فصل اول

سیستم های توزیع شده

۱-۱- مقدمه

امروزه از جمله کاربردهای مهم سیستم های کامپیوتری، انجام عملیات محاسباتی پیچیده بر روی بانک های اطلاعاتی (دیتا) و پردازش های بسیار سریع است. گستردگی و پیچیدگی چنین اعمالی، ارتباط کامپیوترها از طریق شبکه ها را ضروری می سازد. اولین راه حل برای رسیدن به این هدف استفاده از سیستم های متمرکز بود، که در آنها یک کامپیوتر (منفرد) متشکل از یک یا چند Cpu، تمام تقاضاها (Requests) را پردازش می کرد. با در نظر گرفتن عواملی از قبیل هزینه ها قابلیت اطمینان و مجزا و مستقل بودن بخش های مختلف سازمان هایی که از این سیستم ها استفاده می کنند راه حل فوق ناکافی به نظر نمی رسد. بدین دلیل، استفاده از سیستم های توزیع شده، که در آنها از تعداد زیادی کامپیوتر مرتبط به یکدیگر از طریق شبکه های ارتباطی استفاده می شود، مدنظر قرار گرفت. مشخصات استقلال (Independency)، توزیع شدگی (Distribution) و ناهمگونی (Heterogeneity) این سیستم ها، نیاز به سیستم های نرم افزاری توزیع شده را جهت ارائه دید کلی و یکپارچه از آنها ضروری می سازد. یک سیستم توزیع شده، مجموعه ای از کامپیوترهای خودگردان (یا متکی به خود) (Autonomous) است که توسط شبکه به یکدیگر ارتباط داشته و با سیستم های نرم افزاری توزیع شده، تجهیز شده اند. در چنین سیستمی، کامپیوترها می توانند فعالیت های خود را با همدیگر هماهنگ کرده و از منابع عمومی سیستم به طور مشترک بهره ببرند. این سیستم ویژگی شفافیت توزیعی (Distribution Transparency) را برای کاربر فراهم می آورد.

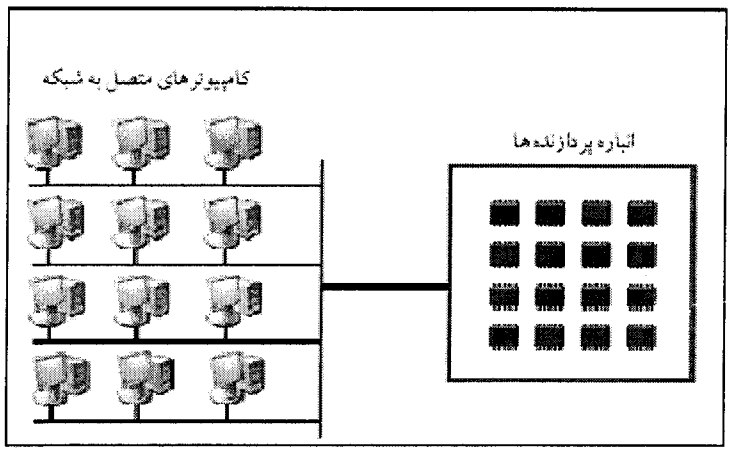
۱-۲- مفهوم سیستم های توزیع شده

یک سیستم توزیع شده مجموعه ای از کامپیوترهای به هم متصل است که در نظر کاربر به صورت یک کامپیوتر واحد جلوه می کنند. دو نکته در این تعریف وجود دارد. نکته اول

جنبه سخت افزاری آن است : به این معنی که تمامی کامپیوترهای موجود در یک سیستم توزیع شده خود مختار و مستقل هستند. نکته دوم که جنبه نرم افزاری دارد این است که کل سیستم توزیع شده در نظر کاربر باید به صورت یک کامپیوتر واحد به نظر بیاید.

جهت روشن تر شدن مفهوم سیستم های توزیع شده مثال هایی را بررسی می کنیم. شبکه ای را در یک مرکز دانشگاهی و یا صنعتی در نظر بگیرید. علاوه بر کامپیوتری که به هر کاربر اختصاص داده می شود ممکن است انباره ای از پردازنده ها نیز در یک ماشین جداگانه موجود باشد که این پردازنده ها مختص به یک کاربر خاص نیستند ولی به طور پویا و بنابر نیاز به کاربران مختلف در سیستم تخصیص می یابند. علاوه بر این، کل سیستم نیز می تواند دارای سیستم فایلی یکپارچه باشد که تمامی فایل های موجود در سیستم به طریق یکسان و از یک مسیر توسط کلیه کاربران قابل دسترسی باشد.

شکل ۱-۱این نوع سیستم را نمایش می دهد.



شکل ۱-۱

در این جا اگر کاربر دستوری را تایپ کند، سیستم می تواند به دنبال محل اجرای مناسب برای این دستور در سیستم بگردد. این محل می تواند داخل ایستگاه کاری خود کاربر یا یک ایستگاه کاری بیکار دیگر یا یکی از پردازنده های موجود در انباره پردازنده ها باشد که به هیچ کاربری تخصیص داده نشده است. اگر این کارها را به گونه ای انجام گیرد که

سیستم در نظر کاربر به صورت یک سیستم تک پردازنده با اشتراک زمانی (Timesharing) جلوه کند، یا به عبارت دقیق تر کل این عملیات به صورتی انجام شود که کاربر متوجه آن نشود، آن گاه این سیستم یک سیستم توزیع شده خواهد بود.

به عنوان مثالی دیگر از سیستم های توزیع شده، یک بانک بزرگ بین المللی با صدها شعبه در سراسر جهان را در نظر بگیرید. هر شعبه دارای یک کامپیوتر مرکزی است که حساب ها و تراکنش های مالی محلی را انجام می دهد. علاوه بر این هر کامپیوتر می تواند با کامپیوترهای شعب دیگر ارتباط برقرار کند. اگر امور بانکی را بتوان بدون توجه به مکان مشتری و حساب بانکی او انجام داد به طوری که مشتری متوجه نشود که امور بانکی او در کامپیوتر شعبه او یا کامپیوتر یک شعبه دیگر (حتی در خارج از کشور) انجام شده، آن گاه این سیستم نیز حالت توزیع شده دارد.

۱-۳- مفاهیم سخت افزاری در سیستم های توزیع شده

سیستم های توزیع شده بر اساس توزیع پردازنده و حافظه در کامپیوترهای موجود در سیستم طبقه بندی می شوند. چنا ن چه در یک سیستم، کل پردانده ها و حافظه در یک ماشین مرکزی باشند و کامپیوترهای کاربران فاقد پردازنده و حافظه اختصاصی باشند در این صورت با یک سیستم متمرکز روبرو هستیم نه سیستم توزیع شده.

انواعی از سیستم ها وجود دارند که در آنها چند پردازنده به یک حافظه مشترک دسترسی داشته و کل پردازنده ها باید تنها از همان حافظه برای پردازش هایشان استفاده کنند. در این نوع سیستم به هر پردازنده بخشی از حافظه کل اختصاص می یابد و پردازنده ها از طریق حافظه مشترک با یکدیگر ارتباط برقرار می کنند، به نحوی که یک پردازنده می تواند داده ای را در محلی از حافظه بنویسد و پردازنده دیگری همان داده را از همان محل از حافظه بخواند. بدین صورت بین این دو پردازنده ارتباط برقرار می شود. به این نوع از سیستم ها، چند پردازنده ای (Multiprocessor) گفته می شود. در بسیاری از موارد

پردازنده ها و حافظه اشتراکی بر روی یک برد اصلی نصب می شوند. این نوع سیستم ها جزو سیستم های « با اتصال نزدیک » (Tightly Coupled) هستند. در این سیستم ها ارتباط بین پردازنده و حافظه به صورت مستقیم و بدون نیاز به عملیات I/O (ورودی/خروجی) برقرار می شود.

در نوع دیگری از سیستم های توزیع شده، هر کامپیوتر دارای پردازنده و حافظه اختصاصی خود است. به این نوع از سیستم ها، چند کامپیوتری (Multicomputer) گفته می شود که از اتصال تعدادی کامپیوتر به هم ساخته می شوند. در واقع تفاوت اصلی نوع اخیر با سیستم های چند پردازنده ای در این است که در این نوع سیستم ها هر کامپیوتر دارای پردازنده و حافظه مختص به خود است ولی علاوه بر این ممکن است از یک انباره مشترک پردازنده ها و بانک های مشترک حافظه در یک ماشین مرکزی نیز سود جوید. علاوه بر این در این نوع سیستم ها برای آن که دو کامپیوتر بتوانند از حافظه یکدیگر بخوانند باید از عملیات I/O استفاده شود، در حالی که در سیستم های چند پردازنده ای برای دسترسی به حافظه مشترک نیازی به عملیات I/O نیست و این کار مستقیماً با دستورات خود پردازنده ها انجام می شود. در ضمن، نحوه آدرس دهی حافظه نیز در این دو نوع سیستم متفاوت است. به عنوان مثال، در یک سیستم چند پردازنده ای چنان چه یک پردازنده در آدرس ۱۰۰۰ عدد ۲۸ را بنویسد آن گاه هر پردازنده دیگری که حافظه آدرس ۱۰۰۰ را بخواند عدد ۲۸ را خواهد خواند، در حالی که در یک سیستم چند کامپیوتری اگر یک پردازنده در آدرس ۱۰۰۰ عدد ۲۸ را بنویسد آن گاه چنان چه پردازنده دیگر آدرس ۱۰۰۰ را بخواند لزوماً عدد ۲۸ را نخواهد خواند، چون پردازنده اولی در آدرس ۱۰۰۰ حافظه متعلق به خود می نویسد و پردازنده دومی نیز از آدرس ۱۰۰۰ حافظه متعلق به خود می خواند و این دو با هم فرق دارند. این نوع سیستم ها جزو سیستم های با « اتصال

سبک » (Lossely Coupled) هستند. در این سیستم ها ارتباط بین پردازنده و حافظه با استفاده از عملیات I/O برقرار می شود.

۱-۴- مفاهیم نرم افزاری در سیستم های توزیع شده

مفاهیم نرم افزاری در سیستم های توزیع شده نسبت به مفاهیم سخت افزاری از اهمیت بیشتری برخوردارند و به مراتب پیچیده ترند و به علت همین پیچیدگی، نمی توان مانند مفاهیم سخت افزاری آنها را در گروه های مشخص قرار داد. با این حال سعی می کنیم نگاهی کلی به آنها داشته باشیم.

سیستم های توزیع شده می توانند از سیستم عامل های شبکه استفاده کنند. در این حالت هر کامپیوتر سرویس گیرنده مجهز به سیستم عامل، پردازنده و حافظه مختص به خود است و تمامی این کامپیوترها می توانند از طریق یک پروتکل مشترک با یکدیگر رابطه برقرار کنند. علاوه بر این، یک فایل سرور نیز می تواند وجود داشته باشد تا درخواست های خواندن و نوشتن فایل را از طرف برنامه های مختلف (که در ماشین های سرویس گیرنده در حال اجرا هستند) اجابت کند. کامپیوترهای سرویس گیرنده می توانند از طریق دستور « ورود از راه دور » (Rlogin) با یکدیگر رابطه برقرار کنند، به این صورت :

Rlogin Machine

یا می توانند از طریق دستور « کپی از راه دور » (RCP : Remote Copy) فایلی را از یک ماشین به ماشین دیگری کپی کنند :

RCP Machine :File Machine۲: File۲

سیستم های توزیع شده واقعی به صورتی هستند که بر روی شبکه ای از کامپیوترهای متصل به هم اجرا می شوند و کل این شبکه ها را در نظر کاربر به صورت یک سیستم تک پردازنده مجازی جلوه می دهند. در واقع هر کامپیوتر دارای سیستم عامل و پردازنده

خودش است. سیستم توزیع شده به صورت یک لایه اضافی بر روی سیستم شبکه ای موجود می نشیند و کاری می کند که کل سیستم فایلی، منابع موجود در شبکه، پردازنده ها و حافظه موجود در سیستم ها به صورت یکپارچه و با اشتراک زمانی در اختیار کاربر قرار گیرد.

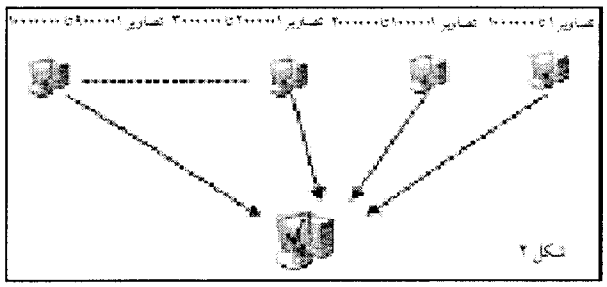
۱-۵- مزایای سیستم های توزیع شده

هدف از ایجاد سیستم های توزیع شده، رفع بعضی نواقص مربوط به سیستم های تک (Single) و متمرکز بود. همان طور که ذکر شد، سیستم توزیع شده دیدگاهی جدید را در مدل محاسباتی مطرح کرد. بدین صورت سیستم های توزیع شده نسبت به سیستم های قبلی دارای مزیت هایی شدند که در ادامه برخی از آنها را مورد بررسی قرار می دهیم.

۱-۵-۱- قدرت پردازش بالا

سیستم های توزیع شده بر اساس اتصال چند کامپیوتر به هم به وجود می آیند. با افزایش تعداد کامپیوترهای متصل به هم (یعنی تعداد پردازنده ها) قدرت پردازش نیز افزایش می یابد. در واقع از آن جا که کامپیوترهای موجود در یک سیستم توزیع شده می توانند به طور موازی با هم کار کنند، به سطحی از پردازش موازی (Parallel Processing) دست می یابیم و چنان چه طراحی سیستم توزیع شده به لحاظ نرم افزاری و سخت افزاری مناسب باشد، می توان با افزایش کامپیوترها به قدرت پردازش بالاتری دست یافت. به عنوان مثال، یک برنامه پردازشگر تصویر را در نظر بگیرید. فرض کنید می خواهیم ۱۰ میلیون تصویر را مورد پردازش قرار دهیم. با استفاده از یک سیستم توزیع شده متشکل از یازده کامپیوتر می توان این کار را انجام داد، به طوری که هر کامپیوتر، یک میلیون تصویر را پردازش می کند و نتیجه کار را به یک کامپیوتر مرکزی برمی گرداند تا در کامپیوتر مرکزی این نتایج با هم ترکیب شوند. به این کامپیوتر مرکزی اصطلاحاً هماهنگ کننده (

Coordinator) گفته می شود. با این حساب، اگر کاربر تنها با کامپیوتر هماهنگ کننده کار کند متوجه وجود ۱۰ کامپیوتر دیگر نخواهد شد و صرفاً « احساس » می کند درخواستش با سرعتی بسیار بالاتر انجام شده است. و این، دقیقاً با تعریف ارائه شده برای سیستم های توزیع شده تطابق دارد. شکل ۱-۲ این سیستم را به طور دقیق تر نمایش می دهد.



شکل ۱-۲

۱-۵-۲- حجم عظیم حافظه

حداکثر حجمی از حافظه اصلی و جانبی که یک کامپیوتر می تواند حمایت کند وابسته و محدود به سخت افزار و سیستم عامل آن است و از یک حد مشخص نمی تواند فراتر برود. با این حال کاربردهای صنعتی، مهندسی و علمی بسیاری وجود دارند که نیاز به ظرفیت ذخیره سازی بسیار زیادی دارند. به عنوان مثال سایت هایی مانند Yahoo و Hotmail که حجم بسیار بالایی (صدها میلیون بایت اطلاعات مختلف از قبیل صفحات خود سایت، نامه های الکترونیکی کاربران و یا مشخصات افراد) را در پایگاه اطلاعاتی خود نگاه می دارند چاره ای جز استفاده از تعداد زیادی کامپیوتر سرویس دهنده به صورت توزیع شده ندارند.

۱-۵-۳- تحمل خطا (Fault Tolerance)

در یک سیستم توزیع شده چنان چه بعضی از کامپیوترها خراب شوند و از سیستم خارج شوند کارکرد کلی سیستم مختل نخواهد شد زیرا کامپیوترهای دیگر می توانند وظیفه کامپیوترهای خارج شده از سیستم را بر عهده گیرند و بدین صورت بار آن کامپیوتر بین بقیه کامپیوترها توزیع می شود. تنها ضعفی که ممکن اسن احساس شود این است که کارایی کلی سیستم کمی کاهش کی یابد، ولی در عوض، سیستم می تواند به کار خود ادامه دهد. برای رسیدن به چنین وضعیتی، یک سیستم مدیریت خطای توزیع شده باید این امور را مدیریت کند.

۱-۶- اجزای سیستم های توزیع شده

هر سیستم توزیع شده صرفنظر از نوع و کاربردش دارای اجزایی است که اعمال مختلفی را به انجام می رسانند. این اجزا در داخل یک سیستم توزیع شده می توانند با هم در تعامل باشند.

۱-۷- مدیریت پروسه های توزیع شده

مدیریت پروسه های توزیع شده باید ایجاد، حذف، شروع و اتمام پروسه ها را طوری انجام دهد که نحوه اجرای آن از یک ماشین به ماشین دیگر (در یک سیستم توزیع شده) تفاوت نکند. در سیستم عامل شیکه وضع بدین صورت بود که مادامی که یک ماشین در یک سیستم سرویس دهنده / سرویس گیرنده از پروتکل های استاندارد شبکه تبعیت کند می تواند هر کاری انجام دهد، در حالی که در سیستم های توزیع شده این کافی نیست، بلکه باید برای تمام ماشین های موجود در شبکه یک سری تابع مخصوص موسوم به System Call وجود داشته باشند تا بتوان به طور هماهنگ روی پروسه ها اعمال کرد (از قبیل توابع ایجاد، حذف، شروع و اتمام) .

علاوه بر این باید یک مکانیزم عمومی و یکتا در سراسر سیستم برای ارتباط بین پروسه ها (Interprocess Communication) موجود باشد تا هر پروسه بتواند به راحتی با پروسه های دیگر ارتباط برقرار کند. مکانیزم این کار باید در ماشین های مختلف موجود در سیستم یکسان باشد و علاوه بر این ارتباط محلی پروسه ها و ارتباط پروسه های دور نیز باید با مکانیزم یکسانی انجام شود. یک شمای حفاظت عمومی (General Protection Scheme) نیز باید وجود داشته باشد که در این راستا می توان از لیست های کنترل دسترسی (Access Control List) نیز سود جست.

۱-۸-مدیریت حافظه توزیع شده

مدیریت حافظه توزیع شده از مدیریت حافظه در سیستم های تک، بسیار مشکل تر است چون حافظه توزیعی در سراسر ماشین های موجود در سیستم پخش است. یکی از مسائل مربوط به مدیریت حافظه توزیعی کنتری دسترسی همزمان به حافظه است. اگر حداقل دو ماشین بخواهند به یک خانه از حافظه توزیعی دسترسی یابند و یکی از آنها بخواهد عمل نوشتن در حافظه را انجام دهد، و دیگری عمل خواندن را، آن گاه این مسئله مطرح می شود که حاصل این اعمال بسته به این که کدام یک زودتر انجام شوند متفاوت است. برای رفع این مشکل باید قبل از نوشتن در محلی از حافظه آن را قفل نمود تا بقیه ماشین ها نتوانند عملیات خواندن یا نوشتن را بر روی آن محل از حافظه انجام دهند. بعد از انجام عمل نوشتن در حافظه، قفل توسط ماشین مربوطه آزاد می شود تا بقیه ماشین ها بتوانند با آن محل از حافظه کار کنند.

مدیریت حافظه توزیعی باید بتواند کلیه عملیات متداول بر روی حافظه که در یک محیط تک قابل اجرا هستند (از قبیل اخذ، آزاد سازی و تغییر اندازه حافظه اخذ شده) را به انجام برساند و این کار را باید به صورتی انجام دهد که کاربر متوجه نشود حافظه

درخواستی او روی چه ماشینی و به چه طریقی اخذ شده است و کل حافظه توزیعی را به صورت یک حافظه واحد ببینند.

۱-۹- مدیریت فایل توزیع شده

مدیریت فایل توزیع شده باید بتواند یک دید یکپارچه از کل فایل های موجود در سیستم بدهد به طوری که تمام فایل های موجود در ماشین ها به یک طریق قابل دسترسی باشند و نحوه مسیردهی فایل ها، وابسته به ماشین نباشد و از ماشینی به ماشین دیگر تفاوت نکند. به عنوان مثال می توان از DFS (Distributed File System) موجود در ویندوز ۲۰۰۰ یاد کرد که تا حدودی خصوصیات یک سیستم فایلی مناسب را دارد. به عنوان مثال در این سیستم، می توان پارتیشنی ساخت که روی چند دیسک سخت پراکنده باشد.

۱-۱۰-برنامه های کاربردی در سیستم های توزیع شده

سیستم های توزیع شده دارای برنامه های جدی فراوانی هستند و برنامه های این گونه سیستم ها در حال افزایش است. در این جا دو مورد از برنامه های کاربردی سیستم های توزیع شده را به اختصار بررسی می کنیم.

۱-۱۰-۱- بانک های اطلاعاتی توزیع شده

بانک های اطلاعاتی در سیستم های متمرکز بر روی یک یا چند سرویس دهنده قرار دارند، ولی ایم مدل در سیستم های توزیع شده بسط یافته و بانک های اطلاعاتی توزیع شده را به وجود آورده است که اشیایی از قبیل جداول و شاخص ها در آن پراکنده هستند و مدیریت آنها نیز بسیار پیچیده تر است. به عنوان مثال می توان جداول بسیار بزرگ را به طریق ستونی یا سطری تجزیه و در محیط توزیع شده پخش نمود. علاوه بر این انجام پرس

و جوها (Queries) و تراکنش ها در یک محیط توزیع شده انجام می شود و نتیجه آن به درخواست کننده ارسال می گردد.

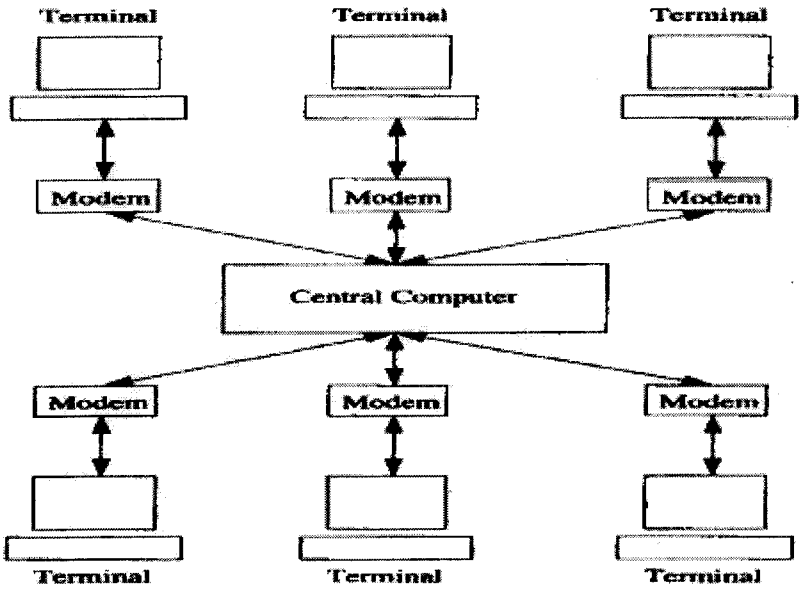
۱-۱۰-۲- سیستم های اطلاعاتی توزیع شده

بسیاری از سیستم های اطلاعاتی که وابسته به مناطق جغرافیایی هستند لزوماً باید به صورت توزیع شده عمل کنند. در واقع توزیع اطلاعات وابسته به محل تولید و نگهداری آنها خواهد بود. به عنوان مثال اطلاعات مشاغل در کلیه استان های کشور از طریق یک سیستم اطلاعاتی توزیع شده قابل دستیابی است. از آن جا که اطلاعات مشاغل هر استان مختص به خود آن استان است. بنابراین در همان استان نگهداری می شود. چنان چه بخواهیم یک گزارش کلی از وضعیت مشاغل در کل کشور داشته باشیم آن گاه سیستم های توزیع شده مربوطه باید به اطلاعات مشاغل استان های مختلف دسترسی پیدا کرده و با پردازش آنها اطلاعات کل استان ها را یک جا به مرکز ارسال کنند. تمامی این کارها طوری انجام می شود که کاربر تصور می کند کل اطلاعات در همان مکان در دسترس او هستند.

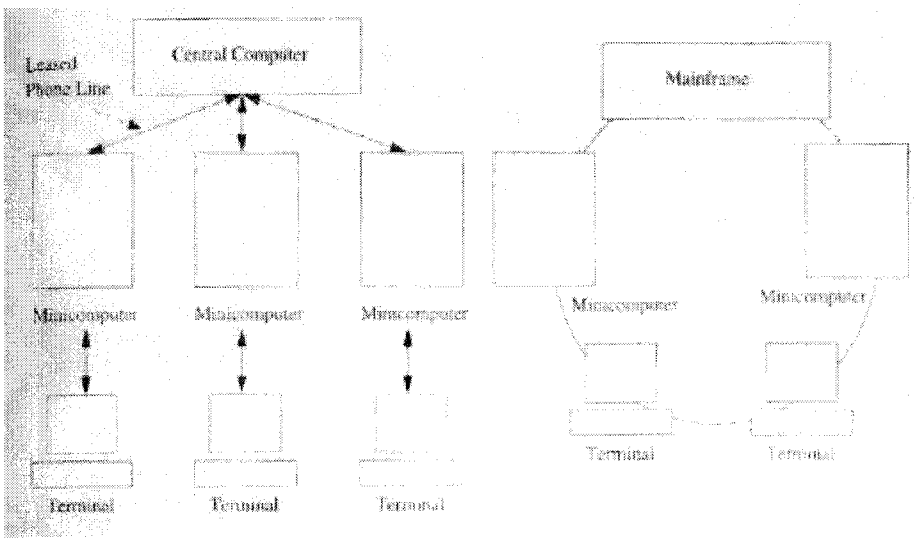
۱-۱۱- معماری های سخت افزاری

تعریف و پیاده سازی سیستم های توزیع شده، از سیستم هایی با ترمینال های راه دور و مینی کامپیوترها شروع شده است. در این سیستم ها، ترمینال ها (مینی کامپیوترها) بخشی از عملیات را مستقلاً انجام داده و ضمناً به طور متناوب با سیستم های مرکزی (مین فریم ها – MainFrames) در ارتباط بودند. در دهه ۱۹۷۰ تا ۱۹۸۰ مفهوم سیستم های توزیع شده بیشتر متناظر با « پردازش توزیع شده » به کار می رفت. بر اساس این مفهوم، یک تقاضای پردازش، به بخش های کوچکتر تقسیم شده و هر یک توسط یک ماشین اجرا می شده است. در آن زمان ساختار معماری سیستم ها به صورت

ستاره، سلسله مراتبی یا حلقه ای طراحی می شدند، شکل کلی ساختارها، در شکل های ۱-۳ و ۱-۴ نشان داده شده اند.



شکل ۱-۳- ساختار ستاره



شکل ۱-۴- ساختار سلسله مراتبی و ساختار حلقه ای

انواع مختلف سیستم های توزیع شده، طبق تعریف مذکور عبارتند از:

۱- سیستم با توزیع عملکردی (عملیاتی) (Functional Distributive) :

در این سیستم ها، بخشی از وظایف سیستم به صورت توزیع شده انجام می گیرد، و لیکن قابلیت پردازش تراکنش کامل وجود ندارد. سیستم، کنترولرها یا ترمینال های هوشمندی برای بعضی کارها، مثل Edit کردن پیام، شکل دهی صفحه، جمع آوری دیتا، ارتباط دوطرفه با اپراتور ترمینال، برخی وظایف امنیتی و فشرده سازی پیام به کار می گیرد.

۲- سیستم کنترل شده مرکزی (متمرکز) (Centrally Controlled) :

این نوع سیستم ها، به صورت مجموعه ای از کامپیئترهای جانبی کوچک، که امکان پردازش کامل را دارند فرض می شوند. هر یک از این کامپیوترها توسط کامپیوترهای سطح بالاتر (از نظر معماری و ساختار) اداره می شوند.

۳- سیستم توزیع شده یکپارچه (Integrated System) :

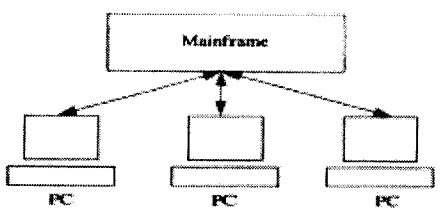
چنین سیستمی مرکب است از سیستم های مجزا که دارای طراحی دیتای یکپارچه بوده و در کامپیوترهای مختلف جای دارند.

۴- سیستم های غیر یکپارچه (Non-Iintegrated System) :

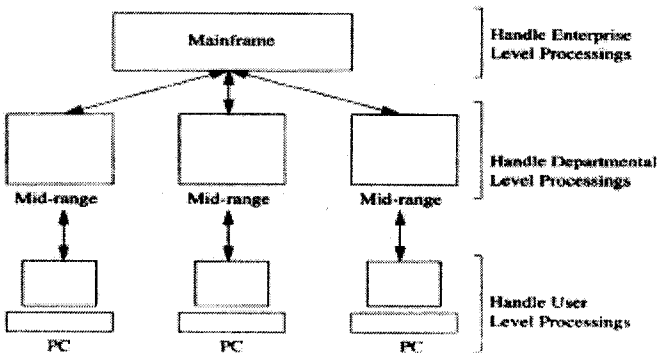
این سیستم ها شامل سیستم های مستقلی هستند که از طریق شبکه کامپیوتری به یکدیگر اتصال دارند.

مفاهیم و مشخصات پیاده سازی سیستم های توزیع شده از اواخر دهه ۸۰-۱۹۷۰ تغییر کرد. این مفاهیم دیدگاه های متفاوتی را در بر گرفته اند، از سویی به کاربردهای مجزا و از نظر جغرافیایی پراکنده که با یکدیگر هماهنگ هستند، اطلاق می شوند، از سویی دیگر به کاربرد منفرد که توسط مؤلفه های برنامه ای مستقل و پراکنده شکل گرفته باشد،نیز سیستم توزیع شده، گفته شده است. این مفاهیم به صورت چهار نوع ساختار قابل پیاده سازی، ارائه شدند. در شکل ۱-۵ ، ساختاری نشان داده شده است، که در آن کامپیوتر

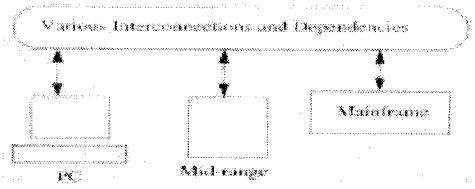
مرکزی اصلی (Main Frame) به کامپیوترهای شخصی، با توان انجام وظایف خاص، وصل بوده و انجام کارهای به خصوصی از کامپیتر مرکزی به کامپیوترهای شخصی محول می شود. نوع دیگر ارتباط ساختاری سیستم، به صورت PC هایی است که به ماشین های سطح میانی وصل بوده و آنها نیز، به صورت سلسله مراتبی، به مین فریم ها متصل هستند (شکل ۱-۶) . در این ساختار کاربران PC ها از ماشین سطح میانی برای دستیابی به کاربرها و دیتای ذخیره شده پشتیبان (Backup) استفاده می کنند. کامپیوترهای اصلی (مین فریم ها) نیز برای پردازش های مفصل تر و سنگین تر بر روی دیتای دریافتی از سطح میانی استفاده می شوند. نوع دیگری از ساختارها (شکل ۱-۷) از سیستم های هم مرتبه (Peer) با درجه های وابستگی و ارتباطی متفاوت تشکیل شده اند و نقطه کنترل مرکزی روشنی ندارند. آخرین دسته از سیستم ها نیز ساختار سلسله مراتبی دارند، (شکل ۱-۸) . در انتهای دهه ۸۰-۱۹۷۰ تعریف مفاهیم و نوع سیستم های توزیع شده، مراحل نهایی خود را طی نمود. همانگونه که در شکل ۱-۹ دیده می شود، در ساختار اولیه، هر پایگاه سیستم توزیع شده، شامل یک یا چند مؤلفه نرم افزاری، برای دستیابی به منابع، در نظر گرفته شده اند. در ساختار نهایی، با توجه به ویژگی توزیع شدن وظایف یک نرم افزار منفرد در کل شبکه، سیستم های توزیع شده به صورت مجموعه ای از کامپیوترهای خودگردان که به صورت شبکه ای به یکدیگر وصل هستند و نرم افزارهای مرکب از مؤلفه های قرار گرفته در سایت های متفاوت و مجزا، تعریف شدند (شکل ۱-۱۰).



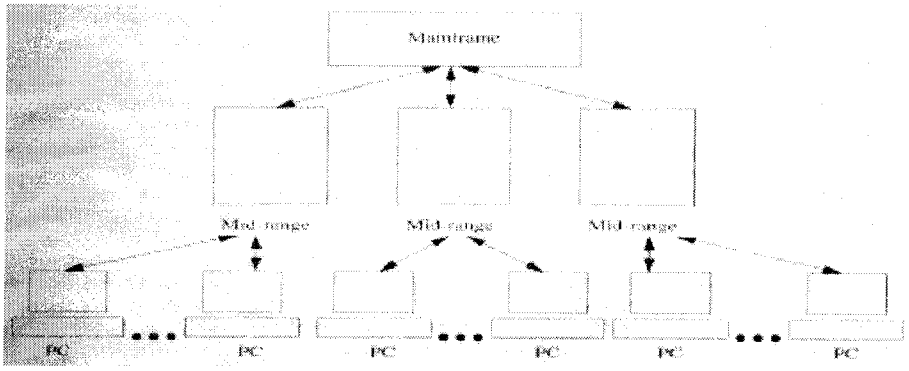
شکل ۱-۵- ترکیب بندی ساختاری و ارتباطی



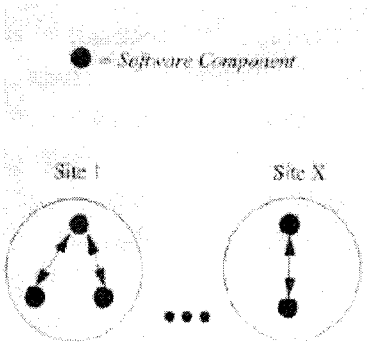
شکل ۱-۶- ساختار سلسله مراتبی



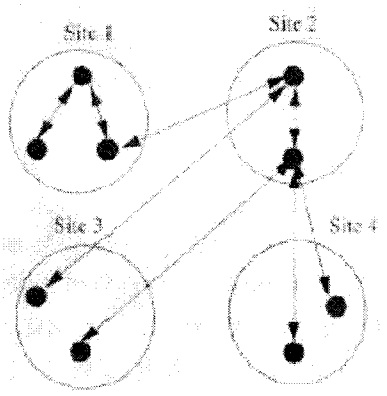
شکل ۱-۷- ترکیب بندی ساختاری تجهیزات هم رتبه (Peer)



شکل ۱-۸- ساختار سلسله مراتبی تجهیزات هم رتبه (Peer)



شکل ۱-۹- تعریف سیستم توزیع شده در دهه ۱۹۸۰



شکل ۱-۱۰- تعریف سیستم توزیع شده در حال حاضر

۱-۱۲- مشخصات و ویژگی های سیستم های توزیع شده

سیستم های توزیع شده، ۶ ویژگی کلی دارند، که در صورت طراحی و پیاده سازی دقیق سیستم، تأمین می شوند. این ویژگی ها عبارتند از:

۱- اشتراک منابع (Resource Sharing) :

بر اساس این ویژگی، منابع مربوط به یک کامپیوتر عضو سیستم توزیع شده، توسط اعضا دیگر سیستم، از طریق شبکه بین کامپیوترها، قابل دستیابی هستند. برای نیل به این ویژگی، لازم است مدیریت منابع مشترک در شرایط محیطی خاص سیستم های توزیع شده به درستی انجام گیرد. ویژگی « اشتراک » و« یکپارچگی »، دسترسی به منابع و کاربردها را ماورای مرزهای سازمانی محلی امکان پذیر می سازد، (Crossing Organization Boundaries) .

۲- باز بودن (Openness) :

ویژگی باز بودن در سیستم های توزیع شده، قابلیت گسترش سیستم را از جنبه های مختلف نشان می دهد. این ویژگی با میزان افزایش سرویس های به اشتراک گذارنده منابع، بدون ایجاد توقف و وقفه در سرویس های قبلی سنجیده می شود. این ویژگی را می توان از دو دیدگاه « قابلیت گسترش سخت افزاری » و « قابلیت گسترش نرم افزاری »

بررسی کرد. در دیدگاه اول امکان افزودن بخش های سخت افزاری، که توسط تولیدکنندگان مختلف ارائه می شوند، و از دیدگاه دوم امکان افزودن بخش های نرم افزاری تولیدکنندگان متفاوت مورد توجه قرار می گیرند. جهت دستیابی به این ویژگی ها، به ویژه جنبه های نرم افزاری، لازم است واسطه های نرم افزاری (Software Interfaces) به صورت استاندارد مورد پذیرش همگانی تعریف و مستندسازی شوند و در دسترس تهیه کنندگان و توسعه دهندگان دیگر قرار داشته باشند. از جمله سیستم های باز، می توان به سیستم UNIX اشاره کرد. منابع مورد استفاده در این سیستم، از طریق رویه های فراخوانی سیستم (System Calls) در دسترس قرار می گیرند. این مجموعه از رویه ها، توسط دیگر برنامه ها یا زبان ها قابل دسترسی هستند. افزایش سخت افزارهای جدید در UNIX با افزودن System Call های جدید یا پارامترهای جدید در رویه های قبلی امکان پذیر است. رویه های فراخوانی سیستم به وسیله بخش هسته (Kernel)، پیاده سازی می شوند. امکان ارتباط متقابل بین پردازه ها در سیستم یونیکس، ویژگی باز بودن را گسترش می دهد. این خاصیت، امکان دستیابی به منابعی که از طریق رویه های فراخوانی قابل دسترسی نیستند، را فراهم می سازد.

۳- همزمانی (Concurrency) :

قابلیت پردازش چندین کار (Task) در یک زمان را همزمانی گویند. وجود کامپیوترهایی با یک یا چند پردازنده در سیستم توزیع شده، امکان اجرای چندین کار را به طور همزمان فراهم می سازد. نرم افزارهای به کار گرفته شده در چنین سیستم هایی از وقوع تداخل (Conflict) در دسترسی به منابع مشترک، جلوگیری می کنند.

۴- قابلیت مقیاس پذیری (Scalability) :

با این ویژگی در سیستم های توزیع شده، هنگام افزایش مقیاس و اندازه سیستم، نیازی به تغییر کاربردها نخواهد بود. این ویژگی بر مبنای افزایش تقاضا در سیستم، تعریف می

شود. در چنین سیستم هایی، قابلیت های انعطاف لازم برای افزایش اندازه، به همراه استفاده مؤثر از سخت افزار و نرم افزار اضافی به وجود می آید.

۵- تحمل پذیری خطا (Fault Tolerance) :

توانایی بررسی و برخورد مناسب با وقوع خطا، تحمل پذیری خطا نام دارد. سیستم های تحمل پذیر خطا از قابلیت دسترسی (Availability) بالایی برخوردار هستند. میزان قابلیت دسترسی یک سیستم (توزیع شده) بر اساس اندازه زمانی استفاده مفید از سیستم مشخص می شود، تحمل پذیری خطا به دو روش به دست می آید:

الف – اضافات سخت افزاری (Hardware Redundancy)

ب – بازیابی نرم افزاری (Software Recovery)

در روش اول، استفاده از بخش های مهم سخت افزاری به صورت دو تایی (Duplication)، از تأثیر منفی خرابی های سخت افزاری جلوگیری می کند. در روش دوم با استفاده از نرم افزارهای ویژه حفاظتی (Safeguarding) تا حد امکان از خرابی ها جلوگیری شده و در صورت وقوع خرابی، فرایندهای مختلف بازیابی و بازسازی اطلاعات و نرم افزارها انجام می گیرند.

۶- شفافیت (Transparency) :

این ویژگی، مجزا بودن و پراکندگی مؤلفه های سیستم های توزیع شده را از دید کاربران و تهیه کنندگان کاربردها، پنهان می سازد و باعث ارائه دیدگاهی یکپارچه و یکدست از سیستم، به صورت یک کل، و نه مجموعه ای از اجزا، می شود. به طور کلی ۸ نوع شفافیت (ویژگی نیازمند پنهان سازی) تعریف شده اند:

الف – شفافیت دستیابی (Access Transparency) :

این ویژگی امکان دستیابی به شی های اطلاعاتی (Information Objects) را چه به صورت محلی و چه از راه دور، به طور یکسان، فراهم می سازد.

ب - شفافیت مکانی (Location Transparency) :

این ویژگی، امکان دستیابی به « شی های اطلاعاتی » بدون اطلاع از محل قرار گرفتن آنها را میسر می سازد.

به دو دسته الف و ب، شفافیت شبکه ای (Network Transparency) نیز اطلاق می شود.

پ - شفافیت همزمانی (Concurrency Transparency) :

امکان اجرای پردازه های همزمان که از منابع (شی های اطلاعاتی) مشترک بهره می جویند و پنهان ماندن این فرایند از دید کاربران، شفافیت همزمانی را فراهم می سازد.

ت - شفافیت تکرار (Replication Transparency) :

یکی از روش های افزایش قابلیت اطمینان و کارایی سیستم های توزیع شده، استفاده از منابع، به شکل چند تایی یا تکرار منابع و اطلاعات است. ویژگی شفافیت تکرار، وجود این تکرارها را از دید کاربران پنهان می سازد.

ث - شفافیت خرابی (Fault Transparency) :

با وجود این ویژگی، امکان اجرای صحیح، کامل و بدون وقفه برنامه های کابران علیرغم وجود خرابی های سخت افزاری و نرم افزاری فراهم می آید.

ج - شفافیت حرکت و تغییر مکان (Migration Transparency) :

با این ویژگی امکان حرکت و جابجایی شی ها (Objects) در کل سیستم، بدون تأثیر بر کاربردهای در حال اجرا فراهم می شود.

چ - شفافیت کارایی (Performance Ttransparency) :

به سیستم اجازه می دهد، با توجه به میزان تغییرات بار سیستم، وضعیت ترکیب بندی داخلی خود را، بدون ایجاد وقفه در اجرای برنامه ها، برای افزایش کارایی، تغییر دهد.

ح - شفافیت مقیاس پذیری (Scaling Transparency) :

این نوع شفافیت، به سیستم و کاربرها امکان می دهد، بدون تغییر ساختار سیستم و الگوریتم کاربردها، مقیاس و اندازه های خود را افزایش دهند.

ویژگی های اشتراک منابع، باز بودن سیستم و شفافیت ها، بر نحوه غلبه بر مشکل ناهمگونی مؤلفه های سیستم های توزیع شده، اثر بسیار زیادی دارند. وجود منابع مشترک (به اشتراک گذاشته شده) سخت افزاری و نرم افزاری تولیدکنندگان مختلف، نیاز به باز بودن سیستم را افزایش می دهد. در غیر این صورت استفاده از منابع، توسط کاربران امکان پذیر نخواهد بود. هم چنین وجود ویژگی « باز بودن » امکان ارتباط و همکاری متقابل نرم افزارها و سخت افزارهای تولیدکنندگان مختلف را فراهم می سازد.

۱-۱۳-مزایا و معایب سیستم های توزیع شده

مزایای سیستم های توزیع شده، به صورت ذیل قابل دسته بندی هستند :

الف – قابلیت اشتراک منابع (Salability) :

امکان استفاده مشترک از منابع یکدیگر توسط عناصر تشکیل دهنده سیستم توزیع شده که توسط پروتکل مشترک در یک شبکه به یکدیگر وصل هستند. سیستم اینترنت مثال خوبی برای سیستم توزیع شده است، که در آن کامپیوترها توسط پروتکل TCP/IP به شبکه وصل شده و از منابع مشترک استفاده می کنند.

ب - قابلیت گسترش (Expandability) :

این مزیت امکان اضافه شدن سیستم های جدید به عنوان عضوی از سیستم کلی را فراهم می آورد. در سیستم های توزیع شده، برای جلوگیری از اتلاف زمان و پول، منابع تنها هنگام نیاز به سیستم اضافه می شوند.

پ - خودگردانی محلی (Local Autonomy) :

در سیستم توزیع شده، هر سیستم عضو (گره)، توانایی مدیریت محلی منابع خود را دارد. این ویژگی برای سازمان هایی که ساختار تشکیلاتی آنها از بخش های مختلف در محل ها و موقعیت های متفاوت تشکیل می شود، مناسب خواهد بود.

ت - بهبود کارایی (Improved Performance) :

وجود تعداد زیاد کامپیوتر در کل سیستم و توزیع منابع بین آنها، توان پردازشی کل سیستم را افزایش داده و امکان ازدیاد کاربران را بدون افزایش زمان پاسخ سیستم فراهم می سازد.

ث - بهبود قابلیت اطمینان و قابلیت دستیابی (Mproved Reliability & Availability) :

به دلیل توزیع شدن منابع بین کامپیوترهای مختلف سیستم و تکرار منابع در نقاط مختلف و مدیریت مستقل آنها، قابلیت اطمینان و دستیابی، در صورت از کار افتادن بخشی از سیستم، با درجه بالایی حفظ می شود.

ج - امکان کاهش هزینه ها (Potential Cost Reduction) :

استفاده از سیستم های توزیع شده برای پردازش تقاضاها به طوری که سازمان های مختلف در آن دخیل باشند، باعث انجام وظایف نگهداری، پشتیبانی و راه اندازی به صورت مستقل توسط هر سازمان شده و هزینه های کل کاهش خواهد یافت.

معایب سیستم های توزیع شده را می توان به شرح زیر نام برد :

الف – وابستگی شبکه ای (Network Reliance) :

چون کامپیوترهای سیستم توزیع شده توسط امکانات شبکه ای به یکدیگر وصل هستند، وقوع هر گونه مشکل در شبکه، همچون مشکلات فیزیکی از قبیل قطع کابل های

ارتباطی، خرابی روترها و پل ها و ... ، بر کارکرد و کارایی سیستم اثر منفی خواهد گذاشت و هزینه های نگهداری و تعمیر سیستم (شبکه) نیز افزایش خواهند یافت.

ب - پیچیدگی ها (Complexities) :

نرم افزارهای تهیه شده برای استفاده در سیستم های توزیع شده جهت مدیریت منابع توزیع شده و نگه داری سیستم ها در مقابل وقوع خرابی ها در بخش های مختلف از پیچیدگی های خاصی برخوردار هستند که تهیه و گسترش آنها را مشکل می سازد.

پ - امنیت (Security) :

توزیع منابع در کامپیوترهای مختلف و امکان دسترسی به آنها از نقاط مختلف، ضعف عدم امنیت را در مقابل حمله غیر مجاز هکرها به منابع شخصی و خصوصی باعث می شود.

۱-۱۴-تکنولوژی های نرم افزاری سیستم های توزیع شده

هر چند مؤلفه های سخت افزاری نقش مهمی در سیستم های توزیع شده دارند، ولیکن نرم افزارهای به کار گرفته شده، چگونگی بینش نسبت به این سیستم ها را تعیین می کنند. از جمله وظایف مهم نرم افزارها به ویژه سیستم عامل ها، می توان به مدیریت منابع مشترک مثل Cpu ها، حافظه ها، تجهیزات جانبی، شبکه، انواع دیتا و نیز پنهان سازی طبیعت ناهمگون سخت افزارها و فراهم آوردن مفهوم « ماشین مجازی » جهت اجرای سهل و مناسب کاربردها اشاره کرد.

سیستم عامل ها در کامپیوترهای توزیع شده به دو دسته کلی تقسیم می شوند :

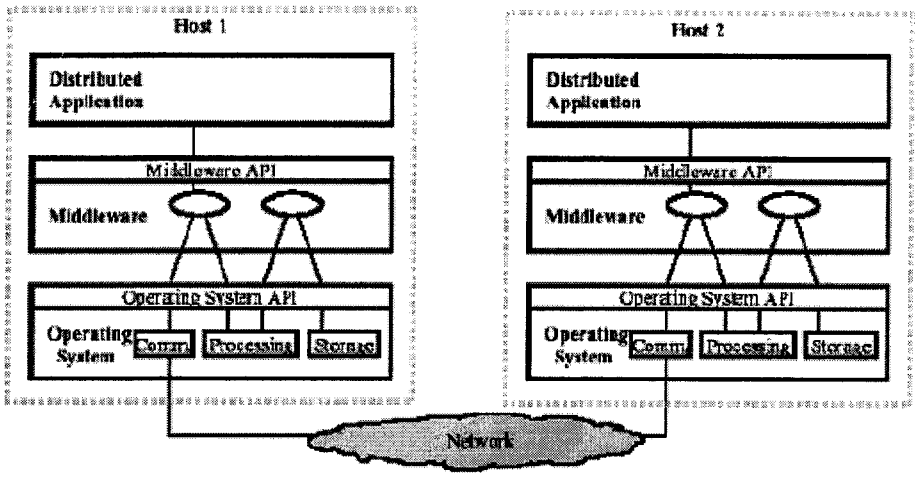
الف – سیستم عامل های توزیع شده (Distributed Operating System– Dos)

ب - سیستم عامل های شبکه ای (Nos–Netword Operating System)

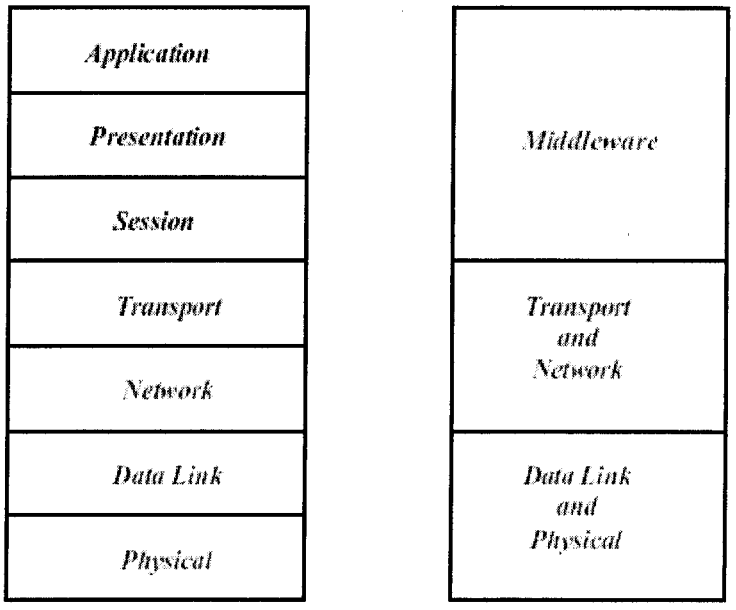
هر یک از سیستم های مذکور برای مدیریت دسته خاصی از سیستم های کامپیوتری توزیع شده استفاده می شوند. نوع الف (Dos) برای مدیریت سیستم های کامپیوتری با ارتباط محکم (Ttightly Coupled) متشکل از سیستم های چند پردازنده ای و سیستم های چند کامپیوتری همگن استفاده می شود. وظایف این دسته از سیستم های عامل عمدتاً مشابه وظایف سیستم عامل در سیستم های کامپیوتری تک پردازنده است، با این تفاوت که سیستم های توزیع شده از چند Cpu تشکیل شده اند. سیستم عامل های نوع ب (Nos) جهت مدیریت سیستم های کامپیوتری با ارتباط ضعیف (Loosely Coupled) متشکل از سیستم های چند کامپیوتری ناهمگن به کار می روند، در چنین سیستم هایی منابع محلی توسط کاربران دور (Remote Clients) نیز قابل دستیابی هستند.

هر دو دسته از سیستم عامل های فوق الذکر، از نظر کارایی جهت ارائه ویژگی های یک سیستم توزیع شده کامل، با تعاریف بیان شده، ناکافی هستند. یک سیستم عامل توزیع شده (Dos) قادر به مدیریت مجموعه ای از کامپیوترهای مستقل نبوده، هم چنین یک سیستم عامل شبکه ای (Nos) نمی تواند دیدگاه یکپارچه ای از کامپیوترهای توزیع شده ارائه کند. برای استفاده از ویژگی های مثبت هر دو دسته از سیستم عامل ها و رفع معایب آنها، مل قابلیت مقیاس پذیری و باز بودن سیستم عامل های شبکه ای و شفافیت (Transparency) وسهولت کاربرد سیستم عامل های توزیع شده، از تکنولوژی جدیدی به نام « میان افزار » یا " Middleware " استفاده می شود. این تکنولوژی به صورت لایه جدید نرم افزاری در سیستم عامل های شبکه ای استفاده می شود تا خاصیت ناهمگونی بسترهای زیرین را پنهان سازد. این لایه بین کاربردها و سیستم عامل شبکه ای قرار گرفته و سطح بالاتری از تجرید(تعمیم) (Bstraction) را ارائه می دهد. در شکل ۱-۱۱ ارتباط لایه « میان افزار » با دیگر بخش ها نشان داده شده است. در هر سیستم،

سیستم عامل شبکه ای محلی وظایف مدیریت منابع محلی را به عهده دارد و نقش « میان افزار » ها فراهم آوردن ویژگی شفافیت و پنهان سازی ناهمگونی بسترهای زیرین از دید کاربردها و کاربران است. نقش عملکردی (Functional) میان افزارها در مقایسه با مدل هفت لایه ای OSI در شکل ۱-۱۲، نشان داده شده است.



شکل ۱-۱۱- ارتباط لایه میان افزار با بخش های دیگر



شکل ۱-۱۲- مقایسه بین مدل ۷ لایه ای OSI و مدل مبتنی بر میان افزار

بر اساس محصولات و استانداردهای موجود در بازار فعلی، می توان میان افزارها را به چند دسته تقسیم کرد، از قبیل **DCE , RMI , RPC , SOCKET , DCOM , CORBA** .

نکته قابل توجه این است که میان افزارها به صورت کلاسیک و آکادمیک طراحی و توسعه نیافته اند. بلکه پس از معرفی و استفاده گسترده از سیستم عامل های شبکه ای (Nos)، سازمان های مختلف خود را با مشکل عدم امکان یکپارچه سازی کاربردهای مختلف شبکه ای مواجه دیدند. بدین دلیل تولیدکنندگان سیستم های نرم افزاری به طراحی و ساخت سرویس های سطح بالاتر و مستقل از کاربرد، مثل تراکنش های توزیع شده و تسهیلات ارتباطی پیشرفته، پرداختند. وجود « میا افزار » های مختلف در بازار، عدم وجود استانداردهای مورد پذیرش همگانی، عدم اطمینان از پایداری میان افزارهای فعلی در بلندمدت با توجه به ویژگی عدم قطعیت و متغیر بودن نیازهای کاربران و نیازهای جدید، عدم وجود چهارچوبی مطمئن و فراگیر برای سیستم های توزیع شده در زمینه های کاربردی خاص (Domain Specific Application) باعث عدم اطمینان سازمان های گسترش دهنده چنین سیستم ها و تکنولوژی هایی شده است. استانداردها و سازمان های استاندارد سازی کنونی نیز هماهنگی زیادی با یکدیگر نداشته و حتی محصولاتی که بر مبنای یک استاندارد خاص پیاده سازی می شوند، به ندرت ارتباط مناسب و مؤثر با یکدیگر برقرار می کنند (Interworking) . وجود چنین مشکلاتی در حیطه سیستم های توزیع شده و گسترش روزافزون این سیستم ها و نیازهای جدید کاربران، سازمان های دست اندر کار و نیروهای متخصص در این زمینه را واداشته است، تا روش های نوینی جهت برخورد با مسائل سیستم های توزیع شده ارائه کنند، از جمله ارائه مفاهیم روش شناسی جدید برای مدل سازی سیستم های توزیع شده و ارائه مدل های مرجع مجرد (Abstract) جهت تعریف و تجزیه و تحلیل دقیق و کامل سیستم های توزیع شده است.

استاندارد های مختلفی که در زمینه سیستم های توزیع شده، وجود دارد به سه دسته کلی تقسیم می شوند :

الف – استانداردهایی که جنبه های مدل سازی سیستم ها را در نظر می گیرند، مثل MOF OMG`s UML.

ب - استانداردهایی که جنبه های معماری و پیاده سازی سیستم را مد نظر دارند، مثل OMG`s CORBA JAVA RMI.

ج - استانداردهایی که هر دو جنبه را مورد بررسی قرار می دهند، مثل ISO`s RM - ODP.

فصل دوم

میان افزارها

۲-۱- مقدمه

نیازهای تجاری بسیاری، طراحی سیستم های توزیع شده را ایجاب کرده اند. اولاً، به هم پیوستن و اتحاد شرکت ها و کمپانی ها در حال افزایش است. بخش های مختلف شرکت های جدید التأسیس، باید سرویس های یکنواختی به مشتریان ارائه دهند و این امر نیاز به یکپارچگی سیستم های IT آنها را به وجود آورده است، از طرفی زمان برای این یکپارچه سازی بسیار کوتاه بوده است. ثانیاً زمان لازم برای ارائه سرویس های جدید کاهش یافته است، لذا باید از مؤلفه ها و امکانات موجود استفاده کرده و آنها را در یک سیستم، یکپارچه ساخت. هر یک از مؤلفه ها ممکن است در بستر سخت افزاری و سیستم عامل های متفاوتی تعریف شده باشند. ثالثاً، اینترنت، امکانات مفید و جدیدی برای معرفی محصولات و سرویس ها برای تعداد بسیار زیادی از افراد فراهم می آورد، به گونه ای که برآورد میزان مقیاس پذیری سیستم مشکل است. سیستم های توزیع شده، می توانند مؤلفه های بازمانده از قبل را یکپارچه ساخته و از سرمایه گذاری های گذشته حفاظت کنند. زمان ارائه محصولات به بازبر را کاهش داده، مقیاس پذیری را افزایش داده و تحمل پذیری در برابر خرابی ها را افزایش دهند. باید به این نکته توجه داشت که، طراحی و ساخت سیستم های توزیع شده واقعی، بسیار مشکل تر از ساخت سیستم های متمرکز Client-Server است، زیرا در چنین سیستم هایی، ارتباطات پیچیده ای بین مؤلفه ها موردنیاز است و باز بودن سیستم خطرات حملات ضدامنیتی را افزایش می دهد. وظیفه میان افزارها پنهان سازی مشکلات فوق از دید طراحان برنامه های کاربردی است. هر چه ویژگی های میان افزارها بهبود و گسترش یابند، زمینه پذیرش آنها در بازار و صنعت افزایش خواهد یافت. برای برپایی سیستم های توزیع شده، مهندسین نرم افزار باید از نوع میان افزار مناسب برای کاربرد خاص موردنظر، مناسب ترین آنها برای حل مشکلات موجود، چگونگی استفاده و بهره برداری از میان افزار انتخاب شده در معماری، طراحی و پیاده سازی سیستم های

توزیع شده اطلاع کافی حاصل کنند. برای انتخاب و استفاده از میان افزار خاص، باید بر شناخت نیازهای غیرعملیاتی (Non-Functional) تأکید کرد و برای این افراد از روش های مناسبی در مهندسی نرم افزار استفاده کرد. انتخاب میان افزار از دیدگاه طراحی سیستم چندان شفاف نیست.

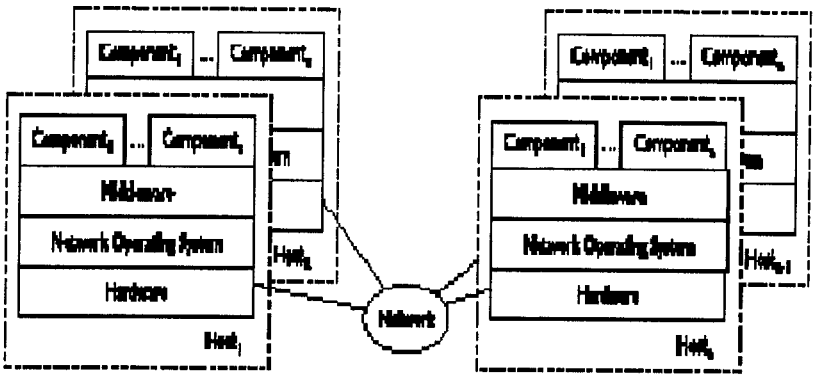
۲-۲-نیازهای میان افزارها

میان افزارها، لایه ای نرم افزاری بین لایه کاربردها و سیستم عامل شبکه ای هستند و ارتباط و هماهنگی بین مؤلفه های سیستم را بر روی سیستم های میزبان مختلف در گستره سیستم توزیع شده به عهده دارند. این لایه، طراحان کاربردها را از انجام برخی وظایف معاف می کنند، مثل کنترل همزمانی اجرای برنامه ها، مدیریت تراکنش ها و ارتباطات شبکه ای.

-- ارتباطات شبکه ای

همان گونه که در شکل زیرنشان داده شده است، مؤلفه های مختلف سیستم توزیع شده ممکن است بر روی سیستم های میزبان مختلف در کل سیستم قرار داشته و برای یکپارچه سازی سیستم، نیاز به ارتباطات گوناگون بین آنها خواهد بود. این ارتباطات از طریق پروتکل های ارتباطی همچون مدل مرجع ۷ لایه ای ISO/OSI انجام می گیرند. سیستم های توزیع شده معمولاً بالاتر از لایه انتقال (Transport)، مثل TCP یا UDP قرار می گیرند. لایه های پایین تر توسط سیستم عامل پیاده سازی می شوند. اگر فقط به این نکته اکتفا شود، و وظایف لایه های نمایش (Presentation) و جلسه (Session) به طراحان کاربردها واگذار گردد، مشکلاتی همچون افزایش هزینه های طراحی، هزینه های زمانی و آسیب پذیری در برابر خطاها پیش خواهد آمد. بر این اساس وظایف مذکور نیز به لایه میان افزار واگذار می گردد. پارامترهایی که مؤلفه ها (Components) برای درخواست سرویس خاصی باید به مؤلفه های ارائه کننده سرویس

بفرستند، از ساختار دیتایی پیچیده ای برخوردارند و پیاده سازی میان افزاری لایه نمایش می تواند این ساختار دیتا را به شکل قابل انتقال از طریق پروتکل انتقال، مثل دنباله ای از بایت ها، تبدیل کند. این عمل با عنوان Marshalling و عمل عکس آن Unmarshalling شناخته می شوند.



شکل ۲-۱

-- هماهنگ سازی (Coordination)

مؤلفه ها یا شی هایی که در یک سیستم میزبان قرار دارند، می توانند همزمان اجرا شوند و باید برای ارتباط با یکدیگر هماهنگ شوند. وظیفه هماهنگی بین شی ها (مؤلفه ها) در لایه جلسه (Session) توسط میان افزار پیاده سازی می شود. این هماهنگی با روش های مختلفی به دست می آید، مثل ارتباطات همزمان (Asynchronous). میان افزارها از مکانیزم های مختلفی برای هماهنگ و همزمان سازی برنامه ها استفاده می کنند، مثل استفاده از مفاهیم فعال سازی (Activation) و غیر فعال سازی (Deactivation) پروسس ها یا استفاده از مفاهیم چند بندی (Threading–Multi) برای پاسخگویی به تقاضاهای مختلف که به صورت همزمان درخواست شده اند.

-- قابلیت اطمینان (Reliability)

یکی از ویژگی های مهم میان افزارها، ارائه درجه قابل قبولی از قابلیت اطمینان در رسیدن پیام ها به مقصد و تنظیم ترتیب درست دریافت آنهاست. برای این منظور از روش های مختلفی استفاده می شود، مثل:

- Best- Effort
- At-Most-Once
- At-Least-Once
- Exactly Once

در روش اول، اطمینانی از اجرای تقاضا وجود ندارد. در روش دوم، اجرای فقط یک بار تقاضا تضمین می شود. در روش سوم اجرای حداقل یک بار تقاضا تضمین می شود. در روش آخر، اجرای یک بار و فقط یک بار تقاضا تضمین می شود. در سیستم های توزیع شده، تراکنش ها (Transaction)، عملیات ابتدایی ساده (Primitive) جهت رسیدن به ویژگی قابلیت اطمینان هستند. تراکنش ها خواص ACID را دارا هستند. میان افزارهایی که در کاربردهای بحرانی به کار می روند، بایستی از تراکنش های توزیع شده پشتیبانی کنند. یکی دیگر از روش های بالا بردن قابلیت اطمینان، استفاده از تکرار و کپی کردن (Replication) مؤلفه ها یا شی ها در کل سیستم توزیع شده است. در این صورت باید حالت های داخلی مؤلفه ها در کپی های مختلف همسان بمانند.

-- مقیاس پذیری (Scalability)

این ویژگی، امکان گسترش و افزایش بار سیستم را در آینده فراهم می سازد. در سیستم های متمرکز یا Client/Server ، مقیاس پذیری سیستم به وسیله حداکثر بار قابل پردازش محدود می شود. در سیستم های توزیع شده به کمک روش تقسیم بار در کل سیستم، تغییرات بار بدون تأثیر و تغییر در معماری سیستم، طراحی و کدنویسی مؤلفه ها

(شی ها) انجام می گیرد. برای رسیدن به این هدف باید به ابعاد مختلف ویژگی شفافیت (Transparency) که در مدل مرجع ODP تعریف شده اند، توجه کرد. به عنوان مثال، ویژگی شفافیت دستیابی (Access Transparency) چگونگی دستیابی مؤلفه ها (شی ها) به سرویس های شی های دیگر، مستقل از دور (Remote) یا نزدیک (Local) بودن آن، را مشخص می کند. به عنوان مثالی دیگر، شفافیت محل (Transparency Location) امکان ارتباط بین مؤلفه ها (شی ها) را بدون نیاز به اطلاع از محل آنها، به وجود می آورد. اگر مؤلفه ها بتوانند بدون اطلاع از محل فیزیکی یکدیگر و بدون نیاز به تغییر رویه (متد)، از سرویس های یکدیگر استفاده کنند، مکانیزم تعادل بار (Load Balancing)، برای تنظیم بار سیستم، می تواند شی ها (مؤلفه ها) را به ماشین دیگر مهاجرت دهد، (Migration). پنهان سازی این حرکت از دید کاربر، شفافیت مهاجرت (Migration Transparency) نام دارد. تهیه کپی های مختلف از شی ها و تکرار آن ها در ماشین های مختلف (Replication) جهت مؤلفه هایی که درخاست های زیادی دارند، و برای تنظیم بار سیستم، با عنوان شفافیت تکرار، مورد بررسی قرار می گیرد. پیاده سازی شفافیت های مذکور (دستیابی، محل، مهاجرت و تکرار) توسط عملیات ساده (Primitive) سیستم عامل های شبکه کار بسیار مشکلی لست و برای فائق آمدن بر این مشکلات از میان افزارها کمک گرفته می شود.

-- ناهمگونی یا عدم تجانس (Heterogeneity)

سیستم های توزیع شده، باید توانایی به کار گیری مؤلفه های موجود و آماده (Off-The-Shelf) و جدید را داشته باشند. چنین مؤلفه هایی معمولاً از زمینه های مختلف مثل سخت افزار، سیستم عامل، زبان برنامه نویسی و حتی خود میان افزار، از طبیعت ناهمگون و نا متجانس برخوردارند. ناهمگونی های مختلف در زمینه های فوق الذکر مثل فرمت دیتا در سیستم های مختلف، زبان ها و روش های برنامه نویسی مختلف بایستی

توسط میان افزارهای مناسب مدیریت و پوشانده شوند. جهت نیل به این هدف از میان افزارهای مختلفی استفاده می شود، زیرا هر دسته ای از آنها در زمینه خاصی، از توانایی مناسب برخوردار هستند. مثلاً سیستم COM بر روی سیستم عامل ویندوز کار می کند. هم چنین بعضی از میان افزارها به عنوان Back Bone و بعضی دیگر به صورت محلی قابل استفاده هستند. لذا بر روی یک سیستم توزیع شده، میان افزارها باید بتوانند با یکدیگر ارتباط برقرار کنند.

۲-۳-راه حل های ارائه شده توسط میان افزارها

برای بررسی توانایی میان افزارهای موجود، آنها را به ۴ دسته کلی تقسیم می کنند :

تراکنشی (Transactional)، پیام گرا (Message-Oriented)، رویه ای (Procedural)، شی گرا (مؤلفه گرا) (Object/Component-Oriented). این تقسیم بندی بر اساس عملیات پایه ای (Primitives) میان افزار انجام گرفته است.

۱- میان افزارهای تراکنشی :

تراکنش ها، فعالیت هایی هستند که خواص ACID را دارا بوده و مؤلفه های آنها بر روی سیستم های میزبان (Host) مختلف و توزیع شده قرار می گیرند. میان افزارهای تراکنشی از پروتکل ثبت دو مرحله ای (Two-Phase Commit Protocol) استفاده می کنند. از جمله محصولات موجود در بازار می توان به Transarc,s و BEA,s Tcxedo و IBM,s CICS اشاره کرد. به کمک این دسته از میان افزارها و مدل Client-Server ، تقاضاهای ارسالی از طرف client ها برای دستیابی به سرویس ها، به صورت تراکنش نوشته می شوند. مؤلفه های سرور و Client بر روی ماشین های مختلف قرار گرفته و انتقل تقاضا در شبکه به صورت شفاف انجام می گیرد. تقاضاها (Requests) به صورت هماهنگ (Synchronous) و غیر هماهنگ (Asynchronous) انجام پذیر هستند. یک Client می تواند، بیش از یک سرویس را در طی یک تراکنش تقاضا

کند. این ویژگی با در نظر گرفتن دو مرحله ای بودن ثبت تراکنش ها انجام پذیر است. اکثر نظارت کنندگان تراکنش ها، تنظیم بار و تکرار (کپی) مؤلفه های سرور را انجام می دهند. این ویژگی قابلیت مقیاس پذیری و اطمینان سیستم را افزایش می دهد. به دلیل امکان قرار گرفتن مؤلفه ها (شی ها) ی میان افزارهای تراکنشی بر روی سخت افزار و سیستم عامل های مختلف، عدم تجانس و ناهمگونی مورد حمایت قرار می گیرد. به دلیل استفاده از پروتکل DTP (Distributed Transaction Protocol) سیستم های مدیریت دیتابیس مختلف نیز قابل استفاده هستند.

میان افزارهای موجود، به طور کامل ویژگی ناهمگونی را نمی پوشانند، چون عملیات اساسی (Primitives) لازم برای نمایش و استفاده پارامتریک از ساختارهای دیتای پیچیده در تقاضای سرویس ها و تبدیل اطلاعات به شکل مناسب برای ارسال چنین ساختارهایی (Marshalling) ، را ندارند. علیرغم توانایی میان افزارهای تراکنشی جهت برپایی سیستم های توزیع شده، آنها ضعف هایی نیز دارند، از جمله به وجود آمدن سرآیندهای غیر ضروری و اضافی، زمانی که نیاز به استفاده از تراکنش نباشد، نگاشت مناسب بین ساختار دیتا و پارامترهای موردنیاز برای تقاضای سرویس ها (Unmarshalling) باید به صورت دستی انجام گیرد، هر چند API برای ثبت دو مرحله استاندارد شده است ولیکن روش استاندارد شده ای برای تعریف سرویس هایی که مؤلفه های سرور ارائه می دهند، به دست نیامده است. این کمبودها قابلیت حمل (Portability) سیستم توزیع شده را بین چند ناظر (Monitor) تراکنش های مختلف را کاهش می دهد.

۲- میان افزارهای پیام- گرا :

میان افزارهای پیام- گرا (MOM) ارتباط مؤلفه ها (شی ها) ی توزیع شده را، به کمک مبادله پیام ها، تسهیل می کنند. از جمله محصولات پیام گرا، می توان به IBM,s

MQ Series و Sun,s Java Message Queues اشاره کرد. ارتباطات شبکه ای بین client ها و سرویس ها، به وسیله مبادله پیام ها انجام می گیرند، مثل اعلام یک رخداد (Event) یا تقاضای یک سرویس، محتوی پیام Client پارامترهای موردنیاز سرور را نیز در بر می گیرد و نتایج حاصل از اجرای سرویس مورد تقاضا، به وسیله پیام پاسخ (Reply-Message) برگردانده می شوند. ویژگی عمده این میان افزارها، امکان ارتباط بین سرور و Client به صورت غیر هماهنگ (Asynchronous) است. بر این اساس همزمان با تحویل پیام توسط میان افزار Client می تواند فعالیت های دیگر پردازشی خود را ادامه دهد. عدم نیاز به همزمانی ارسال و تحویل پیام ها توسط سرور و Client قابلیت مقیاس پذیری سیستم را افزایش می دهد. ویژگی دیگر میان افزارهای پیام گرا، پشتیبانی از ارتباطات گروهی (Group Communication) به صورت شفاف است. نقطه ضعف این میان افزارها پیاده سازی دستی تقاضاهای همزمان (Synchronous) است. به دلیل استفاده از مکانیزم صف پیام ها در MOM، تحمل پذیری خطا و قابلیت اطمینان سیستم افزایش می یابد. پیام هایی که در صف قرار می گیرند تا زمانی که گیرنده آنها را نخواند، در آن باقی می مانند. چون در میان افزارهای پیام- گرا، از مکانیزم صف برای ارتباطات راه دور استفاده شده و برای ارتباطات نزدیک یا محلی از آن استفاده نمی شود، لذا شفافیت دستیابی به دست نمی آید. به همین دلیل امکان شفافیت مهاجرت و شفافیت تکرار نیز وجود نداشته و فرایند مقیاس پذیری سیستم پیچیده خواهد شد. هم چنین برقراری و مدیریت صف ها در سرور و Client نیز مشکل بوده و قابلیت انعطاف را کاهش می دهد. میان افزارهای MOM ، به دلیل عدم انجام فرایند Marshalling ، از ناهمگونی حمایت نمی کنند. با توجه به نقاط ضعف و قوت این دسته از میان افزارها، می توان به این نتیجه رسید که آنها برای استفاده در فرایندهای « اعلان رخداد » توزیع شده

و معماری های Publish/Subscribe مناسب هستند. این میان افزارها از خواص تراکنشی در تحویل پیام ها حمایت نمی کنند.

۳- میان افزارهای رویه ای :

رویه های فراخوانی از راه دور (Remote Procedure Call- RPC) در ابتدا توسط شرکت Sun MicroSystems به صورت بخشی از بستر ONC (Open Network Computing) ارائه شد. شرکت SUN از این ابزار در تمام سیستم عامل های خود استفاده کرده و آن را به عنوان استاندارد کنسرسیوم X/OPEN به عنوان بخشی از DEC ثبت کرد. امروزه PRC ها در اکثر نسخه های UNIX و سیستم عامل های سری Windows میکروسافت به کار می روند. از این روش، مؤلفه (شی) های سرور، به عنوان برنامه های RPC تعریف می شوند. هر برنامه RPC تعدادی رویه پارامتردار ایجاد می کند. Client هایی که بر روی میزبان های مختلف قرار دارند، می توانند رویه های RPC را از سرتاسر شبکه فراخوانی کنند. این عمل توسط میان افزارهای رویه ای و با انجام عمل Marshalling پارامترها به صورت پیام های مناسب ارسال، انجام می گیرد. مؤلفه سرور نیز با تبدیل عکس پیام ها (Unmarshalling) و اجرای رویه های مربوط، نتیجه را به شکل مناسب در آورده و به Client می فرستد. RPC ها ارتباط متقابل همزمان بین یک سرور و یک Client را انجام می دهند. ارتباطات غیر همزمان و پخشی (Multicast) توسط این میان افزارها حمایت نمی شوند. آنها از مفهوم حداکثر یک بار (At-Most-Once)، برای افزایش قابلیت اطمینان استفاده می کنند و مفاهیم تراکنش و دقیقاً یک بار (Exactly Once) حمایت نمی شوند. ویژگی مقیاس پذیری در میان افزارهای رویه ای بسیار محدود است، چون مکانیسم های تکرار (Replication) در آنها وجود ندارد، مثل سیستم های UNIX و Windows . این دسته از میان افزارها، با زبان های برنامه نویسی مختلف و بر روی بسترهای سخت افزاری و سیستم عاملی متفاوت قابل استفاده

بوده و ویژگی حمایت از ناهمگونی و عدم تجانس را دارا هستند. استانداردهای میان افزارهای رویه ای، از نمایش دیتای استاندارد شده جهت انتقال دیتا، استفاده می کنند. به عنوان مثال از استاندارد NDR (Network Data Representation) در DCE نام برده می شود.

از میان افزارهای رویه ای، به عنوان اینترفیس های نگاشت بین ساختار دیتا و پیام های مناسب (Unmarshalling)، جهت بهینه سازی و بهبود میان افزارهای MOM و تراکنشی استفاده می شود. یکی از معایب مهم این میان افزارها، انعکاسی نبودن آنهاست، یعنی رویه هایی که RPC فعال می کند، نمی توانند برنامه PRC دیگری را برگردانند. این مشکل در میان افزارهای شی گرا و مؤلفه ای حل می شود.

۴- میان افزارهای مؤلفه (شی) گرا :

میان افزارهای شی گرا، نسخه تکمیل شده RPC ها هستند. ایده اصلی این تحول، استفاده از مبانی شی گرایی مثل مشخصه سازی از طریق مرجع ها (References) و توارث است، که برای گسترش سیستم های توزیع شده به کار می روند. به عنوان محصولاتی تجاری از این دسته میان افزارها می توان به JAVA RMI, DCOM, CORBA اشاره کرد. از جمله آخرین سیستم ها، تکنولوژی EJB (Enterprise Java Beans) قابل ذکر است. این میان افزارها، از تقاضاهای شی ای توزیع شده حمایت می کنند، یعنی شی Client اجرای عملی را از شی سرور در ماشین میزبان دیگر، تقاضا می کند. برای این کار Client از مرجع شی سرور یا Server Object Reference استفاده می کند. برای هماهنگی عملیات، از روش ارتباط و تقاضای همزمان بین سرور و Client استفاده می شود. در این میان افزارها از روش های مختلفی برای فعال سازی، استفاده می شود. هم چنین از مکانیزم های چند رشته ای (چند بندی) (Multi-

Threading) برای اجرای همزمان چند تقاضا استفاده شده است. میان افزار CORBA از ارتباطات گروهی، با سرویس های Event/Notification حمایت می کند.

قابلیت اطمینان در این میان افزارها به صورت پیش فرض، با روش « حداکثر یک بار » تأمین می شود. هم چنین استفاده از مفهوم « استثناء (Exception) » برای آشکار سازی خطا در حین اجرای تقاضاها امکان پذیر است. مکانیزم های پیام رسانی (Messaging) و سرویس های اعلان (Notification) در CORBA قابلیت اطمینان « دقیقاً یک بار » را فراهم می آورد. در این میان افزارها، مفهوم تراکنش نیز استفاده شده است. در CORBA ، سرویس Object Transaction برای دسته بندی تقاضاها از شی های توزیع شده به صورت تراکنش به کار می رود. میکروسافت نیز تکنولوژی Transaction Server و COM را یکپارچه ساخته است.

سرویس Java Transaction چنین امکانی را برای RMI فراهم آورده است. ویژگی مقیاس پذیری (Scalability)، در میان افزارهای شی گرا محدود است. مثلاً در بعضی از نسخه های CORBA از روش تنظیم یا تعادل بار (Load- Balancing) و به کار گیری Name- Server استفاده شده است. ویژگی پشتیبانی از ناهمگونی در میان افزارهای شی گرا به روش های مختلف تأمین شده است. در CORBA و COM به دلیل پیوند چندگانه زبان های برنامه نویسی (Multiple Programming Language Binding)، شی های سرور و Client را می توان به زبان های مختلف نوشت. در هر دو میان افزار، نمایش دیتای استاندارد جهت پوشش ناهمگونی در کل سیستم استفاده می شود. در Java- RMI از مفهوم ماشین مجازی جاوا (Java Virtual Machine) استفاده شده است.

ارتباط بین میان افزارهای شی گرا، با استفاده از مفاهیم و استانداردهای مختلف امکان پذیر است. در CORBA از استاندارد IIOP (Internet Inter- ORB Protocol)

جهت مبادله دیتا بین مؤلفه ها استفاده می شود. میان افزار Java- RMI از این پروتکل به عنوان پروتکل انتقال برای درخواست متد از راه دور استفاده می کند. بر این اساس Client جاوا می تواند متدی را از سرور CORBA و از راه دور فراخوانی کند (و بر عکس). هم چنین CORBA قابلیت ارتباط با COM را داراست.

میان افزارهای شی گرا ویژگی های مثبت انواع دیگر میان افزارها را دارا هستند و تنها عاملی که از به کار گیری آنها در سیستم های مقیاس گسترده ممانعت می کند، محدود بودن ویژگی مقیاس پذیری (Scalability) آنهاست.

۲-۴-وضعیت موجود میان افزارها (MiddleWare State-Of-The-Art)

هر چند میان افزارها، با موفقیت در بسیاری کاربردهای تجاری و صنعتی به کار گرفته شده اند، بعضی کمبودها استفاده از آنها را در برخی کاربردها، نا ممکن ساخته است. این کمبودها عدم انعطاف پذیری مناسب، عدم امکان پاسخگویی به نیازهای معتبر، عدم قابلیت گسترش به شبکه های گسترده تر از Lan ، عدم وابستگی (Dependability) و عدم امکان استفاده در شبکه های بی سیم، را باعث شده اند. برای کاهش و یا حذف کامل کمبودهایی که باعث ضعف میان افزارها شده اند، تحقیقات و بررسی های گسترده ای در حال انجام است. در ادامه بحث وضعیت موجود تحقیقات انجام شده برای بهبود و بهینه سازی میان افزارها بررسی شده اند.

۲-۵-انعطاف پذیری میان افزارها (Flexible MiddleWares)

-- Trading :

بیشتر میان افزارها برای شناسایی مؤلفه ها (شی ها) از روش نامگذاری (Naming) استفاده می کنند. به عنوان مثال MOM ها از صف های پیام نامگذاری استفاده می

کنند. DCE از سرویس فهرست (Directory Service) استفاده می کند. CORBA سرویس Naming Service دارد. میان افزار COM سرویس Monikers (نام های مستعار) دارد و Java/RMI از RMI Registry استفاده می کند. بر این اساس هر Client قبل از فعال کردن تقاضای خود، باید نام اختصاصی را برای پیدا کردن مرجع مؤلفه سرور بیابد و برای این منظور سرور خاص نام گذاری را شناسایی کند. این امر تا حدی با ویژگی شفافیت مکان تعارض خواهد داشت. در بعضی از کاربردها، نیازی به شناختن شی سرور جهت گرفتن سرویس، توسط شی Client وجود ندارد و استفاده از این رابطه یک به یک انعطاف پذیری سیستم را کاهش می دهد. در این روش Client ها نمی توانند به طور پویا از بهترین ارائه کننده سرویس بهره ببرند.

برای افزایش انعطاف پذیری بیشتر، از مفهوم Trading به جای Naming استفاده شده است. استاندارد ISO/ODP مشخصات بنیادین سرور Trading را تعریف کرده است. در این روش مؤلفه ها یا شی ها بر اساس نوع سرویسی که ارائه می دهند، مشخص می شوند. Trader نوع سرویس قابل ارائه توسط سرور و ویژگی کیفیت سرویس (QOS) تضمین شده آن را ثبت می کند. Client برای دریافت سرویس، شی های سرور مربرط به نوع خاصی از سرویس و QOS مشخصی را از Trader تقاضا می کند. Trader مشخصات دریافتی را بررسی کرده و مؤلفه سرور مناسب را انتخاب کرده و مرجع (Reference) مربوط را به Client بر می گرداند. در انتها سرور و Client بدون مداخله Trader با یکدیگر ارتباط برقرار می کنند. ایده و مفهوم Trading در محصولات میان افزاری بسیاری پذیرفته شده و پیاده سازی شده است. سازمان OMG مفاهیم ODP Trader را پذیرفته و سرویس Trading Service را تعریف کرده است.

-- **Reflection** :

روش دیگر برای افزایش انعطاف پذیری محیط اجرا برای مؤلفه ها (شی ها)، انعکاس یا Reflection است. از این روش، قبلاً در زبان های برنامه نویسی استفاده شده است. برنامه ها، از این روش برای کشف « کلاس » یا « نوع » و تعریف متد فراخوانی و دریافت (Invocation) در زمان اجرا استفاده می کنند. این روش، هم چنین، قبلاً توسط برخی میان افزارها به کار رفته است. مفاهیم مخزن واسطه ها (Iinterface Repository) و درخواست پویا (Dynamic Invocation) در CORBA ، برای کشف « نوع » مؤلفه های شناخته شده سرور و فعال سازی تقاضاهای پویا توسط Client استفاده می شوند.

تحقیقات فعلی در زمینه میان افزارهای انعکاسی، علاوه بر موارد فوق الذکر، پروتکل های فوق شی (Meta-Object) را نیز در بر گرفته است. این پروتکل ها برای بررسی و جستجو (Inspection) و تطبیق (Adaptation) محیط اجرای میان افزار به کار می روند.

یکی از روش های پیشنهادی، فوق مدل محیطی (Environment Meta-Model) است. بررسی و جستجوی فوق مدل محیطی، رفتار میان افزارها را به کمک « رخداد » ها (events) درخواست می کند، مثل دریافت پیام، وارد شدن تقاضاها در صف، تبدیل شکل نمایش دیتا (Unmarshalling)، ایجاد بندها (Threads) و زمان بندی تقاضاها.

۲-۶-پروتکل های انتقال در سطح کاربرد (Application-Level Transport Protocols)

کاربردهایی وجود دارند که هنگام استفاده از آنها، میان افزارها سرآیند اضافی و غیر ضروری، جهت شکل دهی دیتا (Marshalling)، ایجاد می کنند. مثل نمایش دیتای

وابسته به کاربرد (Application-Specific) که نیازی به پوشاندن ناهمگونی شبکه توسط میان افزار ندارد. بنابراین یکی از کاربردهای مهم انعکاس، شکل دهی دیتا خواهد بود. یک راه حل این مسئله، ترکیب میان افزار و زبان های Mark-up مثل XML است. بر این اساس، مستندات XML به صورت رشته بایت های تفسیر نشده، به وسیله میان افزار منتقل می شوند. این روش بر اساس ویژگی XML مبنی بر وجود ترجمه های معنایی بین ساختارهای دیتا و تعاریف موجود در زبان های Mark-up مثل FPML و FIXML ، پیشنهاد شده است. از سوی دیگر پروتکل HTTP که در ابتدا XML از آن استفاده کرده است، برای رسیدن به قابلیت اطمینان بالا مناسب نیست. در حال حاضر سازمان OMG فرایندی را جهت ارائه امکان ارتباط عملکردی متقابل، بدون محدودیت، بین ساختارهای دیتا در CORBA و مستندات ساخت یافته XML آغاز کرده است.

۲-۷-میان افزارهای مقیاس پذیر

همانگونه که اشاره شد، میان افزارهای فعلی در گستره شبکه های LAN به خوبی استفاده شده اند، ولیکن محدودیت هایی در استفاده از آنها در سیستم های گسترده کلی (جهانی) وجود دارد، مثل عدم امکان تکرار منابع در سطح توزیع شده جهانی. تحقیقاتی که در حال حاضر انجام می گیرد، بر روش تکرار غیر شفاف (Non-Transparent Replication) تمرکز دارد. مسئله تکرار (Replication) در سیستم های توزیع شده، توسط آقای تننبام، در پروژه Globe مطرح شده است.

هدف این پروژه، ارائه میان افزاری شی گرا است که بتواند تا حد تعداد بسیار زیادی (میلیون ها) از کاربران افزایش (گسترش) یابد. برای رسیدن به این هدف، به طور گسترده ای، از مفهوم تکرار استفاده می شود. در این پروژه، از استراتژی تکرار استفاده نمی شود، بلکه سیاست های تکرار بستگی به نوع شی دارد (Object-Type Specific) و

توسط طراحی شی مشخص می شود. بنابراین در Globe ، هر نوع شی از استراتژی فعال خاص خود برای تکرار بهره می جوید.

۲-۸-میان افزارهای بلادرنگ (Real-Time)

اکثر میان افزارهای موجود، کارایی محدودی در سیستم های بلادرنگ و تعبیه شده (Embeded) دارند، زیرا در آنها کلیه تقاضاها (Requests) دارای یک اولویت هستند. هم چنین نیازهای حافظه ای در میان افزارها از به کار گیری آنها در سیستم های تعبیه شده جلوگیری می کند. میان افزار TAO نمونه ای از CORBA بلادرنگ است که برای تقاضاها، اولویت و زمان بندی قائل می شود. نسخه CORBA ۳,۰ بر اساس این نمونه و برای استاندارد سازی میان افزارهای حداقل (Minimal) و بلادرنگ طراحی شده است.

۲-۹-میان افزارهای محاسبات موبایل

در حال حاضر، میان افزارها، پهنای باند زیادی جهت ارتباطات شبکه ای استفاده می کنند. پروتکل های شبکه ای بی سیم (Wireless Lan) مثل Wave-Lan پهنای باند محدود و معقولی را جهت ارتباط استفاده می کنند و در این شبکه ها، حداکثر فاصله مجاز برای امکان ارتباط بین گره ها، چند صد متر است، در صورتی که میزبان های موبایل در نواحی تحت کنترل ایستگاه های مبنا (Base Station) مختلف حرکت کنند (Roaming)، ارتباط شبکه ای قطع خواهد شد. در شبکه های بی سیم گسترده مثل GSM نیز، در زمان Hand-Over چنین اتفاقی خواهد افتاد. از طرفی پهنای باند در GSM حداکثر Baud ۹۶۰۰ است. البته پروتکل هایی مثل GSRM و UMTS برای بهبود این شرایط در حال گسترش هستند. اگر از میان افزارهای موجود در شبکه های موبایل و بی سیم استفاده شود، مشکلات مختلفی پیش می آید، از جمله :

- مواجه شدن با عدم دسترسی به مؤلفه ها (شی ها) ی سرور یا Client ، به شکل حالت خاص و استثنایی افزایش خطا.
- پهنای باند شبکه های سیمی، Mbit/s ۱۰۰ ، اندازه مناسبی در شبکه های موبایل نخواهد بود.

از جمله راه حل های تحت بررسی برای فائق آمدن بر مشکلات فوق، ارائه عملیات اساسی هماهنگی (Coordination Primitives) است، مثل فضاهای جدولی (Table Space) که در آن عدم دسترسی مؤلفه ها به صورت شرایط نرمال در نظر گرفته می شود، نه شرایط استثنایی. هم چنین از نمایش دیتای انتقال به صورت فشرده شده، برای صرفه جویی در پهنای باند بهره گرفته می شود.

۲-۱۰-میان افزارها و تحقیقات مهندسی نرم افزار

به دو دلیل مطالعه اثرات میان افزارها در تحقیقات مهندسی نرم افزار، حائز اهمیت است :

اولاً : استفاده از مزایای محصولات میان افزار در سیستم های توزیع شده تجاری و صنعتی به سرعت در حال گسترش است.

ثانیاً : تهیه و ارائه کنندگان میان افزارها از نتایج فعالیت همدیگر، در محصولات خود بهره می جویند. به عنوان مثال، سرویس ISO/ODP Trader که در سال ۱۹۹۳ تعریف شده، در سال ۱۹۹۷ به عنوان استاندارد CORBA پذیرفته شد. لذا تحقیقات در حال انجام در زمینه میان افزارهای جدید، در چند سال آینده به صورت استانداردهای همگانی پذیرفته خواهند شد. مناسب است مسائل مورد بحث در زمینه مهندسی نرم افزار بر مبنای میان افزارها و استفاده از آنها در برپایی سیستم های توزیع شده باشد.

۲-۱۱-مهندسی نیازها (Requirements Engineering)

مسائلی از قبیل هماهنگی، قابلیت اطمینان، مقیاس پذیری و ناهمگونی در سیستم های توزیع شده، مهندسین نرم افزار را با ویژگی Non-Functional مشکلات روبرو می کند و نیاز به طراحی ساختارهای ویژه نرم افزاری جهت فائق آمدن بر آنها را ضروری می سازد. روش های موجود در مهندسی نرم افزار، نیازهای عملکردی (Functional) را مورد توجه قرار می دهند، مثل روش های شی گرا و Use-Case و هدف گرا (Goal-Oriented). البته روش اخیر از دو روش دیگر بیشتر به نیازهای غیر عملکردی می پردازد، ولیکن باید این جنبه بیشتر گسترش یابد. برای مفید بودن اهداف غیر عملکردی، در معماری های میان افزارگرا، بهتر است آن اهداف به صورت اندازه پذیر در آیند (Quantified). در این راستا وجود مدل های اندازه دار نیازها (Quantative Requirements Model) برای پارامترهایی مثل زمان پاسخ، حداکثر بار و حجم دیتا که قابل اندازه گیری و مقیاس پذیر باشند، لازم است.

مهندسی نیازها، برای یافتن روش ها و ابزار مناسب جهت رسیدن به چنین مدل هایی در حال گسترش است. انتخاب یک میان افزار برای معماری نرم افزار، مبتنی بر نیازهای غیر عملکردی انجام می گردد و تغییر مجدد آن هزینه های بسیاری در بردارد. از طرفی چنین نیازهایی نسبت به زمان متغیر هستند و اغلب با تغییر مشخصات (Setting) سیستم تغییر می کند. لذا مهندسی نیازها، علاوه بر در گرفتن نیازهای فعلی، باید توانایی برآورد تغییرات آتی سیستم های توزیع شده را داشته باشد.

۲-۱۲-معماری نرم افزار

مهندسی نرم افزار توزیع شده بایستی روش ها، نمادها و ابزار لازم برای رسیدن به طراحی و معماری چنین سیستم هایی بیافریند. ابزار جدید برای استفاده مهندسان نرم افزار جهت معماری سیستم هایی برای رفع نیازهای غیر عملکردی، مورد نیاز هستند. برای این امر

نیاز به مشخص کردن میان افزار یا ترکیبی از میان افزارهای مناسب وجود دارد. باید فرآیندهای معماری به گونه ای طراحی شوند که اثرات منفی ناشی از انتخاب میان افزار یا معماری غلط کاهش یابند. این پروسس ها، نیاز به روش هایی دارند که بتوانند مقیاس پذیری و کارایی را به صورت اندازه دار مدل سازی کرده و بتوانند از روشهای اعتبار سنجی مثل تست مدل، برای بررسی میزان پاسخ مدل به نیازها، استفاده کنند. تنظیم مدل با استفاده از اندازه های به دست آمده از مشاهده عملکرد کارایی میان افزارها، انجام می گردد.

۲-۱۳-نتیجه

استفاده از میان افزارها در طراحی سیستم ها، همیشه و به طور کامل برای طراح شفاف نیست. چند عامل نیاز به آگاهی طراح سیستم از استفاده از میان افزار در ارتباط بین مؤلفه ای را ضروری می سازد :

الف - کندتر بودن ارتباطات محلی نسبت به ارتباطات مؤلفه های توزیع شده در شبکه.

ب - فعال سازی و غیر فعال سازی مؤلفه های حالت دار (Statefull) ، پایداری و تداوم این مؤلفه ها را ضروی می سازد.

پ - برای مواجهه با ارتباطات متقابل همزمان در محیط توزیع شده، مؤلفه های خاصی مورد نیاز هستند.

ت - مؤلفه های می توانند عملیات اساسی (Primitive) همزمان سازی مختلفی را بر حسب میان افزار خاص، انتخاب کنند و از آنها به طور کامل بهره ببرند. این مؤلفه ها بایستی از مشکلات Deadlock اجتناب کنند.

جامعه مهندسی نرم افزار، بایستی ابزار، روش ها و نمادهای طراحی میان افزار گرا جهت ملحوظ داشتن شرایط فوق الذکر، طراحی و گسترش دهند.