Razvoj softvera 2

Distributivni softverski inženjering

Petković Stefan 1035/2012

Distribuiran sistem

- Sastoji se od više autonomnih računara koji komuniciraju međusobno preko računarske mreže
- Računari međusobno komuniciraju u svrsi postizanja zajedničkih ciljeva
- Program koji se izvršava na distribuiranim sistemima je distribuiran program, a distributivno programiranje je proces pisanja takvih programa

Distribuirani sistemi

- Koriste se za rešavanje računarskih problema, problem se razlaže na zadatke, koji se rešavaju od strane jednog ili više računara koji međusobno komuniciraju *prosleđivanjem poruka*

- *"… kolekcija nezavisnih računara koji se pojavljuje korisniku kao jedan koherentan sistem."*

Tannenbaum i Van Steen

Distribuirani sistemi

- Kompleksniji od sistema koji se izvršavaju na jednom procesoru
- Ne postoji jedan nadležni organ zadužen za sistem tako da je kontrola odozgo na dole nemoguća

Karakteristike

- **Deljenje resursa**
  - Deljenje hardverskih i softverskih resursa
- **Otvorenost**
  - Korišćenje opreme i softvera od različitih proizviođača
- **Konkurentnost**
  - Konkurentno izvršavanje kao poboljšanje performansi
- **Skalabilnost**
  - Povećan protok dodavanjem novih resursa
- **Otpornost na greške**
  - Sposobnost nastavljanja operacije nakon greške

Transparentnost

- Korisnici ne smeju biti svesni da je sistem distribuiran i servisi bi trebali biti nezavisni od distribuiranih karakteristika
- U praksi, to je nemoguće, jer se delovima sistema samostalno upravlja, kao i zbog mrežnih kašnjenja
  - Često je bolje da korisnici budu upoznati sa distribucijom, kako bi se nosili sa problemima
- Da bi se postigla transparentnost, potrebno je apstrahovati resurse, dozvoljen je pristup samo logičkim resursima
  - Middleware je softwerski sloj koji mapira fizičke resurse na njihove logičke reprezentacije

Otvorenost

- Otvoreni distribuirani sistemi građeni prema opšte prihvaćenim standardima
- Komponente bilo kog proizvođača mogu se integrisati u sistem i mogu da komuniciraju i rade nesmetano sa ostalim komponentama sistema
- Otvorenost podrazumeva da se komponente sistema mogu samostalno razvijati u bilo kom programskom jeziku, i ako je to u skladu sa standardima, oni će raditi sa drugim komponentama
- Veb servisni standardi za servisno orjentisane arhitekture su razvijeni da budu otvoreni standardi
  - CORBA(1990)

Skalabilnost

- Sposobnost isporuke visoko kvalitetnih usluga kao zahtev za povećanje sistema
  - *Veličina* - mogućnost dodavanja više komponenata u sistem kako bi se izborio sa većim brojem korisnika
  - *Distribucija* – mogučnost geografskog rasturanja komponenata od sistema bez ugrožavanja performansi sistema
  - *Upravljivost* – mogučnost upravljanja sistemom pri povećanju njegove veličine, čak iako se delovi sistema nalaze u različitim nezavisnim organizacijama

Skaliranje na gore – moćniji sistem

Sklairanje van – više sistemskih instanci

Bezbednost

- Kada je sistem distribuiran, postoji ogroman broj načina na koji sistem može biti napadnut za razliku od centralizovanih sistema
- Ako je jedan deo sistema uspešno napadnut, onda napadač može da koristi ovo kao “back door” za ostale delove sistema
- *Vrste napda:*
  - *Presretanje*
  - *Prekid*
  - *Modifikacija*
  - *Zloupotreba resursa*

Kvalitet usluge

- Odražavanje sposobnosti sistema da svoje usluge isporuči u zavisnosti od vremena odziva i protoka koji je prihvatljiv korisnicima.
- Kritično kada se sistem bavi vremensko zahtevnim podacima kao što su audio i video tokovi.
  - U ovim okolnostima ako kvalitet usluga opadne ispod granice, moguće da audio i video budu toliko degradirani da se kao takvi ne mogu razumeti.

Otpornost na greške(padove)

- Neminovno da će doći do kvarova, stoga sistem mora biti otporan na neuspehe.
- *“Znate da radite sa distribuiranim sistemom, kada pad sistema prouzrokuje greška za koju nikada niste čuli.”*
- Distribuiran sistem treba da podržava mehanizme za pronalaženje dela sistema koji uzrokuje kvar, da uspostavi nesmetan rad ostalih komponenti i da radi na popravci kvara.

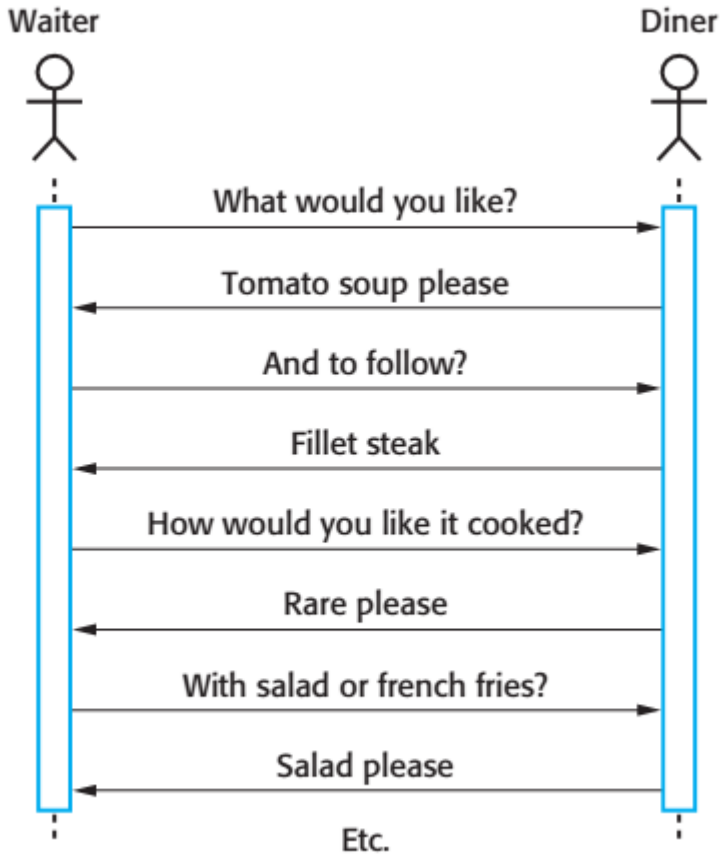
Modeli interakcije

- Postoje dva tipa interakcije između komponenata distribuiranog sistema:
  - ***Proceduralna interakcija*** – gde jedan računar poziva servis koji se nudi na drugom računaru i čeka njegov odgovor.
  - ***Interakcija na bazi poruka*** – gde jedan računar poziva šalje drugom informacije o zahtevima i nema potrebe da čeka na odgovor.

Modeli interakcije

– *Proceduralna interakcija*

Primer gosta i konobara



Modeli interakcije

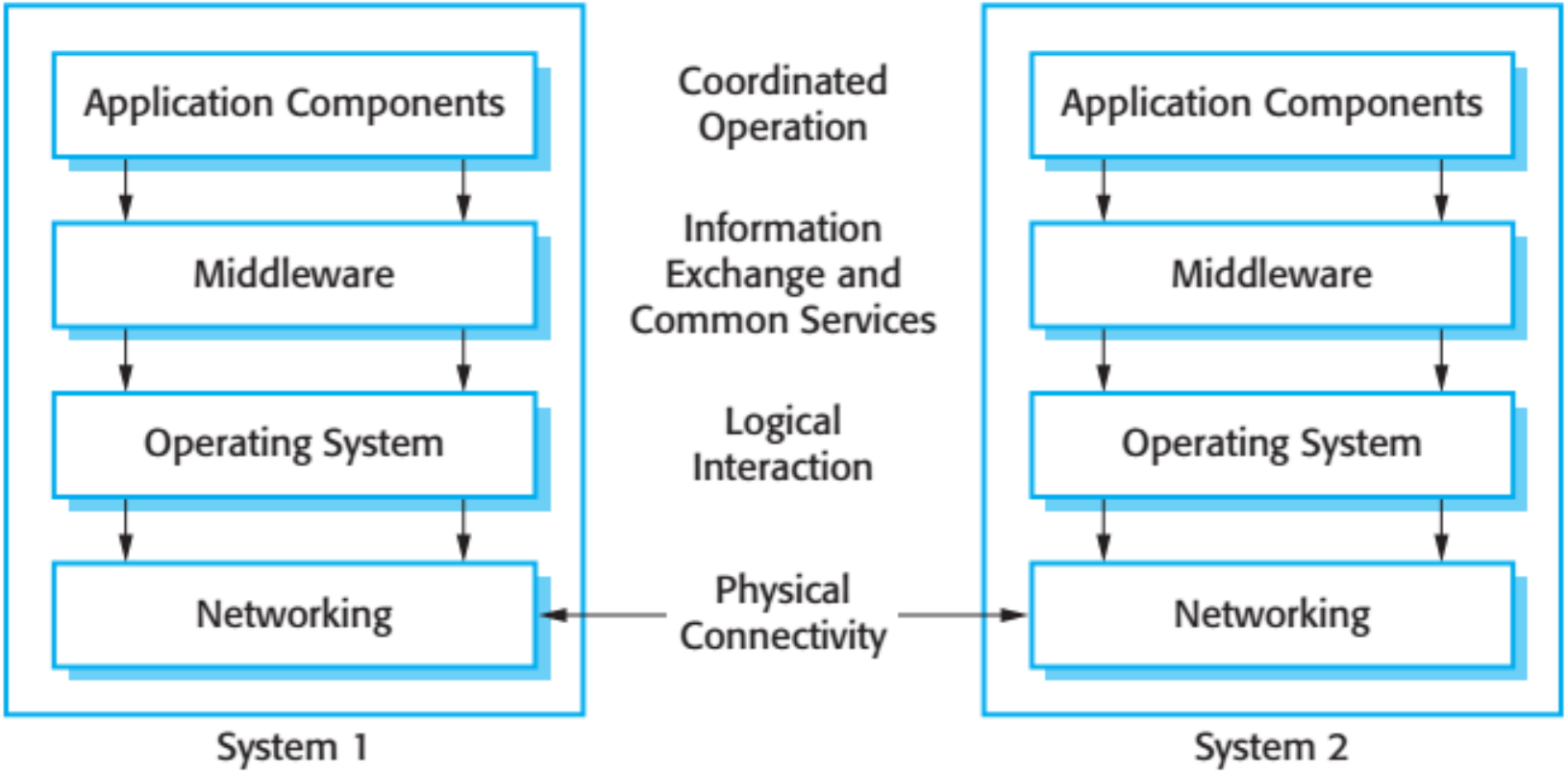
Interakcija na bazi poruka

<starter>
<dish name = “soup” type = “tomato” />
<dish name = “soup” type = “fish” />
<dish name = “pigeon salad” />
</starter>
<main course>
<dish name = “steak” type = “sirloin” cooking = “medium” />
<dish name = “steak” type = “fillet” cooking = “rare” />
<dish name = “sea bass”>
</main>
<accompaniment>
<dish name = “french fries” portions = “2” />
<dish name = “salad” portions = “1” />
</accompaniment>

Remote procedure calls

- Proceduralna komunikacija u proceduralnim sistemima je implementirana preko Remote procedure calls (RPC)
- Kod RPC-a, jedna komponenta poziva drugu, kao da je to lokalna procedura ili metod. Posrednik u sistemu presretne ovaj poziv i prosledi dalje komponenti. Sprovodi se potrebno računanje i preko posrednika se vraća rezultat komponenti koja je pozvala
- Problem kod RPC-a je da sagovornici moraju biti dostupni u celom toku komunikacije, i moraju znati kako se obraćaju jedan drugom

Posrednik u distribuiranom sistemu



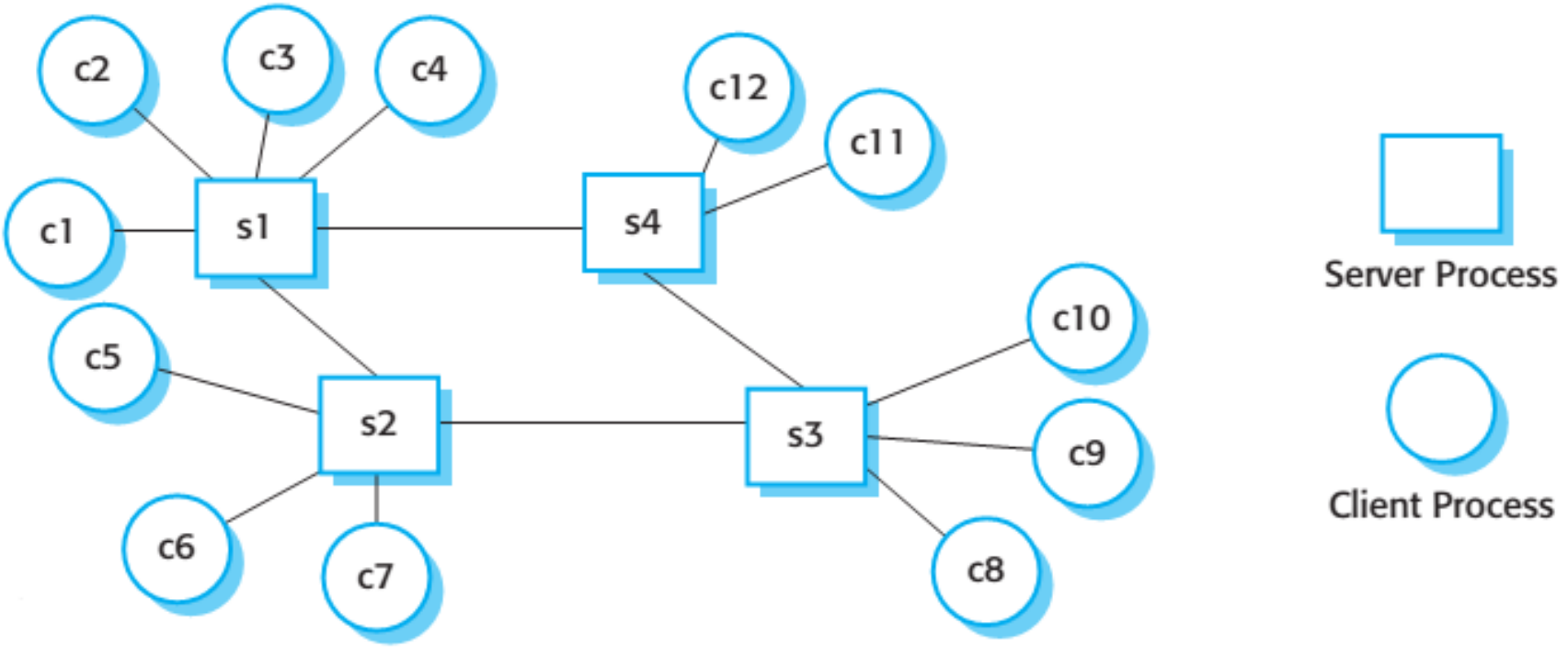
Posrednička(Middleware) podrška

- *Interaktivna podrška* – posrednik kordinira interakcijama između različitih komponenata u sistemu
  - Posrednik obezbeđuje informacije o lokacijama, tako da jednoj komponenti nije potrebno da pamti lokacije ostalih
- *Odredba zajedničkih službi* – posrednik obezbeđuje implementaciju servisa koji može biti korišćen od strane više komponenti u distribuiranom sistemu
  - Korišćenjem ovih zajedničkih usluga, komponente mogu lako da komuniciraju i pružaju korisničke servisne usluge na konzistentan način

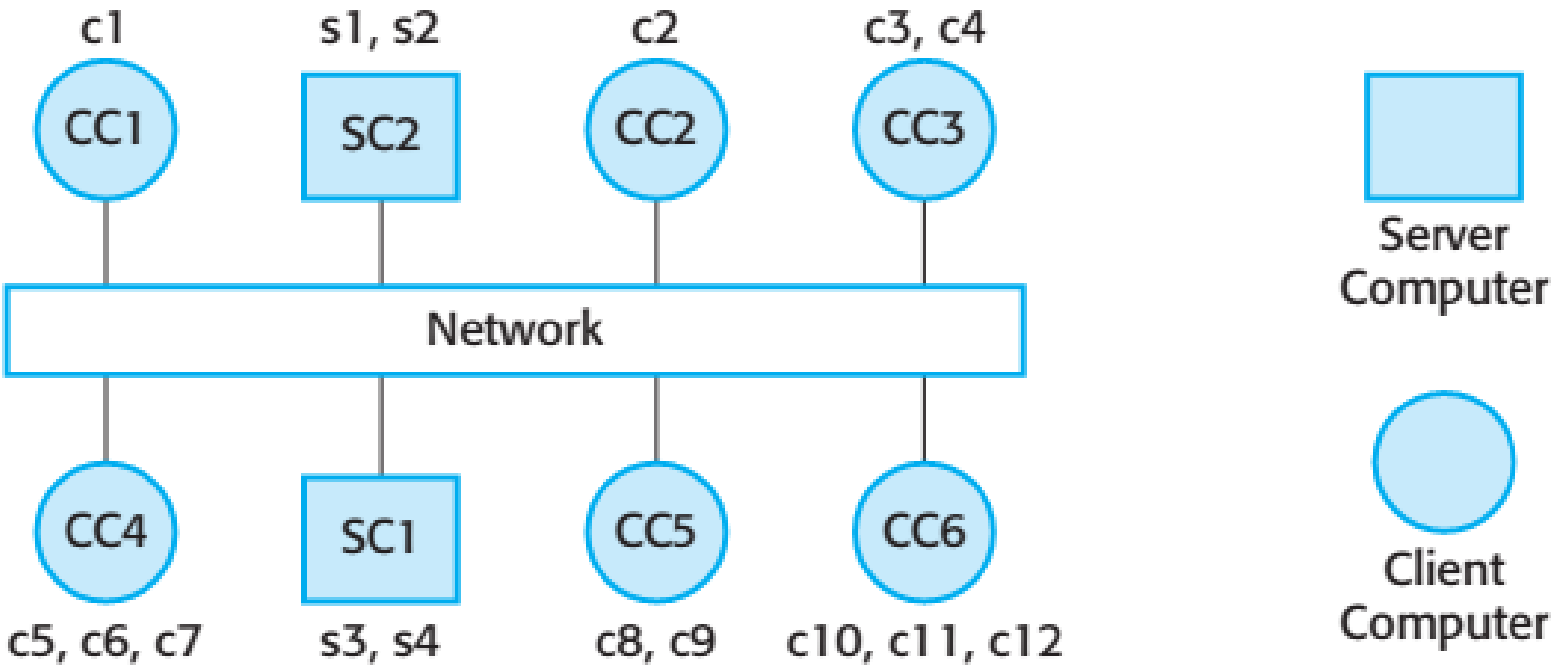
Client-server computing

- Distribuiranim sistemima kojima se pristupa preko mreže su normalno organizovani kao klijentsko-servisni sistemi
- U klijent-server sistemima, korisnik interaguje sa programom koji se izvršava na lokalnom računaru (web browser). On dalje komunicira sa programom koji se pokreće na udaljenom računaru
- Udaljeni računar nudi servise, kao što su pristupi web stranama, koji su dostupni spoljnim klijentima

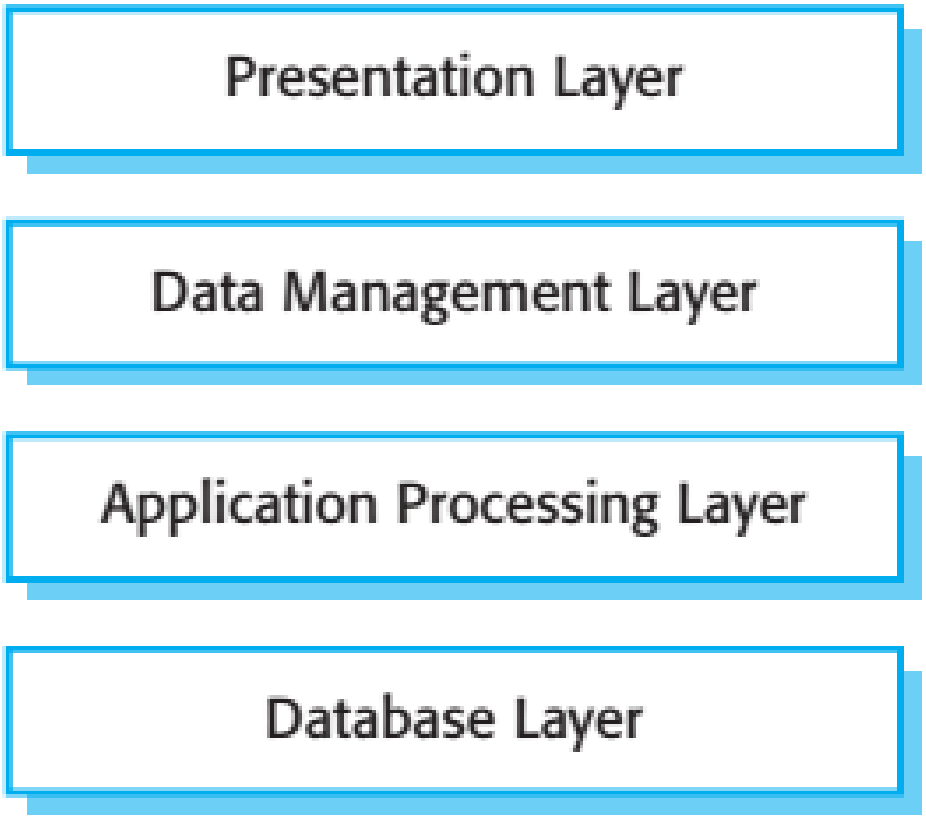
Client-server interakcija



Mapiranje klijenata i servera za umrežene računare



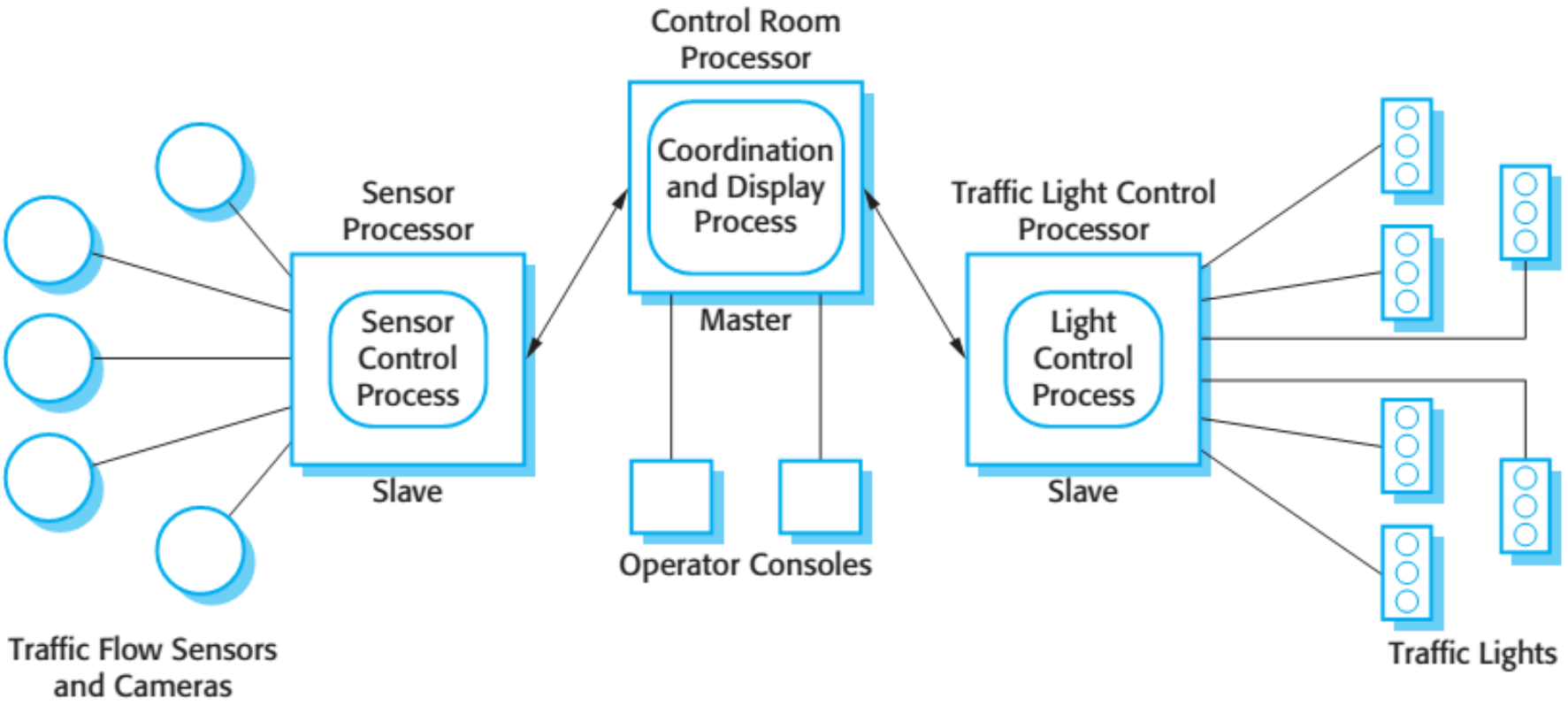
Slojevita arhitektura modela za klijentsko serverske aplikacije



Arhitektonski šabloni

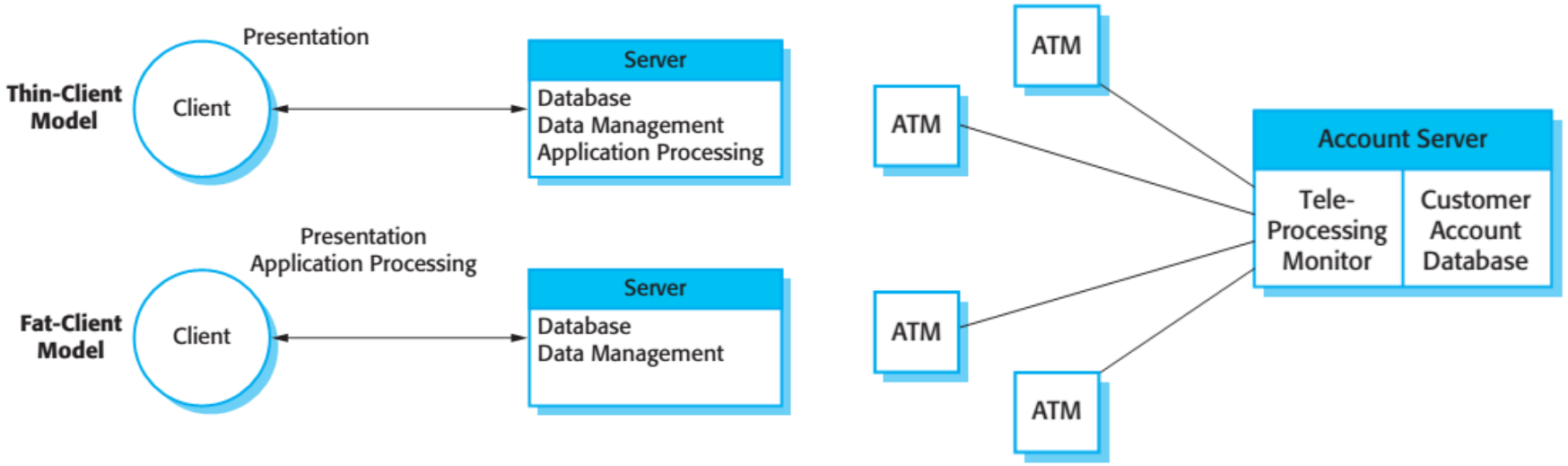
- Način organizovanja upotrebe arhitekrure distribuiranog sistema:
  - *Master-Slave* arhitektura, koristi se u real-time sistemima u kojima se garantuje interakcija koja je zahtevana.
  - *Dvostepena klijentsko-serverska arhitektura*, koristi se u jednostavnim klijentsko-serverskim sistemima, gde je sistem centralizovan iz bezbednosnih razloga.
  - *Višestepena klijentsko-serverska arhitektura*, koristi se kada postoji veliki broj transakcija koje moraju biti obrađene od strane servera.
  - *Arhitektura distribuiranih komponenti*, koristi se kada sredstva iz različitih sistema i baza podataka treba da se kombinuju, ili kao implementacioni model za višestepenu klijentsko-serversku arhitekturu.
  - *Peer-to-peer arhitektura*, koristi se kada klijenti razmenjuju lokalno uskladištene informacije i uloga servera je da upozna klijente jedne sa drugima.

Master-slave arhitektura



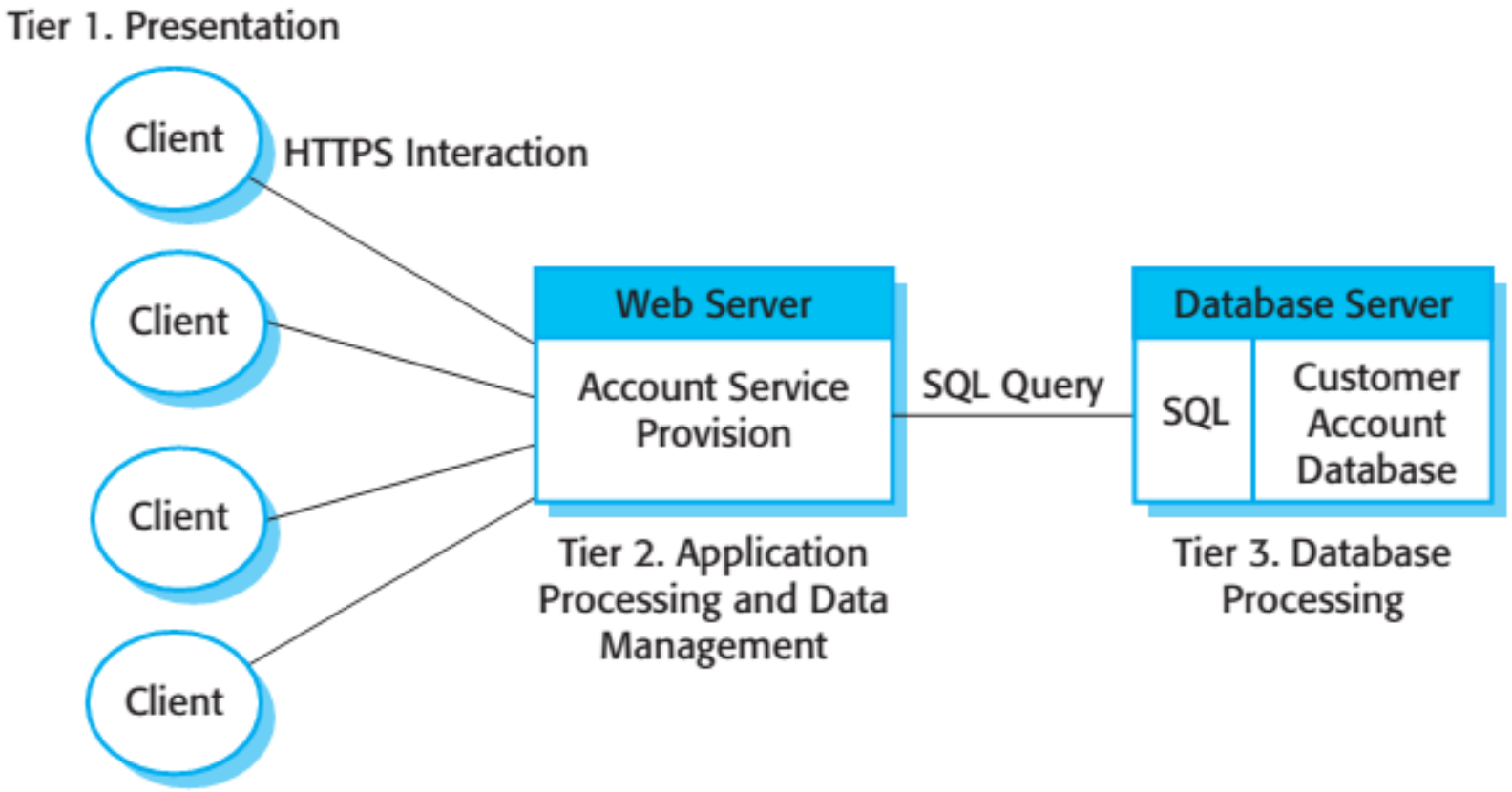
Klijentsko-serverska arhitektura

Dvostepena

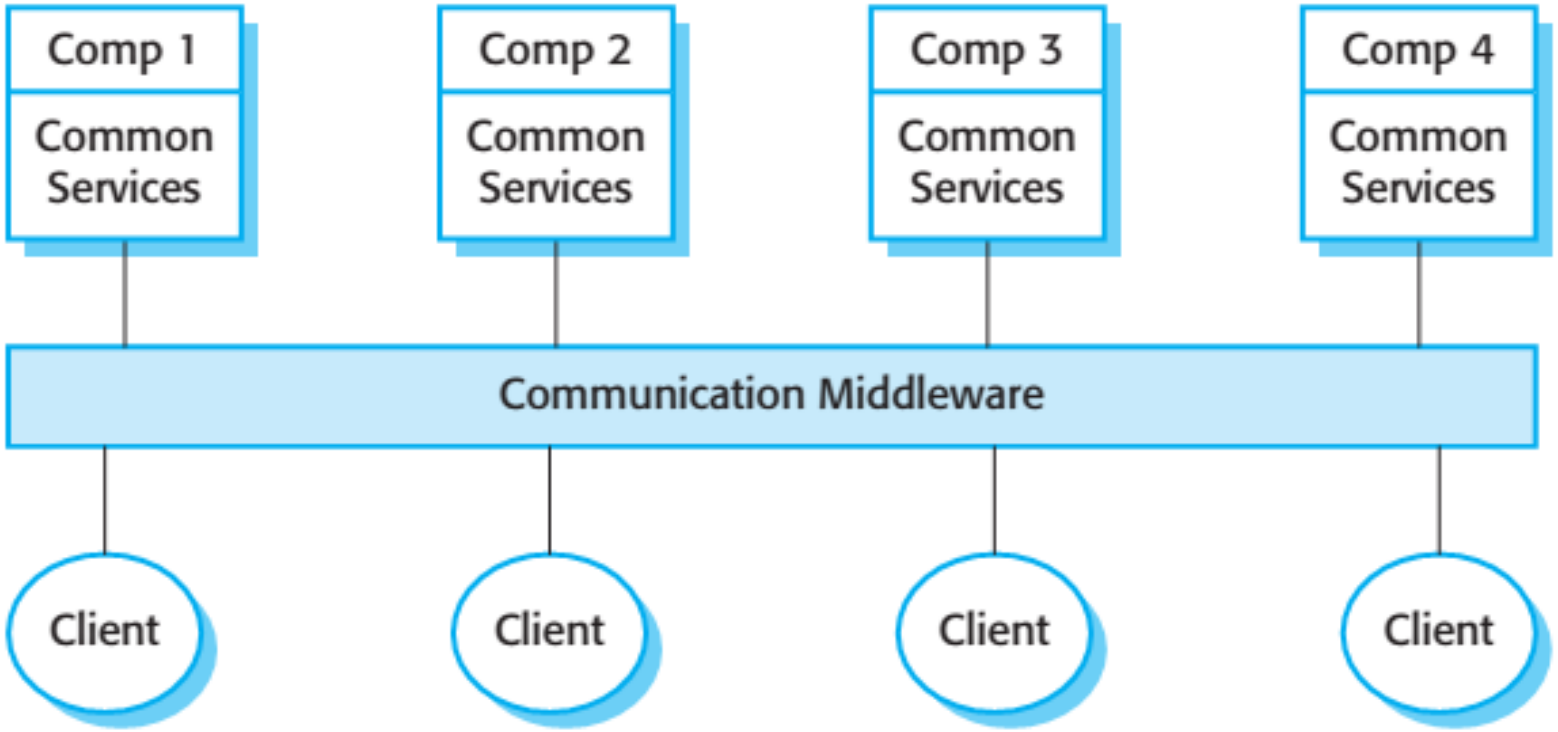


Klijentsko-serverska arhitektura

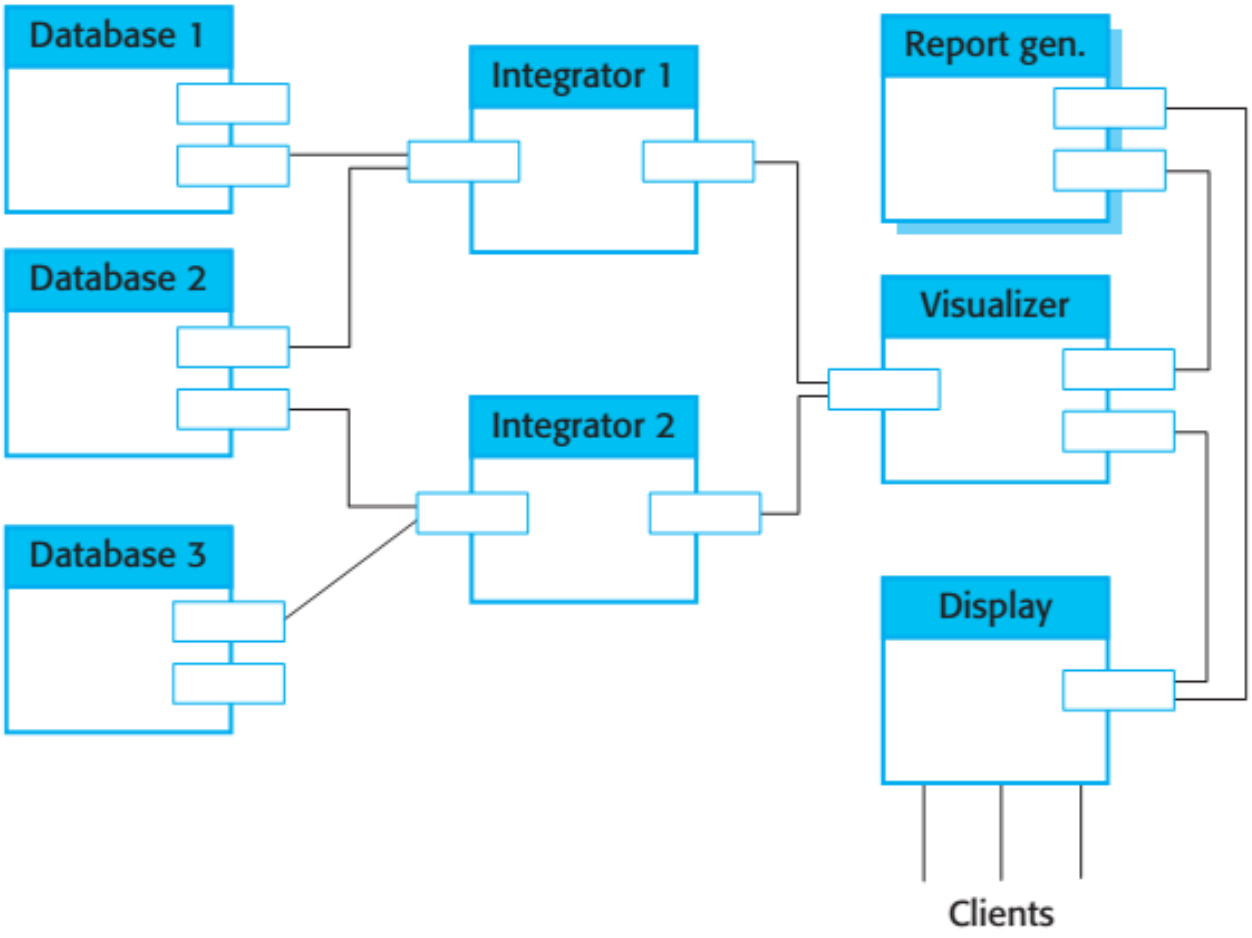
Višestepena



Arhitektura distribuirane komponente

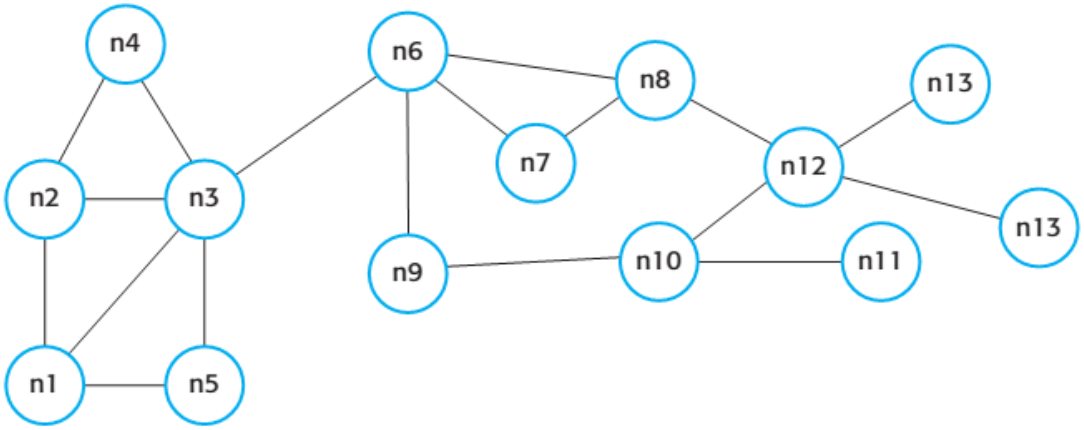


Distribuirane komponente za data mining

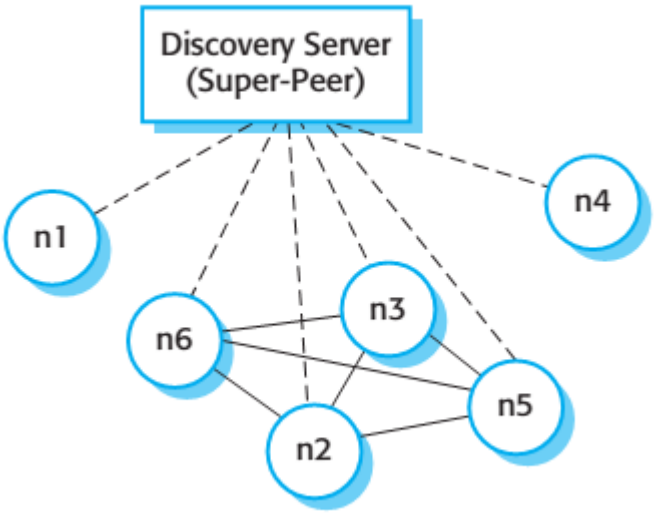


Peer-to-peer arhitektura

Decentralizovana



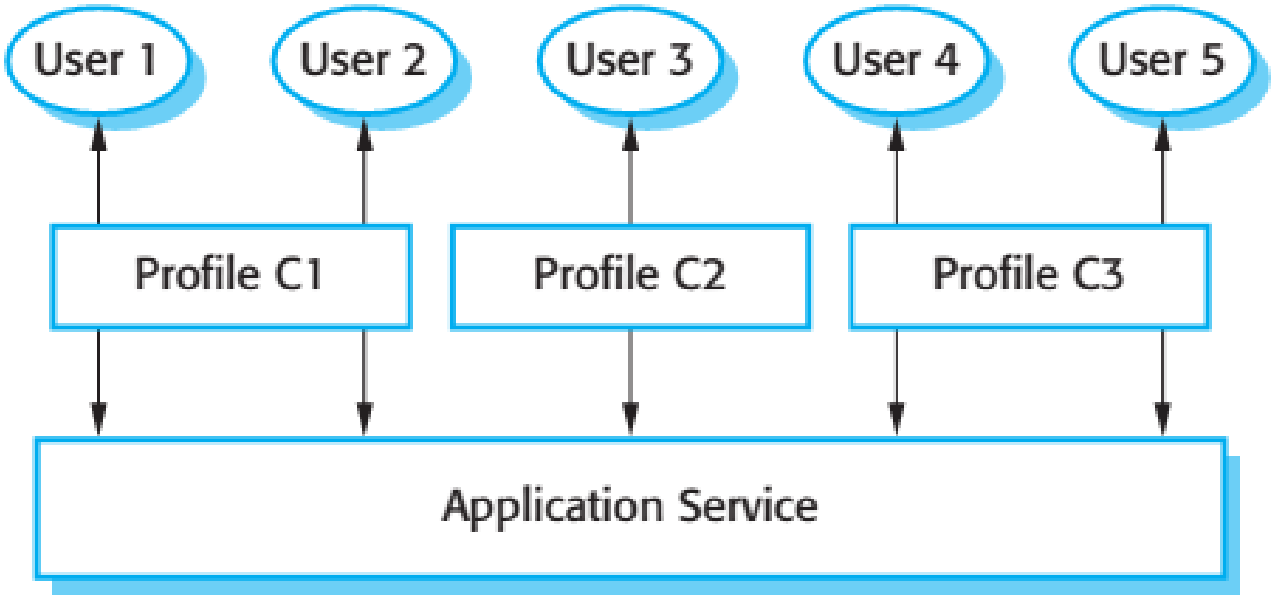
Polu centralizovana



Software as service

- Podrazumeva držanje daljinskog softvera i obezbezbeđivanje pristupa istom preko mreže
  - Softver se raspoređuje na serveru(često i većim brojem servera) kojima se pristupa preko web pretraživača. Softver nije raspoređen na lokalnom računaru.
  - Vlasnik i upravljač softvera je provajder, a ne organizacije koje koriste softver.
  - Korisnici za softver plaćaju po visini upotrebe ili na mesečnom nivou.

Konfiguracija SaaS-a



Pitanja?

Petković Stefan
petkocfc@gmail.com