مفهوم تور و تور OGSA

نویسنده: حمید فطانت

فهرست

1- مقدمه

از اهداف اصلی شبکه محوری دستیابی به تور اطلاعات(دانش) در گستره همه نیرو های نظامی است. برای این منظور نیازمند یک معماری نرم افزاری پیشرفته، بروز و با قابلیت توسعه بوده که بتواند همه سرویس های لازم، زیر ساخت های اطلاعاتی و برنامه های کاربردی لازم را ارائه نماید. در حقیقت اصلی ترین هدف پروژه های شبکه محور دستیابی به توری است که بر اساس یک پورتال مشارکتی قابل احصاء و قابل دستیابی باشد. انچه که حائزاهمیت است وقوف به ضرورت پورتالی است که از دیدگاه ما از اجزای اصلی تور مزبور بوده بطوری که توسعه آن، صفات کیفی متعددی را همانند امنیت، کارائی، مقیاس پذیری، قابلیت استفاده پذیری و یکپارچگی مفهومی نتیجه می دهد. بدلیل اینکه فضای جنگ محیطی توزیع شده با حجم گسترده ای از داده ها و اطلاعات است بنابراین معماری نرم افزار مربوطه هم برای کلیت تور، هم برای پورتال مورد نظر و هم برای توسعه برنامه های کاربردی تور فرماندهی و کنترل و هر وظیفه دانه درشت دیگر اصلی ترین فعالیت در فاز مدل طراحی است. اکنون بر پایه این میان افزار و معماری نرم افزار آن می توان برای پورتال دانشی بعنوان مدخل اصلی دستیابی به این تور اقدام کرده و آن را توسعه داد.

2- مشخصه های یک تور

در سال 1998 توسط فوستر و کسلمن یک تور محاسباتی به صورت زیر تعریف گردید: یک زیر ساخت نرم افزاری و سخت افزاری که دستیابی به قابلیت های محاسباتی کامل[1] با خصوصیات قابلیت اتکاء، ارزانی، فراگیر و سازگاری را موجب می شود. بر پایه این تعریف و در ادامه آن تعریف های دیگری نیزاز تور بعمل امده است که مهمترین انها یکی "یک چارچوب نرم افزاری که ارائه کننده لایه هائی از سرویس های مختلف برای دستیابی و مدیریت منابع سخت افزاری و نرم افزاری توزیع شده است[4]" و دیگری"یک شبکه توزیع شده و گسترده از محاسبات بمنظور دستیابی به کارائی بالا، داده های ذخیره شده، دستورالعمل ها و مشارکت در محیط های جمعی[5] " است.

در سال 2004 مجددا فوستر و کسلمن تعریف زیر را برای تور ارائه نموده که امروز بعنوان تعریفی متداول و رسمی بکار گرفته می شود" یک تور یعنی اشتراکی نمودن منابع هماهنگ شده و حل مسئله در سازمان های مجازی پویا و چند منظوره است"[6]. به این ترتیب دو قابلیت اشتراک سازی منابع و همینطور حل مسئله در زمینه مجازی بعنوان شاخص های اصلی هر تور محسوب می شوند. بر این اساس فوستر و کسلمن : یک سیاهه از سه بخش را برای هر تور معرفی نمودند:

1- اشتراک منابع هماهنگ شده که هیچ گونه کنترل متمرکزی برای آن وجود ندارد.

[1] High-end

2- واسط ها و پروتکل های باز، استاندارد و همه منظوره

3- چگونگی تحویل دهی QOS (چگونه مولفه ها با هم بطور هماهنگ رفتار کرده و سرویس های ترکیب شده را تحویل می دهند).

با این سیاهه معلوم می شود که یک تور می تواند شامل مفاهیمی متعدد با مقیاس وسیع و گسترده باشد. از جمله این موارد شبکه ها، رایانه ها، سنسور ها و حافظه ها و همینطور اعتماد برای اشتراک سازی بین عامل های هوشمند است.سیاست های مبتنی بر منابع، مذاکرات و دیگر موارد مربوط به فعالیت های اجتماعی ذینفعان در این سیاهه قرار می گیرند. مورد مهم دیگری که در ارتباط با تور است حل مسئله در زمینه های مجازی است و بطور کلی مجازی سازی و سازمان مجازی از مواردی است که در هر تور باید به آن پرداخته شود. ایجاد شفافیت در یک محیط توزیع شده با مجازی سازی صورت می گیرد. تور LHC از پروژه CERN [9]نمونه خوبی از مجازی سازی ارائه نموده است.

2-1-استاندارد سازی تور

از مهمترین مسائل در توسعه تور پرداختن به استاندارد ها است که بدلیل نو بودن موضوع و فقدان حافظه لازم در جامعه ذینفعان از این مفاهیم، پرداختن به استاندارد ها یکی از مهمترین نکات در توسعه معماری تور است. در این ارتباط دو نوع تلاش وجود دارد یکی تلاش هائی که مستقیما به استاندارد سازی تور می پردازند و دیگری تلاش هائی است که بطور غیر مستقیم موجب توسعه استاندارد سازی تور می شوند:

1- تلاش های مستقیم شامل دو مورد است، یکی GGF[2] است که از مهمترین سازمان هائی است که به استاندارد های تور می پردازد[10]. دیگری OASIS یک کنسرسیوم غیر انتفاعی است [11]که هدف اصلی آن توسعه،همگرائی، و تطبیق استاندارد های e-busines است که تاثیر مستقیمی برروی استاندارد های تور می گذارند.

2- از جوامعی که بطور غیر مستقیم به استاندارد های تور می پردازند یکی DMTF [12]است که بر روی مدیریت توزیع شده تمرکز داشته و دیگری CIM [13]و WBEM [14]بوده که بر روی استاندارد های مدیریتی کار کرده و بطور غیر مستقیم به استاندارد سازی تور علاقمند است.سومین جامعه ای که بطور غیر مستقیم به استاندارد های تور می پردازند، کنسرسیوم وب جهانی یعنی W3C است که به استاندارد سازی سرویس های وب می پردازد[15] و از این جهت تلاش های آن به جامعه تور مربوط می شود.

در هر صورت اصلی ترین استاندارد سازی تور مربوط به GGF است که مستقیما به تور می پردازد. اسناد استاندارد این سازمان دارای چهار بخش زیر بوده که همگی این اسناد با کلمه GFD و یک عدد طبیعی معلوم می شوند:

[2] Global Grid Froum

1- اطلاعاتی : اسنادی که به بک ایده یا مجموعه ای از ایده ها مربوط می شود. همانند:

a. GFD.7 که یک معماری نظارتی از تور است.

b. GFD.8 که یک سند از یک مورد مطالعاتی که به سیستمی کارامد از تور می پردازد.

c. GFD.11 که یک لغت نامه از عبارات و کلمات مربوط به زمانبندی تور است.

2- تجربی: اسنادی که مبین یک تجربه تست شده یا عملی شده است.

a. GFD.5 که یک رزواسیون پیشرفته از توابع API است.

b. GFD.21 که مربوط به بهبود پروتکل GRIDFTP است.

c. GFD.24 که توسیع GSS-API است.

3- عمل جمعی: عمل یا فرایندی که جامعه عمومی آن را شکل داده باشند. در حال حاضر چهار سند عمل جمعی وجود دارد. همانند:

a. GFD.1 که دسته هائی از اسناد GGF است.

b. GFD.3 که مدیریت GGF است.

c. GFD.16 که مدل CERTIFICATE POLICY است.

4- توصیه: مربوط به اسناد توصیه ای است. همانند:

a. GFD.15 که مربوط به OGSI است.

b. GFD.20 که مربوط به GRIDFTP است(توسیع پروتکل FTP برای تور)

c. GFD.23 که مربوط به سلسله مراتبی از مشخصه های کارائی شبکه جهت سرویس ها و کاربرد های تور است.

از مهمترین استاندارد هائی که توسط GGF حمایت می شود معماری تور مبتنی بر سرویس گرائی است که مهمترین انها شامل موارد زیر است:

1- تعامل برنامه به برنامه(SOAP، WSDL و USSI)

2- اشتراک سازی داده(XML)

3- پیام رسانی (SOPA,WS-ADDRESSING)

4- بارکاری پیام رسانی(WS-Management)

5- حمایت تراکنش(WS-Coordination,WS-AtomicTransaction)

6- مدیریت کردن منابع (WSRF)

7- ایجاد امنیت WS-Security,WS-(WS-SecureConversation,WS-Federation, Trust)

8- حمایت از فراداده(WSDL,UDDL,WS-POLICY)

9- معماری سرویس های تور مبتنی بر سرویس های وب (OGSA)

10- پوشش دادن به جریان فرایند کسب و کار(BPEL4WS)

11- تریگر نمودن رخداد ها (WS-Notification)

در ادامه لازم است که به برنامه نویسی توزیع شده پرداخته شود و به انواع روش های موجود برای توسعه برنامه های کاربردی توزیع شده توجه کرده و در این راستا اصلی ترین مولفه های یک میان افزار تور فرماندهی و کنترل را شناسائی نماییم. در چنین صورتی می توان بهترین نوع مولفه ها جهت توسعه هر برنامه کاربردی از جمله پورتال مشارکتی در محیط توزیع شده را شناسائی کرد.

برای آنکه بتوان یک محیط توزیع شده و یکنواخت از مولفه های سخت و نرم نامتجانس ایجاد کرد تا بر این اساس بتوان به حل مسئله های متعدد بر روی زمینه های مجازی نائل شده می بایست به توسعه تور پرداخت. سازمان GGFبرای منظور فوق استاندارد های لازم را طی چندین مرحله ارائه نموده است:

1- سیستم های پیش از OGSA که شامل GT2[3] است.

2- بموازات GT2توسط IBM، مایکروسافت و سان، سرویس های وب معرفی گردید.

3- اولین نسخه OGSA توسط تیم گلو باس(توسط GGF4) و IBM در فوریه2002ارائه شد این استاندارد موجب پیوند تلاش های GGFو سرویس های وب شد. برای همین منظور تیم OGSA-WS در GGFشکل گرفت. OGSAتوانست برای محیط محاسباتی توزیع شده بر پایه سرویس های وب عمل کند و در نهایت توانست بر اساس پروتکل های HHTP و XMLمحیط توزیع شده سرویس گرا را توسعه دهد. این استاندارد همچنین توانست برای اولین بار سرویس های تور را معرفی نماید که سرویس های وب پویا و با مشخصات مورد نیاز برای تور هستند.

4- توسعه بعدی OGSA برای بهبود اصلی ترین مشکل آن یعنی عدم بکار گیری در زمینه هائی معین بود. OGSA فاقد ارائه واسط هائی برای چگونگی بکار گیری آن در عمل بود که برای این منظور نسخه OGSI-WS ارائه گردید که

[3] Glubus Toolkit 2

بکارگیری OGSA را در زمینه های سرویس های وب معلوم می کرد. بر اساس مشخصه تکنیکی OGSI نسخه GF3[4] معرفی گردید.

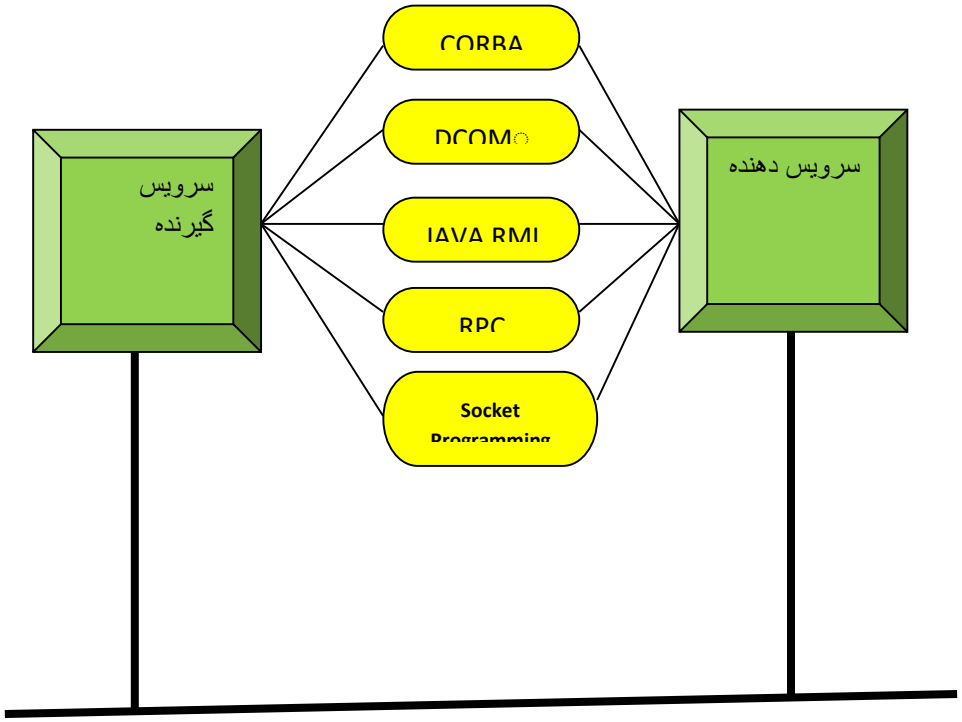
5- توسعه بعدی مربوط به حمایت از منابعی بوده که OGDI بدون انها دارای کارائی اندکی است. برای این منظور سه شرکت HP،IBM و گلوباس، WSRF را توسعه دادند که برای مدلسازی منابع کیفی به همراه سرویس های وب بکار می رفت.

ادامه این مقاله به معماری نرم افزار تور دانش مربوط می شود که در دو بخش اصلی زیر صورت می گیرد: از یک جهت به معماری های مرجع رایج می پردازیم و از این طریق به مشترکات این معماری های مرجع موجود پی برده و بسیاری از مولفه ها و موجودیت های معماری نرم افزار مورد نظر را معلوم می کنیم. دومین جهت گیری پرداختن به سرویس های وب است که استفاده از انها در تور های دانشی مورد نظر الزامی بوده بخصوص اینکه بر اساس آن ها به شفافیت و توزیع شدگی دلخواه دست می یابیم.

3- معماری های مرجع متعارف برای محاسبات توزیع شده

در این بخش به معماری های مرجع اصلی جهت توسعه محیط های محاسباتی سرویس گرا می پردازیم و ویژگی های هر کدام را مشخص نموده و در نهایت به مشترکات آنها پی برده و از این طریق مهمترین مولفه های یک معماری نرم افزار برای برنامه های کاربردی در محیط توزیع شده معلوم می کنیم.. شکل1 پنج معماری مرجع متعارف برای توسعه محاسبات توزیع شده را نشان می دهد.

4Glubus Toolkit 3



شکل 1- نکنیک های متعارف برای توسعه محاسبات توزیع شده

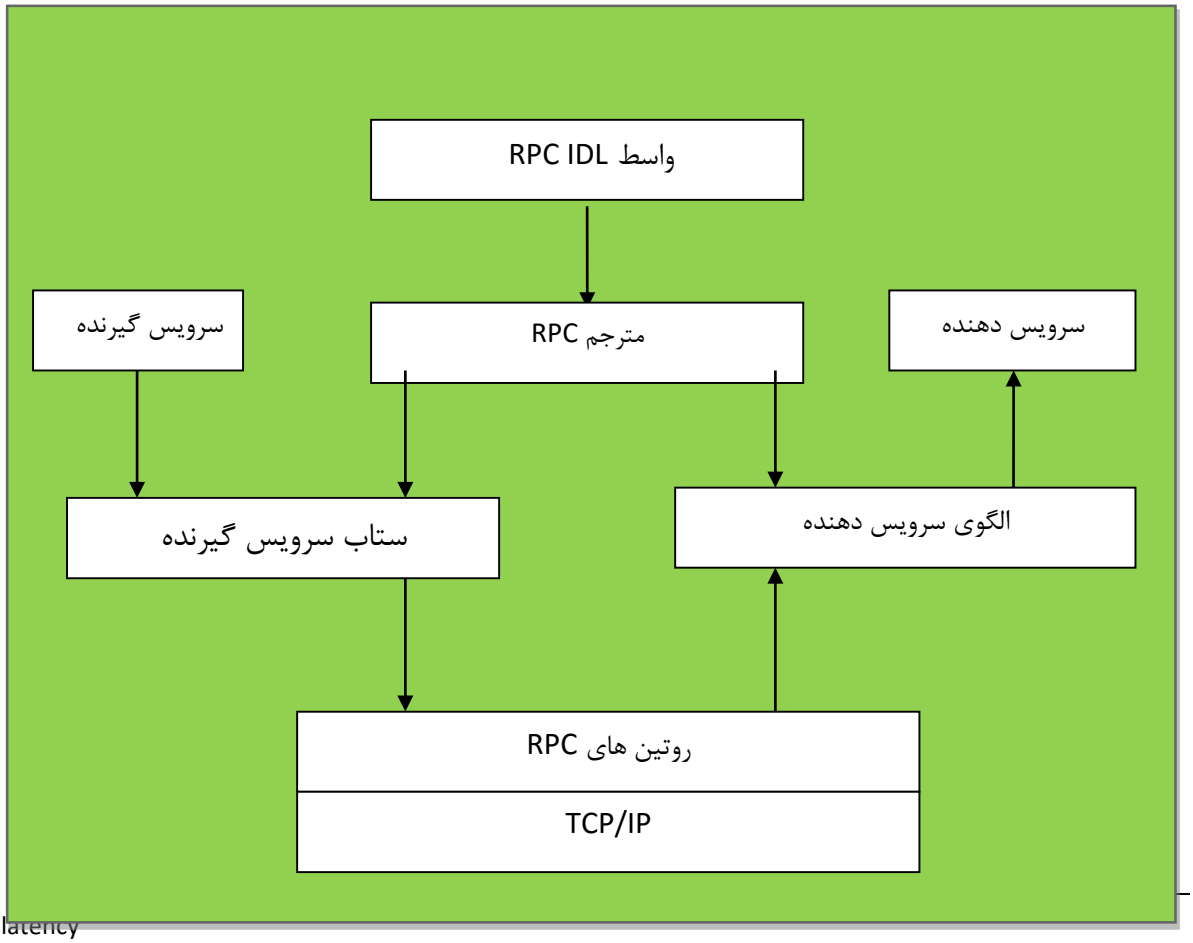
1-برنامه نویسی سوکت: نوعی برنامه نویسی سطح پایین API برای برنامه های کاربردی سرویس دهنده/گیرنده توزیع شده است. قبل از ارتباط سرویس گیرنده با سرویس دهنده می بایست سوکت ایجاد شود.پروتکل انتقال نیز TCPیا UDPدر TCP/IP است. زبان پیاده سازی می تواند مختلف و دلخواه باشد ولی برای سوکت API هر زبان می تواند تغییراتی گسترده داشته باشد. بطور معمول طرف سرویس دهنده و گیرنده سوکت از یک نوع زیان و یک بسته سوکت استفاده می کنند اما می توانند بر روی سیستم های عامل متفاوتی اجرا شوند.

چالش اصلی برنامه نویسی سوکت: چالش اصلی استفاده از تکنیک ارتباطی در سطح پایین و فقدان هر گونه تلاش برای پنهان سازی آن است. از برنامه نویسی سوکت می توان در هنگامی که با حجم گسترده ای از داده ها برای انتقال مواجه

باشیم استفاده کنیم. در حقیقت مزیت استفاده از این معماری یکی این است که پنهان سازی[5] کم بوده و دوم اینکه از پهنای باند گسترده ای در انتقال داده ها استفاده می شود.

3- RPC: برای برنامه های کاربردی سرویس دهنده/گیرنده توزیع شده است پروتکل انتقال TCPیا UDPدر TCP/IP است. ویژگی برجسته این رویکرد نقش آن بعنوان IDLیا در واقع زبان تعریف واسط است که روال های راه دور بر روی ماشین سرویس دهنده را اجرا می کند. از طریق یک واسط RPCIDL یک مترجم RPCمی تواند بطور خود کار یک استاب طرف سرویس گیرنده[6] و یک الگوی طرف سرویس دهنده[7] را ایجاد می کندو به این طریق RPC ارتباطات سطح پایین را پنهان سازی می کند. این ویژگی برجسته ای از RPC نسبت به معماری برنامه نویسی سوکت است که مجازی سازی را بعنوان شرط ضروری برای توزیع شدگی معلوم می کند.

هر چند که زبان در این تکنیک متفاوت می تواند باشد اما عموما از C استفاده می شود. شکل 2 جریان داده ای را در این تکنیک نشان می دهد. در حقیقت این شکل بیانگر توسعه یک برنامه کاربردی در یک برنامه کاربردی تحت RPC است.



[5] latency
[6] Client –side stub
[7] Server-side skeletal

شکل 2- کنترل جریان داده در یک برنامه کاربردی RPC

کنترل جریان داده ها با RPC شامل مراحل زیر است:

1- نوشتن یک واسط RPC در RPC IDL
2- استفاده از یک مترجم RPCبرای ترجمه واسط جهت ایجاد یک استاب طرف سرویس گیرنده و یک الگوی طرف سرویس دهنده
3- پیاده سازی یک سرویس دهنده
4- پیاده سازی یک سرویس گیرنده
5- ترجمه همه کد ها با یک کتابخانه RPC
6- شروع سرویس دهنده
7- شروع سرویس گیرنده

مقایسه RPCو برنامه نویسی سوکت معلوم می کند که اصلی ترین اختلاف ان ها در پنهان سازی ارتباطات سطح پایین در RPC است. هنگامی که بطور خود کار مترجم RPCواسط را به یک استاب طرف سرویس گیرنده و یک الگوی طرف سرویس دهنده تبدیل می کند در حقیقت پنهان سازی ارتباطات سطوح پایین شکل می گیرد. از ویژگی های دیگر RPC آن است که شئ گرا نبوده و از آن حمایت نمی کند.

چالش اصلی RPC: ارتباط در RPC فقط از نوع همزمانی بوده که با **الگوی فراخوانی/انتظار** شکل می گیرد. سرویس گیرنده سرویسی را فراخوانی کرده و تاهنگام دریافت آن به انتظار می ماند.

3- RMI جاوا: این یک مکانیزم شئ گرا است که توسط شرکت سان ارائه شده است. در واقع این تکنیک از زبان جاوا برای ساخت برنامه های کاربردی سرویس دهنده/گیرنده استفاده می کند. مشابه با RPC این رویکردنیز ارتباطات سطح پایین را از کاربر پنهان سازی می کند و به یک سطح مناسب ازتجرید برای حمایت از مجازی سازی در محیط توزیع شده دست می یابد. در حقیقت این تکنیک ارتباطات سطح پایین را به یک استاب طرف سرویس گیرنده و یک الگوی طرف سرویس دهنده تبدیل

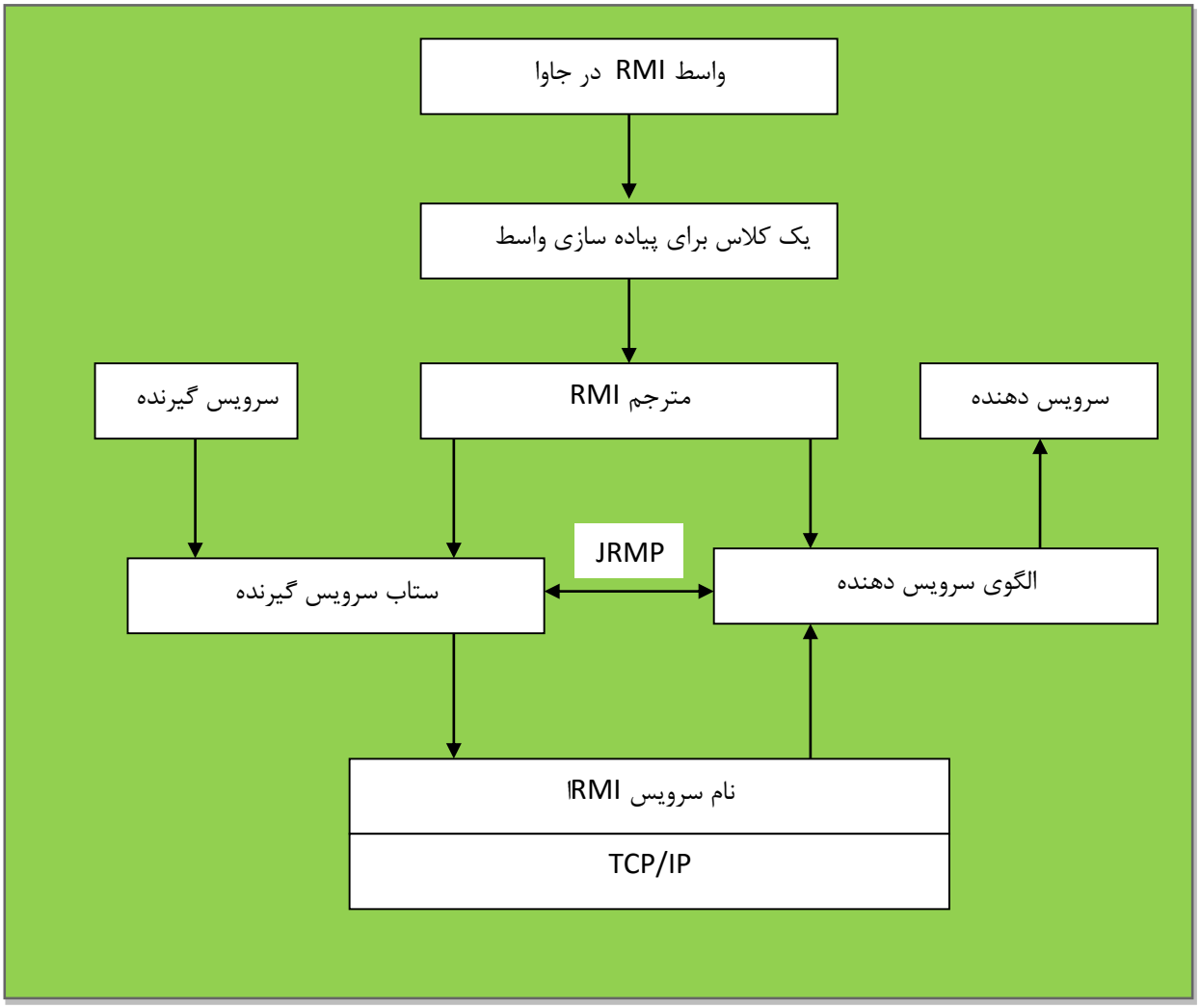
می کند. این عملیات بطور خود کار توسط ایجاد کلاس java.rmi.unicastRemoteObjectو پیاده سازی یک واسط راه دور RMIصورت می گیرد. در این معماری در سطح اجرا سه موجودیت زیر برای توسعه برنامه کاربردی RMI با هم تعامل می کنند:

1- یک سرویس گیرنده که یک متد برای یک شئ راه دور را فراخوانی می کند.

2- یک سرویس دهنده که شئ راه دور را بعنوان یک شئ متعارف در فضای آدرس خود اجرا می کند.

3- یک رجیستر شئ(RMIRegistry) که نام سرویس دهنده با همه اشیای آن را ثبت کرده است. هر گاه

شئ راه دوری در این رجیستر تبت شود در این صورت می توان به آن از طریق دست یابی به نام آن از راه دوردسترسی داشت. در این معماری هم طرف سرویس دهنده و هم سرویس گیرنده از زبان جاوا استفاده می شود اما هر دو می توانند بر روی سیستم های عامل متفاوتی اجرا گردند. هر گاه با سرویس دهنده ای ارتباط برقرار می شود

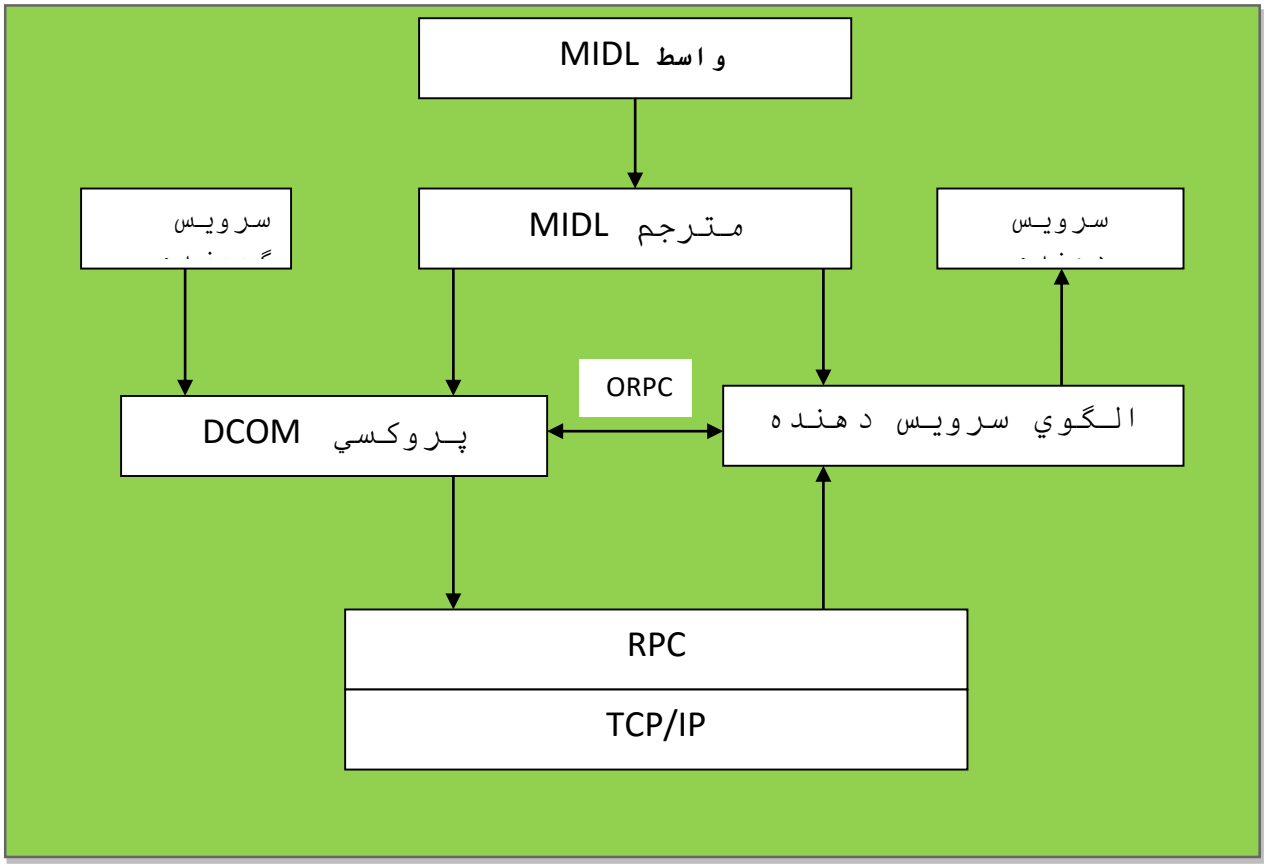


شکل 3- کنترل جریان داده RMI

در این صورت سرویس گیرنده می بایست نام میزبان سرویس دهنده یا آدرس IP آن را مشخص نماید و همینطور می بایست با استفاده از پروتکل متد راه دور جاوا یعنی JRMP به شئ راه دور روی سرویس دهنده دست یابد. شکل 3 جریان داده این رویکرد را برای ایجاد برنامه های کاربردی سرویس هنده/گیرنده نمایش می دهد.

چالش اصلی RMI جاوا: همانند RPC فقط از همزمانی در ارتباطات حمایت می کند.

4- DCOM: معماری مایکروسافت بوده که بر گرفته از COMاست. مشابه با RMI از پنهان سازی ارتباطات سطح پایین استفاده کرده که با استفاده از مترجم MIDLاین ارتباط بطور خود کار به دو بخش طرف سرویس دهنده و طرف سرویس گیرنده بنام پروکسی تبدیل می شود. DCOMاز پروتکل انتقال ORPC برای دسترسی به مولفه راه دور استفاده می کند. DCOM مستقل از زبان برنامه نویسی است این معماری در شکل 4 نمایش یافته است



شکل 4- کنترل جریان داده ها ی DCOM

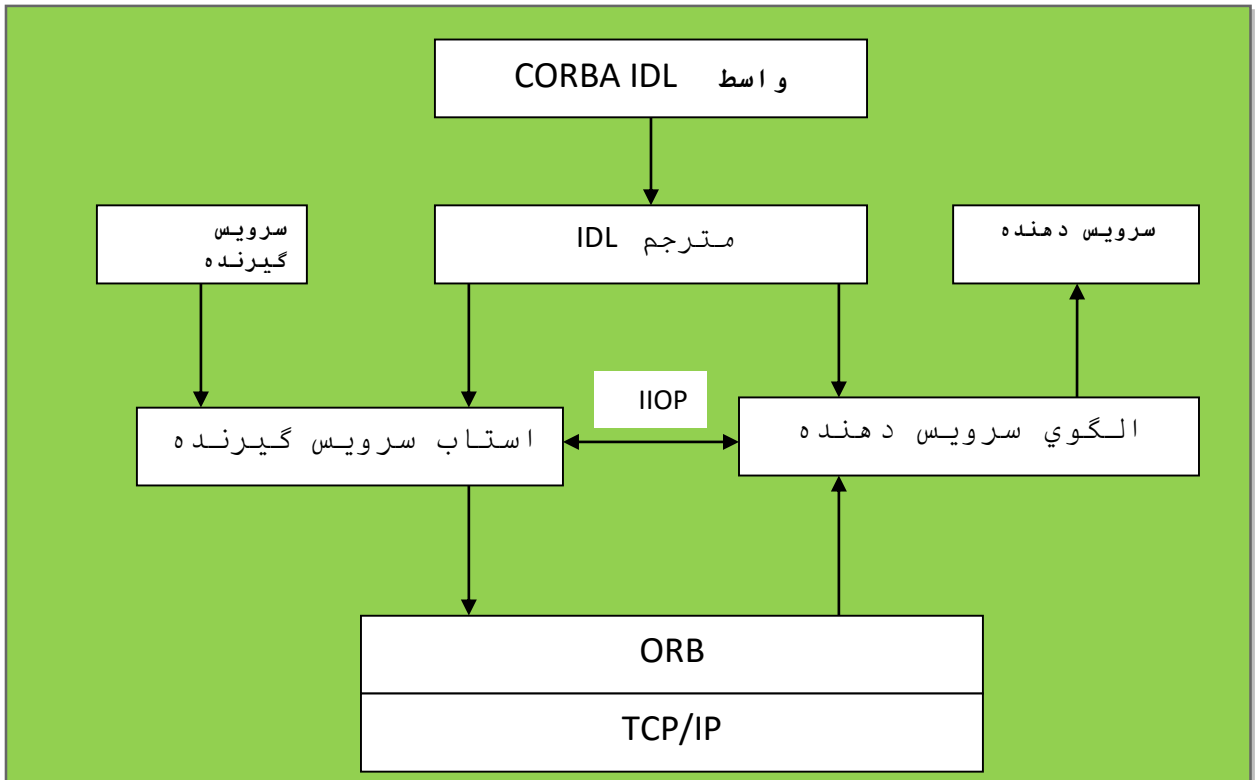
چالش های اصلی DCOM: این معماری با دو چالش اصلی روبرو است یکی اینکه علیرغم اینکه هر چند به لحاظ تئوری در هر سکوئی می تواند اجرا شود اما در عمل فقط از ویندوز استفاده می کند. این نقطه ضعف به همراه حمایت از ارتباطات همزمان بطور منحصر بفرد نقط ضعف اصلی این رویکرد است.

در این روش مراحل، اجرای برنامه های کاربردی شامل مراحل زیر است

1- نوشتن یک واسط MIDL
2- استفاده از یک مترجم midlبرای ترجمه واسط جهت ایجاد یک استاب طرف سرویس گیرنده و یک الگوی طرف سرویس دهنده
3- نوشتن مولفه COM برای پیاده سازی یک واسط
4- پیاده سازی یک سرویس گیرندهDCOM
5- ترجمه همه کد ها
6- ثبت نام مولفه COM در سرویس دهنده DCOM
7- شروع سرویس دهنده DCOM
8- شروع سرویس گیرنده DCOM

5- CORBA: این معماری در واقع یک زیر ساخت میان افزار شئ گرا است که توسط گروه OMG ارائه شده است. این نیز مشابه با RMI و DCOM از پنهان سازی ارتباطات سطح پایین استفاده کرده که با استفاده از واسط IDL(زبان توصیف واسط) این ارتباط بطور خود کار به دو بخش طرف سرویس دهنده و طرف سرویس گیرنده بنام پروکسی تبدیل می شود. CORBAبرای دسترسی به اشیاء دور CORBAاز پروتکل IIOPاستفاده می کند. هسته این معماری، ORB بعنوان بروکر درخواست شئ است CORBA هم از مارشالینگ داده ها و همینطور از فقدان آن در ارتباط بااشیاء و سرویس گیرنده استفاده می کند. **یکی از مزایای برجسته CORBA حمایت از هر دو نوع ارتباط همزمان و غیر همزمان است**. مزیت دوم CORBA استفاده از دایرکتوری پیشرفته سرویس تحت عنوان COSNaming بوده که موجب شفافیت مکان شده و در حقیقت استقلال از مکان را برای CORBA در قیاس با DCOMو RMIجاوا موجب می شود. اما

چالش اصلی CORBA: ضعف اصلی CORBAدر مقایسه با DCOMیا RMIاین است که تنها یک مشخصه ای از OMGاست و تاکنون محصولات متعددی از آن ساخته شده است. شکل 5 این معماری را به نمایش می گذارد.



مراحل اجرای برنامه های کاربردی با معماری CORBA شامل مراحل زیر است

1- نوشتن یک واسط CORBA

2- استفاده از یک متر س گیرنده و یک الگوی طرف سرویس دهنده

شکل 5- کنترل جریان داده ها ی CORBA

3- نوشتن یک شئ CORBA برای پیاده سازی واسط

4- پیاده سازی یک سرویس دهنده CORBA رجیستر نمودن شئ CORBA

5- نوشتن یک سرویس گیرنده CORBA

6- ترجمه همه کد ها

7- شروع یک سرویس دهنده نام CORBA

8- شروع سرویس دهنده CORBA

9- شروع سرویس گیرنده CORBA

3-1- استخراج برخی از ویژگی های تور دانش

با مقایسه 5 معماری مرجع(برنامه نویسی سوکت، RPC،RMI،DCOM و CORBA)جهت توسعه برنامه های کاربردی توزیع شده معلوم می شود که بدلیل عدم حمایت از مجازی سازی، برنامه سازی سوکت انتخاب خوبی نیست و همینطور هر سه معماری RPC، RMIو DCOM فقط از همزمانی در ارتباطات استفاده می کنند. بنابراین استفاده آنها نیز مناسب نیست. چنین ضعف هائی در CORBAوجود نداشته اما CORBA خود مشخصه ای از OMGاست که به همین دلیل محصولات بسیار متنوعی تحت این عنوان وجود داشته که این خود ضعف اصلی آن است. علیرغم ضعف های مزبور مشابهت های زیادی بین آنها وجود داشته که می توان به 5 جنبه زیر اشاره نموده و از آنها در توسعه تور بعنوان محیط توزیع شده بهره گرفت:

1- برای دستیابی به مولفه ها یا اشیاء راه دور ا به یک واسط نیاز است.
2- پنهان نمودن ارتباطات سطح پایین از کاربر با ایجاد کردن خود کار یک استاب سرویس گیرنده و یک الگوی سرویس دهنده از طریق تعریف واسط
3- ایجاد یک پروتکل ارتباطی و اختصاصی جهت دستیابی به اشیاء راه دور ا(همانند JRMP، ORPC، IIOP)
4- تعریف واسط در یک فرمت دودوئی برای سهولت استفاده کاربر ، کاربر در بکارگیری متد لازم برای پرس و جو معمولا دچار مشکل می شود که برای این منظور از فرمتی دودوئی و دو مقداری جهت معلوم نمودن ورودی/خروجی متد استفاده می شود که یادگیری و بکارگیری آن برای کاربران آسان تر از خود متد است.
5- ارتباط سر ویس گیرندگان با واسط های خود سست است. برای مثال تغییر دادن بخشی از سرویس گیرنده بمعنی تغییر بخش های دیگر همانند سرویس دهنده نیز هست.

پس از مطالعه معماری های مرجع جهت برنامه های کاربردی توزیع شده سرویس گرا لازم است که به رویکرد دوم یعنی سرویس های وب نیز پرداخته شود که بهمراه معماری های مرجع مزبور اساس توسعه فناوری های تور را تشکیل می دهند.

4-سرویس های وب

سروس وب موقعیتی امید بخش برای ساخت واقعی محیط محاسباتی توزیع شده ایجاد می کند. این رویکرد کاملا با رویکرد های RMI، RPC یا DCOM معایر است. هیچ کدام از این معماری ها بدون توجه به سرویس های وب نمی توانند محیط محاسباتی توزیع شده کارامد را ایجاد نمایند سرویس های وب بر اساس معماری SOAایجاد شده که در حقیقت نقش سرویس گیرنده درخواست سرویس وب بوده و نقش سرویس دهنده ارائه گر این نوع سرویس ها است. سرویس وب بر اساس پروتکل های ساده ای همانند XML و HTTP عمل کرده که دارای پشتیبان های گسترده صنعتی است و بنابراین می توان گفت که نقش سرویس های وب در قیاس با معماری های دیگر نظیر RMI، RPC یا DCOM همیشگی و با دوام است.

4-1-ویژگی های سرویس وب

یک سرویس وب دارای اتصال ضعیف بوده، محصور سازی شده و مستقل از زبان برنامه سازی وهمینطور سکو است. سرویس وب تلفیق پذیر بوده و مولفه ای در طرف سرویس دهنده است که قابلیت توصیف و انتقال داشته و می تواند توسط شبکه های کوچک، بزرگ و اینترنت مورد دستیابی قرار گیرد. استاندارد های اصلی و هسته سرویس های وب توسط OMG تعریف شده که شامل موارد زیر است:

SOAP: پروتکل ارتباطی بین سرویس دهندگان و سرویس گیرندگان است.

WSDLزبان توصیف سرویس های وب است.

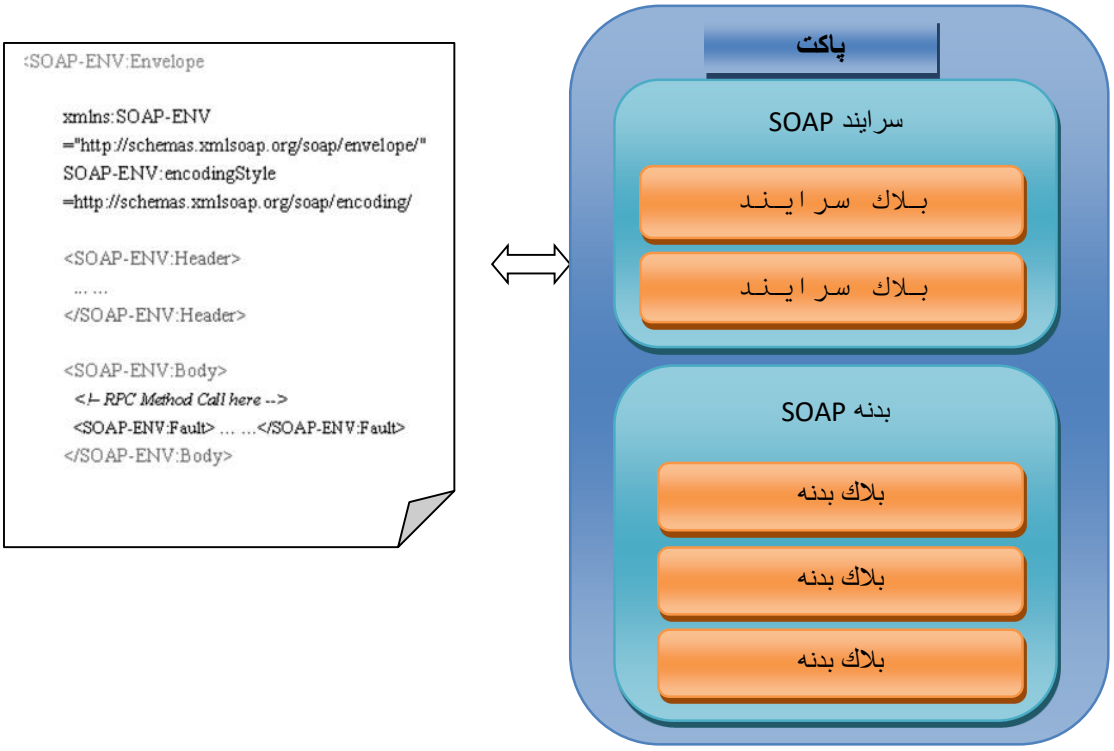
UDDI: تجمیع، اکتشاف و توصیف جهانی سرویس های وب است.

WS-Inspection: دیگر استاندارد مشخصه اکتشاف سرویس که توسط IBM و مایکروسافت معرفی شده است. در این ارتباط زبان WISL برای توصیف و کشف سرویس ارائه شده است.

شناخت استاندارد های مزبور نقشی کلیدی دربرنامه سازی تحت اینترنت و بطور کلی شبکه و تور دارد. به این دلیل در اینجا از دیدگاه نزدیک تری به انها نظر می اندازیم.

4-2- استاندارد SOAP

SOAP پروتکل ارتباطی سرویس ها ی وب بوده که سرویس دهنده /گیرنده را بهم مربوط می کند.این پروتکل ازXML برای تبادل پیام استفاده نموده که این خود براساس هر نوع پروتکلی ازجملهHTTP در انتقال یابد. بطور کلی هر گاه این استاندارد توسعه ای از پروتکل های JRMP، ORPC، IIOP/*ست*. شکل 6 پاکت پیام این پروتکل را در چهار بخش نمایش می دهد.



شکل 6- ساختار یک پاکت SOAP

چهار بخش پاکت SOAP عبارتند از:

1- فضای های نام[8] گوناگون از جمله xmlns:SOAP-ENV برای پاکت، xmlns:xsi برای الگوی XML جهت نمونه و xmlns:xsd برای تعریف الگوی XML
2- مجموعه ای از قواعد رمز نگاری برای بیان نمونه ای از انواع داده های برنامه های کاربردی
3- یک سرایند اختیاری برای حمل اطلاعات کمکی شناسائی هویت، تراکنش و پراختی

[8] NAMEsPACES

4- بدنه برای حمل اطلاعات اصلی ، هر گاه یک RPC برای پیام SOAP فراخوانی می شود، اطلاعات بدنه دارای عنصر منفردی شامل نام متد، آرگومان های آن، یک URI(مشخص کننده منبع یکنواخت) از آدرس سرویس هدف است.

نکته مهم این است که SOAP مستقل از پروتکل انتقال است و می تواند تحت هر نوع پروتکل انتقالی از جمله پروتکل های HTTP، FTP و SMTP یا حتی پروتکل های پیچیده تری نظیر JRMP، ORPC، IIOP انتقال یابد. بدلیل اینکه XML به یک استاندارد جهانی مبدل شده و بدلیل اینکه SOAPاز ان استفاده می کند و همچنین بدلیل تبادل آن بر روی هر پروتکل انتقالی می توان از پاکت های SOAP برای تبادل بین سکو های مختلف و گوناگون استفاده نموده و یک محیط توزیع شده واقعی ایجاد کرد.

4-3 –استاندارد WSDL

WSDL زبان توصیف سرویس های وب در سه وجه چه، کجا و چطوری[9] است. یک واسط WSDL همانند واسط های مورد نظر در CORBAیا DCOM اما با غنای معنائی بیشتری است. WSDL سرویس را بعنوان پورت ها یا نقاط انتهائی شبکه در نظر گرفته که با مکانیزم مبتنی بر RPC یا بر اساس تبادل پیام سند گرا[10] برای تعامل بین سرویس گیرنده و سرویس دهنده استفاده می کنند. سند WSDL در حالت اول (RPC) شامل پارامتر ها و مقادیر برگشتی و در حالت دوم(سند گرا) شامل سند XMLاست.در حالت اول ارتباط از نوع همزمانی و در حالت دوم از نوع غیر همزمانی است.

یک مشخصه WSDL شامل موارد زیر است:

1- نوع داده ها[11]: جهت ماگزییم تعامل پذیری و استقلال از هر سکو از XML XSD بعنوان نوع داده در WSDLپیشنیاز است. نکته مهم این است که این نوع قابل تغییر و قابل توسعه است.

2- پیام ها[12]: عناصر داده ای یک عملگر را در یک سرویس تعریف می کند.هر پیام شامل یک یا چند بخش است.این بخش ها شامل پارامتر های یک تابع یا یک متد در زبان های برنامه سازی معمولی است.

3- پیش الگو ها[13]: پورت ها اصلی ترین نقش را در سرویس ها دارند که همانند مفهوم کلاس در c++ یا واسط های جاوا است. یک پیام در هر پورت نقش ورودی ها و خروجی ها را ایفاء می کنند.

9, WHAT, WHERE و HOW
10 Document- oriented message exchange
11 Data Type
12MASSAGE
13 PORTTYPE

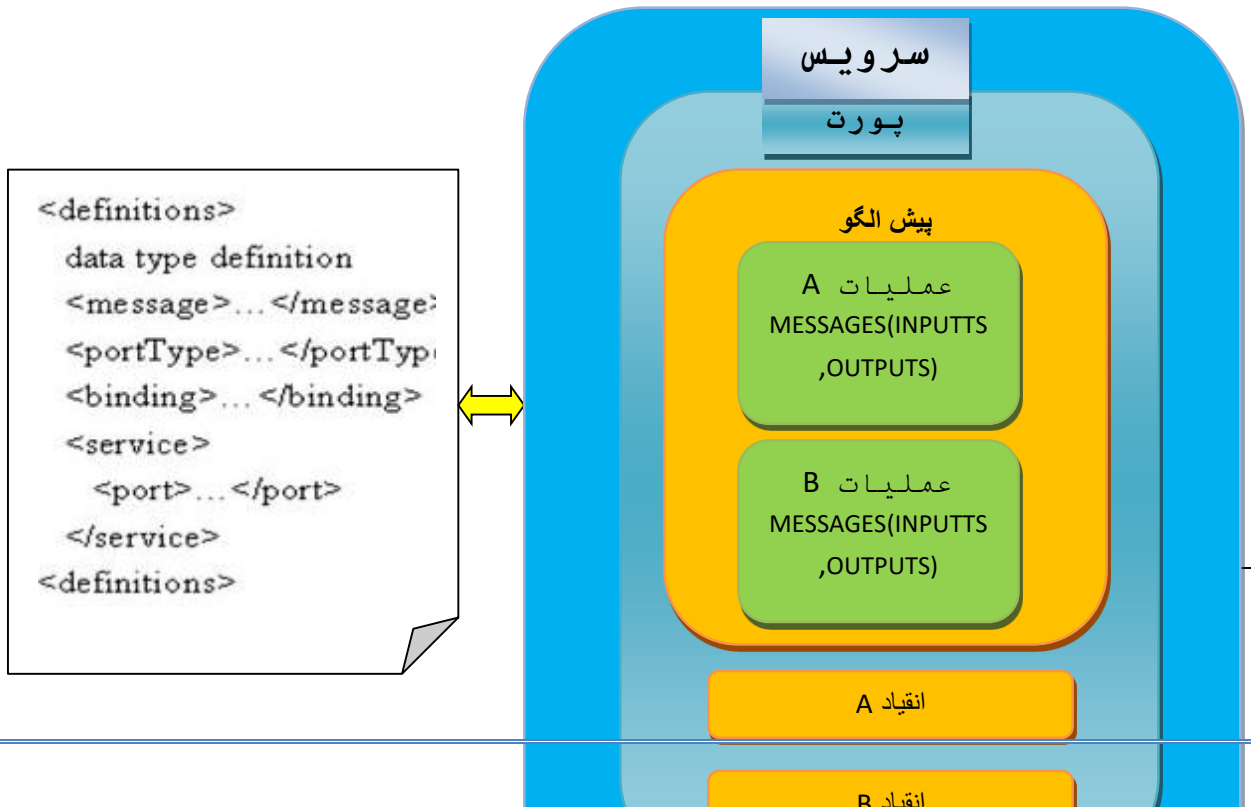
4- انقیاد[۱۴]: این ها بیان کننده یک پروتکل مشخص و فرمت داده برای عملیات و پیام های تعریف شده توسط یک پیش الگو است. یک انقیاد می تواند از نوع RPC یا سند گرا باشد و بنابراین می توان برای هر کدام از این دو عددی دلخواه را منظور کرد.

5- پورت: تعریف کننده یک سرویس منفرد که بوسیله یک آدرس گذاری منفرد جهت ایجاد انقیاد صورت می گیرد.

6- سرویس: تعیین مجموعه ای از پورت های مشخص با ویژگی های زیر

a. عدم تعامل بین پپرت ها ی یک سرویس

b. اگر پورت های یک سرویس هم نوع بوده ولی در آدرس یا انقیاد مختلف باشند در این صورت بطور معنائی می توانند جایگزین همدیگر شوند.



4-4-استاندارد UDDI

یک استاندارد صنعتی برای کشف و ثبت نام(انتشار) سرویس های وب است. یک ارائه کننده سرویس از UDDIبرای نشر سرویس خود استفاده می کند و یک سرویس گیرنده نیز از ان برای یافتن سرویس مورد نظر خود استفاده می کند. یک رجیستر UDDIهمانند سرویس تجاری CORBA یا همانند سرویس DNS برای کسب و کار برنامه های کاربردی است. بطور کلی UDDI دارای دو بازیگر است یکی تجار برای نشر سرویس و دیگری سرویس گیرندگان که برای بدست آوردن سرویس مورد نظر خود از آن استفاده می کنند. داده های UDDI به صورت زیر طبقه بندی می شوند:

1- صفحات سفید: شامل اطلاعاتی کلی از ارائه کننده های سرویس است. همانند نام، اطلاعات تماس و نظایر اینها
2- صفحات زرد: اطلاعات مربوط به توصیف سرویس وب از نقطه نظر رده های مختلف و اجازه به کاربر که بتواند به رده ها دستیابی داشته و اطلاعات مورد نظر خود را دریابد. همانند رده اطلاعات مربوط به تولید نمودن ماشین و رده اطلاعات مربوط به اقتصاد و فروش ماشین
3- صفحات سبز: اطلاعات فنی سرویس های وب که معمولا یا یک ارجاع به یک سند WSDL از سروس مربوطه همراه است. این ارجاع برای ارتباط موثر مشتری با سرویس ارائه می شود.

4-5-استاندارد WS-Inspection

این استاندارد توسط IBMو مایکروسافت ارائه شده و بیشتر از انکه یک محصول رقیب UDDI باشد، مکمل آن است. با استفاده از این استاندارد ، سرویس وب در مکان های مختلف توزیع می شود و توجه گردد که این استاندارد بر اساس توسیعی از XML نوشته می شود. و فرض اصلی در تنظیم این سند این است که سرویس گیرنده از ارائه گر سرویس با خبر و آگاه است.مشخصه WS-Inspection دو وظیفه زیر را ارائه می کند:

1- لیست ارجاع به سرویس های موجود با فرمت XML
2- مجموعه ای از قراردادها را برای تعیین آسان موقعیت اسناد WS-Inspection فراهم می کند.

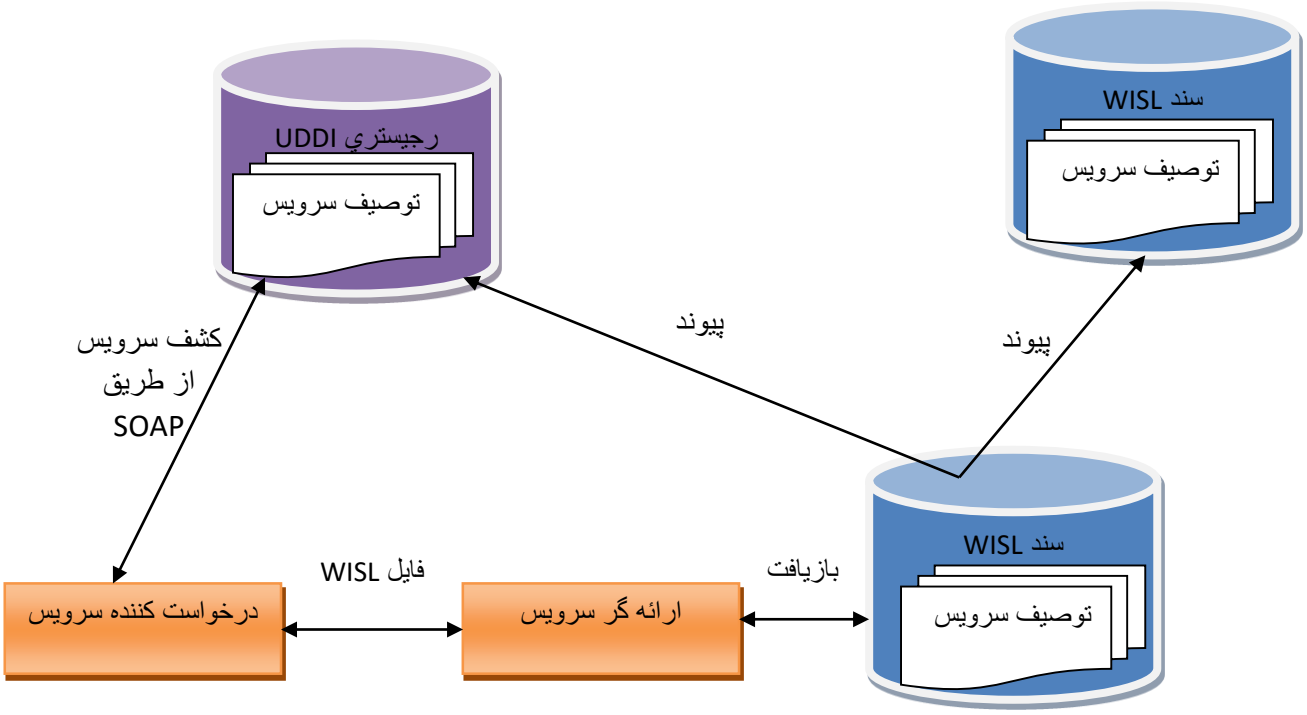
در شکل 8 نمونه ای از سند WS-Inspection نمایش یافته است.

```
<?xml version="1.0"?>
<inspection xmlns="http://schemas.xmlsoap.org/ws/2001/10/inspection/">
<service>
   <description
         referencedNamespace="http://schemas.xmlsoap.org/wsdl/"
            location="http://example.com/exampleservice.wsdl"/>
</service>
<service>
     <description
        referencedNamespace="urn:uddi-org:api">
          <wsiluddi:serviceDescription location=
                                          "http://example.com/uddi/inquiryapi">
      <wsiluddi:serviceKey>
                    2946BBO-BC28-11D5-A432-0004AC49CC1E
      </wsiluddi:serviceKey>
          </wsiluddi:serviceDescription>
      </description>
</service>
<link referencedNamespace=
"http://schemas.xmlsoap.org/ws/2001/10/inspection/"
location="http://example.com/tools/toolservices.wsil"/>
</inspection>
```

شکل 8- نمونه ای از سند WS-Inspection

4-6- کشف سرویس وب با USSI و WS-Inspection

بدلیل مکمل بودن UDDI و WS-Inspection باید از هر دو در معماری خود استفاده نمود. همچنانکه ذگی و بلینگر[15][10] معلوم نموده اند هر کدام از مشخصه های UDDI و WS-Inspection به مجموعه ای از مباحث جداگانه از کشف و انتشار سروییس های های وب می پزدازند. به این ترتیب یکی از وظایف اصلی هر معماری ایجاد توازن بین این دو بوده بطوریکه بتوان از هر دو استفاده موثر کرد. UDDI نرخ بالائی از وظیفه مندی را بدست داده ولی در عین حال موجب افزایش هزینه پیچیدگی می شود. بعکس WS-Inspection در عین اینکه وظیفه مندی اندکی را ارائه می کند اما سرباز زیادی ایجاد نمی کند. به این ترتیب استفاده از هر دو در جای خود می تواند بهترین گزینه باشد فقط می بایست محل استفاده مناسب از هر کدام تشخیص داده شود. هر گاه از UDDI درخواست مشخصه سرویس داشته باشیم در این صورت مشخصه WS-Inspectionنیز ارائه می شود و بعکس با درخواست از WS-Inspection توسط WSIL(زبان WS-Inspection) رجیستری UDDI قابل دسترس می شود. به هر صورت استفاده از هر دو بهترین گزینه ای است که برنامه های کاربردی می توانندداشته باشند. در هنگامی که نتوان از وظیفه مندی های UDDI بهره گرفت(داده ها بطور توزیع شده ذخیره شده باشند) استفاده از WS-Inspection می تواند بهترین گزینه باشد و در هنگامی که داده ها بطور متمرکز ذخیره شده باشند در این صورت استفاده از وظیفه مندی های UDDI بهترین گزینه است. شکل 9 استفاده همزمان از هر دو را نشان می دهد.



شکل 9- استفاده از هر دوی UDDI و WS-Inspection بعنوان کشف سرویس

[15] Nagy

5-پیاده سازی سرویس های وب

در حال حاضر سرویس های وب گوناگون و مختلفی وجود دارد. برای پیاده سازی این نوع سرویس می بایست سه جنبه زیر را در نظر گرفت:

1- یک مدل برنامه نویسی: چگونگی برنامه نویسی سرویس گیرنده برای دسترسی به سرویس مربوطه و چگونگی پیاده سازی سرویس وب و چگونگی توسعه جنبه های مختلف SOAPاز جمله سرایند ها و ضمائم آن
2- یک مدل چیده مان: چارچوب مورد استفاده برای چیده مان سرویس وب ارائه شده و همینطور ارائه فایل wsdd به معنی توصیف گر چیده مان سرویس وب جهت نگاشت پیاده سازی سرویس به پیام های SOAP
3- یک موتور SOAP: برای دریافت پیام های SOAP و دسترسی به سرویس وب پیاده سازی شده

اکنون بر اساس واقعیات مربوط به سرویس های وب چارچوب های مختلفی برای پیاده سازی برنامه های کاربردی بر اساس سرویس های وب وجود داشته که شناخت آنها یکی از مهمترین وظایف در نوع معماری نرم افزار تور خواهد بود. از بین این نوع چارچوب ها سه تای آنها چه به لحاظ قدمت و چه به لحاظ کاربرد بیشتر مطرح هستند. این سه شامل J2EE، .Net و Apachi Axis است.

5-1- چارچو ب J2EE

این چارچوب توسعه برنامه های کاربردی مبتنی بر سرویس های وب را برپایه جاوا قرار می دهد در حال حاضر این چارچوب برای ساخت وب سایت های متعارف، مولفه های نرم افزاری و برنامه های کاربردی بسته شده ورد استفاده قزار می گیرد. این چارچوب شامل API های زیر جهت توسعه سرویس های و.ب است:

1- JAXP که پردازش اسناد XML بر اساس پارسر های گوناگون است.

2- JAXB که معماری جاوا برای انقیاد نمودن XML است. پردازش اسناد XML بر اساس الگوی بدست آمده از کلاس های مولفه JavaBeans است.

3- JAX-RPC یک استاندارد برای RPC است که در واقع API جاوا برای RPC های مبتنی بر XML است.

4- JAXM که API جاوا برای پیام رسانی XML و SOAP بوده که با ضمیمه نمودن API در جاوا(SAAJ) صورت می گیرد. نتیجه نهائی ارسال پیام SOAP بر روی وب با روشی استاندارد می شود.

5- JAXR که API جاوا برای رجیستر نمودن XML است. مو.جب ارائه یک روش استاندارد برای تعامل با رجیستر های کسب و کار UDDI می شود.

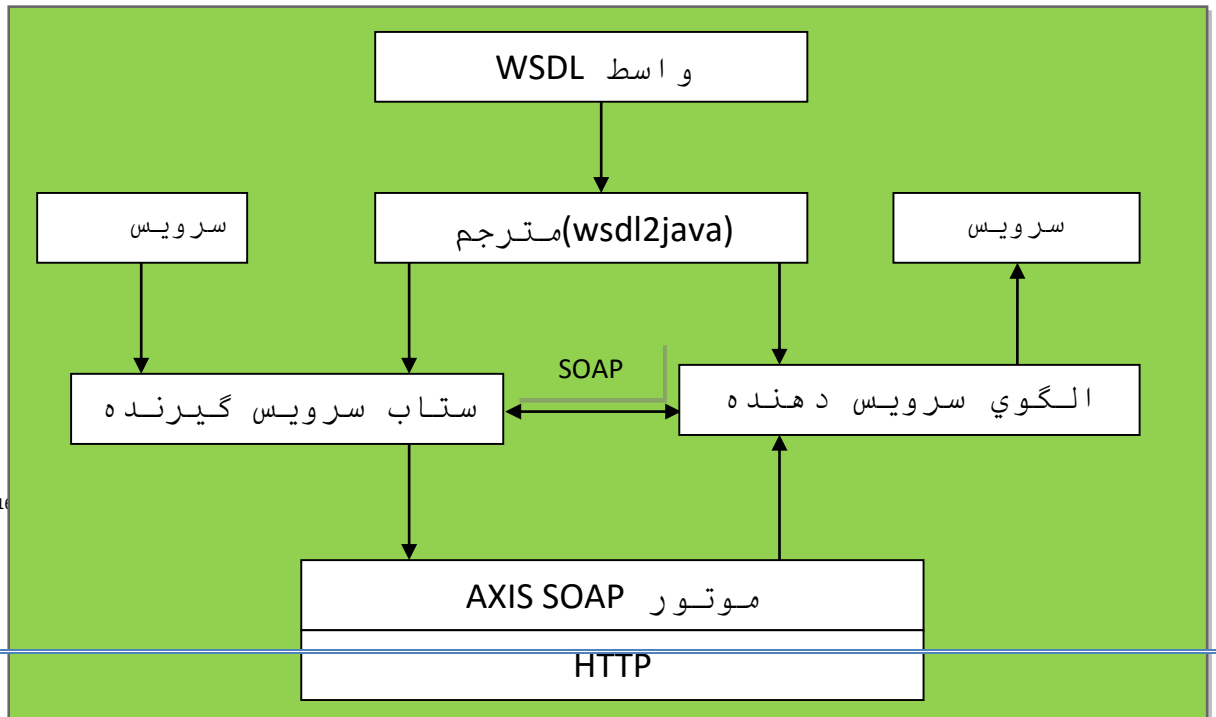
شکل 10 نشان دهنده یک جریان کنترل داده برای دستیابی یک سرویس گیرنده به سرویس وبی است که توسط J2EE JAX-RPC. ارائه شده است.

شکل 10- جریان کنترل داده برای درخواست سرویس وب از J2EE JAX-RPC

2-5-چارچوب Apachi Axis

این چارچوب در واقع نوعی موتور SOAP است. که برای تبادل پیام های SOAP بین سرویس دهندگان/گیرندگان استفاده می شود. آن همچنین برای حمایت از عملگر های WSDL مورد استفاده قرار می گیرد. به این معنی که JAVA2WSDL می تواند برای ایجاد اسناد WSDL از واسط جاوا مورد استفاده قرار گیرد. و همین JAVA2WSDL می تواند استاب طرف سرویس گیرنده و الگوی طرف سرویس دهنده را بر اساس سند WSDLایجاد نماید.اما AXIS برای کشف و انتشار سرویس نمی تواند بکار گرفته شود ولی برای این منظور از محصول UDDIWSDL شرکت IBM استفاده می کند. شکل یک جریان داده ای برای دستیابی به سرویس را توسط یک درخواست از سرویس گیرنده نشان می دهد. برای توسعه یک برنامه کاربردی بر اساس معماری AXIS باید مراحل زیر را طی کرد:

1- نوشتن یک واسط جاوا
2- استفاده از JAVA2WSDL برای ترجمه واسط نوشته شده و بدست آوردن واسط WSDL
3- استفاده از WSDL2JAVA برای ترجمه واسط WSDLبه دو بخش استاب سرویس گیرنده و الگوی سرویس دهنده
4- نوشتن یک سرویس برای پیاده سازی واسط WSDL
5- نوشتن یک سرویس گیرنده
6- ترجمه همه کد ها با javac
7- نوشتن یک فایل wsdd(فایل توصیف کننده توسعه سرویس وب)برای چیده مان سرویس ایجاد شده در وب سرور جاکارتا تام کت[۱۶]
8- شروع تام کت
9- شروع سرویس گیرنده برای دستیابی به سرویس



5-3- چارچوب .NET.

یک سکوی ماکروسافت جهت ساخت برنامه های کاربردی مبتنی بر سرویس های وب است. مشابه با J2EE ازمشخصه WSDL برای سرویس ، شکل 11- جریان کنترل داده ها برای درخواست سرویس وب از APACHI AXIS ی وب استفاده کرده و از مولفه طرف سرویس دهنده برای نگاشت عملگر سرویس های وب به متد شئ COM که در واسط WSDL و همینطور فایل WEMl تعریف شده است. انتشار سرویس های وب از طریق فایل DISCO یا مشخصه UDDIصورت می گیرد.NET یک کیت توسعه نرم افزار UDDI(SDK) برای کشف سرویس های وب ارائه کرده است.با درخواست یک سرویس وب .NET. دارای سه انتخاب زیر است:

1- استفاده از کلاس های توکار .NET. از پیام های SOAP
2- ساخت یک شنونده[17] سرویس وب بطور دستی با استفاده از MSXML(پارسر XML ماکروسافت) یا ASP(صفحات سرویس دهنده فعال) یا ISAPI(واسط برنامه نویسی برنامه های کاربردی سرویس دهنده ها ی اینترنت)
3- با استفاده از جعبه ابزار SOAPمایکروسافت نسخه 2[18] برای ساخت شنونده های سرویس های وب که با سرویس های پیاده سازی شده توسط COM تعامل می کنند. با استفاده از این جعبه ابزار می توان یک استاب طرف سرویس گیرنده از واسط WSDL ایجاد نموده که سرویس گیرنده قادر به تعامل با سرویس را دارد.

5-4- استفاده سرویس های وب در تور

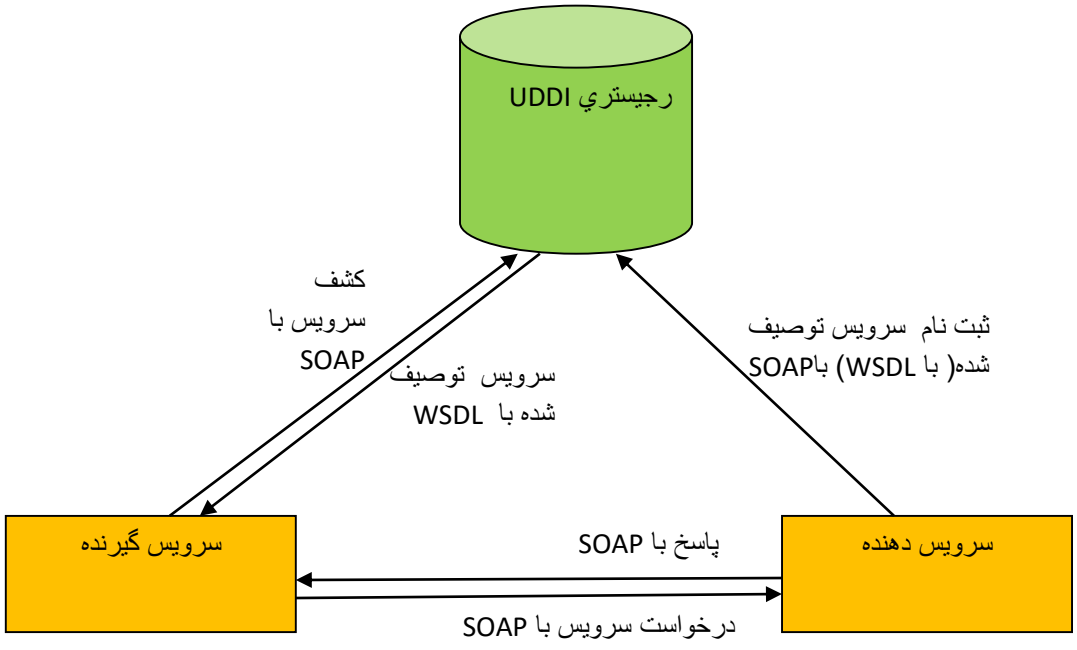
می توان سرویس های وب را استاندارد های باز مبتنی بر XML تعریف نموده که موجب توسعه برنامه های کاربردی توزیع شده در محیط های محاسباتی ناهمگون می شوند. و مهمتر از هر چیزی سرویس های وب مستقل از مکان، سکو و زبان برنامه نویسی هستند. آنها می توانند توصیف شده، انتشار یافته و بطور پویائی با زبان WSDL کشف شود و همه فناوری های متناظر با سرویس های وب می توانند یک سکوی امید بخش برای محیط های محاسباتی ناهمگون باشند. شکل 12 یک معماری از سرویس های وب را نشان داده و نحوه استفاده از این سرویس ها را نشان می دهد.

[17] LISTNER
[18] Microsoft SOAP Toolkit 2.0

اکنون مهمترین سئوال برای ما نحوه استفاده از سرویس های وب در تور های فضای جنگ است. در حقیقت پاسخ به این سئوال فرم اصلی تعامل بین گره های تور را به نمایش می گذارد. یک سرویس وب برای توسعه هر تور یک فرصت ایده ال است که بر اساس آن می توان همه نیازمندی های اصلی ان را بدست آورد. از جمله این نیازمند ی ها شامل موارد زیر است:

1- استفاده از مترجم هائی که ارتباطات در سطوح پایین را پنهان سازی می کنند موجب مجازی سازی شده که این بهمراه توزیع شدگی برای هر تور الزامی است.

3-استفاده از سرویس های وب موجب استفاده از فناوری هائی شده که در سطح جهانی استاندارد بوده که این خود موجب استخراج ابزار ها و سرویس های گوناگون و متنوعی می گردد. نکته مهم این است که با ابزار هائی نظیر پردازش های WSDL می توان این نوع محیط ها را بهم وابسته نموده و یک تور واقعی ایجاد نمود.بکارگیری ابزار هائی از قبیل WSIF، سیستم های جریان کاری و برنامه های جنبی WSDL و همینطور محیط های میزبان سرویس های وب (همانند .NET-) J2EE,Apachi AXIS می توانند این نیازمندی را ایجاد نمایند.



شکل های وب

OGSA-6

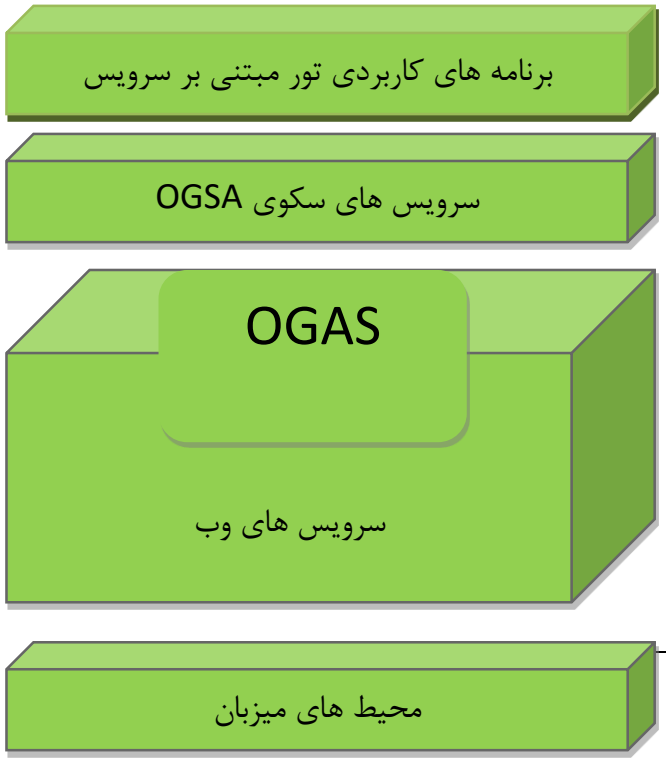
یک استاندارد دو فاکتو برای ساخت سیستم های تور سرویس گرا است.GGFتلاشی جهانی برای کامل نمودن این استاندارد برای تور شروع کرده و در حال انجام است. این جامعه برای ایجاد تور از فناوریهای سرویس وب با تعدادی استثنا و در حقیقت توسعه استفاده کرده است. در حقیقت سرویس های وب برای اینکه بعنوان سرویس های تور مورد استفاده قرار گیرند بایست از سه جهت تغییر نمایند:

1- در هر تور طبیعت هر سرویس پویا و انتقالی است. اینها رفت و آمد نموده و به منابع مختلف پخش و توزیع[19] شده و بعنوان وضعیت های سیستم در حال تغییر هستند. در هر صورت هر سرویس در تور نیازمند به واسط هائی است که ایجاد و حذف تغییرات را مدیریت نموده و مدیریت جرخه حیات را بازیابی نماید.

2- سرویس های تور دارای صفات و داده هائی برای توصیف هر وضعیت سیستم هستند. این حالت مشابه با مفهوم شئ در شئ گرائی است که هر شئ دارای رفتار و داده هائی است به این ترتیب اگر از سرویس های وب بعنوان سرویس های تور استفاده می شود می بایست بتواند هر وضعیت را در خود بازتاب دهد.

3- سرویس گیرنده باید بتواند علایق خود را در سرویس ها معلوم کرده و تصدیق کند که بدلیل تغییر سرویس در محیط پویای تور نیازمند به عملیات فراخوانی رو به عقب[20] از سرویس دهنده ها به سرویس گیرنده ها است.

در شکل 13 برنامه های کاربردی یک تور OGSA که از سرویس های وب بعنوان سرویس استفاده می کند نشان داده شده است.



[19] DISPATCH
[20] CALL-BACK

شکل 13- ساخت برنامه های کاربردی تور متقاضی OGSA با OGSI

سرویس های OGSAترکیبی از دو بخش زیر است:

1- سرویس های سکوی OGSA که شامل شناسائی و اختیارات کاربر، تحمل عیب مجوز کار، نظارت و دستیابی به داده ها

2- سرویس های هسته که شامل ایجاد و حذف سرویس ، مدیریت چرخه حیات آن، رجیستری سرویس، کشف و اعلان آن. است.

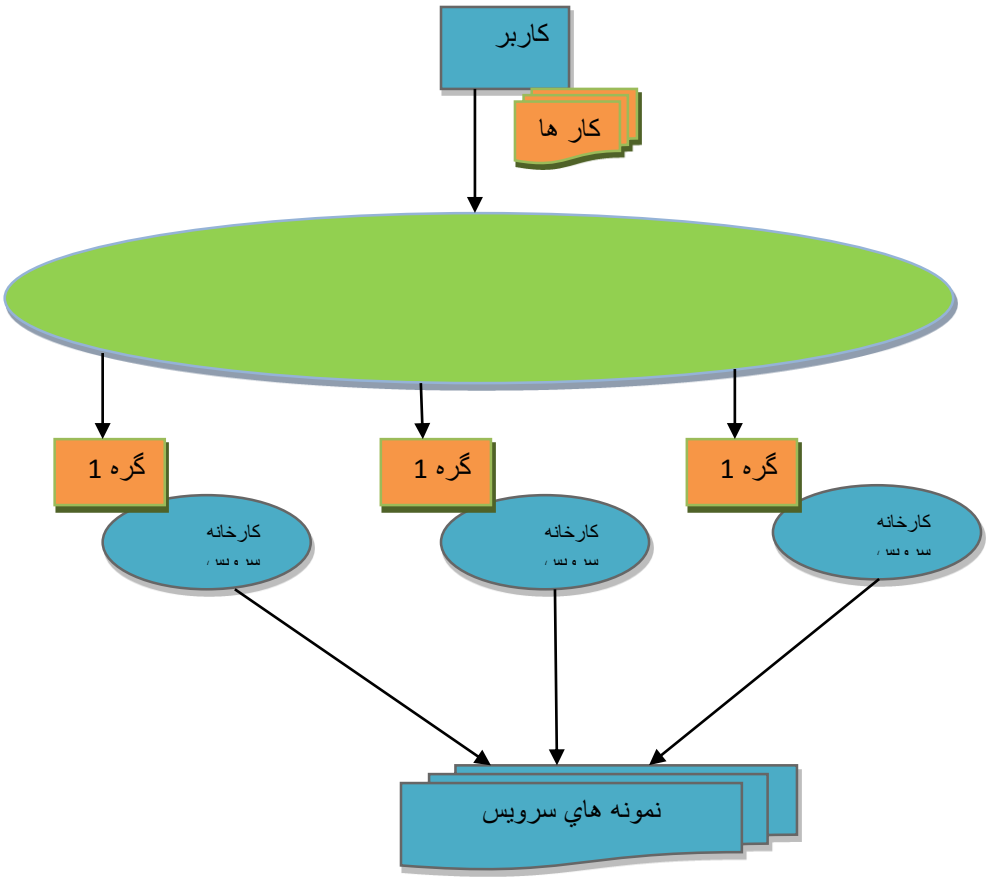
OGSA شامل واسط سرویس بوده که سرویس هائی نظیر Gridservice,Factory,Resistration,HandelResolver,Notification را برای حمایت از سرویس های هسته بکار می برد. همچنین OGSA برای حمایت از پویائی سرویس ها، حمایت از تغییرات و وضعیت های مختلف از مفهوم نمونه سرویس و داده سرویس استفاده می کند. OGSA جنبه های متنوعی را که مربوط به واسط سرویس بوده را تعریف نموده ولی هنوز نمی تواند برای پیاده سازی آن حمایتی بعمل نمی آورد. اینکار وظیفه OGSI است.که مشخصه تمنیکی همه سرویس های تور هسته در OGSA را برای پیاده سازی در بر دارد.

6-1- مفهوم نمونه سرویس[۲۱]

یکی از اصلی ترین معضلات استفاده از سرویس های وب بعنوان سرویس های تور افتراق انها به لحاظ پویائی است. یک سرویس تور ماهیتا پویا بوده و تغییر آن در هر وضعیتی از تور احتمال دارد. از طر ف دیگر سرویس های وب مانا بوده و بخصوص اینکه این مانائی در رجیسیتری UDDI جهت انتشار ثبت نیاز خواهد بود. برای این منظور سرویس های تور از مفهوم نمونه سرویس استفاده نموده که بنام کارخانه سرویس[۲۲] نامیده شده و به این ترتیب مانائی در تجریدی از نمونه های سرویس پدیدار می شود. در چنین صورتی سرویس گیرنده می تواند تقاضای کارخانه کند که نمونه های سرویس برای او عرضه شده و همچنین مشتریان مختلف می توانند به نمونه های یکسانی از یک سرویس دستیابی داشته باشند. GSH و GSR به ترتیب به معنی حمایت از سرویس تور و ارجاع به سرویس تور بوده که اولی یک URI کلی است که متمایز کننده یک نمونه سرویس تور از نمونه های دیگر است. این مفهوم در حقیقت بیانگر و مشخص کننده نمونه سرویس تور است. این نوع سرویس ها ممکن است که در طول حیات خود بروز شوند و بعنوان مثال از نسخه پروتکلی خاص حمایت کنند. GSH نمی تواند بیانگر تغییرات فوق بوده که برای این منظور از GSR استفاده می شود. GSR تغییرات در طول حیات نمونه سرویس را نشان می دهد. زمان اعتبار یا انقضاء سرویس را نشان می دهد و هچنین OGSA برای حصول GSR بروز شده

[21] SERVICE STANCE
[22] SERVIUCE FACTORY

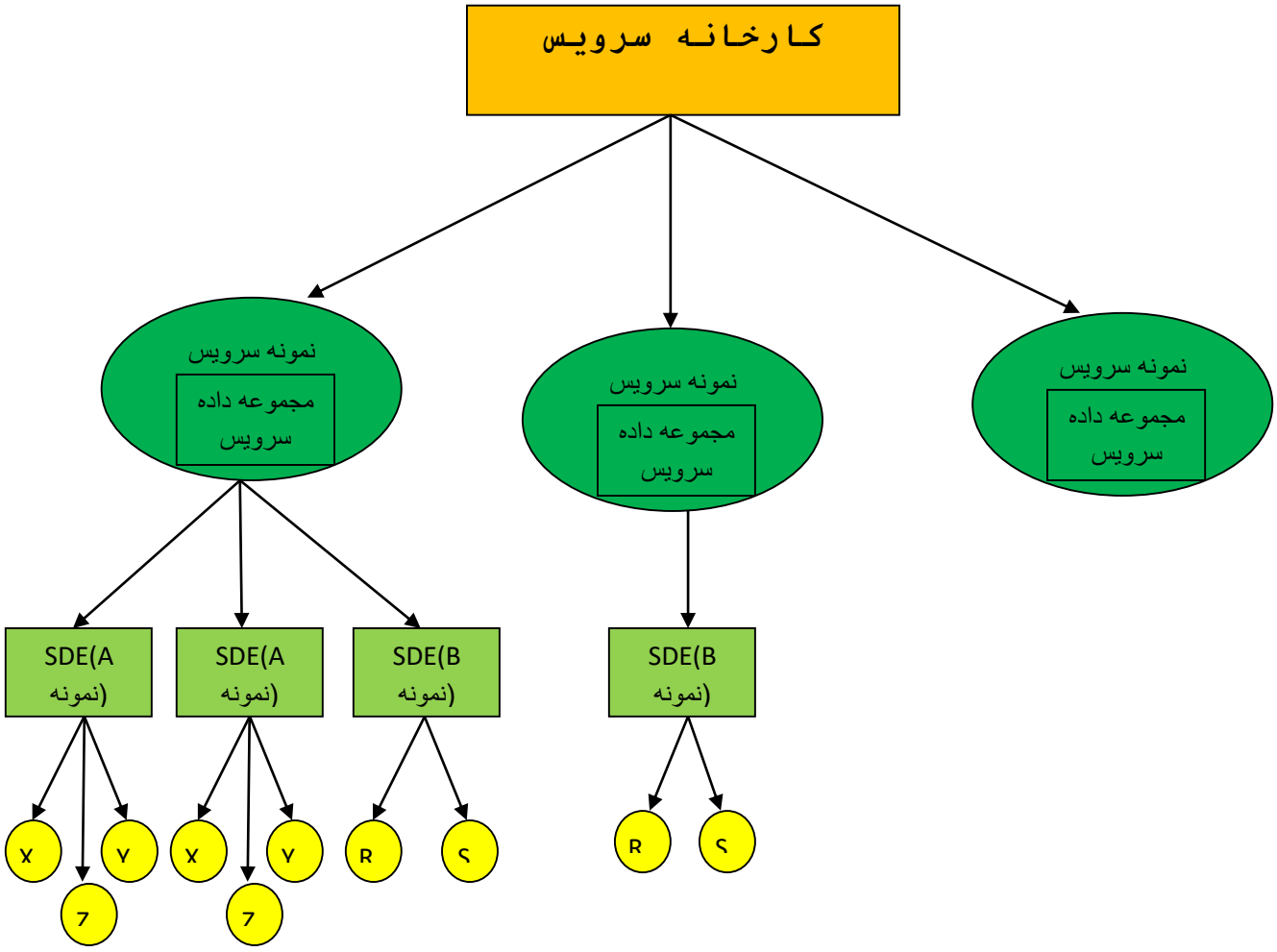
مکانیزم های های خاصی را نگاشت می کند. شکل 14 نشان می دهد که مجوز کار می تواند یک یا چند نمونه سرویس تورا نتجه دهد که اینها از کارخانه های سرویس اجرا شده بر روی سه گره ایجاد می شوند. پیاده سازی هر سه سرویس مستقل از سکو، محل و زبان پیاده سازی است.



شکل 14- یک مجوز کار کاربر با نمونه های سرویس تور چند گانه متناظر

6-2-معانی داده های سرویس

در OGSA، جدای از متد ها ، داده سرویس[۲۳] همراه با هر نمونه سرویس تور است. که بعنوان دسته ای از عناصر محصور سازی شده XML تحت عنوان SDE[24] است. اینها برای توصیف نمونه سرویس و وضعیت آنها بکار می روند. درست در مقابل نمونه سرویس که حالت گرا نبوده داده های سرویس حالت گرا[۲۵] و درون گرا [۲۶] است.یک سرویس گیرنده با استفاده از متد FindServiceData() تعریف شده در پیش الگو GridService جهت پرس و جوو بازیافت داده سرویس متناظر با سرویس تور ثبت شده در رجیستری استفاده می کند. شکل 15 یک سلسله مراتب را برای تعیین ارتباط بین مفاهیم کارخانه سرویس، نمونه های سرویس و داده سرویس نمایش می دهد.



شکل 15- ارتباط بین مفاهیم کارخانه، نمونه، داده های سرویس و SDE

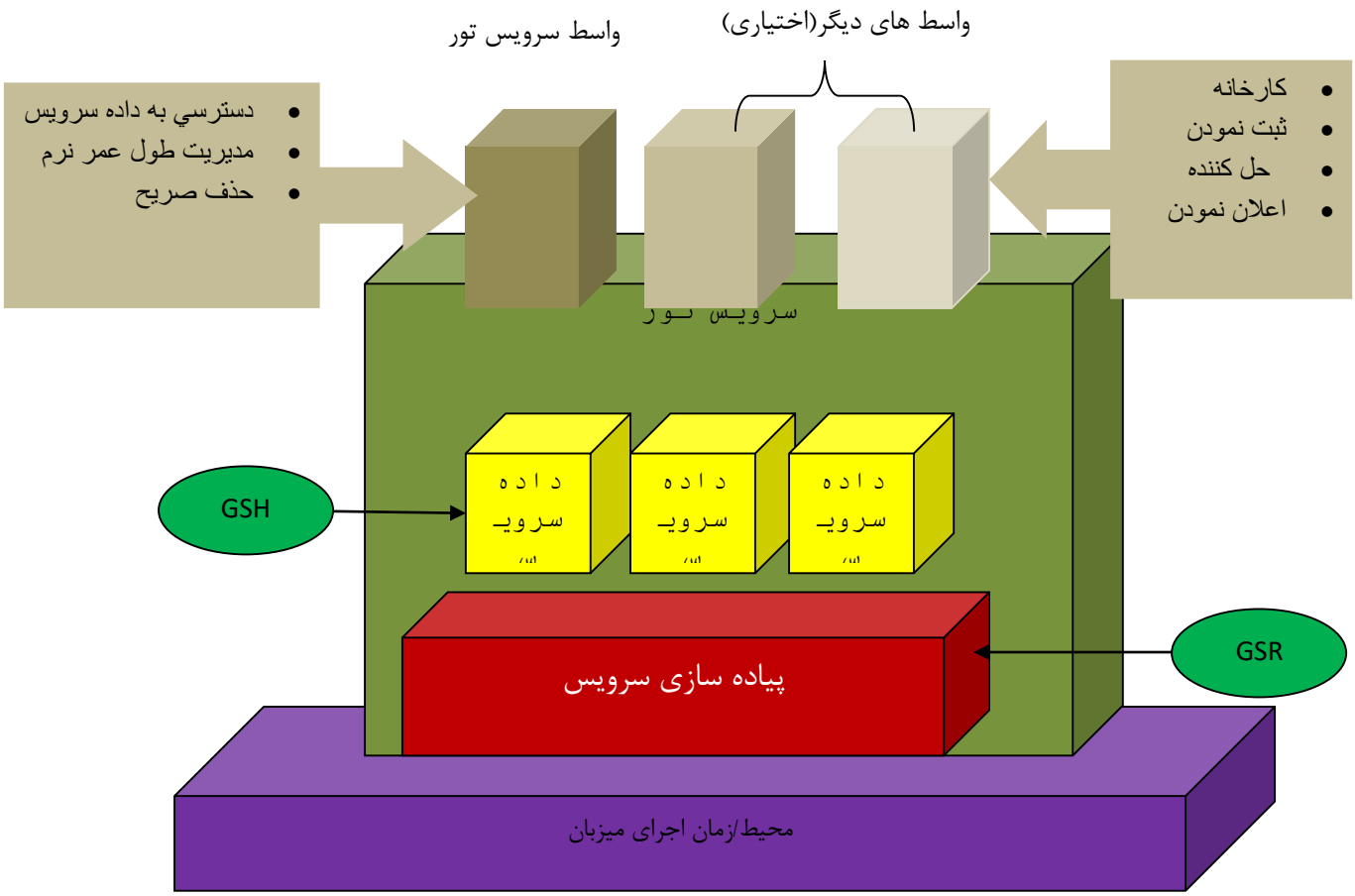
- در این شکل هر SDE شامل نمونه های X,Y,Z,R,S است

[23] SERVI
[24] Servic
[25] Statefu
[26] Introspected

3-6-پیش الگودر OGSA

در OGSAبرای تعریف سرویس های تور از پیش الگوکه توسیعی از پیش الگو در WSDL است استفاده می شود.در OGSA واسط سرویس تور می بایست برای هر سرویس تور پیاده سازی شود. OGSA واسط های زیر را در حال حاضر حمایت می کند:

پیش الگو سرویس تور، کارخانه، ثبت نام کردن، حمایت از حل کننده[۲۷] ، منبع اعلان[۲۸] / مقصد اعلان[۲۹] سرویس های تور در OGSAهستند. شکل 15 ساختار سرویس های تور را در OGSA به نمایش می گذارد



[27] HandleResolver
[28] NotificationSource
[29] NotificationSink

شکل 16- ساختار سرویس تور در OGSA

پیش الگوی سرویس تور: این مشابه با مفهوم کلاس در شئ گرائی است که توسط C++ یا جاوا ساخته می شوند. همانطوریکه رفتار های اصلی(متد) هر کلاس محصور سازی می شوند در اینجا نیز سه متد توسط پیش الگو سرویس تور محصور می شوند. این سه عبارتند از ()FindServiceData، ()SetTerminationTimeو ()Destroy که برای کشف، درون گرائی و مدیریت چرخه حیات بکار می روند.

پیش الگوی کارخانه : یک کارخانه یک سرویس تور مانا است که پیش الگو کارخانه را پیاده سازی می کند. این پیش الگو می تواند برای ایجاد نمونه های سرویس تور انتقالی از متد ()createService استفاده کند.

پیش الگو **حل کننده:** این کلاس برای حل GSH و GSR با استفاده از متد ()FindbyHandle صورت می گیرد.

پیش الگو **ثبت نام:** یک رجیستری یک سرویس تور جهت پیاده سازی پیش الگو ثبت نام برای حمایت از کشف سرویس توسط مدیریت نمودن دسته های GSHو سیاست های پلیسی متناظر است. دو متد ()RegisterService و ()Un RegisterService دو متد این کلاس به ترتیب برای سرویس ثبت نام و حذف آن است.

پیش الگو منبع اعلان/مقصد اعلان: مدل اعلان که به بخش های مورد نظر با عناصر داده سرویس و وقایع دستیابی خواهیم داشت.

جعبه ابزار گلوباس 3(GT3)

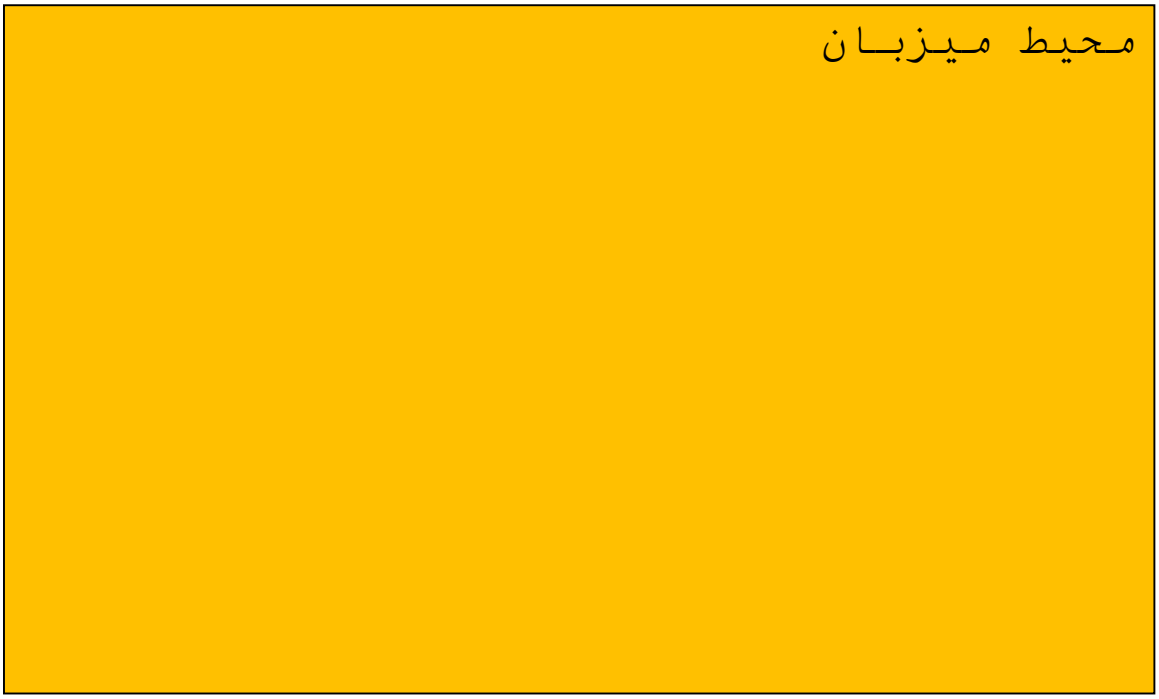
OGSA یک مشخصه معماری برااساس سرویس های وب برای تور است اکنون نیاز به آن است که در سطح پیاده سازی به آن پرئاخته شود که برای این منظور از OGSI استفاده می شود که یک مشخصه فنی از پیاده سازی OGSAاست. این مشخصه توسط حداقل 5 رویکرد پیاده سازی شده است. رویکردهای GT3، MS.NETGrid، OGSI.NET، OGSI::LITE و PyOGSI از این جمله اند که رویکرد GT3 در اینجا تشریح می گردد. شکل 17 ساختار GT3را نمایش می دهد.

1- محیط میزبان : دارای وظیفه مندی های زیر است:

a. تعیین زبان و مدل برنامه نویسی

b. ابزار های دیباگ

c. تضمین التزام به معانی سرویس های تور



شکل 17 ساختار GT3

GT3از محیط های میزبان مبتنی بر جاوا در زیر استفاده می کند. در حقیقت بر اساس صفات کیفی امنیت، مقیاس پذیری، قابلیت اطمینان و کارائی یکی از محیط های میزبان زیر انتخاب می شود.

:

1- نشانده شده: یک کتابخانه که اجازه می دهد که یک محیط میزبان OGSI در برنامه های کابردی J2SI نشانیده شود.

2- خود کفا:یک سرویس دهنده J2SI سبک وزن که میزبان سرویس های تور می شود.

3- کنتینر وب J2EE: یک محیط میزبان OGSIکه در یک سرویس دهنده وب محیط -Servlet compliantجاوا نظیر جاکارتا تام کت

4- کنتینر J2EE EjB: یک مولد کد که آشکار سازی موجودیت J2EE حالت کرا و مشخصه JavaBeans را بعنوان سرویس وب OGSI را می پذیرد.

2- موتور سرویس وب: وظیفه ان تبادل پیام SOAP بین سرویس دهندگان / گیرندگان است.Apachi Axis توسط GT3 بعنوان پیشفرض موتور های سرویس وب استفاده می شود.

3- کنتینر سرویس های تور: بر بالای موتور سرویس های وب قرار گرفته و ارائه کننده یک محیط زمان اجرا برای میزبان نمودن سرویس های متنوع است. یک کنتینر سرویس های تور می تواند برای یک مجموعه از محیط های میزبان چیدمان شده و در نتیجه یک محیط توزیع شده را ارائه می کند. این کنتینر سه نوع وظیفه مندی را حمایت می کند:

1- حمایت از کشف و توسعه جریان های اطلاعات

2- چیدمان پویا و مدیریت حالت نمونه های سرویس

3- یک انتقال مستقل از زیر ساخت امنیتی تور

4-سرویس های هسته GT3 : این سرویس ها بر اساس تمرکز بر روی پیاده سازی مشخصه OGSI شکل می گیرد.امنیت و سرویس های در سطح سیستم نیز از این نوع هستند.

4-1- پیاده سازی OGSI: یک مشخصه فنی برای پیاده سازی سرویسهای تور تعریف شده در OGSA است. در حقیقت پیاده سازی OGSI این مشخصه محموعه ای از پیاده سازی های اولیه از واسط های استاندارد OGSI بوده که تحت عنوان پورت تایپ است. همانند GridService،Factory، Notification(به ترتیب برای منبع، هدف و اشتراک یا تعهد[30] ، HandlerResolver،ServiceGroup(به ترتیب برای ورودی و ثبت نام) . اینها واسط های اولیه یا اصلی هستند که در عمل پیاده سازی شدهخ یا می توانند توسعه داده شوند. دو تای GridService، Factory بخش های پایه ای برای GT3هستند و قابل تغییر یا تعویض نیستند پیاده سازی واسط GridService در GT3ا برای پیاده سازی کنتینر پایه بوده و پیاده سازی واسط Factoryبرای اغلب برای مدیریت سرویس های تور در کنتینر GT3بکار می رود.

4-2- زیر ساخت امنیت : دو سطح امنیت در G3حمایت می شود یکی امنیت در سطح انتقال که توسط پروتکل HTTPG ارائه شده که GSIرا بر روی HTTP اعمال می کند. دومی امنیت در سطح پیام است که توسط مبادله ایمن GSI و امضاء GSI XML با استفاده از WS-Security,XML-Encryption,XML-Signatureصورت می گیرد.

4-3- سرویس های در سطح سیستم: شامل سه مورد زیر است:

1- سرویس ADMIN: مورد استفاده برای PING نمودن یک محیط میزبانی و کوشش برای shutdown یک کنتینر است.
2- مدیریت LOGGING
3- مدیریت که برای نظارت حالات جاری و بار کردن یک کنتینر سرویس تور بکار می رود. همینطور برای فعال نمودن یا غیر فعال کردن نمونه های سرویس تور بکار می رود.
4- 5- سرویس های پایهGT3: بر اساس سرویس های هسته، سرویس های پایه برای مدیریت منابع، سرویس های اطلاعاتی و انتقال مطمئن فایل است:

1- مدیریت منابع: GRAM[31] یا در حقیقت مدیریت تخصیص منابع گلوباس معماری منابع در سطوح پایین است. برای دسترسی یک وظیفه از طریق GRAM یک سرویس گیرنده از زبان مشخصه منابع یا RSL [32] استفاده می کند.
2- سرویس فهرست GT3: برای مدیریت ایستا و پویای داده ها در سیستم تور مورد استفاده قرار می گیرد. وظیفه مندی های زیر را به عهده دارد:

a. مدیریت و ایجاد داده های سرویس پویا از طریق مولفه های ارائه کننده سرویس

[30] Subscribtion
[31] Globus Resource Allocation Manager
[32] Resource Specification Language

b. تجمیع داده های سرویس از نمونه های چند گانه

c. ثبت نام نمونه های سرویس تور

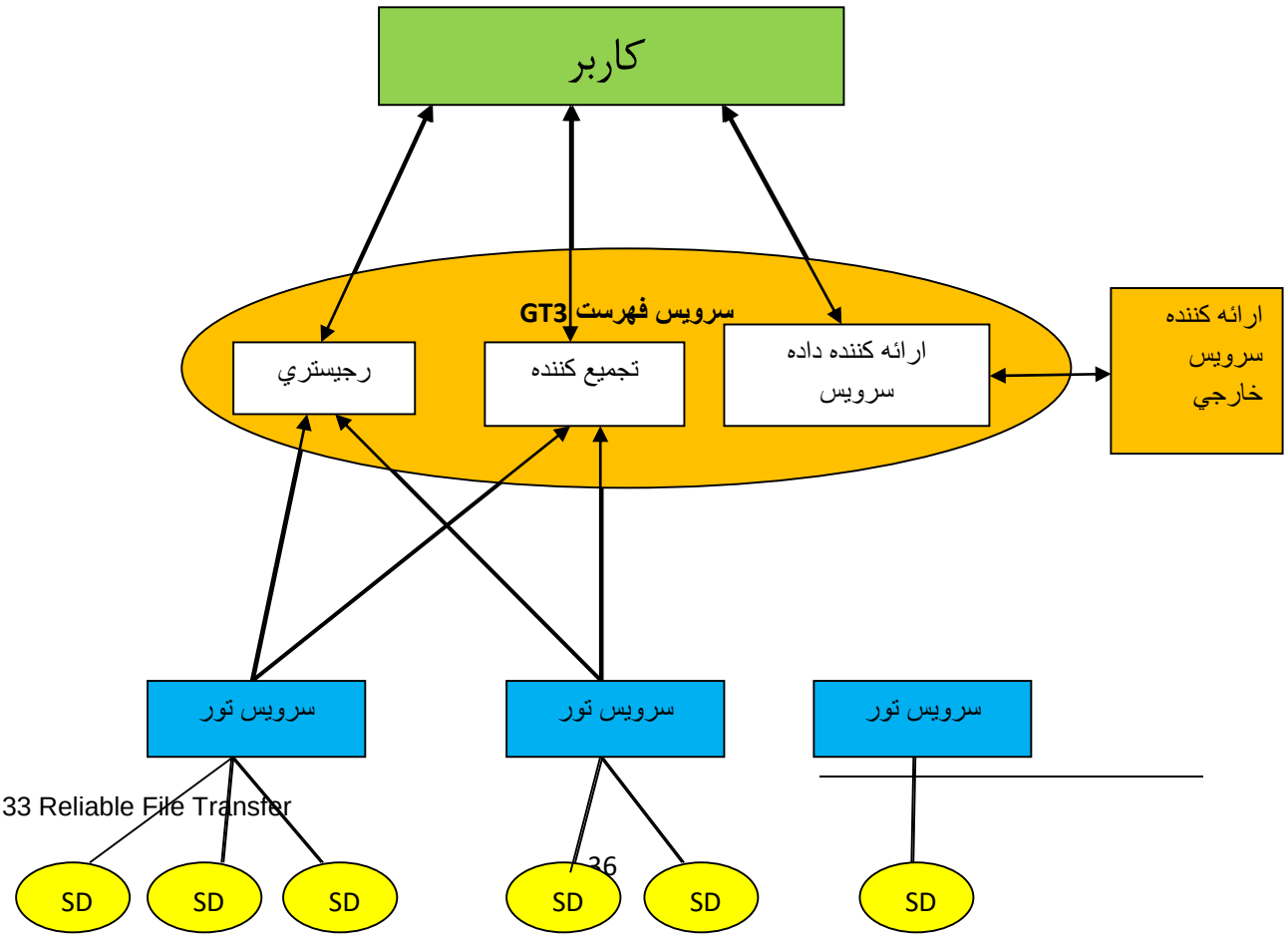
ساختار این سرویس در شکل 18 نمایش یافته است:

3- انتقال دهنده ایمن فایل : سرویس RFT[۳۳] برای ایمنی در انتقال فایل مورد استفاده قرار می گیرد که برای این منظور واسطی را برای نظارت و کنترل در حین انتقال استفاده می کند.

اکنون یکی از مهمترین نتایج این بخش که مدل برنامه نوسی GT3 است را ارائه می کنیم

6-4مدل برنامه نویسی GT3 :

شکل 19 معرف این مدل است :

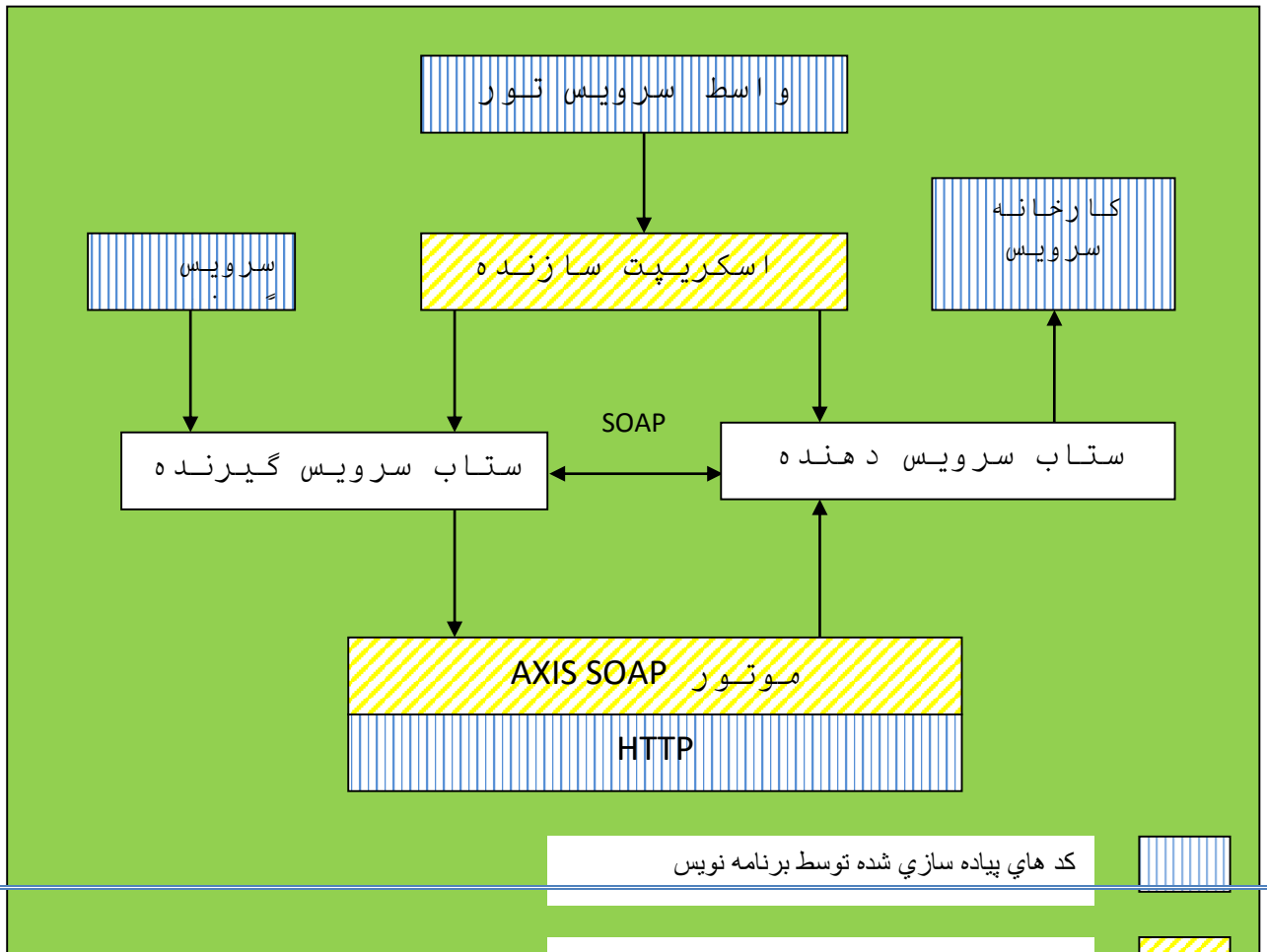


33 Reliable File Transfer

شکل 18- ساختار سرویس فهرست

لیست زیر مراحل کاربرد Apachi Axis را برای توسعه برنامه کاربردی تحت GT3 معلوم می کند:

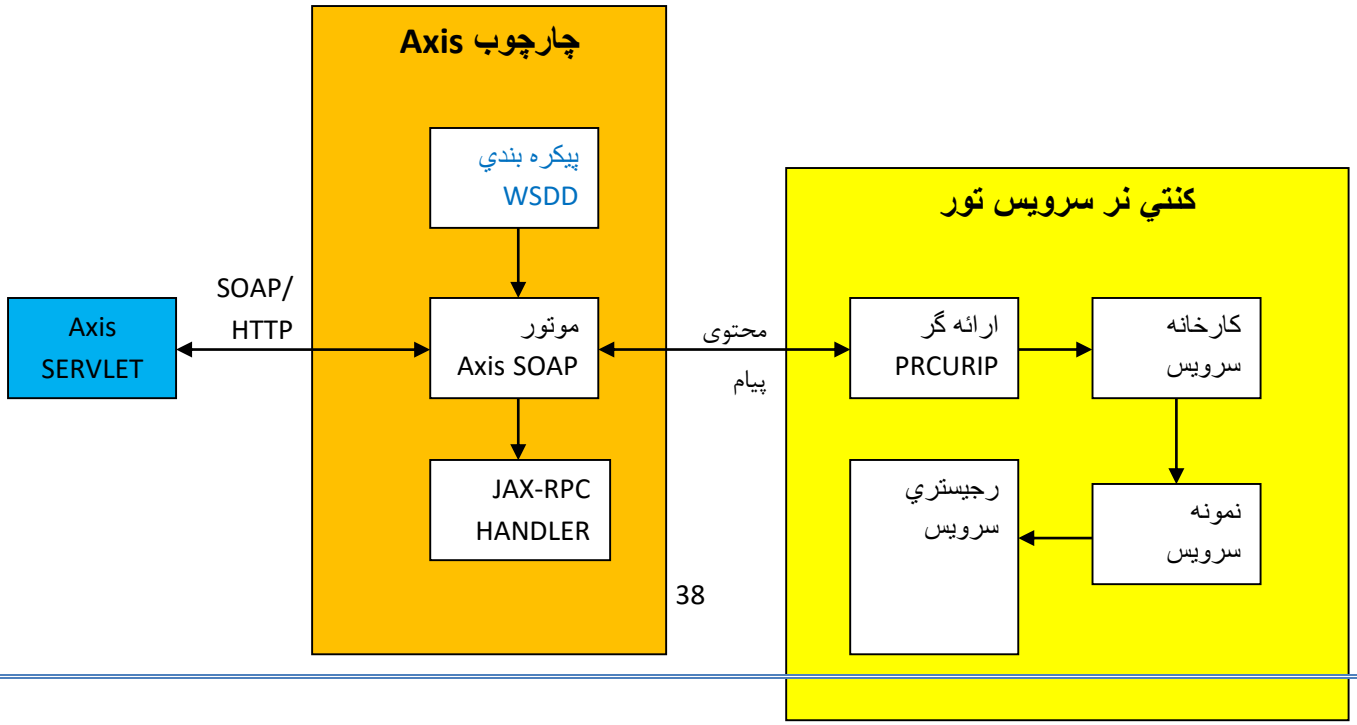
- نوشتن واسط تور با جاوا یا WSDL و یا GWSDL (این زبان نوعی WSDLاست که همه انواع مشخصه OGSIرا حمایت می کند)
- نوشتن سرویسی برای اجرای واسط سرویس تور
- نوصتن فایل WSDDبرای چیدمان کارخانه سرویس تور
- استفاده از اسکریپت سازنده GT3برای ترجمه و پیاده سازی فایل واسط سرویس تور و ایجاد بسته ای از آن به اضافه فایل های مرتبط همانند استاب های ایجاد شده از واسط و فایل WSDD در یک فایل GAR(نوعی فایل JAR)
- استفاده از Apachi Ant برای چیدمان فایل GARدر یک کنتینر سرویس تور جهت انتشار سرویس



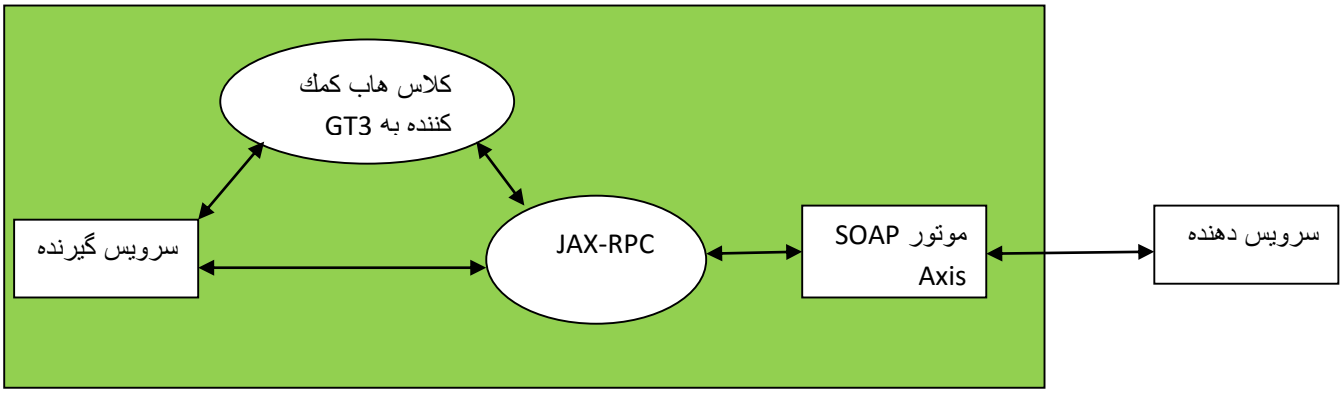
شکل 19- کنترل جریان داده ها در برنامه های کاربردی GT3

- نوشتن یک سرویس گیرنده برای درخواست از کارخانه برای سرویس . این سرویس گیرنده ابتدا GTHرا از کارخانه دریافت نموده که بدنبال آن GTR را نیز دریافت می کند. از این GTRبرای ایجاد نمونه ای از سرویس استفاده می شود
- شروع یک کنتینر سرویس تور
- آغاز نمودن یک سرویس گیرنده جهت درخوایت سرویس

در ادامه مولفه های دو طرف سرویس دهنده و همینطور سرویس گیرنده GT3 را معلوم می کنیم. دو شکل 20و21 این دو را نشان می دهد.



شکل 20- موافه های طرف سرویس دهنده



شکل 21- مولفه های طرف سرویس گیرنده

OGSA-DIA -7

بدول توجه به این مولفه بکارگیری معماری OGSA ناقض است. در واقع OGSI برای ایجاد، حذف، انتشار، رجیستر و یا انتقال سرویس های تور بکار گرفته می شود امابرای تجمیع و یا مدیریت داده های سرویس نمی تواند بکار گرفته شود. برای این منظور OGSA-DIA در بالای OGSA قرار گرفته و دسترسی به داده ها و تجمیع آنها را امکان پذیر می کند. OGSA-DIA[34] یک میان افزار بوده که دسرسی آسان به منابع گوناگون و تجمیع آنها را موجب می شود. این منابع همانند بانک های رابطه ای، بانک های XML و یا سیستم های فایل روی تور هستند. OGSA-DIA هم دارای یک مشخصه و هم دارای یک پیاده سازی است. در مشخصه سرویس ها و واسط های موردنیاز برای دستیابی به داده ها و تجمیع معلوم می شوند واسط ها پروتوتایپ های WSDL توسعه یافته با مشخصه OGSI هستند و در حقیقت هدف OGSA-DIA آن است که منابع خارجی همانند بانک های اطلاعاتی از طریق واسط های سرویس های تور قابل دسترس شوند. بر اساس استفاده از OGSA-

[34] Open Grid Services Architecture Data Access and Integration

DIA همه منابع ناهمگون و جدای از هم بطور یکنواختی مورد دسترس قرار گرفته و در نتیجه محیطی توزیع شده و یکپارچه ایجاد می شود. پیش الگو های(پروتوتایپ) OGSA-DIAدر 5 نوع زیر موجودند:

1- پیش الگوی GDSPortType:که دسترسی داده ، تجمیع و تحویل آنها را از طریق GDS ارائه می کند.(درOGSA-DIA هر سرویس تور یک GDSنامیده می شود) این پیش الگو سه پیش الگوی سرویس تور یعنی GridService، GridDataPerform و GridDataTransport تعریف شده توسط OGSA-DIAرا توسعه می دهد.

2- GridDataPerform : ارائه کننده متدی برای سرویس گیرنده که دسترسی به منابع داده ای و نتایج بازیافتی را موجب می شود.

3- GridDataTransport: انتقال داده ها را بین سرویس دهنده ها, و بین سرویس دهنده ها و سرویس گیرنده ها را موجب می شود.

4- GridDataServiceFactory: برای پیاده سازی یک کارخانه GDS بکار می رود.

5- DAIServiceGroupRegistry : برای پیاده سازی یک رجیستری جهت ثبت هر سرویسی که موجب پیاده سازی هر پیش الگوی OGSA-DIA می گردد.

نتیجه گیری

در این مقاله OGSA بعنوان یکی از مهمترین معماری های متعارف برای توسعه تور معرفی گردید. دراین مقاله معلوم شد که چرا جامعه تور به این معماری جهت ساخت، مدیریت و توسعه تور نیازمند است. در ابتدا در این مقاله به تکنیک های متعارف جهت ساخت میان افزار های محیط های توزیع شده پرداخته شد و 5 مدل برنامه نوسی سوکت، پارادایم RPC و Java RMIو DCOM وCORBA مطالعه شد. در ادامه به تکنیک سرویس های وب برای ساخت تور استفاده گردید که برای این منظور از معماری SOAاستفاده گردید. استفاده از سرویس های وب یک روش آسان، استاندارد جهت ساخت میان افزار برای محیط های محاسباتی توزیع شده و یکپارچه خواهد بود در حقیقت OGSAیک معماری برای ساخت تور بوده که از تکنیک های سرویس وب بطور گسترده ای استفاده می کند. به همین دلیل یک تور ساخته شده بروش OGSA همانند اینترنت دارای معضلاتی است که وابسته یه توصیف و اکتشاف سرویس های وب است. در هر صورت OGSAامروزه روش استاندارد دو فاکتو برای ساخت تور بروش استفاده از سرویس های وب است. این معماری پیش الگو هائی (واسط هائی) را ایجاد نموده که بر اساس آنها می توان به تور مورد نظر از طریق زبان WSDL دستیابی داشت. یکی از ویژگی های معماری

OGSA جدا سازی آن از روش پیاده سازی است OGSA معماری برای سرویس های تور است که خصوصیت پویائی، آنها را از سرویس های وب متمایز می کند.

از مهمترین خصوصیات OGSA استفاده غیر مستقیم از سرویس های وب است و به همین دلیل نیز این معماری دچار چالشی بزرگ می شود که بر گرفته از ماهیت ایستا و غیر انتقالی سرویس های وب در مقابل سرویس های تور انتقالی و وابسته به حالت است. برای حل این مشکل OGSA از مفهوم GSH/GSR به همراه عنصر داده سرویس برای هر سرویس تور استفاده می کند. نتیجه اینکه OGSA از سرویس های تور استفاده نموده و پیاده سازی آنها را رها می کند و , OGSI مشخصه ای از پیاده سازی سرویس های تور است که مستقیما از سرویس های وب استفاده می کند. این OGSIاست که معلوم می کند که کدامین سرویس تور در OGSAقابل بکارگیری است این سرویس ها که بنام هسته نامیده می شوند از جمله شامل ایجاد و حذف، توصیف و اکتشاف، رجیستری و اعلان هستند. به این ترتیب نیازمند GT3 بعنوان یک ارجاع پیاده سازی از OGSI بوده که به نحوه گسترده ای برای برنامه های کاربردی متقاضی OGSA چیدمان شده باشد. چون مفهوم سرویس در OGSA کلی است بنابراین از OGSA-DIAبرای دستیابی به داده های منابع خارجی و تجمیع با سرویس های هسته، جهت پیاده سازی برنامه کاربردی متقاضی OGSAاستفاده می شود.