# Application Layer

## Application Layer Introduction

* Layer 5 in the internet model
* The software that enables users to interact with the network and accomplish tasks
* This is the software that provides business value.
* There are five fundamental types of application architectures used at the application layer: host-based, client-based, client-server, cloud-based, peer-to-peer

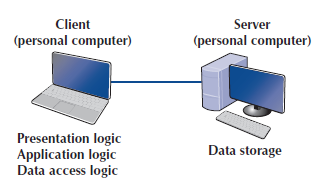
## Application Architecture

* **Application architecture** = the way in which the functions of the application layer software are spread among the clients and servers in the network
* The work done by any application program can be divided into these four general functions…
  + **Data storage** = most application programs require data to be stored and retrieved.
  + **Data access logic** = the logic processing required to access data, which often means database queries in SQL
  + **Application logic** = deals with information processing; can be simple or complex, depending on the application
  + **Presentation logic** = deals with presentation of information to the user and the acceptance of the user’s commands.

## Host-Based Architecture

* Common in the 1960s with mainframe computers (acts as servers) and terminals (acts as clients)
* Server performs all four functions (“server based”)
* Clients merely capture keystrokes, send them to the server for processing, and accept instructions from server on what to display
* Often works very well because there is only one point of control
* Two key disadvantages are that the servers can become overloaded if there are too many network applications and upgrading the mainframe servers is very costly.

## Client-Based Architecture

* Most common in the 1980s, when there was an explosion in the use of personal computers.
* Clients are personal computers on a LAN; application software is responsible for presentation, application, and data access logic
* Server is usually another PC on the same network; only tasked with data storage
* Also works very well most of the time: simple architecture, plus hardware and applications are less expensive
* Fundamental problem is that all data on the server must travel to the client for processing; can overload network circuits

## Client-Server Architecture

* Most common architecture today
* Tries to balance processing between client & server by having both do some functions
* Client is responsible for presentation logic
* Server is responsible for data access logic and data storage
* Application logic may reside on the client, on the server, or be split between both
* Strengths of client-server networks include more efficiency because of distributed processing, allows software and hardware from different vendors to be used together, and less bandwidth required
* Disadvantages are that it may be challenging in configuring all the different software/hardware to work together
* **Middleware** is often needed to bridge the gap between different software/hardware
  + Software that acts as intermediary by “sitting between” client and server applications
  + Provides standard way of translating between software from different vendors
  + Manages message transfers
  + Insulates network changes from the clients (e.g., adding a new server)
  + Dozens of standards for middleware; e.g. Distributed Computing Environment (DCE) and Common Object Request Broker Architecture (CORBA)

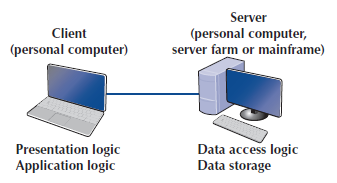


## Tiered Client-Server Architecture

* There are many ways in which the application logic can be split between the client and the server.
* Advantages of tiered architecture include efficient load balancing and it’s easier to scale up
* Disadvantages are that each tier increases network load, and it becomes more complex and difficult to program and test applications

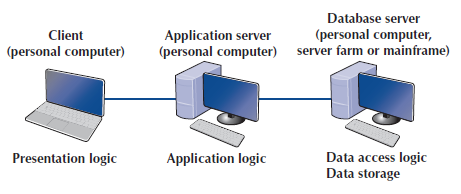
### Two-tier architecture

* + - Server is responsible for the data (data storage + data access logic)
    - Client is responsible for application and presentation logic
    - Uses only two sets of computers (i.e. 1 set of clients, 1 set of servers).



### Three-tier architecture

* + - Uses 3 sets of computers
    - An application server is responsible for application logic
    - Client computer’s software is responsible for presentation logic
    - A separate database server is responsible for the data access logic and data storage.

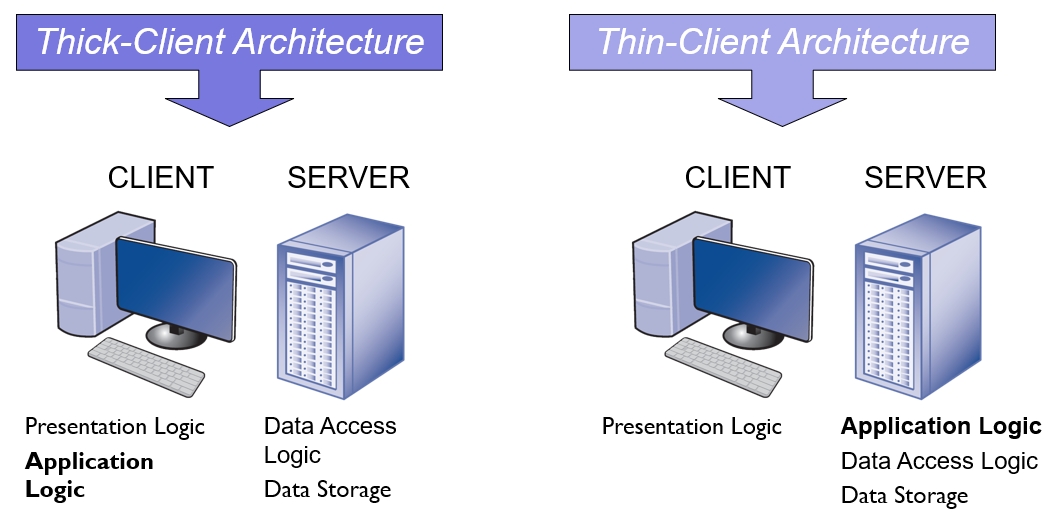


### N-tier architecture

* + - Uses more than 3 sets of computers
    - A set of 2 or more different servers deals with application logic
    - The client is responsible for presentation logic
    - A database server is responsible for data access logic and data storage

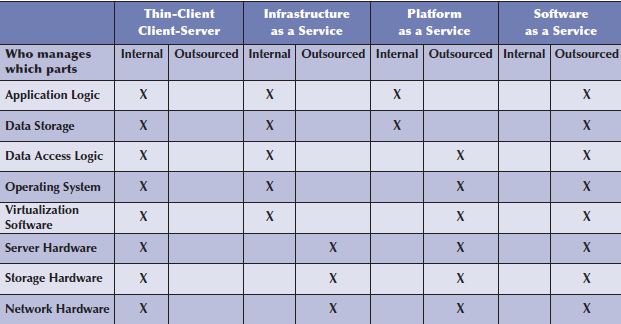
## Thick vs Thin Clients

* Another way of classifying client-server architecture is by examining how much of the application logic is placed on the client computer.
  + **Thick-client** (or **fat-client**) approach places all or almost all of the application logic on the client; has more functionality but is not easy to manage; e.g. if any application change occurs, software on all clients need to be changed
  + **Thin client architecture** places little or no application logic on the client; easier to manage because if an application changes, only the server with the application logic needs to be updated; enables **cloud-architecture**!



## Cloud Computing Architecture

* Involves outsourcing computer services/infrastructure over the network (most commonly the internet) to other firms that specialize in managing that infrastructure
  + Specialized software is often required; servers need an operating system (e.g. Window, Linux); virtualization software to install virtual/logical servers is also common.
  + All this software must run on some hardware, including a server, storage device, and the network itself; **server farms**, i.e. a cluster of computers linked together to act as one computer, may be used in place of one server, as it helps with overloading and computer failure.
  + Storage devices may also be separate from the server’s hard disks; purposely designed to be very large and very fast; an example is a storage area network.
* Key Advantages:
  + Huge resources: IT agility as systems can be sized easily to meet demand
  + No commitment: there is no longer the tradeoff between overprovisioning (waste of capital) and underprovisioning (waste of users)
  + Pay as you go: move IT payments from capital expenditures or CAPEX (i.e. long term expenses) to operating expenditures or OPEX (i.e. day to day expenses); pay only for actual resources consumed; tie IT cost to business benefits received
* Models of cloud computing define who manages each app function and associated hardware/software



### Infrastructure as a Service (IaaS)

* + - Often known as Hardware as a Service (HaaS) because only the hardware is provided.
      * Cloud provider manages the hardware, including servers, storage, and networking components.
      * Organization is responsible for providing all the software, including operating system (and virtualization software), database software, and its apps and data.
    - Allows for a decrease in capital expenditures for maintaining hardware, physical environment (e.g. electricity, cooling, spaces) and backups for data and apps.
    - Providers of IaaS include Amazon Web Services (AWS), Windows Azure, and Akamai.

### Platform as a Service (PaaS)

* + - With PaaS, you can build your own application and manage your own data on the cloud supplier’s infrastructure.
      * The organization develops the app and manages its own data, but uses database software and OS provided by the cloud
      * The cloud provider provides the platform (i.e. the needed hardware, software, network and database infrastructure)
    - PaaS offers a much faster development and deployment of custom apps at a fraction of cost required for traditional client-server architectures.
    - Providers include Amazon Elastic Cloud Compute (EC2), Windows Azure, and Google App Engine.

### Software as a Service

* + - An organization outsources the entire app to the cloud provider and uses it like any other app via a browser
    - SaaS is based on **multitenancy**, the idea that there is only one app shared by everyone but they can customize to their specific needs.
    - Providers include Google Gmail (email is the most common SaaS app) and Salesforce.com (which provides customer relationship management software)

## Peer to Peer (P2P) Architecture

* Very old architecture, but their modern design has become popular in the early 2000s with the rise of P2P file sharing apps (e.g. Napster, BitTorrent).
* All computers are both clients and servers and thus share the workload.
  + All computers can perform all four functions.
  + Arbitrary end systems (i.e. hosts) directly communicate
  + Peers are intermittently connected and change IP addresses
  + With a P2P file-sharing app, users can use the other three functions installed on his/her computer to access data stored on another computer in the network.
  + Users in the network can use others’ computers to access application logic too.
* Advantages: data can be installed anywhere in the network; spreading the storage throughout the network means that its resilient to failure in of any one computer; distributes bandwidth requirements.
* Disadvantages: it is hard to find the data, which is why P2P architectures are often combined with a client-server architecture; security is a major concern, as anyone can access the computers in the network through open ports.

## Summary of Application Architectures

* There are four fundamental application architectures.
  + In **host-based networks**, the server performs virtually all of the work.
  + In **client-based networks**, the client computer does most of the work; the server is used only for data storage.
  + In **client-server networks**, the work is shared between the servers and clients.
    - The client performs all presentation logic, the server handles all data storage and data access logic, and one or both perform the application logic.
    - Client-server networks can be cheaper to install and often better balance the network loads but are more complex to develop and manage.
    - **Cloud computing** is a form of client-server architecture.
  + With **peer-to-peer networks**, client computers also play the role of a server.