

# Standardizacija plasti

Andrej Bagon  
RSM  
2005/06

# Standardizacija plasti

- Poglavje podaja pregled mehanizmov horizontalnega in vertikalnega povezovanja med plastmi informacijsko komunikacijskega sistema. Splošne lastnosti plasti bomo najprej nazorno predstavili na primeru odjemalec-strežnik, na koncu poglavja pa navedli nekaj osnovnih informacij o postopkih standardizacije, ki v marsičem pogojujejo uspešnost sistema. Predstavili bomo dve najvažnejši arhitekturi in ju med seboj tudi primerjali.

# Standardizacija plasti

- V dosedanjih poglavjih smo nakazali, da je sistem heterogen in večplasten. S tem je povezana tudi zahtevnost in kompleksnost njegove izvedbe. Izvedba postane lažje obvladljiva in bolj razumljiva, če arhitekturo sistema obdelujemo postopoma, plast za plastjo. Poleg tega je cela vrsta značilnosti, ki so lastne vsaki plasti in te si bomo ogledali v tem poglavju.

# Standardizacija plasti

- Začeli bomo z nekoliko presenetljivim odstavkom, v katerem se bomo lotili arhitekture principov odjemalec-strežnik. Izkazalo se bo, da je to, pred formalno obravnavo plasti, še kako upravičeno, saj mehanizem odjemalec-strežnik zelo dobro ponazarja delovanje plasti in interakcijo med njimi.

# Mehanizem odjemalec-strežnik

- Funkcioniranje plasti bomo v naslednjih poglavjih opisali tudi s formalnega vidika. Za začetek pa njihovo delovanje lahko prav dobro ponazorimo z opisom mehanizma odjemalec-strežnik. Bistvo takega sistema je, da imamo proces, ki želi določeno storitev – to je **odjemalec** (klient) in proces, ki je sposoben izvesti zahtevani posel in odjemalcu posredovati odgovor – to je **strežnik**. Par zahteva-odgovor bomo poimenovali **transakcija**.

# Mehanizem odjemalec-strežnik

- Rezultat transakcije oziroma transakcija je lahko uspešna, kar pomeni, da je odjemalec dobil "odgovor" na "vprašanje", ali pa neuspešna, če odjemalec od strežnika ne dobi odgovora oziroma je odgovor negativen – na primer "tvoje zahteve ne morem izpolniti". V tem sistemu je bistven parameter **zakasnitev** oziroma **čas izvajanja transakcije**, ki je opredeljen s trenutkom, ko se je zahteva posredovala sistemu, in trenutkom, ko dobimo pozitiven ali negativen odgovor. Včasih mehanizem odjemalec-strežnik imenujemo tudi **transakcijski sistem**.

# Mehanizem odjemalec-strežnik

- Najenostavnejši primer mehanizma odjemalec-strežnik je vsem dobro znani primer razmerij med glavnim programom kot odjemalcem in podprogramom kot strežnikom. Večina jih je sposobna napisati programček, ki prebere število in v podprogramu izračuna njegovo kvadratno vrednost.

# Mehanizem odjemalec-strežnik

- V tem primeru se trenutni parametri precesorja shranijo na sklad, izvede se podprogram, nato pa se stanje pred klicem restavrira s sklada, glavni program pa dobi odgovor na vprašanje "kakšna je vrednost kvadratne vrednosti?". V običajnem primeru niti ne pomislimo, da je v ozadju opisani postopek, saj za vse operacije klika in predajanja rezultatov v veliki meri s svojo kontrolo poskrbita operacijski sistem in prevajalnik, ki prevede naš programček v strojno kodo.



# Mehanizem odjemalec-strežnik

- ***Problem***, ki ga nameravamo obravnavati je sledeč: Kako realizirati aplikacijo, ko je glavni program v enem, podprogram pa v drugem računalniku?

# Mehanizem odjemalec-strežnik

- Povedano z drugimi besedami: predstavljajmo si, da imamo v omrežju na razpolago dva računalnika, napisati pa moramo program, ki bo rešil problem izračuna kvadratne vrednosti, seveda pa bo moral poskrbeti tudi za dodatek, ki ga v klasični izvedbi ni. Kako poskrbeti za klic podprograma in prevzem rešitve med dvema omrežno povezanima računalnikoma?

# Mehanizem odjemalec-strežnik

- Očitno je problem na uporabniškem nivoju trivialen, izračun kvadratne vrednosti, medtem ko na sistemskem nivoju preseda sposobnosti marsikaterega programerja. Začnimo z obravnavanjem stičnih točk med arhitekturo sistema in opisanim primerom oziroma z mehanizmom odjemalec-strežnik.
- V prejšnjih poglavjih smo že večkrat poudarili, da **višja plast** ( $N+1$ ) – **odjemalec** zahteva storitev **nižje plasti** ( $N$ ) – **strežnika**.

# Mehanizem odjemalec-strežnik

- Po uspeli ali neuspeli izvedbi zahteve strežnik odgovori s tako imenovano potrditvijo, ki lahko vsebuje odgovor na zahtevo, včasih pa na primer sporočilo, da se strežnik z nami nima časa ukvarjati. Vidimo lahko, da ima opisani mehanizem identično sekvenco dogodkov, kakor smo jo opisali pri izračunu kvadratne vrednosti. Ta mehanizem ustreza **izvedbi transakcije** na osnovi "dialoga" med odjemalcem in strežnikom.

# Mehanizem odjemalec-strežnik

- V nekoliko bolj računalniški terminologiji bi rekli, da sta odjemalec in strežnik procesa, ki v enostavnejšem primeru "tečeta" v istem računalniku, v sistemsko zahtevnejši različici pa v dveh različnih računalnikih – seveda pa je vsebinski, uporabnikov problem enak. Ta v enem ali drugem primeru pričakuje kvadrato vrednosti. Uporabili smo pojem proces, nismo pa še pojasnili, kaj to je. Proces je običajno posledica izvajanja določenih ukazov. Zanj je značilno, da med izvajanjem ukazov zasega določene računalniške vire (procesor, pomnilnik, periferne enote,...). 13

# Mehanizem odjemalec-strežnik

- **Proces** je množica računalniških virov, ki sodelujejo pri izvajanju določene zahteve.
- To, da včasih namesto pojma proces uporabljamo tudi pojem procedura, aplikacija... je zgolj posledica dejstva, da večina procesov res nastane na osnovi izvajanja računalniškega programa, strogo gledano pa teh pojmov seveda ne bi smeli enačiti.

# Mehanizem odjemalec-strežnik

- Kakor smo že povedali pri opisu primera, je s stališča programiranja izvedba lokalne transakcije podobna programskemu klicu procedure s parametri, saj podprogram po končanem izvajanju glavnemu programu vrne določene podatke – rezultat. Če se klicana procedura izvaja v drugem računalniku, gre za **klic oddaljene procedure**, angleško Remote Procedure Call, kar je v bistvu isto kot pravkar opisani primer sodelovanja procesov v dveh računalnikih.

# Mehanizem odjemalec-strežnik

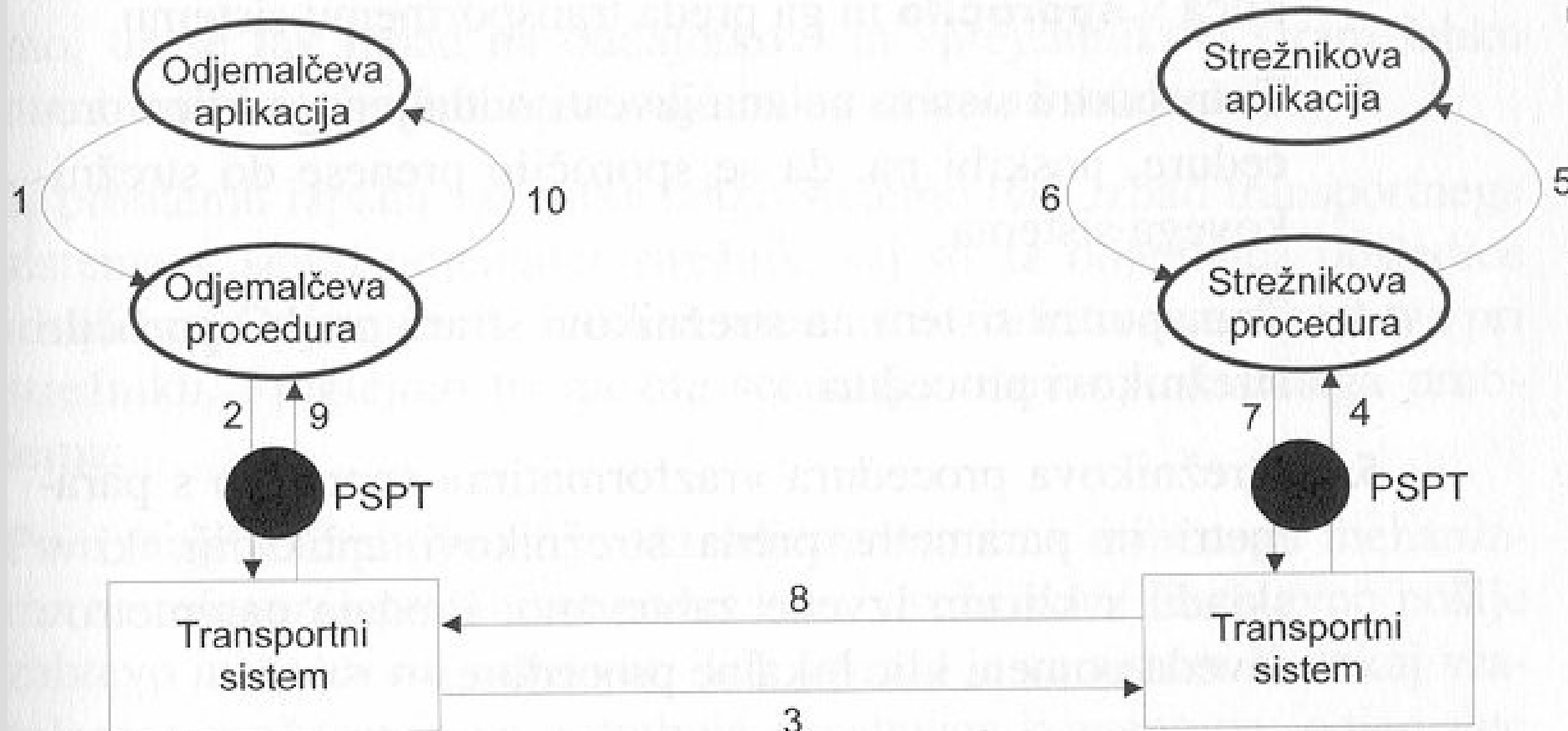
- Ker bomo predvsem na informacijski plasti velikokrat uporabljali namesto pojma proces kar pojma procedura ali aplikacija, postavimo še eno definicijo:
- ***Oddaljena procedura, aplikacija je tista, ki se ne izvaja v računalniku, ki je sprožil zahtevo.***



# Mehanizem odjemalec-strežnik

- Analogijo med plastnim modelom sistema in mehanizmom odjemalec-strežnik najdemo prav v izvedbi klica oddaljene procedure. Osnovna ideja mehanizma odjemalec-strežnik je na uporabniškem nivoju poenotiti sintakso klica oddaljene procedure s klicanjem lokalne. V nadaljevanju bomo velikokrat rekli, da smo na nivoju pisanja uporabniških programov zagotovili **transparentnost** klica procedure glede na to, kje se izvaja. Poglejmo si sedaj, kako lahko problem rešimo.

# Mehanizem odjemalec-strežnik



*Slika 5-1: Komunikacija med odjemalcem in strežnikom*

# Mehanizem odjemalec-strežnik

- Slika prikazuje korake, ki so potrebni za dialog zahteva-odgovor. Iz opisa sekvence dogodkov bomo videli, da je namen **odjemalčeve procedure** omogočiti, da odjemalec zahteva storitev od oddaljenega strežnika enako, v enaki sintaksi, kakor če bi bil strežnik lokalni. Pravimo, da je **izvedba klica transparentna**. Nekaj podobnega velja tudi za **strežnikovo proceduro**, ki gre za vračanje rezultatov k odjemalcu. Oboroženi z vsemi temi definicijami lahko razumno opišemo sekvenco posameznih korakov mehanizma odjemalec-strežnik:

# Mehanizem odjemalec-strežnik

1. Odjemalčeva aplikacija pokliče odjemalčevo proceduro – storitev. Parametri se med njima prenesejo lokalno na način, ki ga podpira odjemalčev sistem. Odjemalčeva aplikacija pri tem ne zazna nič nenavadnega, saj gre za klic lokalne procedure.
2. Odjemalčeva procedure preslika parametre lokalnega klica v **sporočilo** in ga preda transportnemu sistemu.
3. Transportni sistem ne zna izvesti oddaljenega klica procedure, poskrbi pa, da se sporočilo prenese do strežnikovega sistema.

# Mehanizem odjemalec-strežnik

4. Transportni sistem na strežnikovi strani preda sporočilo strežnikovi proceduri.
5. Strežnikova procedura "razformatira" sporočilo s parametri in parametre preda strežnikovi aplikaciji, ki v skladu s klicem izvede zahtevano. Predaja parametrov seveda pomeni klic lokalne procedure.
6. Ko strežnik izvede zahtevo, se rezultat vrne na enak način odjemalcu v korakih 6,7,8,9 in 10, ki vsebinsko ustrezajo korakom 1,2,3,4 in 5.

# Mehanizem odjemalec-strežnik

- Poleg zaporedja klicev in prenosa sporočil je pri klicu oddaljene procedure zelo pomembna tudi združljivost podatkovnih struktur, ki se uporabljajo na strežnikovi in odjemalčevi strani. Tudi te težave morata premostiti odjemalčeva in strežnikova procedura. Morebitne težave z združljivostjo (predstavitvijo) podatkov so pomemben problem, za katerega bomo v nadaljevanju obravnavanja arhitekture sistema rezervirali posebno plast, ki jo bomo poimenovali predstavitvena plast.

# Mehanizem odjemalec-strežnik

- Če sistem deluje regularno, zadostujejo do sedaj opisani mehanizmi. Lahko pa nastopijo izredne situacije, ki pa jih v računalništvu dokaj redno srečujemo. V našem primeru gre za sledeče težave:
  - izpad strežnika – odjemalec s svojo zahtevo ne more doseči strežnika in
  - izpad odjemalca – strežnik po izvedenem poslu ne more doseči odjemalca.

# Mehanizem odjemalec-strežnik

- Izguba sporočila pomeni, da transportni sistem svojega dela ni opravil, kakor se je od njega zahtevalo, vendar kaj hitro lahko ugotovimo, da se tak izpad na oddajnikovi in sprejemnikovi strani lahko manifestira (predstavlja) enako kot izpad odjemalca oziroma strežnika.



# Mehanizem odjemalec-strežnik

- K problemu izpada strežnika lahko štejemo tudi izpad transportnega sistema v smeri odjemalec-strežnik, saj so za odjemalca posledice identične. Odjemalec skratka ne more aktivirati izvajanja storitve pri strežniku. Poglejmo tri možne scenarije reševanja naslalega problema:
  - največ enkrat/send exactly once
  - vsaj enkrat/at least once
  - zadnji od mnogih/last of many

# Mehanizem odjemalec-strežnik

- Prva, najslabša možnost je, da odjemalec nima nobenega mehanizma za reševanje prekinitve povezave s strežnikom. Enostavno pošlje zahtevo in **čaka na odgovor**. Tak pristop je za sistem dokaj vratolomna možnost in ne potrebuje posebnega komentarja: odjemalec čaka odgovor, ki ne bo nikoli prišel. Sočasno pa odjemalec seveda ne more vedeti, ali je strežnik njegovo zahtevo sploh sprejel ali pa je izpadel med njenim izvajanjem.

# Mehanizem odjemalec-strežnik

- Strežnik se po eventualni ponovni vzpostavitvi seveda ne "zaveda" zahtev, ki jih je prejel pred izpadom, in je čisto zadovoljen s tem, da mu ni treba ničesar opraviti. Odjemalec pa je obsojen, da čaka na odgovor "neskončno" dolgo. Edino, kar ga lahko reši iz blokade, je vsem dobro znani "CTRL+ALT+DEL" ali vnovična nastavitev ("resetiranje") odjemalca. Angleški izraz za tako "strategijo" reševanja problema je **send exactly once/največ enkrat** – pošlji zahtevo natanko enkrat, mi pa dodajamo: ... in izgubi živce.

# Mehanizem odjemalec-strežnik

- Izboljšava je, če odjemalec ob oddaji zahteve sproži **časovno kontrolo**. Če do izteka časovne kontrole od strežnika ne dobi odziva, lahko sproži proceduro za razreševanje nastale situacije. V praksi to pomeni, da nam ne bo treba ročno vnovič nastaviti ("resetirati") odjemalčevega računalnika oziroma aplikacije. Pogosta praktična rešitev nastale situacije je, da odjemalec prevzame kontrolo nad zvezo, s čimer postane "gospodar" zveze.

# Mehanizem odjemalec-strežnik

- Po izteku časovne kontrole predpostavi, da je strežnik izpadel ali pa je nedosegljiv, zato z ustreznimi obvestili ustavi dialog med uporabnikom in sistemom. Taki rešitvi bi lahko rekli tudi strategija **gospodar – suženj** – "master – slave" (reče pa se ji **vsaj enkrat**).

# Mehanizem odjemalec-strežnik

- Tretja možnost, ki je fleksibilnejša različica predhodnega primera, je, da odjemalčeva procedura po izteku časovne kontrole **samodejno ponovi klic** oziroma ponovi zahtevo. S tem postane sistem trdoživnejši, pojavi pa se seveda problem, kako reševati situacije, ki so posledica večkratne iste zahteve, na katero odjemalec seveda lahko dobi več odgovorov, tudi različnih. Situacija je rešljiva, če je odjemalec sposoben **identificirati zadnji (last of many/zadnji od mnogih)** odgovor.

# Mehanizem odjemalec-strežnik

- Primer, ki nazorno pojasnjuje težavo, je poizvedba, "ali je na letu tem in tem še kaj prostih mest?". Načelno ni jasno, ali imamo nič, eno ali dve prosti mesti. Če pa odjemalec lahko sprejete sekvence odgovorov, ki ni nujno ista kot na strežnikovi strani, identificira zadnjega, je problem rešen. Potniku ne preostane drugega kot vpis v čakalno listo, sicer pa bo let prezaseden.

# Mehanizem odjemalec-strežnik

- Poglejmo si sedaj še težave, ki so značilne za izpad odjemalca. Strežnik, ki je ostal brez odjemalca, ker je ta nehal delovati ali pa je nedosegljiv, ima aktiviran proces "strežnikova aplikacija", ki pa ne more vzpostaviti zveze z odjemalčevo aplikacijo. Proces je postal **sirota** (orphan), saj je iniciator povezave, roditelj, "umrl" ali pa se ne briga več za svoje "otroke". Tak proces strežniku požira sistemske vire, kar pa še ni najhuje.



# Mehanizem odjemalec-strežnik

- Več težav lahko povzroči zmeda, ki nastopi v vrstah odjemalcev, ko se ponovno vzpostavijo in dobivajo od strežnika odgovore na svoje zahteve iz časa pred izpadom. Teh zahtev se odjemalec namreč ne "spominja" in zato seveda ne ve, kako naj interpretira odgovore. Za razreševanje takšnih situacij poznamo kar nekaj scenarijev:
  - iztrebljanje
  - omejen čas
  - reinkarnacija
  - nežno iztrebljanje

# Mehanizem odjemalec-strežnik

- **Iztrebljanje** je drakonska(“kruta”) strategija: takoj ko se odjemalec – roditelj procesa ponovno zave samega sebe, strežniku naroči, naj ubije vse njegove morebitne procese (extermination) – otroke. Algoritem mora teči rekurzivno, ker lahko najdemo v primeru pogostih izpadov na strežniku tudi procese vnuke ali celo pravnuke posameznih zahtev.

# Mehanizem odjemalec-strežnik

- Druga strategija vsaki zahtevi določi **čas za izvedbo**. Če se zahteva v tem času ne izvrši, strežnik prosi odjemalca za nov časovni interval. Če ga ne dobi, bo seveda ustavil izvajanje. S to strategijo zagotovimo strežnikovo aktivno preverjanje statusa odjemalca.

# Mehanizem odjemalec-strežnik

- Tretja strategija se imenuje **reinkarnacija**. Odjemalec razseka čas v zaporedne intervale, imenovane **epohe**. Če odjemalcu po metodi iztrebljanja ni uspelo ubiti vseh svojih procesov otrok, lahko oznani vsem strežnikom začetek nove epohe. Strežniki pa uničijo vse tiste zahteve, ki ne sodijo v sedanjo epoho.

# Mehanizem odjemalec-strežnik

- Opozoriti je potrebno, da je ubijanje procesov lahko delikatno, saj lahko povzroči različne nekonsistentnosti. Četrta strategija to upošteva in se imenuje **nežno iztrebljanje**. Podobna je prejšnji, le da pred ubijanjem procesov strežnik odjemalca na to opozori (zahteva potrditev), odjemalec pa dobi možnost, da se odloči o usodi svojih prejšnjih zahtev – razveljavljanje ali potrjevanje transakcij (procedure za roll-back in commit).

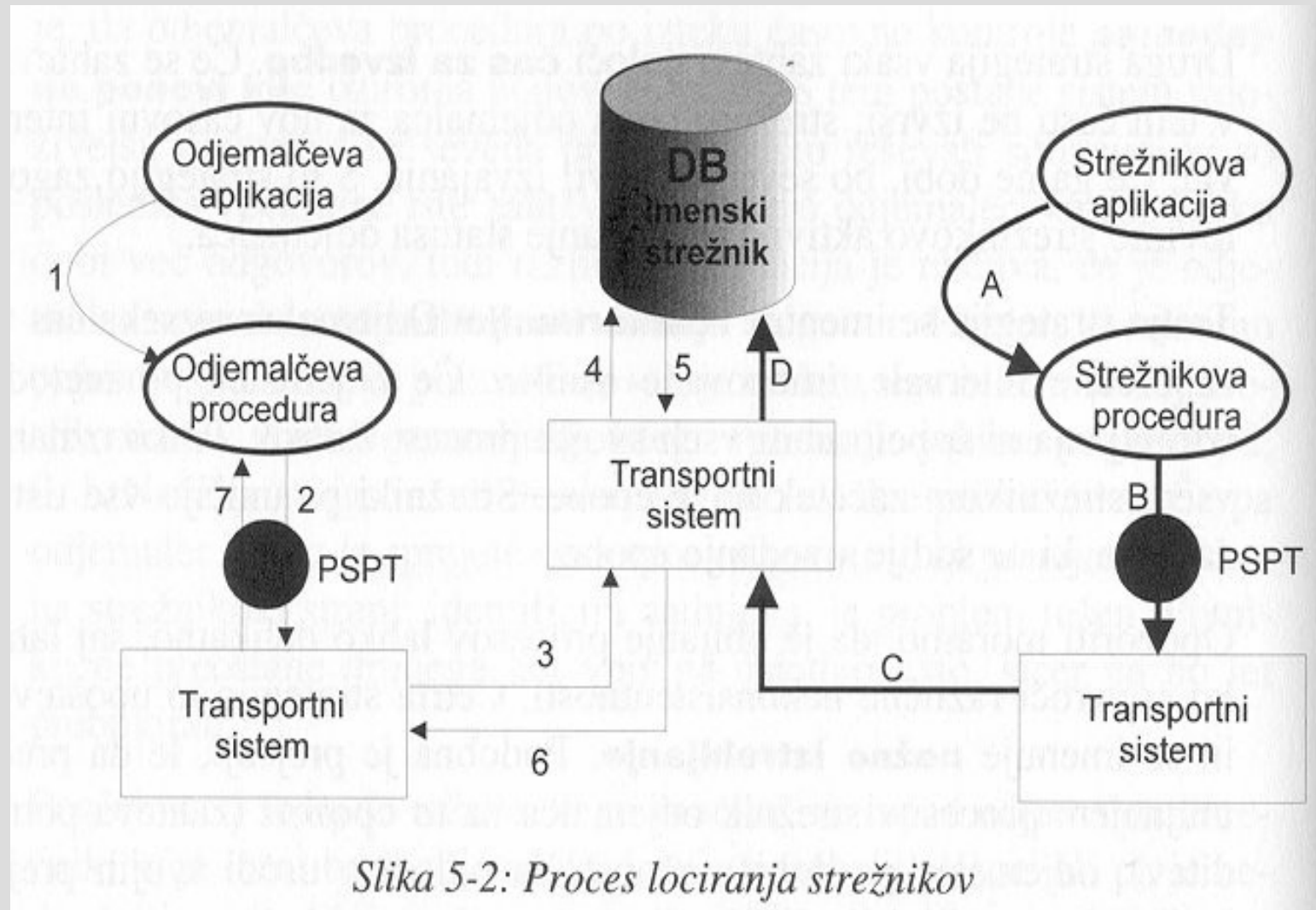
# Mehanizem odjemalec-strežnik

- Izpad strežnika
  - največ enkrat
  - vsaj enkrat
  - zadnji od mnogih
- Izpad odjemalca
  - iztrebljanje
  - omejen čas
  - reinkarnacija
  - nežno iztrebljanje

# Mehanizem odjemalec-strežnik

- Omenimo še odjemalčev problem lociranja strežnikov, kar pomeni, da mora odjemalec "vedeti", na katerega od dostopnih strežnikov naj naslovi določeno zahtevo. Imenski strežnik je tisti, ki vodi evidenco o vseh drugih strežnikih. Odjemalec se s svojim problemom obrne na imenski strežnik in ta mu predlaga izvajalca. Ob vzpostavitvi novega strežnika je treba o njegovem obstoju obvestiti imenski strežnik. Za to lahko poskrbi novi strežnik sam (avtomatsko), največkrat pa je potreben poseg vzdrževalca imenskega strežnika.

# Mehanizem odjemalec-strežnik





# Mehanizem odjemalec-strežnik

- Prijavljanje novega strežnika na sliki nam prikazuje sekvence A, B, C, D, s katero se novi strežnik prijavi imenskemu strežniku. Ko pa odjemalec želi ugotoviti, kateri strežnik je najbolj primeren za izvršitev njegove zahteve, ta podatek dobi od imenskega strežnika. Postopek ponazarja sekvenca korakov od 1 do 7 na sliki.

# Mehanizem odjemalec-strežnik

- V naslednjih odstavkih bomo prenesli ugotovitve, ki veljajo za mehanizem odjemalec-strežnik, neposredno v okvir arhitekture sistema. Šlo bo predvsem za formalno opredeljevanje zasnove in terminologije, ki je značilna za vsako plast, ne glede na njeno funkcionalnost.

# Splošne lastnosti plasti

- Ne glede na število plasti, ki sestavljajo celovit informacijsko komunikacijski sistem, veljajo določene splošne lastnosti za vsako plast določene arhitekture. Večino posebnosti in lastnosti plasti smo do dosedanja obravnavi sistema že omenili, v tem poglavju pa jih bomo predvsem sistematizirali in določili terminologijo, ki jo predpisuje standardizacija.

# Splošne lastnosti plasti

- Najprej ne smemo pozabiti, da je razlog in osnovni cilj platenja, da se zagotovi **strukturiranje sorodnih problemov**, ki so funkcionalna vsebina posamezne plasti. Včasih govorimo tudi o **funkcionalni homogenosti** plasti, kar pa, povedano z drugimi besedami, pomeni, da plast združuje le sorodne storitve.

# Splošne lastnosti plasti

- V prejšnjih poglavjih smo z različnimi pristopi (intuitivno, pregled razvoja, opis posameznih primerov...) opredelili tri **funkcionalne celote**, ki jih lahko obravnavamo kot plasti:
  - Plast **informacijskega sistema** združuje nabor storitev, ki so značilne za vsak informacijski sistem, vendar ob upoštevanju možnosti razpršitve različnih virov, ki so lahko med seboj povezani tesno, rahlo (ohlapno) ali pa omrežno.

# Splošne lastnosti plasti

- Plast **transportnega sistema** poskrbi za tvorjenje sporočil iz podatkov, ki so organizirani v skladu z informacijsko plastjo, prenos sporočil in transformacijo sporočil v obliko, ki jo razume informacijska plast sprejemnika.
- Plast **prenosnega sistema** pa sporočila pretvori v signale, jih prenese prek prenosnega medija in na sprejemnikovi strani iz signalov restavrira poslano sporočilo

# Splošne lastnosti plasti

- Do teh ugotovitev smo prišli tako, da smo poizkušali opredeliti funkcionalnost sistema, standard pa plast opredeljuje nekoliko bolj administrativno, brez tehničnega ozadja. Naštejmo nekaj načel, ki po standardu formalno opredeljujejo plasti:
  - 1 Plast vzpostavimo, ko identificiramo skupino storitev, ki zahtevajo specifično obravnavo
  - 2 Vsaka plast mora pokrivati dobro opredeljene storitve, kar pomeni, da mora biti funkcionalnost plasti zelo natančno opredeljena.

# Splošne lastnosti plasti

- 3 Meje plasti naj bodo opredeljene tako, da se minimizira pretok informacij prek vmesnika storitvene pristopne točke, to je med sosednjimi plastmi.
- 4 Število plasti mora biti dovolj veliko, da lahko smiselno razvrščamo sorodne funkcije v določeno plast, hkrati pa dovolj manjhno, da arhitektura komunikacijskega sistema ostane pregledna.
- 5 Funkcionalnost posamezne plasti oziroma postopek opredeljevanja posameznih storitev naj upošteva dejstvo, da se predlog plasti lahko vključi tudi v mednarodne standardizacijske aktivnosti.



# Splošne lastnosti plasti

- V nadaljevanju se bomo najprej posvetili vprašanju, kako se po standardni terminologiji pretakajo podatki med sosednjima plastema, šele nato nas bo zanimala tudi vsebina storitev. V bistvu ne bomo povedali nič, česar se v dosednji obravnavi že ne bi dotaknili, zato je najenostavneje, da obravnavamo kar vsak pojem posebej.

# Zahteve storitve

- Storitev je funkcija ali aplikacija, ki jo lahko izvede, ponudi določena plast. Tipična storitev transportne plasti je na primer vzpostavljanje zveze med izvornim in ponornim računalnikom. Splošna lastnost vsake plasti sistema pa je naslednja:
- *Plast na noviju  $N$  ponudi svoje storitve (servis) plasti  $N+1$ , ki leži neposredno nad njo. Plast  $N+1$  mora imeti možnost, da plasti  $N$  posreduje zahtevo za določeno storitev.*

# Zahteve storitve

- Višja plast  $N+1$ , ki zahteva določeno storitev od nižje plasti, je **uporabnik storitve** (service user, service provider), storitev pa na plasti  $N$  ponudi **izvajalec storitve**. Zahteva se plasti  $N$  posreduje preko vmesnika, ki ga imenujemo **storitvena pristopna točka** – SPT (Service Access Point).
- *Storitvena pristopna točka je fizični in/ali logični vmesnik, ki opredeljuje nabor storitev plasti  $N$ , in način, kako se posamezno storitev zahteva – sintaksa zahteve.*

# Splošno o plasti

- skupina storitev, ki zahtevajo specifično obravnavo
- funkcionalnost plasti zelo natančno opredeljena
- minimizira pretok informacij prek vmesnika
- število plasti ne preveliko, ne premajhno
- upoštevati moramo dejstvo, da lahko postane plast del standarda
- zahteva na nivoju  $N+1$ , storitev posreduje plast na nivoju  $N$ , sama storitev pa se izvaja na plasti  $N-1$ .

# Komunikacijski protokol

- Izvedba storitve plasti N je pogojena z interakcijo med dvema procesoma, ki tečeta v dveh računalniških okoljih (na dveh točkah v omrežju). Glede na strukturiranost in specializiranost slojev ni težko razumeti druge pomembne splošne lastnosti plasti: **izvedljiva in smiselna je le interakcija med procesoma dveh istoležnih plasti** – dveh plasti na istem nivoju (v smeri horizontalne/logične povezave med njima), saj se le na določeni plasti izvajajo določene storitve. Vsebino in sekvence komuniciranja med takima procesoma imenujemo **komunikacijski protokol**.

# Komunikacijski protokol

- *Pravila, po katerih se z interakcijo dveh procesov na plasti N izvaja storitev, imenujemo komunikacijski protokol. Pogosto uporabimo izraz **N-protokol**, ki torej predpisuje način komuniciranja ob izvajanju določene storitve.*

# Komunikacijski protokol

- N-protokol je lahko na primer omrežni protokol, transportni protokol, aplikacijski protokol,..., kar pomeni, da ima sistem toliko komunikacijskih protokolov, kolikor ima plasti, saj komunikacijski protokol predstavlja vsebino podatkov, ki se pretakajo po logičnem kanalu določene plasti.

# Komunikacijski protokol

- Uporabnik na plasti  $N+1$ , ki je izdal zahtevo za storitve, pričakuje izvedbo storitve od plasti  $N$ . Način izvedbe, torej  $N$ -protokol sam uporabnika storitve ne zanima in ga načelno ne pozna. Izhajajoč iz te ugotovitve, lahko predpostavimo tudi drugačno definicijo komunikacijskega protokola na plasti  $N$ :
- ***N-protokol*** je izvedba storitve plasti  $N$ .



# Komunikacijski protokol

- Prva definicija je bolj opisna, vendar je po našem mnenju lahko tudi dvosmiselna, pravkar podana definicija pa v sebi skriva bistvo mehanizmov oziroma odnosov med plastjo  $N+1$  in plastjo  $N$ . Večina komunikacijskih protokolov, ki se uporabljajo širše, je nekako standardizirana, zato opozorimo, da je pogosto uporabljen izraz za  $N$ -protokol tudi **komunikacijski standard**.

# Entitetni par

- N-protokol je logična komunikacija, ki poteka le med procesoma na plasti N. Posamezni proces, v skladu s standardizirano terminologijo, imenujemo **entiteta**. Povedano z drugimi besedami komunikacija poteka med parom entitet. Entiteta je lahko strojni ali programski, logični element, lahko pa je kombinacija obeh. Sedaj lahko postavimo definicijo entitetnega para:

# Entitetni par

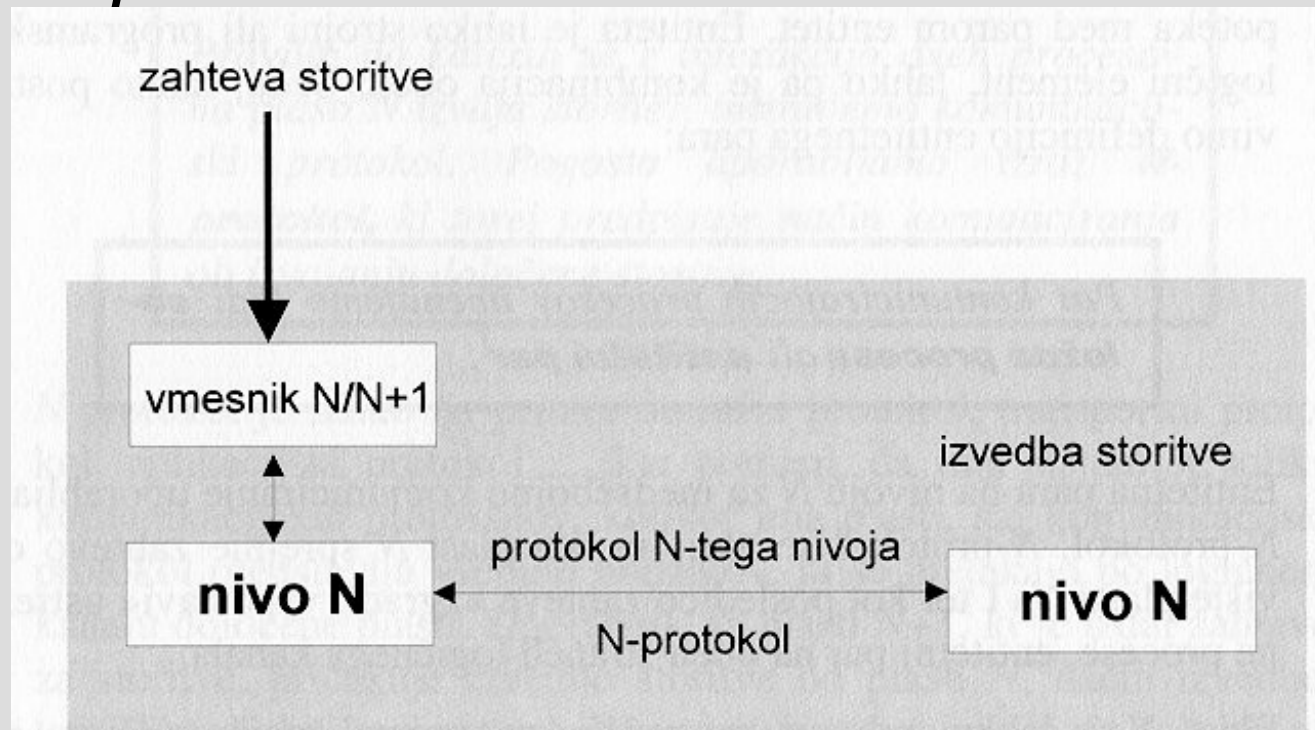
- *Par komunicirajočih procesov imenujemo tudi **soležna procesa** ali **entitetni par** (peer processes).*
- *Entitetna para na nivoju  $N$  za medsebojno komuniciranje uporabljata  $N$ -protokol.  $N$ -protokol se aktivira, ko plast  $N$  sprejme zahtevo od višje plasti  $N+1$  ter kot posledico zahteve "zgradi, vzpostavi" ustrezne procese, entitetni par na obeh straneh logičnega kanala.*

# Entitetni par

- Plast N+1 lahko zahtevo posreduje na najrazličnejše načine, ne smemo pa pozabiti, da je za izvedbo sistema, poleg opredeljenega N-protokola, eden od bistvenih pogojev tudi zagotavljanje združljivosti posameznih storitev. Zato je pomemben element opisa določene plasti N poleg dobro opredeljenih storitev in protokolov tudi **standardni dvosmerni vmesnik** med višjo in nižjo protokolarno plastjo, prek katerega se posredujejo zahteve in prevzamejo rezultati zahtevanih storitev. V predhodnem odstavku smo ta vmesnik poimenovali **SPT**.

# Entitetni par

- ***Storitvena pristopna točka*** je element plasti, ki jo mora opis/standard določene plasti *N* opredeliti enoumno, medtem ko to ne velja za *N*-protokol.



# Entitetni par

- Prejšnja trditev v bistvu pomeni to, da je za plast  $N+1$  pomembno samo to, kako se določeno storitev zahteva, nič pa ni važen način izvedbe, torej  $N$ -protokol. Lahko rečemo, da je  $N$ -protokol za plast  $N+1$  **transparenten**.
- Sedaj ko smo opredelili posamezne pojme, ki so značilni za standardno terminologijo plasti, nam preostane še opis delovanja plasti oziroma opis dinamike izvajanja storitve.

# Entitetni par

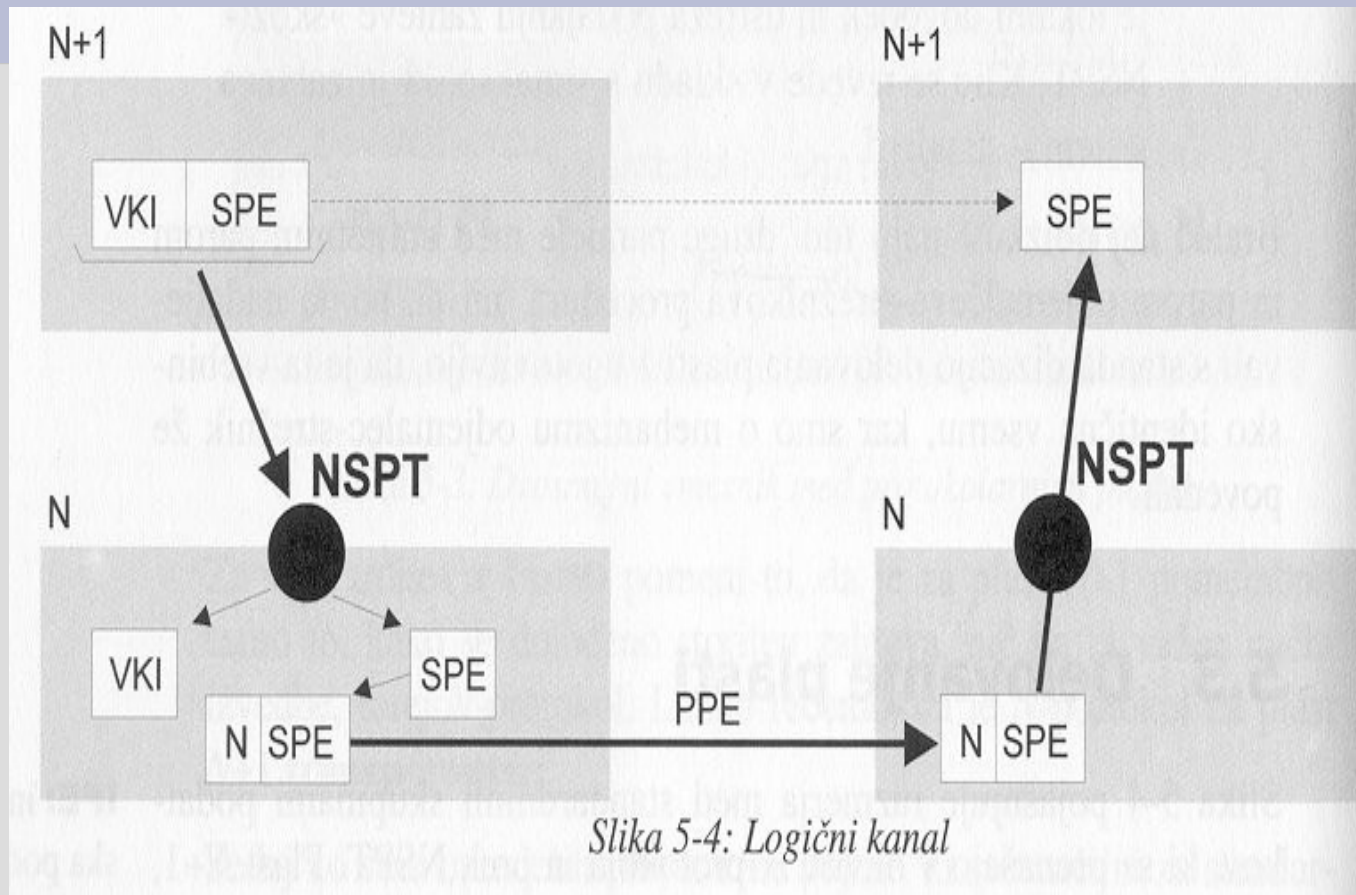
- Podane definicije pomenijo mnogo več kot na videz nepotreben formalizem. Z njimi lahko "zaslutimo" in pojasnimo dinamiko komuniciranja med uporabnikom in izvajalcem storitve – entitetnim parom. Vseeno pa smo prepričani, da bodo definicije "zaživele", če nam bo uspelo poiskati tudi stične točke z mehanizmom odjemalec-strežnik. Naštejmo nekaj najbolj bistvenih sorodnih lastnosti in vzporednic:

# Entitetni par

1. Entitetni par na plasti N je identičen odjemalčevi in strežnikovi proceduri
2. Zgornja ugotovitev implicira, da so vse bistvene funkcije oziroma funkcioniranje mehanizma odjemalec-strežnik vsebovane v N-protokolu.
3. Po enaki logiki odjemalčeva in strežnikova aplikacija ustrezata entitetnemu paru na plasti N+1.
4. Podajanje zahteve oziroma klic odjemalčeve procedure je lokalni dogodek in ustreza podajanju zahteve "skozi" NSPT. Klic se izvede v skladu s sintakso, ki jo zahteva sistem – standard.



# Delovanje plasti



- **IPE** – informacijska podatkovna enota
- **SPE** – storitvena podatkovna enota
- **VKI** – vmesniška kontrolna informacija
- **PPE** – protokolarna podatkovna enota

# Delovanje plasti

- Slika prikazuje razmerja med standardnimi skupinami podatkov, ki se pranašajo v okviru N-protokola in prek NSPT. Plast N+1, v okviru procesa "uporabnik storitve" (odjemalčeva aplikacija) generira določene podatke, ki predstavljajo vsebino komunikacije s soležnim procesom na plasti N+1 (strežnikova aplikacija). Te podatke standard imenuje **SPE – Storitvena Podatkovna Enota**.

# Delovanje plasti

- Glede na to, da se fizična povezava na plasti N+1 vzpostavi s pomočjo plasti N, mora plast N+1 SPE opremiti s podatki, ki predstavljajo vsebino zahteve, ki jo mora plast N v okviru svoje storitve ponuditi odjemalčevi aplikaciji. Tak "ukazni (krmilni)" dodatek imenujemo **VKI – Vmesniška Kontrolna Informacija**.

# Delovanje plasti

- VKI skupaj s SPE po standardni terminologiji imenujemo **Informacijska Podatkovna Enota – IPE**, ki se preda plasti N prek vmesnika NSPT (klica strežnikove procedure).

# Delovanje plasti

- VKI se uporabi na plasti N kot **krmilni podatek** pri izvršitvi zahtevane storitve na plasti N. Enostavneje povedano, VKI vsebuje podatke, ki entiteti na oddajnikovi strani entitetnega para na plasti N pove, kaj naj s podatki SPE stori. VKI si predstavljamo kot parameter določene zahteve. Na primer če gre za zahtevo po vzpostavitvi zveze, je tipičen podatek VKI naslov ponornega procesa.

# Delovanje plasti

- Na plasti N se SPE reorganizira. Na transportni plasti se na primer lahko razdeli v več delov – **paketov**, ki jih plast N opremi z dodatnimi informacijami, običajno v uvodnem delu, **N-glavi** (header) paketa.

# Delovanje plasti

- Tipičen podatek, ki sodi v glavo paketa, je na primer **zaporedna številka paketa**, ki omogoči soležnemu procesu na sprejemni strani sestaviti SPE iz koščkov v originalno obliko. SPE ali njen del, opremljen z glavo (paket), imenujemo **Protokolarna Podatkovna Enota – PPE**. PPE ustreza pojmu sporočila, kakor smo ga imenovali pri obravnavanju mehanizma odjemalec-strežnik.

# Delovanje plasti

- Na sprejemni strani se PPE "obglavi" in se sestavljen, če je pri oddaji razpadel na več paketov, preda plasti N+1 kot SPE (klic odaljemalčeve aplikacije). Vsebina SPE se na plasti N prenese **transparentno**, kar pomeni, da za plast N SPE nima koristne podatkovne vsebine. Zanja so zanimivo na oddajnikovi strani VKI, na sprejemnikovi strani pa le podatki v glavi PPE. Za prenos PPE poskrbi N-protokol.



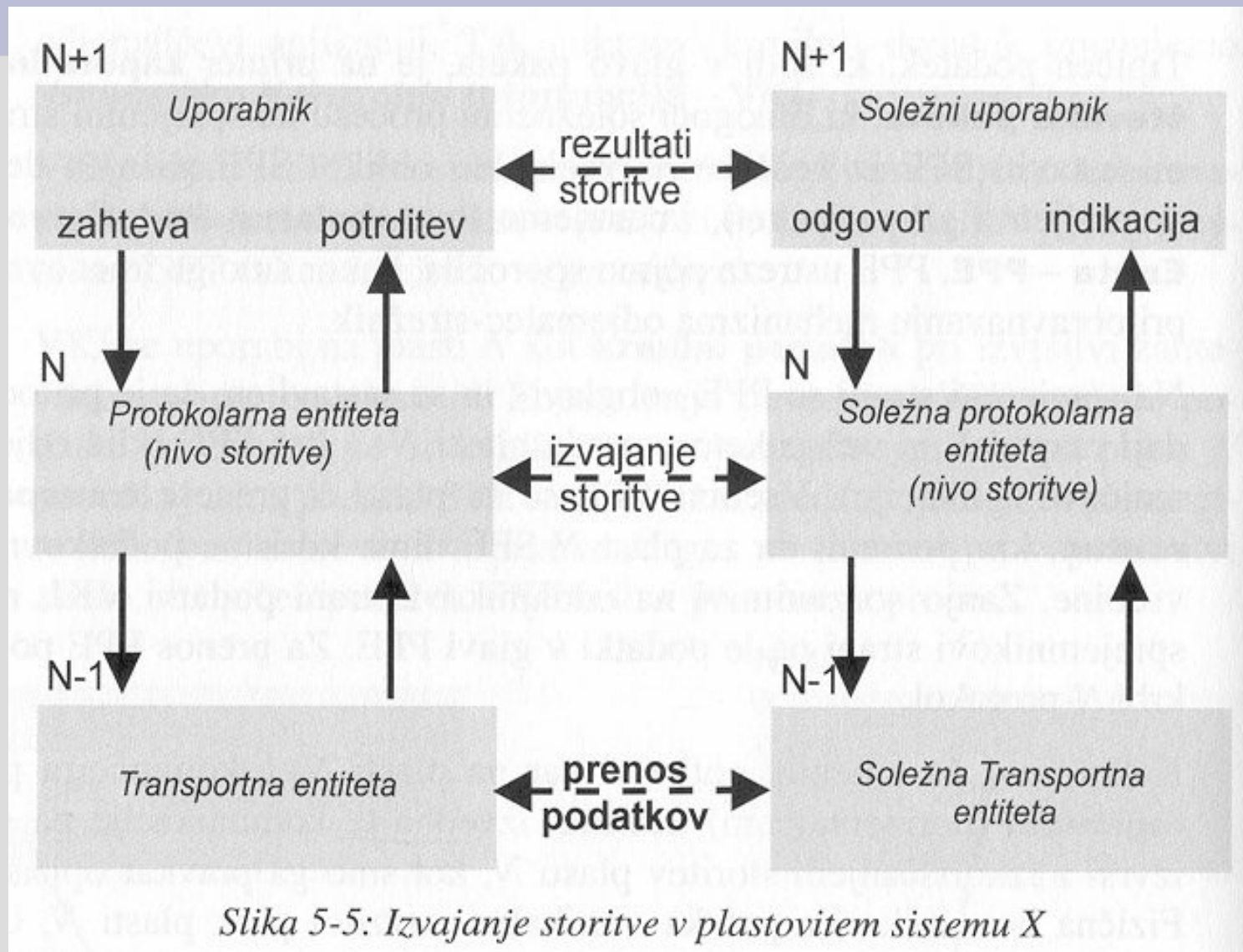
# Delovanje plasti

- Kakor smo že omenili, entitetni par na plasti  $N+1$  komunicira po **logičnem** (horizontalnem) **kanalu**, izvedba te komunikacije pa se izvrši z izkoriščanjem storitev plasti  $N$ , kot smo ga pravkar opisali. Fizična komunikacija poteka vertikalno navzdol prek plasti  $N$ , od katere plast ( $N+1$ ) na sprejemnikovi strani prevzame poslani SPE – **fizični** (vertikalni) **kanal**.

# Delovanje plasti

- **Storitev**, ki jo ponudi plast N, je za uporabnika storitve formalno opredeljena z naborom osnovnih enot **storitvenih** (servisnih) **primitivov**. Ti uporabniku omogočajo dostop do storitve prek vmesnika storitvene pristopne točke NSPT. Primitivi so v bistvu s sintakso in z vsebino opredeljeni podatki in predstavljajo VKI, saj plast N na osnovi te razpozna zahtevo.

# Delovanje plasti



# Delovanje plasti

- Slika pojasnjuje dosedanje ugotovitve o delovanju večplastnega sistema še z nekoliko drugačnega zornega kota. Če zberemo vse dosedanje ugotovitve, lahko trdimo sledeče:
  - Na plasti  $N+1$  se logično (horizontalno) prenašajo rezultati storitve na plasti  $N$ .
  - Na plasti  $N$  se logično prenašajo podatki, ki so potrebni za izvajanje storitve, ki jo je zahtevala plast  $N+1$ .

# Delovanje plasti

- Na plasti  $N-1$  pa paketi PPE z zahtevo  $N+1$  nimajo nobenega vsebinskega stika več. Ta logični kanal je zgolj nekakšen "prenosni sistem" za plast  $N+1$  in  $N$  in zato vsi podatki, ki se tičejo zahteve  $N+1$ , skozi plast  $N-1$  prehajajo transparentno.
- Ugotovitve, ki smo jih podali do sedaj, so ključne za razumevanje delovanja sistema. Če jih postavimo ob bok mehanizmu odjemalec-strežnik, je pojasnjena tudi funkcionalna vsebina  $N$ -protokola.

# Delovanje plasti

- *N-protokol podpira izvajanje storitev, seveda pa vsebuje tudi vse različice reševanja problemov, kot smo jih omenili pri obravnavi arhitekture odjemalec-strežnik.*

# Delovanje plasti

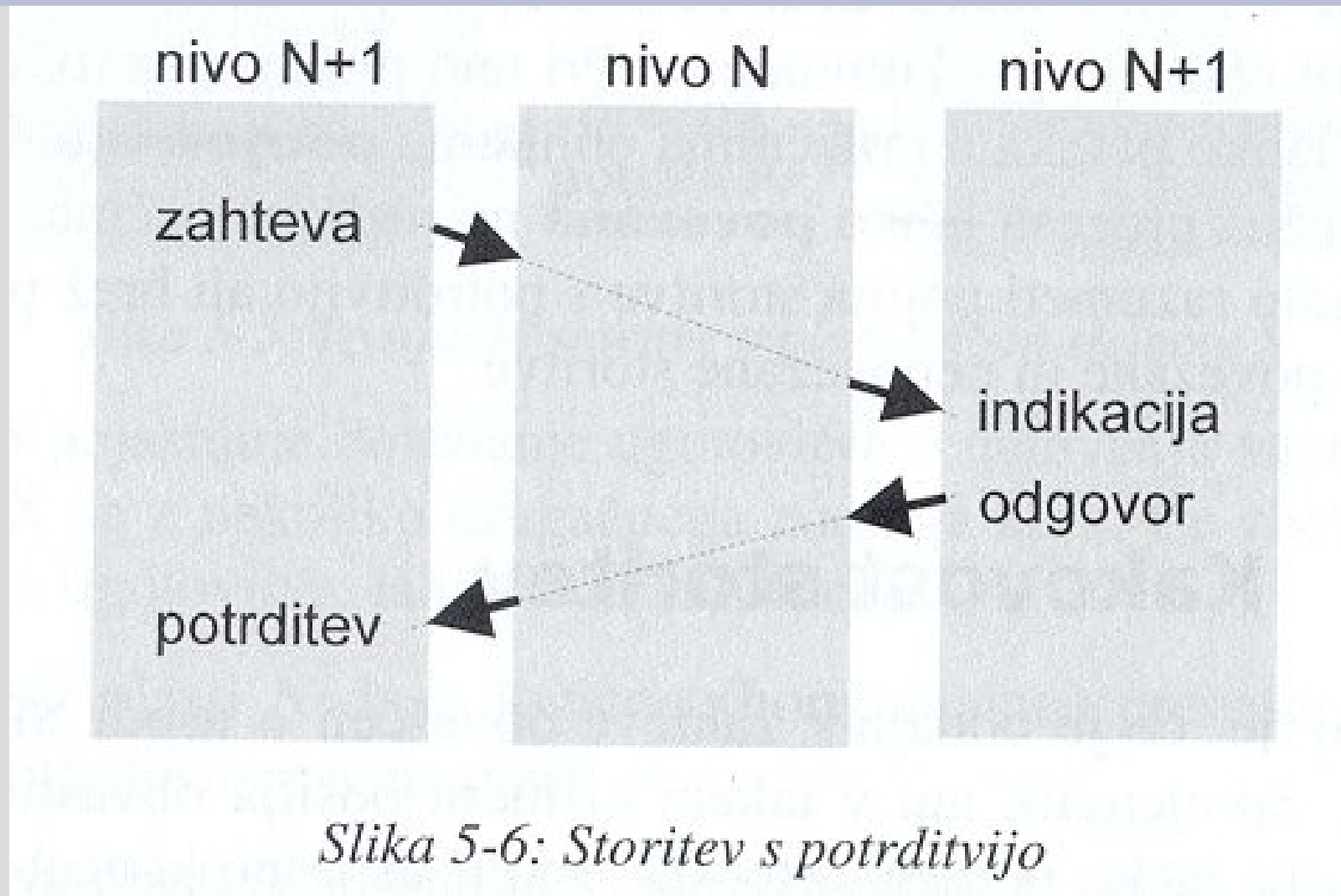
- V nadaljevanju moramo nekoliko podrobneje opredeliti še logični kanal. Na osnovi tega bomo opredelili pojem **kakovost logičnega kanala** oziroma **kakovost storitve**, ki je odvisna od načina, po katerem entitetni par komunicira. Pri tem merimo na to, da komunikacija lahko poteka z različnima oblikama **potrjevanja**, in na to, da sta soležna procesa lahko **povezana** na različne načine. To nam bo pomagalo razumeti pojma storitve s potrditvijo ali brez potrditve ter pojma povezane in nepovezane storitve.

# Kakovost storitve

- "Lepo" je, če oddajnik zahteve obveščen o usodi SPE, ki jo je poslal. Sprejemnik mu v takem primeru pošilja obvestila (**potrditve**), ki pa lahko pomenijo to, da je SPE uspešno končal svoje potovanje ali pa da je šlo nekaj narobe. Taki storitvi s potrjevanjem pravimo **storitev s potrditvijo**. Dialog entitetnega para si lahko ogledamo na sliki, kjer pa opozarjamo predvsem na terminologijo, ki jo predpisuje standard.



# Kakovost storitve



# Kakovost storitve

- Standardna terminologija v dialogu entitetnega para uporablja pojme: **zahteva**, **indikacija**, **odgovor** in **potrditev**. Poglejmo, v kakšnem medsebojnem odnosu so ti štirje pojmi:
  1. Dialog se začne, ko entiteta – oddajnik N+1 pošlje **zahtevo** – ( $IPE = VKI + SPE$ ).
  2. Soležna entiteta – sprejemnik N+1 v skladu zazna zahtevo kot **indikacijo** – SPE.

# Kakovost storitve

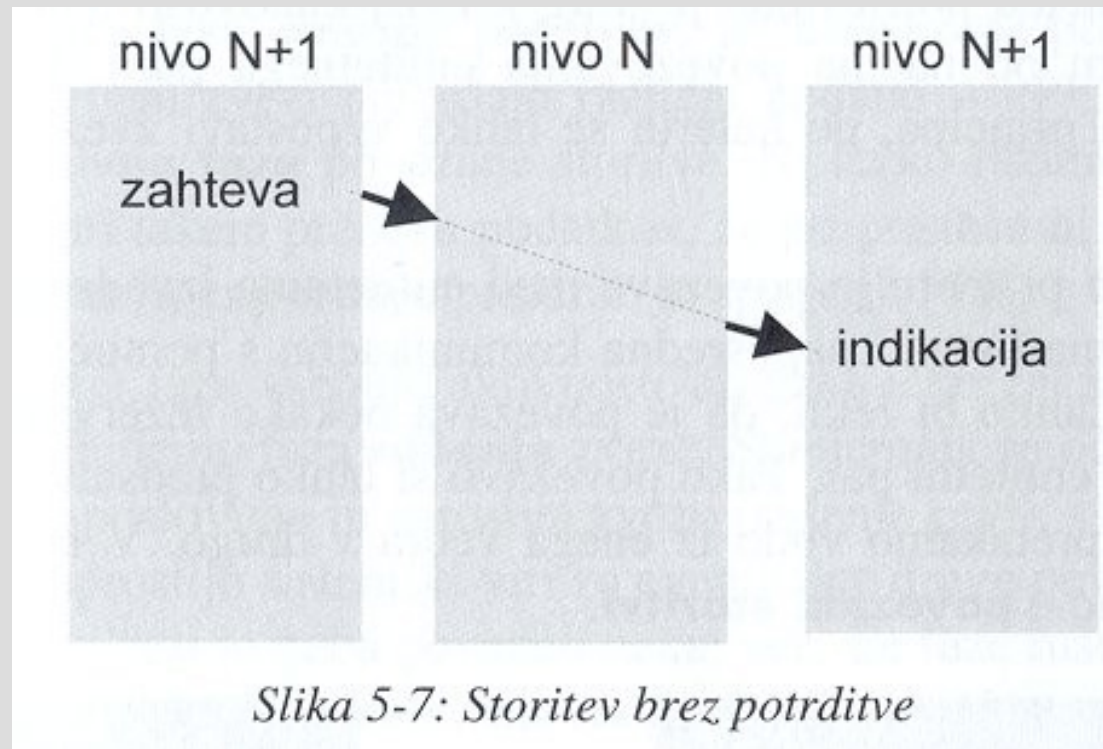
3. Sprejemnik "uporabi" SPE v skladu s storitvijo in o tem obvesti oddajnik – **odgovor**. Odgovor vsebuje podatke o tem, kakšna je bila usoda SPE na sprejemnikovi strani. Na primer, sprejem uspešen ali neuspešen.
4. Oddajnik odgovor interpretira kot pozitivno ali negativno **potrditev** zahtevane storitve.

# Kakovost storitve

- Pravkar opisana sekvenca dogodkov je značilna za, kakor smo jo imenovali, povezano storitev, saj si entiteti neprestano pošiljata ustrezne podatke tako, da vsaka entiteta avtonomno lahko sklepa na status dialoga, ki poteka, kar pa sočasno pomeni tudi to, da lahko ustrezno ukrepata v primeru težav med izvajanjem zahteve.

# Kakovost storitve

- Če v dialogu entitetnega para koraka 3 (odgovor) in 4 (potrditev) nista potrebna ali pa izvedljiva, torej če zahteva nima predvidene potrditve, govorimo o storitve brez potrditve, ki jo predstavlja slika.



# Kakovost storitve

- Tu imamo opraviti zgolj s parom zahteva-indikacija. Seveda oddajnik v tem primeru lahko samo upa, da je poslani SPE uspešno končal svojo "misijo".
- Potrjevanje je prvi kriterij za **kakovost storitve**. Očitna je namreč razlika v možnostih avtonomnega in ustreznega reagiranja entitetnega para v primeru potrjevanja oziroma v primeru, ko potrditve ni. Zato storitve s potrditvijo pogosto imenujemo kar **zanesljive**, storitve brez potrditve pa **nezanesljive** (reliable/unreliable).

# Kakovost storitve

- Kot primer zanesljive storitve lahko navedemo pošiljanje pošte s povratnico, primer nezanesljive storitve pa je običajno pošiljanje pošte brez povratnice. V nekaterih primerih pa zanesljiva storitev zaradi tehnoloških omejitev preprosto ni izvedljiva. Primer take storitve je prenos digitaliziranega govora. Zakasnitve, ki bi nastale zaradi prenosa potrditev, bi poslušalec lahko zaznal kot kratke premore sicer v tekočem govoru. Poglejmo sedaj še drug kriterij, ki prav tako vpliva na kakovost ali pa vsaj na bistvene lastnosti komuniciranja entitetnega para.

# Povezane/nepovezane storitve

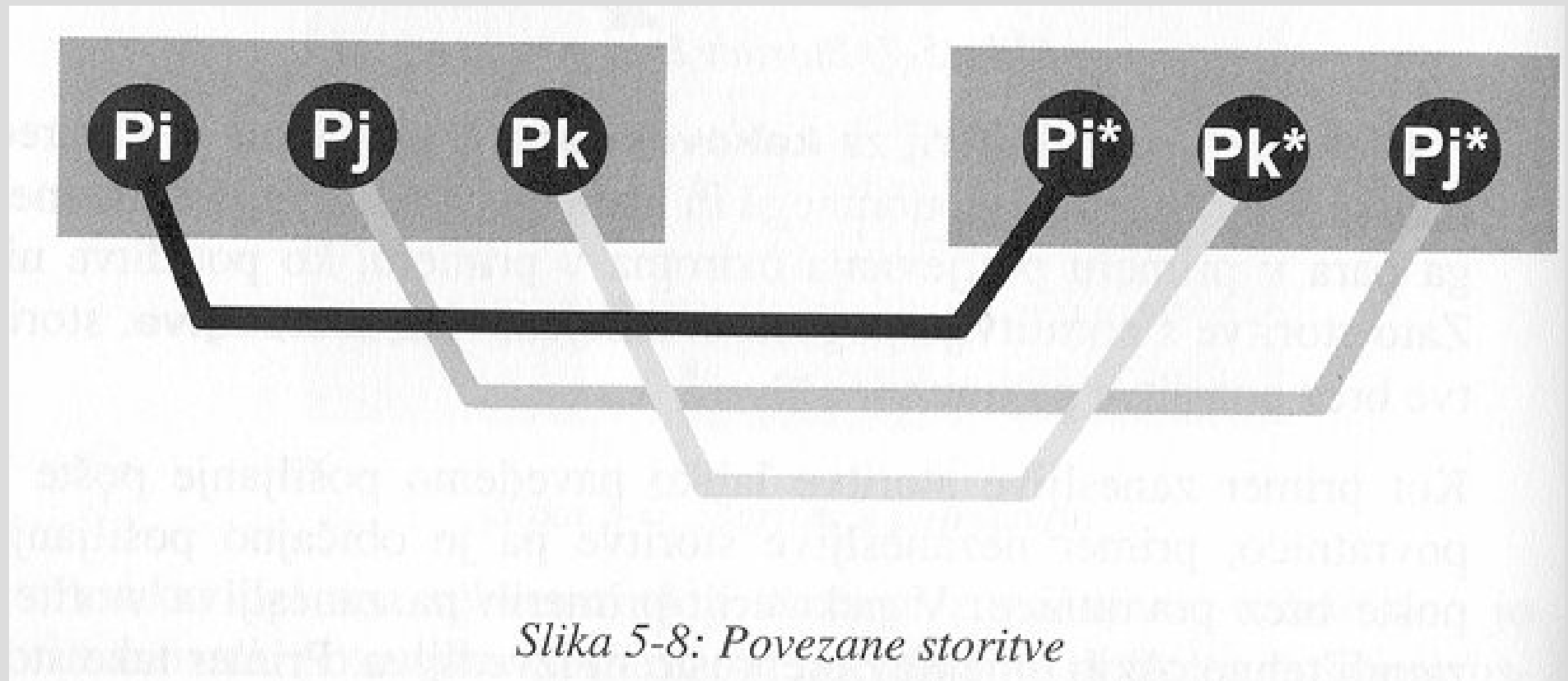
- Poleg načina potrjevanja je drug kriterij kakovosti storitve odvisen predvsem od načina povezovanja entitetnega para. Poznamo dva osnovna principa, po katerih se lahko vzpostavi zveza med procesoma.



# Povezane/nepovezane storitve

- V prvem primeru je povezava med entitetama izvedena tako, da se med njima ustvari neposredna komunikacija s pomočjo virov sistema. Lahko bi rekli, da je povezava nekako rezervirana samo za določen entitetni par. Tako povezavo si lahko predstavljamo, kot da s cevjo pretakamo vodo iz enega vedra v drugo. V takem primeru govorimo o povezani storitvi.

# Povezane/nepovezane storitve



# Povezane/nepovezane storitve

- Pri tem načinu se **logični kanal** med entitetama vzpostavi vnaprej in ostane nespremenjen do konca komuniciranja. Med dvema fizičnima točkama v sistemu je lahko sočasno vzpostavljenih več povezanih entitetnih parov, kar nazorno prikazuje slika. Identifikacija paketa na sprejemni strani je **posredna**. To pomeni, da sprejemnik ve, kateri entiteti je paket namenjen, ker ve, iz katere "cevi" je prišel.

# Povezane/nepovezane storitve

- Paketi zato v glavi nimajo ne naslova ne ponora podatkov o oddajniku, zato pa se zveza pri povezani storitvi vzpostavi na specifičen način in ima več faz:
  - Prva faza, **vzpostavljanje** zveze, je grajenje "cevi". Sprejemnik in oddajnik se medsebojno identificirata in začneta uporabljati vzpostavljeno povezavo.
  - Druga faza, **prenos** podatkov, je faza, ko se po vzpostavljeni zvezi prenašajo različni podatki – PPE. To je **aktivna faza** povezane storitve. Ni težko razumeti, da se pri takem prenosu podatkov, če pri prenosu ni napak, **ohranja zaporedje** poslanih in sprejetih PPE.

# Povezane/nepovezane storitve

- Tretja faza vsebuje aktivnosti ob zaključku zveze in jo imenujemo **faza rušenja zveze**. Sprejemnik in oddajnik se "poslovita" in sprostita zvezo (rušenje cevi), s tem pa se sprostijo sistemski viri za eventualne druge povezave. O tej fazi je treba povedati nekaj več. Če faze rušenja ne bi bilo, bi po določenem času porabili vse sistemske vire, ki so potrebni za vzpostavljanje povezav. Vzpostavljene "cevi" bi v sistemu ostajale ne glede na to, da niso več potrebne. Vsako cev bi "uporabili" samo enkrat in jo potem neizkoriščeno pustili venemar – sistema pa bi bilo zato "vedno manj".

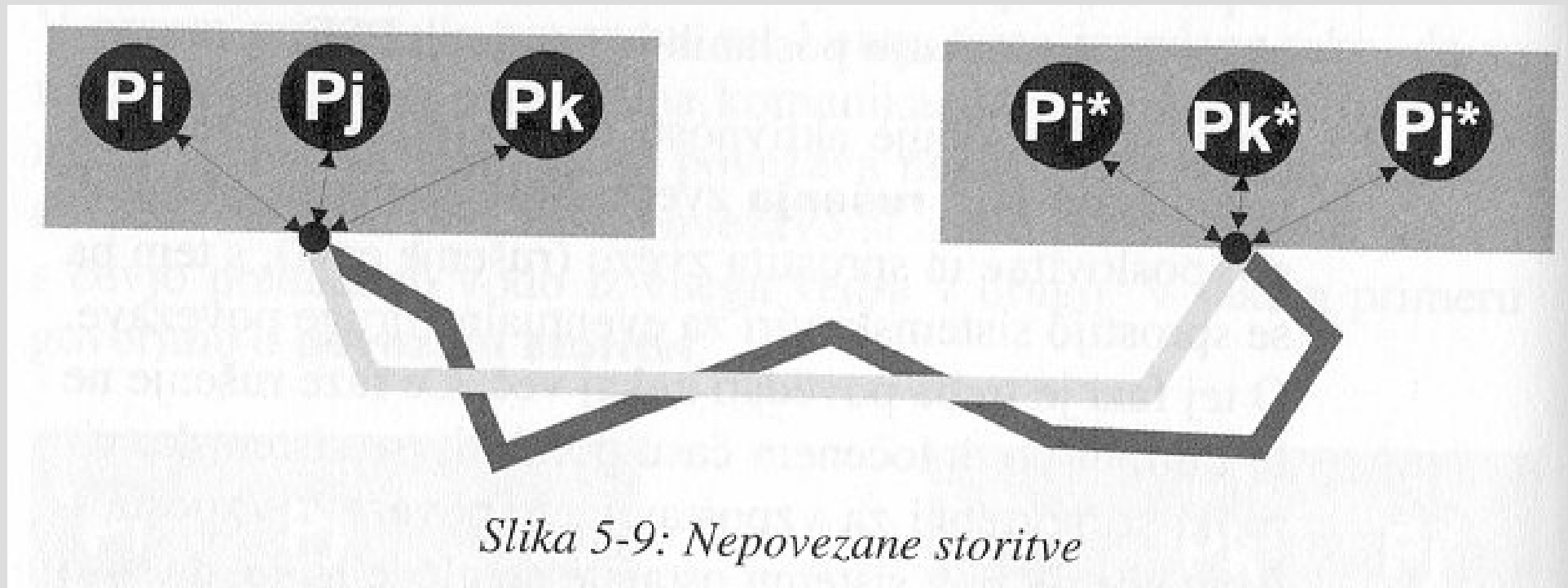
# Povezane/nepovezane storitve

- Klasičen primer povezane storitve je storitev telefonskega pogovora po klasičnem, analognem telefonskem omrežju. Faza vzpostavljanja je izbiranje številke, čemur sledi vzpostavljanje "galvanske zveze" med dvema telefonoma po telefonskem omrežju. Ta se konča, ko klicani dvigne slušalko. Faza prenosa podatkov ustreza pogovoru, ko se govorni signali prek vzpostavljene "žice" neposredno prenašajo med govorcema. Fazo rušenja povzroči odložitev slušalke. Če rušenja nebi bilo, bi to po določenem času pomenilo, da prek telefonske centrale ne bi bilo mogoče vzpostaviti nobene zveze več, ne glede na to, da prek centrale ne potreka nobena aktivna zveza.

# Povezane/nepovezane storitve

- Naj že na tem mestu omenimo najbolj znano različico povezane storitve na omrežni plasti, ki jo imenujemo **virtualna zveza**.
- Lahko pa je povezava narejena tako, da se podatki (PPE) prenašajo med entitetama po različnih poteh. Skratka, gre zato, da se vsak PPE avtonomno "prebija" skozi sistem. Če ima storitev tak način izmenjevanja PPE, govorimo o nepovezani storitvi. Nazoren primer nepovezane storitve je pretakanje vode iz vedra v vedro z več kozarci, kar sočasno počne več ljudi.

# Povezane/nepovezane storitve





# Povezane/nepovezane storitve

- Posledica take povezave je, da ima vsak PPE v glavi podatke o svojem izvoru in ponoru, saj nima cevi, po kateri bi se neposredno "pretočil" do sprejemnika. Faz vzpostavljanja ali rušenja zveze ni, zato pa morajo biti PPE označeni s podatkom o izvoru in ponoru. Tak način povezovanja omogoča večjo fleksibilnost pri prenašanju posameznih delov sporočila (PPE).

# Povezane/nepovezane storitve

- Na primer različni PPE-ji lahko izberejo različne poti do cilja, kar je na primer pogojeno z zasedenostjo sistema v določenih smereh. Prenos po različnih poteh pa lahko traja različno dolgo, zato lahko kasneje oddani paket, ki je potoval po krajši ali hitrejši poti, prehiti kakega predhodnjika, kar pa pomeni, da tak način povezovanja **ne zagotavlja sekvence** poslanih in sprejetih PPE.

# Povezane/nepovezane storitve

- V primeru internetnega omrežja govorimo tudi o **datagramski zvezi**. Datagramska zveza je nepovezana storitev omrežja, ki jo praktično uporabljamo vedno, ko dostopamo do interneta. O virtualni in datagramski zvezi bomo govorili več v poglavju o omrežjih.

# Povezane/nepovezane storitve

- S tem končujemo splošen opis funkcioniranja plasti, v naslednjem poglavju pa bomo povedali nekaj več o samem N-protokolu. Preden pa preidemo na to obravnavo, je smiselno ponoviti bistvene ugotovitve o N-protokolu, ki so sad tega poglavja.

# Povezane/nepovezane storitve

- *Entitete lahko komunicirajo s potrditvami ali brez, s čimer je opredeljena kakovost storitve. Oba tipa storitev, zanesljive in nezanesljive, pa lahko izvedemo povezano ali nepovezano.*
- Potrjevanje in povezanost sta bistvena elementa vsakega komunikacijskega protokola.

# Standardizacija

- Standardizacija obravnavanega področja poteka v dveh vzporednih tokovih:
- De facto
- De jure

# Standardizacija

- Standardi **de facto** nastajajo neodvisno od mednarodno priznanih organizacij za standardizacijo. Že v prej smo nakazali, da se je postopek standardizacije začel oblikovati v okviru močnih proizvajalcev informacijske opreme. Izdelek, ki se je izkazal za uspešnega, je (ne)hote postavil pravila za vse podobne izdelke – vsi poznamo imena VMS, DOS, WINDOWS, UNIX, PC,...

# Standardizacija

- V to skupino sodijo tako imenovani **standardi proizvajalcev** in neodvisnih virov (kot je na primer UNIX). IBM je primer podjetja z množico standardov, za katerimi stojijo njihovi uspešni izdelki. Standardi de facto vstopajo med uporabnike predvsem zaradi politike močnih proizvajalcev (na primer PC – osebni računalnik) ali pa zaradi široke uporabnosti in ugodne cene, za kar je dober primer izdelka operacijskega sistema LINUX.



# Standardizacija

- Standardi **de jure** nastajajo nekako po zakonu pod pokroviteljstvom avtoriziranih **agencij za standardizacijo**. Standardi de jure se uveljavljajo skozi vrata **nacionalne in mednarodne uradne standardizacijske politike**.
- Glavni akterji standardizacije na področju telekomunikacij so nacionalne in mednarodne telekomunikacijske družbe – pri nas Telekom. Mednarodna organizacija tega tipa pa je na primer AT&T.

# Standardizacija

- Mednarodna organizacija za standardizacijo, ki pokriva področje komunikacij, se imenuje ITU-T (International Telecommunication Union – Telecommunication Standardization) (včasih CCITT - Comitee Consultatif International Telegraphique et Telephonique – <http://www.itu.int/home/index.html>). Kot enega od njenih dosežkov lahko omenimo standard X.25 (paketno preklopljeno omrežje - telefon).

# Standardizacija

- Splošnejša mednarodna organizacija za standardizacijo pa je ISO (International Standard Organization – <http://www.iso.org>), katerega članice so mnoge nacionalne organizacije za standardizacijo: ANSI (American National Standards Institute – <http://www.ansi.org/>), BSI (British Standard Institute - <http://www.bsi-global.com/>), DIN (Deutscher Institut für Normung – <http://www2.din.de/>), pa tudi SIST (Slovenski Inštitut za Standardizacijo – <http://www.sist.si/> - prej Urad za Standardizacijo in Meroslovje Republike Slovenije).

# Standardizacija

- Delovanje ISO je organizirano področno po strokah. V okviru nacionalnih organizacij so osnovane delovne skupine (working group), v katerih se usklajujejo nacionalni predlogi za standarde. Nacionalni predstavniki na mednarodnem nivoju ISO sprejemajo mednarodne standarde na podlagi nacionalnih predlogov.

# Standardizacija

- Nastajanje mednarodnega standarda de jure je zavito v dokaj toge strokovno administrativne postopke, ki imajo za posledico veliko počasnost pri sprejemanju standardov. Na drugi strani pa se standardi de facto rojevajo veliko bolj praktično in se potrjujejo s svojo komercialno in uporabniško kvaliteto.

# Standardizacija

- Nekaj neodvisnih standardnih organizacij:
  - IEEE – Institute of Electrical and Electronics Engineers - <http://www.ieee.org>
  - ECMA - European Computer Manufacturers Association - <http://www.ecma-international.org/>
  - W3C – World Wide Web Consortium - <http://www.w3.org/>
  - Request For Comment - <http://www.rfc.net/>

# Standardizacija

- Nekaj standardov, ki je nastalo iz proizvajalcev računalniške opreme:
  - SNA – system network architecture – IBM
  - DNA – distributed network architecture – DEC
  - XNS – Xerox network system – Xerox
  - DSN – distributed system network HP

# Standardizacija

- Na področju, ki ga zajema IKS, danes zasledimo dva izredno močna trenda standardizacije:
  - standardizacijo de jure je utemeljil sedemplastni ISO OSI (Open System Interconnection) referenčni model.
  - standardizacija de facto poteka v okviru štiriplastnega TCP/IP modela.



# ISO/OSI Referenčni model

- Model arhitekture ISO OSI sestoji iz 7 plasti, ki obsegajo tipične storitve sistema. Model se ne ukvarja z notranjo standardizacijo posamezne plasti, temveč predpisuje zgolj vmesnike med lokalno informacijsko infrastrukturo in transportnim sistemom.

# ISO/OSI Referenčni model

- Oglejmo si funkcionalnost njegovih plasti:
  - **Fizična plast** skrbi za prenos bitov prek prenosnega medija in zagotavlja standardno aparaturno priključevanje sistemov na prenosni medij
  - **Povezavna plast** prenaša podatkovne okvire med dvema točkama, ki sta povezana s prenosnim medijem. Osnovna naloga te plasti je odkrivanje napak, ki se zgodijo med prenosom po fizičnem prenosnem mediju.
  - **Omrežna plast** skrbi za usmerjanje paketov skozi topologijo omrežja – izvaja usmerjevalne algoritme.

# ISO/OSI Referenčni model

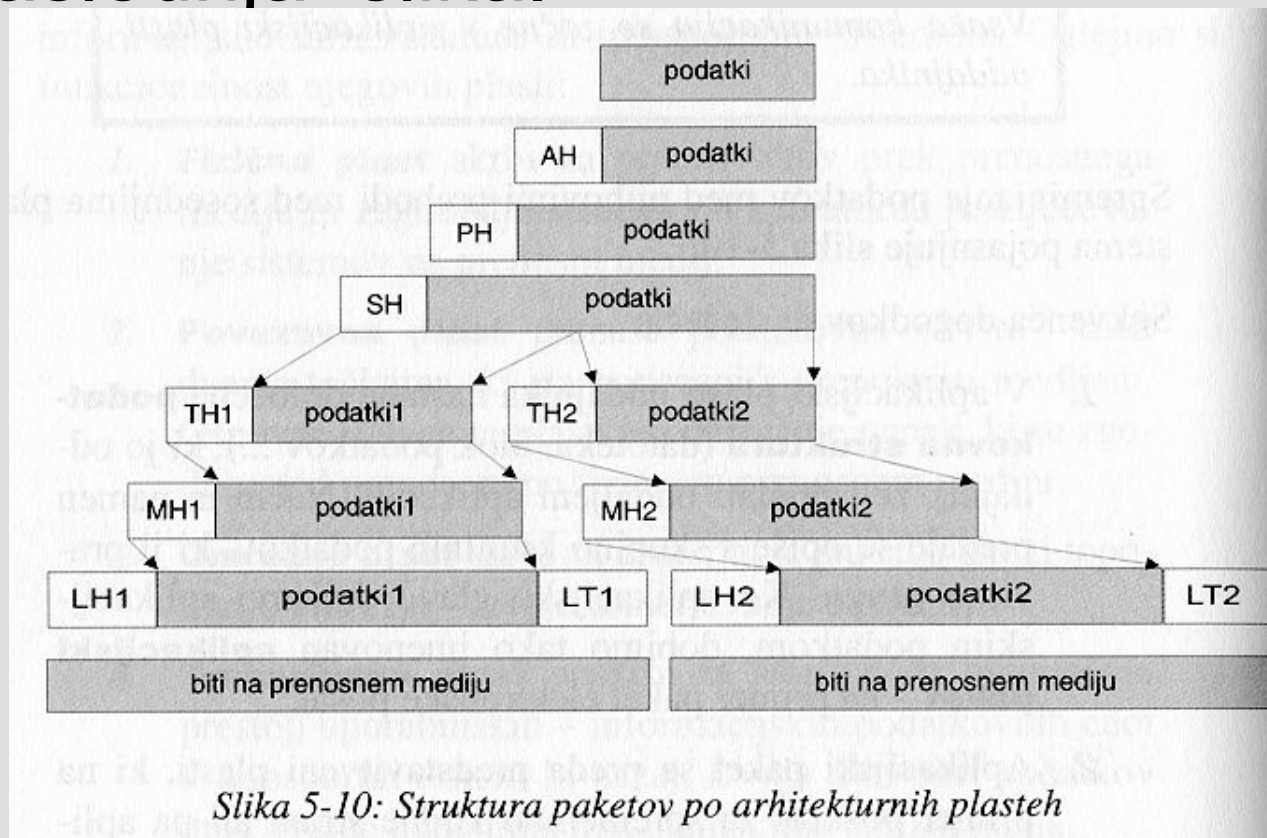
- **Transportna plast** poskrbi za storitve, ki omogočajo prestop uporabniških – informacijskih podatkovnih enot v transportni sistem in nazaj. Izvaja transport podatkov med dvema končnima računalniškima aplikacijama.
- **Plast seje** je namenjena storitvam, ki podpirajo logično povezovanje oddaljenih procesov med seboj.
- **Predstavitvena plast** skrbi za združljivost predstavitve podatkov v različnih računalniških okoljih in za zaščito podatkov.
- **Aplikacijska plast** vsebuje celo vrsto standardnih aplikacij, brez katerih si danes ne moremo več predstavljati sistema (na primer storitev elektronske pošte).

# ISO/OSI Referenčni model

- Seveda smo o vsaki plasti na tem mestu povedali res malo, zgolj osnovne lastnosti in nabor osnovnih storitev. V nadaljevanju bomo sorodne plasti obravnavali v posameznih poglavjih, na tem mestu pa še pojasnimo, kaj se dogaja s podatki, ki prehajajo med plastmi. Pomembno je razumeti:
- *Vsaka komunikacija se začne v aplikacijski plasti oddajnika.*

# ISO/OSI Referenčni model

- Spreminjanje podatkov med njihovimi prehodi med sosednjima plastema pojasnjuje naslednja slika.



# ISO/OSI Referenčni model

- Sekvenca dogodkov je sledeča:
  - 1) V aplikacijski plasti oddajnika nastane določena **podatkovna struktura** (datoteka, blok podatkov...), ki jo oddalnik želi predati oddaljeni aplikaciji. Način in namen predaje se opiše s skupino krmilnih podatkov, ki ji pravimo **glava**. Ko aplikacijsko glavo dodamo aplikacijskim podatkom, dobimo tako imenovani **aplikacijski paket** – na primer paket elektronske pošte.

# ISO/OSI Referenčni model

- 2) Aplikacijski paket se preda predstavitveni plasti, ki na primer poskrbi za spremembo kodne strani ali pa aplikacijske podatke zakodira. Nato se podatkom doda **prezentacijska glava**, s čimer dobimo **prezentacijski paket**.
- 3) Po predaji se na sejni plasti poskrbi za način logičnega komuniciranja med oddaljenima procedurama. Tipično gre za sinhronizacijo ali pa na primer za opredeljitev komunikacijskega kanala (sočasno ali izmenično dvosmerna komunikacija ...). Seveda tudi tu dodamo glavo in pridemo do **sejnega paketa**.

# ISO/OSI Referenčni model

- 4) Sejni paket, ki na primer vsebuje podatke o sliki z velikostjo 4MB, se na transportni plasti transformira v množico **transportnih paketov**, ki so primerne velikosti za prenos po transportnem sistemu (npr. 1024 bajtov). Bistveni podatek **transportne glave** je **sekvenčna številka** paketa.
- 5) Na omrežni plasti se transportnemu paketu v **omrežni glavi** doda predvsem podatke, ki so potrebni za usmerjanje **omrežnih paketov** skozi omrežje.



# ISO/OSI Referenčni model

6) Opazimo, da je **povezavni paket** nekaj posebnega, saj ima poleg **glave** (header) tudi **rep** (trailer). V repu **okvira**, kot imenujemo paket na povezavni plasti, so dodane informacije, ki so namenjene odkrivanju napak pri prenosu prek prenosnega medija. Ime okvir izhaja iz dejstva, da so podatki uokvirjeni med dva kontrolna bloka.

# ISO/OSI Referenčni model

- 7) Na fizični plasti se okvirji **konvertirajo v signal**, ki predstavlja **tok bitov**, ki se prenaša prek prenosnega medija med dvema računalnikoma.
- Opisana sekvenca dogodkov na posamezni plasti se na sprejemnikovi strani odvije v obratnem vrstnem redu: tok bitov se identificira kot množica okvirov, iz okvirov se v primeru korektnega prenosa tvorijo omrežni paketi...

# ISO/OSI Referenčni model

- Velikokrat se prehajanje paketov pojasnjuje na sledeč način: Vsaka plast na oddajnikovi strani zapakira paket višje plasti v novo kuverto (glava tekoče plasti). Na koncu je sporočilo zaprto v sedmih kuvertah. Sprejemnik pa odpira kuverto za kuverto tako, da na koncu sprejemnikova aplikacija dobi izvirne podatke oddajnika. Podobni principi delovanja veljajo tudi za internetno arhitekturo, ki jo bomo na kratko opisali v naslednjem razdelku.

# TCP/IP Referenčni model

- Model TCP/IP je dobil ime po znanih standardih, ki jih uporablja kot jedro transportnega sistema:
  - **TCP** – Transmission Control Protocol ustreza OSI-jevi transportni plasti. Poleg protokola TCP imamo na transportni plasti tudi protokol **UDP** – User Datagram Protocol, ki je nepovezana (datagramska) različica transportnega protokola. Več o povezanih in nepovezanih storitvah bomo povedali v naslednjem poglavju.

# TCP/IP Referenčni model

- **IP** – Internet Protocol ustreza OSI-jevi omrežni plasti. Po tem protokolu je dobilo ime tudi danes najbolj razširjeno omrežje internet.
- TCP/IP-jeva **aplikacijska plast** je znamenita zaradi storitev in protokolov Telnet, FTP, SMTP, SNMP, HTTP, ssh... tiste poslušalce, ki jim našteje kratice ne pomenijo veliko, naj za zdaj potolaži dejstvo, da bodo več o vseh teh storitvah izvedeli v nadaljevanju predavanj.

# TCP/IP Referenčni model

- V tem modelu je precej siromašna **plast računalniškega omrežja** (imenovana host-to-host network), ki ne pove nič več kot to, da je računalnike nekako treba priključiti na omrežje tako, da bodo lahko pošiljali in prejeli pakete IP. Ta plast ustreza povezavni in fizični plasti modela ISO OSI.

# Primerjava modelov OSI in TCP/IP

- Ali je boljši model OSI ali TCP/IP? O tej temi je bilo izrečenih in napisanih že nešteto polemik. Vsakega od modelov goreče brani skupina "tehnoloških vernikov".
- TCP/IP je standard **de facto**, ki se prebija do uporabnikov z neverjetno prilagodljivostjo in uporabnostjo. Na nekaterih mestih pa mu manjka sistematičnosti in konceptualnosti, zato se pojavljajo predlogi o OSI TCP/IP-ju.

# Primerjava modelov OSI in TCP/IP

- TCP/IP je model svetovnega omrežja internet, OSI pa je model nacionalnih operaterjev telekomunikacijskih storitev. TCP/IP je hiter in fleksibilen, OSI pa je sistematičen in konceptualen, vendar neskončno počasen proces standardizacije. TCP/IP je preplavljen z množico uporabnih (pa tudi neuporabnih) storitev, OSI čuti kronično pomanjkanje izvedb in izdelkov. Večina izdelkov iz domene internet je poceni ali celo brezplačnih, izdelki OSI so praviloma zelo dragi.



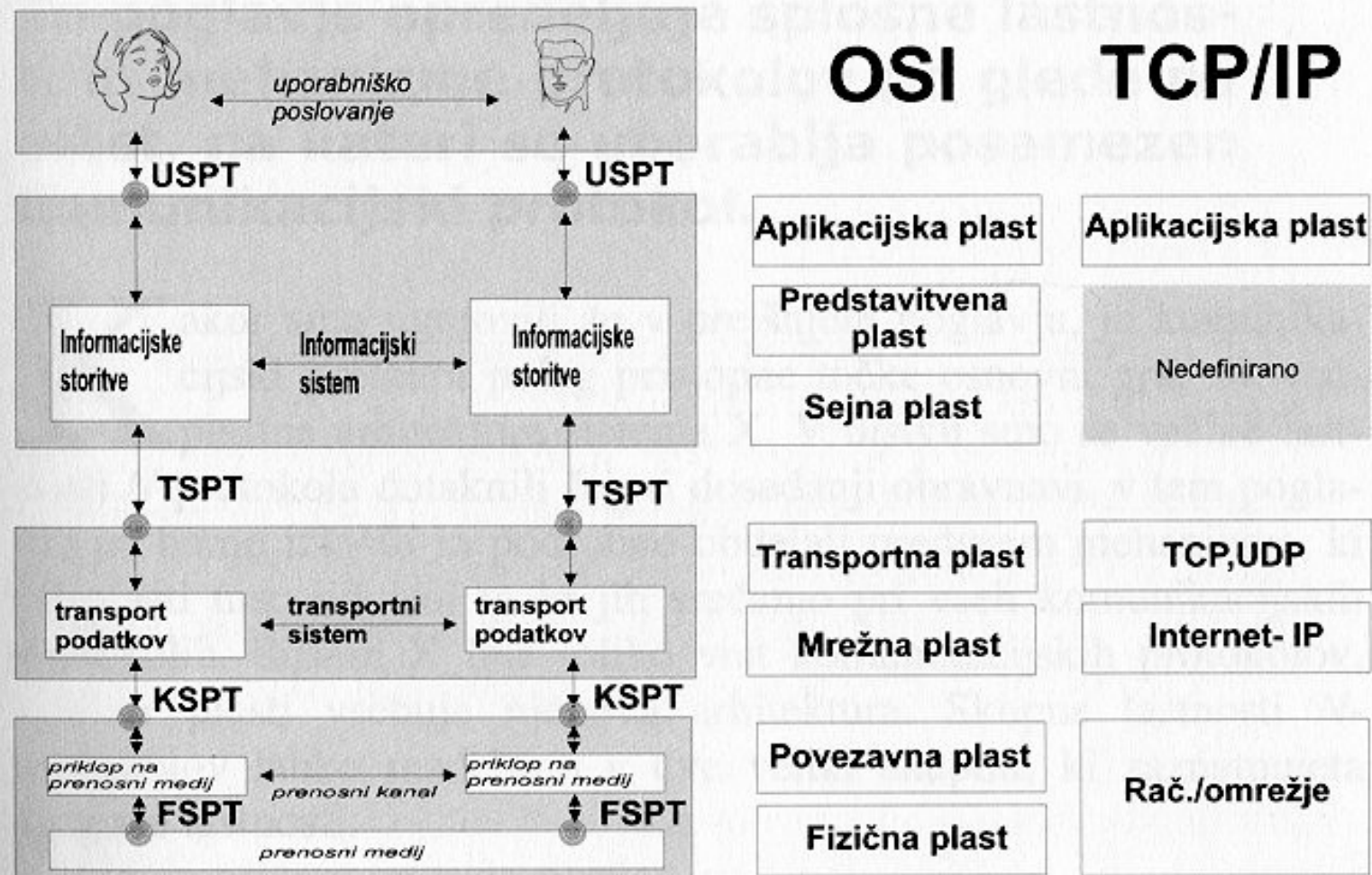
# Primerjava modelov OSI in TCP/IP

- Zanimivo je obe arhitekturi opazovati v luči splošnih ugotovitev. Na naslednji sliki bomo videli, da informacijska plast po OSI-ju ustreza aplikacijski, prezentacijski in sejni plasti, v internetnem skladu pa imamo samo aplikacijsko plast. V obeh primerih transportnemu sistemu ustrezata transportna in omrežna plast, medtem ko ima model OSI za prenosni sistem povezavno in fizično plast, internetna arhitektura pa se s problemi prenosnega sistema sploh ne ukvarja, ampak pričakuje sistem, ki je sposoben prenašati pakete IP.

# Primerjava modelov OSI in TCP/IP

- Zdi se, da je v tem trenutku model TCP/IP ponuja izdelke za vsakdanjo rabo danes in tukaj, OSI pa ponuja koncepte, ki se v zadnjih letih kot izdelki vedno uveljavljajo počasneje. Lep dokaz za to je standard elektronske pošte X.400, ki je v primerjavi z internetnim standardom SMTP šolski primer sistematičnosti. Zlobni jeziki pa omenjajo tudi njegovo neuporabnost.

# Primerjava modelov OSI in TCP/IP



Slika 5-11: Arhitektura omrežja po standardih OSI in TCP/IP