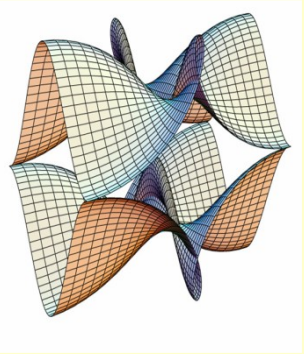
Sveučilište u Zagrebu
PMF – Matematički odsjek

MREŽE RAČUNALA
Predavanja 2022/2023

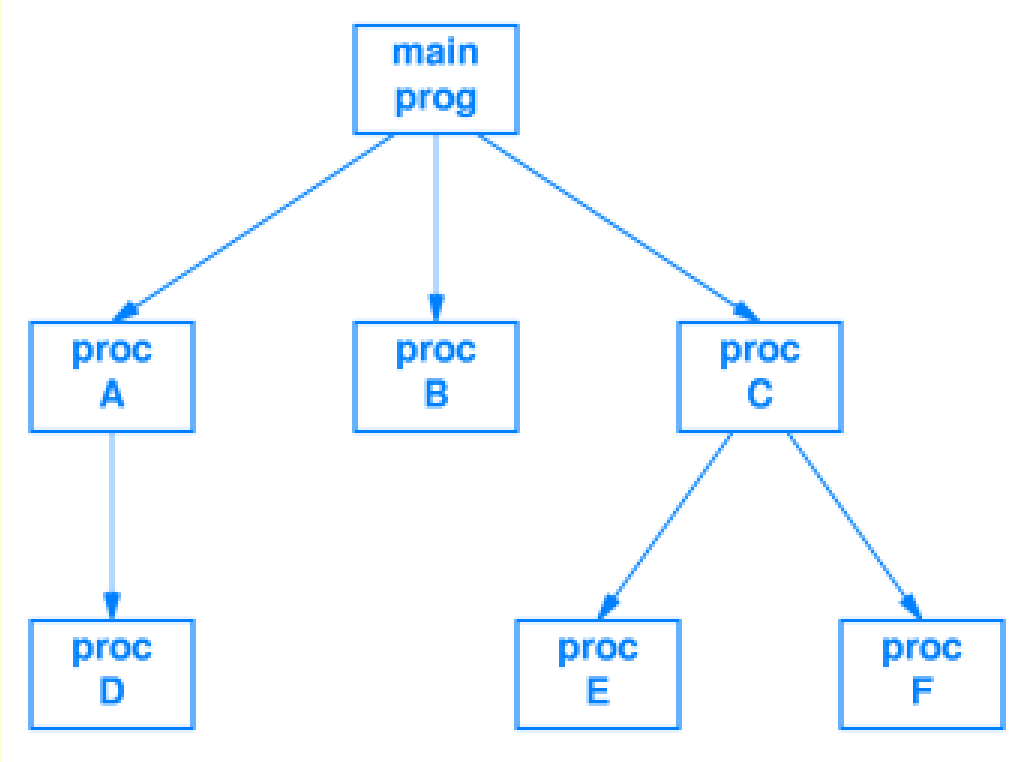


Poglavlje 22: Pozivanje udaljenih procedura – RPC i middleware

Sastavio: Robert Manger
22.12.2015

Potreba za RPC paradigmom (1)

- Većina programera naučila je pisati konvencionalne programe, koji rade na jednom računalu i sastoje se od glavnog programa i procedura (potprograma).



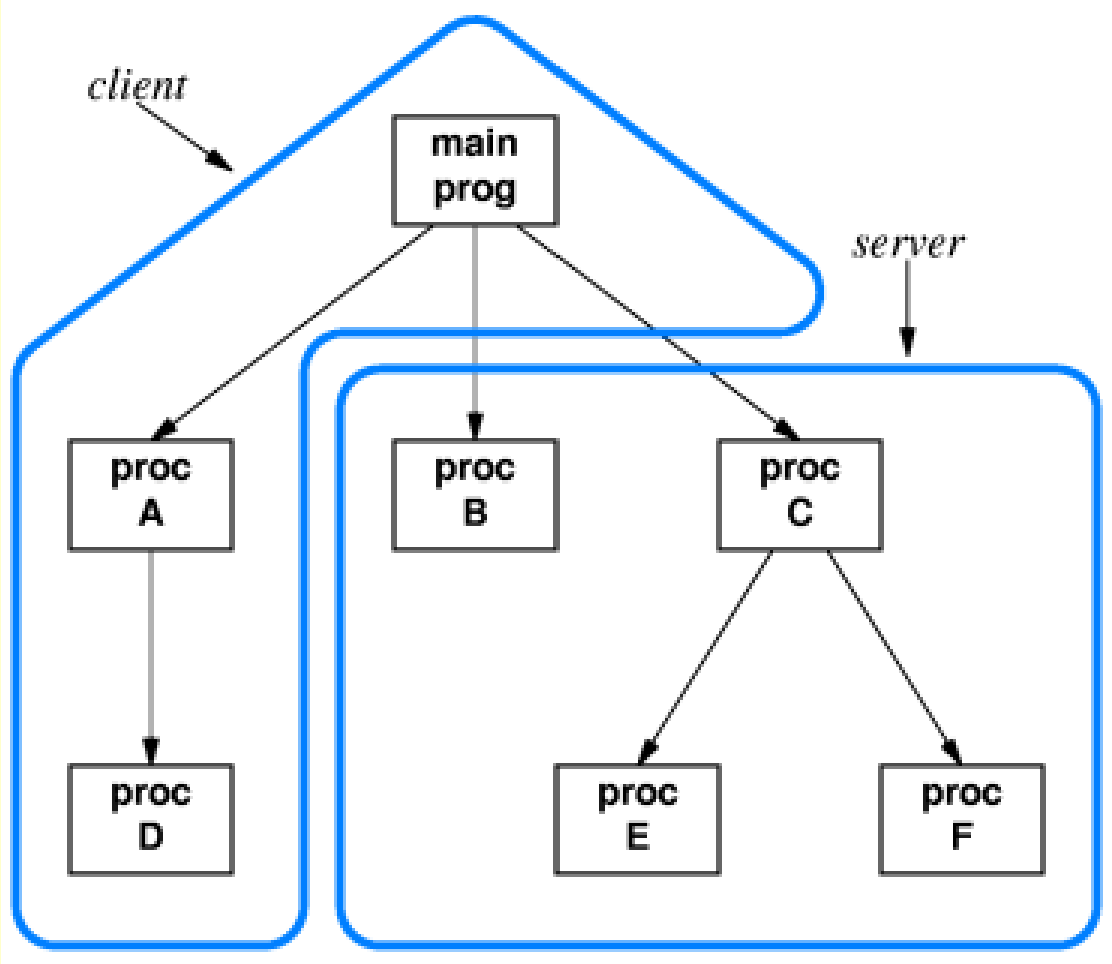
- Programi predviđeni za rad na mreži bitno su kompliciraniji budući da osim uobičajenih elemenata također moraju uključiti i komunikaciju s drugim programima.

Potreba za RPC paradigmom (2)

- Da bi se većini programera olakšao razvoj mrežnih aplikacija, u 1980-tim godinama nastala je paradigma *poziva udaljenih procedura* (Remote Procedure Call – RPC).
- Osnovna ideja RPC paradigme je sakrivanje eksplicitne mrežne komunikacije korištenjem uobičajenog mehanizma pozivanja procedura i prosljeđivanja parametara.
- RPC tehnologija dozvoljava da se dijelovi konvencionalnog programa naknadno rasporede na dva ili više računala.
- Programer se zato ne mora puno opterećivati mrežom i komunikacijskim protokolima. Umjesto toga, on se može koncentrirati na sam problem kojeg njegov softver treba riješiti.

Razvoj aplikacija pomoću RPC (1)

- Programer najprije razvija konvencionalni program koji radi na jednom računalu.
- Nakon toga, programer dijeli program u dva dijela. Dio s glavnim programom postat će klijent. Preostali dio postat će poslužitelj. Podjela uzima u obzir globalne podatke.



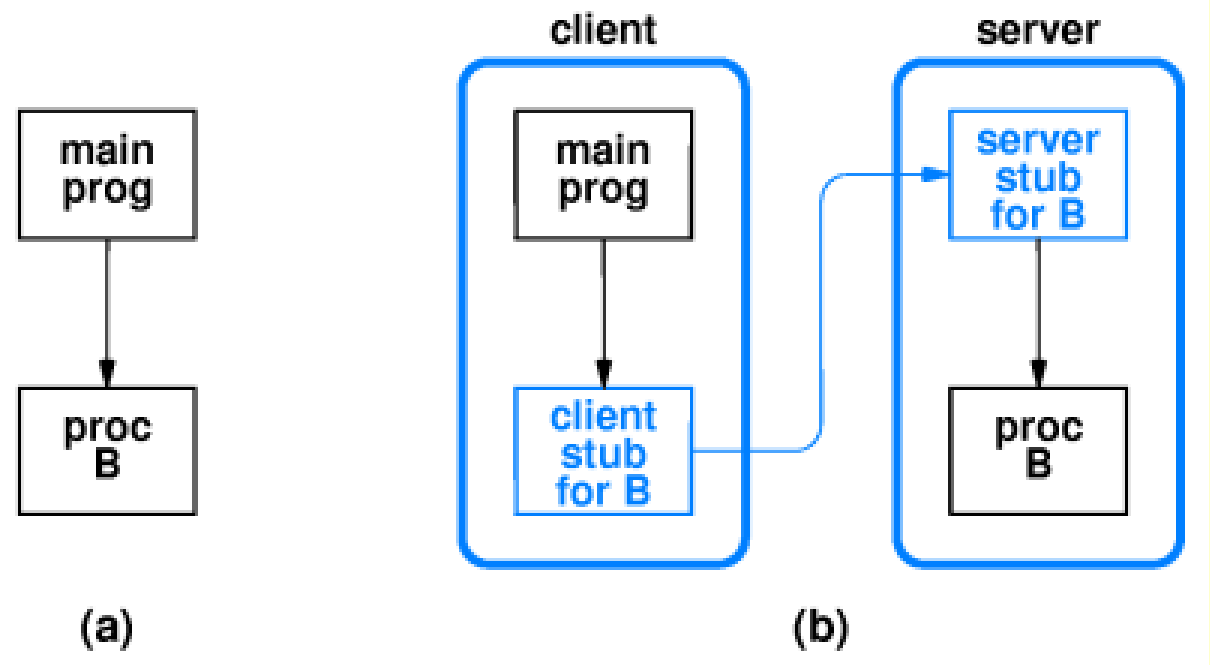
Razvoj aplikacija pomoću RPC (2)

- Dalje programer stvara specifikaciju koja opisuje način podjele programa, odnosno sučelje udaljenih procedura. Specifikacija se piše u *jeziku za definiranje sučelja* (Interface Definition Language – IDL).
- Odgovarajući RPC alat čita IDL specifikaciju i automatski generira programski kod koji omogućuje dijeljenje programa i prividno prebacivanje toka kontrole s jednog računala na drugo.
- Programer na kraju prevodi i povezuje vlastite programske jedinice s dijelovima koje je automatski generirao RPC alat, te tako dobiva klijentski i poslužiteljski program.
- Danas postoje brojni komercijalni RPC alati. Oni se jednim imenom nazivaju *middleware* budući da stoje "između" aplikacije i mrežnog softvera.

Komunikacijski umetci (1)

- Tok kontrole ne može zaista preskočiti iz programa na jednom računalo u proceduru na drugom računalu. Taj privid zapravo se postiže interakcijom između klijenta i poslužitelja.
- Preciznije, riječ je o komunikaciji između dijelova softvera koje je automatski generirao RPC alat, i koji se zovu *komunikacijski umetci* (communication stubs).
- Promotrimo na slici lijevo najjednostavniji slučaj glavnog programa koji poziva jednu proceduru B. Tu proceduru želimo prebaciti na drugo računalo. Nakon ugradnje komunikacijskih umetaka, klijent i poslužitelj izgledaju kao na slici desno.

Komunikacijski umetci (2)



- Klijentov umetak s jedne strane komunicira s poslužiteljem, a s druge strane simulira proceduru B (ima isto ime i parametre). Glavni program “misli” da i dalje poziva proceduru B.
- Poslužiteljev umetak s jedne strane komunicira s klijentom, a s druge strane simulira glavni program. Procedura B “misli” da nju i dalje poziva glavni prog.

Vanjski prikaz podataka (1)

- Osim prijenosa kontrole, pozivanje procedure uključuje i prijenos podataka. Najprije se iz glavnog programa parametri prenašaju u potprogram. Zatim se iz potprograma izračunati rezultati prenašaju u glavni program.
- Kod konvencionalnog poziva procedure unutar jednog računala podaci se prenašaju u internom (binarnom) formatu, dakle onako kako su prikazani u memoriji računala.
- Kod poziva udaljene procedure, podaci se moraju prebacivati preko mreže iz jednog računala u drugo. Ako bi se oni i dalje prenašali u internom formatu, to bi moglo stvoriti probleme jer razna računala koriste različite interne formate.

Vanjski prikaz podataka (2)

- Na primjer, neka računala spremaju najznačajniji byte 32-bitnog cijelog broja na najnižu adresu, a neka na najvišu.
- Većina RPC alata rješava spomenuti problem uvođenjem “vanjskog” (usklađenog) prikaza standardnih tipova podataka.
- Prije slanja, pošiljateljev komunikacijski umetak konvertira podatak iz pošiljateljevog internog prikaza u vanjski prikaz.
- Nakon primanja, primateljev komunikacijski umetak konvertira podatak iz vanjskog prikaza u primateljev interni prikaz.

Stariji alati za RPC

- *SUN RPC.* Službeno ime mu je Open Network Computing Remote Procedure Call – ONC RPC. Razvila ga je kompanija SUN Microsystems u 1980-tim godinama. Prvi RPC alat koji je stekao širu popularnost. Sadrži svoj IDL, protokol za komunikaciju i standard za vanjski prikaz podataka.
- *DCE/RPC.* Razvila ga je organizacija Open Software Foundation u ranim 1990-tim godinama, kao dio sveobuhvatne okoline Distributed Computing Environment – DCE. Definira svoj IDL. Dozvoljava klijentu pristup do više poslužitelja.
- *MS-RPC.* Razvio ga je Microsoft u 1990-tim na temelju DCE/RPC. Makar je po koncepciji sličan, u mnogim detaljima se razlikuje od DCE/RPC.

Objektno programiranje i distribuirani objekti (1)

- U 1990-tim godinama proširili su se objektno-orijentirani programski jezici poput C++, Java, C# i drugih, koji su omogućili razvoj objektnih programa.
- Za razliku od tradicionalnog programa koji se sastoji od procedura i globalnih podataka, objektni program građen je kao skup objekata. Svaki objekt je zasebna programska cjelina koja se sastoji od vlastitih podataka, te vlastitih operacija - tzv. metoda - koje djeluju nad tim podacima.
- Umjesto poziva procedura, u objektnom programu se kao osnovni kontrolni mehanizam pojavljuje *pokretanje metoda* (method invocation).

Objektno programiranje i distribuirani objekti (2)

- Kao posljedica promjena u programskim jezicima, u kasnim 1990-tim godinama došlo je do promjena u RPC paradigmi. Umjesto o pozivu udaljenih procedura, danas se govori o pokretanju metoda u udaljenim objektima.
- Arhitektura klijent-poslužitelj zamjenjuje se nešto općenitijom arhitekturom distribuiranih objekata, gdje pojedini objekt može igrati ulogu i klijenta i poslužitelja.
- Odgovarajući objektno-orijentirani middleware omogućuje transparentno raspoređivanje objekata koji čine program po računalima, te transparentnu međusobnu interakciju takvih distribuiranih objekata.

Objektno-orijentirani middleware (1)

- *CORBA.* Kratica znači: Common Object Request Broker Architecture. Skup standarda koje je definirao konzorcij Object Management Group – OMG – krajem 20. stoljeća. Postoje implementacije za UNIX, Linux i MS Windows. Slijedi spomenute ideje o distribuiranim objektima, s time da se komunikacijski umetci stvaraju dinamički tijekom rada programa.
- *DCOM.* Kratica znači: Distributed Component Object Model. Standard razvijen i implementiran od Microsofta 1998. godine, integriran u operacijske sustave MS Windows. Model distribuiranog računanja manje je općenit od CORBA-inog te je ograničen na Microsoftove platforme.

Objektno-orijentirani middleware (2)

- *.NET Remoting.* Microsoftov proizvod, objavljen 2002. godine. Predstavlja dio cjelovite .NET arhitekture. Predviđen je kao zamjena za stariji DCOM. Opet je riječ o “proprietary” tehnologiji koja je ograničena na Microsoftove platforme.
- *Java RMI.* Razvila ga je kompanija SUN Microsystems početkom 21. stoljeća. Kratica znači: Java Remote Method Invocation. Služi kao proširenje programskog jezika Java kojim se postiže pokretanje udaljenih metoda.

Najava kolegija na diplomskom studiju

- Distribuirani objekti detaljnije će se obraditi u kolegiju “Distribuirani procesi” na diplomskom studiju Računarstvo i matematika.
- U istom kolegiju govorit će se općenitije o distribuiranim procesima i algoritmima, o klasičnim problemima koji nastaju zbog distribuiranog načina rada, te načinima rješavanja tih problema.
- Obradit će se i objektno-orijentirani middleware poput CORBA, Java RMI, DCOM, .NET Remoting.