جلسه پنجم

## مدل فرایند ها:

### مدل Client/Server

**Client :** یک پردازه ای است که همیشه نیاز به اطلاعات دارد

**Server:** اطلاعات را در اختیار دارد و اگر کسی بخواهد آنها را در اختیار آن قرار می دهد که این تبادل اطلاعات با دو دستور
{Send();
Re suve();
انجام می شود.

هدف این مدل: اجرای برنامه کاربردی

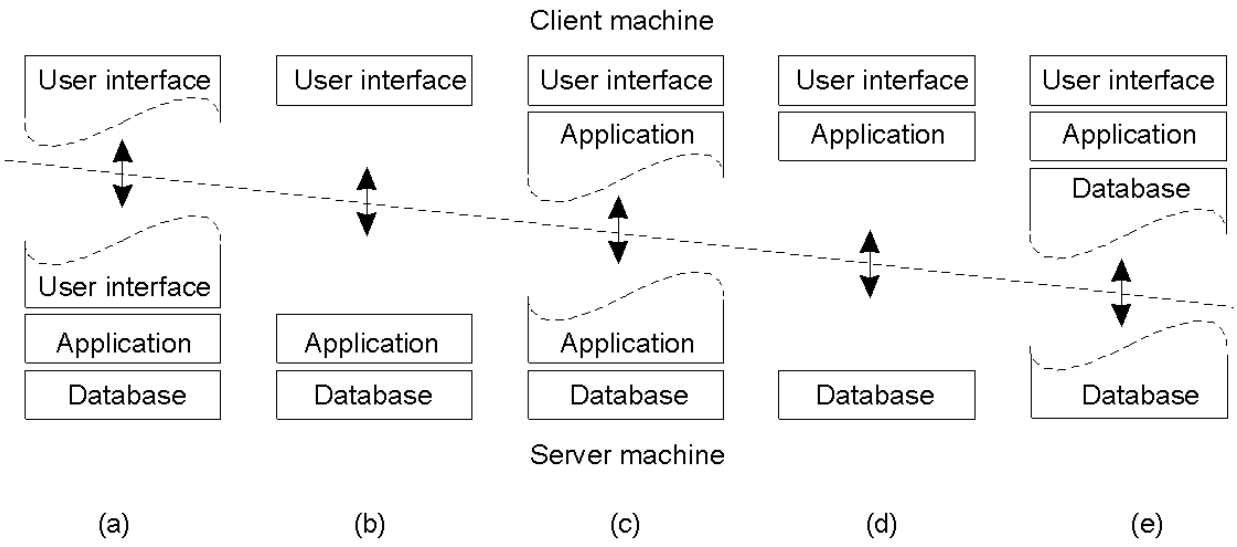
### لایه بندی Application

- User-interface
- Processing
- Data

مثال. موتور جستجو در اینترنت( Internet Search Engin)

Intenet Search Engin
-UI: string of keywords
-processing:DB queries
-Data: indexed web pages
-processing: rank, HTML generator

**چه قسمتی از Application را به Client و چه قسمتی را به Server بدهیم.**

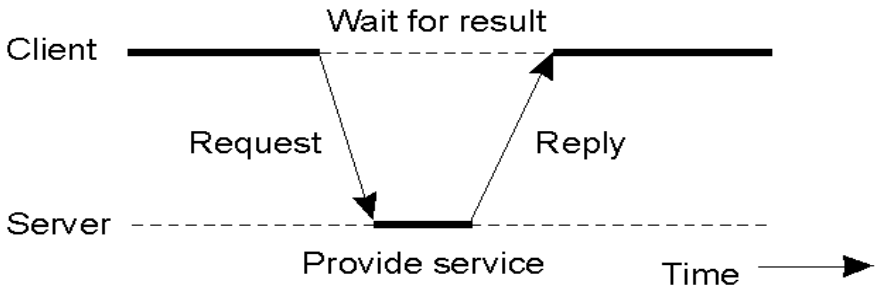



آیا می توان data را کلا روی Client قرار داد؟

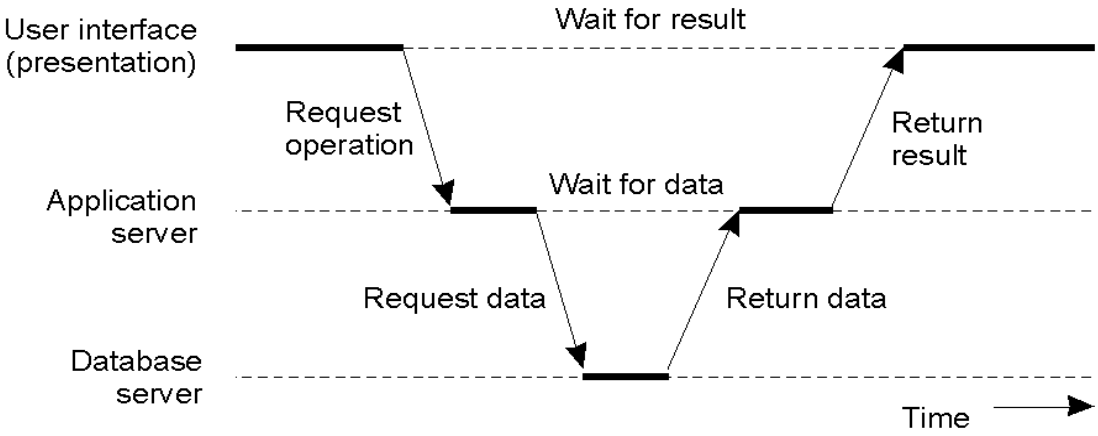
خیر. چون در آن صورت خود Client ، Server می شود.

**چه معماری هائی برای Client/Server وجود دارد.**

1- معماری دو لایه ای

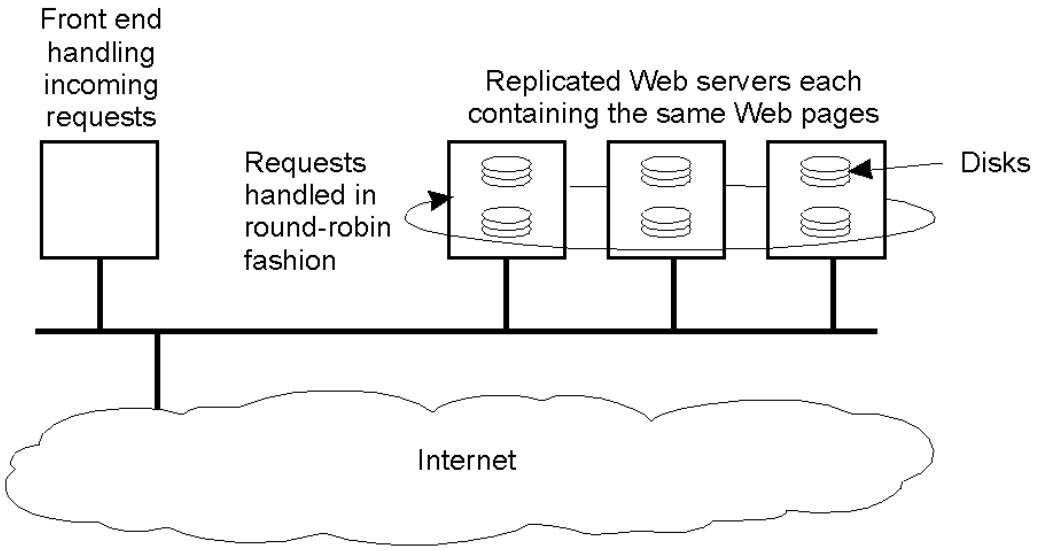



2- معماری سه لایه ای




در این معماری بعضی اوقات یک سرور به عنوان یک Client عمل می کند.

3- Modern Arch.(Horizental distribution)



معماری مدرن یک معماری افقی(Horizontal) می باشد در این معماری یک کلاینت یا سرور ممکن است به صورت فیزیکی به چند قسمت منطقی همسان تقسیم شود، که هر قسمت مستقل عمل می کند و بنابر این توازن بار ایجاد می شود. مثالی برای این نوع معماری یک Web Server می تواند باشد(شکل صفحه قبل ملاحظه شود).

**مزیت معماری Modern Arch را بیان کنید.**

- اشتراک بار(توازن بار)
- قابلیت اطمینان
- تحمل خطا هم زیاد می شود.

چگونه Client ها با Server ها در ارتباط هستند؟ از طریق ارتباط گروهی

گروه: چند پردازه که با هم همکاری و تعامل داشته باشند تشکیل گروه می دهند.

انواع گروه:

باز : هر یک از کلاینت ها از بیرون می توانند پیغام دهند.

بسته: کسی نمی تواند از بیرون پیغام دهد.(در پردازش موازی کاربرد دارد مثل شطرنج)

Group structures

1) peer group
2) hierarchical group

نکته: Client/Server مربوط به گروه باز است

ساختار های گروه:

- ساختار یکسان: تمام اعضای گروه مثل هم هستند ⇐قابلیت اطمینان بالا می رود ولی مدیریت مشکل می شود.
- ساختار سلسله مراتبی: مثل درخت ⇐ قابلیت اطمینان پایین می آید ولی مدیریت راحت می شود.

عضویت در گروه: اگر کسی عضو گروه باشد تمام پیام های رد و بدل شده بین گروه را دریافت کند، بعد از این که گروه را ترک کرد هیچ پیامی دریافت نکند که برای عضو شدن در گروه دو روش وجود دارد.

- متمرکز: یک گروه سرور وجود دارد(سرگروه) که به صورت نرم افزاری پیاده سازی می شود. به عبارتی لیست اعضای گروه در گروه سرور وجود دارد و از طریق گروه سرور اعضای گروه شناخته می شوند.
- توزیع شده: چند گروه سرور وجود دارد.

**انواع آدرس دهی:**

- Broad cast : اگر یکی پیغام بفرستد همه اعضا می گیرد.
- Multi Cast : فقط اعضای خاص می تواند پیغام را بگیرد.
- Uni Cast : اگر برای همه بخواهد بفرستد بایستی تک تک به همه بفرستد.

اتمیک : یا همه اعضای گروه پیام را دریافت کنند یا هیچ کدام.

از کجا بدانیم پیام به همه اعضای گروه رسیده است؟

بایستی به تعداد اعضای گروه Atack دریافت کند اگر دریافت نکند بایستی پیام از بین برود.

**ترتیب پیام ها :** پیام های دریافتی بایستی به ترتیب باشند.

- اگر کلاک واحد داشته باشیم: روی هر پیام مهر زمانی قرار می دهیم تا پیام ها ترتیب داشته باشند.

- اگر کلاک یکسان نداشته باشیم: که اکثرا در MC یکسان نیست راه حل استفاده از ساعت منطقی است

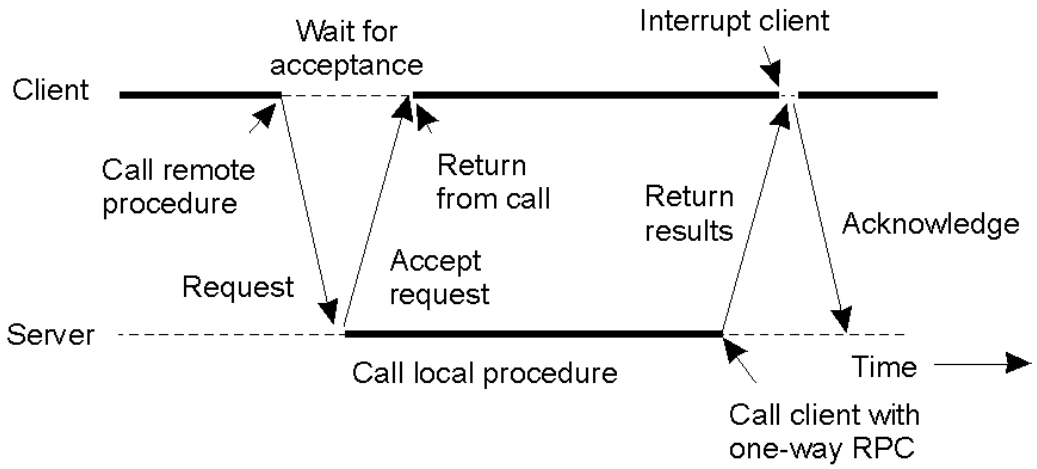
**همپوشانی گروه ها:** یعنی ممکن است یک پرداز عضو چند گروه باشد.

**قابلیت کسترش:**

اگر اعضای گروه کم باشد طبیعتا تعداد پیام های ردوبدل شده نیز کم خواهد شد و لی با افزایش اعضای گروه تعداد پیام های ردوبدل شده به صورت نمائی افزایش می یابد به عبارت دیگر با بزرگ شدن شبکه ترافیک ایجاد خواهد شد راه حل پیشنهادی برای جلوگیری از ترافیک استفاده از روش خوشه بندی می باشد. منظور این که اعضای هر گروه با سرگروه ارتباط داشته باشد و سرگروه ها هم با هم در ارتباط باشند و از این طریق اعضای یک گروه با عضای گروه دیگر می تواند ارتباط پیدا کند.

نکته: روش Client/server یک روش همزمان بود یعنی پردازه درخواست کننده تا زمانی که پاسخ دریافت نکرده منتظر می ماند و هیچ کاری انجام نمی دهد.

شکل زیر روش غیر همزمان را نشان می دهد که در آن کلاینت در زمانی که سرور، در حال مهیا سازی سرویس است مشغول انجام یک کار دیگر است و منتظر دریافت پاسخ نیست.




**آدرس دهی:** کلا با دو دستور ()Send و ()Resive انجام می شود.

فرض کنید یک کلاینت می خواهد یک Request به سرور بدهد آدرس سرور را چگونه پیدا کند؟

1- آدرس سرور را داشته باشد:

Request(آدرس سرور،...) عیب: نقض مخفی سازی، پس روش درستی نیست

2- Broadcast : عیب: ترافیک بالا و سربار بالائی دارد.

3- Static Binding : در این روش جدول Name Server به صورت دستی پر می شود که شامل موارد زیر است

که از طریق این جدول کلاینت ها می توانند به سرو رها دسترسی داشته باشند

لازم به ذکر است که همه کلاینت ها ادرس Name Server را دارند

4- Dynamic Binding : هر سرور که ایجاد می شود مشخصاتش را در Name Server ثبت می کند و سرور هنگام ترک یا عدم ارائه سرویس Deregister می کند. کلاینت آدرس را از طریق یک تابع بدست می آورد

| نام سرور | آدرس | Port | تاریخ | نسخه |
|---|---|---|---|---|
| | | | | |

Bloking :

**Buffering :**

- اگر سرور بافر نداشته باشد ممکن است در حین پاسخ به یک درخواست درخواست های دیگر را از دست دهد.
- اگر بافر وجود داشته باشد درخواست های دیگر در بافر ذخیره می شود یک تابع Listen() می نویسیم و بعد از بافر Accept می کنیم.

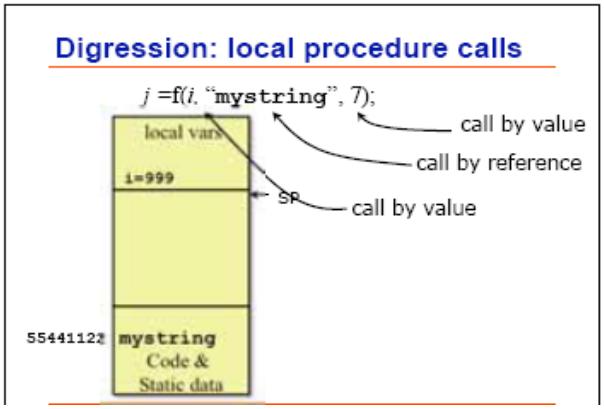
**Relability :** قابلیت اطمینان

Client از کجا می فهمد درخواستش به سرور رسیده است؟

بایستی سرور به کلاینت Ac بدهد

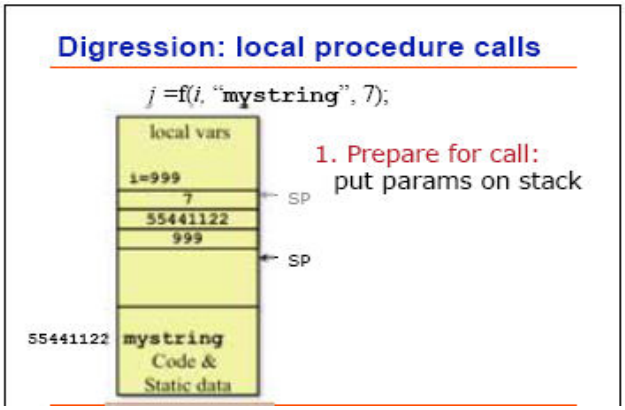
Server از کجا می فهمد درخواستش به Client رسیده است؟

میکروکرنل کلاینت به میکرو کرنل سرور Ac می دهد.

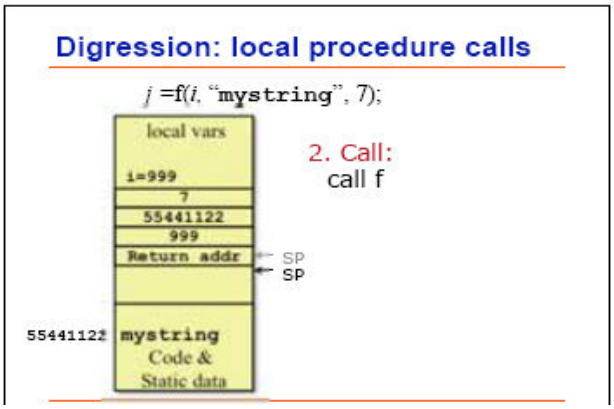



**فراخوانی محلی:** 5 مرحله دارد.

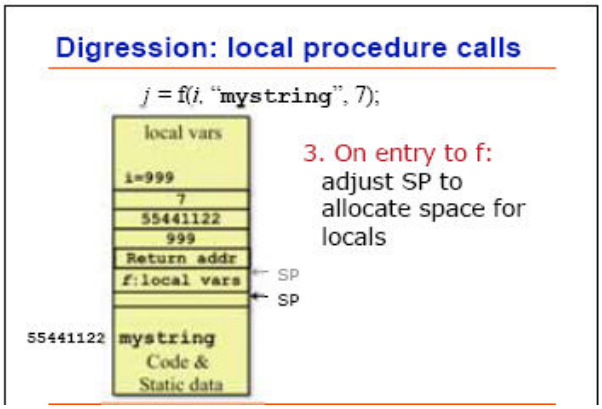
مرحله اول: آرگومان های ورودی را به Stack می فرستد



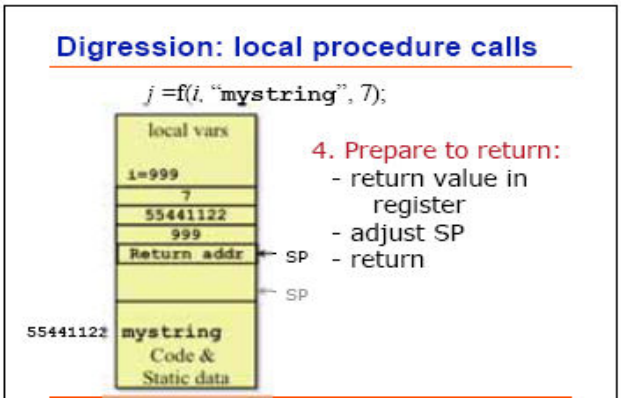

مرحله دوم: وقتی فراحوانی صورت می گیرد باید آدرس برگشت را ذخیره کند



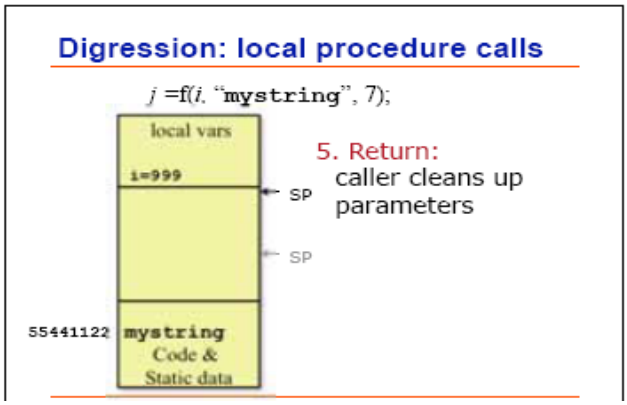
مرحله سوم: SP را طوری تنظیم می کنیم که به آرگومان ها دسترسی داشته باشد و متغیر های محلی هم در پشته قرار بگیرد




مرحله چهارم: آمادگی برای بازگشت




مرحله پنجم: برگرداندن نتیجه



**پیاده سازی RPC (فراخوانی راه دور) :**

هیچ معماری برای پیاده سازی RPC وجود ندارد، فراخوانی های راه دور را با استفاده از فراخوانی های محلی شبیه سازی می کنند و این شبیه سازی هم در سطح زبان برنامه نویسی است (به جای سطح سیستم عامل)

Client Stub : فراخوانی F را داخل بسته قرار می دهد.(پارامتر ها را در داخل پشته قرار می دهد)

مارشال: قرار دادن پارامتر ها در بسته

Unmarshal : تبدیل پارامتر ها(فراخوانی) به رشته

**مزیت RPC :**

نوشتن برنامه نویسی را راحت می کند.

- نیاز ندارد برنامه نویس به جزئیات کد نویسی شبکه آشنا باشد
- کد شبکه را با قرار دادن آنها در توابع Stub مخفی می کند.

تذکر: RPC همان لایه ارائه است در مدل OSI