M-V-C

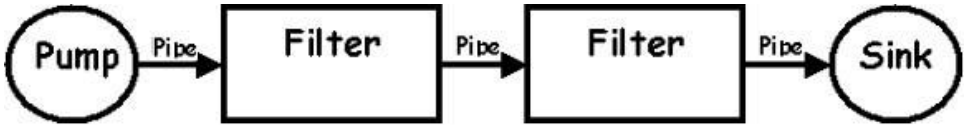
Model **V**iew **C**ontroller or MVC as it is popularly called, is a software design pattern. A Model View Controller pattern is made up of the following three parts:

- **Model** - The lowest level of the pattern which is responsible for maintaining data.
- **View** - This is responsible for displaying all or a portion of the data to the user.
- **Controller** - Software Code that controls the interactions between the Model and View.

MVC is popular as it isolates the application logic from the user interface layer and supports separation of concerns. Here the Controller receives all requests for the application and then works with the Model to prepare any data needed by the View. The View then uses the data prepared by the Controller to generate a final presentable response. The MVC abstraction can be graphically represented as follows.

Pipe-And-Filter

A very simple, yet powerful architecture, that is also very robust. It consists of any number of components (filters) that transform or filter data, before passing it on via connectors (pipes) to other components. The filters are all working *at the same time*. The architecture is often used as a simple sequence, but it may also be used for very complex structures.



The **filter** transforms or *filters* the data it receives via the pipes with which it is connected. A filter can have any number of input pipes and any number of output pipes.

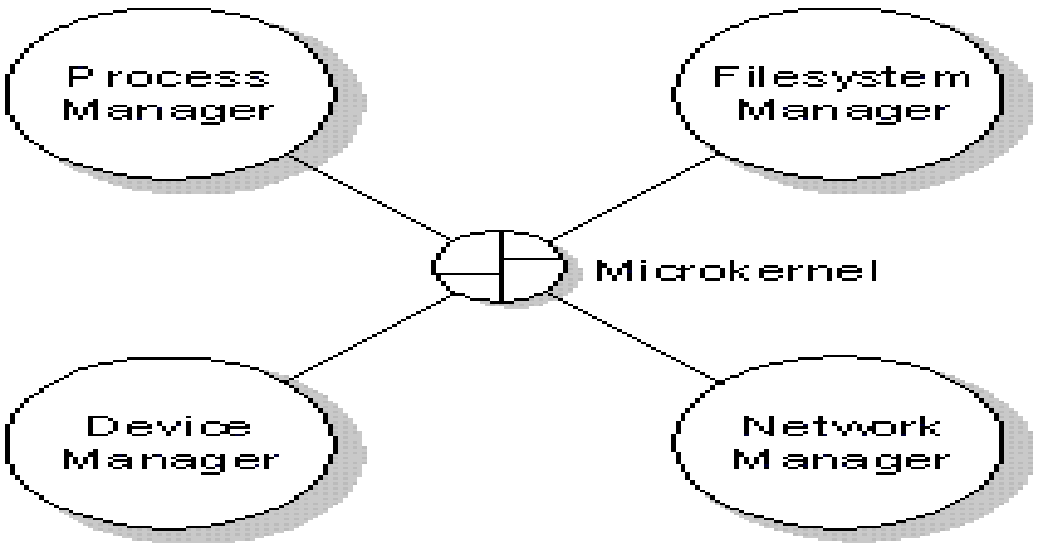
The **pipe** is the connector that passes data from one filter to the next. It is a directional stream of data, that is usually implemented by a data buffer to store all data, until the next filter has time to process it.

The **pump** or producer is the data source. It can be a static text file, or a keyboard input device, continously creating new data.

The **sink** or consumer is the data target. It can be a another file, a database, or a computer screen.

MicroKernal

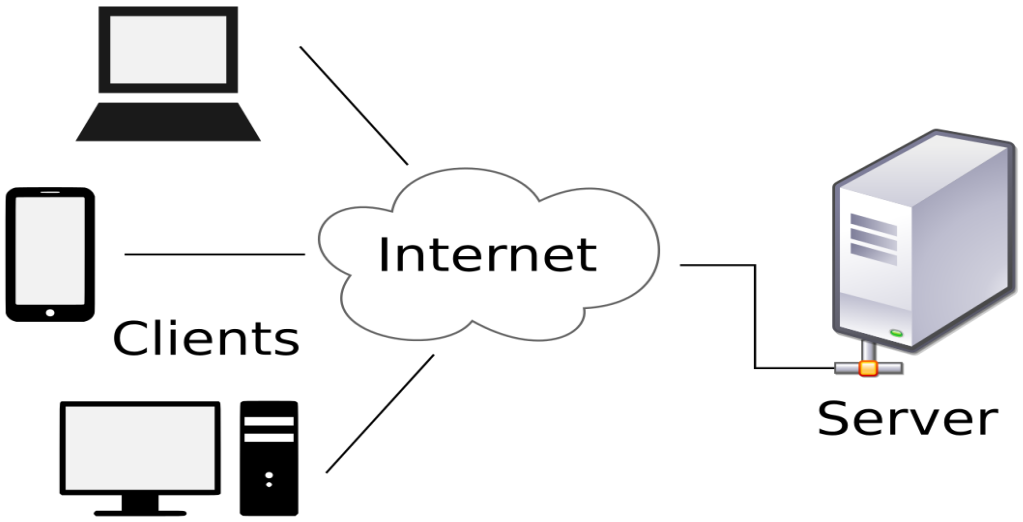
The Microkernel architectural pattern applies to software systems that must be able to adapt to changing system requirements. It separates a minimal functional core from extended functionality and customer-specific parts. The microkernel also serves as a socket for plugging in these extensions and coordinating their collaboration



Client-Server Model

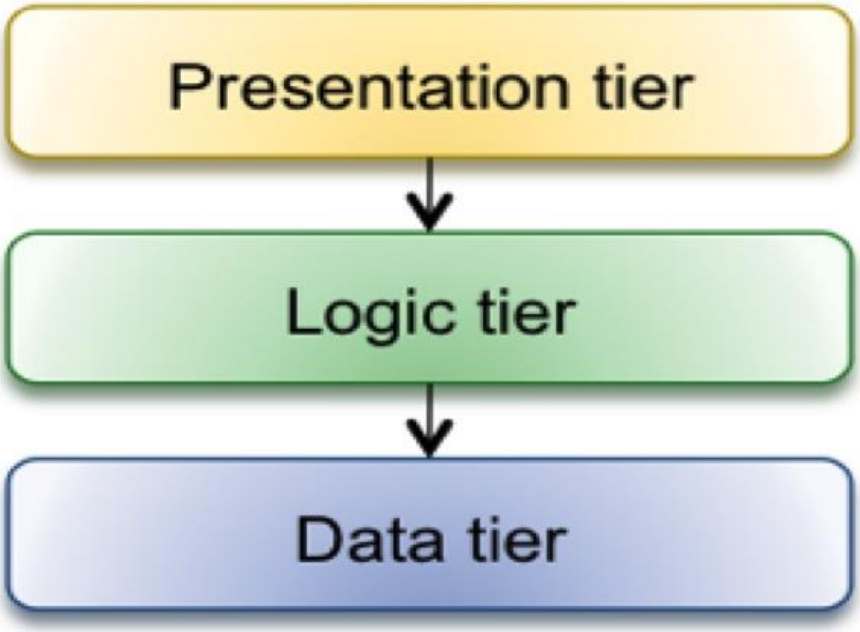
is a distributed application structure that partitions tasks or workloads between the providers of a resource or service, called servers, and service requesters, called

clients.Often clients and servers communicate over a computer network on separate hardware, but both client and server may reside in the same system. A server host runs one or more server programs which share their resources with clients. A client does not share any of its resources, but requests a server's content or service function. Clients therefore initiate communication sessions with servers which await incoming requests. Examples of computer applications that use the client–server model are Email, network printing, and the World Wide Web.



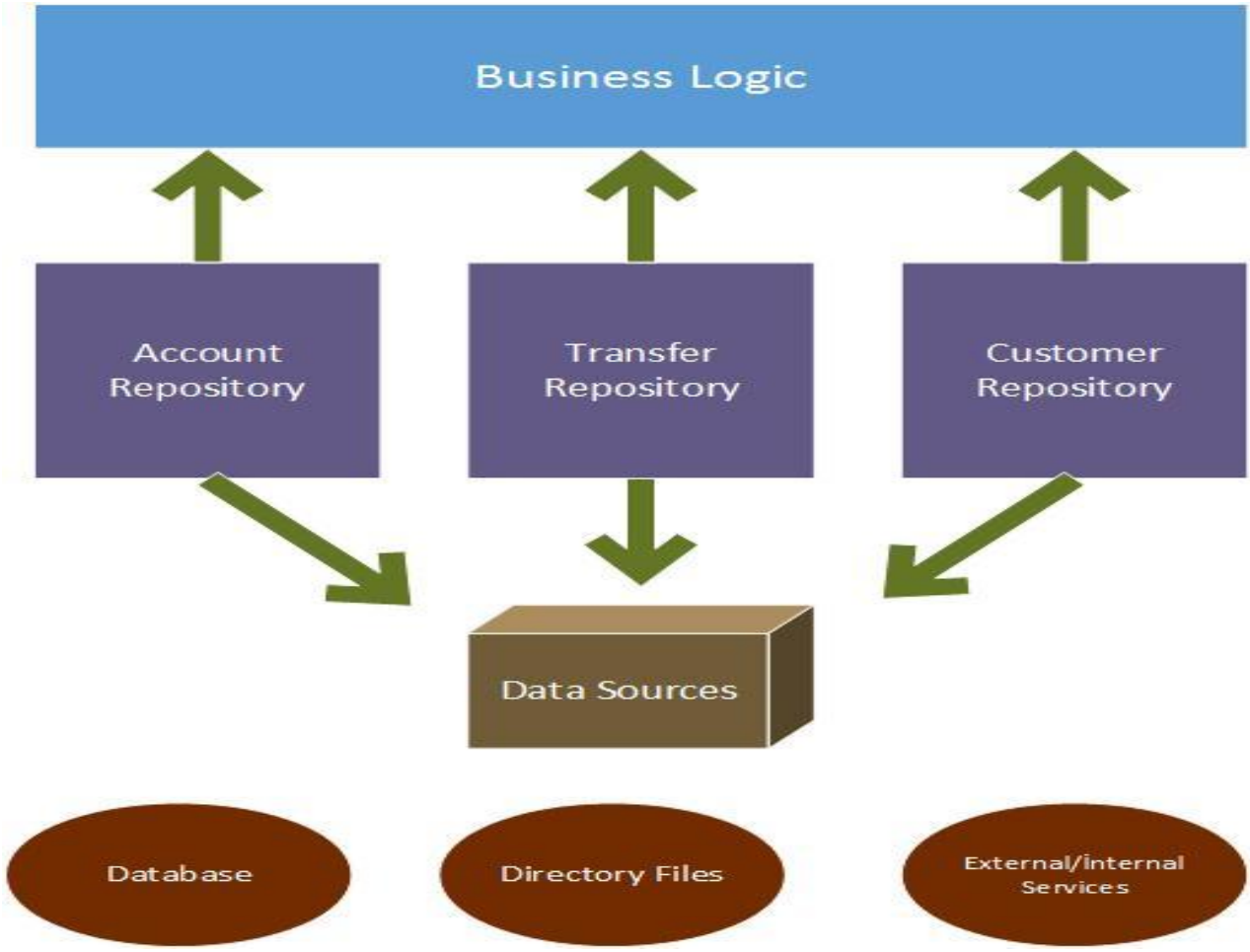
N-Tier

In software engineering, multi-tier architecture (often referred to as n-tier architecture) is a client-server architecture in which, the presentation, the application processing and the data management are logically separate processes. For example, an application that uses middleware to service data requests between a user and a database employs multi-tier architecture. The most widespread use of "multi-tier architecture" refers to three-tier architecture.



Repository Architecture

A repository architecture is a system that will allow several interfacing components to share the same data. Each component interfaces the same dataset that is utilized system wide. Data manipulation taking place in one component will reflect an identical representation of data in another component. Components can be interchanged and are independent of other system components. A good example of a repository architecture would be a database management system. Such a system would provide both a console and graphical user interface to update both the structure and dataset of any particular database.

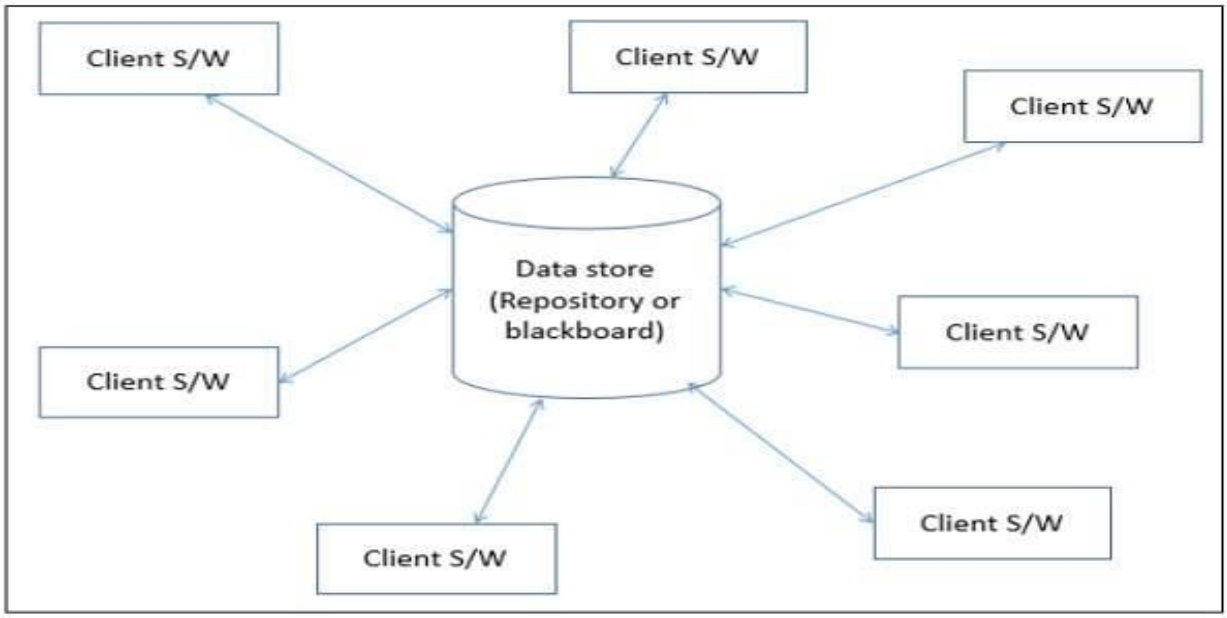


Blackboard Pattern

In software engineering, the **blackboard pattern** is a behavioral design pattern that provides a computational framework for the design and implementation of systems that integrate large and diverse specialized modules, and implement complex, non-deterministic control strategies.

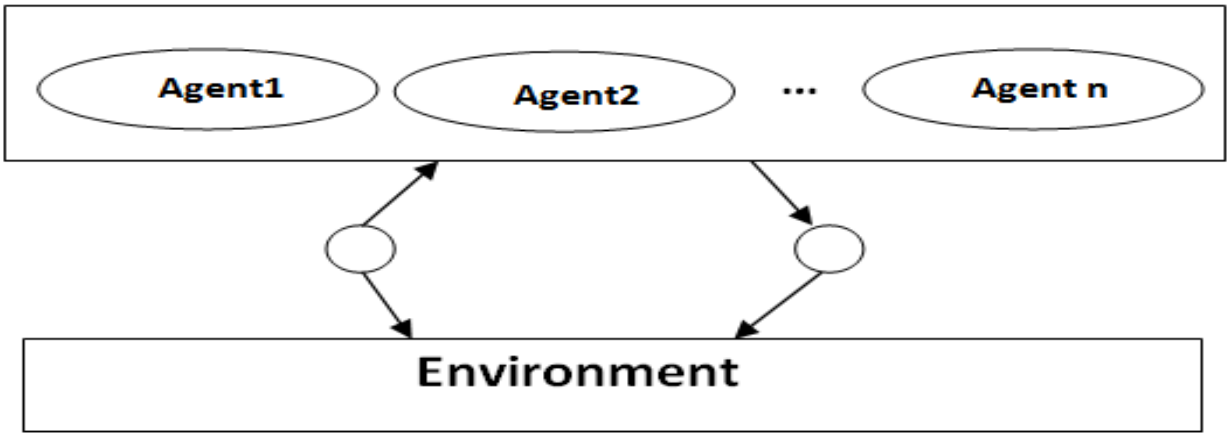
The blackboard model defines three main components:

- blackboard - a structured global memory containing objects from the solution space
- knowledge sources - specialized modules with their own representation
- control component - selects, configures and executes modules.



Multi-Agents System

is a computerized system composed of multiple interacting intelligent agents within an environment. Multi-agent systems can be used to solve problems that are difficult or impossible for an individual agent or a monolithic system to solve. Intelligence may include some methodic, functional, procedural approach, algorithmic search or reinforcement learning. Although there is considerable overlap, a multi-agent system is not always the same as an agent-based model (ABM). The goal of an ABM is to search for explanatory insight into the collective behavior of agents (which don't necessarily need to be "intelligent") obeying simple rules, typically in natural systems, rather than in solving specific practical or engineering problems. The terminology of ABM tends to be used more often in the sciences, and MAS in engineering and technology. Topics where multi-agent systems research may deliver an appropriate approach include online trading,[2] disaster response, and modelling social structures.



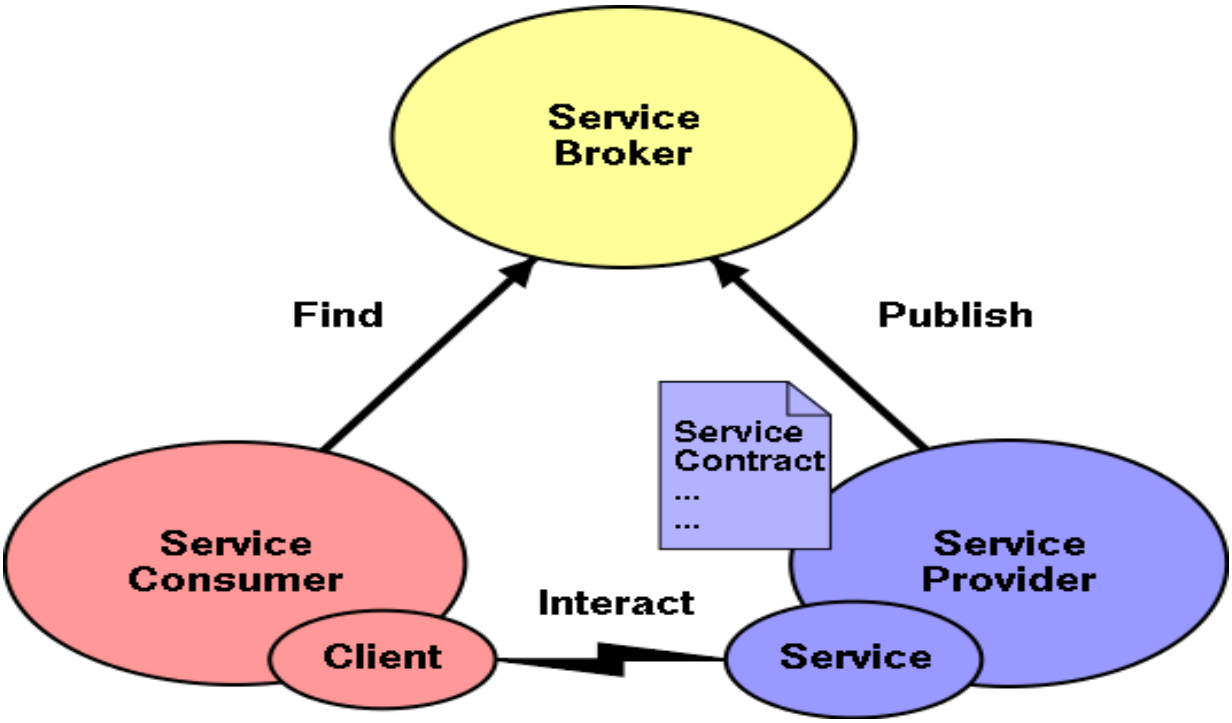
Service-Oriented-Architecture

is a style of software design where services are provided to the other components by application components, through a communication protocol over a network. The basic principles of service-oriented architecture are independent of vendors, products and technologies.[1] A service is a discrete unit of functionality that can be accessed remotely and acted upon and updated independently, such as retrieving a credit card statement online.

A service has four properties according to one of many definitions of SOA:[2]

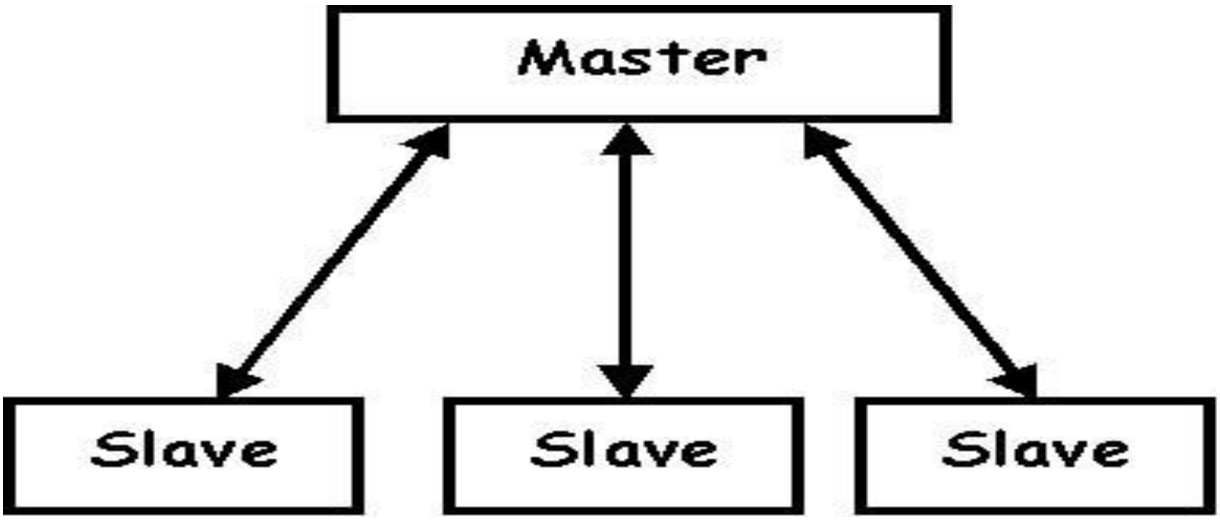
1. It logically represents a business activity with a specified outcome.
2. It is self-contained.
3. It is a black box for its consumers.
4. It may consist of other underlying services.[3]

Different services can be used in conjunction to provide the functionality of a large software application. So far, the definition could be a definition of modular programming in the 1970s. Service-oriented architecture is less about how to modularize an application, and more about how to compose an application by integration of distributed, separately-maintained and deployed software components. It is enabled by technologies and standards that make it easier for components to communicate and cooperate over a network, especially an IP network.



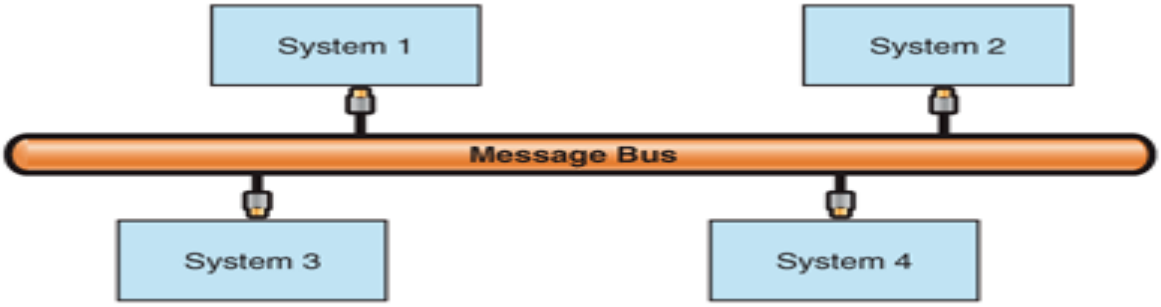
Master-Slave

In computer networking, master/slave is a model for a communication protocol in which one device or process (known as the *master*) controls one or more other devices or processes (known as *slaves*). Once the master/slave relationship is established, the direction of control is always from the master to the slave(s).



Message Bus

An architecture style that prescribes use of a software system that can receive and send messages using one or more communication channels, so that applications can interact without needing to know specific details about each other



Peer-to-Peer (P2P)

is a distributed application architecture that partitions tasks or workloads between peers. Peers are equally privileged, equipotent participants in the application. They are said to form a peer-to-peer network of nodes.

Peers make a portion of their resources, such as processing power, disk storage or network bandwidth, directly available to other network participants, without the need for central coordination by servers or stable hosts.[1] Peers are both suppliers and consumers of resources, in contrast to the traditional client-server model in which the consumption and supply of resources is divided. Emerging collaborative P2P systems are going beyond the era of peers doing similar things while sharing resources, and are looking for diverse peers that can bring in unique resources and capabilities to a virtual community thereby empowering it to engage in greater tasks beyond those that can be accomplished by individual peers, yet that are beneficial to all the peers.