کارشناسی ارشد مهندسی کامپیوتر

گرایش نرم افزار

موضوع :

کوربا (CORBA)

فهرست مطالب

موضوع **صفحه**

مقدمه..... ۱

۱-۱ فصل اول: تاریخچه و پیشینه کوربا..... ۲

۱-۲ تاریخچه کوربا..... ۳

۲-۱ فصل دوم: معماری کوربا..... ۵

۲-۲ اساسی ترین امکانات کوربا..... ۷

۲-۳ دلایل نیاز به کوربا..... ۷

۲-٤ کوربا و ساختار آن..... ۸

۲-۵ کارکرد کوربا..... ۱۰

۲-٦ مزایا و معایب کوربا..... ۱۲

۳-۱ فصل سوم : بررسی کوربا به عنوان زیرساختار ارتباطی بسته های نرم افزاری مرتبط با اسکادا

۳-۲ مقدمه..... ۱٤

۳-۳ معرفی OMG..... ۱٤

۳-٤ معرفی OMA..... ۱۵

۳-۵ معرفی Object Services..... ۱٦

۳-٦ مقایسه کوربا با تکنولوژیهای مشابه ۱۷
۳-٦-۱ RMIچیست ۱۷
۳-٦-۲ کوربا یا RMI ۱۹
۳-٦-۳ تکنولوژیهای DCOM و کوربا ۲۱
۳-۷ نتیجه گیری ۲۳

٤-۱ فصل چهارم: بررسی DCOM و کوربا در شبکه های client /server محاسبه شی توزیع شده

٤-۲ رابط احضار دینامیک ۲۵
٤-۳ رابط وگنجینه اطلاعات پیاده سازی: گنجینه اطلاعات رابط کوربا ۲۵
٤-٤ جعبه ابزار سرویس دهنده/سرویس گیرنده ۲٦
٤-٥ تکنولوژی COM/DCOM ۲۸
٤-٦ نتایج ۲۸

٥-۱ فصل پنجم: کوربا و بررسی امنیت در شبکه ۳۰
٥-۲ امنیت چیست؟ ۳۰
٥-۳ اهداف سیستم امنیتی کوربا ۳۰
٥-٤ سرویس امنیتی کوربا ۳۰
٥-٥ ویژگی های سرویس امنیتی کوربا ۳۱
٥-٦ کوربا و SSL ۳۲

٦-۱ فصل ششم: ارائه و ارزیابی یک مدل ترکیبی برای فراهم کردن توزیع بار و تحمل پذیری خطا در کوربا

٦-۲ مقدمه ۳٤
٦-۳ روشهای موجود برای تحمل خطا در کوربا ۳٤
٦-٤ استراتژی های توزیع بار در کوربا ۳٥
٦-٥ ارائه ی یک روش توزیع شده برای توزیع بار در کوربا ۳٥
٦-٦ معماری کلی سیستم ۳٦

٦-٧ بررسی مزایا و معایب و محدودیت های روش پیشنهادی..... ۳۷
٦-٨ پیاده سازی روش پیشنهادی..... ۳۸
٦-٩ شبیه سازی روش پیشنهادی..... ۳۹
٦-١٠ نتیجه گیری..... ٤١

منابع..... ٤٣

فهرست شکل ها

موضوع **صفحه**

شکل شماره ٢-١: درخواستهای ORB از سمت سرورهای مربوطه..... ٦
شکل شماره ٢-٢: سرویس توزیع یافته ORB..... ٩
شکل شماره ٢-٣: ساختار ساده کوربا..... ١٠
شکل شماره ٢-٤: درخواست از کلاینت بهobject implementation منتقل می‌شود..... ١١
شکل شماره ٢-٥: ارتباط ORB مکانیزم درخواست سرویس از راه دور..... ١٢
شکل شماره ٣-١: بخش های اصلی مدل مرجع OMG..... ١٥
شکل شماره ٣-٢: مکانیزم ارتباطی پایه کوربا..... ١٦
شکل شماره ٣-٣: RMI در مقایسه با کوربا..... ١٧
شکل شماره ٥-١: جریان عمومی درخواست در کوربا..... ٣١
شکل شماره ٢-٥: SSL به عنوان یک لایه امنیتی جداگانه بین IIOP و TCP/IP..... ٣٢
شکل شماره ٦-١: قرارگیری اجزا در معماری توزیع بار توزیع شده..... ٣٦
شکل شماره ٦-٢: معماری کل سیستم پیشنهادی..... ٣٧
شکل شماره ٦-٣: تعداد سرویس دهنده ١٠ ، تعداد سرویس گیرنده ٧ ، ٢ خطا در زمان ٣٠٠٠..... ٤٠
شکل شماره ٦-٤: گذردهی سیستم بدون بروز خطا برای تعداد سرویس دهنده ها و سرویس گیرنده های
مختلف..... ٤٠

شکل شماره ٦-٥: گذردهی سیستم با وجود خطا برای تعداد سرویس دهنده ها و سرویس گیرنده های مختلف(خطا در فاصله های زمانی ۸۰۰ ثانیه ای) ٤١
شکل شماره ٦-٦: مقایسه گذردهی، حالت های بدون خطا و خطادار برای تعداد ٦ سرویس دهنده و تعداد سرویس گیرنده های مختلف (خطا در فاصله های زمانی ۸۰۰ ثانیه ای) و مقایسه آنها با حالت بدون وجود توزیع بار ٤١

فهرست جداول

موضوع **صفحه**

جدول شماره ۲-۱: مقایسه‌ای سیستم‌های متمرکز و توزیع‌یافته ۸
جدول شماره ۳-۱: نقاط قوت و ضعف کوربا ۲۰

مقدمه

کوربا زبانی است که در کامپایلر زبانها قرار می گیرد و مزایای زیادی دارد از جمله باعث پایین آوردن هزینه های برنامه نویسی و همچنین ارتقاء کیفیت برنامه می شود. می توانیم توسط کوربا بسیاری از برنامه هایی که روی اینترنت وجود دارد را بهم وصل کنیم. هدف از این تحقیق شناخت متداولترین object based distributed middlware است. کوربا معروفترین و عمومی ترین زبان برای پیاده سازی اشیا است. ساختار زیر بنایی را تعریف می نماید که توسط آن زیر ساختار قادر به فراخوانی عملیات بر روی شی، بدون در نظر گرفتن محل قرار گیری آن بر روی شبکه است[٤] و [١].

این تحقیق شامل شش فصل می باشد که فصل اول تاریخچه و پیشینه کوربا را بیان کرده ، فصل دوم شامل معماری کوربا[١] ، اساسی ترین امکانات کوربا ، دلایل نیاز به کوربا ، ساختار و کارکرد کوربا ، مزایا و معایب کوربا شده است. در فصل سوم بررسی کوربا به عنوان زیرساختار ارتباطی بسته های نرم افزاری مرتبط با اسکادا و مقایسه آن با تکنولوژیهای مشابه DCOM[٢] و Java/RMI[٣] بیان گردیده ، فصل چهارم بررسی DCOM و کوربا در شبکه های client /server ، محاسبه شی توزیع شده را بیان می کند. در فصل پنج کوربا و بررسی امنیت در شبکه بیان شده و در نهایت در فصل آخر یک مدل ترکیبی برای فراهم کردن توزیع بار و تحمل پذیری خطا در کوربا ارائه و ارزیابی شده است.

---

(١) Common Object Request Broker Architecture

(٢) Distributed Component Object Model

(٣) Romote Methode invocation

۱-۱ فصل اول: تاریخچه و پیشینه کوربا

کوربا استانداردی برای نرم‌افزارهای ترکیبی[۱] است که توسط شرکت OMG[۲] طراحی شده و پشتیبانی می‌گردد. این استاندارد API[۳] ، پروتکل‌های رابطه‌ای و مدل‌های اطلاعاتی[٤] را تعریف می‌کند که می‌تواند نرم‌افزارهای ناهمگون (نوشته شده به زبان‌های متفاوت) را به هم مرتبط سازد. بنابراین با استفاده از کوربا می‌توانیم از شی[٥] در پلتفرم‌های توزیع‌یافته[٦] ، به صورت مشترک استفاده کنیم؛ بدون نگرانی از این که شی در چه موقعیت مکانی قرارگرفته یا به چه پلتفرمی متعلق است. کوربا می‌تواند کدهای نوشته شده (در برخی زبان‌های برنامه‌نویسی) را بسته بندی و به آن اطلاعاتی از قبیل توانایی اجرایی کدها و چگونگی اجرای آن‌ها را اضافه نماید. به طوری که این کدهای بسته بندی شده (یا شی‌ها) بتوانند از برنامه‌های دیگر (یا حتی شی‌های کوربا) که تحت شبکه قرار دارند، اجرا شود. کوربا برای تعیین‌کردن اینترفیس‌هایی که به دیگران ارائه می‌کند، از یک زبان رابطه‌ای به نام IDL[۷] استفاده می‌کند و از طریق این زبان می‌تواند کدهای اجرایی (مثلا جاوا یا C++) را بشناسد. امروزه می‌توان زبان‌های معروفی مثل جاوا ، Smalltalk ، C++ ، Python ، آدا ، سی و لیسپ را با استفاده از این زبان (IDL) به هم مرتبط ساخت و از توانایی‌های آن‌ها استفاده نمود. قسمت اول این تحقیق در آغاز نرم افزارهای توزیع یافته را معرفی کرده و دلایل نیاز به آن را مشخص می‌نماید. سپس این نرم‌افزارها را با سیستم‌های متمرکز یا توزیع یافته مقایسه می‌کند و در ادامه با طرح این سؤال که چرا به کوربا نیاز داریم، کوربا را معرفی و ساختار آن را تشریح می‌کند و بعد چگونگی کار این استاندارد را توضیح می‌دهد. در پایان نیز مزایا و ضعف های آن را به اختصار بیان می‌نماید[٤].

کوربا، پروتوکلی است که براساس آن امکان ایجاد ارتباط میان اشیاء طراحی و تولید شده در زبانهای برنامه نویسی و پلتفرمهای مختلف فراهم می گردد. زمانی که در یک سازمان برقراری ارتباط بین عناصر نرم افزاری روی پلتفرمهای مختلف ضرورت یابد ، کوربا می تواند یکی از گزینه های قابل اجرا باشد. اشیاء سازگار با کوربا بطور مستقیم ارتباطی با یکدیگر برقرار نمی سازند بلکه درخواستهای یک شیء توسط سرویسی به نام ORB[۸] به شیء دیگر منتقل می شود[۲].

---

(۱) componentry

(۲) object management group

(٤) object/service

(٥) object

(٦) distributed

(۷) Interface Definition Language

(۸) Object Request Broker

۱-۲ تاریخچه کوربا

درسال ۱۹۸۹ ، OMG با ۸ عضو که مشتمل بر بیش از ۷٦۰ سازمان و بخش بود به منظور "فراهم سازی یک قالب کاری و ساختاری برای نرم افزارهای شیءگرا که بطور گسترده استفاده می شد" تشکیل شد.

اولین نسخه کوربا در سال ۱۹۹۰ معرفی شد که تنها شامل یک سری مشخصات و ویژگیها بود و به تولید کنندگان نرم افزار اجازه می داد تا این مشخصات را با هر روشی پیاده سازی کنند. چندی بعد نسخه corba۱,۱ به بازار عرضه شد که مهمترین ویژگی آن تعریف زبانی با نامIDL[۱] بود که بعنوان API در این نرم افزارها برای ارتباط باORB[۲] تعبیه شد. مهمترین محدودیت نسخه های CORBA X.۱ عدم وجود پروتکلی استاندارد برای ارتباط با ORB های مختلف تحت شبکه وجود بود که باعث می شد ORB یک شرکت نتواند با ORB شرکت دیگر ارتباط برقرار کند که این نقیصه نیز با ارائه نسخه ۲ CORBA که در دسامبر ۱۹۹٤ ایجاد شد با تعریف پروتکلی بنام IIOP[۳] رفع گردید. با استفاده از این پروتکل دیگر نرم افزارهایی که در آن ها از تکنولوژی کوربا استفاده می شد مستقل از سخت افزار و مستقل از شرکت تولید کننده نرم افزار بود. در سال ۲۰۰۰ میلادی نسخه ۳ CORBA تولید شد که کاملترین نسخه کوربا می باشد[٤].

در سال ۱۹۹۱ OMG[٤] ، نرم‌افزار CORBA۱,۱ را معرفی کرد که IDL و APIها را معین می‌کرد و به صورتی که اثر متقابل از یک شی ء به نام Client/server را همراه یک عملکرد معینی از یک ORB فعال می‌ساخت. CORBA۲,۰ که در دسامبر ۱۹۹٤ تنظیم شد یک گام بزرگی را پیش برد و به شکلی واقعی قابلیت استفاده مفید بین ORBها را از شرکت‌های متفاوتی از طریق IIOP ، که IP/TCP mapping از GIOP[٥] می‌باشد، تعیین کرد. CORBA۲,۰ متعهد این قضیه است که همه ORBهای آماده ، IIOP را برای اطمینان که اثرپذیری متقابل پشتیبانی می‌کند[٤] و [۸].

از آن پس کوربا به آرامی نسخه‌های خود را ارائه کرد (نسخه کنونی ۲,٦,۱ است) ولی در این سیر تکاملی هر مرحله شاهد افزایش ویژگیهای استواری است که به سمت ساخت سیستم‌های ماهرانه[۱] جهت‌دار شده است. شرکت Sun Microsystem در سال ۱۹۹۸ پشتیبانی از کوربا را همراه نسخه جایگزین شده JDK۱,۲ و موارد اضافه شده Java IDL و یک ORB ساده شروع کرد و این امر آغاز یک رابطه دور و

(۱) Interface Definition Language

(۲) Object Request Broker

(۳) Internet – Inter – ORB- Protocol

(٤) object management group

(٥) General Inter-ORB Protocol

دراز بین Java و کوربا شد. امروزه بعد از گذشت ٤ سال J۲SE ۱,٤ این ازدواج را با پشتیبانی از CORBA۲,۳,۱ می‌دهد. (تازه‌ترین نسخه کوربا شماره ۲,٦,۱ می‌باشد؛ تنها تفاوت مهم با نسخه ۲,۳,۱ وجود CORBA messaging است که در نسخه ۲,٤ اضافه شد)[۲] .

۲-۱ فصل دوم: معماری کوربا

اساسی ترین جزء کوربا ، کارگزار درخواست شیء (ORB) می باشد که وظیفه آن تسهیل ارتباط بین اشیاء می باشد.

ORB یک جز از کوربا است که به عنوان میان افزار بین client و Server عمل میکند. با داشتن مرجع و رفرنس اشیاء دارای قابلیت همکاری (IOR)، ORB قادر خواهد بود که اشیاء مورد نظر را مکان یابی نموده و عملیات انتقال داده ها را از طریق فراخوانی های متدها از راه دور انجام دهد.

واسط یک شیء کوربا بوسیله زبان تعریف واسط کوربا که به اختصار IDL نامیده می شود، تعریف و مشخص می شود. یک کامپایلر IDL تعاریف IDL را به یک زبان برنامه نویسی کاربردی از قبیل C++ , Java یا Tcl/Tk ترجمه می کند که موجب ایجاد ریشه ها[۲] و ساختارهای[۳]IDL شده که به ترتیب چارچوب[٤] کد نماینده[٥] سمت client و سمت سرور را فراهم می سازد. عملیات کامپایل برنامه های کاربردی در برگیرنده ریشه های IDL موجب ایجاد یک واسط فراخوانی ثابت فوق العاده کلیشه ای[٦] می گردد[٤].

درمقابل ، DII و DSI به اشیاء اجازه ایجاد شدن قبل از هرگونه دانشی در خصوص واسط IDL را می دهند. درخواستها و پاسخهای بین اشیاء در یک قالب استاندارد که توسط پروتوکل IIOP تعریف می گردد، انتقال و به مقاصد مورد نظر تحویل می گردد. درخواستها بوسیله ریشه کلاینت (یا درون DII) در قالب یک پلتفرم منظم شده و در سمت سرور آنمارشالینگ می شود. مارشالینگ پروسه ای است که پارامترهای ورودی را به شکلی که قابل انتقال توسط شبکه باشد تبدیل می کند.

آنمارشالینگ: عمل معکوس مارشالینگ است و داده های آمده از شبکه را به شکل پارامترهای ورودی تبدیل می کند.برای اینکه ORB بتواند داده ها را در شبکه رد و بدل کند آنها را به فرمت خاصی تبدیل می کند که این فرمت را one-the-wire می گویند[۳].

(۱) enterprise

(۲) stubs
(۳) skeletons
(٤) FrameWork
(٥) Proxy Code
(٦) strongly-typed

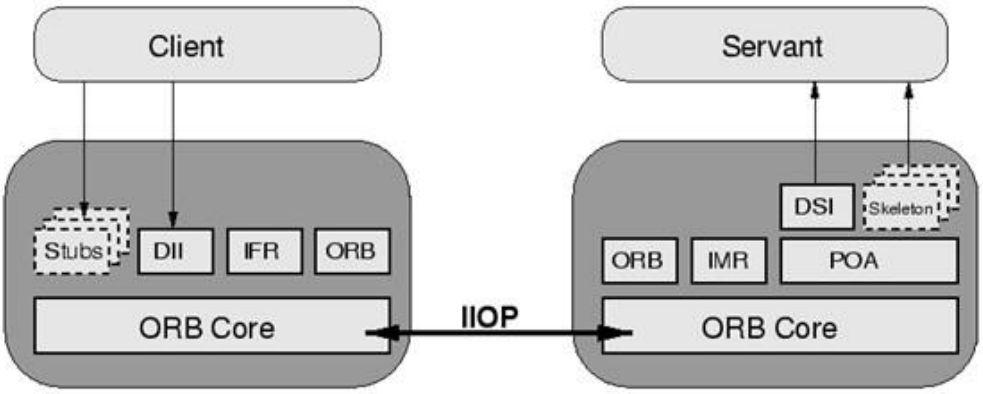
نتیجه مارشالینگ و آنمارشالینگ استقلال سخت افزاری است. بدلیل اینکه پارامترها هنگام انتقال از شبکه به یک شکل مستقل از سخت افزار تبدیل می شود و هنگام دریافت به یک شکل وابسته به سخت افزار تبدیل می گردد. در نتیجه ارتباط بین اجزاء یک رابطه مستقل از سخت افزار است. وفق دهنده سیار اشیاء کوربا را با مجموعه ای از متدهای عمومی مشخص به منظور دسترسی توابع ORB در محدوده تصدیق کاربر به منظور فعال سازی شیء تا ماندگاری و بقاء شیء را فراهم می سازد[۱].

این کار در واقع اساسی ترین وظیفه آن می باشد علاوه بر آنکه کار ایجاد رفرنس های اشیاء و مخابره درخواستهای ORB از سمت سرورهای مربوطه به سمت اشیاء هدف را نیز بر عهده دارد.

ویژگیهای POA در زمان ایجاد آن بوسیله مجموعه ای از سیاستهای POA تعریف می گردد. هرسرور می تواند میزبان هرتعداد از POAها باشد که هرکدام دارای مجموعه سیاستهایی برای کنترل و اداره نمودن درخواستهای پردازشی می باشند.

یکی از خصوصیات پیشرفته تر POA ، مدیریت servantها می باشد که وظیفه فعالسازی مجدد اشیاء سرور یا همان servantها در زمان نیاز به آنها را برعهده دارد[٤].

POA همچنین مکانیسمی را برای ذخیره و بازیابی یک شیء دارد که با استفاده از انباره اجرا (IMR) انجام شده و با مدیریت عملیات خودکار start و restart سرورها متوجه حضور اشیاء می گردد.



شکل شماره ۲-۱: درخواستهای ORB از سمت سرورهای مربوطه[٤]

کوربا هیچگونه ضمانتی را درباره چگونگی انجام این درخواستهای on-way توسط ORBها متعهد نمی‌شود. بطور مثال، یک ORB که هیچگونه درخواست on-way را تحویل نمی‌دهد، می‌خواهد مورد قبول کوربا هم باشد (عجب توقعی!) بالاخره CORBA۲,٤ نیز یک سبک پیام‌رسانی[۱] خبره‌ای را ارائه

(۱)asynchronous

می‌دهد که از آن جایی که جزیی از J۲SE ۱,٤ نمی‌باشد به خارج از محدوده این سری مقالات کشیده می‌شود. (همان‌طور که به یاد دارید CORBA۲,۳,۱ را پشتیبانی می‌کند). نکته دیگر اینکه من به شرح و تفصیل در مورد پروتکل‌های ارتباطی درگیر با روش نیایش[۲] و تشکیلات ORB پرداختم. برای تصور مطلب، ORB از ۲ بخش اصلی تشکیل شده است: یک زمان اجرا client-side و یک زمان اجرا server-side. Stubها بخشی از زمان اجرای client-side بوده و skeletonها بخشی از زمان اجرای server-side می‌باشند. در شکل تصوری قضیه، وقتی که فقط یک ORB درگیر می‌شود، همان‌طور که در مثالمان تاکنون مطرح شد، پروتکل ارتباطی استفاده شده بین زمان اجرای client-side و server-side، ممکن است IIOP یا هر پروتکل دیگری که مورد پسند vendor است باشد. در چنین مواقعی بسیاری از ORBهای تجاری یک پروتکل بسیار کارآمد و بهینه شده‌ای را مورد استفاده قرار می‌دهند.

ولی به خاطر داشته باشید: تمام ORBهایی که با نسخه ۲ یا بالاتر کوربا کار می‌کنند باید ارتباطی که IIOP را استفاده می‌کند، به خوبی پشتیبانی کنند. تا اثرپذیری متقابل با ORBها را مطمئن سازد[۲].

۲-۲ اساسی ترین امکانات کوربا :

الف)زبان تعریف واسط OMG[۱]

ب) فراخوانی عملگرها و استفاده از تسهیلات به صورت ایستا و پویا

ج)رابط های شی و پروتکلهای درون ORB[۲] [٤].

۲-۳ دلایل نیاز به کوربا

دلایل نیاز به کوربا را می‌توان در پنج‌مورد زیر فهرست کرد :

الف- گوناگونی سیستم عامل‌ها

ب- تنوع پلتفرم‌های سخت افزاری

ج- گوناگونی پروتکل‌های شبکه (برای مثال TCP/IP ، ATM و Ethernet)

د-گوناگونی زبان‌های برنامه‌نویسی (C ,C++ ,Java ,COBOL ,Basic ,Perl ,Smalltalk)

ه-نیاز ارتباط یافتن سیستم‌های جدید با سیستم‌های قدیمی [۳].

(۲) invocation

سیستم های توزیع‌یافته	سیستم های متمرکز	ویژگی
کم	زیاد	امنیت
در سیستم‌های توزیع‌یافته که object ها درسیستم‌های مختلفی قرار دارند، به امنیت بیشتری نسبت به سیستم های محلی نیاز است.		
شی‌ها جداگانه خراب می‌شوند	همه شی ها با هم خراب می‌شوند	مشکلات و ازکارافتادگی شی‌ها
اگر پردازنده‌ای که دو object را اجرا می‌کند دچار مشکل شود، هر دو object دچار مشکل خواهند شد.		
دارد	فقط با استفاده از با چندThread	دسترسی همزمان
برنامه‌ها بر روی سیستم‌های متمرکز، به‌صورت پیش فرض از یک thread استفاده می‌کنند. ولی در سیستم‌های توزیع‌یافته به‌صورت پیش فرض از thread استفاده می‌شود.		
آهسته	سریع	ارتباط بین سیستم ها
از آن‌جایی که ارتباط بین object ها دریک پردازنده سریع تر از ارتباط بین object ها درچند پردازنده می باشد، پیشنهاد می‌شود اگر ارتباط تنگاتنگی بین شی‌ها وجود دارد، از سیستم‌های متمرکز استفاده شود.		

جدول شماره ۲-۱: مقایسه‌ای سیستم‌های متمرکز و توزیع‌یافته[۱]

با استفاده از کوربا، دیگر دلیلی برای نگرانی در مورد این‌که از چه زبان برنامه‌نویسی ، سیستم عامل یا پلتفرمی استفاده می‌کنید ، وجود ندارد. از آن‌جایی که کوربا با مکانیزم ساده‌ای می‌تواند سیستم‌های مختلف از هر نوع و هر اندازه را به هم متصل کند، در خیلی از موقعیت‌ها بسیار مفید به نظر می‌رسد. یکی از مصارف کوربا در سرورهایی است که با تعداد زیادی کلاینت و ترافیک سنگین اطلاعاتی باید به خوبی کارکنند. استفاده از کوربا فقط محدود به نرم افزارهای بزرگ نمی‌شود و حتی درسیستم‌های بلا درنگ نیز کاربرد دارد[۹].

۲-٤ کوربا و ساختار آن

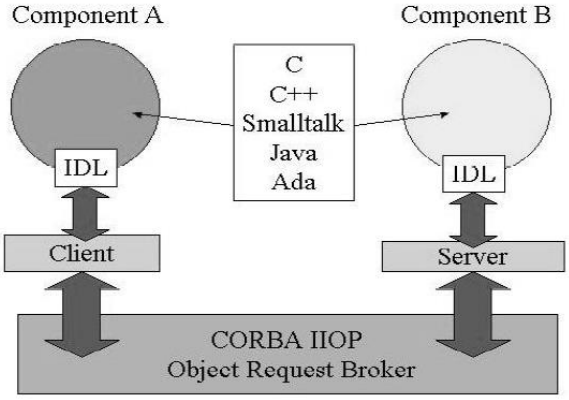
گروه OMG در سال ۱۹۸۹ با هدف ایجاد و پشتیبانی استاندارد برنامه‌های شی‌گرای غیر متمرکز به وجود آمد. البته، این گروه در واقع تولید کننده هیچ نرم‌افزاری نیست، بلکه مشخصات این برنامه‌ها را با استفاده از فناوری و نظرات اعضا مشخص می‌کند[۷].

این گروه شامل بیش از ۷۰۰ شرکت و سازمان تولیدکننده فناوری سیستم‌های توزیع‌یافته ، بانک‌های اطلاعاتی ، پلتفرم‌ها و سازندگان بزرگ نرم‌افزار می‌باشد. تلاش اصلی این گروه در راستای تعریف امکانات و ساختار لازم برای سیستم های شی‌گرای غیر متمرکز است و یکی ازمکانیزم‌های اصلی در این رابطه که هسته مرکزی این گروه نیز می‌باشد ORB[۱] نام دارد.

کوربا ساختار استانداردی برای سیستم‌های توزیع‌یافته شی‌گراست که می‌تواند سیستم‌های ناهمگون و پراکنده را به هم مرتبط سازد. همان‌طور که گفته شد، کوربا ساختاری برای شی ء های توزیع یافته مشخص می‌کند. محور اصلی این ساختار بر اساس درخواست سرویس از شی‌های توزیع یافته می‌باشد. این شی‌ها سرویس‌ها را از طریق اینترفیس‌هایشان که به زبان IDL[۲] مشخص شده‌اند ، ارائه می‌دهند Object های غیر متمرکز را می توان از object references ها که توسط اینترفیس‌های IDL تایپ شده اند، شناسایی کرد .

همان‌طورکه در شکل می‌بینید، درخواست کننده سرویس (کلاینت) شماره مرجع object شی سرویس‌دهنده را از طریق اینترفیس آن در اختیاردارد (Interface A) و ORB درخواست کلاینت را به شی سرویس‌دهنده و جواب درخواست را به شی درخواست کننده می رساند.

ORB در واقع یک سرویس توزیع‌یافته است که به درخواست شی‌های دوردست[۳] رسیدگی می‌کند. بدین ترتیب که این شی‌ها را در شبکه قرار می دهد، با شی‌ها ارتباط برقرار کرده، و درخواست سرویس را مطرح می‌کند. سپس برای جواب سرویس درخواستی صبر می‌کند و سرویس مورد نظر را به کلاینت درخواست کننده منتقل می‌کند. ORB این کار را بدون توجه به موقعیت مکانی سرویس‌دهنده و زبان برنامه درخواست‌کننده انجام می‌دهد. نیازی نیست که کلاینت درخواست کننده به زبان کوربا تقاضای خود را مطرح کند ORB زبان برنامه درخواست کننده را شناسایی کرده (برای اکثر زبان‌های برنامه نویسی) و این زبان را ترجمه می کند[٤].



شکل شماره ۲-۲: سرویس توزیع یافته ORB[٦]

(۱) Object Request Broker

(۲) Interface Definition Language

(۳) Instances

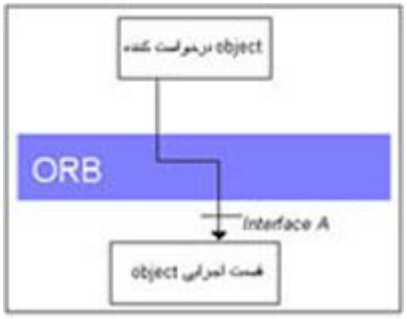
۲-۵ کارکرد کوربا

نرم‌افزارهایی که از کوربا استفاده می کنند، حاوی شی‌هایی‌هستند. البته معمولا از یک شی چندین نمونه[۱] از یک نوع وجود دارد. مثلا در یک سایت تجارت الکترونیک، نمونه سبدهای خرید زیادی وجود دارد که با وجودی که همه آن‌ها کاری یکسان انجام می‌دهند، هر کدام به مشتریان متفاوتی تعلق دارد و اطلاعات خرید آن مشتری را ذخیره می کنند[٤].

برای هر نوع از شی‌ها ، مثلا سبد خرید می‌باید یک اینترفیس در OMG IDL تعریف کنیم که نشانگر سرویس‌هایی است که شی سرور در اختیار کلاینت‌های تقاضاکننده قرار می‌دهد. هر کلاینت که تقاضای اجرای عملیاتی از شی را دارد ، باید از این اینترفیس‌ها استفاده و آرگومان ها را کنار هم چیده و ارسال نماید. وقتی که این تقاضا به شی سرویس‌دهنده می‌رسد ، این شی با استفاده ازاینترفیس مشابه آن ، عملیات درخواستی کلاینت را انجام می دهد و آرگومان ها را جدا سازی می کند. پس از انجام عملیات درخواستی، جواب درخواست ، بسته بندی شده و با استفاده از تعریف اینترفیس از همان مسیری که آمده بود ، بر می گردد[۱۰].

اینترفیس IDL برای اکثر زبان‌های برنامه‌نویسی از جمله Python , IDLscript Lisp , Ada , COBOL , Smalltalk , java ++C , C و از طریق استاندارد OMG تعریف شده است. جداسازی اینترفیس و قسمت اجرایی که توسط OMG IDL مهیا شده است، یکی از مهم‌ترین پایه‌های اصولی کوربا می باشد و ایجاد اینترفیس برای هر شی در کوربا ضروری است. ازطرف دیگر قسمت اجرایی هر شی از دید سیستم مخفی می‌ماند و کلاینت از آن چیزی نمی‌داند. در نتیجه کلاینت‌ها فقط می‌توانند از طریق

اینترفیس‌ها به سرویس‌ها دسترسی داشته و تنها سرویسی را اجرا کنند که توسط اینترفیس‌های IDL معرفی شده اند[۱۱].

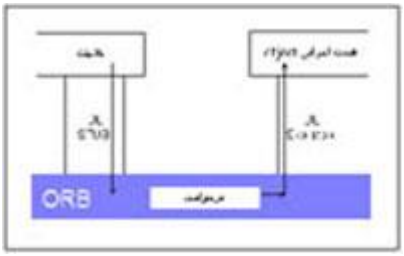


شکل شماره ۲-۳: ساختار ساده کوربا[۱۰]

(۱)unique

همان‌طور که در شکل ۲-۳ نشان داده شده‌است، کلاینت توسط اینترفیس IDL تقاضای سرویس می‌کند. برای این کار IDL باید به کلاینت stubs و object skeletons کامپایل شود stubs و skeletons ها در واقع نقش نمایندگان کلاینت و سرور را بازی می‌کنند و از آن جایی که IDL تعریف مشخصی برای اینترفیس ها دارد، اگر حتی این دو نماینده به دو زبان متفاوت یا حتی بر روی دو ORB از دو شرکت مختلف اجرا شوند، می‌توانند بدون مشکل به هم مرتبط شوند .

همان‌طور که اشاره شد، در کوربا هر نمونه ازاشیا دارای یک کد یکتا[1] به نام object reference هستند. کلاینت‌ها از این اشیا برای هدایت درخواست‌هایشان و شناساندن خود به OR ها استفاده می‌کنند. اگرچه کلاینت‌ها درخواست‌هایشان را به نماینده سرور یا IDL stub منتقل می‌کنند نه به خود سرور، چنین به نظر می‌رسد که کلاینت مستقیما به سرور دسترسی دارد. این درخواست سپس از طریق ORB و skeleton به قسمت اجرایی می‌رسد و این قسمت به این درخواست رسیدگی می‌کند. شکل ۲-٤ مکانیزم درخواست سرویس ازراه دور را نشان می دهد[۲].

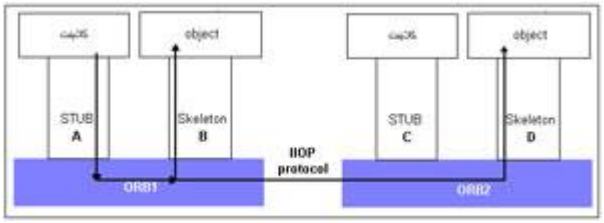


شکل شماره ۲-٤: درخواست از کلاینت به object implementation منتقل می‌شود[٤]

برای ایجاد ارتباط و درخواست از شی دوردست، در اولین قدم ، کلاینت object reference شی موردنظر را از راه‌های گوناگونی مثل Naming Service به دست می آورد و با استفاده از مکانیزم درخواست محلی ، درخواست خود را مطرح می کند. وقتی که ORB متوجه می‌شود object reference موجود، مخصوص یک شی دوردست است ، مسیرخود را عوض کرده و از طریق شبکه پروتکل (IIOP) به دنبال ORB شی دوردست می‌گردد و درخواست خود را مطرح می کند[۹].

چگونگی این مکانیزم از این قرار است که گروه OMG استانداردی دو مرحله‌ای برای این رویه دارد. در مرحله اول کلاینت از نوع شی‌ای که درخواست می کند، اطلاع دارد (مثلا سبد خرید) و stub کلاینت (A) و skeleton شی توسط IDL یکسان تولید شده است. این بدین معنی است که کلاینت دقیقا می‌داند که چه عملیاتی (و با چه پارامترهای ورودی) می‌تواند درخواست کند و چه زمانی درخواست به شی مقصد می‌رسد. در مرحله دوم باید ORB های کلاینت و شی مورد نظر، هر دو از یک پروتکل یکسان تبعیت کنند[۸].

این پروتکل حاوی تمامی اطلاعات مورد نیاز (ازجمله تمامی پارامترهای ورودی/خروجی) برای دسترسی به شی مقصد می‌باشد. اگر چه ORB ها می‌توانند شی‌های محلی را از دوردست تشخیص دهند، از آنجایی که کلاینت حاوی هیچ objec treference ای در موقع درخواست نیست، قادر به تمایز شی‌های دوردست و محلی نمی‌باشد و همان‌طور که قبلا ذکر شد، این از اصول اصلی کوربا می‌باشد (بدون اهمیت بودن موقعیت مکانی شیء)[٤] .



شکل شماره ۲-۵: ارتباط ORB مکانیزم درخواست سرویس از راه دور[٦]

۲-٦ مزایا و معایب کوربا

با توجه به آنچه گذشت، اکنون‌به‌نظر می‌رسد بحث درباره مزایا و معایب این استاندارد، ضروری باشد. از این‌رو به پاره‌ای از مزایا و معایب کوربا اشاره می‌شود:

-کلاینت نیازی به دانستن موقعیت مکانی شی ندارد. یک شی می‌تواند هر جا باشد، به کلاینت متصل باشد یا بر روی سروری در آن سوی کره زمین، هیچ فرقی نمی‌کند.

-موقعیت مکانی شی می‌تواند بدون بروز اشکال در نرم افزار تغییر یابد .

-کلاینت نیازی به دانستن این که سروری برای جواب دادن به درخواست وجود دارد یا نه ، ندارد .

-کلاینت و سرور می‌توانند به دو زبان کاملا متفاوت نوشته شوند و این مزیت مهم کوربا اجازه استفاده از قدرت زبان های مختلف را از طریق (IDL) برای تهیه نرم افزارهای بزرگ می‌دهد.

-کلاینت نمی‌داند یک شی چگونه عمل می‌کند. به همین خاطر یک سرور می‌تواند بدون این که کلاینت اطلاع پیدا کند یک بار از فایل ساده و بار دیگر از پایگاه اطلاعاتی شی‌گرا برای ذخیره اطلاعات استفاده نماید .

-سرعت و کارایی سیستم‌هایی که از کوربا استفاده می کنند بسیار بالا است.

-سیستم‌های عامل و پروتکل سرور و کلاینت می تواند متفاوت باشد .

اگرچه همان‌طورکه می‌بینید استفاده از کوربا می‌تواند مزایای زیادی داشته باشد، اما دارای نقاط ضعف زیر نیز هست :

-از انتقال و جابه‌جایی شی (object) ها پشتیبانی نمی کند.

-کوربا به IDLهایی نیاز دارد که هنوز برای برخی از زبان‌ها تعریف نشده است و وقت زیادی برای یادگیری آن ها نیاز است

-اگر کوربا نتواند نیازهای صنایع امروز را برآورده سازد و رواج نیابد، آینده مبهمی در انتظار خواهد داشت و ممکن است به جمع سیستم‌های قدیمی بپیوندد.

-از آنجایی که خصوصیات کوربا هر چند وقت یک بار عوض می‌شود ، نیاز به آموزش و بروزرسانی بیشتری دارد.

-از آن جایی که شعار کوربا ، سرعت و کارایی بیشتر نسبت به سیستم‌های توزیع‌یافته جاوا مثل RMI است و همه نرم افزارهای کاربردی نیازی به سرعت بالای آن ندارند ، شاید بتوان گفت اگر بخواهیم فقط با جاوا کار کنیم آسانی و کاربرد سیستم‌هایی مثل RMI از کوربا بیشتر است[٤].

۳-۱ فصل سوم : بررسی کوربا به عنوان زیرساختار ارتباطی بسته های نرم افزاری مرتبط با اسکادا

۳-۲ مقدمه

کوربا هسته مرکزی مدل مرجع OMA که معماری استانداردی برای سیستم های نرم افزاری مرتبط به هم تعریف می کند، را تشکیل می دهد. معماری OMA جهت استاندارد کردن مدل ارتباطی نرم افزارهای گسترده که تبادل اطلاعات بین بسته های نرم افزاری از مسائل اصلی آن می باشد، توسط سازمان OMG مطرح شده است. مشخصات استاندارد کوربا بستری را فراهم می کند که این بسته ها بتوانند در یک محیط ناهمگون شامل سخت افزارها، سیستم عامل ها، و زبان های پیاده سازی مختلف از سرویس های یکدیگر بصورت ساختار یافته ای استفاده نمایند. در سیستمهای نرم افزاری کنترل پروسه، بالاخص سیستم های کنترل شبکه قدرت، استاندارد شدن رویه تبادل اطلاعات بین بسته های نرم افزاری همیشه مورد بحث واقع شده است. در این سیستمها معمولاً اطلاعات کل سیستم به صورت متمرکز در ماشین های سرور اسکادا جمع آوری شده و در اختیار بسته های نرم افزاری نمایش اطلاعات نرم افزارهای مدیریت و تحلیل ٥DMS/٤EMS و یا GIS٦ گذاشته می شوند.

در این تحقیق ابتدا سازمان OMG معماری OMA و سرویس های آن را معرفی می کنیم. سپس استاندارد CORBA که زیرساخت مرکزی این مدل می باشد را با تکنولوژیهای مشابه DCOM و Java RMI جهت استفاده در نرم افزار اسکادا مقایسه می کنیم[۳].

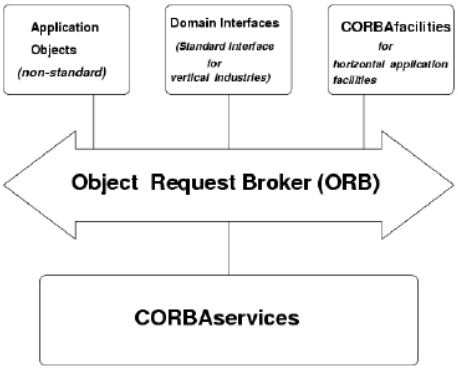
۳-۳ معرفی OMG

OMG یک سازمان بین المللی است که توسط بیش از شرکت مهم و فعال در زمینه تکنولوژی اطلاعات شامل طراحان، توسعه دهندگان نرم افزار پشتیبانی می شود. این سازمان در سال ۱۹۸۹ با هدف ارتقاء تکنولوژی شیءگرا در تولید نرم افزار تاسیس شده است[۳].

منشور کاری سازمان OMG تولید استانداردها و رهنمونهای کاربردی برای صنعت اطلاعات در جهت ایجاد یک قالب مشترک برای توسعه نرم افزارها شیءگرا تعریف شده است .اهداف پایه این استانداردها ایجاد قابلیتهای استفاده مجدد ، حمل پذیری ، سازگاری نرم افزارهای شیءگرا در محیط های گسترده و ناهمگون می باشد. پیروی از این استانداردها این امکان را فراهم می آورد با هم کار کنند OMG به سمت استقرار ساختار" معماری بر مبنای مدل MDA از طریق تهیه برآورد که بتواند نرم افزارهایی تولید کند که در محیط های ناهمگون سخت افزارها و سیستم عامل های مختلف با مشخصات پایه شامل کوربا ، MOF، XMI UML،CORBA/IIOP، Internet Facilities، Domain ، Interface پیش می رود. این استانداردها توسط بسیاری از شرکتهای مهم پیاده سازی و در صنعت اطلاعات در حال استفاده است[۳].

۳-٤ معرفی OMA

OMG مدل مرجع را به منظور مشخص نمودن معماری سطح بالا قطعات شی ءگرا معرفی کرده است .مدل مرجع ویژگیهای کلی بسته ها، اینترفیس ها و پروتکل هایی را در ساختار OMA مشخص می کند اما به تعریف جزئیات نمی پردازد. مدل مرجع OMA که در شکل زیر به نمایش درآمده است دارای پنج بخش اصلی است[۳].



شکل شماره ۳-۱: بخش های اصلی مدل مرجع OMG[٦]

الف-در بخش مرکزی مدل که قلب ارتباطی آن نیز می باشد دلال درخواس تهای اشیاء یا ORB قرار گرفته است. این بخش Client ها را قادر می سازد که با اشیاء در یک محیط گسترده ارتباط برقرار کنند OMG

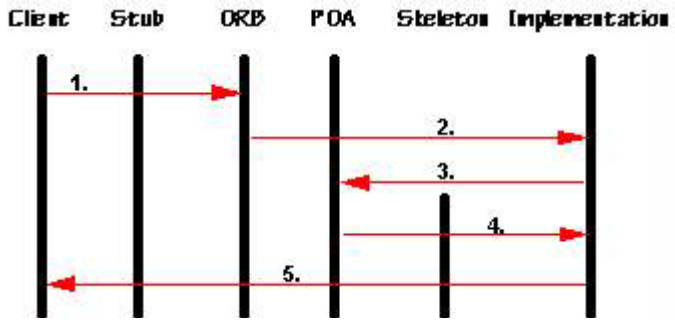
در این بخش مشخصات کوربا را تعریف کرده است که قابلیت فراخوانی عملیات روی اشیاء را در محیط شبکه و در یک محیط ناهمگون شامل سیستم عامل ها یا زبانهای برنامه نویسی مختلف ممکن می سازد.

ب-Object Services مجموعه ای از سرویسها و اینترفیس ها را شامل می شود که عملکردهای پایه را برای تحقق مدیریت اشیاء تعریف می کند. سرویسهای کوربا محیطی را فراهم میکنند که اشیاء به صورت مستقل بتوانند وظایفشان را انجام داده و همچنان رابطه آنها تعریف شده با سطح دسترسی قابل کنترل باقی بماند.

ج-CORBA Facilities شامل مجموع های از کلاسها و اشیاء هستند که برای کاربردهای عام در شاخه های مختلف صنعت نرم افزار می توانند مورد استفاده قرار بگیرد. این بخش که به آن تسهیلات افقی هم گفته می شود، فعلاً محدود به چهار بخش تسهیلات چاپ کردن ، زمان ، بین المللی سازی و واسط موبایل می باشد.

د-Domain Interface که به آن تسهیلات عمودی هم گفته می شود مجموعه ای از اینترفیس های استاندارد برای مصارف خاص در دنیای نرم افزار می باشد. از جمله کاربردهای آن می توان به کاربردهای سیستمهای مالی ، تولید صنعتی و بهداشتی/پزشکی اشاره نمود.

ه-Application Object به بخشی در این معماری گفته می شود که اختصاص به برنامه های کاربردی کاربران دارد. این برنامه ها چه استاندارد شده یا نشده باشند از طریق ORB می توانند به سرویسهای تعریف شده در بخشهای قبلی یا حتی با یکدیگر ارتباط برقرار کنند و عملیات اشیاء آنها را درخواست و اجرا نمایند[۳] و [۷].



شکل شماره ۳-۲: مکانیزم ارتباطی پایه کوربا[۳]

۳-۵ معرفی Object Services

همانطور که قبلاً توضیح داده شد Object Services بخشی از معماری OMA می باشد. این سرویس ها عملکرد پایه ای برای مدلسازی منطقی و انباره فیزیکی اشیاء فراهم می سازند.

عملکردهای Object Services از طریق ORB برای بقیه سیستم قابل دسترسی می باشد .با توجه به آزاد بودن این استاندارد ها، در پیاده سازی می توان اینترفیس های خاصی را علاوه بر مشخصات تعیین شده OMG پشتیبانی نمود و عملکردهای جدیدی را ارائه نمود.

عملکردهایی عمومی که Object Services می توانند پشتیبانی کنند از قرار زیر می باشد:

- **مدیریت کلاسها :** قابلیت ایجاد ، تغییر ، پاک کردن ، کپی کردن و کنترل تعاریف کلاسها و اینترفیس به کلاسها و همچنین روابط بین آنها.

- **مدیریت نمونه ها :** قابلیت ایجاد ، تغییر ، پاک کردن ، کپی کردن ، منتقل کردن ، فراخوانی و کنترل اشیاء و رابطه بین آنها.

- **انباره کردن :** قابلیت انباره کردن به صورت دائمی یا گذرا و برای اشیاء کوچک یا بزرگ شامل حالت آنها و متدهای آنها.

-**همخوانی :** قابلیت ایجاد همخوانی و سازگاری حالت اشیاء هم به صورت تکی (مثلاً از طریق قفل های نرم افزاری) و هم به صورت جمعی(مثلاً از طریق) تراکنشها .

-**امنیت :** قابلیت تعریف و اعمال شرط در سطحهای مختلف برای دسترسی به اشیاء و واحدهای آنها

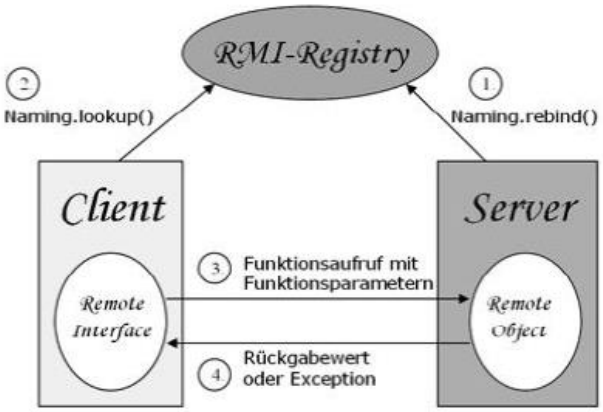
-**Query** : قابلیت انتخاب اشیاء یا کلاسها از مجموعه های تعریف شده.

- **نگارش :** قابلیت نگهداری، مدیریت و همخوانی کردن نگارشهای مختلف اشیاء[۳].

۳-٦ مقایسه کوربا با تکنولوژیهای مشابه

در اینجا به مقایسه کوربا ، DCOM و Java/RMI در نرم افزار اسکادا می پردازیم.

۳-٦-۱ RMI در مقایسه با کوربا:



شکل شماره ۳-۳: RMI در مقایسه با کوربا[٤]

۳-۶-۲ RMI چیست؟

RMI[1] مخفف عبارت Romote Methode invocation به معنی «فراخوانی متد از راه دور» است. همان طور که از نام این اصطلاح مشخص می شود، RMI مکانیسمی در اختیار برنامه نویسان جاوا قرار می دهد که می توانند از طریق آن متد اشیای گوناگون را بر روی یک ماشین مجازی راه دور اجرا کنند. مکانیسم های فراخوانی راه دور متفاوتی در دنیای نرم افزار ایجاد شده اند، اما RMI برخلاف بسیاری از آنها محدود به انواع داده ای اولیه یا ساختار هایی متشکل از داده های ساده نیست و به کمک آن می توان اشیای نرم افزاری را به تمامی همانند یک پارامتر عبور داد یا باز گرداند[۸].

این ویژگی از RMI یک مکانیسم منحصر به فرد می سازد. چنین خاصیتی به این معنی است که یک برنامه نویس جاوا می تواند به کمک RMI ، کدهای جدیدی را در شبکه انتقال دهد و در ماشین های مجازی راه دور به صورت دینامیک آن ها را اجرا نماید. بدین ترتیب برنامه نویسان جاوا در زمان برنامه نویسی سیستم های گسترده،آزادی عمل زیادی به دست خواهند آورد.در یک محیط گسترده، کلاینت های RMI می توانند به نسخه های جدید سرویس های جاوا دسترسی داشته باشند و نیازی به توزیع کردن برنامه به کلاینت ها نخواهد بود. این ویژگی همانطور که در محیط های شبکه محلی اجرا می شود، در محیط وب نیز قابل اجرا است.

در مکانسم RMI ، از ویژگی رجیستری هم پشیتبانی می شود .کلانیت های RMI با مراجعه به رجستری هم پشتیبانی می شود. کلاینت هایی که از سرویس مشخصی استفاده می کنند، می توانند با مراجعه به رجیستری RMI ، به آخرین نسخه سرویس دسترسی یابند. در صورت نیاز به کلاس جدید، می توان آن را از طریق وب سرور بارگذاری کرد.

RMI دارای ویژگی های دیگری علاوه بر آنچه در بالا به آن اشاره شد ، نیز هست . پردازش راه دور و تقسیم بار پردازنده ها نمونه هایی از قابلیت های ساده RMI تلقی می شوند RMI به جهت قابلیت انعطاف و سازگاری زیاد ، از طرف برنامه نویسان به سرعت به عنوان یک ابزار مهم برای توسعه سیستم های گسترده ، پذیرفته شد. اما زمانی که برنامه هایی به زبان های مانند c\c++ یا زبان های دیگری (غیر جاوا) بخواهند با RMI ارتباط بیابند،مشکلاتی ایجاد خواهد شد.

در واقع RMI تکنولوژی ویژه جاوا با نرم افزارهایی که به زبان هایی غیر جاوا توسعه یافته اند استفاده کرد. به همین دلیل ،استفاده از RMI به همان سرعتی رواج یافت ، رو به کاهش گذاشت. در واقع چنین استدلال

می شد که در دنیای امروز روز به روز بر مواردی که نیاز به ارتباط دادن نرم افزارهایی که با تکنولوژی هایی متفاوت توسعه یافته اند، بیشتر می شود که در صورت استفاده از RMI ، امکان توسعه آتی نرم افزار محدود خواهد بود[۹].

(۱) Romote Methode invocation

۳-٦-۳ کوربا یا RMI

همان طور که بیان شد، هر کدام از این دو تکنولوژی ها دارای ویژگی ها و مزایای خاص خود هستند و نمی توان هیچ یک را به طور مطلق جایگزین دیگری کرد. خواص هر یک از دو تکنولوژی موجب شده است که هر یک از آنها در حوزه فعالیت مشخصی به کار روند. خلاصه آن که اگر از مقایسه این دو تکنولوژی در اینجا نام برده می شود، علت آن است که از این طریق بهتر می توان قابلیت ها و نقاط قوت یا ضعف آن ها را شناسایی کرد.

تکنولوژی RMI قابلیت ها و برتری های قابل توجهی نسبت به کوربا دارد که در این نوشته از مهم ترین آن ها یاد کردیم. این تکنولوژی از زمان معرفی JDK۱,۲۰ به این طرف در دسترس همگان قرار داده شده است. به همین دلیل هم بسیاری از برنامه نویسان جاوا با این تکنولوژی آشنایی دارند و از تجربه استفاده از آن برخوردارند و باز به همین دلیل نیز در بسیاری از شرکت ها و سازمان ها از همان زمان ها سیستم هایی به کار گرفته شده اند که بر اساس RMI عمل می کنند. این موارد باعث شده اند که با وجود آنکه تکنولوژی کوربا در بسیاری از موارد کارها را برای برنامه نویسان سهل کرده است اما هنوز بسیاری از برنامه نویسان به کار با RMI ادامه می دهند و صنعت نرم افزار نیز به چنین برنامه نویسانی نیاز دارد.

از طرف دیگر تکنولوژی کوربا به دلیل سهولت کاربرد و قابلیت انعطاف پذیری و از همه مهم تر پرتابل بودن آن (امکان استفاده در زبان های گوناگون) طرفداران زیادی در بین برنامه نویسان یافته است. اما صرف نظر از این موضوع استفاده از این تکنولوژی در سازمان هایی که سیستم های نرم افزاری گسترده ای دارند و امکان جایگزینی تمامی بخش های آن ها با اجزای نرم افزاری جدید میسر نیست، امری غیر قابل اجتناب یا غیر قابل جایگزین تلقی می شود. تکنولوژی کوربا به راحتی می تواند بین بخش های نرم افزاری متفاوت و

سکوی های گوناگون ارتباط برقرار کرده و همانند پل ارتباطی به کار گرفته شود. در واقع تنها محدودیت کوربا در این زمینه آن است که برای آن که بتوان از این تکنولوژی در زبان های برنامه نویسی متفاوت استفاده کرد،باید قبلا پشتیبانی از کوربا در آن زبان فراهم شده باشد.

نکته مهم دیگر ،وضعیت سرعت و کارایی این دو تکنولوژی در مقایسه با یکدیگر است. مشخص شده است که کوربا از نظر سرعت اجرا تا حدودی بر RMI برتری دارد و به همین دلیل جذابیت بیشتری در بین برنامه نویسانی که در حوزه برنامه نویسی real-time کار می کنند ، یافته است[۳].

نقاط قوت	نقاط ضعف
توصیف سرویس‌ها، نیازمند استفاده از زبان توصیفی اینترفیس یا IDL دارد. زبانی که باید آموخته شود. استفاده از سرویس‌ها نیازمند به موجود بودن نگاشت IDL به زبان برنامه‌نویسی مربوطه است. در واقع انتظار برای عرضه نگاشتی برای زبانی که قبلاً نگاشت IDL برای آن فراهم نشده است، معقول نخواهد بود.	سرویس‌ها را می‌توان به زبان‌های متفاوت و تحت سکوهای گوناگون نوشت و از طریق هر زبان دلخواهی به‌کمک نگاشت IDL مربوطه، به آن دسترسی یافت .
ابزارهای تولید نگاشت IDL برای زبان‌های گوناگون، زیربنای کد اینترفیس‌ها را تولید می‌کنند. هرچند که برخی از چنین ابزارهایی احتمالاً تغییرات جدید را در کد پیاده‌سازی شده لحاظ نمی‌کنند.	با وجود IDL ، کد رابط یا اینترفیس به‌طور کامل از کد نرم‌افزار جدا نگه‌داشته می‌شود و به این دلیل برنامه‌نویسان قادر خواهند بود، پیاده‌سازی‌های متفاوتی برای کار با یک اینترفیس طراحی کنند.
از انتقال و تبادل اشیای نرم‌افزاری یا کدهای اجرایی، پشتیبانی نمی‌کند.	تکنولوژی کوربا برای کار با سیستم‌های نرم‌افزاری قدیمی‌تر ایده‌آل بوده وامکان توسعه آتی هر سیستم نرم‌افزاری را فراهم و تضمین می‌کنند.
آینده این تکنولوژی به‌نوعی در ابهام قرار دارد. در صورت عدم رواج و همه‌گیری کوربا در صنعت، این تکنولوژی به جمع تکنولوژی‌های قدیمی خواهد پیوست کوربا. روش ساده‌ای برای برقراری ارتباط بین اشیای نرم‌افزاری و	Corbaروش ساده‌ای برای برقراری ارتباط بین اشیای نرم افزاری و سیستم‌ها، ارائه می دهد.

سیستم‌ها، ارائه می‌دهد.	
تمام کلاس‌های نرم‌افزار نیاز به کارایی RealTime ندارند و می‌توان از سرعت و کارایی در برابر سهولت کاربرد در سیستم‌های تماماً جاوا و خالص، چشم‌پوشی کرد. کارایی و سرعت سیستم‌های مبتنی بر کوربا می‌تواند در حد قابل توجهی بالا باشد.	کارایی و سرعت سیستم‌های مبتنی بر کوربا می تواند در حدقابل توجهی بالا باشد .

جدول شماره ۳-۱: نقاط قوت و ضعف کوربا در مقایسه با RMI[۳]

تکنولوژی RMI قابلیت‌ها و برتری‌های قابل‌توجهی نسبت به کوربا دارد که در این نوشته از مهم‌ترین آن‌ها یاد کردیم. این تکنولوژی از زمان معرفی JDK۱,۲۰ به این طرف در دسترس همگان قرار داده شده است. به همین دلیل هم بسیاری از برنامه‌نویسان جاوا با این تکنولوژی آشنایی دارند و از تجربه استفاده از آن برخوردارند و باز به همین دلیل نیز در بسیاری از شرکت‌ها و سازمان‌ها، از همان زمان‌ها سیستم‌هایی به‌کار گرفته شده‌اند که بر اساس RMI عمل می‌کنند. این موارد باعث شده‌اند که با وجود آنکه تکنولوژی کوربا در بسیاری از موارد کارها را برای برنامه نویسان سهل کرده است، اما هنوز بسیاری از برنامه نویسان به کار با RMI ادامه می‌دهند و صنعت نرم افزار نیز به چنین برنامه نویسانی نیاز دارد.

از طرف دیگر تکنولوژی کوربا به دلیل سهولت کاربرد و قابلیت انعطاف‌پذیری و از همه مهم‌تر، پرتابل بودن آن (امکان استفاده در زبان‌های گوناگون)، طرفداران زیادی در بین برنامه نویسان یافته است. اما صرف نظر از این موضوع، استفاده از این تکنولوژی در سازمان‌هایی که سیستم‌های نرم‌افزاری گسترده‌ای دارند و امکان جایگزینی تمامی بخش‌های آن‌ها با اجزای نرم‌افزاری جدید میسر نیست، امری غیر قابل اجتناب یا غیر قابل جایگزین تلقی می‌شود. تکنولوژی کوربا به‌راحتی می‌تواند بین بخش‌های نرم‌افزاری متفاوت و سکوی‌های گوناگون ارتباط برقرار کرده و همانند پل ارتباطی به‌کار گرفته شود. در واقع تنها محدودیت کوربا در این زمینه آن است که برای آن که بتوان از این تکنولوژی در زبان‌های برنامه‌نویسی متفاوت استفاده کرد، باید قبلاً پشتیبانی از کوربا در آن زبان فراهم شده باشد[۳] و [۱۰].

نکته مهم دیگر، وضعیت سرعت و کارایی این دو تکنولوژی در مقایسه با یکدیگر است. مشخص شده است که کوربا از نظر سرعت اجرا تا حدودی بر RMI برتری دارد و به همین دلیل جذابیت بیشتری در بین برنامه نویسانی که در حوزه برنامه‌نویسی realtime کار می‌کنند، یافته است.

در بین این سه تکنولوژی Java/RMI محدود به زبان برنامه نویسی Java است. زبان Java به لحاظ مستقل بودن از سیستم عاملها و داشتن ساختار شیءگرا قوی ، زبان برنامه نویسی مهمی به شمار می رود . اما از آنجا که نرم افزارهای اسکادا معمولاً بر مبنای زبان C++ بنا نهاده شده است (که مهترین عامل انتخاب

آنها نزدیک بودن C به زبان ماشین و پاسخ دهی مناسب برای سیستمهای زمان حقیقی می باشد) لذا تکنولوژی Java/RMI از شانس کمتری برای انتخاب در نرم افزار اسکادا برخوردار می باشد[۳].

در اینجا به مقایسه تفاوتهای تکنولوژی DCOM و کوربا می پردازیم.

تکنولوژیهای DCOM و کوربا از زبان اینترفیس IDL خاص خود برای پیاده سازی دسترسی Client ها و سرورها که به زبانهای برنامه نویسی مهم نگاشته می شود سود می برند.

در کوربا زبان IDL که (OMG IDL نامیده می شود) بسیار ساخت یافته تر است. از قابلیت ارث بری چندگانه آن میتوان برای دوباره تعریف کردن اینترفیس اشیاء Server سود برد. در OMG IDL قابلیت انعطاف بیشتری در تعریف انواع داده های دارد. از جمله موارد اختلاف می توان به سری با طول ثابت و متغیر string و union اشاره نمود[٦].

اطلاعات اشیاء در کوربا در ساختاری به نام Implementation Repository ذخیره می شود .این ساختار از فایل برای نگهداری اطلاعات استفاده می کند و مشکلات Windows registry را از لحاظ قابلیت حمل پذیری را ندارد. همچنین سرویس نامگذاری ساختار گرافی را برای دستیابی اشیاء Server و Trading Service ساختاری بر مبنای ماهیت را برای پیدا کردن اشیاء ارائه میکنند.

در DCOM قابلیت دستیابی و فراخوانی دینامیکی متدهای شیء از طریق اینترفیس خاص IUknown پشتیبانی می شود. در کوربا این قابلیت از طریق ساختارهای ۳DII در Client و ساختار DSI در سرور پشتیبانی می شود[۳].

در بررسی مشخصات کوربا و DCOM می بینیم که در اصول ارتباط شیءگرا هر کدام روشی را برای حل مسائل مختلف درنظر گرفته اند و هر دو تا حدودی موفق بود ه اند. گرچه استاندارد کوربا با توجه به اینکه قدمت آن کمی بیشتر است و در تهیه آن شرکتهای مختلف سهیم بود ه اند همچنین از آغاز تاکنون تا حال به صورت مداوم در حال تکمیل و تدوین بوده است از ساختار منسجم تری برخوردار است و وظایف بیشتر و بهتر تقسیم شده است. برای مثال سیستم هایی که نیازی به فراخوانی دینامیکی متدها ندارند لازم نیست DII و DSI را پیاده سازی کنند .همچنین جستجو برای دستیابی اشیاء از طریق زیرسیستم های دیگر نظیر سرویس نامگذاری و تجارت صورت می گیرد .این سرویسها انحصاری نبوده و می توان از شرکتهای مختلف تهیه و به هم متصل نمود .همچنین استفاده از آنها اجباری نمی باشد و برای مثال در سیستم های زمان حقیقی برای بالا بردن کارایی می توان آنها را کنار گذاشت[۳].

همچنین با توجه به بازبودن استاندارد کوربا قابلیتهای جدید به صورت مداوم برای صورت مسئله های جدید در حال اضافه شدن است. برای مثال مشخصات minimum CORBA که مشخصات ساده شده ای از کوربا می باشد برای استفاده سیستمهای نهفته که منابع محدودی دارند، تعریف شده است.

مهمترین برتری کوربا نسبت به DCOM آزاد بودن و باز بودن استاندارد کوربا است. این ویژگی باعث می شود مثال به راحتی می توان که از انحصاری بودن محصول جلوگیری شود .برای ORB های مختلف را به صورت ترکیبی استفاده کرد و یا اگر یک ORB خواسته های جدید را برطرف نکرد به راحتی آن را با ORB شرکت دیگر تعویض نمود.

کوربا توسط شرکتهای بزرگ در محصولات مهم فراوانی مورد استفاده قرار گرفته است. با توجه به افزایش مداوم قابلیت های کوربا پیش بینی می شود این توجه روزافزون باشد.

کوربا در استانداردهای دیگر نیز مورد توجه قرار گرفته شده است .به طور خاص استانداردهای DAIS ، HDAIS و UMS-DAF که برای ارتباط ماژولهای نرم افزاری برای نرم افزارهای اسکادا و برنامه های مدیریت سیستم کنترل نظیر EMS/DMS توسعه داده شده اند، براساس استاندارد و تکنولوژی کوربا بنا نهاده شده است[۹].

۳-۷ نتیجه گیری

نتیجه گیری کلی که از مقایسه کوربا و DCOM داشتیم در این کلام خلاصه می شود که هر دو تکنولوژی در حل مسائل اساسی ارتباط شیءگرا در محیط شبکه تمهیداتی را کم و بیش داشته اند. اما در مجموع کوربا دارای ویژگیهای مهمتری از قبیل بازبودن ، عدم انحصاری بودن و وجود سرویسها و قابلیتهای مختلف (بخصوص قابلیت مربوط به زمان حقیقی، تحمل در برابر خطا و قابلیت استفاده در سیستمهای نهفته) می باشد. همچنین مهمترین مسئله ای که می توان در اینجا عنوان کرد، قابلیت استفاده کوربا تحت سیستم عامل های مختلف Unix و حتی تعدادی از سیستم عامل های زمان حقیقی می باشد. نگارش های مختلف UNIX از دیرباز به عنوان یک سیستم عامل مطمئن برای نر مافزارهای اسکادای برق به کار رفته است CORBA ذاتاً به سیستم عامل خاصی وابسته نیست و پیاده سازیهای بسیاری بر روی سیستم عامل های مختلف Unix و Windows وجود دارد.

با توجه به تجربیات در پیاده سازی نرم افزار اسکادا ، کوربا در عین حال که دارای قابلیتهایی زیادی بوده و ساختار یافته تر و انعطاف پذیرتر است، استفاده از آن حتی ساده تر از DCOM است. همچنین با تستهای کارآیی در سرعت ارسال اطلاعات پیاده سازیهای مختلف کوربا انجام گرفته است، سرعت قابل قبولی برای کاربردهای اسکادا داشته است[۳].

٤-١ فصل چهارم: بررسی DCOM[1] کوربا در شبکه های client /server محاسبه شی توزیع شده:

محاسبه شی توزیع شده تعمیم مدل سرویس دهنده/سرویس گیرنده می باشد.مدل سازی وبرنامه نویسی شی گرا برای توسعه سیستم های سرویس گیرنده/ سرویس دهنده بکار برده می شود.

درDOC[1] شی ها قطعاتی از نرم افزارها هستند که وضع داخلی را کپسول سازی می کنند و آن را بوسیله رابطی معین قابل دسترسی می سازند. رابط شامل عملکردهای شی وخصوصیت هایی می باشد که قابل دسترسی از راه دور می باشند[٤].

برنامه های سرویس گیرنده می تواند به یک نمونه دور رابط با کمک سرویس نامی متصل شود و سرانجام سرویس گیرنده ها متوسل به عملیاتهای روی شی دور می شوند. شی دور بنابراین مانند یک سرویس دهنده عمل می کند.

این استفاده از اشیا بدیهی است که ناهمگونی (نامتجانسی) و آزادی عمل (خودمختاری) را سازگار می کند: در ناهمگونی ، چونکه درخواستها به اشیا سرویس دهنده فرستاده می شود به رابط هایشان بستگی دارد و به داخلی هایشان وابستگی ندارد، آزادی عمل ، چونکه پیاده سازی شی می تواندبه وضوح تغییر کندبه شرط اینکه آنها را حفظ کنند. اگر سیستم های سرویس دهنده /سرویس گیرنده پیچیده با اشیا گردهم آورده شوند، پس اشیا باید با یکدیگر هماهنگ شوند. اشیا سرویس دهنده / سرویس گیرنده باید با یکدیگر عمل متقابل داشته باشند حتی اگر آنها در زبانهای برنامه نویسی مختلف نوشته شده باشند و حتی اگر روی سخت افزارهای متفاوت و پایگاه های سیستم عامل متفاوت اجرا شوند.

استانداردهایی برای اشیایی که در محیط های ناهمگونی کار می کنند لازم است.
یکی از استانداردهایی (DOC) مستقل فروشنده که کاملا پذیرفته شده است تعیین کوربا (معماری سیستم کارگزار درخواست شی مشترک) OMG (مدیریت گروه مدیریت شی) می باشد. کوربا شامل بلوکهای ساختمانی زیر می باشد:

زبان تعریف رابط : رابط های شی در زبانی بنام IDL (زبان تعریف رابط) توضیح داده می شوند. IDL کاملا شبیه زبان اعلان C++ می باشد. آن تصور رابط ها (مشابه با طبقه بندی ها) , تصور توارث رابط ها، تصور عملیاتهایی با آرگومان های (نشانوند ، شناسه) ورودی و خروجی و تصـــور انواع داده ها را تامین می کند که می توانند درسرتاسر عملیات انتقال داده شوند. پیاده سازی IDL برای اشیا سرویس دهنده قابل دسترسی دور اعلان در پایگاه و رویه خنثی زبان برنامه نویسی کار می کند و برای پیاده سازی آن اشیا کار

(۱) Distributed Component Object Model

(۲) Distributed object computing

نمی کند. اهداف کوربا در زبانهایی همچون جاوا و زبان برنامه نویسی شی گرایی و C , C++ پیاده سازی می شوند[٤].

کارگزار درخواست شی هدف ORB (کارگزار درخواست شی) پیدا کردن شی سرویس دهنده برای درخواست سرویس گیرنده می باشد وهمچنین آماده کردن شی برای دریافت کردن درخواست می باشد و همچنین منتقل کردن درخواست از سرویس گیرنده به شی سرویس دهنده می باشد و همچنین برگشت دادن آرگومان های خروجی به کاربرد سرویس گیرنده می باشد. ORB عمدتا قابلیت RPC شی گرا را تامین می کند. آداپتور (تطبیق دهنده) شی اصلی BOA (تطبیق دهنده شی اصلی) رابط اصلی می باشد که بوسیله شی سرویس دهنده استفاده می شود تا عملکرد ORB را قابل دسترسی قرار دهد. BOA عملیاتهایی را صادر می کند تا آدرس شی را بوجود بیاورد وهمچنین اشیا سرویس دهنده را فعال می کند و ثبت کند و همچنین درخواستها را شناسایی کند. آدرس شی ، یک ساختمان داده ها می باشد که دلالت بر یک شی سرویس دهنده در شبکه می کند. سرویس دهنده آدرسش را در یک سرویس دهنده نام نصب می کند به طوری که برنامه کاربردی سرویس گیرنده می تواند آدرس را بازیابی کند وسرویس دهنده را احضار کند.آدرس شی , همان رابطی را به عنوان شی سرویس دهنده تهیه می کند که آن ارجاع می دهد. جزییات مربوط به زیر بنای ارتباطی واقعی از سرویس گیرنده پنهان می شوند[٤].

٤-۲ رابط احضار دینامیک

DII متناوب سطح پایینی برای توابع ناقص ارتباطی می باشد که با تعریف IDL بوجود می آیند.پروتکل INTER-ORB اینترنت پروتکل INTER-ORB اینترنت (IIOP) اجازه می دهد تا ORB های کوربا فروشنده های مختلف بوسیله ارتباط TCP/ IP در چند محیط عمل کنند. IIOP پروتکل RPC مختصر شده می باشد که مورد استفاده قرار می گیرد تا اشیا سرویس دهنده بوسیله اینترنت را در روشی کارآمد و قابل حمل احضار کند[٤].

٤-٣ رابط وگنجینه اطلاعات پیاده سازی: گنجینه اطلاعات رابط کوربا

پایگاه داده ای شامل اطلاعات نوع (اسامی رابط , عملکردهای رابط , انواع آرگومان) برای رابطه های موجود در سیستم کوربا می باشد.

این اطلاعات برای احضار دینامیک از طریق DII و برای کنترل تجدید نظر و غیره استفاده می شود. گنجینه اطلاعات پیاده سازی , اطلاعاتی را در دسترس قرار می دهد که اجازه می دهد ORB تعیین محل کند واشیا سرویس دهنده را راه ندازی کند[٤].

٤-٤ جعبه ابزار سرویس دهنده/سرویس گیرنده:

دامنه وسیع جعبه ابزارهای نرم افزاری برای نرم افزار سرویس دهنده/سرویس گیرنده ساختمانی امروز در بازار موجود می باشد. جعبه ابزارهای سرویس دهنده/سرویس گیرنده به عنوان میان نرم افزار ارجاع داده می شوند. پیاده سازی کوربا یک نمونه میان نرم افزارسرویس دهنده/سرویس گیرنده معروف می باشد. نمونه های دیگر OSF DCE , DCOM ونمونه میان نرم افزارپیام گرا , نمایشگرهای پردازش/ تغییرات می باشد[٤].

OSF DCE : پایه واساس نرم افراری باز(OSF) محیط محاسبه توزیعی (DCE) استاندارد غیر رسمی برای سیستم های سرویس دهنده/سرویس گیرنده چند فروشنده ای می باشد .

DCOM : مدل شی مولفه توزیع شده پروتکل (قرداد) شی مایکروسافت می باشد, که این امکان را به مولفه های ActiveX می دهد تا با یکدیگر از طریق شبکه کامپیوتری ارتباط برقرار کنند مولفه ActiveX , شی قابل دسترسی راه دوری می باشد که رابط معین و مشخصی دارد و خودکفا می باشد. مولفه ActiveX می تواند در اسناد وب تعبیه شود بطوری که آنها بطور خودکار برای سرویس گیرنده می گیرند ودانلود می کنند, تا در مرورگر وب سرویس گیرنده اجرا کنند. DCOM سهولت موقعیت راه دور را تامین می کند که اجازه می دهد سرویس گیرنده ها، اشیا سرویس دهنده راه دور را ا یجاد کنند. آن مدل ایمنی را در دسترس قرار می دهد که اجازه می دهد برنامه نویسان محدود کنند کسانی را که ممکن است شی

سرویس دهنده ای را ایجاد کنند و کسانی را که آن را احضار می کنند. سرا نجام , زبان تعریف رابط برای تعریف رابط های شی قابل دسترسی راه دور و فراخوانی روال از راه دور در نظر گرفته می شود.
امروزه بحث و گفتگو درباره اشیا نرم افزار , مولفه ها و کاربردهای مبنی بر مولفه درباره کنترلهای ActiveX , جاوا بین (javaBeen) و سرویس دهنده تغییری مایکروسافت و جاوا بین های شرکتی می باشد و اینکه چگونه آنها می توانند با یکدیگر در چند محیط عمل کنند.نقطه مشترک این هست که همه موارد ذکر شده بالا مدل های مولفه نرم افزاری هستند. در پارگراف های بعدی این بخش ما تعدادی از اقداماتی را ارایه خواهیم کرد که برای پل زدن بر ORBA , DCOM و RMI انجام شده اند که متداول ترین تکنولوژی راه دور میان نرم افزاری تجاری می باشند. پل (واسطه) CORBA-DCOM , به عنوان توسعه COM , دو تکنولو ژی راه دور میان نرم افزار مهم می باشند. اهمیت شان از آبا واجدادشان نشات می گیرد. کوربا فرزند گروه مدیریت شی می باشد که در حقیقت پیوند و رابطه ای شامل سان میکروسیستم و کامپک[1] وهولت پاکارد[2] و Iona و مایکروسافت و دیگران می باشد[٤] و [١١].

---

(١) Compaq

(٢) Sun microsystems

در حالیکه DCOM از شرکت مایکروسافت ناشی می شود که بیشترین بخش از بازار سیستم عامل محیط اصلی دسکتاپ را در اختیار دارد . هرچند که COM و محصول توسعه یافته اش DCOM در OSS مایکروسافت ساخته می شوندو تهیه کنندگان دیگری بر این تکنولوژی ها وجود ندارند اما اختیار فراوان OSS مایکروسافت و توسعه زبانهای برنامه نویسی که ساختارهای DCOM/COM پرتوان و وسیع را حمایت و پشتیبانی می کنند منجر به تولید مولفه های زیادی بر اساس معماری سیستم مایکروسافت شد. از طرف دیگر ، این واقعیت که OMG ، کوربا را به عنوان مشخصاتی برای ORB ها سیستمی کرد در عوض اینکه به تولیدی منجر شود، منجر به این امر شد که شرکت زیادی کارگزاران درخواست آماده انجام کوربا خودشان را ایجاد کنند, و دامنه ای از ORB هایی را که قادر به ارضای خواسته های متفاوت بودند در اختیار مصرف کنندگان و شرکت های توسعه دهنده می گذاشت[١١].
OMG , نیازبرای پل زدن بر تفاوت هایش را درک کرد و بعد از پل OLE/CORBA نخستین از سوی تکنولوژی های IONA در سال ١٩٩٥ تصمیم گرفت تجدید نظر و اصلاح امروزی شده اش ، معماری سیستم کوربا و معماری سیستم کار متقابل تعیین را در برداشته باشد کـه تعیینی برای پل زدن بر کوربا و OLE/COM می باشد. معماری سیستم کار متقابل سه نقطه را آدرس می دهد: نقشه برداری (طرح دهی) رابط از آنجایی که هر دو مدل IDL هایی را برای تعریف رابط ها استفاده می کنند وچونکه هر شی

توسط رابطشان نشان داده می شوند, باید نقشه برداری ای میان آنها برای شی کوربا وجود داشته باشد تا بتوان شی COM و بر عکس در نظر گرفته شود . مخصوصا , OMG چهار نقشه برداری (طرح دهی) متفاوت را مشخص می کند که عبارتند از خودکار کردن CORBA/COM , CORBA/OLE , COM/CORBA و خودکار کردن OLE نقشه برداری کوربا[٤].

یکی از تفاوتهای اساسی میان رابط های CORBA و COM , ویژگی های وراثتی می باشد. در حالیکه کوربا وراثت رابط چند گانه را حمایت می کند ، COM فقط وراثت منفرد را تایید می کند . برای اینکه پل موفقیت آمیز می باشد باید نقشه ای از وراثت چند گانه CORBA برای وراثت منفرد COM و برعکس وجود داشته باشد.

نقشه برداری شناسایی این تعین به نقشه برداری میان ID های رابط متفاوت مربوط می شود که بوسیله کوربا ، COM استفاده می شوند[٤].

---

(١) peer logic

(٢) Orbix Comet

OMG پیاده سازی پل COM/CORBA را تهیه نمی کند بلکه فقط مشخصاتی را در دسترس قرار می دهد . کار پیاده سازی برای شرکت هایی گذاشته شده است که انجام ابزار پلی زیادی را با تعیین OMG عرضه کرده اند . تعدادی از این محصولات COM٢CORBA پی یر لاجیک[١], محیط اصلی از IONA وپل مقصد لبه ای بصری (ویژوال) می باشند . همه محصولات بالا یکی از نقشه برداری های رابط را تحقق می بخشد که OMG مشخص می کند و هدف اصلی شان تهیه کردن کار متقابل دو طرفه میان برنامه های کاربردی COM و کوربا می باشد[٤] و [١١].

٤-٥ تکنولوژی COM/DCOM

POLLX یک مولفه COM می باشد که معمولا وظایفی را که مربوط به ایجاد ومدیریت (اداره) انتخابات از قبیل زمانبندی کردن انتخابات ، اضافه کردن انتخابات جدید ، تصحیح کردن انتخابات موجود ، بررسی کردن نتایج انتخابات به طور همزمان را خودکار می کند. POLLX شامل سه طبقه می باشد که روشها وخواص انتشار را متوقف می کند[٤].

٤-٦ نتایج

عمل کردن در چند محیط در میان اشیا تکنولوژی متفاوت , درعمل بسیار پیچیده تر و مشکل تر نسبت به تیوری می باشد. اگر چه سعی وکوشش ها واقدامات زیادی متعهد شده اند که فاصله و شکاف میان معماریهای زیر بنایی اشیا را پل بزنند اما آنها به اندازه کافی مقارن این حال نیستند تا فروشنده حقیقی , زبان حقیقی وعمل کردن در چند محیط مستقل میان اشیا نرم افزاری مختلف را تهیه کنند. متاسفانه , استفاده از محصول میان افزار منفرد, قابل اعتماد ترین راه حل بوده است . مشکلات ناسازگاری میان محصولات کمپانی های مختلف ادامه دارد حتی اگر محصولات تکنولوژی یکسان داشته باشد. حتی برای ابزارهای پلی در دسترس اظهاراتشان غالبا به گزینش وانتخاب فروشندگان منفردی مربوط می شود که نظریه استقلال فروشنده را حمایت نمی کنند.

پژوهشمان به توسعه معماری قادر به تهیه بافت مستقل برای ساختن واسطه ها و به توسعه معماری قادر به یکپارچه سازی اشیا توزیع شده چند تکنولوژی مربوط می شود. ما زبان جاوا را به عنوان ابزار برنامه نویسی برای ایجاد مکانیسم واسط استفاده کردیم و همچنین آن را به عنوان چسباننده شی چند منظوره استفاده کرده ایم. نتایج کارمان تاکنون پابت کرده است که یکپارچه سازی تکنولوژی های راه دور میان نرم افزار مختلف ممکن می باشد[٤].

تاکنون معماری سیستم عمل کردن در چند محیط در میان سرویس گیرنده RMI و سرویس دهنده کوربا، سرویس گیرنده کوربا و سرویس دهنده RMI ، سرویس گیرنده کوربا وسرویس دهنده COM و سرویس گیرنده RMI و سرویس دهنده COM را اجازه می دهد. عمل کردمان در چند محیط در میان RMI و کوربا به حمایت پروتکل IIOP در معماری RMI بستگی ندارد[٤].

۵-۱ فصل پنج: کوربا و بررسی امنیت در شبکه

امنیت برای سیستم های کامپیوتری مدرن امری ضروری است. بخصوص برای سیستم های توزیع شده که نسبت به سیستم های قدیمی نفوذ پذیرتر می باشند. زیرا مکانهای بیشتری برای حمله کردن به آنها وجود دارد. بنابراین سیستم های کوربا نیز که طبیعت توزیعی دارند به امنیت نیاز بیشتری دارند[۱].

۵-۲ امنیت چیست؟

بطور کلی امنیت یک سیستم اطلاعاتی را از دسترسی غیرمجاز و نفوذ و تداخل در عملکردهایش محافظت می نماید[٦].

۵-۳ اهداف سیستم امنیتی کوربا

همان اصول سیستم های امن اساس سرویس های امنیتی کوربا را تشکیل می دهند.

اگرچه سیستم های کوربا به طور کلاسیک سیستم اطلاعاتی نیستند اما طبیعت ذاتی آنها منجر به ایجاد تهدیدات بالقوه می شود که ممکن است در معماری آنها لحاظ نشده باشد. بنابراین بعضی از اهداف امنیتی مختص امنیت کوربا می باشد. این اهداف شامل شامل موارد زیر است:

-فراهم نمودن امنیت از طریق سیستم ناهمگون

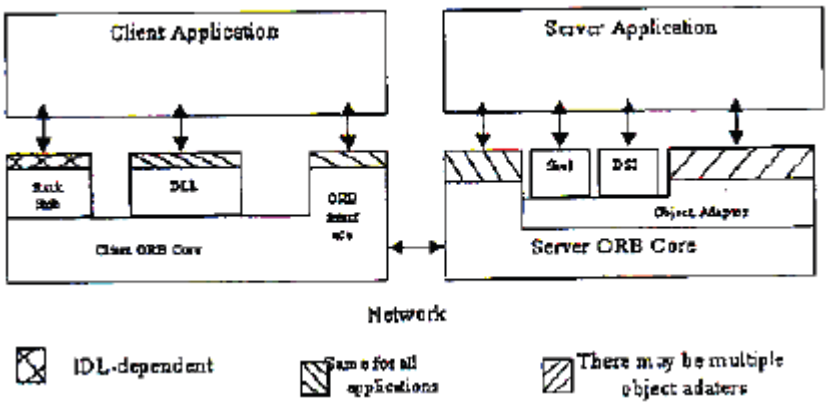
-شی ء گرایی

-فراخوانی های امن اطمینان بخشی از حفظ امنیت فراخوانی ها

-کنترل و نظارت (control & audiling): اطمینان به شی ء ازینکه نظارت و کنترل بر روی دسترسی و فراخوانی های object ها انجام می گیرد[۱].

۵-٤ سرویس امنیتی کوربا

سرویس امنیتی کوربا یک مشخصه موثر و یا حجم بالا است که قبل از تصویب آن در سال ۱۹۹٦ بیش از دو سال تلاش به ثمر رساند. بعد از تصویب آن تقریبا دو سال دیگر برای اولین پیاده سازی آن وقت صرف شده و محصول مورد نظر به بازار آید. علت زمان طولانی برای برای ورود آن به بازار دو مسئله بود اول آنکه امنیت توزیع شده یک مسئله پیچیده و مشکل است و دوم اینکه فضایی که بوسیله ویژگی کوربا فراهم می شود معمولا تنها در سیستم های بسیار بزرگ مورد نیاز است که تامین زمانهای اخیر تعداد خیلی کمی از آنها بر پایه کوربا عمل می کردند. با فراگیر شدن کوربا به عنوان یک تکنولوژی غالب در سیستم های بزرگ تجاری ، تقاضا برای یک راه حل امنیتی قوی نیز افزایش یافت[۱] و [۵].



شکل شماره ۵-۱: جریان عمومی درخواست در کوربا[۱]

تا قبل از CORBA ۲,۰ نیاز زیادی برای تعریف یک پروتکل استانداردی که بتواند امکان ارتباط برنامه های کاربردی را با یکدیگر ایجاد کند ضروری به نظر می رسید تا آن زمان هر فروشنده ORB پروتکل شبکه خود را به کار می برد و این امر باعث می شد تا جزایر برنامه های کاربردی ORB ایجاد شود. بطوریکهORB ها نمی توانستند با هم ارتباط برقرار کنند. CORBA ۲,۰ معماری عمومی را برای قابلیت میان عملیاتی ORB ارائه داد که آنرا پروتکل عمومی درون ORB با اختصار GIOP نامیدند[۱].

GIOP پروتکلی است که قواعد انتقال و فرمت پیامها را به نحوی که مستقل از ORB باشند، تعریف می کند. پروتکل درون ORB اینترنت مشخص می کند که چگونه GIOP روی پروتکل TCP/IP پیاده سازی می شود. با استفاده از پروتکل درون ORB اینترنت ، یک CORBA ORB از یک فروشنده می تواند با ORB از فروشنده دیگر ارتباط برقرار کند. محصولاتی با کوربا سازگار هستند که با پروتکل درون ORB اینترنت پیاده سازی شده باشند. در چنین حالتی پروتکل درون ORB اینترنت قابلیت میان عملیاتی بین محصولات کوربا را تعیین می کند. پروتکل های دیگری برای ارتباط بین ORB ها و وجود دارد، اما پروتکل درون ORB اینترنت، به خاطر استاندارد بودن و نیز پروتکل انتقال TCP/IP سریعتر از بقیه همه گیر گشت[۵].

۵-۵ ویژگی های سرویس امنیتی کوربا

سرویس امنیتی کوربا دامنه ای وسیع از مقوله های مربوط به امنیت را شامل می شود. این سرویس علاوه بر احتیاجات اولیه امنیتی مانند تایید اعتبار ، یکپارچگی داده ها، privacy و Authorization شامل چندین مقوله مهم دیگر نیز می باشد که عبارتند از:

گزارش و نظارت Auditing :برای حفظ امنیت ، وقایع مهمی مانند رد تایید اعتبار و تغییرات حق امتیاز و ....باید گزارش و در صورت لزوم پی گیری شوند.

عدم رد Nonrepudiations: فراهم نمودن مدرکی برای عدم رد فعالیت ها، برای مثال به گونه ای که منشا اصلی داده را برای گیرنده نشان داده و یا عمل دریافت داده ها را برای فرستنده اثبات نماید.

Interoperabillity: یکی از ویژگی های برجسته کوربا می باشد. این سرویس امنیتی برای به انجام رساندن این هدف پروتکلی خاص به نام seciop[۱] را تعریف نموده است.

قابلیت جایگزینیReplaceabillity: هر زمان که ممکن باشد ،سرویس امنیتی کوربا اجازه جایگزینی سرویس های امنیتی مانند سرویس های تایید اعتبار و... را می دهد[۱].

۵-۶ CORBA و SSL[۲]:

پروتکل ارتباطی اولیه ای که امروزه به وسیله ORB های کوربا استفاده می شود ، پروتکل IIOP می باشد.
IIOP به خودی خود شامل یک مکانیزم امنیتی نمی باشد
برای جبران این فقدان ، شرکت های مختلف برنامه های امنیتی پدید آورده اند که بر SSL تکیه دارند.
SSL تاثیر ناچیزی بر درخواست های کاربر و سرور در کوربا دارد و برای آنها مزاحمتی ایجاد نمی نماید زیرا SSL به عنوان یک لایه امنیتی جداگانه بین IIOP و TCP/IP قرار گرفته است(مطابق شکل زیر).

شکل شماره ۲-۵: SSL به عنوان یک لایه امنیتی جداگانه بین IIOP و TCP/IP[۱]

فواید امنیتی حاصل از تجهیز کوربا با SSL شامل موارد زیر است:

AUTHENTICATION
DATA INTEGRITY
CONFIDENTIALITY یا PRIVACY

علی رغم امتیازات ذکر شده فوق ، CORBA/SSL هنوز احتیاجات امنیتی اعطای مجوز[۳] را برای کاربر برآورده نمی کند.

(۱) secure inter- ORB protocol

(۲) secure sockets layer

(۳) AUTHORIAZATION

با توجه به مساله ذکر شده برای محیط هایی که در آنها به موازات تایید اعتبار ، یکپارچگی داده ها و privacy ، اعطای مجوز کاربر نیز لازم است ، سرویس امنیتی کوربا به بازار عرضه شد[۱].

٦-١ فصل ششم: ارائه و ارزیابی یک مدل ترکیبی برای فراهم کردن توزیع بار و تحمل پذیری خطا در کوربا

٦-٢ مقدمه

ابتدا قبل از پرداختن به روش پیشنهادی مروری خواهیم داشت بر روشهایی که برای هر کدام از تحمل پذیری خطا و توزیع بار در کوربا وجود دارند .از این میان ابتدا به روشهای تحمل پذیری خطا می پردازیم.

٦-٣ روشهای موجود برای تحمل خطا در کوربا

روشهای تحمل خطا در کوربا را بطور کلی می توان به سه روش کلی طبقه بندی کرد ، انجام می گیرد .

روش یکپارچگی: تکرار سازی ، به طور یکپارچه با ORB انجام می گیرد.
روش حائل: تکرار سازی ، بصورت شفاف و زیر لایه ای ORB فراهم شده
روش سرویس: تکرار سازی ، توسط مجموعه ای از اشیاء کوربا بالای لایه ای ORB به عنوان یک سرویس فراهم شده است[٤].

با وجود تفاوتهای روشهای مختلف ذکر شده، این روشها در مورد استفاده از تکرار سازی ها در مقابل بروز خطا در برنامه ها بصورت مشابه عمل می کنند .برنامه های کوربا می توانند توسط تکرار کردن اشیاء سازنده شان تحمل پذیر خطا شوند و این تکرار ها را بین پردازشگر های مختلف در شبکه توزیع کنند. ایده ای که در تکرار کردن اشیاء وجود دارد این است که در صورت خراب شدن یکی از تکرارها (یا خراب شدن و از رده خارج شدن یکی از پردازشگرها) این خرابی می تواند از دید سرویس گیرنده مخفی شود و سرویس دهی توسط تکرار های دیگر به سرویس گیرنده ادامه پیدا کند.
بطور کلی ۲ نوع تکرار سازی وجود دارد :تکرار سازی فعال و تکرار سازی غیر فعال .با تکرار سازی فعال هر تکرار سرویس دهنده تمام فراخوانی های سرویس گیرنده را پردازش می کند و جواب را به سرویس گیرنده باز می گرداند. در مورد تکرار سازی غیرفعال فقط یکی از تکرار های سرویس دهنده به عنوان تکرار اصلی در نظر گرفته می شود .این تکرار تمام فراخوانی های سرویس گیرنده را پردازش می کند و پاسخ ها را به سرویس گیرنده ها باز می گرداند .تکرار سازی غیرفعال، خود به ۲ نوع تقسیم می شود: تکرار سازی غیر فعال گرم و تکرار سازی غیرفعال سرد [۸] و [٤] .

٦-٤ استراتژی های توزیع بار در کوربا

هنگام طراحی سرویس توزیع بار در کوربا می توان استراتژی های مختلفی را مد نظر قرار داد .این استراتژی ها را می توان به ۲ دسته تقسیم کرد.
-دانه بندی مقیدسازی سرویس گیرنده: هر بار که یک تصمیم توزیع بار گرفته می شود، یک توزیع کننده بار درخواست یک سرویس گیرنده را به یک تکرار مقید می کند. به طور خاص، یک درخواست سرویس گیرنده به تکراری که توسط توزیع کننده بار انتخاب شده مقید می شود.
مکانیزم انقیاد سرویس گیرنده شامل پیغام GIOP در LOCATION_FORWARD و یا سرویس های استاندارد تغییر یافته ی کوربا می شود .بدون توجه به مکانیزم ، انقیاد سرویس گیرنده را می توان به دسته های زیر تقسیم کرد:

جلسه ای : درخواست های سرویس گیرنده در مدت یک جلسه به همان تکرار فرستاده می شود.

درخواستی : هر درخواست سرویس گیرنده به تکرارهای مختلف فرستاده می شود.

تقاضایی : در صورت نیاز از طرف توزیع کننده بار، درخواست های سرویس گیرنده می تواند دوباره به یک تکرار دیگر مقید شود.

سیاست توزیع : هنگام طراحی یک سرویس توزیع بار مهم است که یک الگوریتم مناسب برای تصمیم گیری اینکه کدام تکرار درخواست را پردازش خواهد کرد، انتخاب شود .عموماً سیاست های توزیع بار را می توان به شاخه های زیر تقسیم کرد:

غیر تطبیقی : توزیع کننده بار می تواند از سیاست های غیرتطبیقی مانند الگوریتم گردشی ساده و یا الگوریتم تصادفی برای انتخاب تکرار استفاده کند.

تطبیقی : توزیع کننده بار می تواند از سیاست های تطبیقی که از اطلاعات زمان اجرا مانند مقدار زمان استفاده از CPU در سرویس دهنده ها بهره می برند، برای انتخاب تکرار، استفاده کند[٤].

٦-٥ ارائه ی یک روش توزیع شده برای توزیع بار در کربا

ساختار و اجزاء تشکیل دهنده و وظایف هر عنصر:

آ.سرویس دهنده های تکرار شده

ب .سرویس گیرنده های تکرارشده

ج .گروه های اشیاء هم نوع: سرویس دهنده های همنوع همراه با سرویس گیرنده هایی که به تکرارهای آن سرویس دهنده نیاز داشته باشند، در یک گروه شئ قرار می گیرند.

د .یک پروتکل ارتباط گروهی قابل اطمینان: برای اطلاع از بار سرویس دهنده ها به یک پروتکل ارتباط گروهی نیاز است تا مانیتور کننده بار بتواند اطلاعات بار سرویس دهنده های مختلف را در خود ثبت کند.

ه .مانیتور کننده بار: مانیتور کننده بار وظیفه خواندن و ثبت بار سرویس دهنده ها را بر عهده دارد.

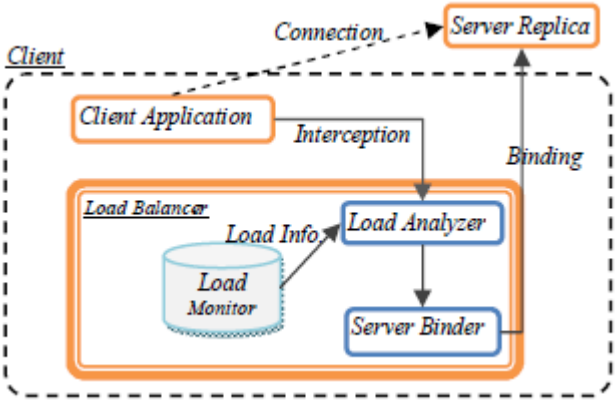
و .آنالیز کننده بار: این بخش اطلاعات بار سرویس دهنده را از مانیتور بار دریافت کرده و بر اساس همان اطلاعات و سیاست داده تعریف شده سرویس دهنده را انتخاب می کند.

ز .پخش کننده بار: این یک بخش کلی است که شامل قسمت های آنالیز کننده بار، مانیتور بار و مقید کننده سرویس دهنده می شود.

ح .مقید کننده سرویس دهنده: این بخش، سرویس دهنده انتخاب شده در قسمت آنالیز کننده بار را دریافت می کند سپس ابتدا از وجود سرویس دهنده مورد نظر اطمینان حاصل می کند و آن را به سرویس گیرنده

جاری مقید می کند. در صورت موجود نبودن سرویس دهنده، به آنالیز کننده بار گزارش می دهد تا سرویس دهنده دیگری انتخاب شود.

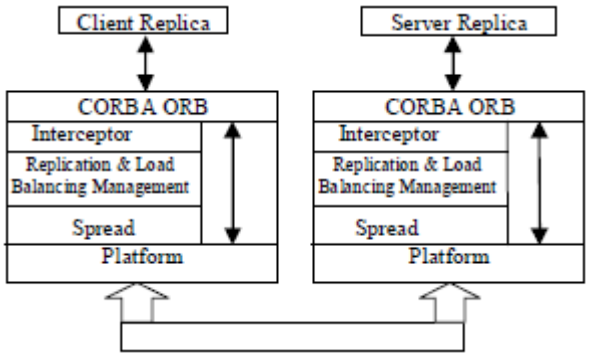
ط .حائل: این بخش به عنوان یک حائل بین درخواست سرویس گیرنده و سرویس دهنده قرار می گیرد تا سیاست های سیستم روی درخواست ارسالی اعمال شود[٤].



شکل شماره ٦-١: قرارگیری اجزا در معماری توزیع بار توزیع شده[٤]

٦-٦ معماری کلی سیستم

در سیستم پیشنهادی همانطور که در شکل ٦-٢ مشاهده می شود از یک پروتکل ارتباط گروهی به نام spread استفاده شده است spread یک ابزار متن باز است که سرویس ارسال پیام را با کارایی بالا انجام می دهد. این ابزار در مقابل خرابی ها در شبکه های محلی و WAN مقاوم است. SPread به عنوان یک گذرگاه یکپارچه پیام برای برنامه های کاربردی توزیع شده استفاده می شود و چند پخشی در سطح برنامه کاربردی ، ارتباط گروهی و پشتیبانی نقطه به نقطه را فراهم می کند[٤].



شکل شماره ٦-٢: معماری کل سیستم پیشنهادی[٤]

٦-٧ بررسی مزایا و معایب و محدودیت های روش پیشنهادی

بررسی سیستم پیشنهادی از نظر توزیع بار:روش پیشنهادی با تکیه به زیرساخت ارتباطی خود و ویژگیهای سرویس دهنده ها می تواند هم از توزیع بار تطبیقی و هم غیر تطبیقی استفاده کند و نیز سیاست های مختلف از قبیل درخواستی، جلسه ای و تقاضایی را پشتیبانی کند[٤].

- **بررسی سیستم پیشنهادی از نظر تحمل پذیری خطا:** سیستم پیشنهادی ازنظر تحمل پذیری خطا از روش حائل استفاده می کند.

همانطور که قبلا نیز به آن اشاره شد، این روش قابلیت تحمل پذیری خطا را بدون آگاهی هیچکدام از سرویس دهنده یا سرویس گیرنده انجام می دهد .که از این نظر یک نقطه ی قوت برای سیستم پیشنهادی محسوب می شود. با توجه به ثبت درخواست ها و checkpoint گیری از سرویس دهنده ها و ذخیره آنها در سرویس دهنده های دیگر، امکان پشتیبانی از یکپارچگی قوی تکرارها ممکن است ، و می تواند وضعیت سرویس دهنده ها را بروز کند. یکپارچگی قوی تکرارها این اطمینان را می دهد که تمام تکرارهای یک شئ حالت واحد دارند و رفتار یکسانی را نشان می دهند[٨] و [٤] .

- **بررسی سیستم پیشنهادی از نظر قابلیت توسعه:** از آنجایی که سیستم پیشنهادی بصورت توزیع شده و غیر متمرکز عمل می کند، بنابراین می تواند براحتی توسعه یابد .هر سرویس گیرنده هنگام انتخاب سرویس دهنده بطور کاملا مستقل بر پایه اطلاعات دریافتی عمل می کند و نقطه ی مرکزی که باعث بوجود آمدن گلوگاه شود، وجود ندارد[٨] و [٤] .

-**بررسی سیستم پیشنهادی از نظر پیچیدگی:** این سیستم از لحاظ پیاده سازی پیچیدگی چندانی ندارد .اما در صورتیکه بنا بر آن باشد که از استراتژی تطبیقی استفاده شود، پیچیدگی سیستم افزایش خواهد یافت . زیرا برای تعیین تکرار سرویس دهنده ی بعدی باید از میزان بار آن مطلع شویم[٤].

- **بررسی سیستم پیشنهادی از نظر شفافیت:** همانطور که گفته شد ، سیستم به علت استفاده از روش حائل، نسبت به سرویس دهنده و سرویس گیرنده، کاملا شفاف است. معماری سیستم پیشنهادی هیچ ارتباطی به چگونگی پیاده سازی سرویس دهنده یا سرویس گیرنده ندارد. به همین دلیل می توان روش پیشنهادی را برای سیستم های سرویس دهنده و سرویس گیرنده ی مختلف نیز استفاده کرد بدون اینکه اطلاعی از وجود مکانیزم داشته باشند.

-**معایب سیستم:** اصلی ترین عیب سیستم، تعداد زیاد ارسال پیغام چند پخشی در سیستم است .که به علت هماهنگ سازی سرویس دهنده ها با یکدیگر و سرویس دهنده ها با سرویس گیرنده ها صورت می گیرد.

٦-٨ پیاده سازی روش پیشنهادی

این روش با استفاده از ابزار دیتورز از شرکت میکروسافت در سیستم عامل ویندوز پیاده سازی شده است ORB ای که استفاده شده است، MICO نام دارد .با استفاده از MICO برنامه های سرویس دهنده و سرویس گیرنده ی مورد نظر را ایجاد می کنیم، سپس فایل های DLL تولید شده برای حائل سازی را با استفاده از ابزارهای دیتورز به برنامه های سرویس دهنده و سرویس گیرنده اضافه می کنیم.

در این حالت DLL مربوط به حائل سازی تابع کتابخانه ای socket به برنامه سرویس گیرنده و DLL مربوط به حائل سازی تابع bind به برنامه سرویس دهنده متصل می شوند .توابع دیتور در داخل فایل های DLL قرار دارند به همین دلیل هنگام اجرای برنامه ها، باید فایل های DLL مربوطه کنارشان باشد.

با استفاده از این روش هیچ نیازی به تغییر کد در برنامه ها وجود ندارد و برنامه کاربردی نیز اطلاعی از مکانیزم توزیع بار و تحمل خطا نخواهد داشت[٤].

هنگام اجرای یک برنامه سرویس دهنده ، قبل از اجرا شدن تابع کتابخانه ای bind تابع interceptedBind از فایل DLL مربوطه خوانده و اجرا می شود .در این زمان اطلاعات مربوط به آدرس IP و پورت و IOR سرویس دهنده خوانده شده و به حوضچه ی سرویس دهنده ها اضافه می شود .وقتی یک برنامه سرویس گیرنده اجرا می شود برای اتصال به سرویس دهنده، تابع socket را فراخوانی خواهد کرد که در این صورت بجای اجرای تابع کتابخانه ای، تابع دیتور intercepted Socket از فایل DLL مربوطه فراخوانی و اجرا می شود .این تابع اطلاعات یکی از سرویس دهنده های موجود در حوضچه ی سرویس دهنده ها را انتخاب می کند تا به آن متصل شود. در صورتیکه اتصال با موفقیت انجام نشد، مکانیزم توزیع بار و تحمل پذیری خطای تعریف شده، سرویس دهنده ی دیگری را از حوضچه بر اساس سیاست تعیین شده (گردشی) انتخاب می کند و دوباره برای اتصال تلاش می کند .لازم به ذکر است که سرویس دهنده ای که به دلیل هر مشکلی وجود نداشته باشد، از حوضچه حذف خواهد شد[٤].

٦-٩ شبیه سازی روش پیشنهادی

در این قسمت نتایج شبیه سازی به عمل آمده را عرضه کرده و مورد بررسی قرار خواهیم داد .در این شبیه سازی سعی شده است که به نتایج مربوط به میانگین زمان پاسخ برای درخواستها و همچنین گذردهی سیستم در حالات مختلف پرداخته شود .این شبیه سازی برای تعداد ٥٠٠٠٠ درخواست اجرا شده است .از مقادیر دیگر داده شده برای تمام حالات شبیه سازی می توان به مقادیر زیر اشاره کرد:

سرویس دهی متوسط سرویس دهنده ها ١ درخواست در ثانیه

متوسط تولید درخواست از سوی سرویس گیرنده ها ١ در ثانیه

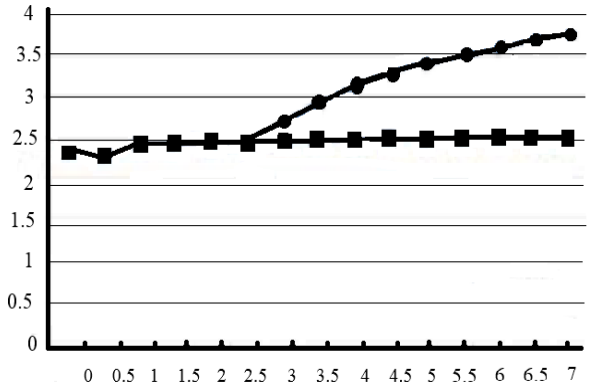
زمان مورد نیاز برای انتقال حالت ١ تا ٣ ثانیه

زمان ping دوره ای برای شناسایی خطا ٤ ثانیه

مقادیر ذکر شده برای تمامی نتایج یکسان است .برخی از مقادیر دیگر نیز وجود دارد که برای هر شبیه سازی متفاوت است و هنگام بررسی، ذکر خواهند شد .لازم به ذکر است، تابع مورد استفاده برای تولید مقادیر تصادفی از توزیع نمایی پیروی می کند[٤].

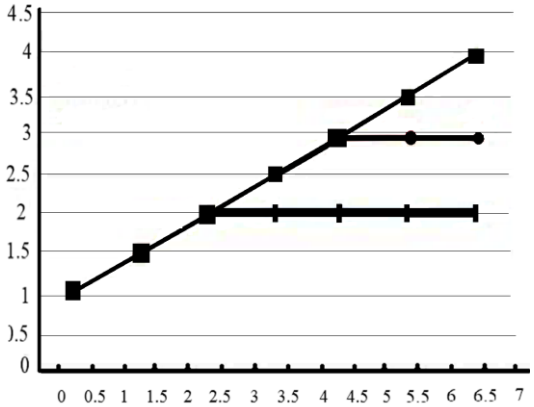
-بررسی رفتار سیستم هنگام خطا در استراتژی درخواستی

در این آزمایش می خواهیم رفتار سیستم پیشنهادی را هنگام بروز خطا مورد بررسی قرار دهیم .تعداد سرویس دهنده های این آزمایش را ١٠ و تعداد سرویس گیرنده های آنرا ٧ سرویس گیرنده تعیین کردیم . این آزمایش در دو حالت بدون خطا و با در نظر گرفتن خطا انجام شده است .خطاهای اتفاق افتاده در سیستم ٢ خطا است که هر دو در زمان ٣٠٠٠ اتفاق افتاده اند (شکل ٦-٣)[٤].



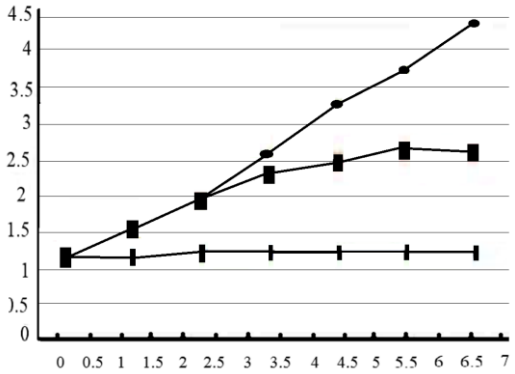
شکل شماره ٦-٣: تعداد سرویس دهنده ١٠ ، تعداد سرویس گیرنده ٧، ٢خطا در زمان ٣٠٠٠[٤]

گذردهی سیستم پیشنهادی در حالت های مختلف: این آزمایش به گذردهی سیستم پیشنهادی در حالات مختلف اختصاص دارد .قسمت اول آزمایش، گذردهی روش پیشنهادی را برای تعداد سرویس دهنده ها و سرویس گیرنده های مختف و بدون بروز خطا نشان می دهد(شکل ٦-٤)

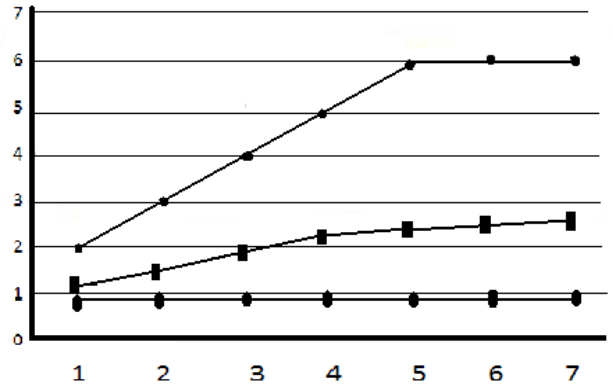


شکل شماره ٦-٤: گذردهی سیستم بدون بروز خطا برای تعداد سرویس دهنده ها و سرویس گیرنده های مختلف[٤]

شکل ٦-٥ به گذردهی سیستم هنگام بروز خطا اشاره می کند .همانطور که مشاهده می شود، هنگام بروز خطا در سیستم، گذردهی کاهش پیدا می کند.



شکل شماره ٦-٥: گذردهی سیستم با وجود خطا برای تعداد سرویس دهنده ها و سرویس گیرنده های مختلف(خطا در فاصله های زمانی ٨٠٠ ثانیه ای)[٤]



شکل شماره ٦-٦: مقایسه گذردهی، حالت های بدون خطا و خطادار برای تعداد ٦ سرویس دهنده و تعداد سرویس گیرنده های مختلف (خطا در فاصله های زمانی ٨٠٠ ثانیه ای) و مقایسه آنها با حالت بدون وجود توزیع بار[٤]

شکل ٦-٥ گذردهی سیستم را با شرایط شکل ٦-٤ نمایش می دهد با این تفاوت که در این شکل خطا نیز وجود دارد .برای اینکه تفاوت گذردهی در این دو حالت بهتر دیده شود، نمودارهای تعداد ٦ سرویس دهنده را بدون خطا و خطا دار کنار هم رسم کرده ایم . شکل ٦-٦ مقایسه این دو حالت را نشان می دهد . در این شکل همچنین، حالت بدون توزیع بار را نشان می دهد[٤].

٦-١٠ نتیجه گیری

در اینجا روشی را برای یکپارچه سازی توزیع بار و تحمل پذیری خطا بر پایه ی متوقف سازی ارائه شد که می تواند در سیستمهای توزیع شده بطور خاص در کوربا استفاده شود. همچنین در مورد ضعف های روش پیشنهادی (تکرارسازی شبه غیر فعال) مواردی مورد بررسی قرار گرفت .این روش کاملا از برنامه های کاربردی کوربا (سرویس دهنده و سرویس گیرنده) مستقل است .روش ذکر شده در سیستم عامل ویندوز پیاده سازی شده است و مستقل از سیستم عامل نیست. این روش می تواند در برنامه های کاربردی که بر پایه ی کوربا هستند و در سیستم عامل ویندوز اجرا می شوند و به یکپارچگی قوی تکرارها نیاز ندارند، برای بالا بردن گذردهی و اضافه کردن تحمل پذیری خطا استفاده شود .از نتایجی که در شبیه سازی بدست آمد، مشخص شد که این روش می تواند در کاهش زمان پاسخ و افزایش گذردهی سیستم هنگام خطا تاثیر بسزا و مثبتی داشته باشد.

منابع:

[۱] دهقان ، شاداب و حسینی ۱۳۸۸،"CORBA و بررسی امنیت در شبکه"، رویکردهای نوین در مهندسی کامپیوتر و فناوری اطلاعات ، رودسر.

[۲] کاوسیان مهدی ،۱۳۸۳،" بررسی CORBA به عنوان زیرساختار ارتباطی بسته های نرم افزاری مرتبط با اسکادا"، پژوهشگاه نیرو ایران ، نوزدهمین کنفرانس بین المللی برق، تهران.

[۳] محمودی آرزو ،۱۳۹۰ ، "بررسی مدل مولفه ای CORBA "، اولین همایش منطقه ای رویکردهای نوین در مهندسی کامپیوتر و فناوری اطلاعات ، رودسر.

[٤] نیکزاد علی و خرسندی سیاوش ،١٣٨٦، " ارائه و ارزیابی یک مدل ترکیبی برای فراهم کردن توزیع بار و تحمل پذیری خطا در کوربا"، سیزدهمین کنفرانس ملی انجمن کامپیوتر ، کیش.

[٥]I.Rose , ٢٠٠٤ , “CORBA – at - large , CITR technical white paper , CITR . Queensland ” , Australia.

[٦]j.siegol , ٢٠٠٩ , “Advanced c++ programming white CORBA”, second Edition , OMG , U.S.A.

[٧] M.SHarifi And H.Salimi , ٢٠١١ , “A Method to configure CORBA remote calls in support of ft- CORBA fault-detection echanisms ”, japan.

[٨] q.kong , g.chen , ٢٠٠٨ , “integrating CORBA and TMN environment,” proceedings of the ١٩٩٦ network operations management symposium , Kyoto , japan.

[٩] Silvano maffies , ٢٠١١ , “Constracting Reliable Distributed Communication Systems With CORBA” IEEE Communications Magazine e.

[١٠] Steve Vinoski , ٢٠٠٠ , “Distributed Object Computing With CORBA” Hewlett - Packard Company.

[١١] Z.Yang and K.Duddy , ٢٠٠٥ , “CORBA: A Platform for Distributed Object Computing (A State – of – the - Art Report on ٠MG/CORBA)”, Germany.