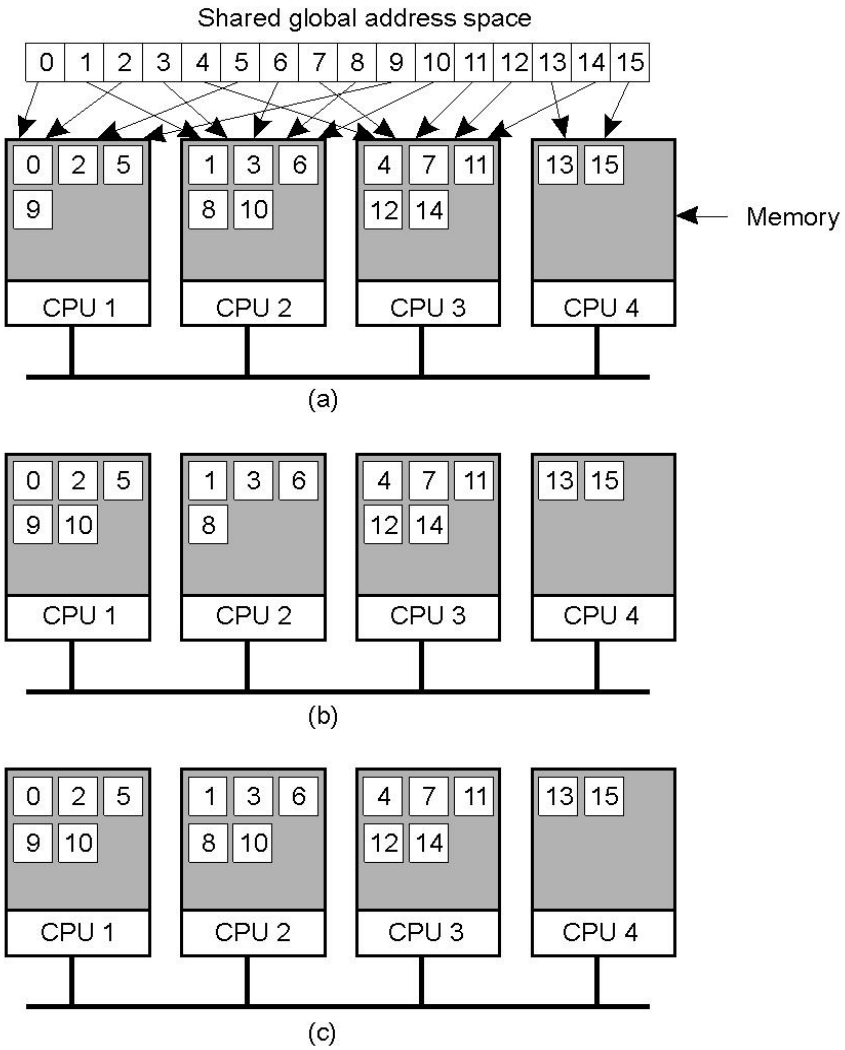
حافظه های توزیع شده ی اشتراکی (Distribution Share Memory)

مفهوم دیگری که در سیستم های DOS مورد توجه قرار می گیرد Distributed Share Memory یا حافظه‌های توزیع شده ی اشتراکی است که آنها را با نام DSM می شناسیم.

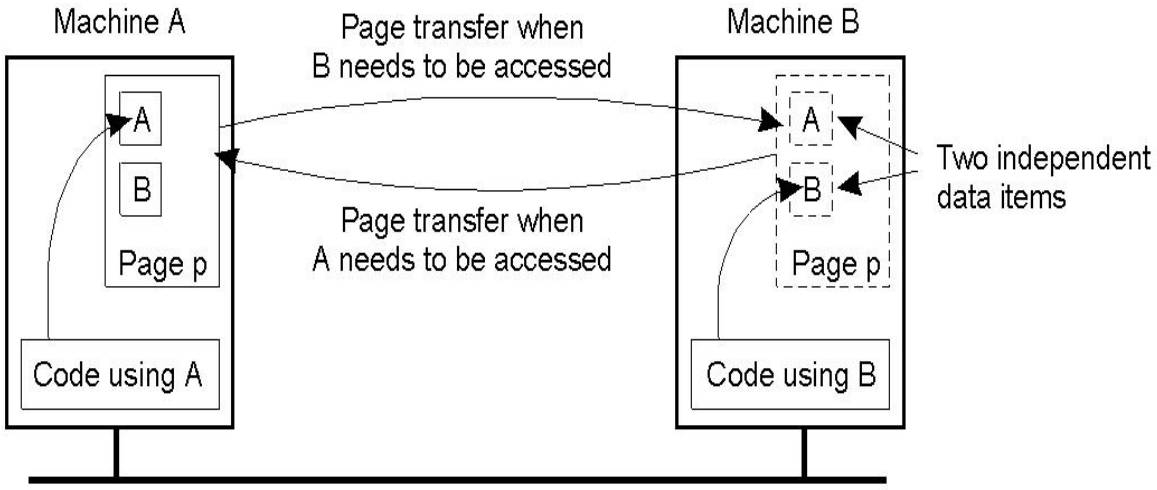
برای مدیریت این نوع حافظه ها یعنی صفحاتی که بر روی ماشین های مختلف توزیع شده اند نکاتی را باید مورد دقت قرار داد که در ادامه به آن می پردازیم. آنچه که حائز اهمیت است این است که در اینگونه سیستم ها تعدادی از صفحات معمولاً به عنوان صفحات محلی مورد توجه قرار می گیرند، اما امکان دسترسی به صفحات محلی یک پردازنده برای پردازنده های دیگر وجود دارد. در این راستا وقفه های (Trap) نرم افزاری تعریف می شوند که به کمک آنها این امکان نیاز، بیان می شود و در مقابل این امکان نیاز و صدور چنین وقفه‌ای معمولاً عملیات Page moving یعنی انتقال یک صفحه از فضای حافظه‌ی مجازی یک گره به گره‌ی دیگر انجام می شود. تکنیک دیگری که مورد استفاده قرار می‌گیرد بحث Page replication یا تکرار یک صفحه است که در شرایطی به جای انتقال صفحه، کپی خواهد شد که در واقع مشابه تمامی مباحثreplication بحث Coherency و Consistency باید مورد توجه قرار گیرد.

جهت روشن تر شدن مفاهیم این بخش به این مثال توجه کنید:



در این مثال فرض شده است که یک فضای آدرس عمومی که شامل ١٦ صفحه است وجود دارد و ٤ پردازنده در سیستم وجود دارند که همگی می خواهند به صورت توزیع شده عمل کنند و احیاناً قسمتی از این فضای حافظه ی عمومی را مورد دسترسی قرار دهند. در سیستم distributed share memory از نوع paging همانطور که عنوان شد، هر یک از این پردازنده ها تعدادی از صفحات را به عنوان صفحات محلی خود در نظر می گیرند. بنابراین فضای ١٦ صفحه ای اشتراکی بین این ٤ پردازنده تقسیم می شود و این تقسیم به این معنی نیست که پردازنده ها تعداد صفحات مساوی دریافت می کنند. به عنوان مثال با توجه به آنچه در شکل مشاهده

می کنید، پردازنده ی شماره ی ۱ چهار صفحه، پردازنده شماره ی ۲ پنج صفحه، پردازنده ی شماره ۳ پنج صفحه و پردازنده ی شماره ٤ دو صفحه در اختیار دارد که آنها را به عنوان صفحات محلی خود می شناسد. حال وضعیتی را تصور کنید که پردازنده ی شماره ۱ نیازمند صفحه شماره ۱۰ می‌باشد که این صفحه ی شماره ۱۰ به عنوان یک صفحه محلی در پردازنده ی شماره ۲ قرار گرفته است. حال عملیات Page moving و یا Page replication ممکن است انجام شود. اگر Page moving اتفاق افتد با توجه به آنچه در شکل (b) می‌بینید صفحه ی شماره ۱۰ از فضای پردازنده ی شماره ۲ منتقل شده است و به پردازنده ی شماره ۱ وارد می شود. اما اگر از تکنیک replication استفاده کنیم، در واقع یک صفحه محلی با محتوای آنچه که در صفحه ی شماره ۱۰ وجود دارد در صفحه شماره ۱ قرار می‌گیرد. در واقع شکل(c) این واقعیت را نشان می‌دهد و نحوه ی ایجاد صفحه ی جدید به صورت تکرار شده را می توانید در این شکل مشاهده نمایید. اما آنچه که در DSM مهم است، برای اینکه بتوان تکنیک های مطرح شده را به خوبی پیاده سازی کرد و دچار مشکلات جانبی آنها نشد بحث اندازه ی صفحه است. بدیهی است که هر چه اندازه ی صفحه بزرگتر باشد، تعداد صفحات کمتری جابجا می شود. اما مفهومی به نام False Sharing یا اشتراک اشتباه ، ممکن است رخ دهد .برای آشنایی با مفهوم False Sharing به سناریویی که در این شکل نشان داده شده است دقت فرمائید.



فرض کرده ایم که ماشین شماره ی A صفحات A و B را به عنوان صفحات محلی در اختیار دارد و از این دو صفحه استفاده می کند. حال ماشین شماره B فضای صفحات محلی آن فعلاً خالی است یا شامل صفحاتی است که مورد بحث ما نیست، در واقع نیازمند دسترسی به صفحات A و B است. صفحات A و B که از نظر ما دو صفحه هستند و ممکن است شامل داده های مستقل نیز باشد با تکنیک Page moving به فضای آدرس B منتقل می شوند. حالتی را در نظر می‌گیریم که هر دوی این صفحات مجدداً مورد نیاز ماشین شماره ی A باشند، در اینجا مجدداً انتقال دیگری نیز انجام خواهد شد و هر دوی این صفحات به فضای حافظه ی A منتقل می شوند.

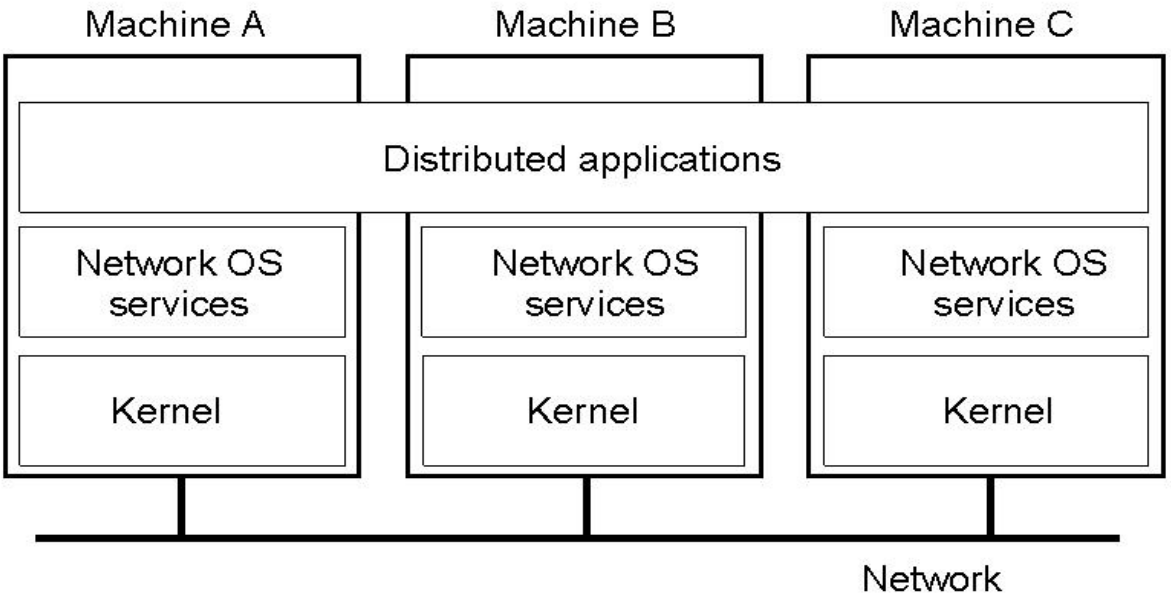
حال اگر بخواهیم سناریو را به گونه ی دیگری تغییر دهیم یعنی با توجه به این سناریو هر دو صفحه مورد نیاز بودند، اما اگر فرض کنیم که ماشین A تنها صفحه A را نیاز دارد و صفحه ی B را نیاز ندارد آنگاه در روند انتقال پس از اینکه صفحه ی A به ماشین A منتقل می شد و صفحه ی B به ماشین B منتقل می شد نیاز به جابجایی هر دو این صفحات وجود نداشت. به عبارت دیگر، اگر اندازه ی صفحه به گونه ای بزرگ بود که مانند آنچه در شکل به عنوان Page P نشان داده شده است هر دو صفحه ی A و B را در اختیار می گرفت یعنی صفحه ای داشتیم که طول آن دو برابر صفحات A و B بود و این دو صفحه در صفحه ی P قرار می‌گرفت آنگاه دسترسی ماشین A و B به محتوای صفحه ی P سناریوی حالت اول را ایجاد می کرد. یعنی همواره محتویات A و B بین این دو ماشین در حال حرکت بودند و عملاً بار انتقال بسیاری را به سیستم منتقل می کردند.

اما در سناریوی دوم ، یعنی حالتی را فرض کنیم که صفحه ی P وجود ندارد و اندازه ی صفحات به گونه ای است که قسمت A و قسمت B هر یک، صفحه مستقل را نشان می دهند و دسترسی به صفحه ی A در ماشین A و دسترسی به صفحه ی B در ماشین B اتفاق می افتد، آنگاه انتقال این دو صفحه پس از یکبار انتقال، باعث می شد که نیازمند انجام انتقال های بعدی نباشیم و این مسأله، یعنی انتخاب اندازه ی صفحه ی کوچکتر باعث شده است که تعداد جابجایی‌های کمتری داشته باشیم؛ بنابراین مبحث False Sharing به این معنا

است که اگر اندازه‌ی صفحه، بزرگ انتخاب شود به طوری که دسترسی به اجزاء یک صفحه توسط ماشین های مختلف انجام گیرد و ماشین‌های مختلف به قسمت های مختلف یک صفحه ی خاص نیاز داشته باشند، (مانند اتفاقی که برای صفحه‌ی P که خود شامل دو صفحه ی A و B بود) آنگاه نرخ انتقال بسیار زیادی در شبکه برای مفهوم مربوط به Page moving رخ خواهد داد.

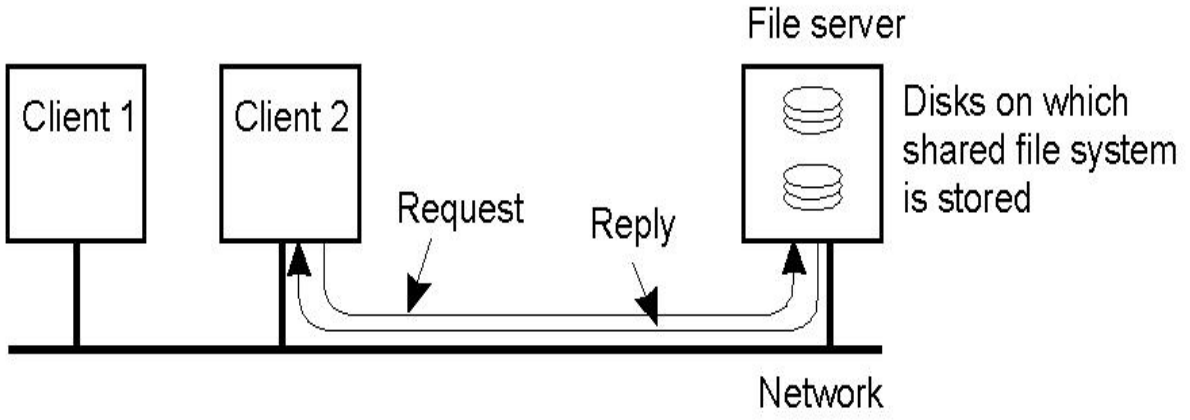
ساختار کلی سیستم های عامل شبکه‌ای

با توجه به مطالب پیشین، معماری عمومی یک Network operating system مشابه آن چیزی است که در شکل نشان داده شده است.



سه ماشین A,B,C به عنوان نمونه حضور دارند، این ماشین ها می توانند مشابه هم باشند که حالتهای همگن را ایجاد کنند و یا غیر مشابه هم باشند که حالت غیرهمگن را ایجاد کنند و در واقع بر روی هر یک از اینها یک سیستم عامل شبکه وجود دارد که سرویس هایی را برای دیگران ارائه می دهد و Application های توزیع

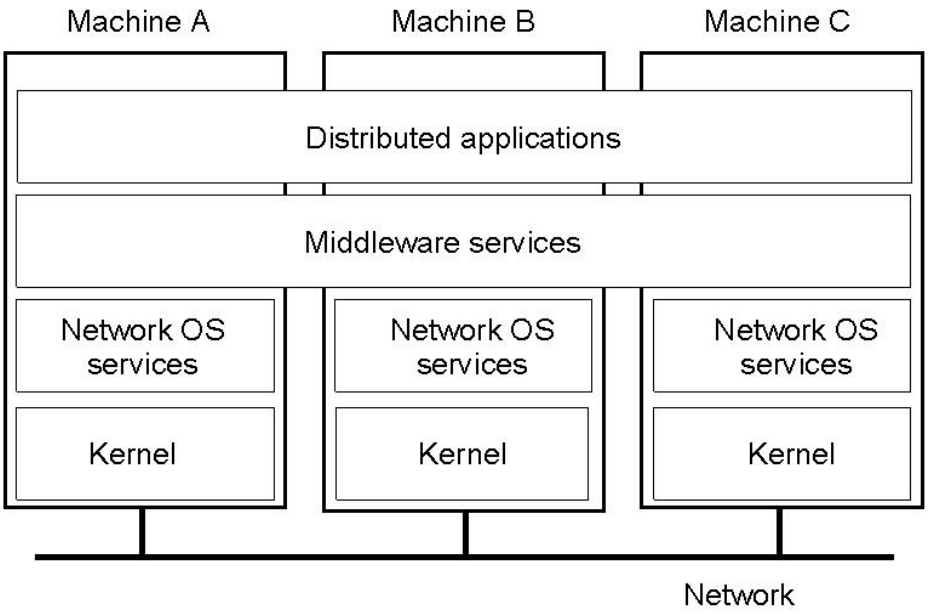
شده به عنوان یک لایه کلی بر روی همه ی ماشین ها قرار می گیرد. با توجه به آنچه در موارد پیشین عنوان شد، در یک سیستم شبکه ای ممکن است ماشین ها و سیستم های عامل آنها متفاوت باشد. آنچه که در مورد یک Network operating system در مقابل یک distributed operating system می توان گفت در واقع کاهش سطح شفافیت سیستم است که معمولاً استفاده از آنها را مشکل تر می سازد. در این گونه سیستم ها معمولاً سرویس هایی که در سطح شبکه توسط هر ماشین ارائه می شود در قالب سرورها دسته بندی می شوند. به عنوان نمونه با توجه به شکل، یک File server وجود دارد که کلاینت ها می توانند فایل سرور را در قسمت های مختلف خود Mount کنند، به عبارت دیگر کلاینت‌ها می توانند فایل سرور را به عنوان جزئی از خود حتی نصب کرده و دسترسی به آن انجام دهند. در این راستا برای انجام دسترسی به اینگونه سیستمی از تکنیک Request-Reply استفاده می شود، یعنی Clientها با توجه به آنچه که در شکل نشان داده شده است، درخواست‌هایی را برای سرورها می فرستند و پاسخ آن را دریافت می کنند.



دقت کنید File server خود یک ند از شبکه است که سرویس هایی را جهت استفاده دیگران از این سرویس‌ها آماده کرده است. این بحث را می توان برای هر یک از ندهای موجود در شبکه مطرح نمود.

میان افزار یا Middleware

مفهوم دیگری که در سیستم های توزیع شده به آن اشاره شد و در این بخش توضیحات بیشتری برای آن ارائه خواهد شد، مفهوم میان افزار یا Middle ware است.



در این شبکه ساختار عمومی یک سیستم توزیع شده به عنوان یک Middle ware را مشاهده می‌کنید با توجه به شکل، Middle ware یک مجموعه از سرویس‌های سطح بالاست که در داخل لایه‌ی زیرین سیستم‌های توزیع شده قرار دارد و سرویس‌هایی را که سرویس‌های سطح بالایی هستند، معمولاً در اختیار کاربران یا همان Application های توزیع شده قرار می‌دهند. اینکه Middle ware چه سرویس هایی را ارائه کند؟ و این سطح Abstract بالا را چگونه ایجاد کنند؟ در واقع از خواصی است که در مورد هر Middle ware ممکن است متفاوت باشد. در واقع می توان گفت هر Middle ware دارای یک بخش پایانی (Back end) و یک بخش جلویی (Front end) است. بخش جلویی در اختیار کاربران است یعنی آن را مشاهده می کنند و بخش انتهایی بخشی است که برای برقراری ارتباط و پیاده سازی سرویس ها استفاده می شود؛ بدیهی است که

همانطور که در شکل نشان داده شده است، مفهوم Middle ware می تواند سرویس‌هایی را ارائه کند که در ماشین‌های مختلف وجود دارد.

Middle ware به نوع خود می تواند به عنوان یک سیستم عامل توزیع شده نیز از دیدگاهی تلقی شود بنابراین به برخی نکات Middle ware با جزئیات بیشتر اشاره خواهد شد. اولین مفهوم از Middle ware بحث مدل های موجود در Middle ware است . یکی از مدل های بسیار معروف Middle ware مدلی است که در سیستم عامل Unix استفاده می شود. در این سیستم عامل هر مورد یک File است. یعنی پرینتر، اسکنر، مانیتور و هر Device سخت افزاری دیگر را شما به عنوان یک فایل در این Middle ware می‌شناسید. بنابراین در واقع استفاده از Middle ware در اینجا باعث خواهد شد چنانچه هر کس آشنا به عملیات مربوط به فایل باشد بتواند به سادگی از تمام اجزاء سخت افزاری موجود در سیستم با انجام تنها همان عملیات استفاده نماید.

مدل دیگری که در مورد Middle ware مطرح است مدلی است که در واقع بر اساس سیستم های فایل توزیع شده تعریف می شود. در این مدل که در واقع مدلی است که قابلیت گسترش خوب و قابل توجهی دارد در واقع تنها شفافیت ایجاد شده برای مدل در قالب سیستم های فایل قدیمی یا فایل های داده ای است. هدف این Middle ware ارائه تکنیکی است که فایل سیستم های سنتی یا به طور خاص فایل های داده ای در سطحی از شفافیت مناسب قرار گیرند.

از دیگر مدل های مبتنی بر Middle ware ، مدل های مبتنی بر Remote procedure call (RPC) هستند. هدف در مدل‌های مربوط به RPC پنهان کردن ارتباطات بین شبکه‌ای است و در این مدل ها می‌توانیم procedure هایی که در سطح ماشین‌های دیگر هستند را اجرا کنیم و Middle ware در واقع سعی خواهد کرد مباحث مربوط به نحوه ی فراخوانی و استفاده از زیر برنامه ای که در ماشین دیگری است را پنهان کرده و سطح ارتباطی بالاتری را ارائه دهد.

از دیگر مدل‌ها می‌توان به Objectهای توزیع شده اشاره کرد که در اینجا نیز هدف از Middle ware پنهان کردن سرویس ها و نحوه ارائه سرویس‌ها است که در سطح بالاتری توسط Middle ware ارائه خواهند شد. سرویس ها در اینجا به عنوان متدهای اشیاء موجود در سیستم پیاده سازی خواهند شد.

مستندات توزیع شده مانند سیستم های مبتنی بر web و مدیریت آنها و مدیریت دسترسی به این مستندات از مدل های دیگری است که در Middle ware ممکن است مورد توجه قرار گیرد.

Middle ware سرویس هایی را ارائه می دهد که در این بخش دسته بندی مختصری از سرویس هایی که در یک Middle ware مورد توجه قرار می گیرد ارائه خواهد شد. از جمله سرویس هایی که در یک Middle ware متداول است امکان ارائه سطح ارتباطی بالاتر یا شفافیت دسترسی است. در واقع سرویس‌های Middle ware سعی می‌کنند این امکان را فراهم کنند تا مفاهیم مربوط به ارتباطات و نکاتی که در ارتباط بین یک ماشین و ماشین دیگر مطرح است توسط Middle ware، پنهان شود.

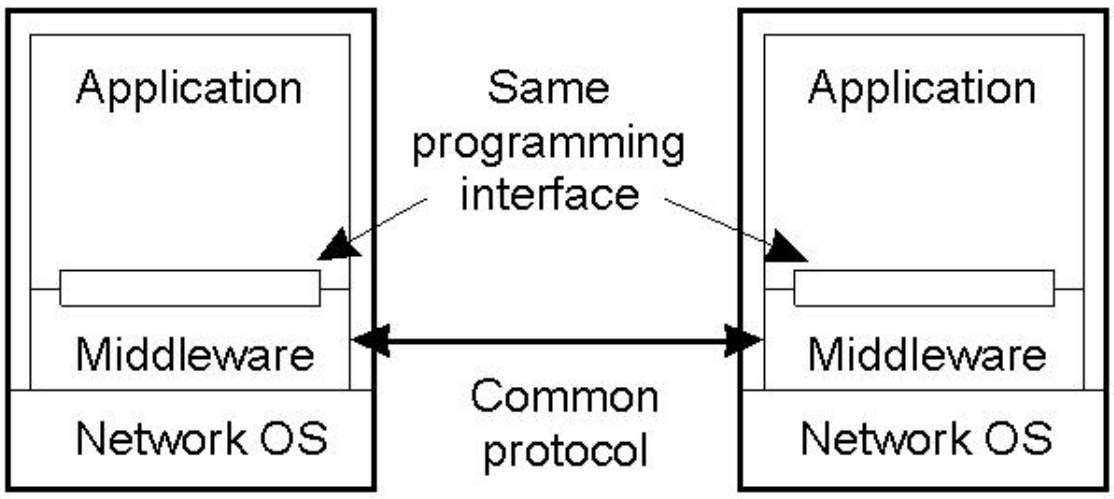
سرویس دیگر، سرویس نامگذاری است در واقع با استفاده از Middle ware می توان هر ماشینی را با نام یکتایی شناخت، حتی اگر نام این ماشین در سطح شبکه مفهوم دیگری داشته باشد.

ماندگاری (Persistence) در واقع امکانی برای storage است و از مفاهیم دیگری است که به عنوان یک سرویس در سطح Middle ware ممکن است ارائه شود. هدف Middle ware این است که نحوه ی دسترسی و استفاده از storage را پنهان کند؛ از این گونه سرویس‌ها در سیستم های فایل توزیع شده یا پایگاه‌های داده مجتمع استفاده می شود.

تراکنش های توزیع شده نیز مبحث دیگری است که ممکن است به عنوان یک سرویس در سطح Middle ware تعریف شود.

سرویس دیگری که معمولاً Middle ware آن را پشتیبانی می کند و یا Middle ware خاصی می تواند آن را پشتیبانی کند، بحث امنیت است.

نکته‌ای که باید در ارتباط با Middle ware به آن دقت کرد مفهوم آن در مقابل Openness است. با توجه به آنچه قبلا عنوان شد، زمانیکه Openness در سیستم مطرح می شود هدف این خواهد بود که سیستم با ارائه سرویس‌هایی این امکان را فراهم کند که در آینده نیز نفرات جدید به سیستم و یا حتیHard ware های جدید به سیستم اضافه شود. در اینجا نیز این مفهوم را به این شکل بیان خواهیم کرد که در یک Middleware open مبتنی بر سیستم های توزیع شده، پروتکل‌هایی که با هر Middle ware بکار می رود بدون توجه به اینکه این Middle ware در چه سطحی از Abstract قرار دارد باید مشابه هم باشد. از جمله این موارد می توان به Interfaceهایی اشاره کرد که توسط Middle ware ها به Application ها ارائه می‌شود. بنابراین انتظار می‌رود که Middle ware های مختلف برای ارائه سرویس هایشان از استانداردهای خاص سرویس استفاده کنند. به عنوان مثال با توجه به آنچه در شکل قرار دارد، فرض می کنیم که در هر یک از این دو ماشین دو Middle ware مختلف وجود دارد، اما سرویس Programming (سرویس برنامه‌نویسی) که هر یک از این دو Middle ware ارائه داده اند، گرچه ممکن است در واقع نحوه ی پیاده سازی سرویس‌های آنها متفاوت باشد، یکسان است. بنابراین از Interface یکسانی استفاده می کنند و لذا Applicationها می توانند بر روی هر دوی آنها اجرا شوند.



با توجه به مباحث مطرح شده در مورد Dos، Network operating system و Middle ware می‌توان در جدول زیر این مفاهیم را بررسی نمود:

با توجه به مطالب پیشین، Dos ها در هر دو حالت Multiprocessor و Multicomputer قابل استفاده هستند. در حالت Multicomputer ، ماشین ها همگی همگن هستند. بنابراین به طور خلاصه می توان گفت، درجه شفافیت در یک Dos Multiprocess معمولاً بسیار بالا است. Dos multicomputer درجه‌ی شفافیت بالایی دارد، اما این درجه در مورد Network operating system ها پائین است و Middle ware ها معمولاً در واقع درجه شفافیت بالایی را پیشنهاد می کنند.

سیستم عامل بکار برده شده بر روی تمام ندها در Dos Multiprocessor در همه جا یکسان است در Dos Multicomputer نیز چون سیستم ها همگن هستند در همه جا یکسان است. اما در Network operating system ها و Middle ware ها این سیستم عامل ها ممکن است متفاوت باشد.

تعداد نسخه‌های متفاوت از سیستم های عامل در Dos Multiprocessor یکی است، یعنی فقط یک سیستم عامل بر روی تمام ماشین ها وجود دارد اما در Dos multicomputer، N سیستم عامل یعنی به تعداد ندها وجود دارد که این سیستم های عامل از نظر ساختاری مشابه هم هستند. در Network operating System ها نیز تعداد سیستم عامل ها N خواهد بود. در Middle ware base ها نیز N است.

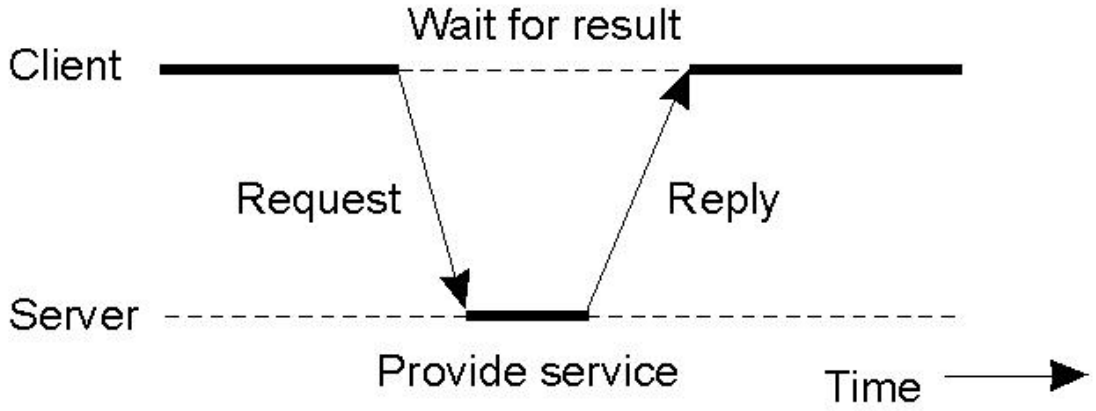
در ارتباط با سیستم های Middle ware اینکه چه ابزاری برای ارتباط استفاده شود، وابسته به مدل‌های مختلف Middle ware است.

مدیریت منابع معمولاً در Dos های Multiprocessor و Multicomputer به ترتیب در حالت عمومی قابل انجام است. در عین حال برای Dos Multiprocessor حالت مرکزی نیز قابل توجه است. اما در Dos Multicomputer حتماً منابع به صورت توزیع شده هستند. در Network operating System به ازای هر نوع، مدیریت ممکن است تغییر کند و این قضیه در Middle ware ها نیز برقرار است.

به طور خلاصه قابلیت گسترش(Scalability) در Dos multiprocessor وجود ندارد، در Dos multicomputer خوب است، در Network operating system قابلیت بالایی دارد. در Middle wareها در واقع با توجه به مدل و نحوه ی پیاده سازی ممکن است متفاوت باشد و در نهایت اینکه Openness در Dos multiprocessor وجود ندارد، در Dos multicomputer وجود ندارد، اما Network operating systemها و Middle war ها معمولاً Open هستند.

مدل Client Server یا مشتری کارگزار

مهمترین مدلی که برای سیستم های توزیع شده عنوان می‌شود، مدل Client Server یا مشتری کارگزار است. در این مدل،سرور، فرایندی است که یک سرویس خاص را پیاده سازی می کند. در این مدل و سایر مدل های پیاده سازی، هدف، دسته بندی فرآیندها و پردازش‌هاست. بنابراین در Client Server یا سرورهایی وجود دارد، که فرآیندی است که یک سرویس خاص را پیاده سازی می کند و Client یا مشتری یک فرآیندی است که یک سرویس خاص را پیاده سازی می کند. معمولاً رفتار موجود در یک Client Server رفتار درخواست و پاسخ (Request reply) است. که نحوه ی انجام آن در شکل نشان داده شده است. Client درخواستی را به سرور اعلام می کند و خود منتظر پاسخ می ماند، سرور با توجه به Load کاری این درخواست را پردازش می کند و پاسخ را ارائه می دهد.

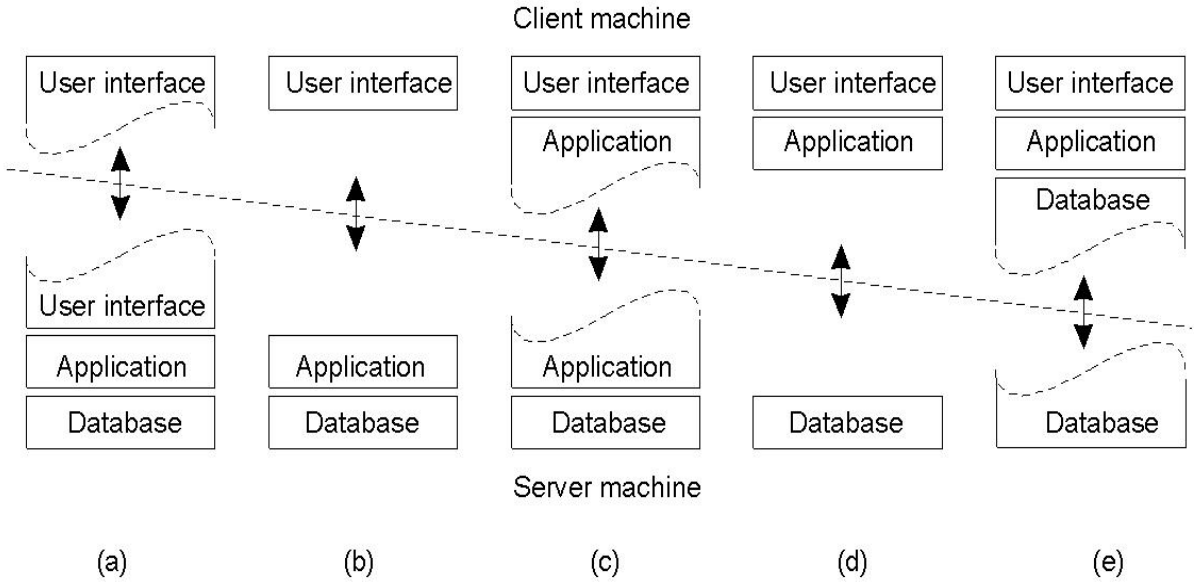


مبحث مهم در یک مدل مشتری کارگزار، بحث ارتباطات یا Communication است. این مطلب به دو حالت Connection oriented و Connectionless وجود دارد. اگر از یک تکنیک بدون ارتباط (Connectionless) برای ارتباط بین Client و Server استفاده کنیم، معمولاً عنوان خواهد شد که کارایی مناسبی به دست می آید. اما این ارتباط، قابل اعتماد نیست. در این مدل معمولاً داده هایی که برای رد و بدل شدن وجود دارند در داخل Packageهایی بر روی شبکه ارسال می شوند. در مدل اتصال گرا (Connection oriented) معمولاً Performance کمی دارد اما قابل اعتماد است، همواره ابتدا ارتباط بین سرور و کارگزار به طور دقیق تعیین می شود و پس از این، مسیر ارتباطی همواره برای فرستادن درخواست ها و گرفتن پاسخ ها استفاده خواهد شد.

در کنار مباحث مربوط به ارتباطات (Communication) ، بحث لایه بندی کاربرد (Application) و مفاهیم مربوط به واسط کاربر و سطح پردازش و سطح داده باید به خوبی طراحی شده و مورد توجه قرار گیرد. با توجه به این نکته‌ی مهم، حال می‌توان معماری‌های مطرح در سیستم مشتری (Client Server) و خدمتگزار را بررسی کنیم.

در ساده ترین معماری، یک ماشین Client و یک ماشین Server وجود دارد و بحث distribution در این مرحله چندان مورد توجه قرار نمی‌گیرد. اما معمولاً معماری‌های چند لایه مورد توجه هستند. در این گونه معماری‌ها معمولاً برنامه‌ها در لایه‌ی Application بر روی ماشین های مختلف توزیع می شوند. دو نوع عمده برای معماری‌های چند لایه در نظر گرفته می شود. اولین نوع، معماری دو سطحی یا Two tired نام گذاری خواهد شد. دومین نوع با عنوان معماری سه سطحی یا Tree tired نام گذاری می‌شود.

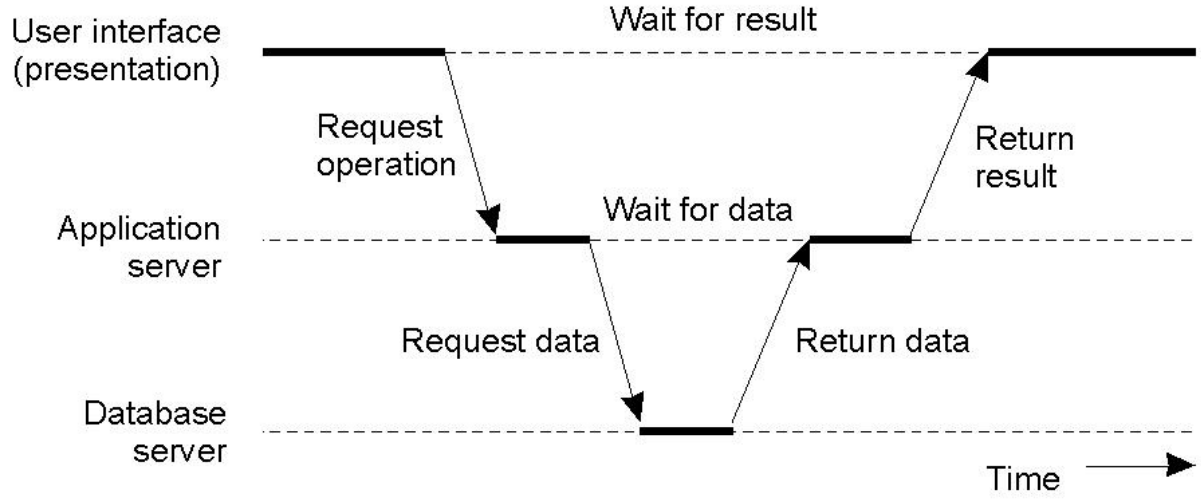
به عنوان نمونه از یک معماری دو سطحی در اینجا و در این شکل گونه های مختلفی بحث شده است.



با توجه به شکل، معماری دو سطحی در واقع دارای دو سطح است، اما آنچه که جای اختلاف دارد این است که این دو سطح در کجا انقطاع بین دو سطح ایجاد می‌کند. به عبارت دیگر انقطاع بین دو سطح یا اتصال بین دو سطح در کدام نقطه برقرار می شود. در شکل A می بینید که ممکن است در سطح بالایی، یک واسط کاربر و در سطح پائین نیز واسط کاربری حضور داشته باشد، به عبارت دیگر سطح اول و دوم و نقطه ی ارتباط سطح اول و دوم در بخش مربوط به واسط کاربر است.

دیدگاه دیگر این است که واسط کاربر به عنوان یک ماژول مستقل در یک سطح مستقل قرار گیرد و پایگاه‌های داده و برنامه‌های کاربردی به عنوان سطح دوم این معماری در نظر گرفته شوند. در شکل سوم می‌بینید که محل انقطاع در واقع بر روی برنامه های کاربردی است و در شکل چهارم بر روی محل اتصال Application و Data base است. در شکل پنجم در واقع این محل انقطاع به یک لایه پائین تر نیز وارد شده و بخشی از Data base در لایه بالاتر و بخشی در لایه پائین تر قرار گرفته است. بدیهی است که هر یک از این معماری‌ها دارای مزایا و معایب خاص خود هستند.

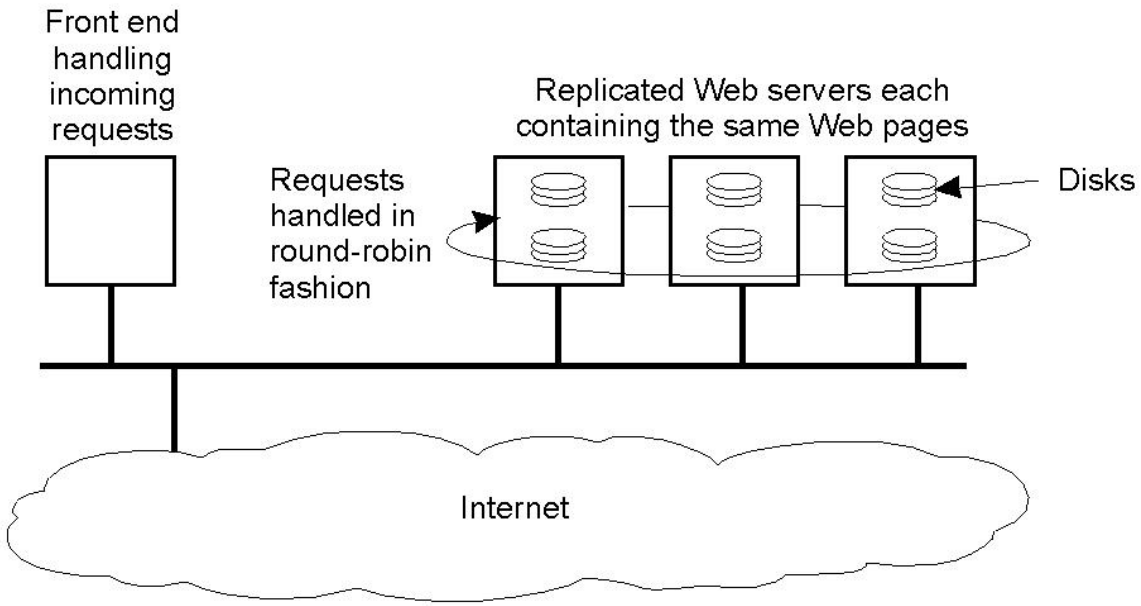
در مبحث مربوط به معماری سه لایه، برنامه کاربردی، واسط کاربر و سرور پایگاه داده یا سرور بخش داده ای، هر یک در یک لایه مستقل هستند. بنابراین درخواست از لایه ی کاربری حتماً به بخش Application و سپس از آنجا مستقل از آنکه چه اتفاقی در درخواست قبلی افتاده است به بخش Data base ارسال می‌شود. این بحث استقلال به این معنی است که ترتیب درخواست‌های ارسالی از User interface به Application server لزوماً مشابه ترتیب درخواست‌های ارسالی از Application به Database server نیست و بالعکس. در واقع ممکن است مدیریت این درخواست‌ها در هر سطحی مستقل انجام شود و سپس داده‌ی محاسبه شده به سطوح بالاتر ارسال شود و در تمام این مراحل درخواست کننده در انتظار قرار خواهد گرفت.



معماری پیشرفته (Modern Architecture)

یک معماری Multitier که تا به حال مورد بررسی قرار گرفت، یک معماری عمودی و از بالا به پائین است اما در معماری‌های نوین سعی می شود یک توزیع افقی باشد. یعنی توزیعی از Client ها و Server ها به صورت مسطح وجود داشته باشد و ممکن است یک Client یا Server خود به صورت فیزیکی به بخش‌های هم عرض منطقی تقسیم شده باشد و هر قسمتی بر روی فضای Share شده‌ی خود عمل می‌کند. به عبارت دیگر هر قسمتی بخشی از داده های Share شده را در اختیار می گیرد.

به عنوان مثال شکل زیر یک فضایی از معماری افقی را نشان می دهد.



آنچه که در این شکل نشان داده شده است نمونه ای از یک سیستم مبتنی بر Web است که اصولاً یک معماری مسطح افقی دارد. در این شکل نشان داده شده است که Web server های تکرار شده متعددی که هر کدام دارای Web pageها ی یکسانی هستند در سیستم قرار دارند و درخواست های بین این Web server ها به صورت Round robin مدیریت می شوند.

در سطح Front end این مجموعه از سیستم‌ها، ماشینی قرار دارد که درخواست ها را پذیرفته و در بین آنها تقسیم می‌کند. در اینجا می بینید که با توجه به تعدد و Replicate شدن داده‌ها، امکان پاسخگویی به درخواست ها با سرعت بیشتری مهیا می‌شود. در واقع این معماری خود سعی کرده است که با قرار دادن یک Front end یا یک ماشین در مقابل ماشین‌های تکرار شده یا Web serverهای تکرار شده، یک سطح دسترسی بالاتر برای کاربران مهیا کند و کاربران از اینکه بیش از یک ماشین پشت این Front end قرار دارد خبر چندانی نخواهند داشت.