

CSE 760

- Traditional Operating Systems deal with typical system software designed to be:
 - general purpose
 - running on single processor machines
- Advanced Operating Systems are designed for either a special purpose, or for more complex architectures
 - Special purpose o.s. open new design dimensions
 - examples: real-time systems, fault tolerant design, database systems
 - Complex architectures
 - Multiprocessor machines, distributed architectures (i.e. over a SAN, over the Internet, ...)
- In this course we will be dealing with some of the **conceptual issues** that arise in the design of system software for distributed and special purpose systems.

Course content and objectives

“CSE 760 is a course on advanced concepts and mechanisms in operating systems”

Course topics

Weeks	Topic
1	Intro
1	Process Synchronization
1	Mechanisms for Mutual Exclusion
1	Interprocess Communication
2	Distributed Mutual Exclusion
1	Deadlock
1	Fault-tolerance
1	Resource & Data Security
1	Distributed File Systems

Course objectives

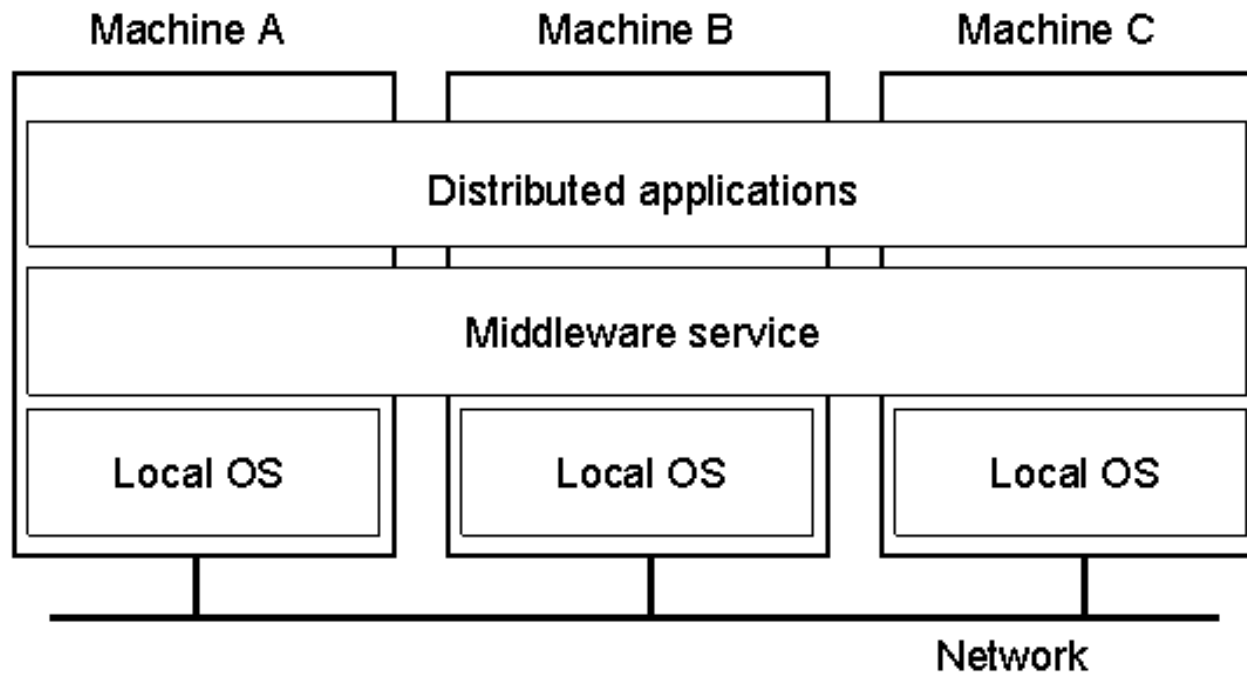
- to become aware of issues found in the design of distributed systems in general and the design of advanced operating systems in particular
- to become familiar with techniques that have been proposed for dealing with these issues.

Definition of a Distributed System (1)

- A distributed system is:

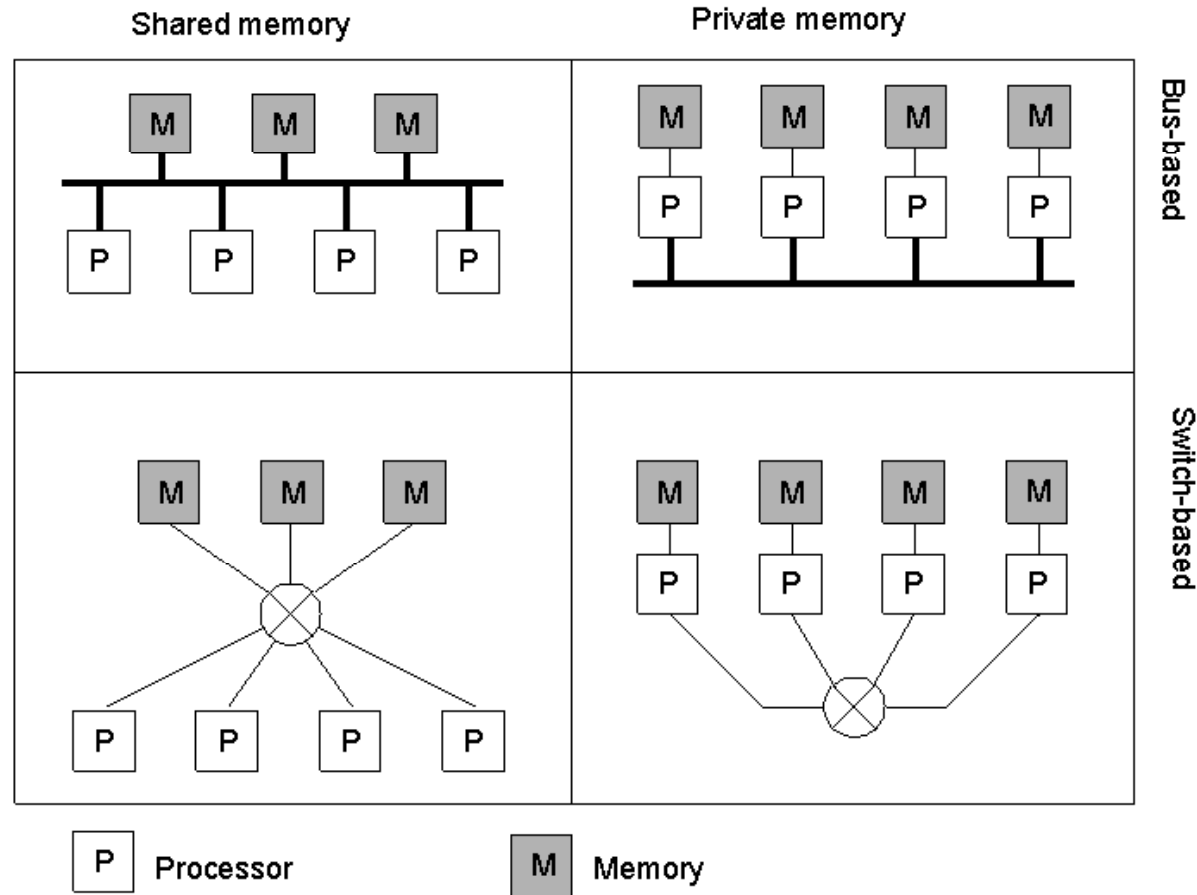
A collection of independent computers that appears to its users as a single coherent system.

Distributed System Organization



- Example of middleware-based organization of a distributed system.
- The thickness of the middleware layer can range from extremely thin to very thick depending on the degree of integration of a particular system

Hardware Concepts



Different basic organizations and memories in distributed computer systems

Distributed systems issues

- Distributed systems introduce a whole new set of design issues w.r.t traditional system design
- Scalability
- Transparency
- On multicomputers:
 - Lack of common address space
 - Lack of common clock

Transparency in a Distributed System

Transparency	Description
Access	Hide differences in data representation and how a resource is accessed
Location	Hide where a resource is located
Migration	Hide that a resource may move to another location
Relocation	Hide that a resource may be moved to another location while in use
Replication	Hide that there may be multiple copies of a resource
Concurrency	Hide that a resource may be shared by several competitive users
Failure	Hide the failure and recovery of a resource
Persistence	Hide whether a (software) resource is in memory or on disk

Different forms of transparency in a distributed system.

Scalability Problems

Concept	Example
Centralized services	A single server for all users
Centralized data	A single on-line telephone book
Centralized algorithms	Doing routing based on complete information

Examples of scalability limitations.