Name: Shivam Tiwari
Rollno: 5117060

Experiment 01

Aim: To Study Middleware in Distributed Computing

Theory:

Middleware is a software layer situated between applications and operating systems. Middleware is typically used in distributed systems where it simplifies software development by doing the following:

- Hides the intricacies of distributed applications
- Hides the heterogeneity of hardware, operating systems and protocols
- Provides uniform and high-level interfaces used to make interoperable, reusable and portable applications
- Provides a set of common services that minimizes duplication of efforts and enhances collaboration between applications

Middleware is similar to an operating system because it can support other application programs, provide controlled interaction, prevent interference between computations and facilitate interaction between computations on different computers via network communication services.

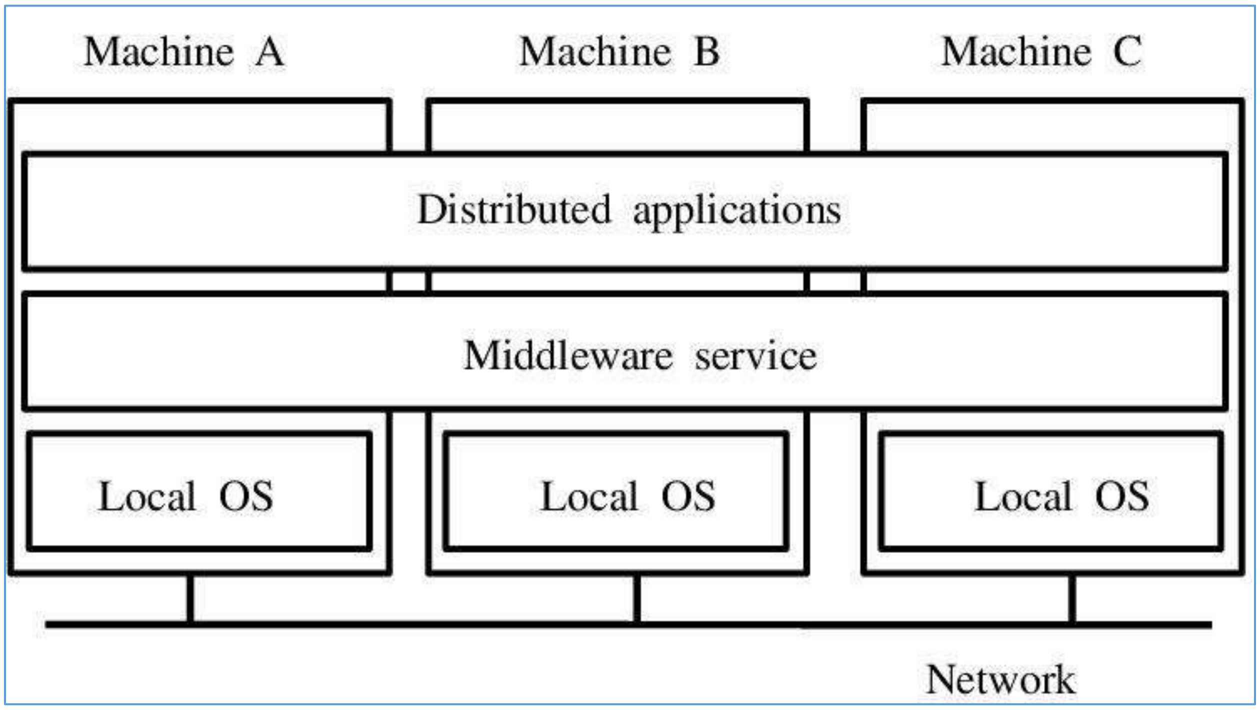
A typical operating system provides an application programming interface (API) for programs to utilize underlying hardware features. Middleware, however, provides an API for utilizing underlying operating system features.

There are several technology frameworks to support distributed architectures, including .NET, J2EE, CORBA, .NET Web services, AXIS Java Web services, and Globus Grid services.

Middleware is an infrastructure that appropriately supports the development and execution of distributed applications. It provides a buffer between the applications and the network.

It sits in the middle of system and manages or supports the different components of a distributed system. Examples are transaction processing monitors, data convertors and communication controllers etc.Middleware is a set of service that enables application and end-user to interacts with each other across a heterogeneous distributed system.

Middleware in the context of distributed applications is software that provides services beyond those provided by the operating system to enable the various components of a distributed system to communicate and manage data. Middleware supports and simplifies complex distributed applications Middleware is software glue. Middleware is the slash in Client/Server. Middleware is an important class of technology that is helping to decrease the cycle-time, level of effort, and complexity associated with developing high-quality, flexible, and interoperable distributed systems. Increasingly, these types of systems are developed using reusable software (middleware) component services, rather than being implemented entirely from scratch for each use. When implemented properly, middleware can help to: Shield developers of distributed systems from low-level, tedious, and error-prone platform details, such as socket-level network programming. • Amortize software lifecycle costs by leveraging previous development expertise and capturing implementations of key patterns in reusable frameworks, rather than rebuilding them manually for each use. • Provide a consistent set of higher-level network-oriented abstractions that are much closer to application and system requirements to simplify the development of distributed systems.



Provide a wide array of developer-oriented services, such as logging and security that have proven necessary to operate effectively in a networked environment. Middleware was invented in an attempt to help simplify the software development of distributed computing systems, and bring those capabilities within the reach of many more developers than the few experts at the time who could master the complexities of these environments. Complex system integration requirements were not being met from the application perspective, where it was too hard and not reusable, or the network or host operating system perspectives, which were necessarily concerned with providing the communication and end system resource management layers, respectively. Over the past decade, middleware has emerged as a set of software service layers that help to solve the problems specifically associated with heterogeneity and interoperability. It has also contributed considerably to better environments for building distributed systems and managing their decentralized resources securely and dependably. Consequently, one of the major trends driving industry involves moving toward a multi-layered architecture (applications, middleware, network and operating system infrastructure) that is oriented around application composition from reusable components, and away from the more traditional architecture, where applications were developed directly atop the network and operating system abstractions. This middleware-centric, multi-layered architecture descends directly from the adoption of a network-centric viewpoint brought about by the emergence of the Internet and the componentization and commoditization of hardware and software. Infrastructure that supports middleware are (distributed) component based application development.

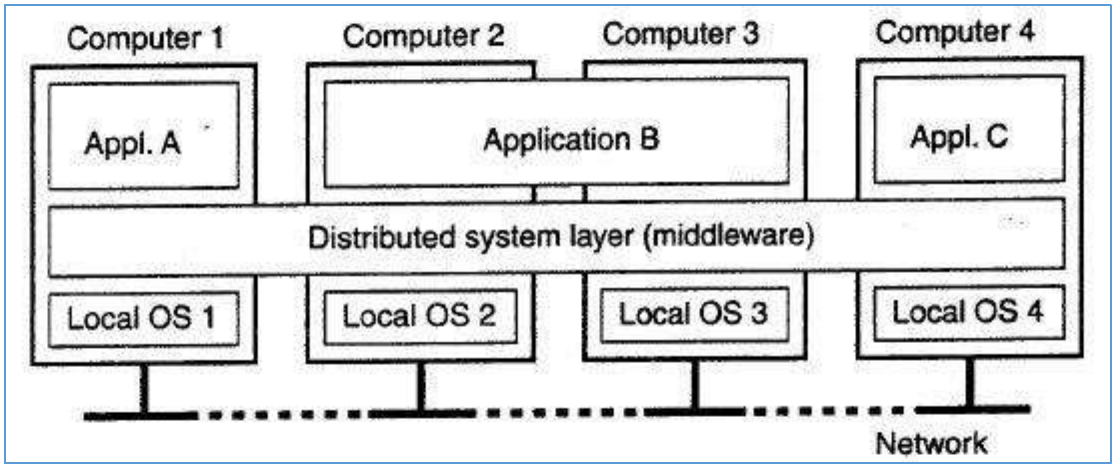
NEED FOR MIDDLEWARE

In the age of computers, each user seeking knowledge has a desktop appliance that connects to the utility, i.e. client asking server for the information. Here desktop appliance can be computer or devise like computer e.g. a terminal, personal computer, workstation, word processor etc. The utility is an enterprise wide network of information services which includes applications, databases on LAN and WAN. Servers on LAN support files and file based applications, such as e-mail, bulletin board, document preparation and printing. They also support directory services. Directory services help desktop users to search other users and services of their interest. Servers on WAN generally support database access.

Database access includes corporate directories and electronic libraries or transaction processing applications purchasing, billing and inventory control. Most of the organizations have large variety of heterogeneous hardware systems which include personal computers, workstations, minicomputers and main frames. All these heterogeneous systems use different OS and network architecture. So integration of these systems is difficult. Here comes the middleware in picture. Middleware deals with providing environments for developing systems that can be distributed effectively over a variety of topologies, computing devises and communication network. It aims to provide developers of networked applications with the required platform and tools to formalize and coordinate how parts of applications are composed and how they interoperate, monitor, enable and validate the configuration of resources to ensure appropriate application service quality, in case of failure also.

Network communication

- marshaling/unmarshalling Coordination
- Activation/termination, threading, group requests,
  synchronicity Reliability
- Delivery guarantees, total/partial ordering, atomicity, replication.



TYPES OF MIDDLEWARE

A. <u>Message Oriented Middleware</u>

This is a large category and includes communication via message exchange. It represents asynchronous interactions between systems. It reduces complexity of developing applications that span multiple operating systems and network protocols by insulating the application developer from details of various operating system and network interfaces.

APIs that extend across diverse platforms and networks are typically provided by the MOM. MOM is software that resides in both portions of client/server architecture and typically supports asynchronous calls between the client and server applications. Message queues provide temporary storage when the destination program is busy or not connected. MOM reduces the involvement of application developers with the complexity of the master-slave nature of the client/server mechanism. E.g. Sun’s JMS.

B. Object Request Broker

In distributed computing, an object request broker (ORB) is a piece of middleware software that allows programmers to make program calls from one computer to another via a network. ORB's handle the transformation of in-process data structures to and from the byte sequence, which is transmitted over the network. This is called marshaling or serialization. Some ORB's, such as CORBA-compliant systems, use an Interface Description Language (IDL) to describe the data which is to be transmitted on remote calls. E.g. CORBA

C. RPC Middleware.

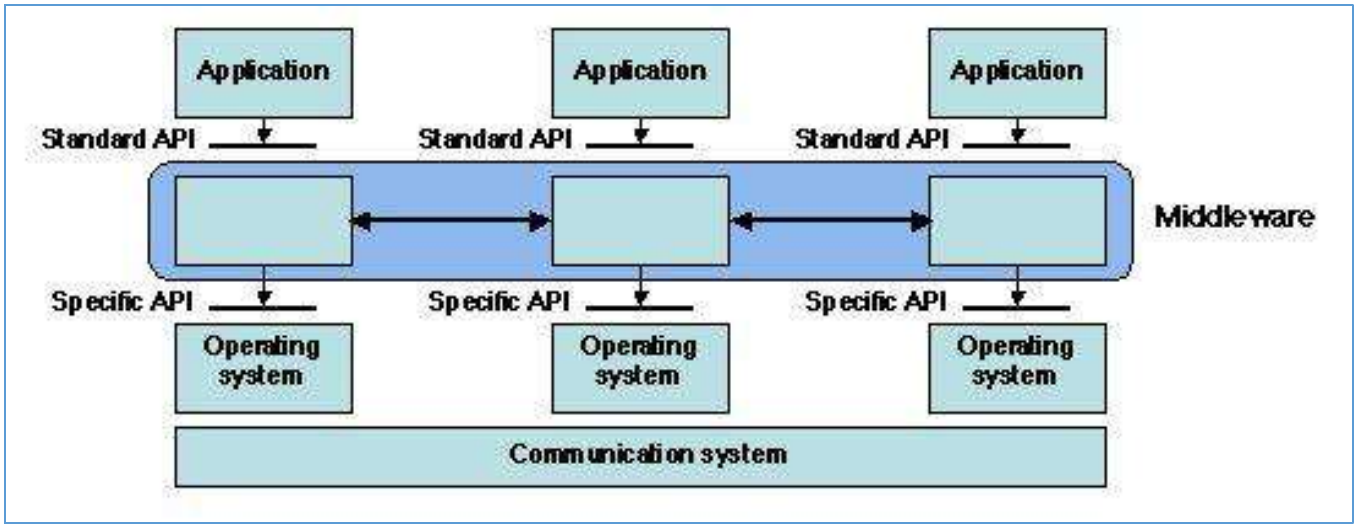
This type of middleware provides for calling procedures on remote systems, so called as Remote Procedure Call.Unlike message oriented middleware, RPC middleware represents synchronous interactions between systems and is commonly used within an application.Thus, the programmer would write essentially the same code whether the subroutine is local to the executing program, or remote. When the software in question is written using object-oriented principles, RPC may be referred to as remote invocation or remote method invocation. Client makes calls to procedures running on remote systems, which can be asynchronous or synchronous. E.g. DCE RPC.

D. Database Middleware.

Database middleware allows direct access to data structures and provides interaction directly with databases. There are database gateways and a variety of connectivity options. Extract, Transform, and Load (ETL) packages are included in this category. E.g. CRAVE is a web-accessible JAVA application that accesses an underlying MySQL database of ontologies via a JAVA persistent middleware layer Chameleon).

E. Transaction Middleware. This category as used in the Middleware Resource Center includes traditional transaction processing monitors (TPM) and web application servers. e.g. IBM’s CICS.

F. <u>Portals</u>
Enterprise portal servers are included as middleware largely because they facilitate "front end" integration. They allow interaction between the user's desktop and back end systems and services. e.g Web Logic.



EXAMPLES OF MIDDLEWARE

The term middleware is used in other contexts as well. Middleware is sometimes used in a similar sense to a software driver, an abstraction layer that hides detail about hardware devices or other software from an application.

1. The Android environment uses the Linux operating system at its core, and also provides an application framework that developers incorporate into their applications. In addition, Android provides a middleware layer including libraries that provide services such as data storage, screen display, multimedia, and web browsing. The Android middleware layer also contains the Dalvik virtual machine and its core Java application libraries.
2. Game engine software such as Gamebryo and Render ware are sometimes described as middleware, because they provide many services to simplify game development.
3. In simulation technology, middleware is generally used in the context of the high level architecture (HLA) that applies to many distributed simulations. It is a layer of software that lies between the application code and the run-time infrastructure.

4. Wireless networking developers can use middleware to meet the challenges associated with wireless sensor network (WSN), or WSN technologies. Implementing a middleware application allows WSN developers to integrate operating systems and hardware with the wide variety of various applications that are currently available.
5. The QNX operating system offers middleware for providing multimedia services for use in automobiles, aircraft and other environments.
6. Multimedia Home Platform (DVB-MHP) is an open middleware system standard designed by the DVB project for interactive digital television. The MHP enables the reception and execution of interactive, Java-based applications on a television set.

Conclusion: Thus, study of middleware is done with need, types and examples of middleware as well.