Session 4
Architectural Patterns

Course Leader: Jishmi Jos Choondal

Objectives

At the end of the lecture, the student will be able to

- Discuss the various architectural patterns and associated middleware solutions

Contents

- Architectural patterns
- Associated middleware solutions

Architectural Patterns

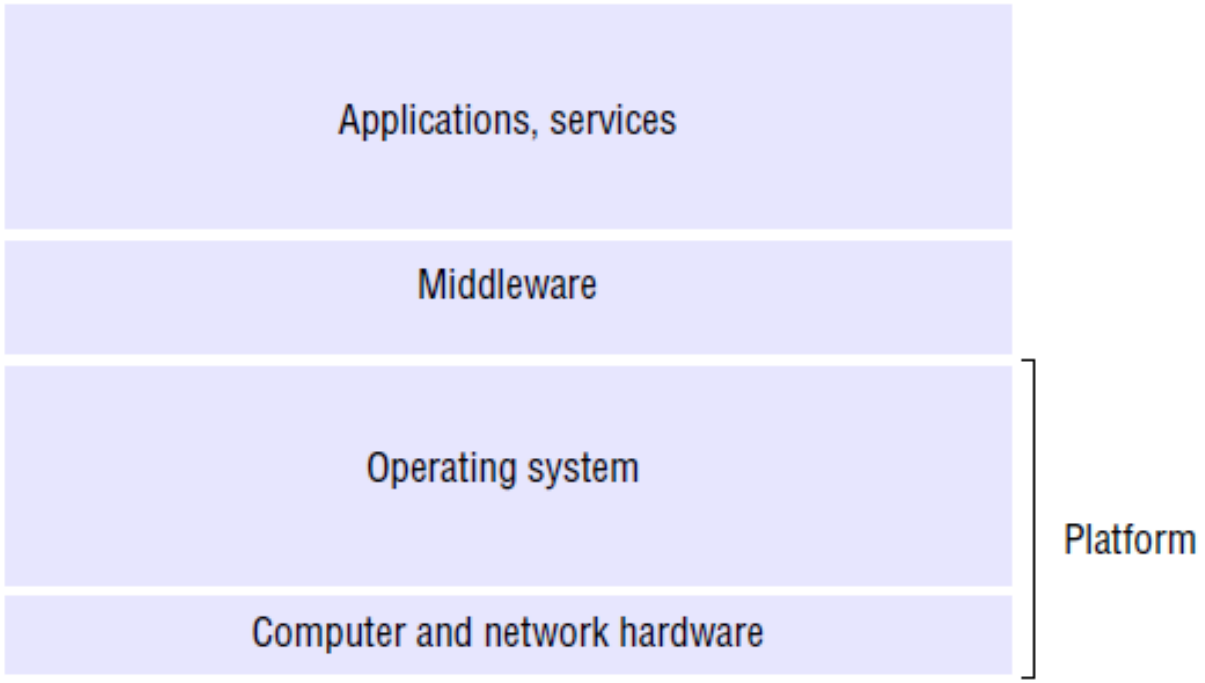
- Layering
  - Deals with vertical organization of services
- Tiered architecture
  - To organize functionality of a layer and place into appropriate servers or onto physical nodes
  - Two tiered architecture and Three tiered architecture
- Thin Clients
- Other commonly occurring Patterns
  - Proxy, Brokerage, Reflection

Layering

- In a layered approach, a complex system is partitioned into a number of layers, with a given layer making use of the services offered by the layer below.
- A given layer therefore offers a software abstraction, with higher layers being unaware of implementation details, or indeed of any other layers beneath them.
- In DS, it deals with vertical organization of services into service layers
- A distributed service can be provided by one or more server processes, interacting with each other and with client processes in order to maintain a consistent system-wide view of the service's resources.
- Eg. Network Time Service

Layering

Figure 2.7 Software and hardware service layers in distributed systems



Layering

- Low-level layers provide services to the layers above them, which are implemented independently in each computer
- Middleware is a layer of software whose purpose is to mask heterogeneity and to provide a convenient programming model to application programmers.
- Middleware is represented by processes or objects in a set of computers that interact with each other to implement communication and resource-sharing support for distributed applications.
- Remote method invocation; communication between a group of processes; notification of events; the partitioning, placement and retrieval of shared data objects amongst cooperating computers; the replication of shared data objects; and the transmission of multimedia data in real time.

Tiered architecture

- Tiering is a technique to organize functionality of a given layer and place this functionality into appropriate servers and, as a secondary consideration, on to physical nodes
- This technique is most commonly associated with the organization of applications and services but it also applies to all layers of a distributed systems architecture

Tiered architecture

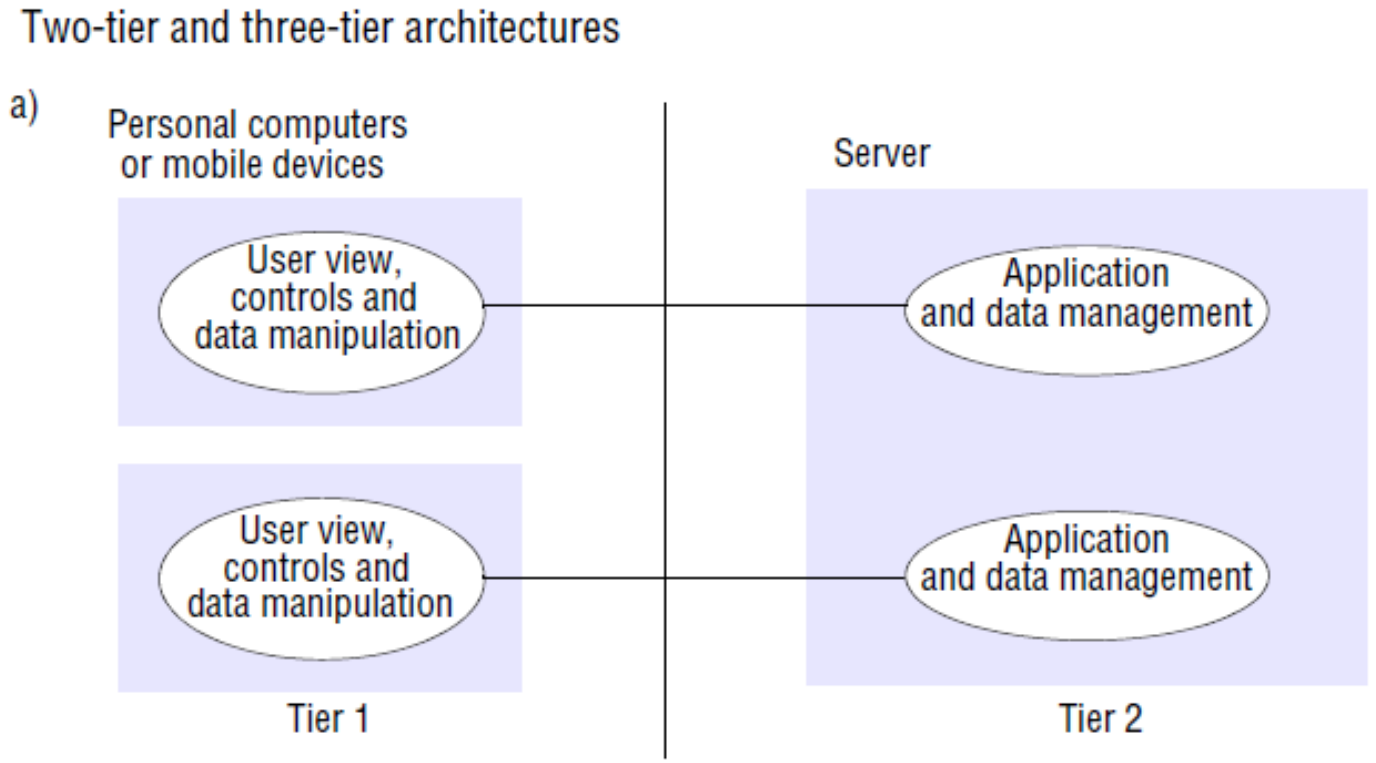
functional decomposition of a given application as follows:

- the presentation logic, which is concerned with handling user interaction and updating the view of the application as presented to the user;
- the application logic, which is concerned with the detailed application-specific processing associated with the application (also referred to as the business logic, although the concept is not limited only to business applications);
- the data logic, which is concerned with the persistent storage of the application, typically in a database management system.

Two Tiered architecture

- In the two-tier solution, the three aspects mentioned above must be partitioned into two processes, the client and the server.
- This is most commonly done by splitting the application logic, with some residing in the client and the remainder in the server
- Advantage: low latency in terms of interaction, with only one exchange of messages to invoke an operation
- Disadvantage: splitting of application logic across a process boundary, with the consequent restriction on which parts of the logic can be directly invoked from which other part.

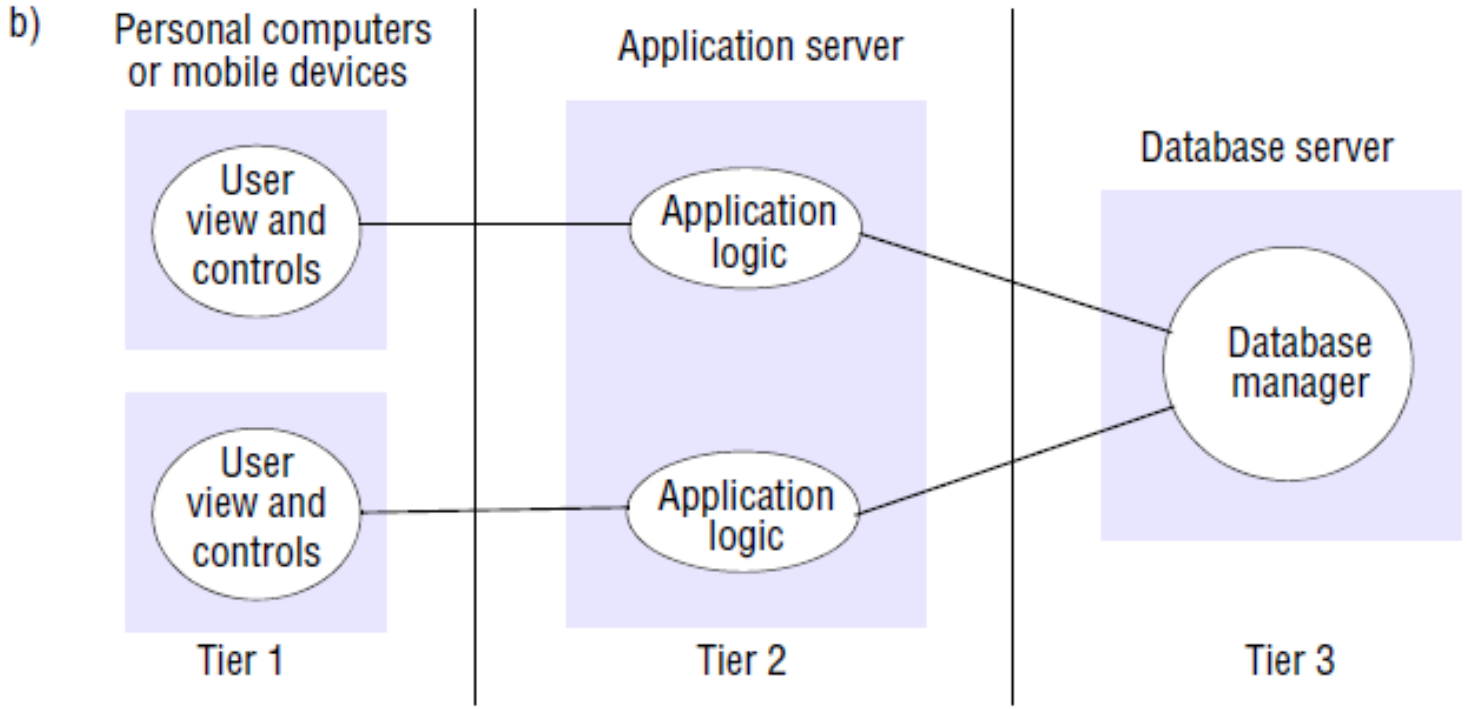
Two Tiered architecture



Three Tiered architecture

- There is a one-to-one mapping from logical elements to physical servers and hence, for example, the application logic is held in one place, which in turn can enhance maintainability of the software.
- Each tier also has a well-defined role; for example, the third tier is simply a database offering a (potentially standardized) relational service interface.
- The first tier can also be a simple user interface allowing intrinsic support for thin clients
- The drawbacks are the added complexity of managing three servers and also the added network traffic and latency associated with each operation

Three Tiered architecture



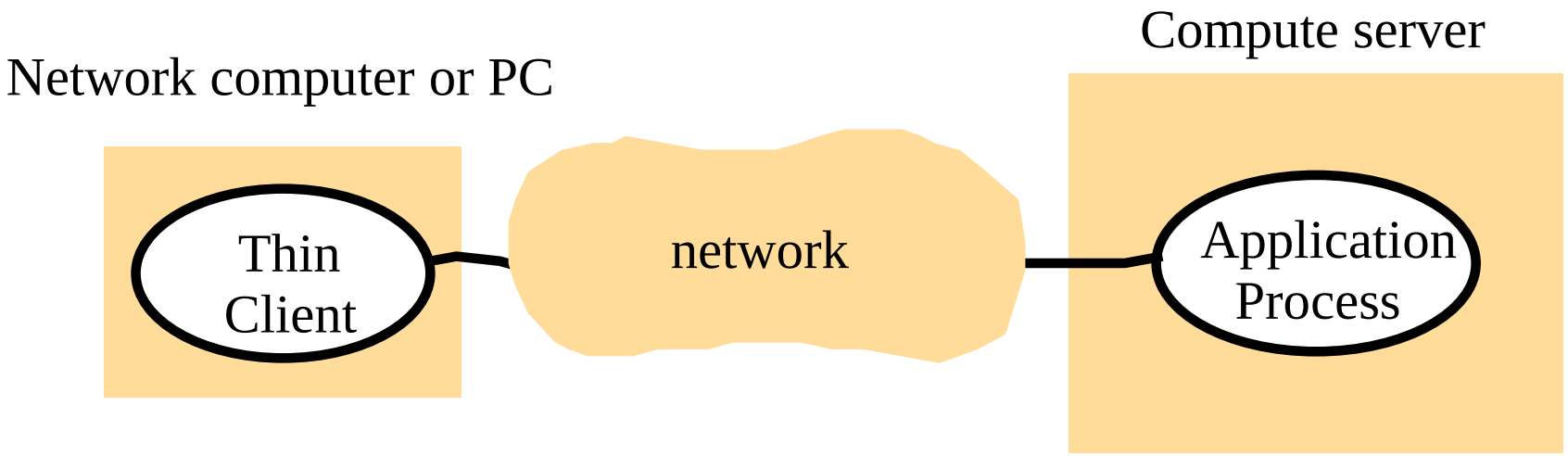
n-tiered (or multi-tier) architecture

- this approach generalizes to n-tiered (or multi-tier) solutions where a given application domain is partitioned into n logical elements, each mapped to a given server element.
- As an example, Wikipedia, the web-based publicly editable encyclopedia, adopts a multi-tier architecture to deal with the high volume of web requests (up to 60,000 page requests per second).

Thin Clients

- It is a software layer that supports a window based user interface on a computer that is local to the user while executing application programs on a remote computer.
- This architecture has the same low management and hardware cost as the network computer scheme.
- Instead of downloading the code of applications into the user's computer, it runs them on a *compute server*-
  - A powerful computer that has the capacity to run large numbers of applications simultaneously.
  - It will be a multiprocessor or a cluster computer.
- The main drawback is in highly interactive graphical activities such as CAD and image processing, where the delays experienced by users are increased by the need to transfer.

Thin Clients and Compute Servers



Categories of Middleware

Major categories:	*Subcategory*	*Example systems*
Distributed objects (Chapters 5, 8)	Standard	RM-ODP
	Platform	CORBA
	Platform	Java RMI
Distributed components (Chapter 8)	Lightweight components	Fractal
	Lightweight components	OpenCOM
	Application servers	SUN EJB
	Application servers	CORBA Component Model
	Application servers	JBoss
Publish-subscribe systems (Chapter 6)	-	CORBA Event Service
	-	Scribe
	-	JMS
Message queues (Chapter 6)	-	Websphere MQ
	-	JMS
Web services (Chapter 9)	Web services	Apache Axis
	Grid services	The Globus Toolkit
Peer-to-peer (Chapter 10)	Routing overlays	Pastry
	Routing overlays	Tapestry
	Application-specific	Squirrel
	Application-specific	OceanStore
	Application-specific	Ivy
	Application-specific	Gnutella

Limitations of Middleware

- Because of its prohibitively high development costs, not every business can afford to maintain and grow the potential of Middleware.
- Benchmarks for Middleware haven't been set, thus there are hardly any standard marks for Middleware performance levels.
- Most Middleware tools have not yet been fully developed for optimal operations.
- There are too many platforms in existence today that are not yet covered by Middleware.
- In some cases, Middleware often jeopardizes some systems' real-time performance

Summary

- System models: physical model, architectural model and fundamnetal model
- Architectural models of Distributed Systems include Client-Server, Peer to peer and their variations

References and Questions

- https://www.youtube.com/watch?v=L5BlpPU_muY

Thank you