Requirements

Transparency - the ability of the system to mask its complexity behind a simple interface

Possible transparencies

- Location - cannot tell where resources are located
- Migration - resources may move without the user knowing
- Replication - cannot tell how many copies of resource exist
- Concurrency - cannot tell how many users there are
- Parallelism - may speed up large jobs by splitting them into smaller pieces
- Fault Tolerance - system may hide various things that go wrong

➡ Transparency and collaboration require some way for different processors to communicate with one another

Clients and Servers

The prevalent model for structuring distributed computation is the client/server paradigm

➡ A **server** is a program (or collection of programs) that provide a service (file server, name service, etc.)

- The server may exist on one or more nodes
- Often the node is called the server, too, which is confusing

➡ A **client** is a program that uses the service

- A client first binds to the server (locates it and establishes a connection to it)
- A client then sends requests, with data, to perform actions, and the servers sends responses, also with data