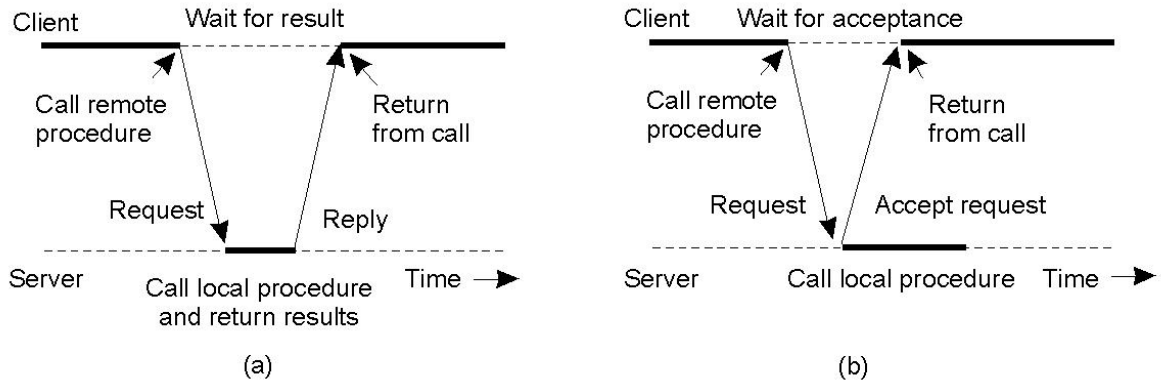
RPC غیر همگام (Asynchronous RPC)

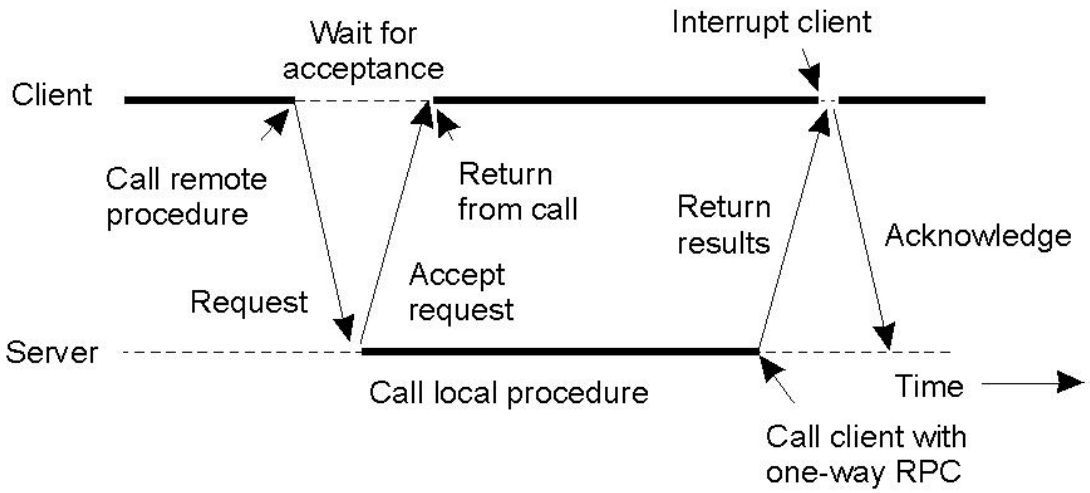
یکی دیگر از مفاهیم RPC توسعه یافته ، RPC غیرهمگام می‌باشد. در واقع تنها چیزی که در RPC غیرهمگام تغییر می‌کند این است که کلاینت بلافاصله اجرای خود را بعد از فراخوانی Remote Procedure دنبال می کند و سرور بلافاصله پس از آماده شدن پاسخ، آن را به کلاینت ارسال می‌کند. جهت بررسی مفهوم دقیق تر Asynchronous RPC به شکل دقت کنید.



در شکل اول یک RPC سنتی نشان داده شده است، با توجه به شکل، در RPC سنتی کلاینت فراخوانی را انجام می دهد و منتظر می ماند، در این لحظه، سرور، فراخوانی محلی انجام می دهد و پاسخ را به کلاینت ارسال می کند و به محض دریافت پاسخ، کلاینت می تواند فعالیت خود را از سر گیرد. اما در RPC غیر همگام، کلاینت درخواست را ارسال می کند، اما مدت زمان انتظار آن در این حالت کمتر از دفعه قبل خواهد بود، چرا که در اینجا تنها کلاینت منتظر می ماند تا پذیرش درخواست خود را بگیرد و انتظار برای دریافت پاسخ نیست. بنابراین پس از اینکه در خواست ارسال شد و فراخوانی مورد نظر صادر شد، کلاینت در واقع مدت زمان کمتری را منتظر می ماند و می تواند پس از آنکه پذیرش درخواست خود را داشت کارهای دیگر خود را انجام دهد.

در شکل بعد این مفهوم با دقت بیشتری نشان داده شده است. با توجه به شکل، بعد از پذیرش درخواست، client می‌تواند کارهای خود را انجام دهد و همزمان در سمت سرور فراخوانی به زیر برنامه محلی انجام می شود و پس از اینکه پاسخ مهیا شد، پاسخ مورد نظر در قالب یک پیغام به client ارسال می شود و

کلاینت در حالی که کارهای دیگر خود را انجام می دهد پاسخ مربوط به درخواستی که قبلاً ارسال کرده است از طریق یک interrupt دریافت خواهد کرد و پس از این دریافت، تأیید دریافت پاسخ را به سمت سرور ارسال خواهد کرد. استفاده از این نوع RPC باعث می شود که کلاینت هایی که عملیات پردازشی بسیار زیادی دارند بتوانند به نوعی برخی عملیات پردازشی خود را با استفاده از RPCها به صورت پارالل انجام دهند و به این صورت می توانند زمان پاسخ خود را به مشتری های مختلف و درخواست های مختلف کاهش دهند.



بیدار سازی Object های راه دور (Remote Object Invocation)

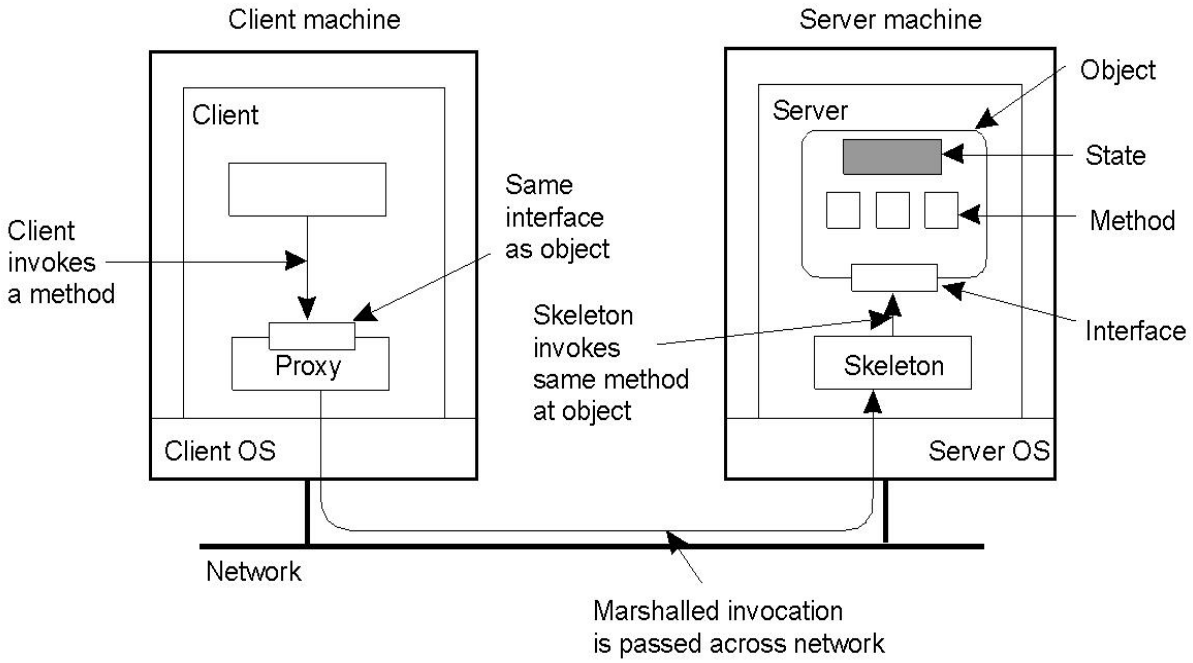
مفهوم دیگری که به عنوان یکی از مدل های مربوط به ارتباط توضیح داده خواهد شد، بحث Remote Object Invocation یا بیدار سازی Object های راه دور است. Objectها مفاهیمی هستند که ساختار داخلی خود را از بیننده مخفی می کنند. این امر باعث می شود که سطح دسترسی بالایی ایجاد شود و تنها می توان Object هایی را که Interfaceهای یکسانی دارند؛ یعنی واسط یکسانی برای آنها تعریف شده است به سادگی با یکدیگر جابجا کرد. بنابراین زمانیکه عنوان می‌شود ساختار داخلی Objectها مخفی است در واقع یک مزیت در Object حساب می‌شود و سطح Transparency را افزایش داده است. همانطور که عنوان شد، مخفی بودن سطح Object ها و مخفی بودن داخل Object ها این مزیت را می

دهد که Object هایی که ممکن است ساختار داخلی کاملاً متفاوتی داشته باشند را با یکدیگر جابجا نماییم. منوط بر اینکه واسط آنها یعنی نحوه استفاده از Objectها مشابه هم باشند.

می توان ایده RPC را بر روی Object ها بکار برد. برای این کار باید برخی از مفاهیم مطرح در Objectها را مورد استفاده قرار داد. معمولاً یکی از مفاهیم مطرح در Object ها، Object state یا وضعیت یک Object است. در واقع از نظر ما Object state داده های موجود در یک Object یا متغیرهای تعریف شده در داخل Object است که همانطور که عنوان شد معمولاً از دید کاربر بیرونی مخفی است و مفهوم دیگر Object ها، Object Interface یا واسطهای Object ها می‌باشد. این مفهوم مجموعه ای از متدها یا زیر برنامه هایی هستند که در داخل Object قرار دارند و دسترسی به داده های Object از طریق آنها انجام می شود و معمولاً کاربران بیرونی این متدها را می بینند. بنابراین دسترسی قانونی و منطقی به یک Object از طریق متدهای آن و فراخوانی متدهای مربوط به آن انجام می شود. فراخوانی متدهای یک Object را معمولاً Invoking می نامند؛ در واقع به نوعی در اینجا سعی می شود بین فراخوانی های معمولی که معادل On call است و Invoking تفاوت قائل شد. بنابراین هر گاه از کلمه Invoke استفاده شد به این معنی است که متد مربوط به یک Object فراخوانی می‌شود. وجود Object باعث می‌شود که بتوان وجود Objectهای توزیع شده را داشت. به این معنی که ممکن است Interface یک Object یعنی تعاریف مربوط به متدهای آن بر روی یک ماشین و پیاده سازی و بدنه مربوط به Object و متدهای آن بر روی ماشین دیگر قرار گیرد. چنین Objectهایی را Objectهای توزیع شده می نامند. حال Remote object invocation خود نیازمند مفاهیمی است که گرچه این مفاهیم بسیار شبیه به مفاهیم RPC است، اما سعی می‌شود به طور کلی آنها را بررسی کنیم.

اجزای ROI (Remote Object Invocation)

در شکل زیر اجزای لازم برای یک Remote object invocation نشان داده شده است.



با توجه به شکل، فرض شده است یک ماشین کلاینت و یک ماشین سرور وجود دارد و objectی بر روی ماشین سرور وجود دارد که این object دارای مجموعه‌ای از داده ها به نام state و مجموعه ای از متدها می‌باشد. و یک Interface که لیست متدهای مربوط به این object خواهد بود. بدنه متدها مستقل از Interface است. در ROI مفهومی به نام Skeleton وجود دارد. Skeleton را می‌توان معادل server stub در RPC نامید. مفهوم دیگری به نام proxy که مفهوم نوینی در ROI است نیز وجود دارد. proxy در واقع یک پیاده سازی از واسط object است، به نوعی واسط‌های objectهای توزیع شده بر روی ماشین های کلاینت نیز قرار می گیرند که آنان را proxy می نامیم. Invocation در اینجا به معنی فراخوانی متدها است که معمولاً با استفاده از Marshal method انجام می‌شود. با توجه به مطالبی که قبلا عنوان شد، Marshal method پارامترها را به صورت Value در داخل پیغام ها Pack می کرد. پاسخ ها unmarshal هستند و بنابراین با توجه به این ساختارها می‌توان رویه مربوط به ROI را به این

شکل عنوان کرد که کلاینت، درخواست مربوط به Invoke کردن یک متد که در سطح یک object توزیع شده است را از طریق proxy و با اتفاقاتی که بسیار شبیه RPC است مدیریت می کند. می توان به نوعی عملیات انجام شده در سطح proxy را نیز مشابه client stub نیز دانست. اما همان طور که عنوان شد، علاوه بر عملیات مربوط به Client Stub، پروکسی یک پیاده سازی از Interface مربوط به objectها را نیز در خود دارد.

اشیاء ROI (ROI Object)

در سطح ROI، objectهای مختلفی تعریف می‌شوند. از جمله این objectها، objectهای، Compile time هستند که مفاهیم مربوط به برنامه نویسی هستند. Objectهای Runtime، object هایی هستند که ممکن است از زبانهای مختلفی ایجاد شده باشند و در زمان اجرا به این برنامه‌ها می‌پیوندند. از جمله مفاهیم مبتنی بر Runtime objectها می توان به ابزارهای موجود مانند Com اشاره کرد. در سطح ROI مفهوم دیگری نیز وجود دارد که آن را به عنوان Object adapter می‌نامیم. در واقع این مفهوم اجازه می دهد که بتوان Interface مربوط به یک object را به گونه ای تبدیل به ساختاری کرد که توسط یک کلاینت قابل درک و فهم باشد. از آنجایی که objectها ممکن است با زبان‌های مختلفی طراحی شده باشند و در سطح شبکه توزیع شده باشند، ممکن است ساختارهای مورد نیاز برای فراخوانی objectها به گونه‌ای باشد که در یک زبان دیگر قابل استفاده نباشد. در اینجا ضرورت Object adapter به عنوان کسی که این ساختارها را به هم تبدیل می‌کنند احساس می شود. Objectهای ایستا یا Persistent، objectهایی هستند که به سرورها بستگی ندارند و Transient object در واقع Objectهایی هستند که به سرور بستگی دارند و به محض اینکه سرور از بین برود این Objectها نیز از بین خواهند رفت.

مفهوم دیگری که در ROI به آن تکیه می شود مفهوم Object reference است. Object reference ها به نوعی مشابه پارامتر referenceها در RPC هستند.Object reference ها می‌توانند به آسانی در بین فرایندهای مختلف ماشین‌های مختلف حرکت کنند. وقتی که یک فرایند یک Object reference را در اختیار دارد قبل از اینکه بخواهد این Object reference، Invocation یا فراخوانی انجام دهد باید آن را Bind کرد. Bind کردن به معنای اتصال Object reference به یک Object واقعی است. در

واقع Object reference به نوعی آدرس یک Object است که در هنگام اتصال این آدرس اعتبارسنجی می شود و شایددرصورت نیاز تغییر کند. در هنگام عملیات Binding ، پروکسی ایجاد می‌شود. با توجه به این توضیحات می‌توان عنوان کرد که Object reference باید شامل آدرس شبکه ای مربوط به مکانی که Object واقعی قرار گرفته است باشد. آدرس یا شماره پورت مربوط به سرور را داشته باشد و شاخصه های مربوط به Object را نیز در خود نگهداری کند.

معایب و نقاط ضعف سیستم‌های ROI

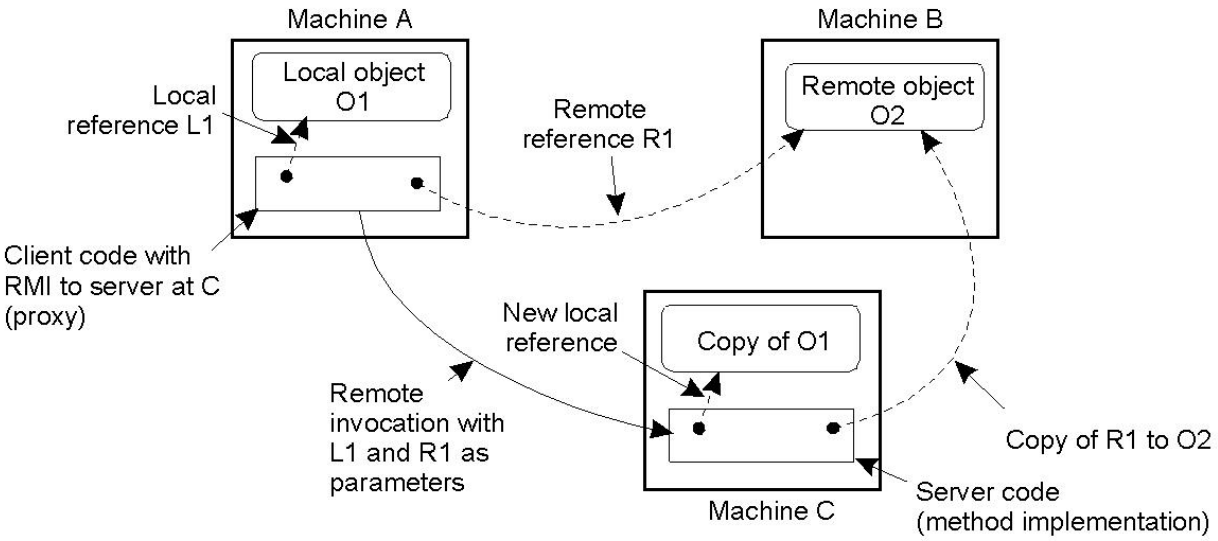
نکته ای که باید به آن اشاره کرد و یکی از نقاط ضعف ROI است این است که با چنین ساختاری، از بین رفتن یک ماشین سرور ممکن است باعث ایجاد مشکلاتی شود. چرا که از با بین رفتن ماشین سرور شماره پورت جدیدی بعد از Recoverای آن ممکن است به آن نسبت داده شود و در واقع این باعث می شود که تمامی Object reference ها نامعتبر شوند. راه حلی که برای این مفهوم ارائه می شود استفاده از Location server است. Location server ها یا جداول مکان یابی، جداولی هستند که آدرس مربوط به سرورها را نگهداری می کنند و ردیابی می کنند که Object server مربوط به یکObject در کجا قرار دارد. نکته دیگری که در ROI باید به آن دقت شود این است که کلاینت ها و سرورها در این روش لزوماً از Protocol stack مشابهی استفاده نمی کنند، یعنی ترتیب Coding و decoding پیغام های آنها لزوماً برعکس هم نیست. در واقع باید اطلاعات بیشتری را به Object referenceها اضافه کرد. از جمله انها می‌توانbinding protocol Object Reference و شماره مربوط به سرور را نام برد. با این اوصاف یکی از وظایف client این است که یک پیاده سازی دقیق از پروکسی داشته باشد که حداقل بتواند یکی از پروتکل های موجود در Object reference را به اجرا درآورد و ابزارهای لازم برای پیاده سازی را در Object reference قرار دهد. پیاده سازی پروکسی باید به گونه ای باشد که client بتواند ابزار لازم را در هنگام binding به صورت داینامیک، Load کرده و مورد استفاده قرار دهد. با این اوصاف می‌توان گفت که ROI بستری برای RMI فراهم کرده است. به عبارت دیگر چنانچه متدهای مربوط به یک Object فراخوانی شوند، با استفاده از بستر Remote object، بستر Remote method فراهم شده

است. لذا RMIمفهوم بسیار شبیه RPC است که مفاهیم marshaling و parameter passing را دقیقاً مانند مفاهیم مطرح شده در RPC به کار می برد.

معمولاً در RMI دو نوع Invocation وجود دارد: یکی Static invocation یا invocation ایستا است که در آن از واسط های از پیش تعریف شده استفاده می شود. دیگری Dynamic invocation است و کاری که انجام می شود این است که Invocation کاملاً در زمان اجرا برنامه ریزی می شود و با توجه به اتفاقاتی که به صورت پویا در شبکه افتاده است Dynamic invocation صورت می گیرد.

ارسال پارامتر در RMI

ارسال پارامتر در RMI مبتنی بر مفهوم Object reference انجام می‌شود. ارسال پارامترها در RMI در هر دو حالت ارسال پارامتر با مقدار و ارسال پارامتر با ارجاع امکان پذیر است. شکل زیر نمونه خوبی است برای آنکه نکات مربوطه را روشن سازد. با توجه به شکل، سه ماشین A,B,C وجود دارند که قصد دسترسی به Objectهای O_2 و O_1 را دارند. با توجه به مطالب پیشین، RMI در واقع زیر مجموعه ای از ROI است، بنابراین می توان به نوعی مفهوم را در سطح Remote object نیز بیان نمود.



در ماشین A، O_1 object در واقع یک object محلی است و در ماشین B نیاز به دسترسی به object O_2 وجود دارد. با توجه به آنچه در شکل دیده می‌شود، فرآیندی که بر روی ماشین A در حال اجراست به صورت محلی به object O_1 دسترسی دارد و با استفاده از تکنیک مربوط به Object reference ها به صورت Remote به object O_2 نیز دسترسی دارد. object O_2 بر روی ماشین B قرار دارد و در عین حال فرآیندی که بر روی ماشین C در حال اجراست نیازمند دسترسی به object O_1 ای است که بر روی ماشین A قرار دارد. در اینجا دو تکنیک می توان بکار برد، یعنی امکان دسترسی به صورت remote Access و با استفاده از Object reference به Object ی که بر روی ماشین O_1 وجود دارد و یا انتقال یک کپی ازO_1 object از روی ماشین A به ماشین C خواهد بود. بدیهی است که شکل نشان می دهد که فرآیند اجرا شده بر روی ماشین C به صورت remote در واقع به object O_2 که بر روی B قرار گرفته است، دسترسی دارد.

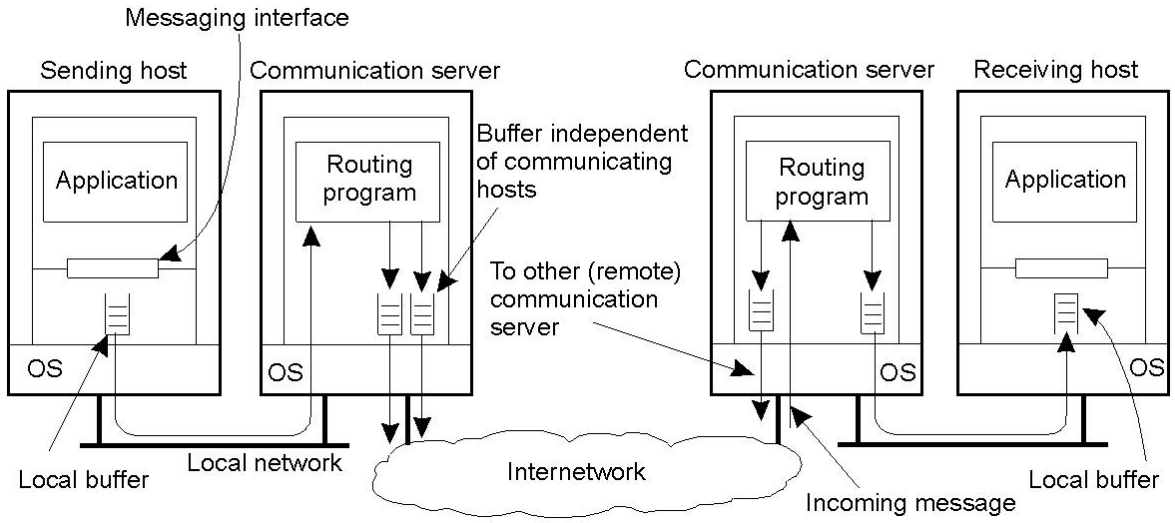
بنابراین می توان گفت، در دسترسی به متدهای یک Object یا می توان به صورت متدهای محلی با آنها برخورد کرد و یا می توان از مفهوم Object reference ها استفاده کرد. در واقع اگر به صورت متدهای محلی با آنها برخورد شود باید object مورد نظر به صورت یک پارامتر به ماشین مقصد منتقل شود که این عمل برای برخی از objectها مقرون به صرفه و برای برخی دیگر مقرون به صرفه نیست. به عنوان مثال، برای objectهای کوچک که به کرّات مورد دسترسی واقع می‌شوند، این انتقال، انتقال صحیحی نیست، چرا که سربار سیستم را افزایش خواهد داد. برای دسترسی به objectهای راه دور معمولاً بهترین ابزار، استفاده از Object reference خواهد بود.

Local object ها نیز به شکل Objectهای محلی رؤیت می شوند که انتقال آنها از یک ماشین به یک ماشین دیگر انجام می شود. بدیهی است که انتقال Objectها از یک ماشین به ماشینی دیگر عملیاتی است که سربار بیشتری خواهد داشت، ولی پس از انتقال، سرعت دسترسی به Object احتمالاً بیشتر خواهد بود. بنابراین برای اینکه بدانید کدام یک از این ابزارها را مورد استفاده قرار دهید، این امر بستگی به وضعیت شبکه، ترافیک آن و در واقع کارایی سیستم ها خواهد داشت.

ارتباط مبتنی بر پیغام (Message Oriented Communication)

از مباحث عنوان شده در این بخش، که برای ارتباط بین فرآیندها مورد بررسی قرار می گیرد و یک مفهوم عمومی است، بحث ارتباط مبتنی بر پیغام است که جنبه های مختلف و نکات مختلف در این مبحث به تفکیک و با جزئیات بیشتری مورد بررسی قرار می‌گیرد.

با توجه به شکل، برای یک سیستم مبتنی بر پیغام و برای برقراری ارتباط، اجزایی مانند communication server به سیستم اضافه می شوند.



در واقع در این شکل می بینید که دریافت کننده و ارسال کننده از طریق یک communication server یا یک خدمتگزار ارتباط که بر روی بستر شبکه محلی آنها قرار گرفته است، مدیریت ارتباط را انجام می دهد. در واقع پیغامی که قرار است از سمت ارسال کننده به هر مقصدی ارسال شود در یک مدل عمومی ابتدا به communication server می‌رود. communication server با استفاده از برنامه های Routing یعنی برنامه های مسیریابی، مسیر مورد نظر را در سطح شبکه بزرگتری پیدا کرده و آن را به communication server دیگری که در سطح شبکه محلی دیگری ممکن است نصب شده باشد ارسال می کند و سپس با تکنیک های معکوس در واقع دریافت کننده مشخص شده و پیغام به سمت او

ارسال می شود. همانطور که در شکل می بینید بحث Buffering در این ارتباطات بسیار مهم است و علاوه بر اینکه ارسال کننده دارای یک بافر محلی است و از یک interface مربوط به پیغام برای مدیریت پیغام‌ها و ارسال دستورهای مختلف مربوط به پیغام استفاده می‌کند، communication server نیز دارای مجموعه ای از بافرها است که این بافرها مستقل از نحوه ارتباط تعبیه شده اند و در واقع برای نگهداری پیغام هایی هستند که به سمت سرورهای دیگر ارسال می‌شوند. همین پدیده در مورد communication server که در سطح دریافت کننده است نیز وجود دارد. با توجه به شکل، communication server که در سطح دریافت کننده است خود می‌تواند یک پیغام را دریافت کرده و برای یک سرور دیگری ارسال کند. در واقع communication server یک ماشین است که وظیفه ارتباط بین تمامی اجزاء شبکه را بر عهده دارد و این ارتباط می تواند، Send و یا Receive باشد. در سطح دریافت کننده نیز یک بافر محلی به همراه واسط مربوط به ارسال پیغام تعبیه شده است، این مدل یک مدل کلی و عمومی است که بسیاری از جزئیات در آن لحاظ نشده است و حال با بررسی نکات مختلف در سیستم های ارسال پیغام جزئیات مهمتر این مدل معلوم می‌شود.

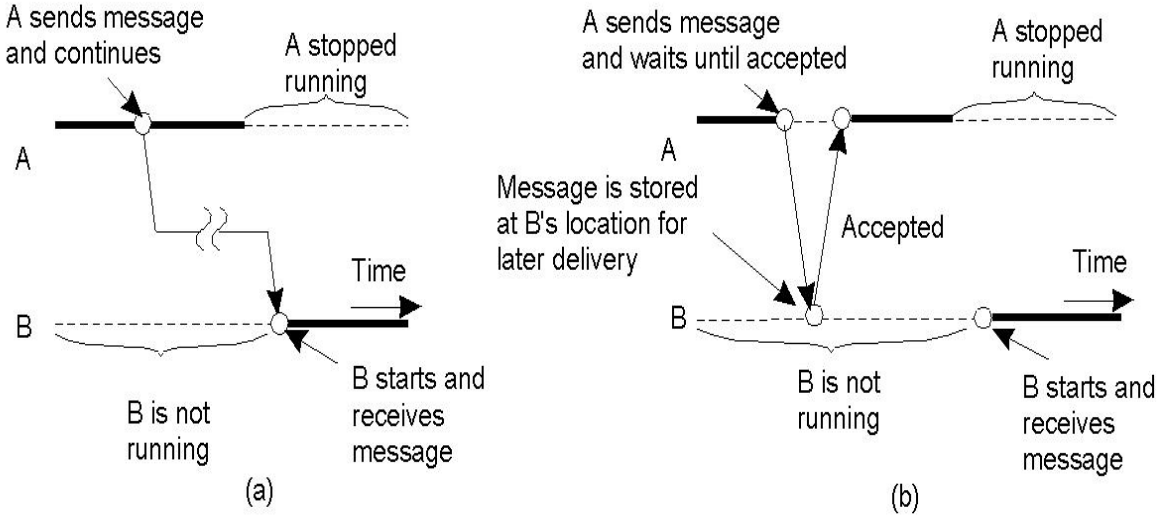
نکته‌ی حائز اهمیت این است که ارتباط بر اساس پیغام به دو گونه صورت می‌گیرد:

۱- ارتباط ماندگار یا Persistent

۲- ارتباط گذرا یا Transient.

در ارتباط ماندگار یک پیغام توسط سیستم ارتباطی نگهداری می شود تا وقتی که بتواند به سمت دریافت کننده ارسال شود به عبارت دیگر بطور حتم پیغام تا ارسال در سیستم باقی می‌ماند. معنی سیستم Persistent هنگامی با دقت بیشتری مشخص می شود که معنی سیستم Transient یا گذرا را بدانید. در سیستم Transient یک پیغام توسط سیستم communication system یا سیستم ارتباطی ذخیره می‌شود تنها هنگامی که Applicationها یا برنامه های فرستنده و گیرنده در حال اجرا هستند، یعنی در واقع اگر یکی از این Applicationها از حالت اجرا خارج شوند امکان اینکه پیغام مربوطه از سیستم حذف شود وجود دارد. نکته بعدی که در بحث ارتباطات مبتنی بر پیغام وجود دارد بحث ارتباطات همگام و غیرهمگام است. در یک ارتباط آسنکرون یا غیرهمگام، فرستنده‌ای که پیغام را ارسال می کند عملیات

مربوط به خود را پس از ارسال کماکان ادامه می دهد و پیغام مربوطه در بافرهای محلی سمت ارسال کننده ذخیره می شود تا عملیات Delivery و ارسال به سمت گیرنده تکمیل شود. اما در یک ارتباط سنکرون و یا همگام، Sender، Block می شود تا وقتی که پیغام مورد نظر در بافر محلی دریافت کننده قرار گیرد و یا حتی در یک Level سخت تر دریافت کننده پیغام را برداشته، مطالعه کند. در هر حال استفاده از هر کدام از این انواع ارتباطات، سربارها، نکات مربوط به خود را دارد؛ به عنوان مثال می توان گفت، دیتاگرام مربوط به لایه Transfort در مدل OSI مانند UDP از تکنیک Transient و به صورت غیرهمگام یا آسنکرون برای برقراری ارتباط استفاده می‌کند و چه بسا سیستم های دیگری از تکنیک های دیگری استفاده کنند. در این شکل یک سیستم ماندگار با ارتباط از نوع آسنکرون نشان داده شده است.

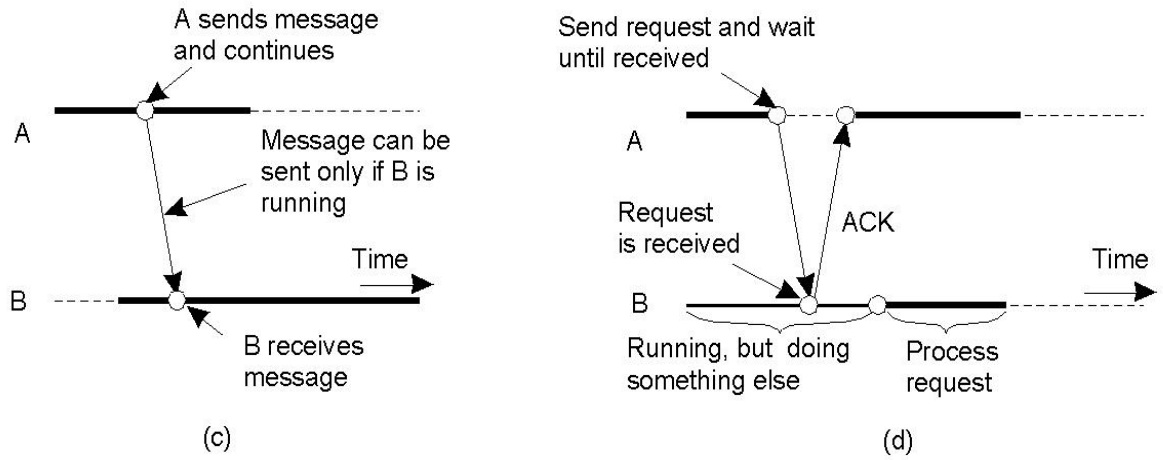


با توجه به شکل، از آنجایی که این سیستم ماندگار است در واقع نیازی به اینکه هر دو فرآیند دریافت کننده و ارسال کننده بطور همزمان در حال اجرا باشند نیست، بنابراین پیغام ارسال شده از سمت فرآیند ارسال کننده هنگامی ارسال می شود که فرآیند دریافت کننده در حال اجرا نیست. از آنجایی که ارتباط یک ارتباط آسنکرون است پس از ارسال پیغام، فرآیند ارسال کننده به دنبال کار خود می رود و هیچ انتظاری وجود نخواهد داشت. در شکل بعد یک سیستم پایا یا ماندگار سنکرون را می بینید، در واقع در اینجا نیز نکته مهم این است که از آنجایی که سیستم Persistent است فرآیند B نیازی به اجرا شدن ندارد. یعنی اینکه در هنگام ارسال پیغام لزوماً نباید فرآیند B یک فرآیند اجرا شده باشد یا در هنگام

دریافت پیغام توسط دریافت کننده لزوماً نباید فرآیند A یک فرآیند در حال اجرا باشد. در هر حال آنچه که در اینجا نسبت به مفهوم قبلی تغییر کرد بحث سنکرون بودن است. بنابراین برای اینکه بحث سنکرون بودن رعایت شود می بینید که بعد از اینکه فرآیند A پیغام را ارسال می کند اندکی منتظر می‌ماند تا تأیید دریافت پیغام به او برسد. بدیهی است که ارسال این تأیید و Acceptance بر عهده سیستم ارتباطی است و در واقع سیستم ارتباطی این پیغام را در داخل بافر مربوط به B تعبیه می‌کند تا زمانی که B شروع به اجرا کرد بتواند پیغام را دریافت کند. بنابراین در شکل نشان داده شده است که به محض اجرای B پیغام مربوطه را دریافت می کند.

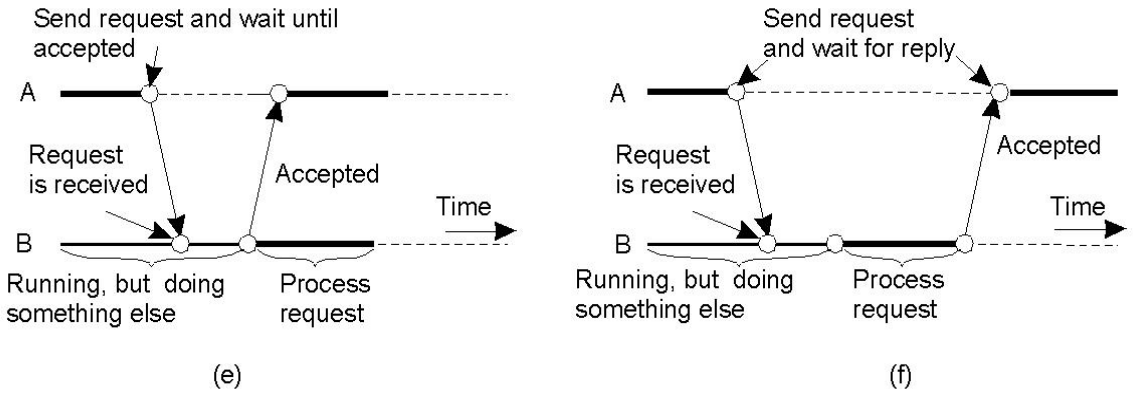
ارتباط مبتنی بر پیغام

در این شکل ارتباط گذرای آسنکرون نشان داده شده است. از آنجایی که ارتباط گذراست، برای ارسال پیغام باید هر دو فرآیند در حال اجرا باشد و از آنجا که آسنکرون از فرآیند Sender یا ارسال کننده پس از اجرا مستقیماً به کار خود ادامه می دهد و در انتظار قرار نخواهد گرفت. حال برای مدلینگ سیستم ارتباطی گذرای غیرهمگام مبتنی بر دریافت، شکل زیر ارائه شده است..



با توجه به شکل، فرآیند A و B به طور همزمان باید در حال اجرا باشند از آن جهت که سیستم Transient یا گذرا است، و پیغامی را ذخیره نخواهد کرد، اگر فرآیندی در حال اجرا نباشد.

فرآیند A نسبت به ارسال پیغام اقدام می کند از آنجایی که سیستم synchronous یا همگام است باید تائید مربوطه را دریافت کند بنابراین تا زمان دریافت تأیید رسیدن پیغام، منتظر خواهد ماند و پس از آن می‌تواند به کار خود ادامه دهد. نکته‌ای که در اینجا ضروری است به آن دقت شود و در شکل نشان داده شده است این است که فرآیند B هنگامی که پیغام را دریافت می کند لزوماً شروع به پردازش آن نمی کند و ممکن است در زمان‌های بعدی پردازش مربوطه را آغاز کند. همانطور که عنوان شد، این شکل مبتنی بر دریافت است، بنابراین به محض اینکه پیغام مربوطه در بافر مربوط به B قرار گرفت پیغام تائید ارسال شد می توان با سخت گیری بیشتری به این مفهوم نگاه کرد و سیستم را در حالتی بررسی کرد که سیستم مبتنی بر انتقال پیغام است، لذا در شکل بعد این مسأله با دقت نشان داده شده است.



با توجه به شکل، سیستمی مبتنی بر Deliver وجود دارد که از نوع گذرای همگام است. تفاوت موجود در این دو شکل آن است که هنگامی پیغام Accept به سمت فرآیند ارسال کننده ارسال می شود که پیغام مربوطه علاوه بر اینکه در بافر مربوط به فرآیند B قرار گرفت توسط فرآیند B، واکشی شده و پردازش آن آغاز شود و یا ارسال پیغام تائید را منوط به این کرد که پیغام مورد نظر کاملاً پردازش شده باشد. در این حالت سیستم ارتباطی گذرای همگام مبتنی بر پاسخ می نامند. در واقع در اینجا اتفاقی که افتاده است این است که زمان انتظار ارسال کننده بیشتر شده است چرا که در واقع ارسال کننده زمانی پیغام Accept را در نظر می گیرد که پردازش پیغام ارسالی به B توسط B به پایان رسیده باشد. در این مباحث انواع مختلف سیستم های ارتباطی مورد بررسی قرار گرفت، حال ابزارهای لازم برای برقراری یک سیستم ارتباطی مبتنی بر پیغام بررسی خواهند شد.