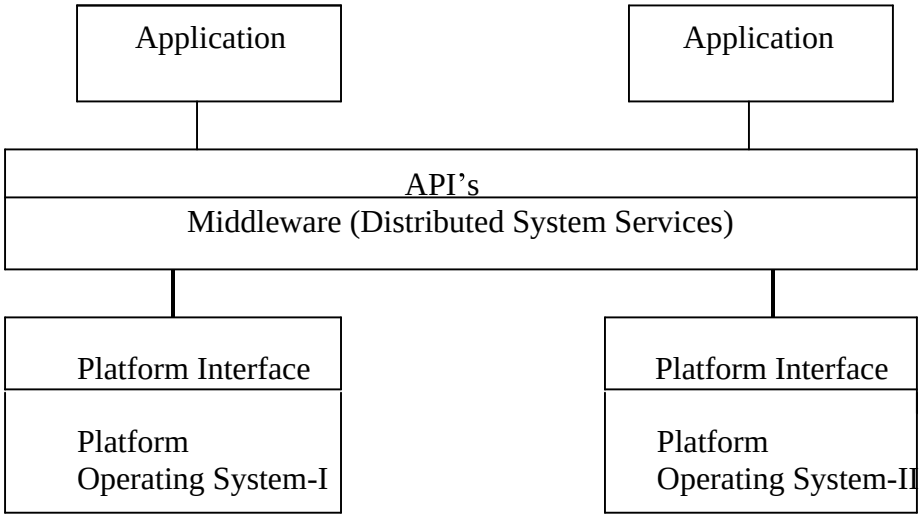
Middleware- Emerging Technology & Its Controls.

Till 1980's most of computing was based on central host computers equipped with powerful processors and memory. Users interact with the host through the terminals that captures keystrokes and sends the information to host. A major bottleneck for this architecture was that the processing power was limited to that of central host system, over dependence on the vendor for application software, lack of support for GUI and access to multiple databases. The mainframes prevalent at that time were based on this architecture. With advent of PC's the files were downloaded from the shared location, processed and uploaded back to file server. This had major drawback as it generated too much of network traffic. However with emergence of client /server architecture, the computing power or process management was distributed between the client and server. For example client could query database server using relational database management system (DBMS) through standard query language (SQL). The results of query are sent to the client, which then manipulates and processes the data. This two-tier client/server architecture has limitation as the number of users grows beyond certain limit, due to the fact that server has to maintain a dialog of connection even when client is idle. Moreover any changes in application or parameter would entail changes at all clients like a change in VAT rate would need update on all the users workstation. To overcome these limitations middle-tier was added between the user system interface client environment and database management server environment. The middle tier or middleware is now one of the emerging technologies in client server paradigm. It provides for connectivity across heterogenous platform and for more generalization of Application Programming Interface (API) than operating system or network services as shown in Fig.1



Fig(1).

It further raises the level of abstraction of programming of distributed applications, as developer need not worry about the platform or operating systems. It can have various implementations such as transaction processing monitors, message servers, remote procedure calls, Object Request Broker or application servers.

Let us have examine each of the implementations in detail-

1. The most basic type of three tier architecture is used in Online Transaction Processing Technologies (OLTP) applications using middle layer consisting of Transaction Processing (TP) monitor technology. This is a type of message queuing, transaction scheduling and prioritization service where the client connects to the middle tier viz. TP monitor which in turn connects to the back end database. The transaction is accepted by the monitor, which queues it and then takes responsibility for managing thus relieving the client. It has ability to connect to different DBMS's in single transactions irrespective of whether it is flat file or non-relational DBMS. This architecture is considerably more scalable than a two tier.
2. Message Servers: This implementation, also known as Message-oriented middleware (MOM), provides program-to-program data exchange with intelligent messages sent asynchronously. It is similar to email exchanged between the programs. It requires recipient programs to interpret these messages and take appropriate action. MOM increases flexibility of architecture by enabling applications to exchange messages with each other without need to bother about the underlying operating system or the processors. MOM is most appropriate for event-driven applications. For an airline, for example, passenger ticket reservation and cargo bookings events are source for load factor, flight arrival and departure events through ground operations, is source for aircraft movement. All this information moves using MOM. The executive staff can know the effective aircraft utilization.
3. Remote Procedure Call (RPCs): enables the logic of application to be distributed across the network. Program logic on remote systems can be executed by simply calling a routine. For example network printer or shared folder can be located across the network as locally attached resource
4. Application Servers: There is a shared host on which business logic, computation and data retrieval engine resides. The GUI component resides on the front-end client making this architecture highly scalable, secure and lends itself to changes easily. For example in banking scenario in which interest rates change frequently. This would entail changing a parameter only on shared host without change at teller end or at database end. The fig 2 below shows clients accessing the web server which optimized to serve web pages while application server based on the inputs from clients and business decision logic residing on it, queries the database. The results are pushed on the web server for serving to end user browser.

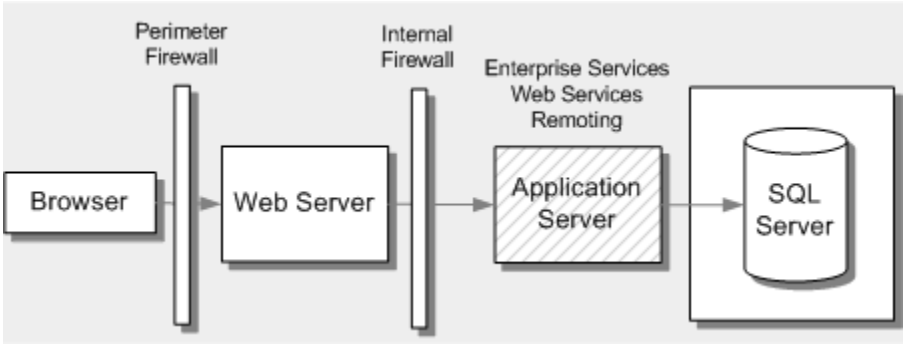


Fig. 2

5. Object Request Broker Architecture: This refers to specification and implementation framework for interoperability and reusability of distributed objects. These initiatives are driven by two rival camps - Microsoft with COM/DCOM technology and Object Management Group (OMG) with Common Object Request Broker Architecture (CORBA). These defines application program interface (API) through which various components interact independent of language or platform.

Business Considerations in selecting Middleware:

While middleware increases the level of abstraction, developers need to be prudent enough in their choice of services in deciding which components are to placed on which tier. Though the middleware implementations are suppose to be platform independent, many of these are vendor specific like COM/DCOM from Microsoft. Thus they need to be compiled for a specific platform or need an interpreter. The availability of development tools like C++, Visual Basic, Java are key for customized development middleware services. The components in general and those that involve business logic should be easily replaceable. Another consideration is that a good middleware should not be visible to client. It should seamlessly connect the client to back end. While these are technical and aesthetic considerations, there are also strategic business factors to be considered. Normally in an enterprise there are islands of application developed over period of time. They reside on heterogeneous platform across various functional units of an organization. As businesses become competitive, there is crucial need by business owners to have information on state of business at any moment. Moreover the need for better customer service demands integration of these applications. This is where middleware has to play an important role in Enterprise Integration.

Controls & Security Considerations

The scope of middleware deployment is broad and as such should be tackled from business perspective rather than from only technical one. When middleware deployment should focus on these issues:-

- The selection of middleware is a crucial decision. While legacy systems are web enabled leveraging the power of middleware, certain controls existent may not be relevant or has to be reengineered. Data, which hitherto was accessible only to select few in an enterprise, there is a risk of it being available to malicious hackers.
- Authentication and Authorization: In message oriented middleware (MOM), as programs communicate with other programs, messages need to be authenticated, encrypted and authorized by MOM managers. As various applications publish their messages, due care has to be taken as which recipient applications can subscribe to these messages. Similarly in application servers communication with front end webserver and back-end database server need to protected from unauthorized access and network eavesdropping as shown in fig 3

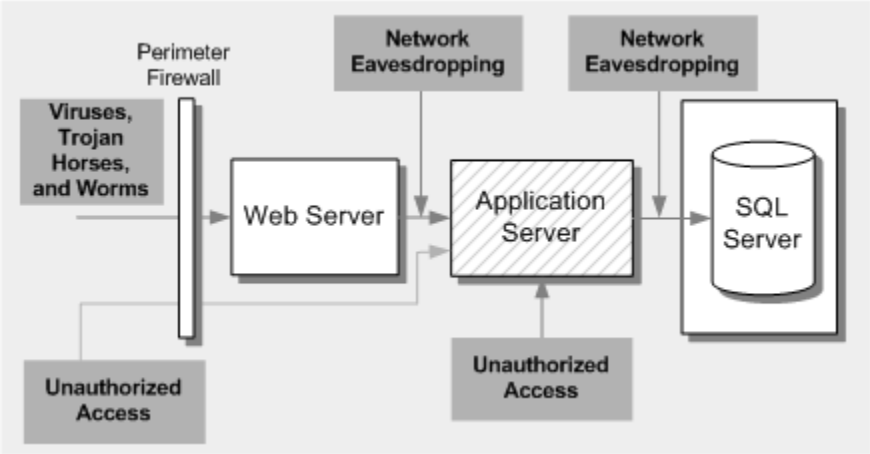


Fig. 3

Similarly in TP monitors, transaction context type in database need to be secured. In this, the context or permissions to select, insert, update, delete and execute needs to be controlled.

- Auditing: Middleware deployments irrespective of type of implementation should be auditable through logs and reporting tools. This includes unauthorized access, enhancing the privilege attempts and application warning messages. Besides middleware code-review should be done through assistance of expert application programmers.

Conclusions:

Middleware technology is firmly entrenched in distributed computing horizon. It is enabler for enterprise application integration in today's "state of business at the moment" paradigm. While functional units across the enterprise may operate independently, middleware technology can be leveraged to provide integrated solution for better customer service and enhanced management information services.

Naushad Rajani is working, as IS Security Analyst with Riyad Bank, Saudi Arabia. He is CISA, CISSP, CCNP, NCSA (NetScreen OS 4.0), MCSE and can be reached at naushadr@riyadbank.com.