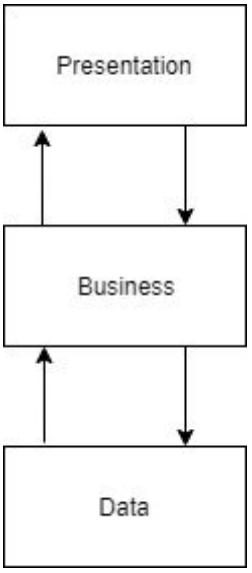
Software Architecture Specification
Chapter 1

3-Tier (N-Tier) Architecture

3-Tier architecture is a type of client-server architecture in which different layers are developed as modules separate from each other, the tiers are utilized as follows:

- **Presentation**
  - Hosts User Interface/Experience of any kind. Allowing users to directly access the application - usually through some type of GUI.
- **Business**
  - Sometimes referred to as the Application layer.
  - Processes any algorithms which define the application.
- **Data**
  - Contains any data persistence (servers, files, etc.) and provides access to such data to specific layers.

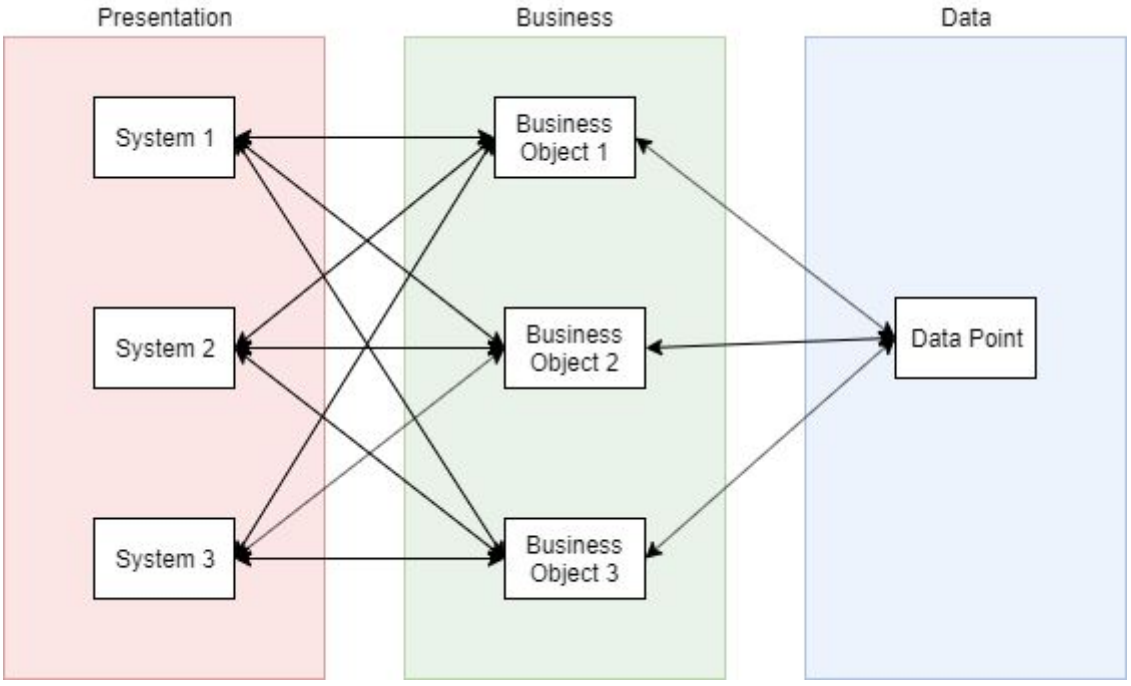


As shown, only the Business layer has access to both other layers. The flow of use is as follows:

1. A user/client will utilize the Presentation layer by some form of UI to request a service.
2. The Presentation layer will then request the Business layer to acquire any required data from the Data layer
3. The Data layer will supply only the requested data to the Business layer
4. The Business layer will then process the data via any algorithms that have been specified by the Presentation layer and then return the result to the Presentation layer.

As a client-server type architecture, a client, of which there can be many will request action from a singular server which will then in turn fulfill the request. In N-tier the Presentation layer takes the role of the Client and both the Business and Data layers take the role of Server, although these two do not have to be the same server in this scenario.

In a designed architecture for DE-Store, multiple systems would run the presentation layer software to allow users to interact with the network. Each system would connect to the respective business layer object to acquire and process any required data such as stock levels and pricing.



3-Tier architecture is a potential selection for DE-Store for multiple reasons:

- **Scalable**
    - As the components of the architecture are separated, each component can be easily upgraded with low/no impact on the other components. For example, a growing stock may require more storage to keep track of or a RAID type system may be implemented for data security - the Data layer can be upgraded with no impact to the Business or Presentation.
- **Resilience**
    - As the layers are separated, they can be kept geographically separate. Should a failure occur in the form of damage to the location or power outages, only the layer which is at said location would be physically

affected - this would in turn disable the other layers from functioning correctly but only one layer would need to be repaired.

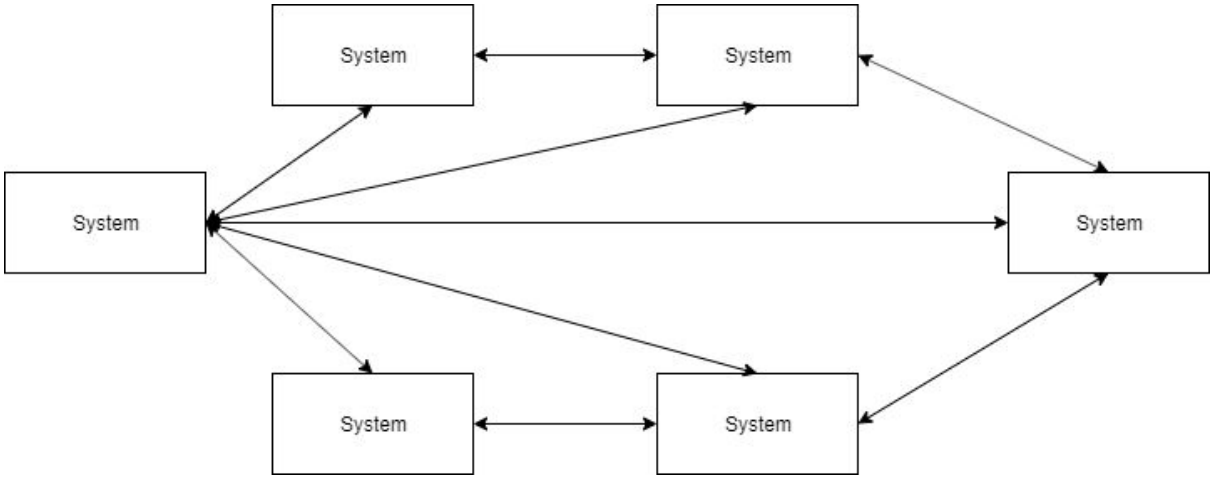
- **Developer Oriented**
    - In the scenario that DE-Store's distributed system was being developed between multiple developers, tasks can easily be assigned by layers once a structured view of the architecture has been created allowing the individual to focus on their task without resources being shared.
    - In a single developer scenario - N-Tier can assist in providing a structured view and allow a developer to view a project as if many were in the place by assigning tasks to specific layers.

However there are several disadvantages to this architecture methodology:

- **Bottlenecks**
    - Should layers be separated, the processing power of the Business layer may be greater than the Data layer's ability to send data to the Business layer creating a bottleneck in operating speeds.
    - Network speeds could also negatively affect transfer rates for information from layers to layers.
- **Layer Security**
    - Efforts must be taken to ensure users of differing rights only have access to their permitted layers, a user should have no direct access to the data layer

Peer-to-Peer (P2P) Architecture

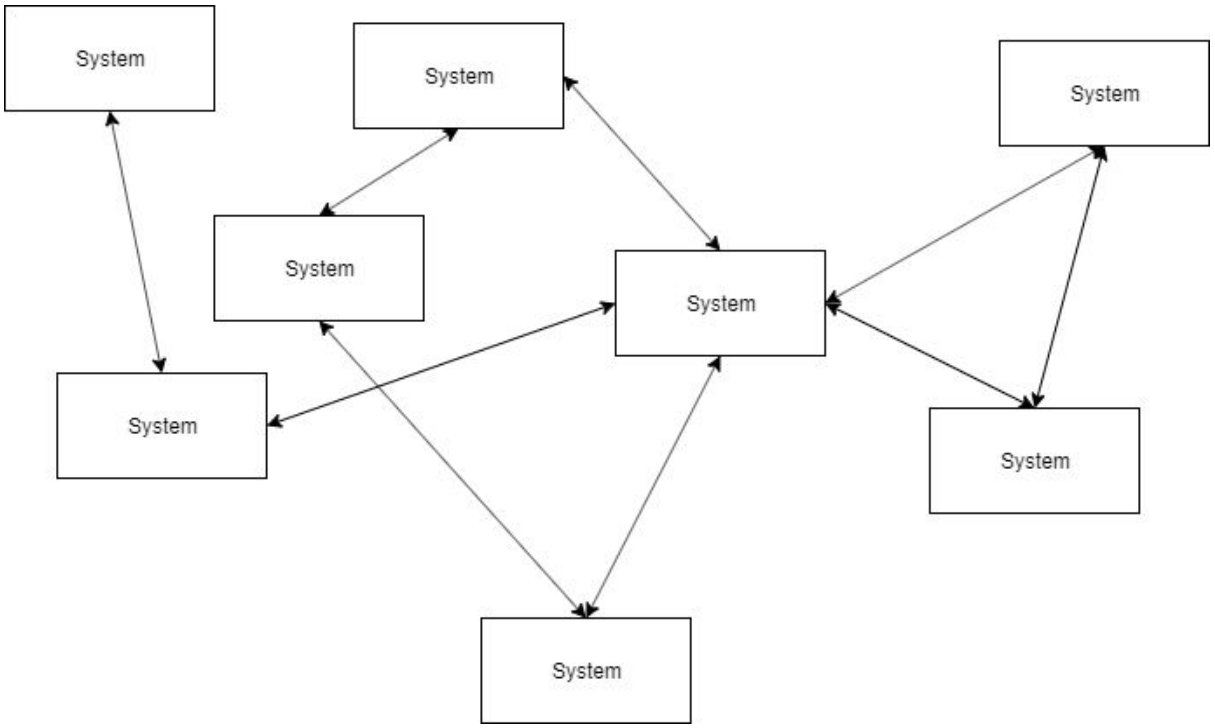
In client-server architecture, the server is a service provider and the client is a service consumer. However in Peer-to-Peer architecture each system can be both a server and a client which decentralises control and decreases dependency on a central unit making the network autonomous. This is done several systems joining to make a network with no central unit - each system on the network takes on both roles as client or server for differing tasks.



Peer-to-Peer networks can still contain servers to some degree. A pure Peer-to-Peer network will be truly autonomous with no central server in which peers designate self rules and environments, however the network can implement a Discovery and Lookup server which contains a list of connected peers and the available resources - however to remain as a Peer-to-Peer network all peers collaborate and have a degree of control over this server. This server may also track data transfers by pushing and pulling data from peers to other peers.

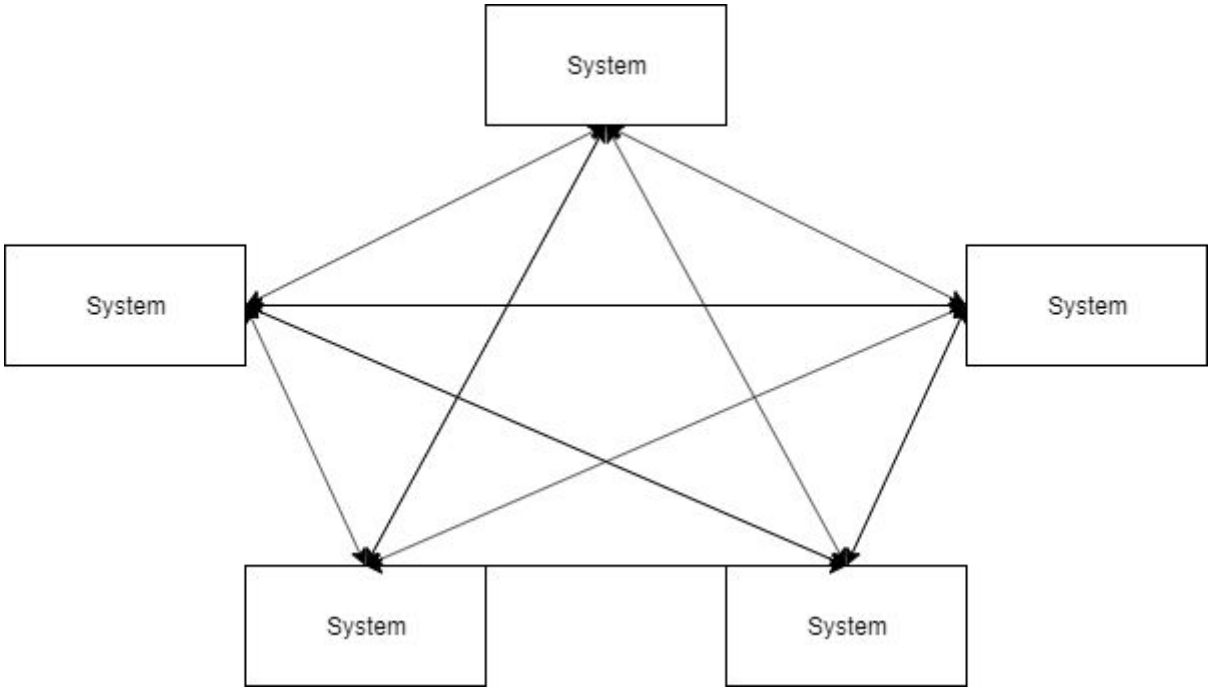
The Peer-to-Peer network relies on voluntary resource sharing in most scenarios, but in a distributed system specifically designed for DE-Store this would not be an issue.

Networks can be designed structured or unstructured, in unstructured networks the connections are random and difficult to visualise which can create slower transfers between machines with many differing connections between the two:



As shown, the number of connections to each system can vary greatly - usually systems which are part of the network for longer will have more connections.

Structured networks is a much more efficient for a network for DE-Store being a distributed system for a client. In structured networks systems are organised and connected with intent rather than ad-hoc which can allow faster transfers to each peer but increases efforts needed to implement new systems:



The number of connections for a structured network are exponential in size, for five systems as shown there are ten connections, but with seven systems there would be an extra two connections (six connections per machine) for each system plus the connections for the new systems brining the number of connections up to forty-two. If DE-Store was planning to expand, this would have to be taken into consideration for a structured vs an unstructured network as scalability can become an issue with a limited bandwidth.

Advantages to a peer-to-peer network include:

- **Ease of Setup**
    - Only systems are required, no main server must be implemented to the system, but a lookup server could be implemented with relative ease due to the functions required to locate data held by other peers being simplistic.
- **Price friendly**
    - No main server must be purchased as only systems will need to be acquired for a network to be established
- **Ease of small scale management**
    - On a smaller, internal scale systems can be maintained easily as individuals.

Several disadvantages are brought upon by a Peer-to-Peer network however:

- **Security**
    - As each system plays part of a server and client role, a virus can easily spread throughout the network if sent data is not properly checked.
- **Memory**
    - As the network is decentralized, it is difficult to make a back-up of any important information
- **Scalability**
    - With a structured network like advised for DE-Store, it becomes difficult to increase the network size with an exponential number of connections required.

Selection: 3-Tier

3-Tier has been selected from the two described architectures for several reasons:

- **Developer Friendly**
    - With the increased ease of task allocation, the architecture can help increase the rate of development speed and robustness of the system desired.
    - Should any changes be needed to improve the system at a later date, the layered design allows ease of improvement to systems within each layer without affecting the other layers, this would also allow a degree of ease of understanding the distributed system as a whole to a new developer should this be the case.
- **Scalability**
    - Should DE-Store become a popularized distributed system, increasing capacity of each layer to allow greater use is far easier compared to a peer-to-peer network.

System Design Overview

As a 3-Tier architecture, DE-Store would be designed as follows:

- **Presentation**
    - The presentation layer would host a UI for store managers which allows stock to be selected with the ability to change stock pricing, and quantity, should quantity of stock be low - a warning is displayed, should stock quantity equal zero - more stock is ordered
    - Managers can also add rules to the loyalty card in the form of offers (buy one get one free, discount on bulk buys, etc.)
    - Finance approval will be a selectable feature for orders received, managers can review orders and approve or remove orders
    - Purchase history, a log of customer purchases is kept for manager analysis.
- **Business**
    - Processes the requests from presentation and verifies orders
- **Data**
    - Hosts data such as stock quantity and customer order logs

Evaluation