica zaporedij elementov enega tipa, indeksiranih z vrednostmi drugega tipa)
- `Struct` (strukturiran podatkovni tip, katerega vrednosti so sestavljene iz več komponent različnih tipov - v matematiki takšno množico imenujemo Kartezijev produkt, v programskem jeziku Pascal je to record, v programskem jeziku C pa prav tako struct)

Podatkovna tipa `Duration` in `Time` se uporabljata največ v zvezi s časovniki, kar bomo podrobneje spoznali v naslednjem razdelku. Pri tem je pomemben operator `now`, ki nam vrne trenutni čas sistema (tipa `Time`). Specifikacijo strukturiranega tipa `struct` opišimo kar s primerom tipa `Oseba`:

```
newtype Oseba struct
  ime Charstring;
  priimek Charstring;
  leto_rojstva Integer;
endnewtype Oseba;
```

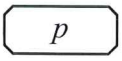
Podatkovni tip `Array` naj pojasni specifikacija

```
newtype Pretvorba
  Array (Character, Integer)
endnewtype Pretvorba;
```

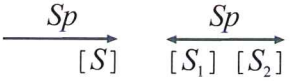
ki pomeni, da spremenljivka tega tipa vsaki vrednosti tipa `Character` priredi vrednost tipa `Integer`.

7.5.3 Specifikacija funkcionalnosti bloka

Za bloke, ki jih ne razgrajujemo več dalje, specificiramo njihovo funkcionalnost. Funkcionalnost bloka opišemo z enim ali več procesi. Vsi ti procesi delujejo hkrati in si med seboj izmenjujejo signale; signale pa si procesi lahko izmenjujejo tudi z okolico, torej z zunanjostjo bloka, katerega funkcionalnost opisujemo. Signali, ki si jih procesi izmenjujejo, so prav taki kot tisti, ki si jih izmenjujejo bloki. Navidezne poti, po katerih si procesi izmenjujejo signale, imenujemo signalne poti. Signalno pot opišemo pri funkcionalnosti bloka na enak način kot kanal pri strukturi bloka, le da signalne poti nikoli nimajo zakasnitev. Bistvena razlika med kanalom in signalno potjo pa je predvsem v tem, da predstavlja kanal dejanski sestavni del bloka, signalna pot pa le koncept medprocesne komunikacije. Signalna pot povezuje med seboj dva procesa, ali pa proces z okvirjem bloka, katerega funkcionalnost specificiramo. V slednjem primeru gre za vhodne oziroma izhodne signale bloka. Poleg procesov in signalnih poti lahko pri opisu funkcionalnosti bloka uporabimo tudi tekstovne deklaracije. Za proces uporabimo simbol

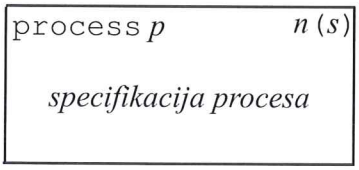


kjer je p ime procesa, za signalne poti pa, podobno kot v primeru kanalov, enega izmed simbolov



Specifikacija procesa

Procesi so torej poleg signalov osnovni gradniki opisa funkcionalnosti. Proces definiramo v podobnem okvirju, kot smo to že pokazali za blok.



S procesom torej opišemo del funkcionalnosti bloka. Model SDL procesa je v bistvu razširjeni končni avtomat. Proces ima vhode in izhode, saj komunicira z ostalimi procesi s pomočjo vhodnih in izhodnih signalov (sporočil). Proces ima tudi svoje stanje, ki se

kaže v vrednosti njegovih internih spremenljivk. Poleg tega pa ima lahko proces tudi svoje časovnike. Vhodne dogodke procesa predstavlja sprejem vhodnih signalov ali iztek časovnikov. To so torej dogodki, ki sprožijo spremembo stanja procesa. Izhodne dogodke predstavlja oddaja izhodnih signalov. Sprememba stanja procesa se kaže v spremembi vrednosti ene ali več spremenljivk.

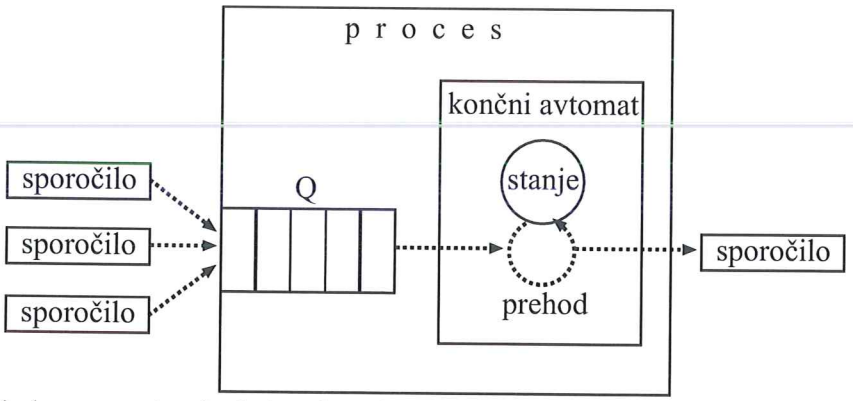
Vsaka spremenljivka SDL procesa pripada določenemu podatkovnemu tipu. Spremenljivke deklariramo v okviru specifikacije SDL procesa. Deklaracija spremenljivk izgleda takole:

`dcl` ime_spremenljivke *podatkovni_tip* ;

kjer torej *podatkovni_tip* definira množico vrednosti, ki jih spremenljivka z imenom *ime_spremenljivke* lahko zavzame. Časovnik procesa takole deklariramo:

`timer` *ime_časovnika* ;

Čeprav lahko sprejema proces sporočila od različnih drugih procesov, vsebuje model SDL procesa eno samo vhodno čakalno vrsto, kamor se uvrščajo vsa vhodna sporočila, vključno z izteki časovnikov, po vrsti, kakor prihajajo, proces pa ta sporočila jemlje iz vrste v istem vrstnem redu (torej gre za vrsto tipa FIFO - First In, First Out). Proces se vedno nahaja v nekem stanju ali pa prehaja iz enega stanja v drugo. Glede na stanje, v katerem je, in glede na sporočilo, ki ga je pravkar dobil iz vhodne čakalne vrste, bo proces izvedel določeno akcijo in prešel v neko drugo stanje. Akcija tu pomeni, da proces spremeni vrednost ene ali več svojih spremenljivk, sprejme kakšne logične odločitve, vpliva na svoje časovnike in/ali pošlje kakšnemu drugemu procesu sporočilo. Prehod iz enega v drugo stanje se teoretično zgodi v trenutku. Dogajanju v procesu torej dajejo časovno dimenzijo le vhodni dogodki in njegovi lastni časovniki. Opisani model prikazuje naslednja slika.



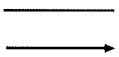
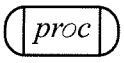
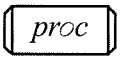
Osnovni element opisa funkcionalnosti SDL procesa je torej zaporedje

staro stanje → *sprejem sporočila* → *zaporedje akcij* → *novo stanje* ,

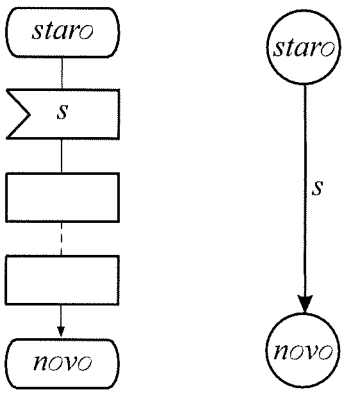
pri čemer pa je akcija bodisi sprememba vrednosti spremenljivk procesa, sprejem odločitev, upravljanje s časovnikom in/ali oddaja sporočila.

V grafični obliki jezika SDL uporabljamo za opis funkcionalnosti procesa simbole, ki jih kaže naslednja tabela. Specifikacija funkcionalnosti procesa v grafični obliki je zato podobna diagramom poteka, ki jih večkrat uporabljamo pri načrtovanju ali opisu programov.

simbol	**pomen**
	začetni simbol - Začetni simbol definira začetek delovanja procesa; prehod iz tega simbola v simbol nekega drugega stanja predstavlja inicializacijo procesa.
stanje	*stanje* - V tem simbolu je zapisano ime stanja. Tak simbol uporabimo na začetku in koncu prehoda iz stanja v stanje.
s *s* (*v*)	*vhodno sporočilo* - Simbol vhodnega sporočila pomeni, da je proces pravkar dobil iz vhodne čakalne vrste sporočilo (vhodni signal ali iztek časovnika) *s*; temu lahko dodamo še, kateri proces je to sporočilo poslal, če se to ne razume samo po sebi; če signal *s* nosi s seboj vrednost, lahko z zapisom *s* (*v*) v simbolu to vrednost shranimo v spremenljivko z imenom *v*.
s *s* (*v*)	*izhodno sporočilo* - Simbol izhodnega sporočila pomeni, da je proces oddal signal *s*; temu lahko dodamo še, kateremu procesu je to sporočilo namenjeno, če se to ne razume samo po sebi; če signal *s* nosi s seboj neko vrednost, določimo to vrednost kot vrednost spremenljivke *v* z zapisom *s*(*v*) v simbolu.
s	*shrani vhodno sporočilo* - Če dobi proces iz vhodne čakalne vrste sporočilo *s*, ga niti ne obdela niti ne zavrže, ampak to sporočilo ostane v vhodni čakalni vrsti.
stavek	*aktivnost* - V takem okvirju opišemo neko aktivnost procesa z enim od treh naslednjih formalnih stavkov: prireditveni stavek, proženje časovnika, ustavitev časovnika; lahko pa opišemo neko dogajanje tudi v neformalni obliki.

x := *expr*	*prireditveni stavek* - S prireditvenim stavkom, ki ga zapišemo v simbolu aktivnosti, podamo spremembo vrednosti spremenljivke procesa; *x* je ime spremenljivke, *expr* pa izraz
set (*t*, *id*)	*proženje časovnika* - S tem stavkom sprožimo časovnik *id* tako, da se bo iztekel ob času *t*; ker pa največkrat želimo določiti, po kolikem času naj se časovnik izteče, lahko uporabimo namesto *t* izraz now+*d*, kjer pomeni now trenutni čas, *d* pa čas izteka časovnika.
reset (*id*)	*ustavitev časovnika* - S tem stavkom ustavimo časovnik *id*.
	*kretnica* - Kretnica predstavlja logično razvejišče. Nadaljnje akcije so odvisne od vrednosti izraza *expr* v simbolu.
	*povezava* - Črta ali puščica povezuje ostale simbole med seboj in tako kaže zaporedje izvajanja; puščico uporabimo pri prehodu v novo stanje.
	*začetek procedure* - S tem simbolom se začne specifikacija procedure z imenom *proc* (podprograma).
	*konec procedure* - S tem simbolom se zaključi definicija procedure, potek dogajanja pa se s tem prenese na mesto, od koder je bila procedura klicana.
	*klicanje procedure* - s tem simbolom na nekem mestu kličemo proceduro *proc*, ki je definirana na drugem mestu

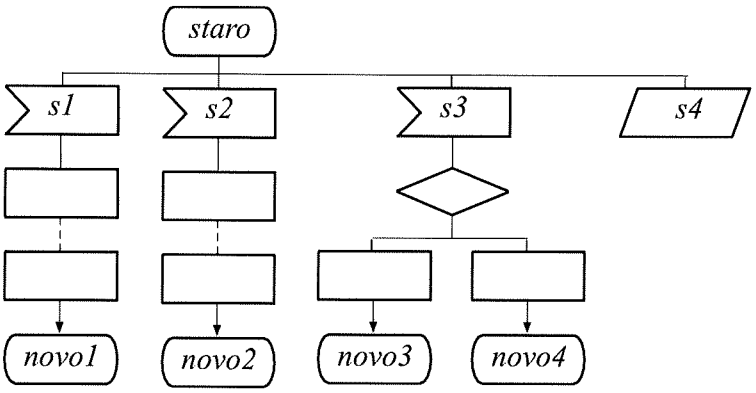
Sedaj lahko torej osnovno shemo prehoda iz enega stanja v drugo podamo tudi grafično. To prikazuje naslednja slika (a) skupaj z diagramom prehoda stanj ustreznega končnega avtomata (b).



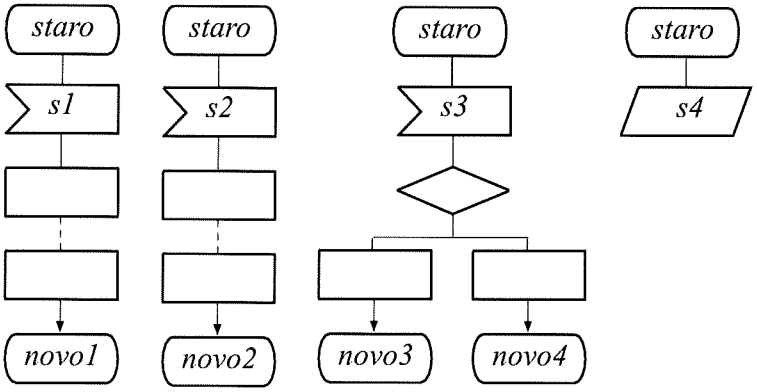
(a) (b)

V simbolu *vhodno sporočilo* je navedeno sporočilo, ki povzroči, če oziroma ko ga proces sprejme, prehod iz enega stanja v drugo. Seveda pa je iz nekega stanja lahko možnih več prehodov v različna ali isto novo stanje, ki jih vzbudijo različna sporočila. Poleg tega se lahko en prehod s pomočjo kretnice cepi v dva ali več prehodov. so Vsa tista sporočila, ki

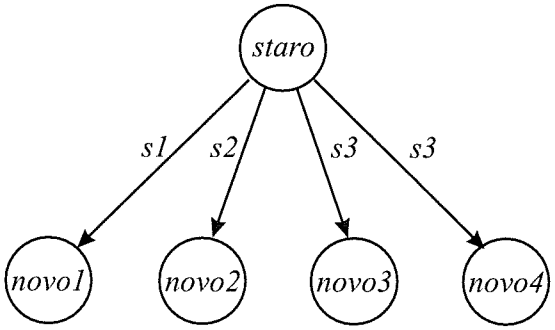
niso omenjena v nobenem simbolu *vhodno sporočilo* pri prehodu iz nekega stanja, so izgubljena, če so v tem stanju sprejeta. Če neko sporočilo ne sme biti izgubljeno, ampak mora počakati v vhodni čakalni vrsti na sprejem v ustreznem stanju, navedemo to sporočilo v simbolu *shrani vhodno sporočilo*. Naslednja slika specificira štiri možne prehode iz stanja *staro* v neko novo stanje (*novo1*, *novo2*, *novo3* oziroma *novo4*), ki jih vzbudijo tri sporočila *s1*, *s2* oziroma *s3*. Če se na vhodu v proces pojavi sporočilo *s4*, se shrani v vhodni čakalni vrsti. Vsa ostala vhodna sporočila (npr. *s5*) pa proces v stanju *staro* zavrže.



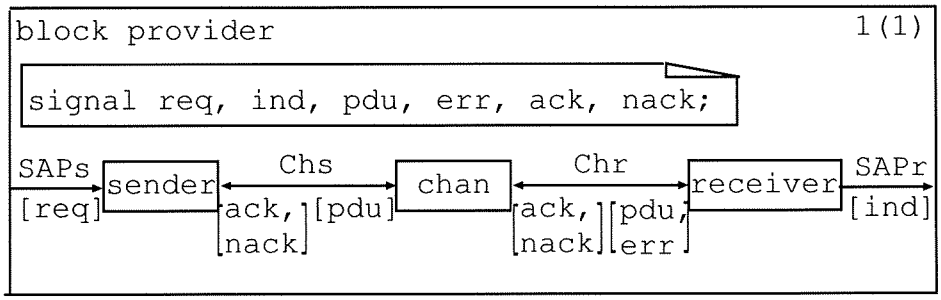
Tej sliki je enakovredna naslednja slika:



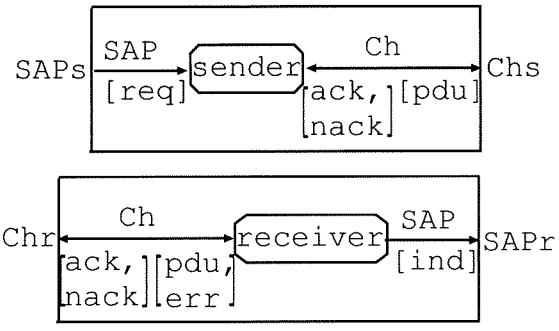
Lahko pa za tak primer narišemo tudi diagram prehajanja stanj končnega avtomata:



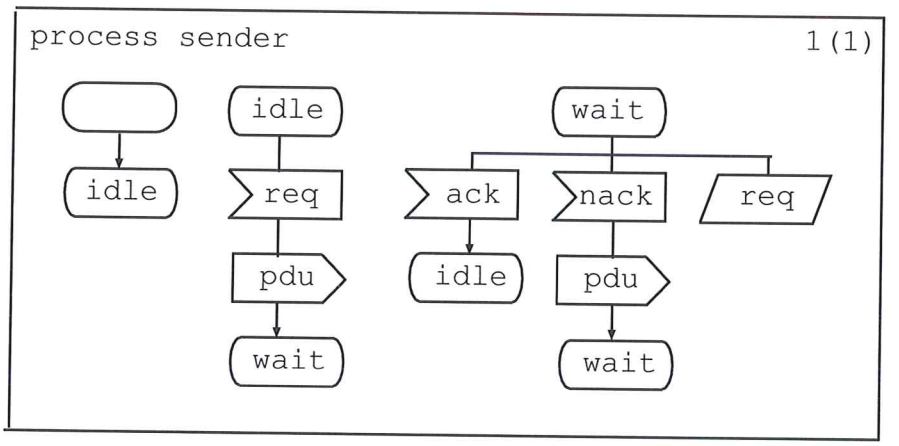
Zgled - V tem zgledu si bomo ogledali komunikacijo med dvema partnerjema - oddajnikom in sprejemnikom, ki sta sposobna popravljati napake, do katerih pride na poti skozi kanal, tako da oddajnik pokvarjene bloke ponovno pošilja. Pri tem bomo uporabili izredno preprost protokol, ki smo ga že opisali v razdelku "Implementacija storitve, protokol" poglavja "Storitve in protokoli". Najprej si oglejmo strukturo bloka `provider`, ki uporabniku skozi točki dostopa `SAPs` in `SAPr` nudi storitev prenosa sporočil. Blok `provider` bomo sestavili iz treh podblokov: `sender` bo oddajnik, ki od zunaj sprejema zahteve `req`, v kanal `chan` pa oddaja sporočila `pdu`; `receiver` bo sprejemnik, ki iz kanala `chan` sprejme bodisi sporočilo `pdu` ali pa sporočilo `err` (to drugo seveda v primeru, ko se je originalno sporočilo `pdu` v kanalu pokvarilo), uporabniku pa odda sporočilo `ind`; `chan` pa bo kanal, v katerem se sporočilo `pdu` lahko pokvari, ne more pa se izgubiti. Poleg tega pa lahko sprejemnik pošlje kanalu in kanal oddajniku sporočilo `ack` ali sporočilo `nack`. Očitno smo tu predpostavili, da se sporočila, ki jih proces `receiver` pošilja procesu `sender`, ne morejo pokvariti. Strukturo bloka `provider` kaže naslednja slika.



Bloka `sender` in `receiver` bomo funkcionalno opisali s po enim procesom (glej naslednji sliki), funkcionalnega opisa bloka `chan` pa ne bomo podali.



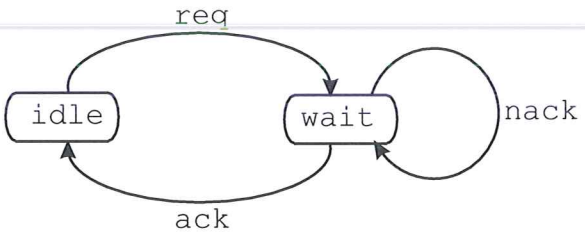
Funkcionalna specifikacija procesa sender je izredno preprosta, saj ta proces ne potrebuje niti internih spremenljivk♦, niti časovnika. Res pa je tudi, da protokolna sporočila pdu v tem preprostem primeru sploh ne prenašajo nikakršne uporabniške informacije. Uporabnik na sprejemni strani torej od takega protokola nima druge koristi, kot da lahko šteje zahteve, ki jih je postavil uporabnik na oddajni strani! Oglejmo si to specifikacijo v naslednji sliki.



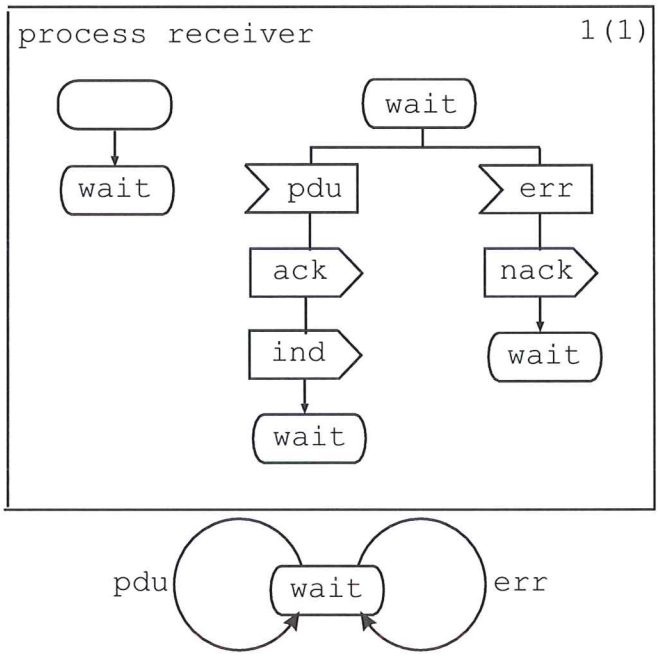
Vidimo, da je začetno stanje tega procesa `idle`, ko entiteta čaka na zahtevo uporabnika. Ko jo sprejme, pošlje protokolno podatkovno enoto `pdu` v kanal in v stanju `wait` čaka na potrditev. Če to potrditev (`ack`) dobi, spet čaka na novo zahtevo uporabnika (v stanju `idle`). Če pa dobi negativno potrditev `nack`, ponovno pošlje `pdu` in spet čaka na potrditev. Seveda pa v času, ko čaka na potrditev enega sporočila, ne more sprejeti zahteve po pošiljanju drugega. Morebitna nova zahteva mora torej tedaj, ko je `sender` v stanju `wait`, počakati v vhodni čakalni vrsti. Implementacija takega procesa mora torej vključevati tudi čakalno vrsto. V našem bloku `provider` bi seveda lahko čakalno vrsto tudi modelirali s posebnim procesom, v tem primeru pa ne bi potrebovali simbola *shrani vhodno sporočilo*. Tretja možnost pa bi bila, da bi storitev brez potrditve spremenili v storitev s potrditvijo; v tem primeru bi `sender` po sprejemu sporočila `ack` poslal uporabniku potrditev `confirm`, kar bi pomenilo, da lahko uporabnik pošlje `provider`-ju novo zahtevo. V tem primeru bi moral za čakalno vrsto poskrbeti sam uporabnik. Če na kratko povzamemo specifikacijo procesa `sender`, lahko ugotovimo, da ima ta proces

♦ Ta trditev ni popolnoma pravilna, saj predstavlja stanje procesa vrednost nekakšne njegove implicitne spremenljivke, ki pa je v SDL specifikaciji ni potrebno posebej deklarirati.

dve možni stanji. Možne prehode med tema stanjema v smislu končnega avtomata nam kaže naslednja slika.



Naslednji sliki pa nam kažeta specifikacijo procesa `receiver` in diagram prehodov med stanji tega procesa.



Funkcionalne specifikacije bloka `chan` tu ne bomo podajali. Bistvo pa je seveda v tem, da vhodno sporočilo `pdu` nespremenjeno prenese na izhod, ali pa naključno (z neko verjetnostjo) spremeni v sporočilo `err`. V vsakem primeru pa to sporočilo tudi zakasni.

Če bi želeli naš protokol napraviti vsaj kolikor toliko uporaben, bi moral vhodni signal `req` prinesti procesu `sender` uporabniško sporočilo, ta pa bi ga najprej začasno shranil in nato poslal kot parameter protokolnega sporočila `pdu`. Ko bi proces `receiver` to sporočilo sprejel, ga bi skupaj s primitivom `ind` predal uporabniku. ☺

Ob koncu pa še kratek komentar. Ni edina zasluga formalnega jezika ta, da nam omogoča nedvoumno specifikacijo telekomunikacijskega sistema in tudi služi kot osnova za strojno obdelavo. Že samo dejstvo, da skušamo specifikacijo npr. protokola "spraviti"

na papir, nas prisili k razmišljanju. Pri zadnjem zgledu smo gotovo pomislili, kaj bi se zgodilo, če bi se lahko pokvarile tudi potrditve. Kako bi morali spremeniti protokol, da bi bil kos tudi taki situaciji?

7.6 Drugi formalizmi za specifikacijo protokolov

V tem poglavju smo omenili dva formalna jezika: ASN.1 in SDL. Čeprav je prvega definirala ISO, drugega pa ITU-T, je mogoče uporabo obeh jezikov tudi kombinirati. O tem govori priporočilo ITU-T Z.105.

Čeprav s časovnimi shemami, ki smo jih omenili v tem poglavju, pa tudi že prej, ne moremo popolnoma in nedvoumno specificirati protokolov, pa vendarle predstavljajo tako praktičen pripomoček pri načrtovanju ali analizi protokolov, da jih je ITU-T v razširjeni obliki specificirala v priporočilu ITU-T Z.120 pod imenom **Message Sequence Charts** ali krajše **MSC**.

Formalnih jezikov za specifikacijo protokolov je še mnogo, dva izmed najbolj znanih pa sta Estelle in Lotos. Oba je standardizirala ISO.

Estelle (ISO standard IS9074) je jezik, namenjen specifikaciji protokolov. Podobno kot funkcionalni del SDL, je tudi Estelle osnovan na razširjenem konceptu končnega avtomata, sintaksa jezika pa izhaja iz programskega jezika Pascal.

Lotos (ISO standard IS8807) je jezik, ki je bolj abstrakten kot SDL ali Estelle, saj izhaja iz algebre procesov (matematičnega orodja za obravnavo sočasnih procesov). Njegova matematična naravnanost pomeni, da je primernejši za dokazovanje pravilnosti specificiranih protokolov. Če je bolj abstrakten, pa to pomeni, da je bolj nevtralen glede možnih implementacij, kar je vsekakor njegova prednost pred SDL in Estelle. Je pa zaradi tega težji za razumevanje. Ime Lotos je kratica za Language Of Temporal Ordering Specifications.

8 DETEKCIJA IN POPRAVLJANJE NAPAK V SPOROČILIH

8.1 Izvor in pogostnost napak

V vseh fizikalno izvedljivih oddajnih in sprejemnih napravah ter prenosnih poteh je vedno prisoten šum, zaradi vpliva drugih sistemov, ki delujejo v bližini prenosne poti, pa se v tej pojavljajo tudi motnje (presluh med prenosnimi potmi, industrijske motnje...). Šum in motnje se superponirajo koristnemu signalu na prenosni poti, zaradi česar lahko sprejemnik eno ali več sprejetih binarnih vrednosti napačno interpretira (kot 1 namesto 0 ali obratno). Pravimo, da se je zgodila bitna napaka. Vzroki za nastanek napak pa se lahko nahajajo tudi v sami prenosni poti (npr. linearna in nelinearna popačenja, disperzija, intersimbolna interferenca, odboji zaradi nepravilne zaključitve linije...).

Napake se pojavljajo bolj ali manj naključno, zaradi česar jih lahko le statistično ovrednotimo. Najpomembnejša mera za vrednotenje vpliva napak na prenos informacije je **pogostnost bitnih napak** *ber* (ang. **bit error rate**), ki jo definiramo kot povprečno vrednost razmerja števila pokvarjenih binarnih vrednosti proti številu vseh oddanih binarnih vrednosti (seveda v istem, običajno dolgem časovnem razdobju). Pogostnost bitnih napak pa hkrati pomeni tudi verjetnost p_{Ber}, da se bo oddana binarna vrednost pri prenosu pokvarila.

Porajanje napak je torej naključen proces, ki pa ima lahko različne, od okoliščin in vzrokov za nastanek napak odvisne lastnosti. V praksi pogosto predpostavimo Poissonov proces (ki smo ga v tem delu že omenili), kar pomeni, da je okvara ene prenešene binarne vrednosti popolnoma neodvisna od (ne)uspeha vseh ostalih binarnih vrednosti. Če bi bila prisotnost belega šuma edini vzrok za nastanek napak, bi predpostavka o Poissonovem procesu držala, v realnih okoliščinah pa ne drži. Predvsem zaradi vpliva motenj se binarne napake običajno pojavljajo v skupinah (rafalih).

V višjih slojih protokolnega sklada nas prenos posameznih binarnih vrednosti ne zanima več, saj prenašamo protokolna sporočila. Zato nas bo v takih primerih bolj zanimala **pogostnost napačno prenešenih sporočil** oziroma **verjetnost napačno prenešenega sporočila** p_{Per}, pa tudi verjetnost, da bo sporočilo prenešeno brez napake p_{Ps}. Prenešeno sporočilo bo seveda napačno, če bo napačno prenešena vsaj ena binarna

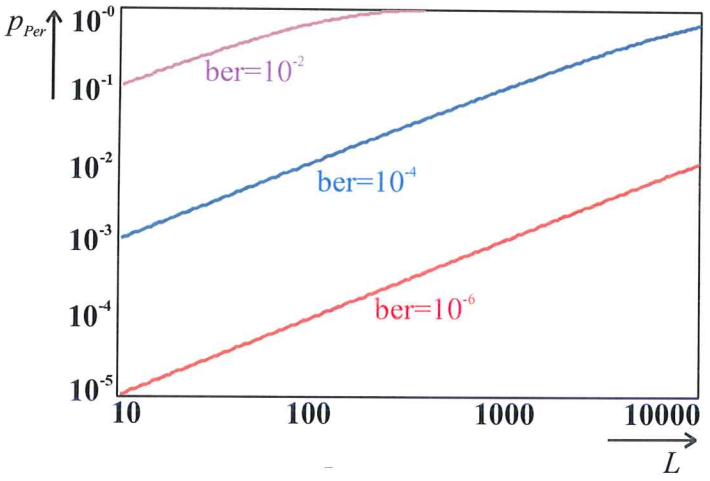
vrednost v njem. Brez globokega razmišljanja lahko torej ugotovimo, da bo pri neki dani pogostnosti bitnih napak p_{Ber} pogostnost napačno prenešenih sporočil tem večja, čim daljša bodo sporočila. Če pa predpostavimo Poissonov proces porajanja napak, lahko takole sklepamo: pri verjetnosti p_{Ber}, da bo napačno prenešena ena binarna vrednost, bo verjetnost, da bo ena binarna vrednost pravilno prenešena, enaka $1-p_{Ber}$, verjetnost, da bodo pravilno prenešene vse binarne vrednosti sporočila z dolžino L bitov, in s tem tudi verjetnost, da bo sporočilo v celoti pravilno prenešeno, pa bo

$$p_{Ps} = (1-p_{Ber})^L \ . \tag{44}$$

Pogostnost napačno prenešenih sporočil bo potem

$$p_{Per} = 1-p_{Ps} = 1-(1-p_{Ber})^L \ . \tag{45}$$

Kot vidimo v sliki Slika 8-1, ki je risana po enačbi 6, pogostnost napačno prenešenih sporočil res narašča z naraščanjem povprečne dolžine sporočila L.



Slika 8-1. Pogostnost napačnih sporočil

8.2 Paritetna metoda

Ena izmed prvih, pa tudi najpreprostejših metod za zaščito podatkov pred napakami v telekomunikacijskem kanalu, je bila in je še vedno paritetna metoda. Poznamo dve različici - **metodo sode paritete** in **metodo lihe paritete**. V obeh primerih dodamo vsakemu sporočilu pred oddajo eno binarno vrednost tako, da je celotno število logičnih enic v sporočilu (s paritetnim bitom vred!) sodo v primeru sode paritete oziroma liho v primeru lihe paritete. Če se v tako zaščitenem bloku pokvari liho število binarnih

vrednosti, bo sprejemnik sprejel blok z lihih številom enic v primeru sode paritete oziroma blok s sodim številom enic v primeru lihe paritete. Ker pa morata biti oddajni in sprejemni protokolni osebek pred oddajo dogovorjena za način zaščite, bo v takem primeru sprejemnik seveda takoj vedel, da je pri prenosu bloka prišlo do napake. Metoda paritete torej omogoča detekcijo napak, če je v prenešenem bloku podatkov prišlo do lihega števila binarnih napak, medtem ko sodega števila napak v bloku po tej metodi ne moremo odkriti.

Čeprav ima blok podatkov, ki ga zaščitimo po paritetni metodi, lahko kakršno koli dolžino, so to metodo pogosto uporabljali za zaščito prenašanih znakov. V Tabeli 8-1 podajamo zgled, v katerem smo besedilo "PROTOKOL", zakodirano s pomočjo ASCII kode, zaščitili po metodi sode paritete. Paritetni bit je bit 0.

Tabela 8-1. Primer zaščite besedila po metodi sode paritete

znak	*7*	*6*	*5*	*4*	*3*	*2*	*1*	*0*
P	1	0	1	0	0	0	0	0
R	1	0	1	0	0	1	0	1
O	1	0	0	1	1	1	1	1
T	1	0	1	0	1	0	0	1
O	1	0	0	1	1	1	1	1
K	1	0	0	1	0	1	1	0
O	1	0	0	1	1	1	1	1
L	1	0	0	1	1	0	0	1

Paritetno metodo so zasnovali na predpostavki, da je verjetnost ene bitne napake v bloku (liho število!) precej večja od verjetnosti dveh bitnih napak v bloku (sodo število!), zaradi česar naj bi po paritetni metodi odkrili večino napak. Vendar pa pri večjih pogostnostih napak in pri pojavljanju napak v skupinah (rafalih) ta predpostavka ne drži več. Moč zaščite po paritetni metodi je zato relativno šibka (delež neodkritih napak je relativno velik). Vendar pa je poglavitna privlačnost paritetne metode njena izredna preprostost. Paritetni bit je pri metodi sode paritete namreč kar vsota po modulu 2 vseh ostalih binarnih vrednosti v bloku, pri metodi lihe paritete pa njegov komplement (negacija). Vsota po modulu 2 vseh bitov (vključno s paritetnim) v bloku mora biti v sprejemniku pri metodi sode paritete 0, pri metodi lihe paritete pa 1, sicer sprejeti blok zagotovo vsebuje vsaj eno napako. Tako računanje paritetnega bita na oddajni strani, kot tudi preverjanje pravilnosti na sprejemni strani je torej izredno preprosto, bodisi v programski ali v strojni izvedbi. Možnost strojne izvedbe zaščite in detekcije napak je

zelo pomembna, saj je detekcija napak pomembna funkcija kanalnega sloja po referenčnem modelu OSI, kanalni osebki pa so največkrat izvedeni strojno.

8.3 Metoda seštevanja

Pri metodi seštevanja (ang. **checksum**) zaščitimo prenašano sporočilo tako, da dodamo zaščitne bite po nekoliko kompleksnejšem algoritmu kot pri paritetni metodi, temu ustrezno pa metoda seštevanja nudi močnejšo zaščito, kar pomeni, da je delež po tej metodi detektiranih napak večji.

Pri metodi seštevanja sporočilo razdelimo na delna zaporedja dolžine n bitov – **besede**. Te besede obravnavamo kot vrednosti in jih seštejemo po modulu 2^n, kar pomeni, da od dobljene vsote ohranimo le n bitov z najmanjšo težo (oziroma n desnih bitov), tako dobljeno vsoto pa dodamo prenašanemu sporočilu kot zaščitno besedo. Sprejemnik seveda tudi sam računa vsoto po istem modulu in preveri, ali se izračunana vsota ujema s tisto, ki mu jo je poslal oddajnik. Če se ne, to pomeni, da je pri prenosu prišlo do napake. V sliki 3 vidimo sporočilo, sestavljeno iz petih 4-bitnih besed, vsoto vseh besed ter po modulu 2^4=16 izračunano vsoto vseh besed, v tabeli pa vidimo oddano sporočilo.

	računanje vsote	poslano sporočilo	
1. beseda	1011	1011	1. beseda
2. beseda	0101	0101	2. beseda
3. beseda	1000	1000	3. beseda
4. beseda	1100	1100	4. beseda
vsota	100100	0100	zaščitna beseda
vsota po modulu 16	0100		

Slika 8-2. Zaščita sporočila po metodi seštevanja

Če dobro premislimo, lahko ugotovimo, da je paritetna metoda pravzaprav le poseben primer metode seštevanja, kjer sporočilo sestavlja zaporedje enobitnih besed, ki jih torej seštevamo po modulu 2.

8.4 Metoda krožnega redundančnega preizkusa

V tem razdelku si bomo ogledali na videz še bolj zapleten način izračunavanja zaščitne kode, ki pa nam prinese še učinkovitejšo detekcijo napak, poleg tega pa ne zahteva razdelitve sporočila na besede, kot to zahteva metoda seštevanja. Gre za metodo krožnega redundančnega preizkusa, znano pod kratico **CRC** (ang. **Cyclic Redundancy**

Check). Kodo CRC štejemo med **polinomske kode**, katerih teoretična razlaga temelji na predstavitvi zaporedij binarnih vrednosti kot polinomov.

Neko zaporedje m binarnih vrednosti $M=\mu_{m-1}\mu_{m-2}...\mu_1\mu_0$ lahko predstavimo kot polinom

$$M = \mu_{m-1} \cdot X^{m-1} + \mu_{m-2} \cdot X^{m-2} + ... \mu_1 \cdot X^1 + \mu_0 \cdot X^0 \ , \tag{46}$$

kjer binarni koeficienti μ_i predstavljajo elemente zaporedja. Seveda lahko s polinomi tudi računamo – lahko jih seštevamo, odštevamo, množimo in delimo. Definicije odštevanja, množenja in deljenja temeljijo na definiciji vsote. V našem primeru bomo operacije odštevanja, množenja in deljenja osnovali na operaciji seštevanja po modulu 2, katerega pravilnostno tabelo vidimo v tabeli 8-2.

Tabela 8-2. Seštevanje po modulu 2

x	y	$x \oplus y$
0	0	0
0	1	1
1	0	1
1	1	0

Pri tem opozorimo zlasti na dejstvo, da ima pri računanju po modulu 2 vsota $1 \oplus 1$ vrednost 0 in ne 10, kar pomeni, da pri seštevanju ne uporabljamo prenosa vrednosti na bit z višjo težo. Z drugimi besedami to pomeni, da pri seštevanju dveh polinomov po modulu 2 seštevamo po modulu 2 pare istoležnih bitov oziroma pare bitov z isto potenco spremenljivke X neodvisno enega od drugega.

Množenje polinoma (3) s faktorjem X^r pomeni isto kot dodajanje r ničel na koncu zaporedja M. Zato lahko stik zaporedij (3) in $R=\rho_{r-1}\rho_{r-2}...\rho_1\rho_0$ v polinomski obliki takole zapišemo:

$$T = M \bullet R = \mu_{m-1}\mu_{m-2}...\mu_1\mu_0\rho_{r-1}\rho_{r-2}...\rho_1\rho_0 = M \cdot X^r \oplus R \ . \tag{47}$$

Naj bo zaporedje M sporočilo dolžine m, ki ga želimo zaščititi, R pa zaporedje zaščitnih bitov dolžine r. Oddajnik bo torej oddal zaporedje T iz enačbe (4). Izberimo zaporedje zaščitnih bitov R tako, da bo oddano zaporedje T deljivo brez ostanka z nekim izbranim zaporedjem $G=\gamma_g\gamma_{g-1}...\gamma_1\gamma_0$ dolžine $g+1$, ki ga imenujemo **generacijski polinom**:

$$T = M \cdot X^r \oplus R = Q \cdot G \ . \tag{48}$$

To z drugimi besedami pomeni, da bo pri pri deljenju zaporedja T z generacijskim polinomom G ostanek nič. Ker pa je seštevanje po modulu 2 ekvivalentno odštevanju po modulu dva, lahko enačbo (5) zapišemo tudi v obliki

$$M \cdot X^r = Q \cdot G \oplus R \ , \tag{49}$$

kar pomeni, da dobimo zaporedje zaščitnih bitov R dolžine r kot ostanek pri deljenju sporočila M, podaljšanega z r ničlami, če je stopnja generacijskega polinoma g enaka številu zaščitnih bitov r,

$$g=r \ . \tag{50}$$

Postopek zaščite podatkov in detekcije napak po metodi krožnega redundančnega preizkusa lahko torej takole povzamemo. Oddajnik doda zaporedju M dolžine m, ki ga želi zaščititi, r ničel ter tako dobljeno zaporedje dolžine $m+r$ deli z generacijskim polinomom G stopnje g oziroma dolžine $g+1$. Z dobljenim ostankom R dolžine r nadomesti dodanih r ničel in tako dobljeno zaporedje dolžine $m+r$ pošlje sprejemniku. Le-ta sprejeto zaporedje deli z istim generacijskim polinomom G. Če pri tem deljenju dobi ostanek 0, sklepa, da pri prenosu ni prišlo do napake; če je ostanek od nič različen, je pri prenosu zagotovo prišlo do napake. Pri tem oddajnik in sprejemnik izvajata isti postopek deljenja, ki temelji na seštevanju po modulu 2 in ga imenujemo **dolgo deljenje**.

Na sliki Slika 8-3 lahko vidimo primer sporočila M=100110111, postopka dolgega deljenja z generacijskim polinomom G=1101, oddanega zaporedja T=100110111101 ter preverjanja pravilnosti v sprejemniku.

nezaščiteno sporočilo: M=100110111 generacijski polinom: G=1101

$M \cdot X^3$=100110111000

postopek dolgega deljenja:

```
100110111000:1101=111110001
 1001
  1000
   1011
    1101
     0001
      0010
       0100
        1000
         101
```

ostanek: R=101 poslano sporočilo: T=100110111101

preverjanje pravilnosti:

```
100110111101:1101=111110001
 1001
  1000
   1011
    1101
     0001
      0011
       0110
        1101
         000
```

ostanek je 0!

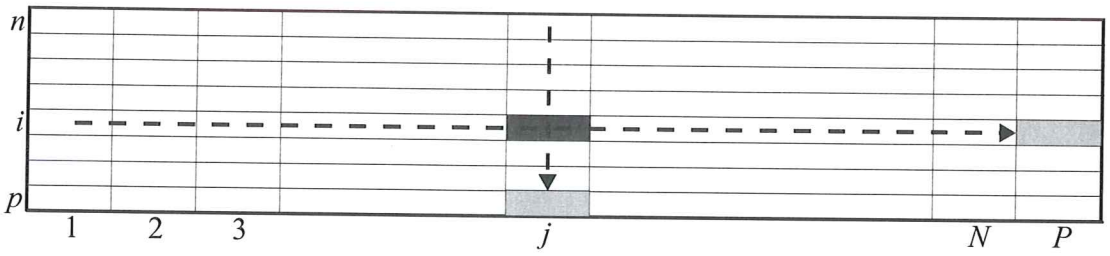
Slika 8-3. Zaščita podatkov in preverjanje pravilnosti po metodi CRC

V praksi uporabljamo standardne generacijske polinome različnih dolžin, ki pa so tako izbrani, da je delež detektiranih napak čim večji. Ker se da postopek dolgega deljenja zelo preprosto izvesti na strojni način, so za deljenje s standardnimi generacijskimi polinomi na voljo poceni integrirana vezja, kar uporabo metode redundančnega krožnega preizkusa, zlasti v kanalnem sloju protokolnega sklada, zelo poenostavi in poceni.

8.5 Metoda dvodimenzionalne paritete

V tem poglavju smo si že ogledali paritetno metodo. Pa vzemimo, da sporočilo spet zapišemo kot zaporedje N n-bitnih besed, vsaki besedi pa dodamo še paritetni bit p. Na koncu sporočila dodajmo še paritetno besedo P tako, da bo vsak bit paritetne besede paritetni bit zaporedja istoležnih bitov vseh ostalih besed. Sedaj lahko sprejemnik preverja pravilnost sprejetega sporočila dvodimenzionalno, torej po vrsticah in po stolpcih (glej sliko 9-6). Če se bo pri prenosu pokvaril i-ti bit j-te besede, bo sprejemnik

našel paritetno napako v *j*-ti besedi, prav tako pa tudi v *i*-tem bitu paritetne besede. Sprejemnik bo torej ne le detektiral napako, ampak jo bo tudi lociral – natanko bo vedel, kateri bit v sporočilu se je pokvaril. Ker pa ima vsak bit lahko le vrednost 0 ali 1, bo sprejemnik z invertiranjem pokvarjenega bita popravil celotno sporočilo. To bo seveda vedno mogoče, če bo v sporočilu nastala ena binarna napaka. V primeru več napak v sporočilu napake ne bo vedno možno popraviti. Postopek prikazuje Slika 8-4



Slika 8-4. Dvodimenzionalna paritetna zaščita

V zgledu v tabeli 8-3 vidimo dvodimenzionalno sodo zaščito besedila "PROTOKOL".

Tabela 8-3. Primer zaščite besedila po metodi dvodimenzionalne sode paritete

znak	*7*	*6*	*5*	*4*	*3*	*2*	*1*	*p*
P	1	0	1	0	0	0	0	0
R	1	0	1	0	0	1	0	1
O	1	0	0	1	1	1	1	1
T	1	0	1	0	1	0	0	1
O	1	0	0	1	1	1	1	1
K	1	0	0	1	0	1	1	0
O	1	0	0	1	1	1	1	1
L	1	0	0	1	1	0	0	1
PAR	0	0	1	1	1	1	0	0

8.6 Posplošena obravnava bločnega kodiranja

Doslej smo si že ogledali nekaj načinov kodiranja sporočil, ki sprejemniku omogočajo detekcijo ali celo popravljanje napak, nastalih pri prenosu. V vseh primerih smo sporočilu z dolžino *m* bitov dodali *r* zaščitnih bitov, ki seveda niso del uporabniškega sporočila in torej predstavljajo režijo pri prenosu informacije. Pogosto jih imenujemo **redundantni biti**. Razmerje

$$\rho = \frac{r}{m+r} \tag{51}$$

lahko torej imenujemo stopnja redundance, doprinos redundantnih bitov k faktorju režije pa je

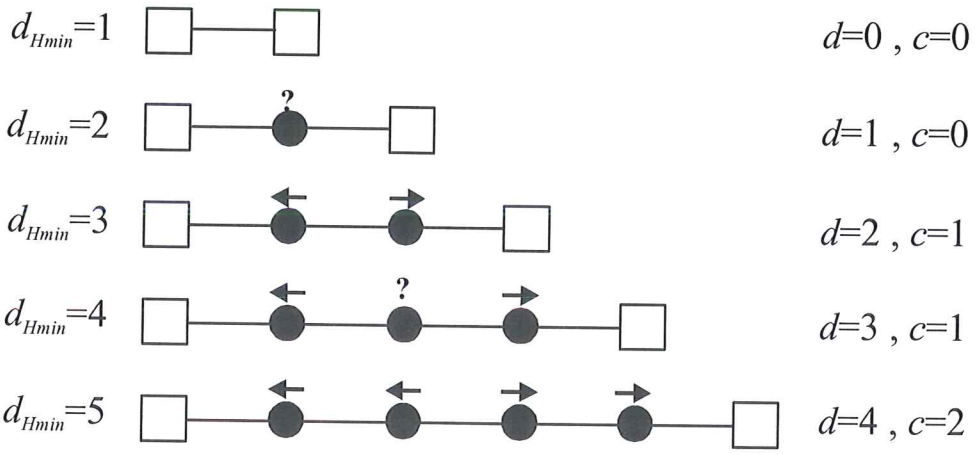
$$k_\rho = 1 - \rho = \frac{m}{m+r} \,. \tag{52}$$

Poleg tega pa smo lahko opazili, da smo doslej vedno zaščitili neko sporočilo oziroma blok podatkov kot celoto. Tak način kodiranja imenujemo **bločno kodiranje**, ustrezne kode pa **bločne kode**.

Z dodajanjem r zaščitnih bitov m bitom sporočila dobimo blok z dolžino $m+r$. Ker vsak način kodiranja natančno in enoumno predpisuje način določanja r zaščitnih bitov k m bitom sporočila, lahko oddajnik generira le 2^m veljavnih binarnih zaporedij dolžine $m+r$, ki jih imenujemo **kodne besede**, kljub temu, da je število vseh možnih binarnih zaporedij 2^{m+r}. Medtem ko oddajnik seveda generira in oddaja le veljavna zaporedja, torej kodne besede, lahko sprejemnik zaradi napak v kanalu sprejme katero koli izmed 2^{m+r} kombinacij. Če bo sprejemnik sprejel veljavno kodno besedo, bo predpostavil, da do napake ni prišlo, sicer pa bo vedel, da je pri prenosu kodne besede skozi kanal prišlo do ene ali več binarnih napak. Seveda pa bo prav mogoče, da je v kanalu prišlo do več napak tako, da sprejemnik kljub tem napakam spet sprejme veljavno kodno besedo, vendar ne tiste, ki je bila oddana. Ker pa sprejemnik ne ve, kaj je oddajnik v resnici oddal, takih napak z uporabo dane metode kodiranja seveda ne moremo niti odkriti niti popraviti.

Sposobnost nekega postopka kodiranja pogosto ocenjujemo na podlagi **Hammingove razdalje**. Hammingova razdalja d_H med dvema kodnima besedama je definirana kot število binarnih vrednosti, v katerih se ti dve besedi med seboj razlikujeta. Predvsem pa je pomembna **minimalna Hammingova razdalja kode** d_{Hmin}, ki je definirana kot minimalno število binarnih vrednosti, v katerih se med seboj razlikujeta poljubni dve kodni besedi. Pri prenosu skozi kanal se mora torej spremeniti vsaj d_{Hmin} binarnih vrednosti v eni veljavni kodni besedi, da sprejemnik spet sprejme veljavno kodno besedo, vendar različno od tiste, ki je bila poslana. Odvisnost sposobnosti postopka kodiranja za detekcijo in korekcijo napak od minimalne Hammingove razdalje lahko vidimo v sliki Slika 8-5, kjer prazni kvadrati predstavljajo veljavne kodne besede, polni krogi neveljavne besede, d je število napak v sporočilu, ki jih pri določeni kodi lahko detektiramo, c pa število napak, ki jih lahko popravimo. Puščice označujejo napake, ki jih lahko popravimo, vprašaji pa tiste, ki jih lahko le odkrijemo. Če je minimalna Hammingova razdalja enaka 1, dobimo v najslabšem primeru s spremembo ene binarne

vrednosti v veljavni kodni besedi drugo veljavno kodno besedo; taka koda torej ne predstavlja nobene zaščite – ne omogoča niti detekcije niti popravljanja napak. Če redundantnih bitov sploh ne dodajamo, je Hammingova razdalja med poljubnima kodnima besedama, s tem pa seveda tudi minimalna Hammingova razdalja, enaka 1. Če je minimalna Hammingova razdalja 2, bomo zaradi enojne napake zagotovo dobili neveljavno sporočilo in ga detektirali, pri večkratni napaki pa lahko sprejmemo veljavno kodno besedo. Koda z enim paritetnim bitom ima Hammingovo razdaljo 2. V primeru $d_{Hmin}=3$ se bo sporočilo, ki ga bomo sprejeli ob eni sami binarni napaki, razlikovalo od oddane kodne besede le v enem bitu, od vseh ostalih veljavnih kodnih besed pa vsaj v dveh bitih. Če bomo napake popravljali, bomo sprejeto sporočilo, ki ne bo enako veljavni kodni besedi, popravili na najbližjo kodno besedo; v primeru enojne napake po ta popravek pravilen, sicer pa bo napačen. Če pa bomo napake le detektirali, bomo lahko odkrili napačno sporočilo v vseh primerih, ko je prišlo v kanalu do največ dveh bitnih napak. Na enak način lahko sklepamo, da bomo v primeru $d_{Hmin}=4$ lahko zanesljivo popravili sporočila z največ eno bitno napako in detektirali tista z največ tremi bitnimi napakami, v primeru $d_{Hmin}=5$ pa bomo lahko zanesljivo popravili sporočila z največ dvema in detektirali sporočila z največ štirimi bitnimi napakami.



Slika 8-5. Minimalna Hammingova razdalja in stopnja zaščite

Iz pravkar povedanega lahko sklepamo, da se bomo morali večinoma vnaprej odločiti, ali bomo na podlagi uporabljene zaščitne kode napake le odkrivali, ali pa jih bomo tudi popravljali. V prvem primeru bomo lahko zanesljivo detektirali napačno prenešena sporočila le v primeru, če se je v sporočilu pokvarilo največ d binarnih vrednosti,

$$d = d_{H\min} - 1 \ . \tag{53}$$

V drugem primeru pa bomo lahko sprejeto sporočilo zanesljivo pravilno popravili le v primeru, ko se je v sporočilu pokvarilo največ c binarnih vrednosti, kjer ima c vrednost

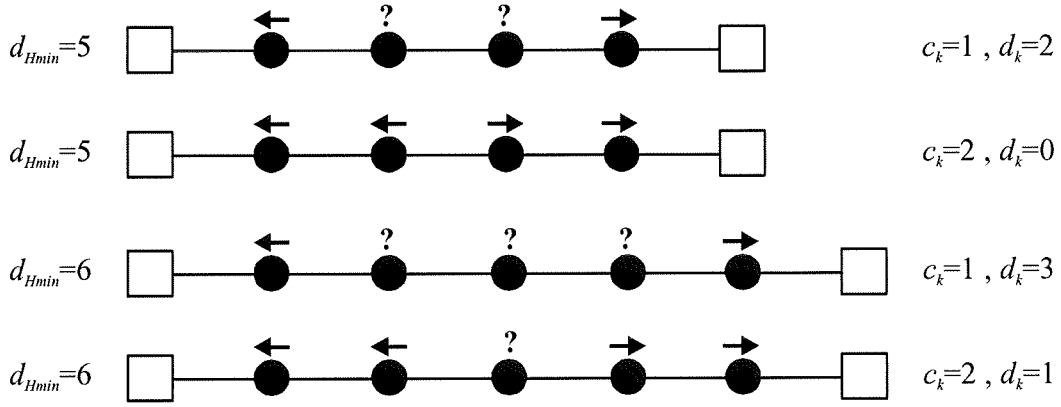
$$c = \frac{d_{H\min} - 1}{2} \ . \tag{54}$$

c je seveda lahko le celo število; razultat, ki ga dobimo po enačbi (11), moramo zaokrožiti navzdol.

Ker pa lahko sprejemnik ob sprejemu napačnega sporočila vedno ugotovi, kolikšna je Hammingova razdalja sprejetega sporočila do najbližje veljavne kodne besede, se lahko v primeru, da je ta razdalja manjša ali enaka neki vnaprej določeni vrednosti c_k, odloči, da bo napako popravil, sicer pa jo le detektira. Vrednost c_k mora biti pri tem manjša ali enaka vrednosti, ki jo določa pogoj (11). Največje število napak, ki jih na ta način lahko detektiramo, pa je določeno z enačbo

$$d_k = d_{H\min} - 2 \cdot c_k - 1 \ . \tag{55}$$

Na sliki 9-8 vidimo možnost kombiniranega popravljanja in odkrivanja napak za primer d_{Hmin}=5 in d_{Hmin}=6 ter c_k=1 in c_k=2.



Slika 8-6. Kombinirano popravljanje in odkrivanje napak

8.7 Prepletanje in konvolucijsko kodiranje

Moč bločnih kod zbledi, če se napake pojavljajo v izrazitih skupinah. Skupina napak je namreč lahko tako velika, da tudi sicer relativno močna metoda kodiranja ne zmore detektirati ali popraviti napak v preveč prizadetem bloku. Proti napakam v skupinah se lahko borimo na dva načina: s prepletanjem že zaščitenih podatkovnih blokov in s konvolucijskimi kodami.

Pri postopku **prepletanja** (ang. **interleaving**) oddajni osebek pred oddajanjem več že zaščitenih blokov premeša (preplete) med seboj in jih tako prepletene odda, sprejemnik pa z obratnim postopkom bite uredi v prvotnem vrstnem redu po blokih. Večja skupina napak bo tako sicer lahko prizadela večje število blokov, vendar vsakega izmed njih manj močno, zato bomo morda lahko z dovolj močno kodo popravili ali vsaj odkrili napake v večini okvarjenih blokov. Učinkovitost postopka prepletanja je torej tesno povezana z uporabljeno metodo kodiranja, seveda pa tudi s pričakovano dolžino izbruhov napak. Primer postopka prepletanja lahko vidimo na sliki Slika 8-7. Slaba lastnost prepletanja je večja zakasnitev, ki jo s tem postopkom vnašamo v prenos.

pred prepletanjem:

11	12	13	14	21	22	23	24	31	32	33	34	41	42	43	44	51	52	53	54

po prepletanju:

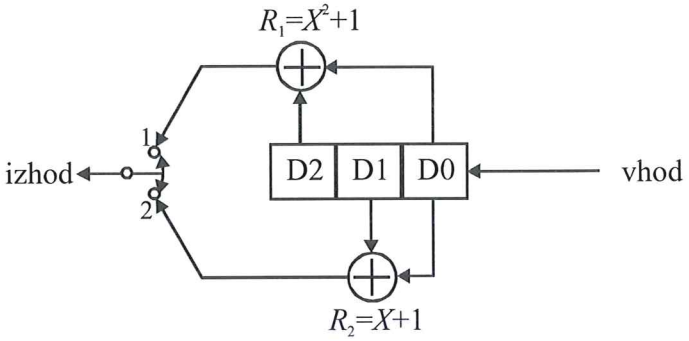
11	21	31	41	51	12	22	32	42	52	13	23	33	43	53	14	24	34	44	54

Slika 8-7. Prepletanje blokov

Pri **konvolucijskem kodiranju** s stopnjo redundance $(n-k)/n$ oddamo n kodnih bitov na vsakih k sporočilnih bitov. Zaporedje kodnih bitov lahko vključuje informacijske bite originalnega sporočila, lahko pa tudi ne. Izhodni kodni biti so odvisni od trenutnih vrednosti informacijskih bitov, pa tudi od več prejšnjih vrednosti informacijskih bitov, ki so shranjeni v pomikalnem registru kodirnega vezja. Ker je torej zaporedje kodnih bitov odvisno ne le od trenutnih, ampak tudi od preteklih vrednosti vhodnih bitov, gre za kodiranje s spominom. Pri tem je zelo pomembna največja možna razdalja med zaščitenim in zaščitnim bitom, saj je od te razdalje po eni strani odvisna potrebna velikost registra, ki ga potrebujemo za kodiranje oziroma dekodiranje, pa tudi računska kompleksnost, potrebna zlasti za dekodiranje, po drugi strani pa odpornost tako zaščitenega zaporedja proti daljšim skupinam binarnih napak. V prenašanem binarnem zaporedju ne moremo razločevati blokov, ki vsebujejo m sporočilnih bitov, in r tistih, ki jih ščitijo. Konvolucijsko kodiranje je zlasti primerno za zaščito **toka podatkov** (ang. **data stream**), ki ni organiziran po blokih. Tudi konvolucijske kode lahko, podobno kot bločne kode, omogočajo bodisi le odkrivanje, ali pa tudi popravljanje napak.

Konvolucijski koder je običajno sestavljen iz pomikalnega registra, nekaj seštevalnikov po modulu 2 in multipleksorja, ki izhodne bite multipleksira v izhodni tok.

Izhodne bite dobimo kot vsote vhodnega in zakasnjenih bitov po modulu dva. Te vsote pogosto podajamo s polinomi operatorja X, ki predstavlja zakasnitev vhodne binarne vrednosti. Dekoder, katerega namen je popravljanje napak, je precej bolj kompleksen, saj skuša rekonstruirati najverjetnejšo zgodovino vzbujanja. Na sliki Slika 8-8 vidimo preprost konvolucijski koder s stopnjo redundance 0,5, vhodno zaporedje in izhodno zaporedje v primeru, da je bil pred prihodom vhodnega zaporedja pomikalni register prazen (torej je vseboval same ničle).



vhod: 1011010
izhod: 11010110110101

Slika 8-8. Primer konvolucijskega koderja

Povedali smo že, da je postopek konvolucijskega kodiranja primeren predvsem za kodiranje toka podatkov. Če ta postopek uporabimo za kodiranje bloka podatkov, moramo bloku, ki ga želimo zaščititi, najprej dodati še nekaj slepih bitov (npr. samih ničel), ki omogočajo, da vsi uporabniški biti pri kodiranju uspejo priti skozi pomikalni register in so s tem deležni iste stopnje zaščite.

8.8 Odkrivanje in popravljanje napak

V tem poglavju smo najprej našteli najosnovnejše vzroke za nastajanje napak pri prenosu, nato pa smo spoznali nekaj najpreprostejših metod kodiranja, ki omogočajo bodisi samo odkrivanje, ali pa tudi popravljanje napak. To vrsto kodiranja v literaturi imenujejo **kanalsko kodiranje** (izraz gotovo ni naključno izbran, saj to vrsto kodiranja najpogosteje in najizdatneje uporabljamo v kanalskem sloju po modelu OSI).

Včasih nas pojav nekaj napak, še zlasti če se te poredko pojavljajo, ne moti kaj dosti. Tak primer je prenos govora ali slike. Tovrstna informacija namreč že sama po sebi vsebuje precej redundance, ki jo lahko na sprejemni strani uporabimo za "popravljanje" prenešene informacije (npr. z interpolacijo); pa tudi, če tega ne delamo, se bo pojav

napak ob sprejemu manifestiral le kot šum, ki pa bo lahko dovolj nemoteč in ne bo preveč kvaril kvalitete prenosa, če bo pogostnost napak pri prenosu dovolj majhna in zato tudi razmerje signalne in šumne moči pri sprejemu dovolj veliko.

Povsem drugače pa je seveda pri prenosu podatkov. Ti so pogosto tako občutljivi, da se tudi eno samo sporočilo ali paket pri prenosu ne sme pokvariti*. Tu imamo dve možnosti: bodisi napačno prenešeno sporočilo zavržemo, saj je bolje, da sprejemnik ne sprejme ničesar, kot da sprejme napačno sporočilo, ali pa pokvarjeno sporočilo popravimo. Slednje pa spet lahko storimo na dva različna načina. Pri metodi, ki jo bomo imenovali **popravljanje na sprejemni strani** (v angleščini se za ta postopek uporablja nekoliko ponesrečen izraz **Forward Error Correction – FEC**), oddajnik sporočila zaščiti s kanalsko kodo z dovolj veliko minimalno Hammingovo razdaljo, ki sprejemniku samemu, brez sodelovanja oddajnika, omogoča popravljanje napak, seveda ob predpostavki, da le-teh ni preveč. Pri metodi, ki ji bomo rekli **popravljanje s ponovnim pošiljanjem** (ang. **Backward Error Correction - BEC**), pa oddajnik uporabi kanalsko kodo z manjšo minimalno Hammingovo razdaljo, ki sprejemniku omogoča le odkrivanje napak, le-ta pa potem implicitno ali eksplicitno zahteva od oddajnika ponovno pošiljanje pokvarjenih sporočil; protokole, ki uporabljajo tako metodo popravljanja napak, pogosto imenujemo **protokoli s ponavljanjem** ali **protokoli ARQ** (ang. **Automatic Repeat reQuest – ARQ**).

Če na kratko povzamemo, imamo torej glede odkrivanja in popravljanja napak štiri osnovne možnosti:

- napak niti ne odkrivamo, niti ne popravljamo;
- napake odkrivamo, vendar napačno prenešenih sporočil ne popravljamo, ampak jih le zavržemo;
- napake popravljamo na sprejemni strani (torej jih oddajnik tako zaščiti, da jih lahko sprejemnik sam popravi);
- napake popravljamo s ponovnim pošiljanjem, torej uporabimo enega izmed protokolov ARQ.

* Predstavljajte si samo, da bi se npr. pri prenosu bančnih podatkov o stanju na računu bogatega komitenta pokvaril le podatek o predznaku!

Pri tem pa se je nujno treba zavedati, da nobena izmed metod kodiranja ni popolna, kar pomeni, da z nobeno izmed njih ne odkrijemo oziroma popravimo vseh napak. Kanal, ki ga implementira protokol, ki napake odkriva in popravlja, ima torej še vedno neko pogostnost napak, ki jo imenujemo **preostala pogostnost napak** ali s tujko **rezidualna pogostnost napak**, ki pa mora biti seveda veliko manjša od pogostnosti napak v nižjem sloju, če sta bila način kanalskega kodiranja in protokol dovolj kvalitetno izbrana.

Med metodama popravljanja na sprejemni strani oziroma s ponovnim pošiljanjem obstaja nekaj pomembnih razlik, ki jih moramo vsekakor upoštevati, preden se odločamo za eno ali za drugo metodo. Najprej je treba poudariti, da tiste kode, ki omogočajo sprejemniku popravljanje k-kratnih napak, vsebujejo precej (nekajkrat!) večji delež redundantnih bitov, s tem pa tudi veliko večjo režijo, kot tiste, ki omogočajo le odkrivanje istega števila napak, seveda pri enako dolgih sporočilih. Seveda tudi ponovno pošiljanje pokvarjenih sporočil po metodi ARQ zahteva svoj delež kapacitete uporabljenega kanala; treba pa se je zavedati, da po metodi ARQ v večini primerov sporočila ponovno pošiljamo le tedaj, ko so se res pokvarila, medtem ko pri popravljanju na sprejemni strani opremimo z veliko režijo prav vsa sporočila, saj vnaprej seveda ne moremo vedeti, katera se bodo pokvarila. Iz tega sledi, da bo metoda popravljanja na sprejemni strani s stališča učinkovite izrabe kapacitete kanala ugodnejša le pri velikih pogostnostih napak; v praksi nastopajo tako velike pogostnosti napak le v radijskih, zlasti še mobilnih komunikacijah. Seveda pa protokoli ARQ zahtevajo vsaj polduplexsni, če ne že dupleksni kanal, saj mora sprejemni protokolni osebek sproti obveščati oddajni osebek o uspehu oziroma neuspehu posameznih protokolnih sporočil. Protokoli ARQ pa ne le zahtevajo vsaj polduplexsni kanal, ampak so tudi tem bolj učinkoviti, čim manjše so zakasnitve v tem kanalu. Pri velikih zakasnitvah v kanalu bo namreč oddajnik zelo pozno obveščen o morebitnih izgubah v kanalu in bo zato na te izgube tudi pozno reagiral. Podatke, ki jih na Zemljo pošiljajo razne vesoljske sonde, bomo torej zaščitili s kodami, ki omogočajo ne le odkrivanje, ampak tudi popravljanje napak na sprejemni strani. Če se neko sporočilo pri prenosu skozi kanal pokvari in ga mora oddajnik v skladu s protokolom ARQ ponovno poslati, bo prav to sporočilo pri prenosu seveda znatno bolj zakasnjeno kot večina drugih. To pa pomeni, da protokoli ARQ vnašajo v prenos sporočil neenakomerne zakasnitve, ki nas sicer pri prenosu podatkov ne motijo preveč, so pa zelo

nezaželene pri prenosu informacije, ki jo želimo prenašati v realnem času, to sta npr. govor in glasba (sinhroni promet). Pri prenosu informacije v realnem času torej ne bomo uporabljali protokolov s ponavljanjem. To je tudi eden izmed vzrokov, zakaj internetski protokol za prenos informacije v realnem času **RTP (Real-time Transport Protocol)** v transportnem sloju uporablja protokol UDP, ki pokvarjenih paketov sploh ne popravlja, ne pa protokola TCP (ki je protokol s ponavljanjem), tako kot večina internetskih aplikacijskih protokolov. Razlike med popravljanjem sporočil na sprejemni strani oziroma s ponavljanjem povzemamo v Tabeli 8-4.

Tabela 8-4. Primerjava popravljanja na sprejemni strani in s ponavljanjem

	popravljanje na sprejemni strani	*popravljanje s ponavljanjem*
režija	velika	majhna
kanal	simpleksni	(pol)dupleksni
zakasnitve	konstantne	spremenljive
sinhroni promet	primerno	neprimerno

Iz povedanega lahko sklepamo, da je izbiro kanalske kode in metode morebitnega popravljanja pokvarjenih sporočil treba skrbno pretehtati in se pri tem odločiti za kompromis med režijo in učinkovitostjo protokola, potrebo po popravljanju napak oziroma potrebo po prenosu v realnem času ter rezidualno pogostnostjo napak.

Seveda lahko tudi v samem protokolnem sporočilu ločimo med bolj in manj pomembno informacijo in glede na tako kategorizacijo različne dele protokolnega sporočila tudi različno zaščitimo. V splošnem lahko rečemo, da je najpomembnejši del protokolnega sporočila nadzorna protokolna informacija, saj prav ta nadzira pravilen potek izmenjave informacije; zato pri marsikaterem protokolu zaščitimo le glavo protokolnega sporočila ter le v njej na sprejemni strani odkrivamo ali celo tudi popravljamo napake*.

V nekaterih protokolih kombiniramo odkrivanje in popravljanje napak na ta način, da sprejemnik oceni (glede na Hammingovo razdaljo sprejete besede do najbližje veljavne kodne besede), ali je sposoben napako sam popraviti, ali pa bo kako drugače ukrepal.

Predvsem pa kombiniramo različne stopnje zaščite, odkrivanja in popravljanja napak v protokolnem skladu. Protokole, ki so sposobni odkrivati in popravljati napake,

uporabljamo predvsem v tistih slojih, kjer napake dovolj pogosto nastajajo. Pred desetimi in več leti je bila pogostnost napak v fizičnem sloju zelo velika, zato so napake popravljali predvsem v kanalnem sloju. Kasneje so začeli to klasično funkcijo kanalnega sloja nekoliko opuščati, saj je pogostnost napak v fizičnem sloju nekoliko upadla, v prostranih omrežjih zaradi izboljšanih tehnologij modulacije in prenosa prek optičnih vlaken, v lokalnih omrežjih pa zaradi razmeroma kratkih razdalj. Seveda pa moramo pri porazdelitvi funkcij po slojih protokolnega sklada upoštevati tudi dejstvo, da pogosto napake niso niti edini niti najpomembnejši vzrok za izgube v telekomunikacijskem sistemu, druge vrste izgub (npr. zaradi kolizij ali zgostitev prometa) pa lahko nastajajo tudi v višjih slojih.

* Podobno mora ob morebitni nesreči tudi pilot letala najprej poskrbeti zase, saj bi še tako zdravim potnikom v letalu brez pilota trda predla!

9 PROTOKOLI S PONAVLJANJEM

V prejšnjem poglavju smo že spoznali, da lahko napake, do katerih pride pri prenosu sporočil skozi kanal, popravljamo le na sprejemni strani ali s ponovnim pošiljanjem pokvarjenih sporočil; v slednjem primeru uporabljamo protokole s ponavljanjem (protokole ARQ). Z le-temi lahko seveda tudi nadomeščamo sporočila, ki so se izgubila. Pričujoče poglavje bo posvečeno prav protokolom s ponavljanjem. V naslednjem poglavju pa bomo spoznali, da popravljanje napak s ponavljanjem pogosto združimo s krmiljenjem pretoka v istih protokolih.

Protokolna sporočila se lahko v kanalu ne le pokvarijo, ampak tudi popolnoma izgubijo. Do izgube sporočila pride v kanalnem sloju po modelu OSI pogosto na ta način, da se pokvari ena izmed binarnih vrednosti, ki sprejemniku omogočajo sinhronizacijo (npr. v sinhronizacijski sekvenci), zaradi česar sprejemnik sploh ne zazna začetka sporočila, ki je zanj tako izgubljeno. V omrežnem sloju pa je pogost vzrok za izgubo sporočila zavrženje paketa zaradi polne čakalne vrste v eni izmed omrežnih naprav, včasih pa tudi napaka pri usmerjanju. Protokol s ponavljanjem bo torej moral poskrbeti, da bo vsa sporočila prenesel od enega uporabnika do drugega brez napak, brez izgub, prav tako pa tudi v pravem vrstnem redu (kar pomeni, da bo sprejemni uporabnik sporočila sprejel v istem vrstnem redu, kot jih je oddajni uporabnik oddal). Seveda pa mora uporabnik na sprejemni strani vsako sporočilo, ki mu ga je poslal njegov partner, sprejeti točno enkrat (torej ne sme kakšnega sporočila sprejeti v duplikatu). Prenosu sporočil z navedenimi lastnostmi pogosto rečemo na kratko **zanesljiv prenos sporočil**.

Zanesljiv prenos sporočil je načeloma mogoč le, če se sodelujoča protokolna osebka zavedata zaporedja sporočil, ki jih prenašata, in ne le vsakega sporočila posebej. To pa pomeni, da se mora proces zanesljivega prenosa sporočil vedno nahajati v nekem stanju, ki se lahko spremeni, ko en osebek neko sporočilo odda ali ga drugi sprejme. Vsak osebek torej v zvezi z zanesljivim prenosom sporočil uporablja spomin, vsebini spomina obeh osebkov pa morata biti med seboj do določene mere tudi usklajeni, za kar skrbita oba protokolna osebka z izmenjavo nadzorne protokolne informacije. Pred začetkom komuniciranja je treba spomina obeh osebkov najprej pridobiti oziroma rezervirati* ter

* Programska izvedba protokolnega osebka bo v ta namen pogosto od operacijskega sistema zahtevala dodelitev potrebne količine spomina. Večina sodobnih večopravilnih operacijskih sistemov namreč

njuni vsebini inicializirati (torej postaviti v neko definirano začetno stanje). To pa je mogoče le, če gre za povezavni način komuniciranja, kjer je komunikacijski proces vezan na določeno zvezo. Ob vzpostavitvi zveze protokolna osebka rezervirata spomin in ga inicializirata, med potekom zveze skrbita za to, da sta vsebini obeh spominov vsaj delno usklajeni (kolikor je to pač mogoče v sistemu, ki je v principu porazdeljen!), ob sprostitvi zveze pa lahko osebka sprostita za zvezo uporabljeni spomin, če jima je bil le-ta dinamično dodeljen. Iz vsega tega pa sledi, da datagramski protokoli načeloma ne morejo s ponavljanjem popravljati napak ali nadomeščati izgub oziroma urejati sporočil v pravem vrstnem redu, saj osebki za prenos datagramov ne uporabljajo spomina in se ukvarjajo le z vsakim datagramom posebej). Datagramski način prenosa torej pogosto imenujemo tudi **nezanesljiv prenos sporočil**, saj storitev, ki jo ti protokoli nudijo višjemu sloju, ne zagotavlja prenosa sporočil brez napak, brez izgub in v pravem vrstnem redu.

9.1 Elementi protokolov s ponavljanjem

Omenili smo že, da za protokol s ponavljanjem potrebujemo vsaj polduplekSni kanal, saj je potrebno prenašati informacijo (uporabniško oziroma protokolno nadzorno) v obe smeri. Najpreprostejši primer je seveda, ko eden izmed obeh protokolnih osebkov (imenovali ga bomo kar oddajnik O) pošilja drugemu (ki ga bomo imenovali sprejemnik S) informacijska protokolna sporočila, drugi (torej sprejemnik) pa mora prvega (oddajnik) le obveščati, katera informacijska protokolna sporočila je sprejel (to pomeni brez napake) in katerih ne. Sprejemnik lahko oddajniku poroča o odkritih napakah, ali pa vsa sporočila, v katerih je odkril napako, zavrže. V slednjih primerih torej ne razlikujemo med pokvarjenimi in izgubljenimi sporočili.

Vsi protokoli s ponavljanjem za logično pravilni potek zanesljive komunikacije zahtevajo naslednje osnovne mehanizme:

- oddajnik po vrsti pošilja oštevilčena in kanalsko kodirana informacijska protokolna sporočila
- sprejemnik pošilja oštevilčene potrditve tistih informacijskih protokolnih sporočil, ki jih je sprejel brez napake (pozitivne potrditve)

---

omogoča dinamično dodeljevanje spomina procesom, ki ga potrebujejo, in sicer tedaj, ko ga potrebujejo, in toliko, kolikor ga potrebujejo.

- oddajnik uporablja časovnik, ki se mu izteče, če predolgo časa ne prejme potrditve, nakar ponovno pošlje informacijsko protokolno sporočilo, za katerega meni, da se je pokvarilo ali izgubilo; če pa sprejme pričakovano potrditev, časovnik ustavi

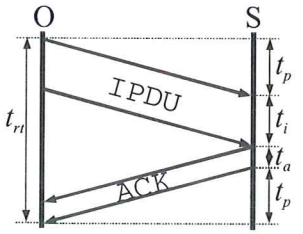
Pri tem je seveda pomemben čas izteka časovnika. Ta čas ne sme biti takšen, da bi se lahko oddajniku časovnik iztekel, preden bi prejel potrditev poslanega informacijskega sporočila, če se sporočilo ni niti pokvarilo, niti izgubilo. Čas izteka časovnika mora biti torej vsaj tolikšen, kot je pričakovani čas do potrditve

$$t_{to} > t_{rt}, \tag{56}$$

kjer je **čas do potrditve** (ang. **round trip time**) enak

$$t_{rt} = t_i + t_a + 2 \cdot t_p, \tag{57}$$

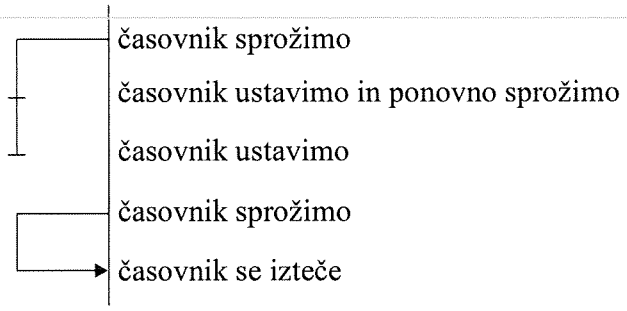
kot lahko razberemo iz slike Slika 9-1. Pri tem je t_i čas oddajanja (oziroma sprejemanja) informacijskega protokolnega sporočila, t_a čas oddajanja oziroma sprejemanja potrditve, t_p pa čas propagacije elektromagnetnega signala skozi kanal. Čas obdelave protokolnih sporočil v obeh protokolnih osebkih smo zanemarili, prav tako pa smo predpostavili, da je sprejemni protokolni osebek začel oddajati potrditev takoj, ko je sprejel informacijsko protokolno sporočilo. Enačbo (2) lahko uporabimo le, če uporabljamo fizičen kanal, v katerem so zakasnitve deterministične. V primeru navideznega kanala z nedeterminističnimi zakasnitvami pa moramo čas do potrditve oceniti ali meriti. V sliki Slika 9-1 smo informacijsko protokolno sporočilo označili s kratico `IPDU`, potrditev pa s kratico `ACK` (ang. acknowledge)*. Medtem ko informacijsko protokolno sporočilo vsebuje uporabniško sporočilo (storitveno podatkovno enoto SDU) in nadzorno protokolno informacijo (režijo), vsebuje potrditev le nadzorno protokolno informacijo, zato je lahko bistveno krajša od informacijskega protokolnega sporočila.



Slika 9-1. Čas do potrditve

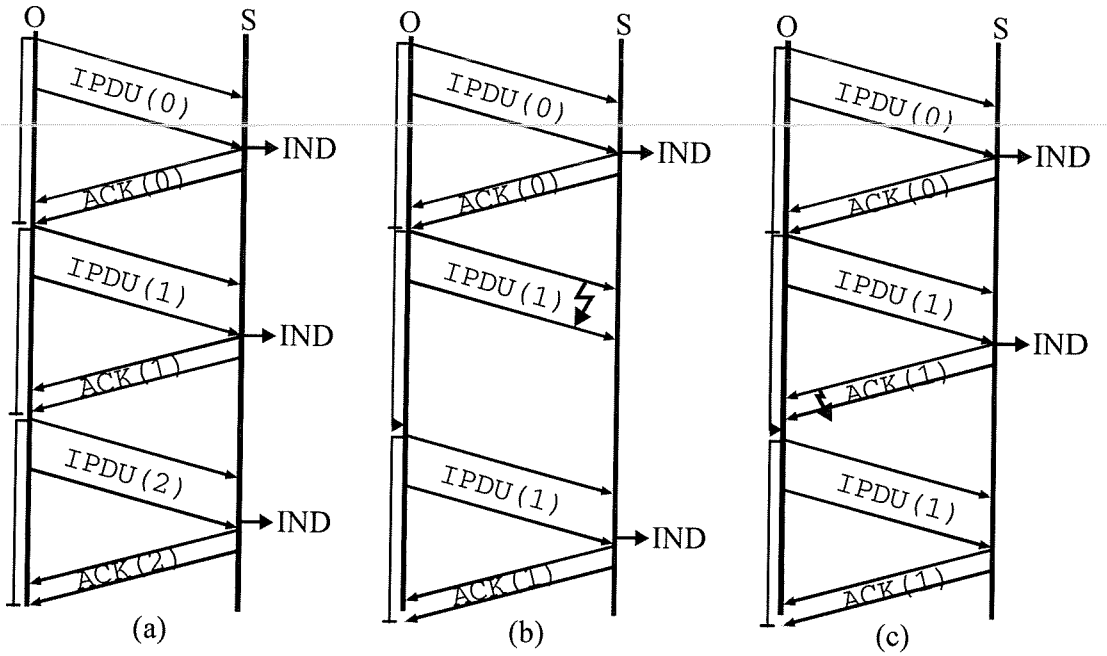
* ACK je tudi eden izmed krmilnih znakov ASCII, ki ima enak namen

V naslednjih časovnih shemah bomo krmiljenje in iztek časovnika prikazali tako kot na sliki Slika 9-2.



Slika 9-2. Prikaz krmiljenja časovnika v časovni shemi

V sliki Slika 9-3(a) vidimo primer poteka komunikacijskega procesa, ko se v kanalu ne pokvari ali izgubi nobeno sporočilo, v sliki Slika 9-3(b) primer, ko se je v kanalu pokvarilo informacijsko protokolno sporočilo, v sliki Slika 9-3(c) pa primer, ko se je v kanalu pokvarila potrditev. Na levi strani časovnih shem v sliki Slika 9-3 so označeni proženje, predčasna ustavitev in iztek časovnika. Na desni strani časovnih shem pa vidimo primitive `IND`, s katerimi sprejemnik preda svojemu uporabniku v sprejetih informacijskih protokolnih sporočilih vsebovana uporabniška sporočila. S primerjavo slik Slika 9-3(b) in Slika 9-3(c) lahko uvidimo, zakaj je potrebno številčenje protokolnih sporočil: sprejemniku omogoča razlikovanje med novo sprejetimi informacijskimi sporočili in duplikati. Morda se bo bralec začudil, zakaj je sprejemnik v sliki Slika 9-3(c) po sprejemu duplikatnega informacijskega sporočila ponovno poslal že oddano potrditev; oddajnik to potrditev seveda čaka, saj ne ve, da jo je sprejemnik že poslal in se je pokvarila ali izgubila.



Slika 9-3. Primer izmenjave informacijskih protokolnih sporočil in potrditev: (a) brez napake, (b) z napačno prenešenim informacijskim sporočilom, (c) z napačno prenešeno potrditvijo

Vsako informacijsko protokolno sporočilo torej v okviru svoje protokolne nadzorne informacije vsebuje sekvenčno številko. Prav tako pa sekvenčno številko vsebuje potrditev, saj s tem sprejemnik oddajniku sporoča, sprejem katerega informacijskega protokolnega sporočila potrjuje. Zato bomo za informacijska protokolna sporočila in potrditve uporabljali abstraktno sintakso `IPDU(`n`)` oziroma `ACK(`n`)`, kjer parameter n predstavlja sekvenčno številko, čeprav vemo, da informacijsko protokolno sporočilo nosi s seboj poleg sekvenčne številke tudi uporabniško sporočilo. Na podlagi sekvenčnih številk ne odkrivamo le duplikatov, ampak tudi izgubljena informacijska protokolna sporočila, prav tako pa tudi preverjamo vrstni red sprejetih informacijskih protokolnih sporočil.

Protokolna sporočila seveda nosijo sekvenčno številko s seboj v enem izmed polj svoje režije. Ker pa ima to polje končno dolžino m bitov, lahko v to polje zapišemo le končno število M različnih sekvenčnih številk, kjer je M enak

$$M = 2^m . \tag{58}$$

Zato moramo sporočila šteti po modulu M, kar pomeni, da sekvenčni številki M-1 sledi sekvenčna številka 0. M imenujemo **modul štetja**. Da bo protokol logično pravilen, mora biti modul štetja tako velik, da je v vsakem trenutku komunikacijskega procesa enoumno jasno, katero informacijsko protokolno sporočilo sekvenčna številka označuje. To pa pomeni, da mora biti število informacijskih protokolnih sporočil, ki so v katerem koli trenutku v celotnem komunikacijskem sistemu v uporabi, končno. O izbiri modula štetja bomo spregovorili v nadaljevanju tega poglavja.

Pogosto pa poleg osnovnih mehanizmov uporabljamo še nekatere dodatne, katerih namen je le izboljšati učinkovitost protokola. Tu bomo omenili dva izmed njih.

Dolžina informacijskega protokolnega sporočila se zaradi vsebovanega uporabniškega sporočila lahko od sporočila do sporočila naključno spreminja. Pri izračunu časa izteka časovnika moramo seveda upoštevati maksimalni čas oddajanja informacijskega protokolnega sporočila, ki pa je pogosto lahko precej večji od povprečnega ali celo minimalnega. Zato je s stališča učinkovitosti protokola smiselno, da sprejemnik v primeru, ko odkrije napako v sprejetem informacijskem protokolnem sporočilu, o tem z **negativno potrditvijo** čim prej obvesti oddajnik, ki na ta način lahko na napako prej reagira in sporočilo ponovno pošlje. Seveda lahko sprejemnik reagira neposredno na napačno sprejeto sporočilo le v primeru komunikacije točka-točka, ko torej pričakuje sporočila le od enega oddajnika in v okviru ene same zveze, saj sicer ne bi vedel, od koga je dobil napačno sporočilo (pokvaril se je lahko tudi naslov pošiljatelja ali oznaka zveze). Lahko pa sprejemnik ne reagira neposredno na pokvarjeno sporočilo, ampak pošlje oddajniku negativno potrditev (ki seveda pomeni eksplicitno zahtevo po ponovnem pošiljanju) šele, ko je po pokvarjenem sporočilu dobil sporočilo brez napake, ki pa ga še ne pričakuje. Tak postopek lahko uporabljamo tudi v primerih, ko ne gre za komunikacijo točka-točka, vendar pa oddajnik pred oddajo informacijskega protokolnega sporočila ne čaka na potrditev predhodnega sporočila. V tem primeru je reakcija sprejemnika na sprejem pokvarjenega sporočila popolnoma enaka kot reakcija na sprejem izgubljenega sporočila. Izboljšanje učinkovitosti protokola z uporabo negativnih potrditev bo tem večje, čim manjši bo povprečni čas do potrditve v primerjavi s časom izteka časovnika. Pri tovrstnih protokolih pričakujemo, da bodo v primeru zmernih pogostnosti napak negativne potrditve poskrbele za popravilo večine napak, časovnik pa se bo začel iztekati

šele pri večjih pogostnostih napak. Potrebno pa se je zavedati, da je za logično pravilnost protokola časovnik v vsakem primeru potreben, saj se lahko pri prenosu pokvari ne le informacijsko protokolno sporočilo, ampak tudi pozitivna oziroma negativna potrditev. Na tem mestu bi radi opozorili, da tudi uporaba samo negativnih potrditev in časovnika zaradi možnosti izgube informacijskih protokolnih sporočil ne zadostujeta.

V sliki Slika 9-3(c) smo videli primer, ko se je pokvarila le potrditev in je oddajnik po nepotrebnem ponovno poslal informacijsko protokolno sporočilo. Glede na to, da so informacijska protokolna sporočila lahko v nekaterih omrežjih tudi zelo dolga (običajno dosti daljša od nadzornih protokolnih sporočil, ki ne vsebujejo uporabniške informacije), to predstavlja precejšnjo potrato kapacitete kanala, kar neugodno vpliva na učinkovitost protokola. Zato lahko oddajnik po izteku časovnika odda posebno poizvedbeno nadzorno protokolno sporočilo, sprejemnik pa je po sprejemu take poizvedbe dolžan ponovno poslati nazadnje oddano potrditev, iz katere lahko oddajnik razbere, ali se je izgubilo informacijsko protokolno sporočilo ali potrditev.

V nadaljevanju tega besedila se bomo osredotočili na obravnavo protokolov, ki uporabljajo le tri osnovne elemente: oštevilčena informacijska sporočila, oštevilčene potrditve in časovnik.

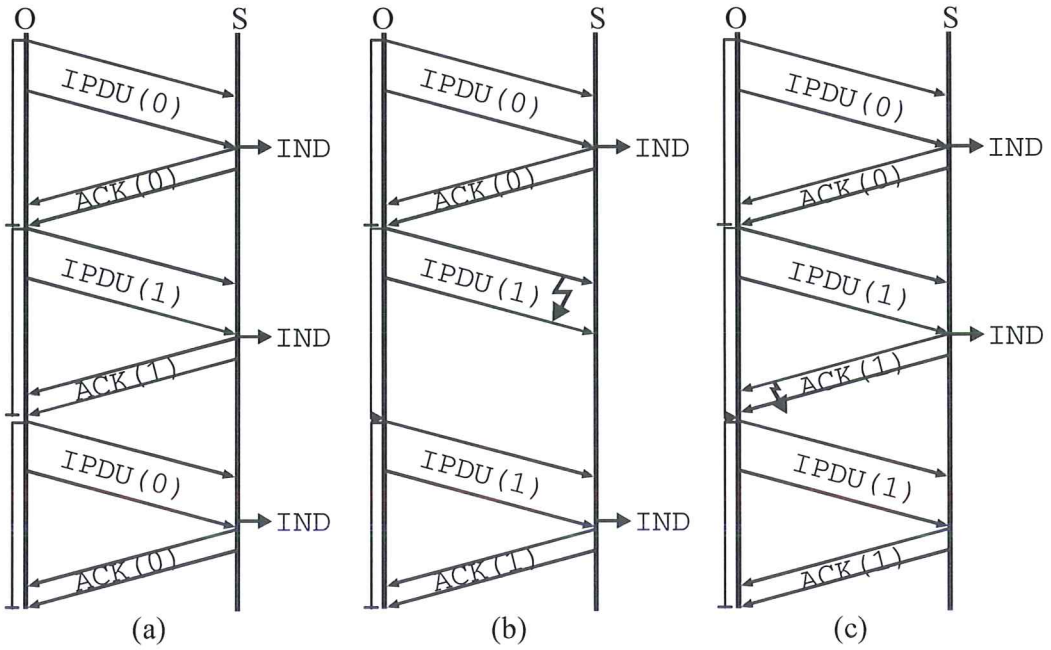
9.2 Protokol s čakanjem

9.2.1 Opis protokolov s čakanjem

Najpreprostejša različica protokola s ponavljanjem je **protokol s čakanjem** (ang. **stop-and-wait protocol**). Pri tovrstnem protokolu oddajnik odda informacijsko protokolno sporočilo in nato čaka na njegovo potrditev; šele, ko le-to sprejme, lahko odda naslednje informacijsko sporočilo. Izkaže se, da je za logično pravilnost protokola dovolj, če informacijska protokolna sporočila štejemo po modulu 2, oziroma, če oddajnik in sprejemnik razlikujeta le med informacijskimi protokolnimi sporočili s sodimi in tistimi z lihimi sekvenčnimi številkami. Sekvenčne številke lahko torej zapišemo z enim samim bitom ($2=2^1$), katerega vrednost pa od sporočila do sporočila alternira (0-1). Zato ta protokol pogosto imenujemo tudi **protokol z alternirajočim bitom** (ang. **Alternating Bit Protocol**). Iz angleškega imena izhaja tudi kratica **ABP**, ki jo bomo včasih uporabljali za označevanje tega protokola.

Protokol z alternirajočim bitom bomo natančneje opisali v najpreprostejši možni izvedbi, torej takega. ki uporablja le pozitivne potrditve in časovnik. Pri tem nas bo zanimal le prenos podatkov med uporabnikoma, ne pa tudi vzpostavljanje, vzdrževanje in sproščanje zveze.

V sliki Slika 9-4 podajamo tri zglede komunikacijskega procesa po protokolu ABP, in sicer (a) v primeru, ko pri prenosu ni prišlo do napak, (b) v primeru, ko se je pokvarilo informacijsko protokolno sporočilo, in (c) v primeru, ko se je pokvarila potrditev. V sliki lahko sicer poleg izmenjave protokolnih sporočil vidimo tudi uporabo primitivov `DataInd`, ne pa tudi primitivov `DataReq`, saj le ti prihajajo do čakalne vrste v točki dostopa do storitve na oddajni strani neodvisno od dogajanja v izvajalcu storitve. Smo pa pri risanju slike predpostavili, da ta čakalna vrsta ni nikoli prazna.



Slika 9-4. Zgled komunikacijskega procesa po protokolu ABP

9.2.2 Specifikacija protokola z alternirajočim bitom

Protokol z alternirajočim bitom lahko v besedilu opišemo z naslednjimi pravili za delovanje oddajnika oziroma sprejemnika.

Ko **oddajnik** od uporabnika sprejme primitiv `DataReq`, odpošlje informacijsko protokolno sporočilo `IPDU`, sproži časovnik in čaka na potrditev `ACK`. Če je nazadnje uspešno oddal `IPDU` s sekvenčno številko 0, bo sedaj oddal `IPDU` s sekvenčno številko

1, oziroma obratno. (Da je uspešno poslal IPDU, pomeni, da je tudi že prejel njegovo potrditev.) Prvi IPDU, ki ga pošlje po vzpostavitvi začetnega stanja (inicializaciji), ima sekvenčno številko 0. Če je oddal IPDU s številko 0, čaka na ACK s številko 0, če pa je oddal IPDU št. 1, čaka na ACK št. 1. Če čaka ACK s številko 0, pa sprejme ACK s številko 1, ga zavrže, in obratno. Če sprejme tisto potrditev, ki jo čaka, ustavi časovnik in je pripravljen na oddajo novega informacijskega protokolnega sporočila IPDU (čaka na primitiv DataReq). Če se časovnik izteče, ponovno odda isti IPDU kot zadnjikrat (torej tistega, katerega potrditev čaka) z isto številko in sproži časovnik. Če oddajnik sprejme sporočilo z napako, ga ignorira.

Sprejemnik čaka na sprejem informacijskega protokolnega sporočila IPDU. Po vzpostavitvi začetnega stanja (inicializaciji) čaka na IPDU s sekvenčno številko 0. Po uspešnem sprejemu IPDU-ja s številko 0 čaka na IPDU s številko 1, po uspešnem sprejemu IPDU-ja s številko 1 čaka na IPDU s številko 0. Uspešen sprejem IPDU-ja pomeni, da je sprejemnik brez napake sprejel tisti IPDU, ki ga pričakuje. Tudi tu moramo pripomniti, da bi vrednost spremenljivke V v realnem življenju inicializirali ob vzpostavitvi zveze. Po sprejemu IPDU-ja, ki ne vsebuje napake, sprejemnik pošlje oddajniku potrdilo ACK s številko tega IPDU-ja. Če sprejemnik brez napake sprejme tisti IPDU, ki ga pričakuje, odstrani režijo in s primitivom DataInd preda vsebovano uporabniško sporočilo uporabniku. IPDU, ki ga sprejme, ko ga ne pričakuje, ali ki vsebuje napako, sprejemnik zavrže.

Pri specifikaciji procesa sprejemnik smo videli, da le ta potrdi z nadzornim protokolnim sporočilom ACK sprejem vsakega informacijskega protokolnega sporočila IPDU, ki ne vsebuje napake, ne glede, ali sprejme sporočilo s pričakovano sekvenčno številko ali ne (to pa se tudi ujema s tekstovno specifikacijo delovanja sprejemnika). Marsikdo ob površnem razmisleku pomisli, da to ni potrebno, ampak da mora sprejemnik potrjevati sprejem le tistih informacijskih protokolnih sporočil, ki jih pričakuje. Da pa to ni dovolj, naj se bralec sam prepriča tako, da nekoliko spremeni časovno shemo na sliki 4 11(c). Hitro se bo lahko prepričal, da se bo oddajnik zapletel v neskončno zanko, v kateri se mu bo periodično iztekal časovnik, vendar ne bo mogel priti iz stanja, v katerem čaka

na potrditev, če sprejemnik ne bo poslal tudi potrditve nepričakovanega ponovnega sprejema informacijskega protokolnega sporočila, katerega potrditev se je izgubila.

9.2.3 Učinkovitost protokolov s čakanjem

Protokoli s čakanjem so izredno preprosti, vendar pa je njihova poglavitna slaba lastnost majhna izkoriščenost kanala in s tem tudi relativno glede na nazivno hitrost kanala nizka dejanska hitrost prenosa informacije, saj oddajnik medtem, ko čaka na potrditev poslanega sporočila, miruje, torej ne oddaja. Vprašajmo se torej, kakšna je učinkovitost protokola z alternirajočim bitom.

Izračun učinkovitosti protokola je v splošnem zahtevna naloga, zato si jo pogosto poskušamo poenostaviti. Najpriročnejša, pa tudi daleč najučinkovitejša poenostavitev je predpostavka, da v kanalu nimamo izgub. Taka predpostavka je po svoje smešna in nesmiselna. Če namreč v kanalu ni izgub, potem seveda tudi protokola s ponavljanjem ne potrebujemo. Vendar pa nam ta predpostavka omogoča na preprost način izračunati učinkovitost protokola, ki vsaj približno drži tudi pri majhnih pogostnostih napak v kanalu. V primeru, ko izgube v kanalu so, je nekatera sporočila treba prenesti skozi kanal več kot enkrat, k opravljenemu prometnemu pretoku pa prispevajo le enkrat. Zatorej je v takem primeru učinkovitost protokola manjša od vrednosti, ki jo dobimo, če izgub ne upoštevamo. Vendar pa se v tovrsten izračun ne bomo spuščali, čeprav bi bil preprost v primeru protokola ABP, pa precej težavnejši pri drugih protokolih s ponavljanjem, ki jih bomo še obravnavali. Vsekakor pa se učinkovitost pri majhnih pogostnostih napak ne bo bistveno razlikovala od vrednosti, ki jo dobimo brez upoštevanja izgub. Pomembneje bo, da bomo učinkovitost različnih protokolov s ponavljanjem obravnavali pod istimi pogoji. Če pa bomo potrebovali natančne rezultate za učinkovitost protokola v sistemu z izgubami, bomo do teh rezultatov raje prišli s simulacijo.

Oglejmo si torej učinkovitost protokola ABP, če v kanalu ni izgub. Učinkovitost protokola smo definirali kot delež časa, ko je kanal oziroma oddajnik oziroma sprejemnik koristno izkoriščen. Če izgub ni (torej se nobeno sporočilo v kanalu ne pokvari ali izgubi), se dogajanje periodično ponavlja s periodo, ki je enako času do potrditve t_{rt}. V tej periodi pa je dejansko izkoriščen le čas oddajanja informacijskega protokolnega sporočila t_i. Torej je učinkovitost protokola ABP, če v kanalu ni izgub, podana z enačbo

$$\eta = \frac{t_i}{t_{rt}} = \frac{t_i}{t_i + t_a + 2 \cdot t_p}, \tag{59}$$

kjer pomenijo, če še enkrat ponovimo, t_i čas oddajanja informacijskega protokolnega sporočila, t_a čas oddajanja potrditve in t_p čas širjenja (propagacije) elektromagnetnega signala skozi kanal. Izračunamo lahko tudi maksimalno hitrost prenosa informacije skozi kanal izvajalca storitve, ki je

$$r_{s_{\max}} = \eta \cdot R = \frac{t_i}{t_i + t_a + 2t_p} \cdot R, \tag{60}$$

kjer je R, kot že vemo, nazivna hitrost prenosa informacije skozi kanal izvajalca storitve. Seveda pa uporabnika zaradi režije ne vidita niti učinkovitosti protokola η niti dejanske hitrosti r_s. Če sta L_i in L_r dolžina informacijskega protokolnega sporočila oziroma dolžina režije, vidita uporabnika za faktor režije $(L_i - L_r)/L_i$ manjšo nazivno hitrost prenosa informacije skozi abstraktni kanal

$$R_u = k_r \cdot r_{s_{\max}} = \frac{L_i - L_r}{L_i} \cdot \frac{t_i}{t_i + t_a + 2t_p} \cdot R \tag{61}$$

oziroma izkoristek η_u

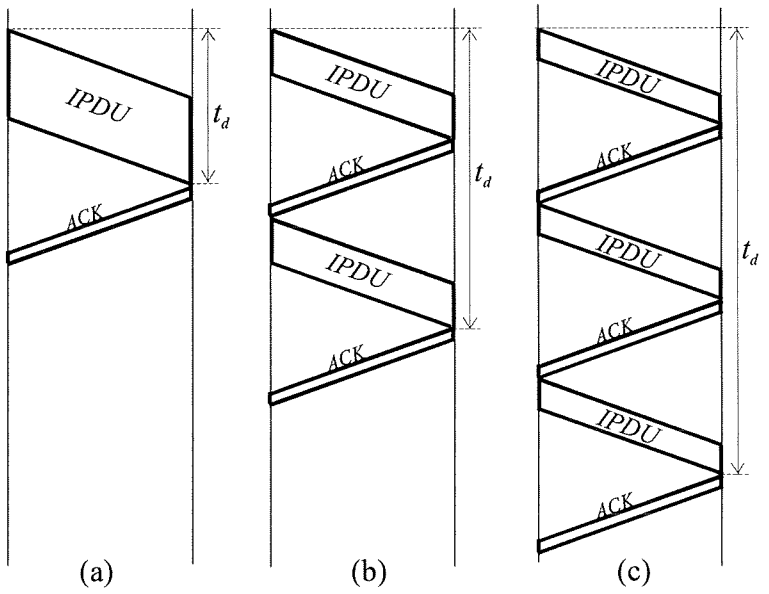
$$\eta_u = k_r \cdot \eta = \frac{L_i - L_r}{L_i} \cdot \frac{t_i}{t_i + t_a + 2t_p}. \tag{62}$$

Ker je nadzorno protokolno sporočilo pogosto precej krajše od informacijskega, imata glavno vlogo v tej formuli čas oddajanja informacijskega protokolnega sporočila $t_i = L_i/R$ in čas prenosa signala skozi kanal t_p. Če gre za fizični kanal, je slednji enak $t_p = D/v$, kjer je D fizična razdalja med osebkoma, v pa hitrost razširjanja. Če bodo sporočila dolga, hitrost prenosa informacije nizka, razdalje pa kratke, bo izkoristek blizu 1, v nasprotnem primeru pa bo lahko tudi zelo majhen. Ilustrirajmo to z dvema številčnima zgledoma, ki sta prikazana v tabeli 9-1. Pri tem smo upoštevali, da je dolžina potrditvenega protokolnega sporočila enaka kar dolžini režije, kar je tudi v praksi pogosto res. Vidimo pa tudi, da v drugem primeru, ko je delež režije v informacijskem protokolnem sporočilu večji, uporabnik vidi še manjši izkoristek oziroma še manjšo nazivno hitrost, ki je v tem primeru manj kot polovica nazivne hitrosti kanala.

Tabela 9-1. Dva zgleda učinkovitosti protokola ABP

R	L_i	L_r	D	v	η	η_u	R
64 kbit/s	1000 okt.	3 okt.	10 km	c	99,6 %	99,3 %	63.6 kbit/s
64 kbit/s	50 okt.	3 okt.	1000 km	c	47,0 %	44,2 %	28.3 kbit/s

Zakasnitev enega samega sporočila ni zaradi čakanja oddajnika na potrditev sprejema prav nič daljša, kot če bi oddajnik oddajal ves čas, saj sprejemnik takoj po uspešnem sprejemu informacijskega protokolnega sporočila uporabniško sporočilo preda uporabniku. Se pa pogosto zgodi, da neko daljše sporočilo razstavimo v več paketov, ki jih zaporedoma pošiljamo po kanalu. V takem primeru je zakasnitev celotnega sporočila precej daljša kot bi bila, če bi oddajnik oddal celotno sporočilo v enem kosu. Na sliki Slika 9-5 je ilustrirana zakasnitev sporočila, če ga prenesemo prek kanala po protokolu s čakanjem (a) v celoti, (b) v dveh delih oziroma (c) v treh delih. Pri tem smo vpliv režije na zakasnitev uporabniškega sporočila zanemarili.



Slika 9-5. Zakasnitev pri prenosu sporočila v celoti oziroma v dveh ali treh delih

Vzemimo, da imamo informacijsko protokolno sporočilo z dolžino L_i oziroma s časom oddajanja $t_i = L_i / R$. Kot že vemo, bo zakasnitev takega sporočila

$$t_{d_1} = t_i + t_p \ . \tag{63}$$

Če pa celotno sporočilo razdelimo na N enako dolgih paketov, bo zakasnitev celotnega sporočila (pri tem si lahko pomagamo s sliko Slika 9-5) enaka

$$t_{dN} = t_i + (N-1)\cdot t_a + (2N-1)\cdot t_p \,, \tag{64}$$

kar pomeni povečanje zakasnitve za

$$\Delta t_d = (N-1)\cdot t_a + (2N-2)\cdot t_p \,. \tag{65}$$

Povečanje zakasnitve je torej večje od nič, če je število paketov večje od ena. Povečanje zakasnitve je tem večje, čim večje je število paketov *N*. Seveda pa ta izračun drži le v primeru, ko je dolžina režije zanemarljiva proti dolžini uporabniškega sporočila. Če namreč celotno sporočilo prenašamo v več paketih, bo največkrat vsak izmed njih nosil s seboj enako režijo, kot paket, v katerem bi prenašali celotno sporočilo, kar pa celotno zakasnitev le še poslabša.

Tudi iz enačbe (4) sledi, da se bodo protokoli s čakanjem še zlasti slabo odrezali v tistih primerih, ko bomo prenašali razmeroma kratka sporočila preko dolgih kanalov.

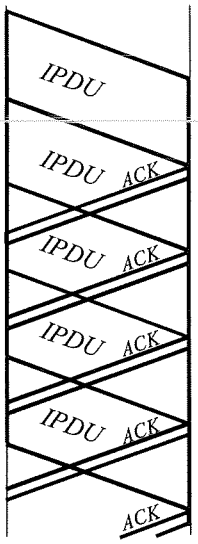
9.3 Kontinuirni protokol s ponavljanjem - protokol z drsečim oknom

Kot smo že ugotovili, je poglavitna hiba protokolov s čakanjem ta, da oddajnik med čakanjem na potrditev ne oddaja in tako ne izkorišča dane kapacitete kanala. Če je kanal multipleksen, kar pomeni, da ga sočasno* uporablja več oddajnikov, njegova izkoriščenost s strani enega oddajnika sicer ni nujno problematična, vendar pa vsak par protokolnih osebkov zato vidi dejansko hitrost prenosa informacije, ki je relativno (glede na nazivno hitrost) majhna. Poleg tega pa je ob uporabi protokolov s čakanjem lahko, kot smo ugotovili v prejšnjem razdelku, tudi skupna zakasnitev zaporedja sporočil precej večja. Zato je ugodno, če lahko oddajnik pošilja protokolna sporočila neposredno eno za drugim, ne da bi pri tem čakal na potrditev vsakega posebej. Protokole, ki to omogočajo, imenujemo **kontinuirni protokoli s ponavljanjem** (ang. **continuous ARQ protocols**).

9.3.1 Opis kontinuirnih protokolov s ponavljanjem

Pri kontinuirnem protokolu s ponavljanjem torej oddajni protokolni osebek pošilja informacijska protokolna sporočila neposredno eno za drugim, sočasno pa sprejemni protokolni osebek pošilja potrditve teh sporočil, kot to vidimo na sliki Slika 9-6. Povsem jasno je, da za tak način komunikacije ne zadostuje več izmenično enosmerni, ampak potrebujemo dvosmerni kanal.

* Izraz "sočasno" je treba tu razumeti v smislu paketnega multipleksa



Slika 9-6. Kontinuirni protokol s ponavljanjem

Kot smo že spoznali pri protokolu s čakanjem, morata tudi pri kontinuirnem protokolu oddajni in sprejemni protokolni osebek številčiti informacijska protokolna sporočila in potrditve. Ločimo lahko tri vrste potrditev: kumulativne, individualne in intervalne.

Kumulativna potrditev (ang **cumulative acknowledgment**) potrjuje pravilen sprejem informacijskih protokolnih sporočil od prvega poslanega (v okviru zveze) do vključno tistega, ki je v potrditvi specificirano. Tako lahko s potrditvijo `ACK(`n`)` potrdimo sprejem vseh informacijskih protokolnih sporočil od tistega z začetno sekvenčno številko pa do vključno tistega s sekvenčno številko n ali n-1, pač glede na dogovor oziroma protokol.

Individualna potrditev (ang **individual acknowledgment**) potrjuje pravilen sprejem enega samega, v potrditvi specificiranega informacijskega protokolnega sporočila. Uporaba takih potrditev je učinkovitejša v primeru, ko se lahko v kanalu pomeša vrstni red prenešenih sporočil, vendar pa prometno precej obremenjuje kanal, saj je treba vsako sprejeto informacijsko protokolno sporočilo posebej potrditi.

Intervalna potrditev (ang **block acknowledgment**) potrjuje pravilen sprejem enega ali več zaporednih informacijskih protokolnih sporočil. Intervalna potrditev seveda vsebuje dve sekvenčni številki. Tako lahko s potrditvijo `ACK(`m`,`n`)` potrdimo pravilen sprejem vseh sporočil s sekvenčnimi številkami med m in n. Intervalne potrditve

predstavljajo kompromis med preprostostjo kumulativnih in učinkovitostjo individualnih potrditev.

V nadaljevanju tega poglavja se bomo osredotočili na protokole, ki uporabljajo kumulativne potrditve, saj se ta vrsta potrditev tudi v praksi največ uporablja.

Že pri protokolu s čakanjem smo spoznali, da mora oddajni protokolni osebek oddano informacijsko protokolno sporočilo (ali pa vsaj v njem vsebovano uporabniško sporočilo) shraniti v svojem spominu (oddajnem vmesniku) toliko časa, dokler od sprejemnega protokolnega osebka ne prejme potrditve tega sporočila; v primeru, da se sporočilo med prenosom izgubi, ga bo namreč moral ponovno poslati. Ker pa bo oddajnik pri kontinuirnem protokolu praviloma oddal ali vsaj začel oddajati več kot le eno informacijsko protokolno sporočilo, preden bo sprejel kakršno koli potrditev (glej sliko Slika 9-6), bo v ta namen potreboval tudi spomin (oddajni vmesnik) za več informacijskih protokolnih (ali vsaj uporabniških) sporočil, ki jih je že oddal, a njihovih potrditev še ni prejel.

Ker je torej za shranjevanje že oddanih, a še ne potrjenih informacijskih protokolnih sporočil potreben spomin, le-tega pa imamo vedno končno mnogo na razpolago, oddajni protokolni osebek vedno razpolaga z oddajnim vmesnikom za končno število sporočil; to število imenujemo **širina oddajnega okna** (ang. **transmit window width**) in označimo s simbolom W_s. Širina oddajnega okna torej omejuje število že oddanih, a še ne potrjenih zaporednih informacijskih protokolnih sporočil. Za vzdrževanje oddajnega okna oziroma pripadajočega oddajnega vmesnika potrebuje oddajni protokolni osebek količino spomina

$$M_s = W_s \cdot L_i \ , \tag{66}$$

kjer je L_i, tako kot doslej, dolžina informacijskega protokolnega sporočila. Kot bomo videli v nadaljevanju tega poglavja, je s širino okna tesno povezan tudi potreben modul štetja informacijskih protokolnih sporočil.

Sprejemni protokolni osebek torej pošilja potrditve sprejetih informacijskih protokolnih sporočil. Odločili smo se že, da bomo obravnavali le varianto protokola, ki uporablja kumulativne potrditve. Naj vsaka potrditev vsebuje sekvenčno številko, ki pove, sprejem katerega informacijskega protokolnega sporočila sprejemnik pričakuje; s tem pa sprejemnik implicitno potrdi pravilen sprejem vseh prejšnjih sporočil. Potrditev $\mathrm{ACK}(a)$ torej potrjuje sprejem informacijskih protokolnih sporočil s sekvenčnimi

številkami *a*-1, *a*-2, *a*-3,... Kumulativne potrditve imajo več koristnih posledic. Ena izmed njih je, da sprejemnemu protokolnemu osebku ni treba potrjevati vsakega informacijskega protokolnega sporočila posebej, ampak lahko potrditev pošilja le občasno, vse te potrditve pa veljajo tudi za nazaj. Potrditev, ki je sprejemnik ne pošlje takoj po sprejemu potrjevanega informacijskega protokolnega sporočila, ampak šele kasneje, imenujemo **zakasnjena potrditev** (ang. **delayed acknowledgment**). Seveda pa mora sprejemnik oddati tako zakasnjeno potrditev najkasneje po nekem predpisanem času t_{da}, da se oddajniku medtem ne izteče časovnik. Čas t_{da} moramo upoštevati pri računanju časa do potrditve oziroma pri določanju časa izteka časovnika:

$$t_{to} > t_{rt} = t_i + t_a + t_{da} + 2 \cdot t_p . \tag{67}$$

V primeru zakasnjenih potrditev tudi sprejemnik uporablja časovnik za merjenje časa t_{da}.

Druga prednost kumulativnih potrditev se pokaže v primeru, če se ena izmed potrditev izgubi. Ko oddajnik sprejme naslednjo potrditev, ta velja tudi za nazaj, zaradi česar se oddajniku časovnik ne bo iztekel (če se že ni).

Prava vrednost kumulativnih potrditev pa se pokaže šele na strani oddajnika. To omogoča oddajniku, da uporablja le en časovnik kljub temu, da ima lahko hkrati več že oddanih, a še ne potrjenih informacijskih protokolnih sporočil. Sproži ga vedno, kadar odda novo informacijsko protokolno sporočilo, če tedaj časovnik ni aktiven (ne teče). Ustavi ga vedno, kadar sprejme neko novo potrditev. Če ima pri tem v svojem oddajnem vmesniku informacijska protokolna sporočila, ki jih je sicer že oddal, a njihove potrditve še ni prejel, časovnik takoj spet sproži. Časovnik mora torej teči vselej, kadar ima oddajnik v svojem vmesniku kakšno že oddano, a še ne potrjeno spročilo. Pri tem mora biti čas izteka časovnika tak, kot smo ga zahtevali v neenačbi 12. Upoštevati moramo seveda največji možni čas oddajanja informacijskega protokolnega sporočila t_i in, če je čas zakasnitve v kanalu t_p nedeterminističen, tudi največjo možno vrednost le-tega*.

Predvsem pa nam kumulativne potrditve pomembno poenostavijo upravljanje oddajnega okna in njemu pripadajočega oddajnega vmesnika. **Oddajno okno** (ang. **transmit window**) bomo definirali kot nabor W_s zaporednih sekvenčnih številk, ki jih sme oddajnik oddati, ne da bi prejel njihove potrditve. Najnižja sekvenčna številka v oknu je sekvenčna številka najstarejšega še nepotrjenega informacijskega protokolnega

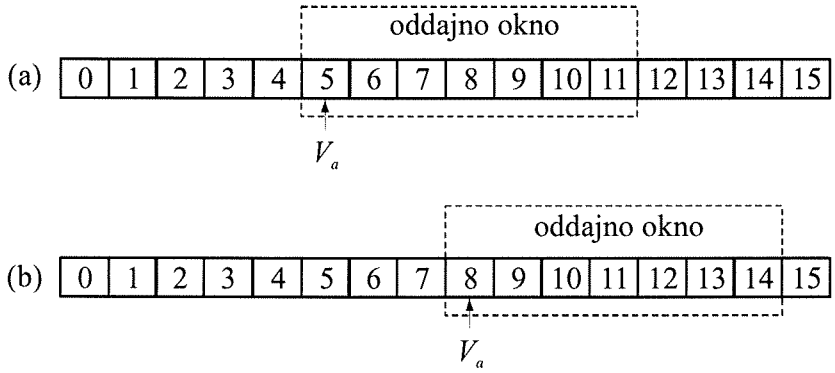
sporočila. Vzemimo, da oddajni osebek vrednost te številke hrani v spremenljivki V_a. Torej so v oddajnem oknu vsak trenutek sekvenčne številke $\{V_a, V_a+1, V_a+2, ..., V_a+W_s-1\}$, skupaj W_s zaporednih sekvenčnih številk. Ko oddajnik sprejme potrditev s sekvenčno številko a,

$$a>V_a, \tag{68}$$

je s tem dejansko na novo prejel potrditev informacijskih protokolnih sporočil s sekvenčnimi številkami V_a, V_a+1, ..., $a-1$. Zato lahko informacijska protokolna sporočila s sekvenčnimi številkami s, $V_a \le s \le a-1$, izbriše iz oddajnega vmesnika in premakne oddajno okno tako, da se le-to spet začne z najmanjšo sekvenčno številko, katere potrditev pričakuje. Torej oddajnik ob sprejemu potrditve s sekvenčno številko a, $a>V_a$, izvede naslednjo operacijo:

$$V_a := a \ , \tag{69}$$

pri čemer je ":=" operator prireditve vrednosti na desni strani spremenljivki na levi strani operatorja. Tako oddajni protokolni osebek premakne svoje okno naprej vsakič, kadar prejme neko novo potrditev. Okno torej s sprejemanjem novih potrditev drsi po številski premici. Zato tovrstne protokole imenujemo **protokoli z drsečim oknom** (ang. **sliding window protocols**). Primer drsečega oddajnega okna s širino W_s=7 pred (a) in po (b) sprejemu potrditve s sekvenčno številko a=8 vidimo na sliki Slika 9-7.



Slika 9-7. Položaj drsečega oddajnega okna pred (a) oziroma po (b) prejemu potrditve s številko a=8

Vzemimo sedaj, da ima oddajni protokolni osebek v spremenljivki V_s shranjeno sekvenčno številko informacijskega protokolnega sporočila, ki ga želi naslednjega oddati.

* Če je čas propagacije t_p nedeterminističen, moramo seveda njegovo največjo vrednost nekako oceniti.

Če oddajnik po vzpostavitvi zveze najprej odda informacijsko protokolno sporočilo s sekvenčno številko 0, je to tudi začetna vrednost spremenljivke V_s, prav tako pa tudi začetna vrednost spremenljivke V_a. Vrednost spremenljivke V_s seveda tudi pove, da je oddajnik zaporedje informacijskih protokolnih sporočil s sekvenčnimi številkami s, $s<V_s$, že oddal (v primeru, da je V_s=0, seveda še ni ničesar oddal!). Ker pa oddajnik ne bo pričakoval sprejema potrditve informacijskega protokolnega sporočila, ki ga še ni oddal, bomo pogoj (13) raje zapisali nekoliko natančneje*:

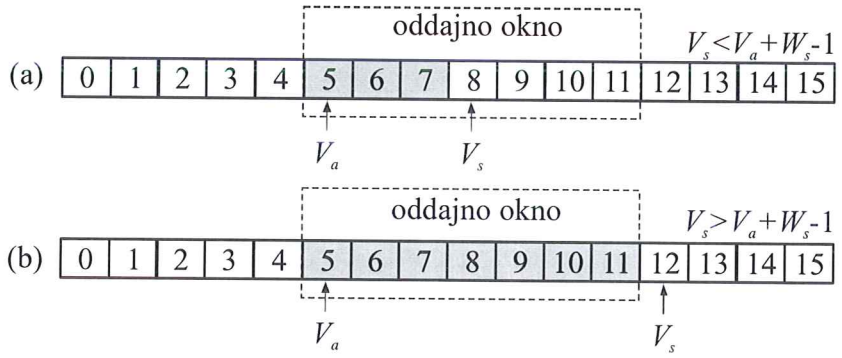
$$V_a < a \leq V_s \ . \tag{70}$$

Ker sme oddajnik imeti v oddajnem vmesniku največ W_s že oddanih, a še ne potrjenih informacijskih protokolnih sporočil, sme oddati sporočilo s številko V_s le, če je izpolnjen pogoj

$$V_s \leq V_a + W_s - 1 \ . \tag{71}$$

Če ta pogoj ni izpolnjen, to pomeni, da sekvenčna številka, shranjena v V_s, ni več v oknu, oziroma, da ima oddajnik že W_s oddanih, a ne potrjenih informacijskih protokolnih sporočil. Pravimo, da je okno polno. Na sliki Slika 9-8 vidimo primer okna, ki še ni polno (a), in okna, ki je že polno (b). Širina okna je v obeh primerih enaka 7. Vrednost spremenljivke V_s seveda ne more biti manjša od vrednosti spremenljivke V_a, zato lahko pogoj (16) natančneje zapišemo tudi kot

$$V_a \leq V_s \leq V_a + W_s - 1 \ . \tag{72}$$

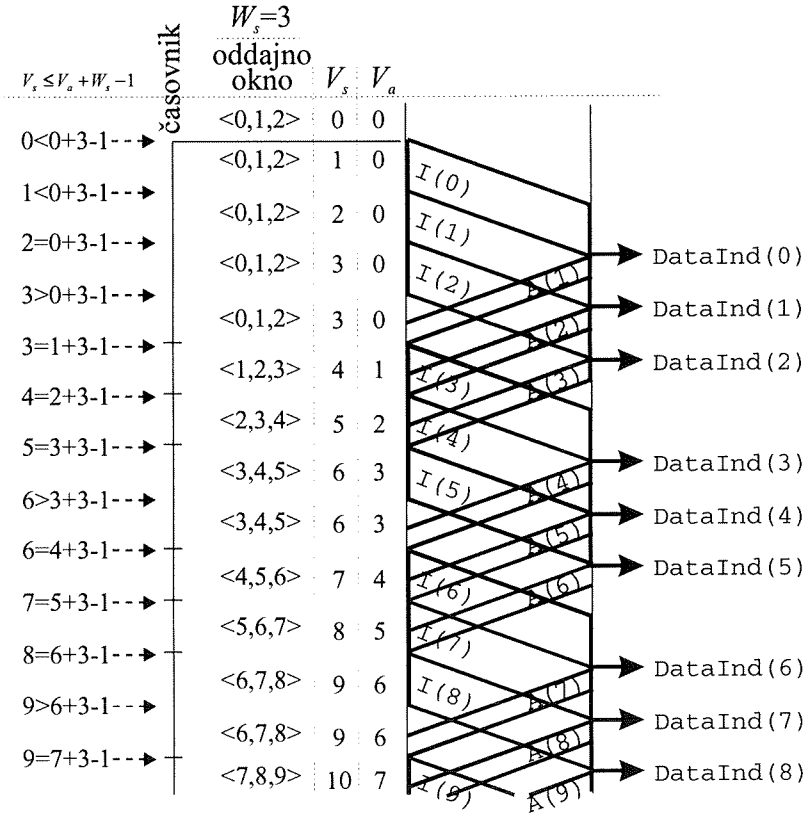


Slika 9-8. Nepolno (a) oziroma polno (b) oddajno okno

Slika Slika 9-9 nam prikazuje preprost primer komunikacijskega procesa, ko pri prenosu protokolnih sporočil ne prihaja do izgub, sprejemnik pa ne zadržuje potrditev.

* To nam bo kasneje še prav prišlo!

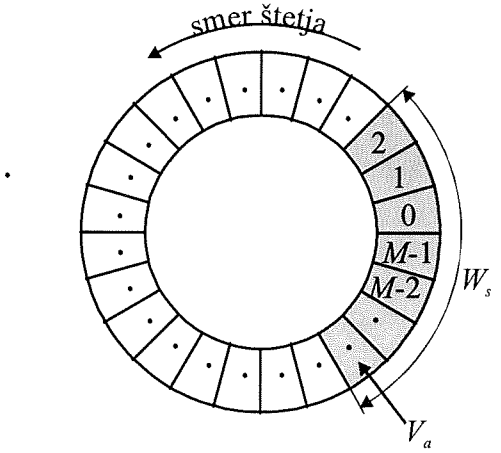
Širina oddajnega okna v tej sliki je 3, vsebino oddajnega okna (zaporedje sekvenčnih številk) pa vidimo na levi strani časovne sheme v formatu <*xyz*>. Označena je tudi vrednost spremenljivke V_s, ki se spremeni vsakič takoj po začetku oddajanja informacijskega protokolnega sporočila, in vrednost spremenljivke V_a, ki se spremeni vsakič po sprejemu nove potrditve glede na pogoj 13. Pred začetkom oddajanja vsakega informacijskega protokolnega sporočila je izpisan pogoj za začetek oddajanja po neenačbi 17. V primeru, ko oddajnik na videz ob istem času sprejme potrditev in odda informacijsko protokolno sporočilo, smo pri risanju slike predpostavili naslednji vrstni red dogodkov: sprejem potrditve, osvežitev vrednosti spremenljivke V_a po enačbi 14, preverjanje pogoja za oddajanje po neenačbi 17 in oddajo informacijskega protokolnega sporočila. Za informacijska protokolna sporočila uporabljamo abstraktno sintakso `I(`s`)`, za potrditve pa abstraktno sintakso `A(`a`)`, kjer parametra s in a predstavljata sekvenčni številki informacijskega protokolnega sporočila oziroma potrditve. Na sliki Slika 9-9 prav tako vidimo delovanje časovnika.



Slika 9-9. Primer komunikacijskega procesa brez izgub v kanalu

9.3.2 Štetje sporočil in indeksiranje oddajnega vmesnika

Informacijska protokolna sporočila štejemo in jih tudi indeksiramo v oddajnem vmesniku s celimi števili tipa `Integer`. Velikost oddajnega vmesnika, pa tudi število bitov, potrebnih za zapis sekvenčne številke, bi zato s časom naraščala proti neskončnosti, kar pa v praksi seveda ni dopustno. Ugotovili smo že, da je velikost vmesnika omejena s širino oddajnega okna W_s, kar pomeni, da so vrednosti indeksov v oddajnem vmesniku shranjenih sporočil dejansko le med 0 in W_s-1. Zato lahko sporočila štejemo po modulu M, kar pomeni, da imajo sekvenčne številke sporočil vrednosti le med 0 in M-1. Princip štetja po modulu M vidimo v sliki Slika 9-10. V sliki je tudi označen primer oddajnega okna. Iz te slike nam je brž jasno, da širina oddajnega okna ne sme biti večja od modula štetja. Je pa to le potreben, ne pa tudi zadosten pogoj za logično pravilnost protokolov. O tem, kakšna mora biti zveza med modulom štetja M in širino oddajnega okna W_s, pa več kasneje.



Slika 9-10. Princip štetja po modulu M in primer oddajnega okna

Če sekvenčne številke štejemo po modulu M, moramo po tem modulu tudi izvajati vse matematične operacije, ki se tičejo sekvenčnih številk. Tako je npr. (M-1)+1=0 ali $a+M=a$, relacijo $a \leq x \leq b$ pa preberemo "x je vključno med a in b" v smislu slike Slika 9-10.

Medtem ko torej sekvenčne številke štejemo po modulu M, moramo indekse sporočil v oddajnem vmesniku računati po modulu W_s. Preslikavo sekvenčnih številk informacijskih protokolnih sporočil v ustrezne indekse v oddajnem vmesniku bomo šteli za implementacijsko podrobnost protokola z drsečim oknom in je tu ne bomo podrobneje

opisali. Zaradi preprostejše obravnave bomo še naprej uporabljali tako štetje in indeksiranje kot v sliki Slika 9-9, torej s števili tipa integer.

9.3.3 Učinkovitost protokola z drsečim oknom pri kanalu brez izgub

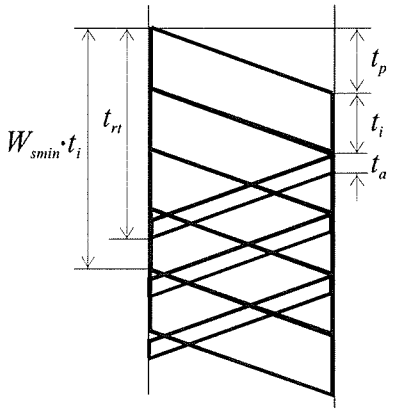
Preden se lotimo delovanja protokola z drsečim oknom v primeru, ko se eno ali več protokolnih sporočil pokvari ali izgubi, pa si oglejmo še učinkovitost protokola v primeru, da v kanalu ni izgub. K obravnavi kontinuirnih protokolov nas je namreč vzpodbudila prav neučinkovitost protokola s čakanjem, ki smo jo tudi študirali za primer, ko ni izgub. Poglejmo torej, ali smo z uvedbo kontinuirnih protokolov kaj pridobili.

S stališča učinkovitosti protokola je koristno, če je širina oddajnega okna tolikšna, da lahko oddajnik oddaja ves čas, dokler ne pride nova potrditev in zato premakne drseče okno; z drugimi besedami, pri normalnem delovanju brez napak naj se oddajno okno nikoli ne bi zapolnilo, oddajnik pa bi lahko ves čas oddajal, če bi le dobival dovolj uporabniških sporočil od uporabnika (če čakalna vrsta med uporabnikom in oddajnim protokolnim osebkom ne bi bila nikoli prazna). Seveda bi prav v tem primeru dosegli največji napredek z ozirom na protokol s čakanjem. Glede na definicijo učinkovitosti protokola bi bila potem učinkovitost protokola enaka 1. Kot lahko vidimo na sliki Slika 9-11, je pogoj za to

$$W_s \cdot t_i \geq t_{rt} = t_i + t_a + 2 \cdot t_p \tag{73}$$

oziroma

$$W_s \geq \frac{t_{rt}}{t_i} = \frac{t_i + t_a + 2 \cdot t_p}{t_i} \quad . \tag{74}$$



Slika 9-11.Minimalna širina oddajnega okna

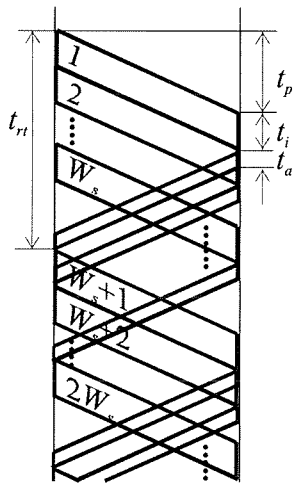
Širina oddajnega okna W_s mora biti seveda naravno število, zato moramo rezultat po neenačbi (19) zaokrožiti navzgor. Izpolnitev pogoja (19) je lahko problematična pri dolgih prenosnih poteh, kjer je zakasnitev signala t_p lahko precejšnja, in velikih hitrostih prenosa informacije, kjer je čas oddajanja kratek. V takih primerih bi za izpolnitev gornjega pogoja včasih potrebovali zelo veliko širino oddajnega okna, kar pa tudi pomeni velik oddajni vmesnik oziroma veliko količino spomina za hranjenje že oddanih, a še ne potrjenih sporočil. Ta količina spomina je namreč

$$M_s = W_s \cdot L_i \geq R \cdot \left(t_i + t_a + 2 \cdot t_p + t_{proc}\right). \tag{75}$$

Če je pogoj (19) izpolnjen, je torej učinkovitost protokola pri predpostavki, da napak ni, enaka 1. Če ta pogoj ni izpolnjen, pa je učinkovitost po sliki Slika 9-12enaka

$$\eta = \frac{W_s \cdot t_i}{t_i + t_a + 2 \cdot t_p + t_{proc}} \quad . \tag{76}$$

V sliki Slika 9-12, pa tudi v primeru na sliki Slika 9-9, se namreč lahko prepričamo, da lahko oddajni osebek v času do potrditve t_{rt} odda W_s informacijskih protokolnih sporočil, kar traja čas $W_s \cdot t_i$.



Slika 9-12. K izračunu učinkovitosti protokola z drsečim oknom

Spoznali smo že, da je komunikacijski proces stohastičen. Zaradi tega moramo v vseh enačbah za učinkovitost upoštevati povprečne vrednosti naključnih veličin.

Enačbo za učinkovitost protokola z drsečim oknom v primeru, ko v kanalu ni izgub, lahko sedaj v posplošeni obliki takole zapišemo:

$$\left\{ \begin{array}{l} W_s < \dfrac{t_i + t_a + 2 \cdot t_p}{t_i} \rightarrow \eta = \dfrac{W_s \cdot t_i}{t_i + t_a + 2 \cdot t_p + t_{proc}} \\ W_s \geq \dfrac{t_i + t_a + 2 \cdot t_p}{t_i} \rightarrow \eta = 1 \end{array} \right\}. \quad (77)$$

9.4 Protokol z drsečim oknom, delovanje v primeru izgub

Doslej še nismo nič povedali o tem, kako pri protokolu z drsečim oknom oddajni protokolni osebek reagira na iztek časovnika oziroma kako deluje sprejemni osebek. Glede na to, ali mora oddajni protokolni osebek ponovno poslati le tista sporočila, ki so se pri prenosu pokvarila ali izgubila, ali pa tudi vsa tista, ki so sicer brez napake prispela na cilj, a se je pred njimi vsaj eno sporočilo izgubilo, ločimo dve različici protokola z drsečim oknom: to sta **protokol s selektivnim ponavljanjem** (ang. **Selective Repeat Protocol**) in **protokol s ponavljanjem N sporočil** (ang. **Go-Back-N Protocol**). Najprej si bomo ogledali preprostejše protokole s ponavljanjem N sporočil.

9.4.1 Protokoli s ponavljanjem N sporočil

9.4.1.1 Opis in specifikacija protokola s ponavljanjem N sporočil

Pri teh protokolih morajo protokolna sporočila prihajati do sprejemnega protokolnega osebka v pravilnem vrstnem redu, kar pomeni, da jih na sprejemni strani ni treba še posebej urejati po vrsti. Sprejemni protokolni osebek torej potrebuje sprejemni vmesnik le za eno sporočilo; vanj shrani sprejeto sporočilo, da ga obdela. Če se kakšno protokolno sporočilo med prenosom pokvari ali izgubi, mora oddajnik po izteku časovnika ponovno poslati ne le to sporočilo, ampak tudi vsa sporočila, ki mu sledijo, čeprav so nekatera izmed njih morda brez napake prispela do sprejemnika. Na ta način zagotovimo, da sprejemni protokolni osebek predaja uporabniku njemu namenjena sporočila v pravilnem vrstnem redu.

Sprejemnik sprejme in preda uporabniku v njih vsebovana uporabniška sporočila le tista protokolna sporočila, ki jih po vrstnem redu pričakuje,. Če sprejemnik hrani sekvenčno številko pričakovanega informacijskega sporočila v spremenljivki V_r, bo torej po sprejemu sporočila s številko, ki je enaka vrednosti V_r, z ustreznim primitivom predal vsebovano uporabniško sporočilo uporabniku in inkrementiral vrednost spremenljivke V_r. Sporočila, ki se med prenosom pokvarijo, sprejemnik zavrže. Ko sprejemnik sprejme

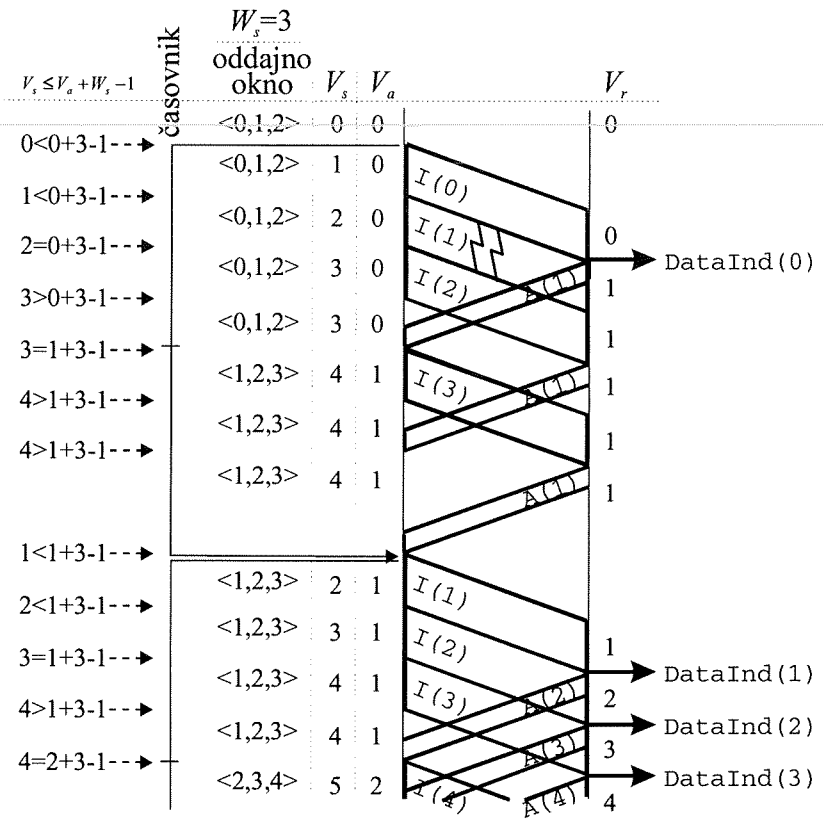
informacijsko protokolno sporočilo brez napake (torej ne glede na to, ali je prišlo v pravem vrstnem redu), odda oštevilčeno potrditev, v kateri sporoča, sprejem katerega sporočila pričakuje.

Če se oddajniku časovnik izteče, predpostavlja, da se je izgubilo informacijsko protokolno sporočilo, katerega potrditev najprej pričakuje in katerega sekvenčno številko hrani v spremenljivki V_a (ta sekvenčna številka tudi označuje začetek oddajnega okna). Zato najprej spremeni vrednost spremenljivke V_s

$$V_s:=V_a \tag{78}$$

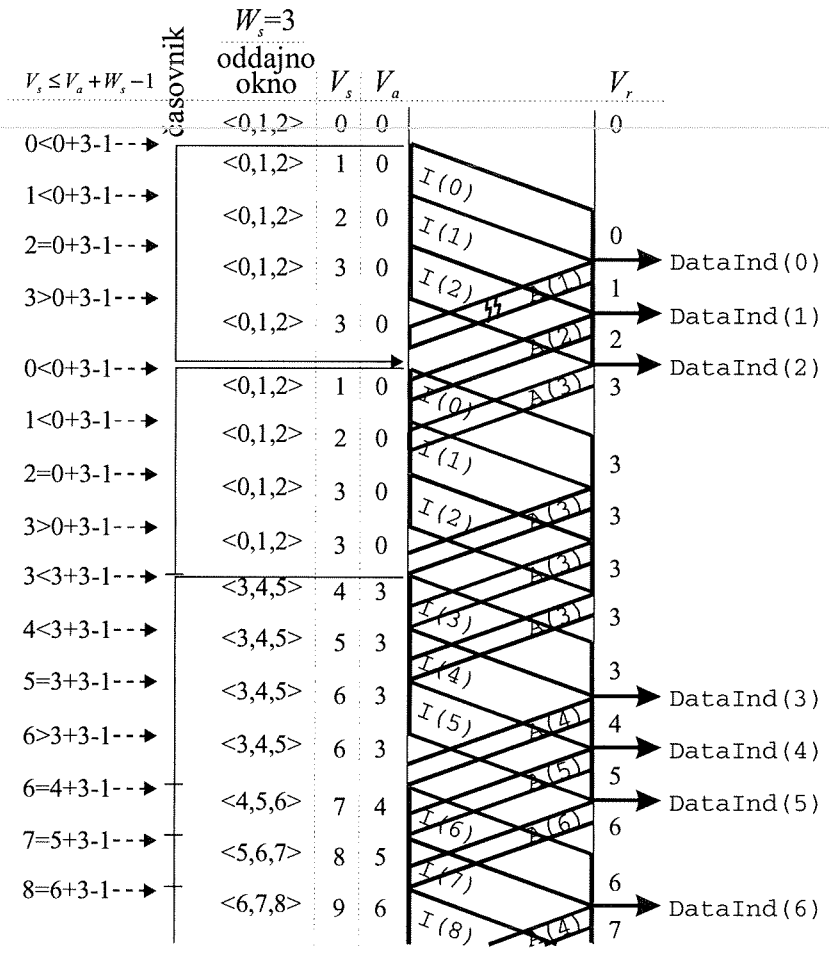
(torej "gre nazaj" - "go back"), ponovno odda informacijsko protokolno sporočilo s to številko in nato nadaljuje z oddajanjem naslednjih sporočil.

Če sprejemnik lepo po vrsti sprejema informacijska protokolna sporočila, pošilja oddajniku vedno nove potrditve, zaradi česar le-ta vsakič ustavi časovnik (in ga po potrebi spet požene) ter pomakne drseče okno naprej. Če pa se neko informacijsko protokolno sporočilo med prenosom pokvari ali izgubi, sprejemnik ne pošlje potrditve, po sprejemu naslednjih sporočil pa ponovno pošilja že poslano potrditev, saj še vedno pričakuje sporočilo, ki se je izgubilo; ker oddajnik ne sprejema novih potrditev, se mu časovnik izteče. Tak primer poteka komunikacije kaže slika Slika 9-13.



Slika 9-13. Primer komunikacijskega procesa po protokolu s ponavljanjem N sporočil

Prav tako pa se tudi lahko zgodi, da sprejemnik uspešno sprejme informacijsko protokolno sporočilo, preda v njem vsebovano uporabniško sporočilo uporabniku in pošlje potrditev, ki pa se, smola!, izgubi. Če sprejemnik kmalu za tem prejme naslednje informacijsko protokolno sporočilo in odda potrditev, ki jo oddajnik sprejme, še preden se mu izteče časovnik, bo oddajnik časovnik ustavil, saj nova potrditev velja tudi za nazaj. Če pa se oddajniku časovnik prej izteče, bo ponovno oddal sicer že oddano in tudi uspešno sprejeto informacijsko protokolno sporočilo, prav tako pa tudi vsa naslednja. Kot vidimo v scenariju na sliki Slika 9-14, sprejemnik ponovljenih sporočil sicer ne sprejema, saj jih ne pričakuje, pošilja pa potrditve, v katerih sporoča, katero sporočilo pričakuje. Bralec se lahko sam prepriča, da v scenariju na sliki Slika 9-14 protokol ne bi pravilno deloval, če sprejemnik ne bi poslal potrditve potem, ko je sprejel informacijsko protokolno sporočilo, ki sicer ne vsebuje napake, a ga ne pričakuje.



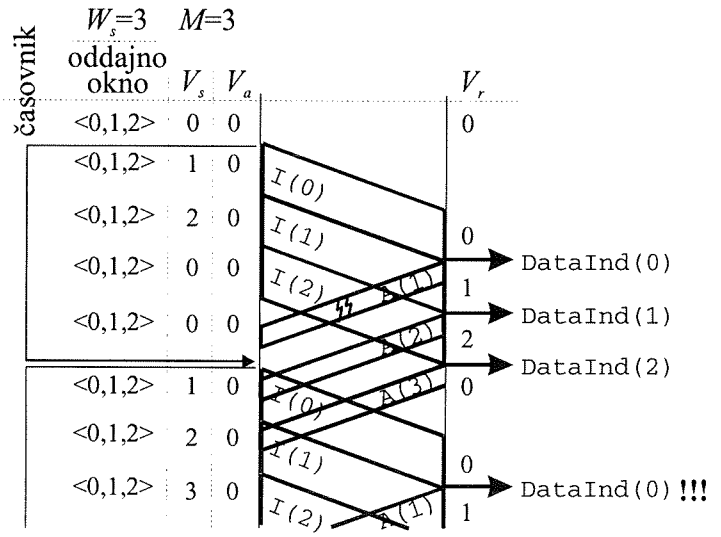
Slika 9-14. Primer komunikacijskega procesa ob izgubi potrditve

9.4.1.2 Modul štetja

Kot smo že povedali, informacijska protokolna sporočila štejemo po modulu M; to pomeni, da lahko vsako sekvenčno številko po določenem času ponovno spet uporabimo. Z drugimi besedami, sporočila štejemo od 0 po vrsti do M-1, nato pa začnemo spet z 0. Sedaj moramo seveda vse operacije, ki vključujejo sekvenčne številke (inkrementiranje, dekrementiranje), računati po modulu M. Pogoj $V_a < a \le V_s$ interpretiramo kot "a je med izključno V_a in vključno V_s, pogoj $V_a \le V_s \le V_a + W_s - 1$ pa interpretiramo kot "V_s je vključno med V_a in V_a+W_s-1".

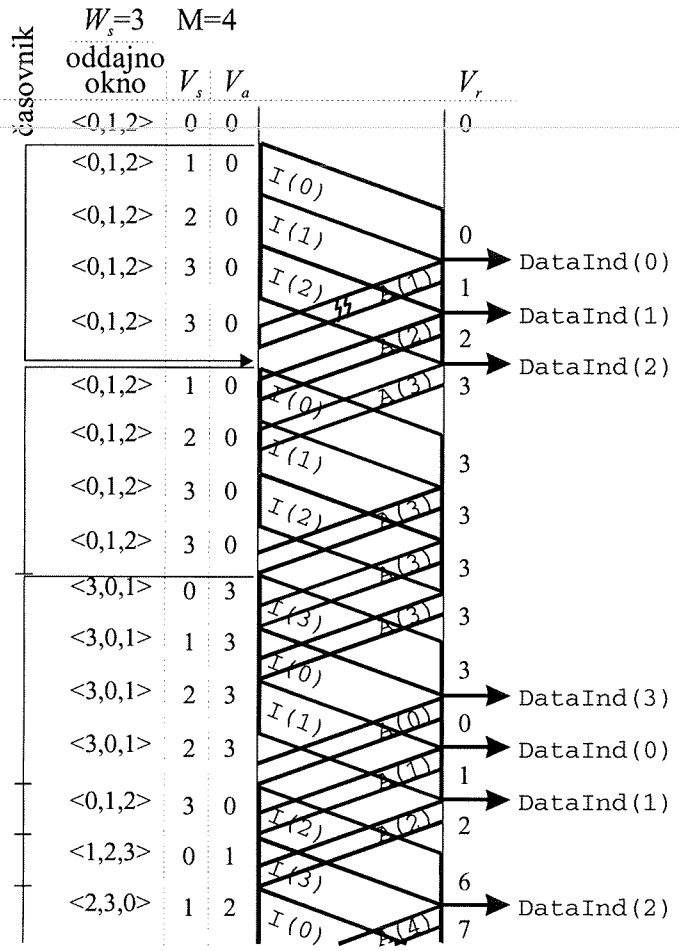
Seveda pa se nam tu postavi pomembno vprašanje, kakšen mora biti modul štetja M. Biti mora seveda takšen, da pri štetju in potrjevanju sporočil ne more priti do pomot in zmešnjav in s tem do napačnega delovanja protokola. Na prvi pogled bi lahko sodili, da

mora biti modul štetja enak ali večji od širine oddajnega okna. Vendar pa ni tako. O tem se bomo prepričali kar na scenariju iz slike Slika 9-14, le da bomo informacijska protokolna sporočila tokrat šteli po modulu $M=W_s=3$. Kot vidimo na sliki Slika 9-15, se potrditev sporočila št. 0 pokvari, zato se oddajniku izteče časovnik in ta ponovno pošlje informacijsko protokolno sporočilo št. 0. Ko to prispe do sprejemnika, je vrednost njegove spremenljivke $V_r=0$, zato sprejemnik sporočilo z veseljem* sprejme in vsebovano uporabniško sporočilo preda uporabniku. V resnici pa je uporabnik prav to sporočilo že dobil, saj se je pri prejšnjem poskusu pokvarilo potrdilo, ne pa informacijsko protokolno sporočilo. Torej je uporabnik prejel duplikat (dve kopiji istega sporočila), kar pomeni napako protokola. Da se v tem primeru dogaja nekaj čudnega, pa lahko vidimo tudi po dejstvu, da ima v nekem trenutku sekvenčna številka 0 trojno vlogo, saj je hkrati številka novega sporočila, številka sporočila, ki ga pričakuje sprejemnik in najstarejša še ne potrjena številka. Kot lahko vidimo na sliki Slika 9-16, pa do tovrstnih težav ne pride, če sporočila štejemo po modulu $M=4$, bralec pa lahko sam preveri, da tudi ne, če je pri štetju po modulu $M=3$ širina oddajnega okna le $W_s=2$. Pri študiju slike Slika 9-16 se moramo zavedati, da prvo in drugo informacijsko protokolno sporočilo s sekvenčno številko 0 nosita isto uporabniško sporočilo, medtem ko tretje sporočilo `I(0)` nosi drugo uporabniško sporočilo.



Slika 9-15. Napačno delovanje zaradi premajhne širine okna

* Bralec naj vzame tovrstne hudomušne pripombe avtorja z rezervo; v resnici protokolni osebki seveda ne občutijo prav nikakršnih čustev!



Slika 9-16. Pravilno delovanje pri dovolj veliki širini okna

To spoznanje pa lahko tudi posplošimo. Vzemimo, da imamo oddajno okno širine W_s polno, kar pomeni, da velja $V_s=V_a+W_s$ (seveda po modulu M). Če je sprejemnik brez napake in v pravem vrstnem redu sprejel vsa oddana informacijska protokolna sporočila, velja tudi $V_r=V_s$. Pri modulu štetja $M=W_s$ to hkrati pomeni tudi $V_r=V_a+M=V_a$. Vrednost $V_r=V_a$ je v celotnem sistemu torej dvoumna. Če se sedaj zaradi izgube potrditve izteče časovnik, bo oddajnik ponovno oddal informacijsko protokolno sporočilo s številko V_a, to pa je prav tista sekvenčna številka, ki jo sprejemnik pričakuje. V resnici pa gre za duplikat. Če modul štetja povečamo za vrednost 1, tako da velja $W_s=M-1$, v gornjih pogojih (oddajno okno polno, vsa oddana informacijska protokolna sporočila tudi pravilno sprejeta) velja $V_r=V_a+W_s=V_a+M-1=V_a-1$, kar pa je že izven okna. Dvoumnost torej izgine in nimamo težav. Ker pa smo pravkar obravnavali scenarij, ki je, kar se tiče razlike med številkama V_r in V_a, najneugodnejši (pri tem scenariju je ta razlika največja),

lahko za protokol s ponavljanjem N sporočil takole zapišemo potrebno zvezo med širino oddajnega okna in modulom štetja:

$$W_s \leq M - 1 \quad . \tag{79}$$

9.4.1.3 Različice protokola s ponavljanjem N sporočil

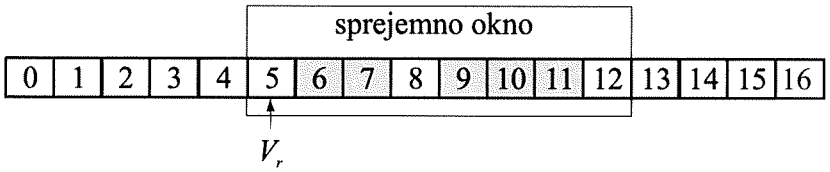
V tem razdelku smo si ogledali osnovno delovanje protokola s ponavljanjem N sporočil, kar nam je omogočilo, da smo ta protokol opisali na razmeroma preprost način. Pri načrtovanju protokolov pa je mogoče uporabiti tudi dodatne mehanizme, ki lahko lastnosti protokola še izboljšajo. Eden takih mehanizmov je uporaba negativnih potrditev poleg pozitivnih. Če oddajnik sprejme negativno potrditev oziroma zavrnitev, mora prav tako ponovno poslati ne le zavrnjeno sporočilo, ampak tudi vsa naslednja. Uporaba negativnih potrditev lahko izboljša učinkovitost protokola in zakasnitve, saj lahko negativna potrditev pride do oddajnika prej, kot pa bi se le-temu iztekel časovnik.

9.4.2 Protokoli s selektivnim ponavljanjem

9.4.2.1 Opis in specifikacija protokola s selektivnim ponavljanjem

Pri komunikaciji po protokolu s ponavljanjem N sporočil mora oddajnik v primeru, da se je eno samo sporočilo pokvarilo ali izgubilo, ponovno poslati ne le to sporočilo, ampak tudi vsa naslednja, čeprav so bila le-ta prenešena brez napak. To seveda predstavlja precejšnjo potrato kapacitete kanala. Izkoristek protokola je zaradi tega pri prenosu preko realnega kanala (pri idealnem metoda ARQ sploh ni potrebna!) slabši, povprečna zakasnitev pa večja. Temu problemu se izognemo z uporabo protokola s selektivnim ponavljanjem, kjer, kot že samo ime pove, oddajnik ponovno pošlje le tista sporočila, ki so se izgubila, ne pa tudi tistih, ki jim sledijo. S tem pa seveda sporočila ne prihajajo k sprejemnemu protokolnemu osebku več po vrsti. Ker še vedno velja zahteva, da mora ponudnik storitve predati uporabniku uporabniška sporočila v pravilnem vrstnem redu, jih mora sprejemni protokolni osebek po sprejemu in pred predajo uporabniku urejati. Zato pa potrebuje nekaj vmesnikov (to je določeno količino spomina), v katerih hrani tista sporočila, ki jih je že sprejel (seveda brez napake), a jih še ni uredil po vrsti. Očitno torej moramo učinkovitejšo izrabo kapacitete kanala plačati z večjo porabo drugih virov (spomina na sprejemni strani).

Vzemimo sedaj, da ima sprejemnik za urejanje sporočil na razpolago W_r vmesnikov. Sporočilo z najnižjo sekvenčno številko, ki še ni sprejeto, naj ima sekvenčno številko V_r, kar pomeni, da so vsa protokolna sporočila do vključno V_r-1-tega že sprejeta in urejena po vrsti ter v njih vsebovana uporabniška sporočila predana uporabniku. Sprejemnik bo torej lahko sprejel in urejal po vrsti V_r-to in še W_r-1 naslednjih sporočil. Le sporočila V_r, V_r+1, ..., V_r+W_r-1 bo torej pripravljen sprejeti in jih shraniti v ustrezen vmesnik (če so seveda prispela brez napake), vsa ostala pa bo zavrgel, tudi če prispejo brez napake. Interval informacijskih protokolnih sporočil, ki jih je sprejemnik pripravljen sprejeti, oziroma njihovih sekvenčnih številk, imenujemo **sprejemno okno** (ang. **receive window**), število teh sporočil W_r pa **širina sprejemnega okna** (ang. **receive window width**). Sprejemnik mora za vsako lokacijo v sprejemnem oknu vedeti, ali je že zasedena z veljavnim sporočilom. Slika Slika 9-17 kaže primer delno zapolnjenega sprejemnega okna s širino W_r=8. Že zapolnjene lokacije so sivo obarvane.



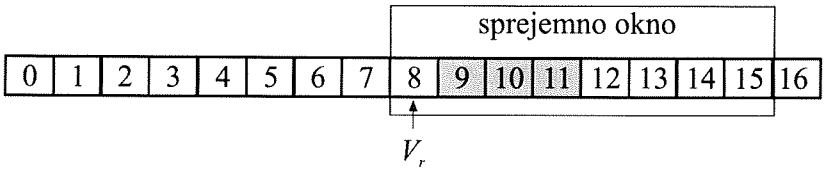
Slika 9-17. Sprejemno okno širine 8

Sprejemnik torej sme sprejeti sporočilo s sekvenčno številko s, če velja

$$V_r \le s \le V_r + W_r - 1 \quad . \tag{80}$$

Ko sprejemnik brez napake sprejme sporočilo, ki zadošča pogoju (80), ga shrani na mestu, ki ustreza sekvenčni številki sprejetega sporočila, in si zapomni zasedenost tega mesta v sprejemnem oknu. Če sprejme sporočilo s sekvenčno številko V_r, v njem vsebovano uporabniško sporočilo preda uporabniku, isto pa stori z vsemi že sprejetimi sporočili, ki neposredno sledijo V_r-temu (torej do prve "luknje"), Nato spremeni vrednost spremenljivke V_r tako, da spet kaže na prvo še ne sprejeto sporočilo (torej "luknjo" v sprejemnem oknu) in pošlje kumulativno potrditev s številko V_r. S tem pomakne sprejemno okno za eno ali več mest naprej in potrdi sprejem vseh informacijskih protokolnih sporočil do vključno V_r-1-tega. Če npr. v stanju, ki ga kaže slika Slika 9-17, sprejemnik brez napake sprejme protokolno sporočilo št. 5, preda uporabniku uporabniška sporočila, ki jih vsebujejo protokolna sporočila št. 5, 6 in 7 (seveda v tem

vrstnem redu), priredi spremenljivki V_r vrednost 8 in pošlje oddajniku potrditev s sekvenčno številko 8. Sprejemno okno je po tej operaciji takšno, kot ga kaže slika Slika 9-18.



Slika 9-18. Sprejemno okno iz slike 9 17 po sprejemu informacijskega protokolnega sporočila s številko 5

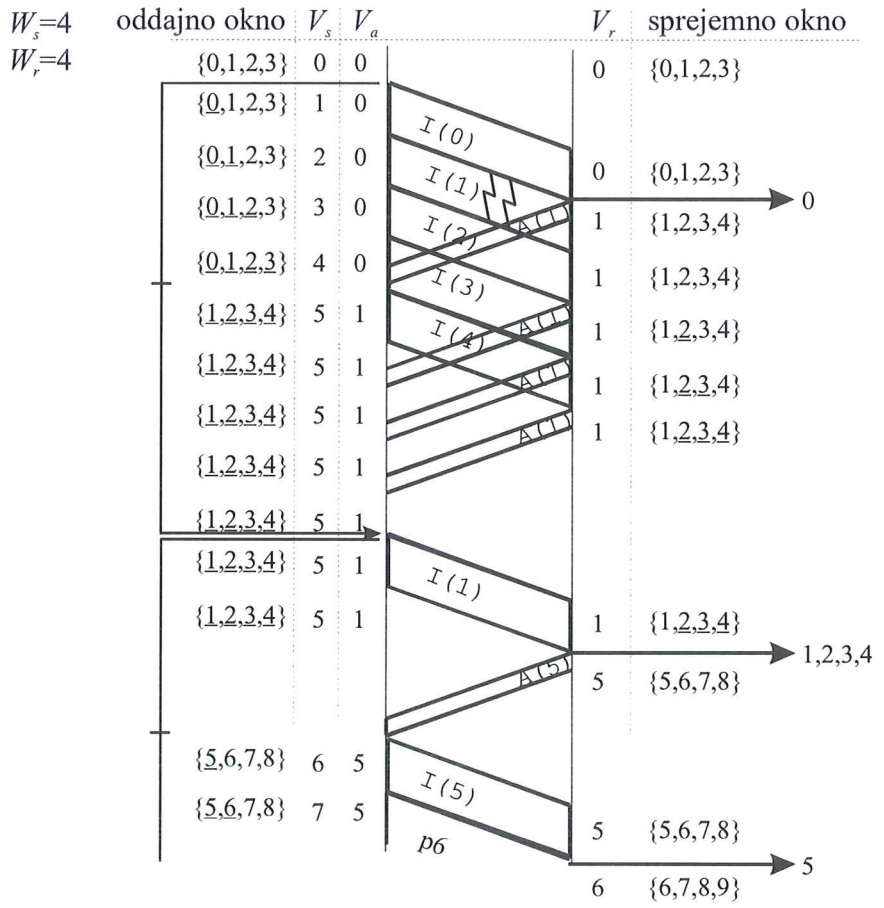
Očitno tudi sprejemno okno, podobno kot oddajno, "drsi" vzdolž zaporedja sekvenčnih številk.

Oddajni protokolni osebek pri protokolu s selektivnim ponavljanjem prav tako uporablja oddajno okno širine W_s ter spremenljivki V_s in V_a kot pri protokolu s ponavljanjem N sporočil. Vendar pa se tu vrednost spremenljivke V_s ne zmanjša, če se je iztekel časovnik. Oddajnik le ponovno odda sporočilo s sekvenčno številko V_a, saj predpostavlja, da se je izgubilo prav to (najstarejše še ne potrjeno) sporočilo, pri tem pa upa, da so temu sledeča sporočila, ki jih je tudi že oddal, varno shranjena v vmesniku sprejemnika. Zato lahko nadaljuje z oddajanjem sporočila, ki ga določa vrednost spremenljivke V_s, če seveda ima na voljo uporabniške podatke in oddajno okno ni polno.

Pri protokolu s selektivnim ponavljanjem sta torej širini obeh oken, oddajnega in sprejemnega, različni od 1. Vprašajmo se torej, ali naj tudi ti dve širini povezuje kakšna relacija. Imamo seveda tri izključujoče se možnosti: $W_s>W_r$, $W_s<W_r$ in $W_s=W_r$. V prvem primeru, ko je širina sprejemnega okna manjša od širine oddajnega okna, sprejemnik v najneugodnejšem primeru zaradi preozkega okna ne bo mogel shraniti vseh že oddanih sporočil, ki sledijo izgubljenemu sporočilu, in jih bo moral zavreči, čeprav se v kanalu niso niti pokvarila niti izgubila - torej tak primer ni smiselen. Po drugi strani pa sporočila v sprejemnem oknu ne morejo zavzeti več kot W_s mest, saj mora vedno veljati $V_a \leq V_r$, zato bi v drugem primeru imeli nekoristno velik sprejemni vmesnik. Torej je edini smiselni pogoj

$$W_s = W_r \tag{81}$$

Podobno, kot smo si v prejšnjem razdelku na zgledu v časovnem diagramu ogledali delovanje protokola s ponavljanjem N sporočil, si zdaj na sliki Slika 9-19 oglejmo na podobnem zgledu še delovanje protokola s selektivnim ponavljanjem.



Slika 9-19. Komunikacijski proces po protokolu s selektivnim ponavljanjem

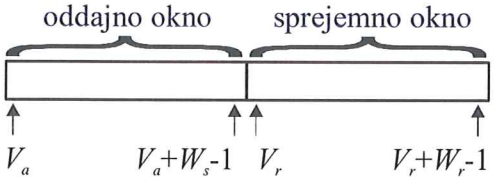
Če ta zgled primerjamo z zgledom protokola s ponavljanjem N sporočil, lahko ugotovimo, da so tu sporočila, ki sledijo pokvarjenemu sporočilu, poslana le enkrat in ne dvakrat. S tem manj obremenjujemo kanal (kar je še zlasti pomembno, če je kanal multipleksen in ga sočasno uporabljajo tudi drugi oddajniki). Seveda pa s tem tudi odpade možnost, da bi se eno izmed teh sporočil pokvarilo ali izgubilo pri drugi "vožnji".

9.4.2.2 Modul štetja

Seveda tudi pri protokolu selektivnega ponavljanja štejemo sporočila po modulu *M*. Tudi tu mora biti modul štetja dovolj velik, da ne pride do napačnega delovanja protokola. Najneugodnejši primer je vsekakor ta, ko je oddajno okno polno in se obe okni

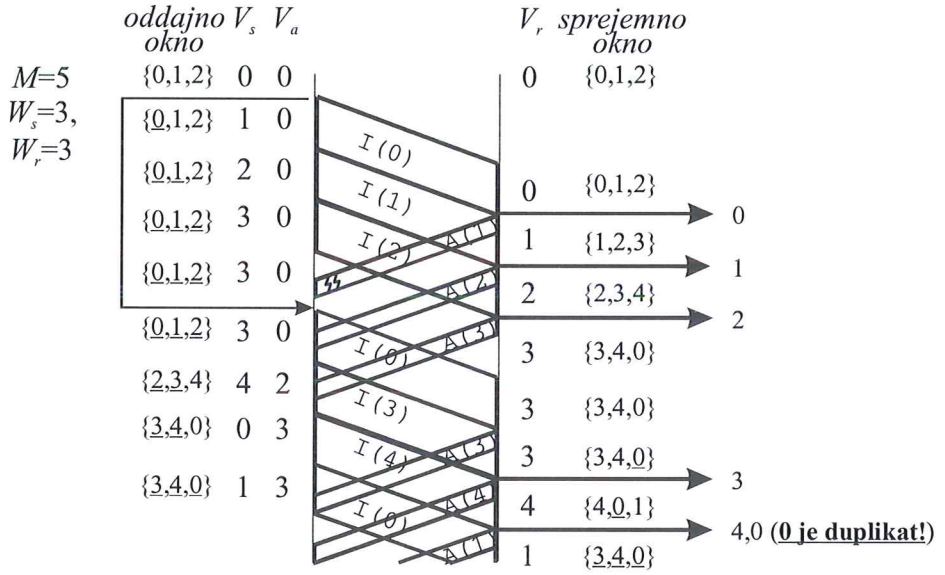
(oddajno in sprejemno) ne prekrivata (V_r=V_s=V_a+W_s), kar lahko vidimo na sliki Slika 9-20. Iz takega razmišljanja oziroma iz slike sledi pogoj

$$M \geq W_s + W_r = 2 \cdot W_s = 2 \cdot W_r \,. \tag{82}$$



Slika 9-20. Najneugodnejši položaj oddajnega in sprejemnega okna

Tudi to trditev bomo ilustrirali z zgledom. Položaj na sliki Slika 9-20 nastopi v primeru, ko je oddajno okno polno, izgubijo pa se prva (ali vse) potrditev. Primer je torej enak kot smo ga že obravnavali pri protokolu s ponavljanjem N sporočil. Slika Slika 9-21 prikazuje potek komunikacije pri kombinaciji W_s=W_r=3 in M=5<2W_s. Vidimo, da tudi tu uporabnik prejme duplikat sporočila, ki ga nosi PDU s sekvenčno številko 0. Bralec se bo lahko hitro sam prepričal, da do te napake ne pride, če vzamemo M=6, s čimer izpolnimo pogoj (82).



Slika 9-21. Nepravilno delovanje zaradi premajhnega modula štetja

9.4.2.3 Različice protokola

Tudi protokol s selektivnim ponavljanjem je možno izboljšati na ta način, da ima sprejemnik tudi možnost selektivne zavrnitve tistih sporočil, za katera sumi, da so se pri

prenosu pokvarila. Do takega suma pride bodisi, če dobi pokvarjeno sporočilo (torej tako z neveljavno zaščitno kodo), ali pa pri sprejemu enega izmed sporočil ne dobi, čeprav je že sprejel prejšnje in kasnejše sporočilo. Potem lahko ta protokol imenujemo tudi **protokol selektivnega zavračanja** (ang. **selective reject protocol**).

9.5 Primerjava protokolov z drsečim oknom

Če pazljivo primerjamo protokole ARQ s čakanjem, s ponavljanjem N sporočil in s selektivnim ponavljanjem, lahko ugotovimo, da gre pravzaprav za en protokol s tremi osnovnimi različicami. Zato vsi trije protokoli (torej s čakanjem, ponavljanjem N sporočil in selektivnim ponavljanjem) spadajo v razred **protokolov z drsečim oknom**.

Vsi trije osnovni protokoli se med seboj razlikujejo po širini oddajnega in sprejemnega okna ter po strategiji ponovljenega oddajanja po izteku časovnika.

Najbogatejše možnosti ima protokol s selektivnim ponavljanjem, saj oddajnik in sprejemnik razpolagata z oknom širine, ki je večja od ena, $W_s=W_r>1$, in seveda tudi z ustreznim oddajnim oziroma sprejemnim vmesnikom. Zato lahko oddajnik zaporedoma odda več informacijskih protokolnih sporočil, ne da bi prejel njihove potrditve, sprejemnik pa lahko shrani več sprejetih sporočil, čeprav med njimi nekatera še manjkajo.

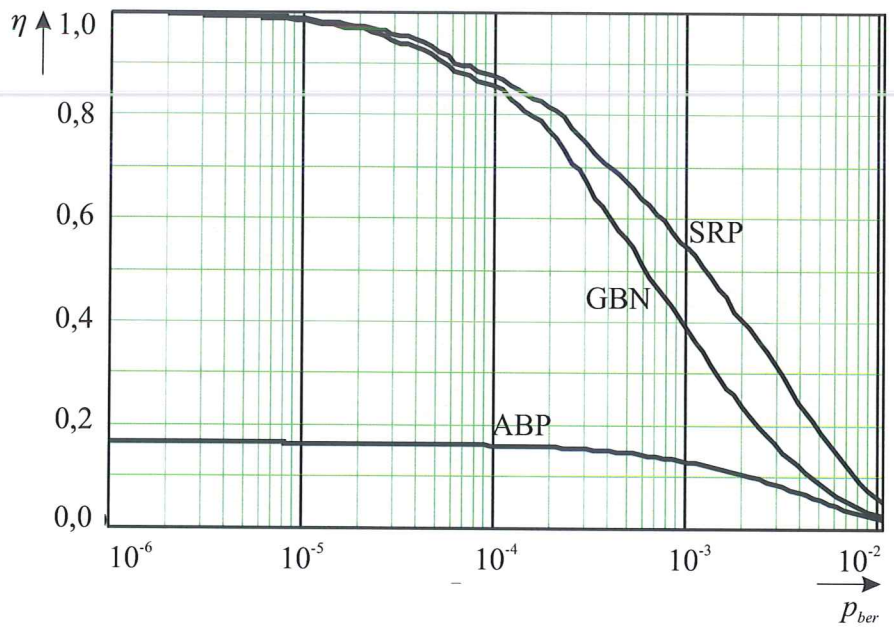
Če je širina sprejemnega okna enaka $W_r=1$, ima sprejemnik na voljo le eno samo sporočilo, ki ga sme sprejeti, to pa je točno tisto sporočilo, ki ga pričakuje; vsa ostala sprejeta sporočila zavrača. S tem pa sprejemnik seveda nima več možnosti za urejanje vrstnega reda sporočil. Sprejemnik torej sme sprejemati sporočila edinole v pravem vrstnem redu. Lahko pa pri tem oddajnik še vedno odda nekaj (največ W_s) sporočil, čeprav še nima njihovih potrditev. Vse to pa so značilnosti protokola s ponavljanjem N sporočil. Protokol s ponavljanjem N sporočil je torej primer protokola z drsečim oknom s širino sprejemnega okna 1. Pogoj (82) $W_s + W_r \leq M$ v primeru protokola s ponavljanjem N sporočil dobi obliko $W_s + 1 \leq M$ oziroma $W_s \leq M - 1$.

Sedaj pa vzemimo še primer, ko poleg pogoja $W_r=1$ velja tudi pogoj $W_s=1$. Ne le, da sme tudi sedaj sprejemnik sprejeti le tisto sporočilo, ki ga pričakuje, ampak se tudi oddajniku po vsakem oddanem sporočilu zapolni oddajno okno, zato mora po vsakem oddanem sporočilu čakati na njegovo potrditev, preden sme oddati naslednje sporočilo. To pa je značilnost protokolov s čakanjem. Protokol s čakanjem je torej primer protokola

z drsečim oknom s pogojema W_r=1 in W_s=1. Tudi tu je izpolnjen pogoj (82) $W_s + W_r \leq M$ oziroma $M \geq 2$.

Obravnavali smo že učinkovitost protokolov s čakanjem in učinkovitost kontinuirnih protokolov v primeru, ko v kanalu ni izgub. Slednja seveda velja tako za protokol s ponavljanjem N sporočil, kot tudi za protokol s selektivnim ponavljanjem, saj med tema protokoloma ni razlik, če ni izgub. V primeru, da izgube so, pa bo učinkovitost katerega koli protokola seveda manjša.

Intuitivno smo že pričakovali, da bo protokol s selektivnim ponavljanjem učinkovitejši kot protokol s ponavljanjem N sporočil, ta pa učinkovitejši kot protokol s čakanjem. Sedaj bomo ta predvidevanja podkrepili s simulacijskimi rezultati. Učinkovitosti treh protokolov bomo primerjali za primer kanala s propagacijskim časom 500 μs in hitrostjo prenosa informacije 1 Mb/s. Dolžina celotnega informacijskega protokolnega sporočila naj bo 200 bitov, dolžina režije v njem in hkrati tudi dolžina potrditvenega sporočila pa 40 bitov. Širina oddajnega okna naj bo pri obeh kontinuirnih protokolih taka, da lahko oddajnik ves čas oddaja, če napak ni (W_s=7), širina sprejemnega okna pri protokolu s selektivnim ponavljanjem pa naj bo enaka širini oddajnega okna. Pogostnost bitnih napak v kanalu (*ber*), ki je hkrati tudi verjetnost bitne napake p_{ber}, se spreminja od 10^{-6} do 10^{-2}. Slika Slika 9-22 kaže diagram učinkovitosti protokola η, ki jo vidita protokolna osebka, v odvisnosti od pogostnosti bitnih napak v kanalu; oznake ABP, GBN in SRP pomenijo protokol z alternirajočim bitom (Alternating Bit Protocol), protokol s ponavljanjem N sporočil (Go-Back-N) oziroma protokol s selektivnim ponavljanjem (Selective Repeat Protocol).

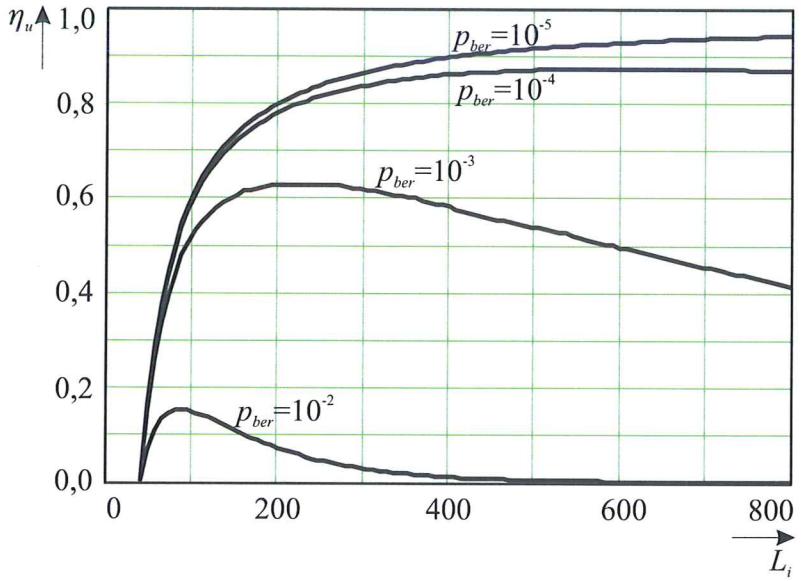


Slika 9-22. Odvisnost učinkovitosti protokola od pogostnosti napak

Razmere se lahko nekoliko spremenijo, če protokolna osebka uporabljata navidezni kanal, ki je implementiran z nepovezavno orientiranim protokolom. Tu lahko pridejo protokolna sporočila do sprejemnika v drugačnem vrstnem redu, kot jih je oddal oddajnik. V tem primeru je ugodneje izbrati protokol s selektivnim ponavljanjem kot protokol s ponavljanjem N sporočil, saj bi pri slednjem sprejemnik zavrgel marsikatero sporočilo zgolj zaradi tega, ker je v kanalu prehitelo neko drugo sporočilo, pa čeprav v samem kanalu ne bi bilo izgub. To pa bi zmanjšalo učinkovitost.

Odvisnost učinkovitosti, ki jo vidita uporabnika, od pogostnosti napak se od tiste na sliki Slika 9-22 razlikuje le za faktor režije. Odvisnost učinkovitosti, ki jo vidita uporabnika, od dolžine protokolnega sporočila pa je bolj kompleksna. Večanje dolžine sporočila namreč po eni strani veča verjetnost izgube sporočila v kanalu in s tem manjša učinkovitost, po drugi strani pa se z daljšanjem sporočila pri enaki režiji izboljšuje faktor režije in z njim vred učinkovitost, ki jo vidita uporabnika. Zato ima ta učinkovitost pri dani pogostnosti napak svoj maksimum pri neki optimalni dolžini protokolnega sporočila. Pričakujemo lahko, da bo optimalna dolžina tem večja, čim manjša bo pogostnost napak, kar nam potrjujejo tudi simulacijski rezultati za protokol s selektivnim ponavljanjem na

sliki Slika 9-23. Spet smo vzeli kanal s propagacijskim časom 500 μs in hitrostjo prenosa informacije 1 Mb/s, dolžino režije 40 bitov in širino oken W_s=W_r=100.



Slika 9-23. Odvisnost učinkovitosti, ki jo vidita uporabnika, od dolžine protokolnega sporočila

9.6 Enosmerna, izmenično enosmerna in dvosmerna izmenjava sporočil

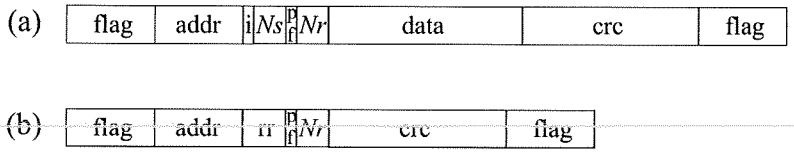
Doslej smo si zaradi preprostejše obravnave predstavljali, da en protokolni osebek (oddajnik) le oddaja informacijska protokolna sporočila, drugi (sprejemnik) pa jih le sprejema. Pri potrditvah je vloga seveda ravno obratna. Prenos uporabniške informacije je v takem primeru seveda enosmeren (simpleksen). V realnem svetu pa pogosto potrebujemo vsaj polduplekSno, če ne že kar dupleksno izmenjavo sporočil.

Pri polduplekSni izmenjavi sporočil gre za to, da ima nekaj časa en protokolni osebek vlogo oddajnika sporočil in sprejemnika potrditev, drugi pa ravno nasprotno vlogo, potem pa oba osebka svoji vlogi zamenjata. Vprašanje je le, na kakšen način se dogovorita za zamenjavo vlog. Ena možnost je, da je eden izmed protokolnih osebkov nadrejen in ima pravico pozvati podrejeni osebek k oddaji. Tak protokol je **neuravnotežen** (ang. **unbalanced**). Pri **uravnoteženem** protokolu (ang. **balanced**) se ta problem lahko rešuje s konceptom **žetona** (ang. **token**). Osebek, ki poseduje žeton, ima pravico oddaje sporočil, drugi osebek pa pošilja le potrditve. Seveda pa žeton lahko zamenja lastnika. To se zgodi tako, da osebek, ki trenutno ima žeton, le-tega s posebnim

sporočilom preda drugemu osebku. Protokol, ki je pravičen do obeh osebkov, bo seveda določil, kdaj je osebek dolžan predati žeton: bodisi po vnaprej določenem največjem številu poslanih sporočil ali po določenem času, vsekakor pa, če nima več sporočila, ki bi ga poslal.

Če imamo na razpolago dupleksni kanal in uporabljamo kontinuirno metodo ARQ, je seveda edinole smiselno, da lahko oba protokolna osebka hkrati oddajata informacijska protokolna sporočila in jih hkrati tudi oba sprejemata. Vsak izmed obeh osebkov torej oddaja tako informacijska kot tudi nadzorna protokolna sporočila. Običajno so informacijska protokolna sporočila precej daljša od nadzornih, saj prva vsebujejo uporabniška sporočila, druga pa ne. Režija pa je pri obeh vrstah sporočil enako dolga, saj katerokoli sporočilo potrebuje sinhronizacijsko sekvenco, identifikacijo tipa sporočila, zaščitno kodo za odkrivanje napak in pa sekvenčno številko informacijskega sporočila v prvem primeru oziroma sekvenčno številko potrjenega sporočila v drugem primeru. V idealnem primeru, ko bi imeli brezizgubni kanal, seveda nadzornih protokolnih sporočil sploh ne bi potrebovali, saj nam dodatno obremenjujejo kanal, podobno kot tudi režija samih informacijskih protokolnih sporočil. Žal pa na to ne moremo računati. Zato se skušamo izogniti vsaj režiji potrditvenih sporočil (razen seveda njihovega bistvenega dela - številk potrjenih sporočil). To storimo tako, da protokolni osebek potrditve sporočil, ki jih je sprejel kot sprejemnik, preprosto kar vključi v informacijska protokolna sporočila, ki jih oddaja kot oddajnik. Tako potrditve uporabijo sinhronizacijsko sekvenco in zaščitno kodo informacijskih sporočil, pa tudi posebne identifikacije ne potrebujejo, saj protokol določa, na katerem mestu v informacijskem protokolnem sporočilu se nahaja potrditev informacijskih sporočil, ki potujejo v obratni smeri. Takšno taktiko dodajanja potrditev informacijskim sporočilom imenujemo **natovarjanje** (ang. **piggybacking**).

Kot primer natovarjanja nam slika Slika 9-24 kaže (a) informacijsko protokolno sporočilo in (b) nadzorno protokolno sporočilo pozitivne potrditve ISO protokola HDLC (High-level Data Link Control). Polji "i" in "rr" vsebujeta identifikacijo informacijskega protokolnega sporočila oziroma pozitivne potrditve, polje "data" je uporabniško sporočilo, *Ns* je sekvenčna številka informacijskega sporočila, vrednost *Nr* pa v obeh primerih predstavlja kumulativno potrditev pravilnega sprejema vseh informacijskih sporočil do vključno *Nr*-1-tega.



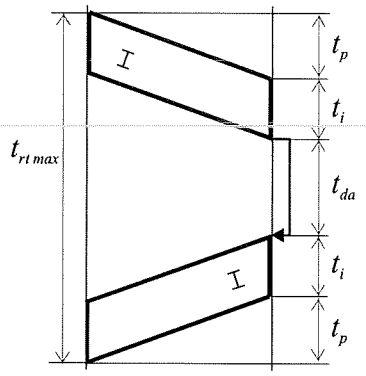
Slika 9-24. Format protokolnega sporočila po protokolu HDLC

Vendar pa protokolni osebek informacijskega protokolnega sporočila za prenos preko kanala nima kadarkoli na razpolago, ker pač nima vedno na razpolago uporabniškega sporočila za oddajo; če bi ga imel, bi to namreč pomenilo, da je polno obremenjen in bi zato njegova čakalna vrsta naraščala preko vseh mej; to pa bi pomenilo poddimenzioniran sistem, ki bi slabo deloval. Če v trenutku, ko bi moral osebek poslati potrditev, informacijskega protokolnega sporočila ni na razpolago, lahko osebek s potrditvijo počaka na prvo informacijsko protokolno sporočilo, ali pa pošlje potrditev v posebnem nadzornem protokolnem sporočilu. Čakanje na informacijsko protokolno sporočilo seveda ne sme biti predolgo, da se partnerskemu osebku medtem ne bi iztekel časovnik. Zato osebek običajno uporablja poseben časovnik, ki ga z iztekom opomni, da je že čas za oddajo potrditve, če le predolgo ni bilo nobenega informacijskega sporočila za oddajo. Na ta način se osebek sam prilagodi prometnemu pretoku po kanalu: ko je ta velik, natovarja potrditve v informacijska protokolna sporočila; ko je prometni pretok majhen, pa potrditve pošilja v posebnih nadzornih sporočilih.

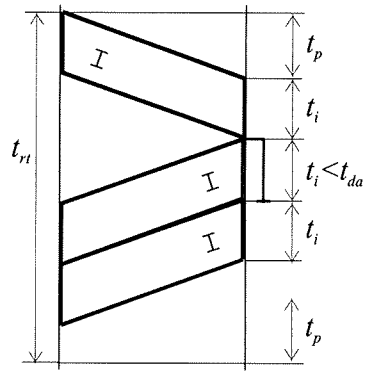
Seveda morata biti nastavitvi obeh časovnikov (oddajnika in sprejemnika) med seboj usklajeni. Čas do potrditve je namreč v najneugodnejšem primeru enak

$$t_{rt} = 2 \cdot t_i + t_{da} + 2 \cdot t_p \ . \tag{83}$$

Pri tem smo čas izteka časovnika označili kot t_{da}, saj gre v bistvu za zakasnjeno potrditev. Tak najneugodnejši primer lahko vidimo na sliki Slika 9-25. Pri tem pa se moramo zavedati, da čas izteka časovnika t_{da} ne sme biti krajši od časa oddajanja informacijskega protokolnega sporočila t_i, saj je sporočilo, ki ga moramo potrditi, morda prispelo do osebka v trenutku, ko je le-ta pravkar začel oddajati neko informacijsko protokolno sporočilo; potrditev bomo lahko vključili šele v naslednje (informacijsko ali nadzorno) protokolno sporočilo. Takšno situacijo vidimo na sliki Slika 9-26.



Slika 9-25. Čas do potrditve pri natovarjanju



Slika 9-26. Minimalni čas izteka časovnika

Morebitni večji čas do potrditve pa je treba upoštevati tudi pri določanju širine okna, če želimo doseči čim boljšo učinkovitost protokola.

10 KRMILJENJE PRETOKA

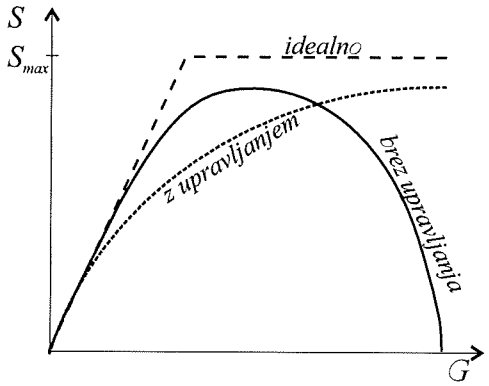
Komunikacijska osebka (protokolni entiteti), ki sta med seboj neposredno povezana, seveda v fizičnem sloju oddajata oziroma sprejemata informacijo z isto hitrostjo. Vendar pa morata oba osebka poleg oddajanja oziroma sprejemanja informacijo tudi obdelovati, kar pa že lahko poteka z različnima hitrostima. Če ti dve hitrosti nista preveč različni med seboj, če ena od njiju ni stalno večja od druge in če prometni pretok med osebkoma ni prevelik, lahko razlike v hitrostih izglajujemo s čakalnima vrstama. Če pa so razlike prevelike in/ali prometni pretok prevelik (vsaj za počasnejšo entiteto), bo dolžina čakalne vrste pri počasnejši entiteti naraščala, dokler ne bo dosegla svoje maksimalne dolžine. Prišlo bo torej do kopičenja informacije pri enem od udeležencev komunikacijskega procesa. Tako kopičenje imenujemo zgostitev prometa ali krajše zgostitev (angleško *congestion*). Prišlo bo do izgub protokolnih sporočil. To seveda ni zaželen pojav. Torej mora imeti vsaka izmed protokolnih entitet možnost vpliva na pretok informacije skozi kanal. Vplivanje protokolnih entitet na pretok informacije imenujemo krmiljenje pretoka informacije (angleško *flow control*). Krmiljenje pretoka je torej nadvse pomemben element telekomunikacijskih protokolov.

Problem krmiljenja pretoka postane še akutnejši v primeru, ko protokolni entiteti nista med seboj povezani neposredno, ampak skozi telekomunikacijsko omrežje, se pravi skozi več vmesnih posredovalnih oziroma komutacijskih naprav. Ker je izvorov prometa v telekomunikacijskem omrežju veliko in so med seboj neodvisni, znotraj omrežja pa se informacijski pretoki naključno kombinirajo (zlivajo in razlivajo), je seveda povsem možno, da je ob nekem času ponujani prometni pretok v nekem delu omrežja večji od maksimalnega možnega opravljenega prometnega pretoka. Občasno lahko torej ne le pri protokolnih entitetah v končnih napravah, ampak tudi pri protokolnih entitetah komutacijskih naprav prihaja do zgostitev prometa.

Pomembno se je zavedati, da se telekomunikacijski sistem brez krmiljenja pretoka informacije začne v primeru zgostitve obnašati nestabilno. Ko namreč entitete s polnimi čakalnimi vrstami začno izgubljati sporočila, se izvorom teh sporočil začno iztekati časovniki. Ker pa izvori ne poznajo vzroka za izgube, v skladu z uporabljeno metodo ARQ izgubljena sporočila ponovno pošiljajo. S tem pa ponujani prometni pretok skozi vozlišče z zgostitvijo še povečujejo, kar ima za posledico še večje izgube... Opravljeni

prometni pretok skozi zgoščeno vozlišče se manjša in lahko v skrajnem primeru celo pade na nič. V tem primeru govorimo o mrtvi točki (*deadlock*).

Poglavitna naloga krmiljenja pretoka je preprečevanje izgub in seveda tudi mrtve točke. Pri pravilni uporabi metod krmiljenja pretoka se torej opravljeni prometni pretok z naraščanjem ponujanega prometnega pretoka ne bo manjšal, ampak bo kvečjemu prešel v nasičenje, ko praktično ne bo več naraščal. V idealnem primeru bi bil opravljeni prometni pretok enak ponujanemu, dokler ne bi dosegel maksimalne vrednosti, pri večjih vrednostih ponujanega pretoka pa bi opravljeni obdržal maksimalno vrednost. Žal pa to v praksi ni mogoče. Davek, ki ga moramo plačati za to, da ne pride do zgostitve, je, da omejimo prometni pretok že pred zgostitvijo. Ko enkrat do zgostitve pride, je že prepozno. Zato bo opravljeni prometni pretok v sistemu s krmiljenjem pretoka pri nekaterih vrednostih ponujanega pretoka manjši, kot bi bil, če pretoka ne bi krmilili. Razmere kaže naslednja slika.



Krmilimo lahko bodisi prometni pretok med dvema končnima komunikacijskima subjektoma ali pa med dvema sosednjima komunikacijskima napravama. Medtem ko prvo vrsto krmiljenja uporabljamo predvsem v transportnem sloju protokolnega sklada, nam drugo vrsto omogočajo protokoli kanalnega, pa tudi omrežnega sloja. Krmiljenje pretoka med končnima napravama ščiti končno napravo pred tem, da bi jo njen partner "zasul" s prevelikim prometnim pretokom, ki ga ne bi zmogla. Krmiljenje pretoka skozi kanale med sosednjimi napravami v omrežju pa ščiti te naprave pred zgostitvami, ki pa bi posledično tudi vodile do izgub. V takem primeru bi se zgostitev širila od mesta nastanka navzven. Če krmiljenje pretoka uporabimo na tak način, se bo tudi učinek tega krmiljenja širil navzven in končno dosegel tudi končni entiteti.

Krmiljenje hitrosti prenosa med dvema končnima napravama imenujemo **krmiljenje pretoka** v ožjem pomenu besede. Krmiljenje hitrosti prenosa skozi vmesne posredovalne naprave pa imenujemo **krmiljenje zamašitev**.

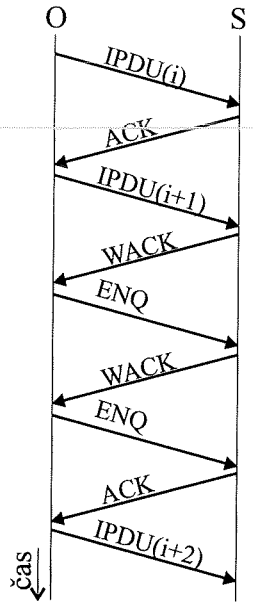
Metode krmiljenja pretoka v osnovi delimo na eksplicitne in implicitne.

10.1 Eksplicitne metode krmiljenja pretoka

Pri eksplicitnih metodah krmiljenja pretoka mora osebek, ki se mu čakalna vrsta polni, o tem eksplicitno obvestiti partnersko entiteto; ta preneha z oddajanjem, dokler za to ne dobi ponovnega dovoljenja.

Eno od najpreprostejših metod eksplicitnega krmiljenja pretoka so uporabljali pri asinhronem znakovno orientiranem prenosu (start-stop prenosu); to metodo bi lahko imenovali *XON-XOFF metoda.* V primeru, ko sprejemnik ni več mogel sprejemati podatkov, je poslal oddajniku tako imenovani XOFF znak - to je ASCII znak DC2. Ko pa je bil spet pripravljen na sprejem, je to sporočil oddajniku z znakom XON - to je ASCII znak DC 1. Na tipkovnici računalnika lahko uporabnik preprosto generira ta dva znaka s hkratnim pritiskom Ctrl-S (XOFF) oziroma Ctrl-Q (XON).

Ob uporabi ARQ metode s čakanjem je lahko krmiljenje pretoka zelo preprosto; sprejemnik pač neha potrjevati prispela sporočila, vse dokler se toliko ne sprosti, da lahko spet sprejema. Da pa se ne bi medtem oddajniku iztekel časovnik, sprejemnik eksplicitno sporoči oddajniku, da ne more več sprejemati, in sicer s sporočilom, ki ga bomo tu preprosto označili z oznako WACK (Wait Acknowledge). Ko oddajnik sprejme sporočilo WACK, preneha oddajati informacijska protokolna sporočila in oddaja le poizvedovalna sporočila ENQ, vse dokler od sprejemnika ne prejme potrditvenega sporočila ACK. Nato nadaljuje z oddajanjem informacijskih protokolnih sporočil. Značilen potek takega nadzora kaže naslednja slika.



10.2 Implicitne metode krmiljenja pretoka

Implicitne metode krmiljenja pretoka so tiste, katerih osnovni namen ni le ta, ampak hkrati opravljajo še druge naloge. Najpogosteje protokoli kombinirajo metodo krmiljenja pretoka z metodo popravljanja napak ARQ.

10.2.1 Pošiljanje potrditev in metoda drsečega okna

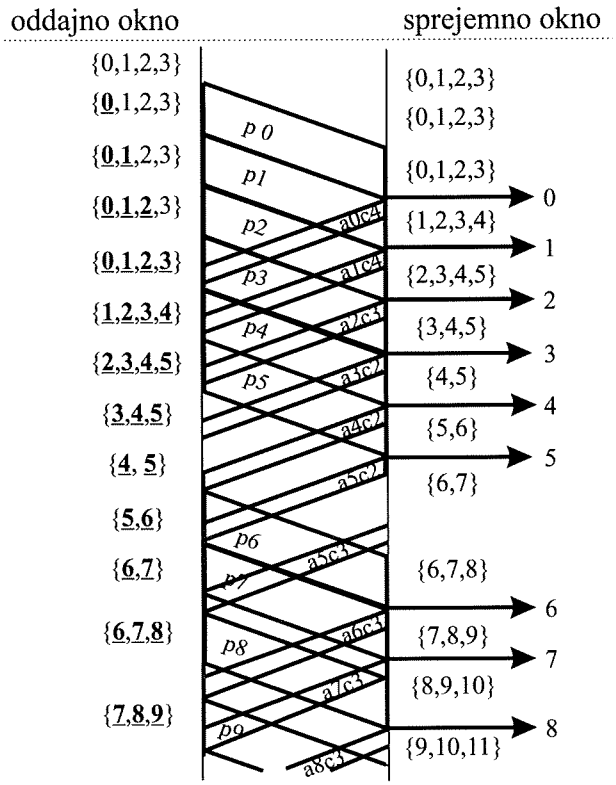
Najpreprostejša implicitna metoda krmiljenja pretoka je pri protokolih ARQ seveda ta, da sprejemnik zavlačuje s potrditvami, s čimer pa zavira tudi drsenje oddajnega okna in s tem oddajanje oddajnika. Seveda je pri tem treba paziti na to, da tako ravnanje sprejemnika ne povzroča izteka oddajnikovega časovnika in s tem ponovnega pošiljanja sicer že uspešno sprejetih sporočil. Zato protokoli pogosto uporabljajo kombinacijo metode drsečega okna in eksplicitnih metod krmiljenja pretoka.

10.2.2 Metoda drsečega okna s spremenljivo širino okna

Enega izmed bolj sofisticiranih načinov krmiljenja pretoka uporabljajo protokoli drsečega okna, pri katerih ima sprejemna entiteta med samim delovanjem vpliv na širino okna. Gre za kombinacijo eksplicitnega in implicitnega krmiljenja. Širina okna se torej dinamično spreminja. Tovrstni protokoli uporabljajo tako imenovani kreditni mehanizem. Sprejemnik poleg potrditev pošilja oddajniku tudi informacijo o tem, koliko sporočil je po potrjenem sporočilu še pripravljen sprejeti ("kredit"). Če npr. pošlje sprejemnik

oddajniku informacijo *ACK n CRD m*, to pomeni, da potrjuje sporočilo *n*, hkrati pa sporoča, da je pripravljen sprejeti še sporočila *n*+1, *n*+2, ..., *n*+*m*. S tem pa dejansko postavi zgornjo mejo oddajnikovega okna. Na ta način ločimo potrjevanje od upravljanja pretoka! Če pa pri tem sprejemnik prehitro zmanjša širino okna, se lahko zgodi, da je oddajnik že pred sprejemom novega kredita oddal sporočilo, ki po novem ni več v oknu. Zato je bolje, da sprejemnik širino okna zmanjšuje postopoma.

Naslednja slika prikazuje primer krmiljenja pretoka z drsečim oknom spremenljive širine. Začetna širina oddajnega in sprejemnega okna je 4. Po sprejemu sporočila št. 2 sprejemnik ugotovi, da sprejetih sporočil ne more sproti obdelovati in širino okna postopoma zmanjša na 2, nekoliko kasneje pa jo poveča na 3. V sliki lahko lepo opazujemo, kako se pri tem spreminja efektivna hitrost oddajanja sporočil!



Protokole s spremenljivo širino okna uporabljamo zlasti v višjih slojih protokolnega sklada, kjer komunikacijski subjekti vzdržujejo hkrati po več navideznih zvez z različnimi partnerji. Če so pri tem sporočila dolga, kanali relativno počasni in propagacijski časi dolgi, s tem pa tudi okna široka, je potrebno imeti za vzdrževanje vmesnikov na oddajni ali sprejemni strani kar precej spomina na razpolago. V takih primerih lahko postaja dinamično razporeja razpoložljivi spomin med različne navidezne

zveze in s tem lovi ravnotežje med večjo učinkovitostjo protokola pri širšem oknu in manjšo potrebo po spominu. Če ji zmanjkuje razpoložljivega spomina, bo (tudi za ceno manjše učinkovitosti protokola) širino okna zmanjšala, če ima prostega spomina dovolj, pa bo širino okna povečala in s tem povečala tudi učinkovitost. Znan protokol, ki uporablja kreditni mehanizem, je protokol TCP.

11 MRTVE DUŠE

V omrežju se lahko zgodi, da se neko sporočilo zaradi okvare vozlišča ali zaradi zgostitve prometnega pretoka v nekaterih delih omrežja nenormalno dolgo zakasni. Tisti protokolni entiteti, ki je sporočilo oddala, se je že iztekel časovnik in je sporočilo ponovno poslala, sprejemna protokolna entiteta pa je novo sporočilo sprejela. Potem pa ti, ne bodi ga treba, pride na cilj še staro sporočilo, ki sta ga entiteti tako rekoč že proglasili za mrtvo. Sprejemna entiteta seveda ne ve, kaj se je v resnici zgodilo, zato lahko takšno sporočilo sprejme, kar pa povzroči pravo zmedo in napačno delovanje protokola. Takšnim sporočilom bomo rekli mrtve duše (angleško *floating corpses*). Pojav mrtvih duš je še zlasti nevaren pri prenosu sporočil, ki niso oštevilčena (npr. sporočila za upravljanje zveze ali datagramska sporočila), ali pa v primeru, če je prišla sekvenčna številka "mrtve duše" ponovno v uporabo.

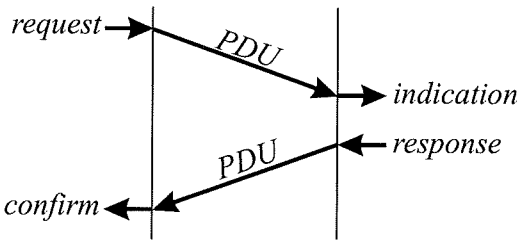
Najenostavnejši način zaščite pred mrtvimi dušami so časovni parametri, ki jih dodamo sporočilom in predstavljajo življenjsko dobo sporočila (angleško *time-to-live*). Ko sporočilo čaka v čakalni vrsti vozlišča v omrežju, mu procesor tega vozlišča občasno dekrementira življenjsko dobo. Če življenjska doba doseže vrednost 0, preden sporočilo doseže svoj cilj, ga omrežje šteje za izgubljenega in zavrže. Iz različnih vzrokov je življenjsko dobo sporočila težko natančno meriti in spremljati. Zato namesto prave življenjske dobe pogosto uporabljajo število posredovalnih naprav, skozi katere sme preiti sporočilo. Vsaka komutacijska naprava (npr. usmerjevalnik) to število zmanjša za ena in sporočilo zavrže, če je to število doseglo vrednost 0.

12 UPRAVLJANJE ZVEZE

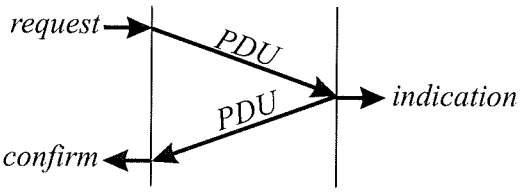
Pogosto se morata dve protokolni entiteti (ali več protokolnih entitet) medsebojno dogovoriti o stanju komunikacijskega sistema oziroma to stanje uskladiti. Med tovrstne naloge spadajo vzpostavljanje navidezne zveze, pogajanja o parametrih zveze, sproščanje zveze, spremembe načina komuniciranja ipd. V naslednjih odstavkih si bomo ogledali nekaj tehnik, ki jih protokoli uporabljajo v te namene.

12.1 Dvojni dogovor (two-way handshake)

Najobičajnejši način vzpostavljanja ali sproščanja zveze in podobnih usklajevalnih postopkov med dvema protokolnima entitetama v ne preveč zahtevnih okoljih je dvojni dogovor, ki temelji na medsebojni izmenjavi dveh nadzornih protokolnih sporočil. Tu gre vedno za storitev s potrditvijo, kar pomeni, da dobi tisti uporabnik, ki je pobudnik storitve, o opravljeni storitvi potrdilo; z drugimi besedami pa to pomeni, da se postopek zaključi s primitivom *confirm*. Kljub temu pa vseeno ločimo dve varianti: pri prvi ima tisti uporabnik, ki je prejemnik storitve, možnost storitev sprejeti ali ne (to mu omogoča primitiv *response*). To je storitev s potrditvijo uporabnika (angl. *user-confirmed service*), postopek pa kaže naslednja slika.



Drugo možnost pa predstavlja storitev s potrditvijo izvajalca (angl. *provider-confirmed service*), kjer tisti uporabnik, ki je prejemnik storitve, nima možnosti, da bi storitev sprejel ali ne; v tem primeru o tem odloči izvajalec storitve. Ta možnost je prikazana na naslednji sliki.

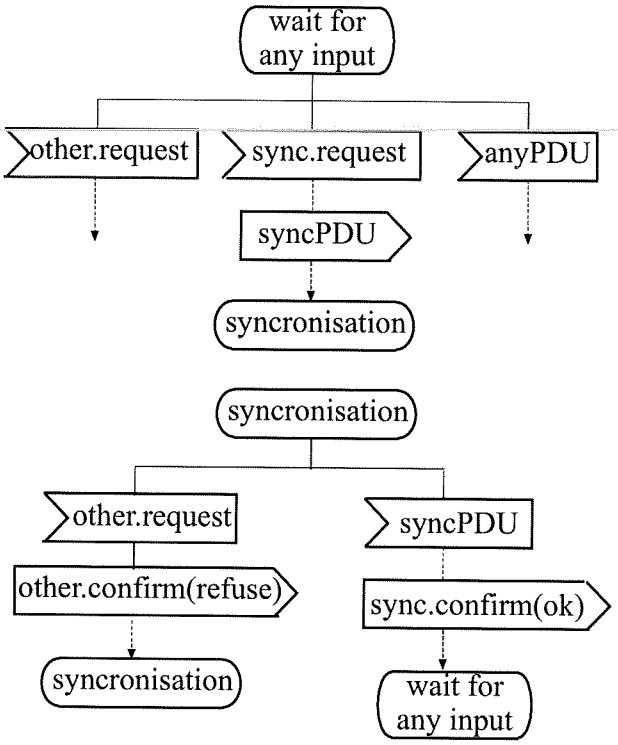


V obeh primerih nosi prvo protokolno sporočilo (od pobudnika k prejemniku) zahtevo po vzpostavitvi zveze, sproščanju zveze, spremembi parametrov ipd., drugo (v obratni

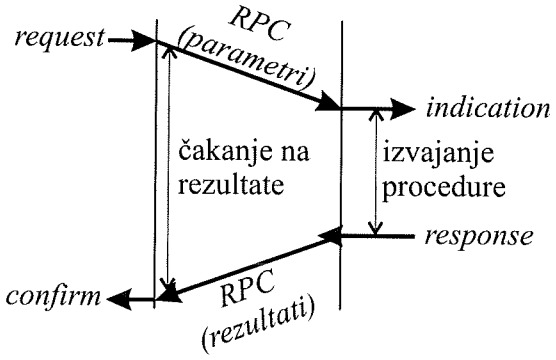
smeri) pa potrditev prvega sporočila. Protokolna entiteta na strani pobudnika storitve mora seveda uporabiti tudi časovnik in PDU ponovno poslati, če se časovnik izteče.

Največkrat je pobudnik takega postopka ena izmed protokolnih entitet na zahtevo svojega uporabnika. Seveda pa se pri simetričnih protokolih (takih, kjer sta obe protokolni entiteti enakopravni) lahko zgodi tudi, da oba partnerja skoraj hkrati začneta zgoraj opisani postopek. Protokoli morajo predvideti tudi take kolizije in specificirati postopek, ki vodi do tega, da se postopek tudi v takšni situaciji uspešno konča, ali pa se začne še enkrat na novo odvijati.

Če gre pri tem postopku za vzpostavljanje zveze med entitetama, največkrat ni pričakovati, da bi med samim postopkom entiteti dobivali še kakšne druge zahteve, saj le-te tudi niso smiselne, dokler zveza ni vzpostavljena. So pa tudi primeri, ko med samim postopkom dogovarjanja lahko pride tudi do drugih zahtev, njihovo izpolnjevanje pa bi postopek dogovarjanja zmotilo. Značilen tak primer je postavitev sinhronizacijske točke, ki predstavlja definirano stanje komunikacijskega procesa v konferenčnem sloju OSI referenčnega modela. V takem primeru bo entiteta na strani pobudnika vse zahteve, ki prispejo med zahtevama *request* in *confirm*, in vse medtem prispele PDU-je razen pričakovanega zavrnila ali pustila čakati v čakalni vrsti; podobno bo ravnala tudi entiteta na strani prejemnika. Tak postopek imenujemo elementarna uskladitev (angl. *atomic exchange*). Na naslednji sliki je prikazana SDL specifikacija delovanja protokolne entitete na strani pobudnika, ki med postopkom usklajevanja vse zahteve zavrne in vse PDU-je razen pričakovanega zavrže.



Postopek elementarne uskladitve se pogosto uporablja tudi pri izvajanju operacij na oddaljenem računalniku (remote procedure call, RPC). V tem primeru protokolna entiteta odjemalca zahteva izvajanje procedure na strežniku, nato pa čaka na rezultate te procedure in medtem ne sprejema drugih zahtev. Potek prikazuje naslednja slika.



12.2 Trojni dogovor

Postopek dvojnega dogovora lahko da čudne rezultate, če se v sistemu pojavijo mrtve duše. V primeru, ko pričakujemo takšne težave, to pa je največkrat v sloju protokolnega sklada, pod katerim uporabljamo nepovezavno orientirane protokole, uporabimo metodo trojnega dogovora (angl. *three-way handshake*). Pri tem postopku si obe entiteti izmenjata tri protokolna sporočila; označita jih s posebnimi oznakami, ki se morajo ujemati. Dogovor je videti nekako tako, kot vidimo v časovni shemi na naslednji sliki. Pri

tem protokolni entiteti izbereta vsaka svojo identifikacijsko oznako (*a* oziroma *b*) za vsak dogovor posebej. Če se identifikacija *a* v sporočilu *accept* oziroma identifikaciji *a* ali *b* v sporočilu *check* ne ujemajo s pričakovanimi, bo entiteta, ki napačno identifikacijo sprejme, prenehala z dogovarjanjem, drugi entiteti pa se potem izteče časovnik in prav tako odstopi od dogovora.