LE 0:

Distributed applications could simply be developed based on “sockets”. This would entail that developers worry about:

* Fallacies of distributed systems development
* Interoperability in spite of heterogeneity
* Data representation and encoding

Middleware layer:

* Middleware as layer **between transport & application layer** or **between the operating system and the application**.
* Simplifies the development of distributed applications
* Offers services & ilities (reliability, availability, scalability,etc)
* Hides complexities from developer & user

Definitions:

"**Middleware** is the intersection of the stuff that network engineers don't want to do with the stuff that applications developers don't want to do.“

"Middleware today encompasses everything to the right of the clients and to the left of the legacy systems. It resides above the OS and below the application logic.“

Middleware comprises services and abstractions that facilitate the design, development, and deployment of distributed applications in heterogeneous, networked environments.

Middleware:

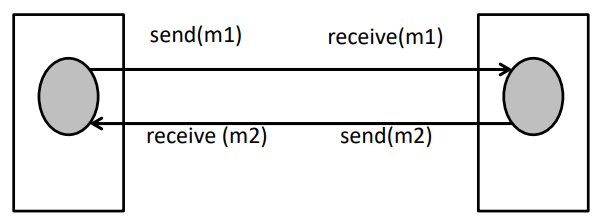
* Constitutes **building blocks**
* Captures **common** **functionalities**
  + Message passing, remote invocation
  + Message queuing, publish/subscribe
  + Transaction processing
* Deals with **interoperability**
* Deals with **system** **integration**

Heterogeneity

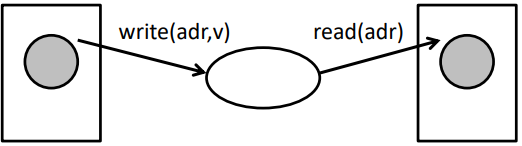
* Computer architecture
  + Ex: Number representations (big vs. little endian)
* Operating system
* Communications architecture
* Programming language
* Application programming interfaces

Types of middleware:

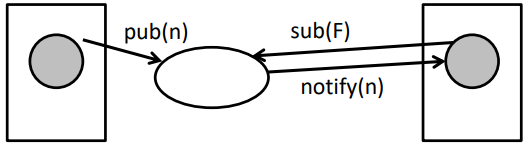
* Message passing:



* Virtual shared memory



* Publish/Subscribe



* Remote Invocation (RPC)

**8 fallacies:**

* **The network is reliable**
* **Latency is zero**
* Bandwidth is infinite
* **The network is secure**
* Topology doesn't change
* There is one administrator
* Transport cost is zero
* The network is homogeneous