Distributed Systems Lecture 2

Professor Louise E. Moser

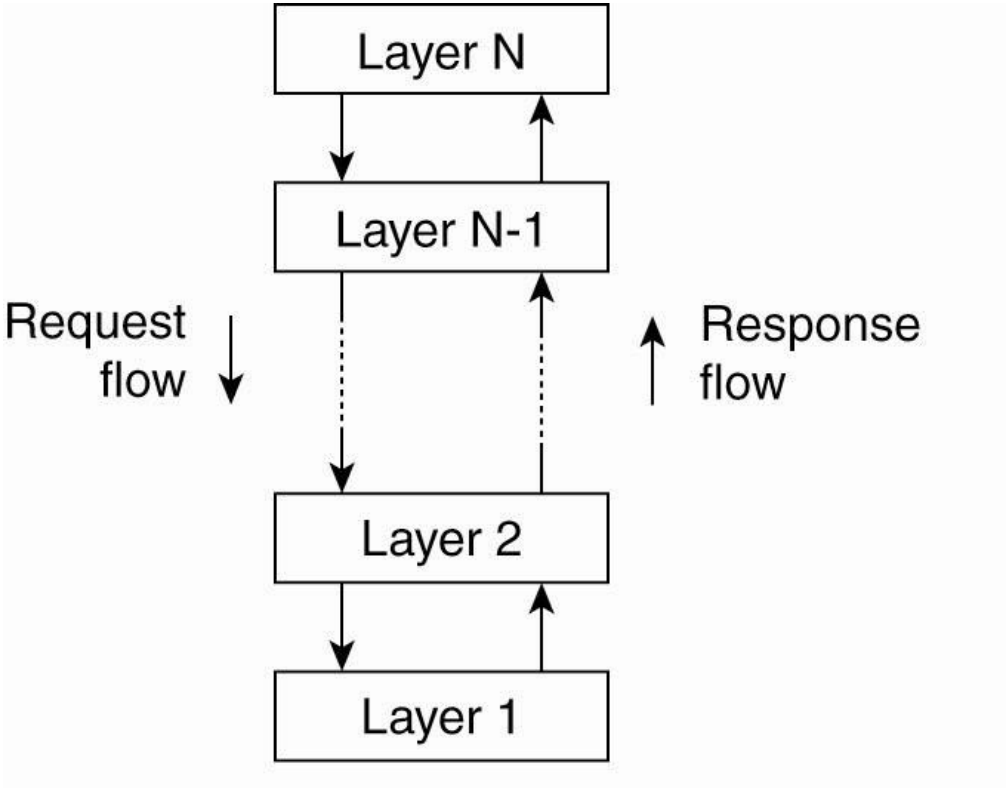
Spring 2013

Architectural Styles

- Important architectural styles for distributed systems
  - **Layered architecture**
  - **Object-based architecture**
  - **Event-based architecture**
  - **Shared data space architecture**

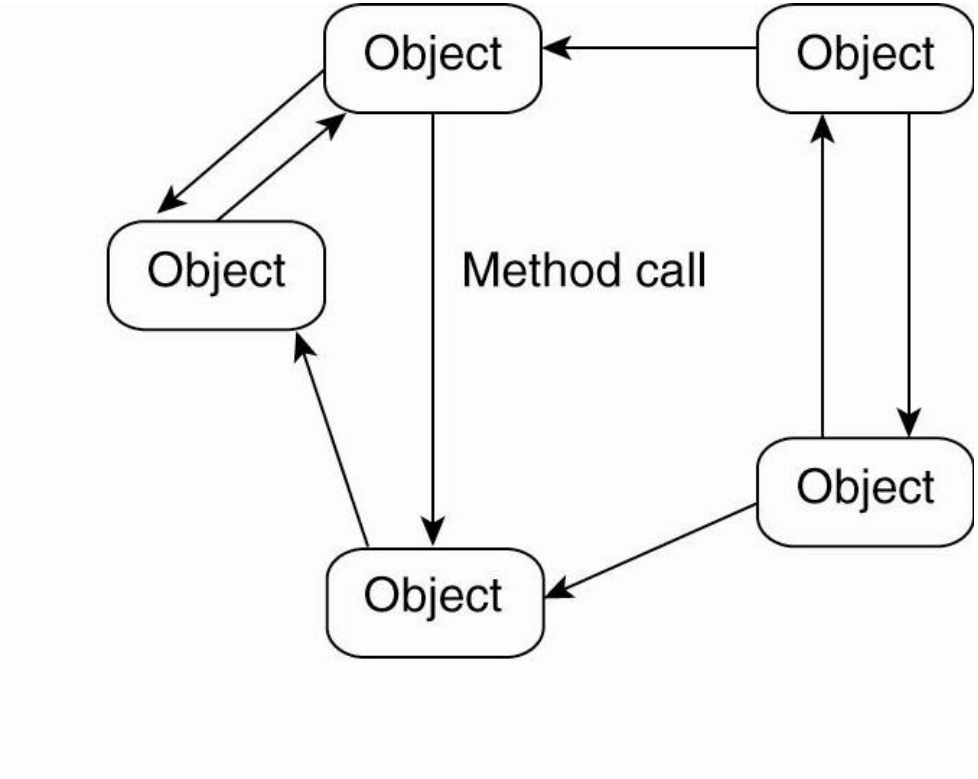
Architectural Styles

- Layered architecture



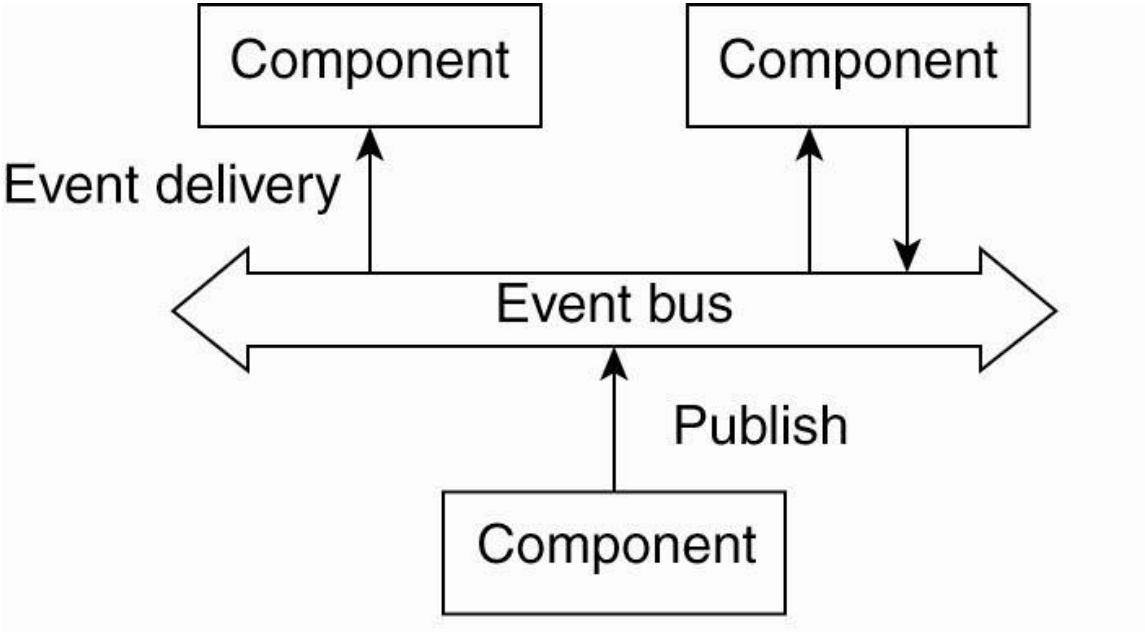
Architectural Styles

- Object-based architecture



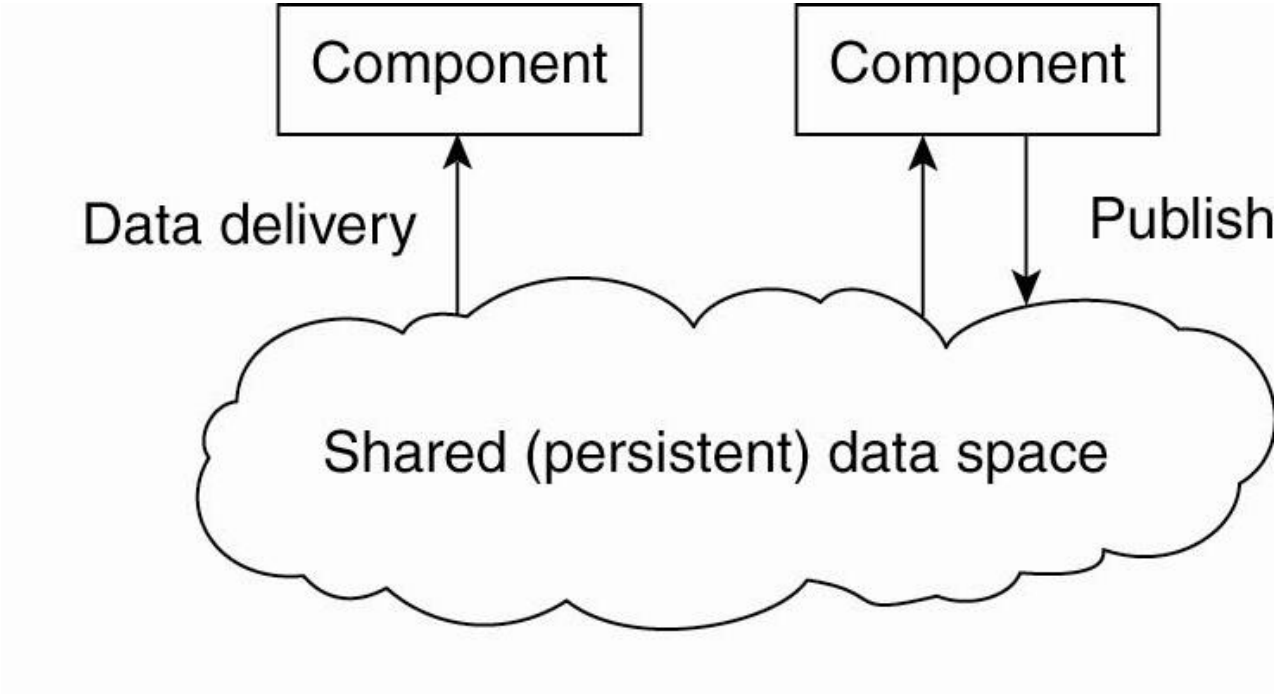
Architectural Styles

- Event-based architecture



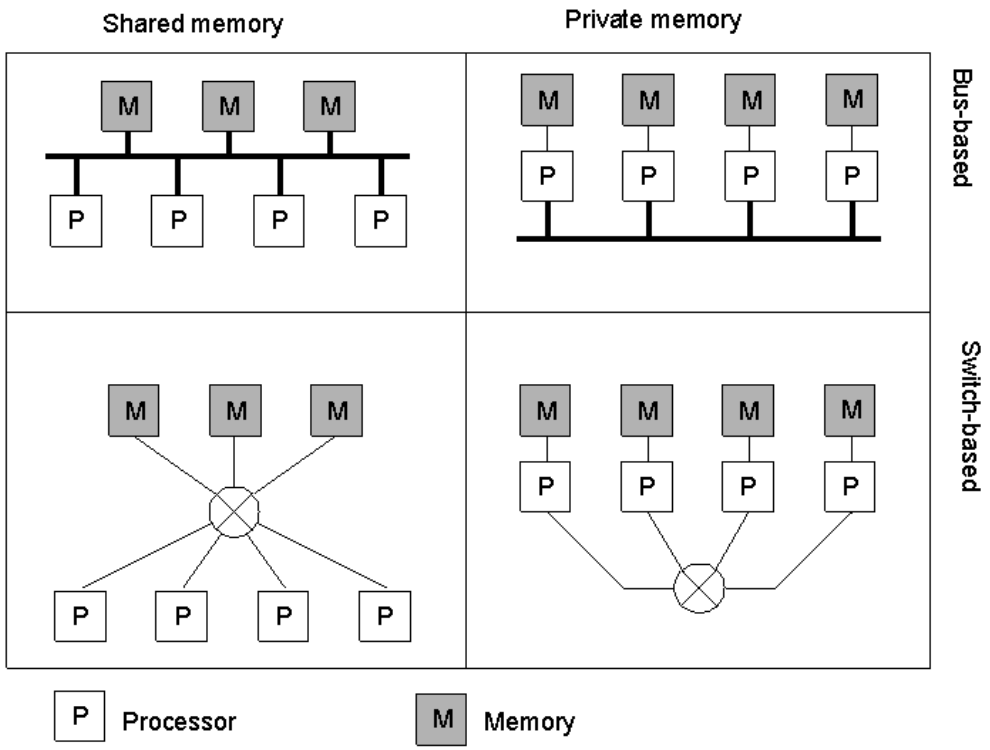
Architectural Styles

- Shared data space architecture



Architectural Organizations

- Different organizations of processors and memories in distributed systems



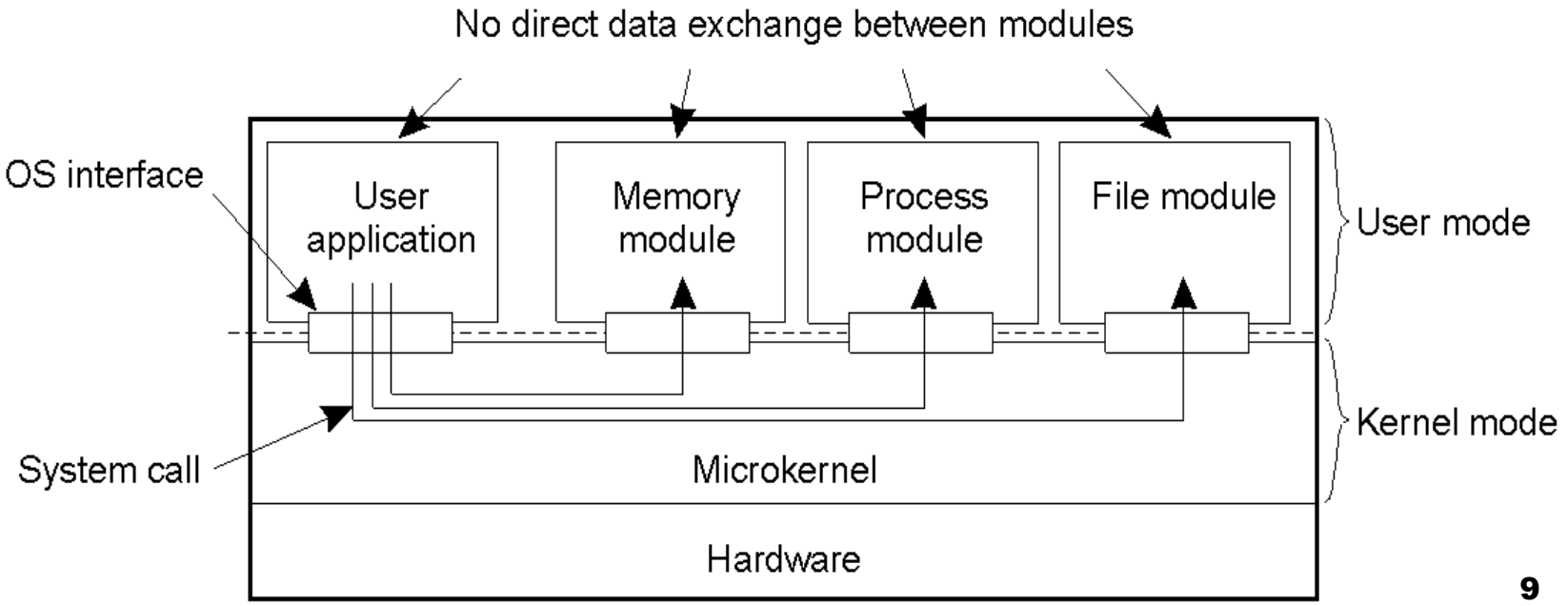
Distributed System Software

System	Description	Main Goal
DOS	Tightly-coupled operating system for shared memory multi-processors and homogeneous multi-computers	Hides and manages hardware resources
NOS	Loosely-coupled operating system for heterogeneous multi-computers (LAN and WAN)	Offers local services to remote clients
Middleware	Additional layer on top of NOS that implements general-purpose services	Provides distribution transparency

- Distributed Operating System (DOS)
- Network Operating System (NOS)
- Middleware

Uni-Processor Operating System

- Operating system manages memory pages, processes and threads, files for the user application
- User application itself does not directly exchange data with those modules

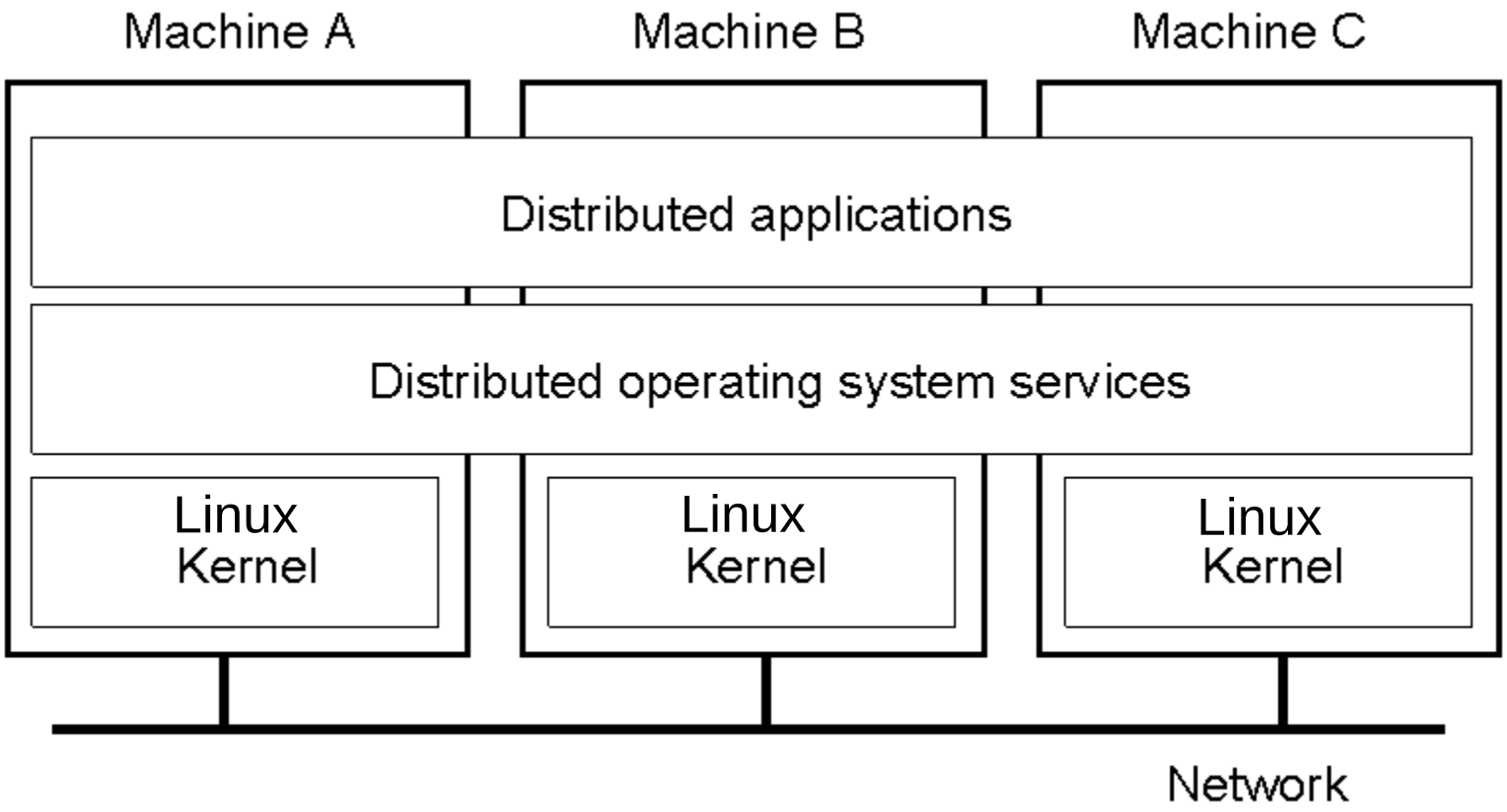


Distributed Operating System

- With **homogeneous multi-computers:**
  all computers in the network run the same operating system (e.g., all Windows or all Linux)
- No shared memory, communication by message passing
- Often, no simple system-wide communication mechanisms
  - **Only bus-based multi-computers provide HW broadcasting**
  - **Efficient broadcasting requires special network interface programming**
- No simple system-wide synchronization mechanisms
- Distributed (virtual) shared memory requires operating system to maintain a global memory map in software
- Distributed resource management requires distributed algorithms for resource allocation

Distributed Operating System

- Distributed operating system services

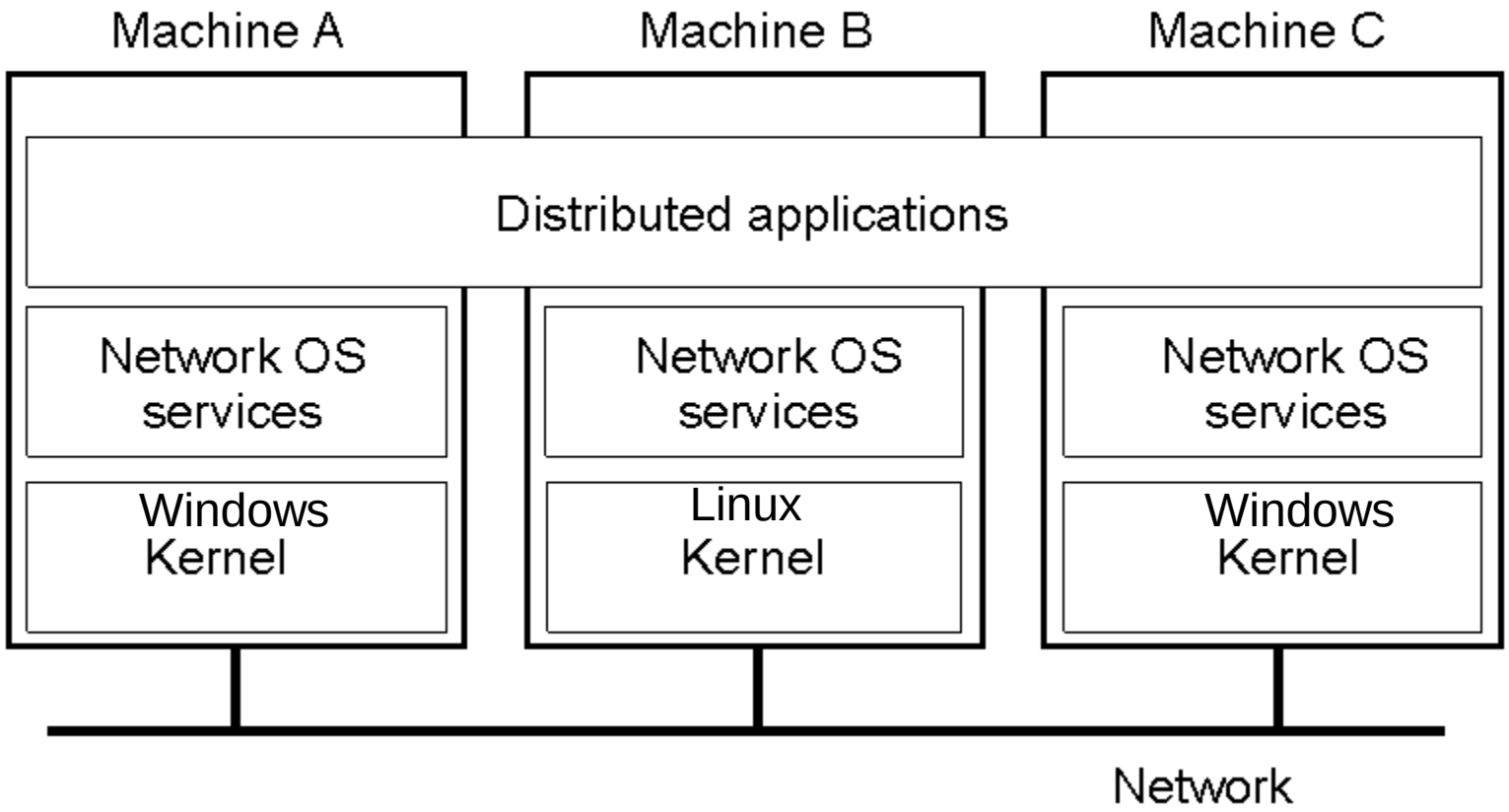


Network Operating System

- Multiple computers are networked together, but they work independently
- Each computer has its own operating system with networking facilities
  - **Different computers may have different operating systems, e.g., some run Windows while others run Linux**
- Services are tied to individual nodes in the network (ftp, telnet, WWW)
- Highly oriented towards files (basically, processors only share files)

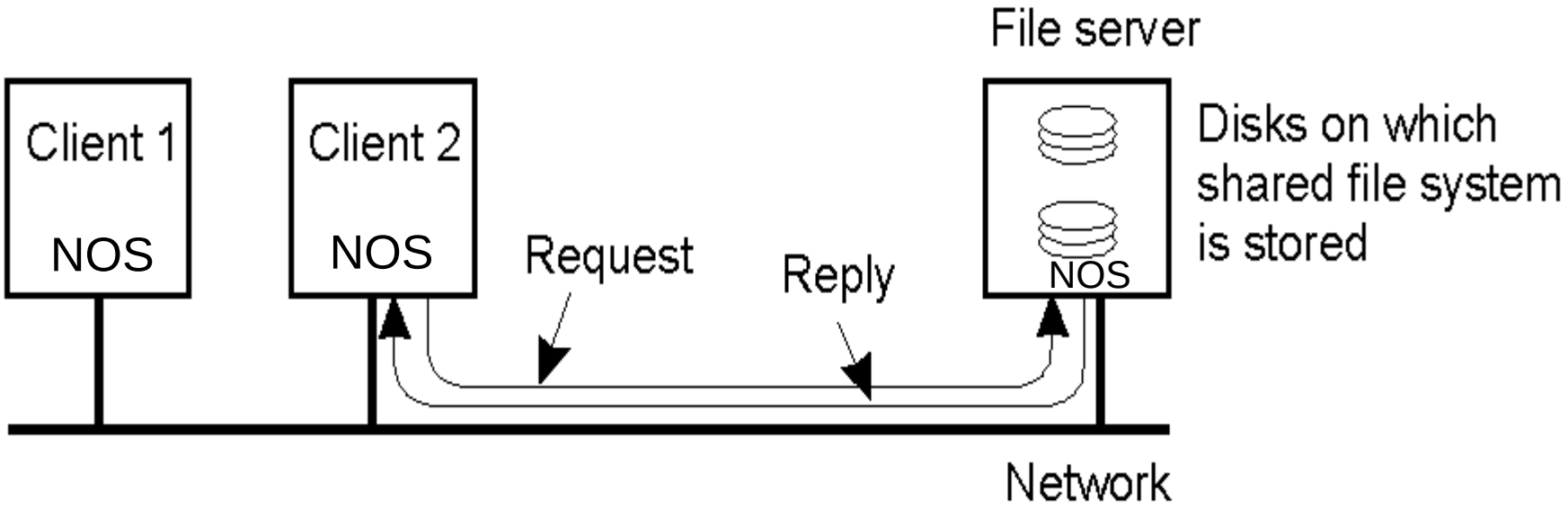
Network Operating System

- Network operating system services



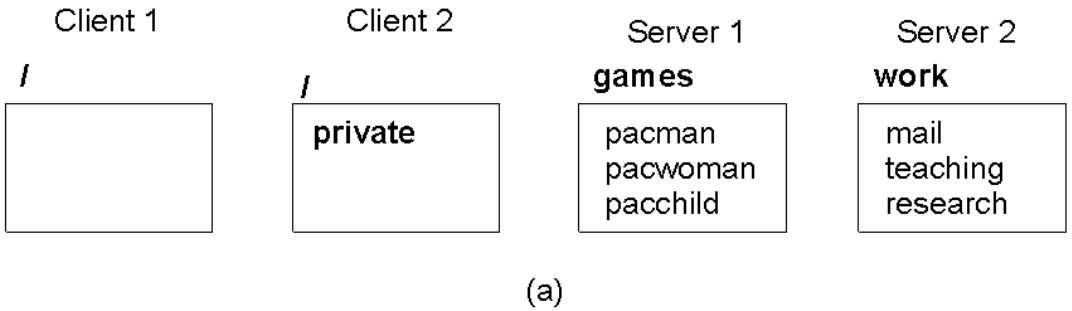
Network Operating System

- Two clients and a server with a network operating system using a shared file system

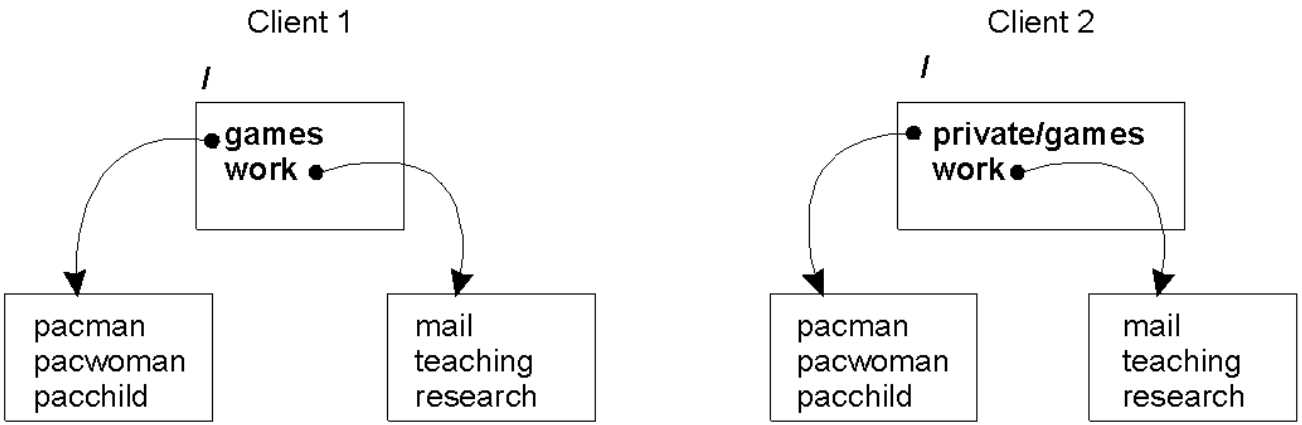


Network Operating System

- Different clients may mount the same remote files or the same directories (on the servers) in different subdirectories



(a)



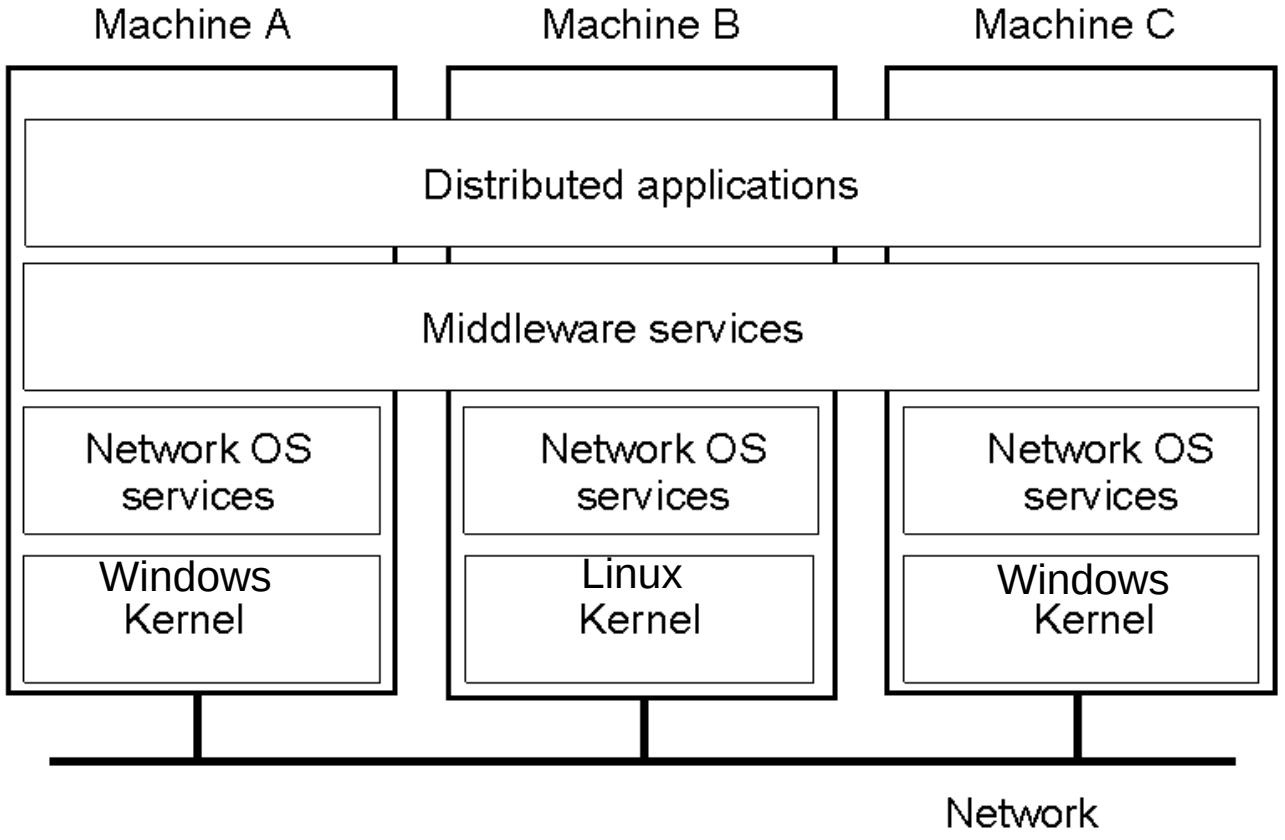
(b) (c)

Distributed System Middleware

- Need for distributed system middleware
  - **Many networked applications were difficult to integrate**
  - **Organizations and individuals were running different NOSs**
  - **Integration and interoperability occurred only at the level of primitive NOS services**
- Need for federated distributed information systems
  - **Combine different databases, but provide a single view to the applications**
  - **Offer enterprise-wide Internet services, but make use of existing information systems**
  - **Allow transactions across different databases**
  - **Allow extensibility for future services (e.g., mobility, collaboration, tele-networking)**
- Use existing OSs, and treat them as the underlying environment (they provide the same basic functionality anyway)

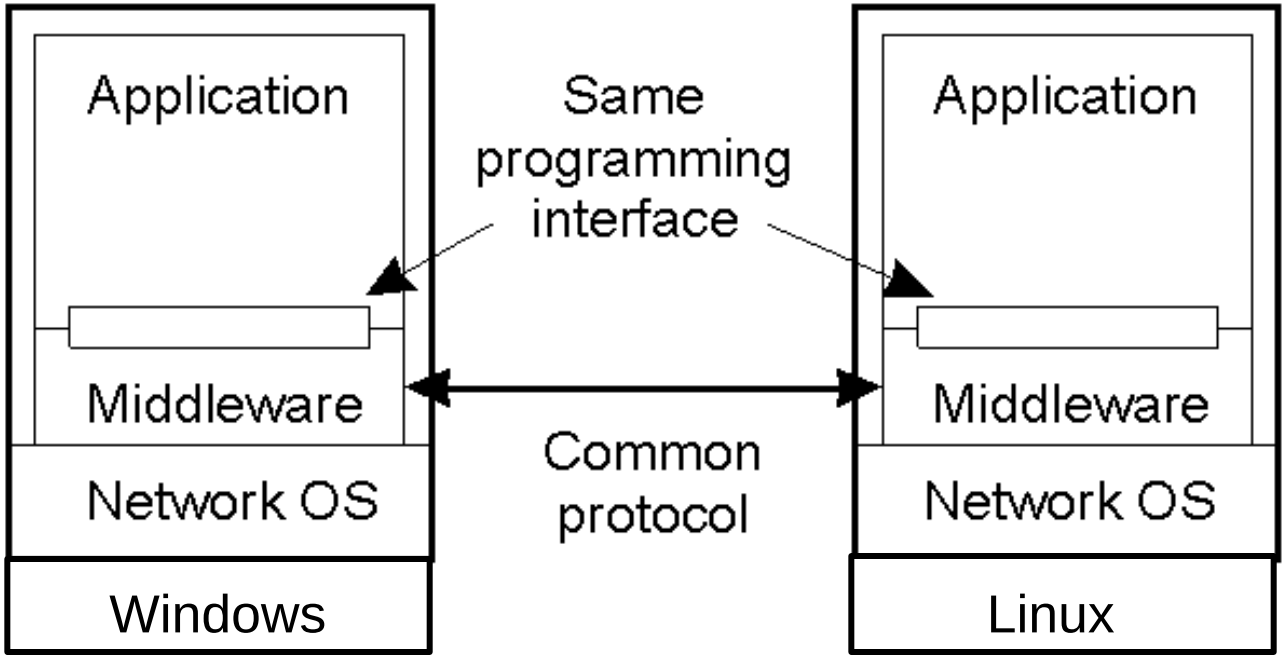
Distributed System Middleware

- Middleware services between the Distributed applications and the Network OS services



Middleware and Openness

- The protocols used by the middleware layer on each machine, and the interfaces they offer to the applications, are the same
- Thus, middleware supports
  - **Interoperability between different hardware and software platforms**
  - **Portability of applications from one SW / HW platform to another**



Comparison between Systems

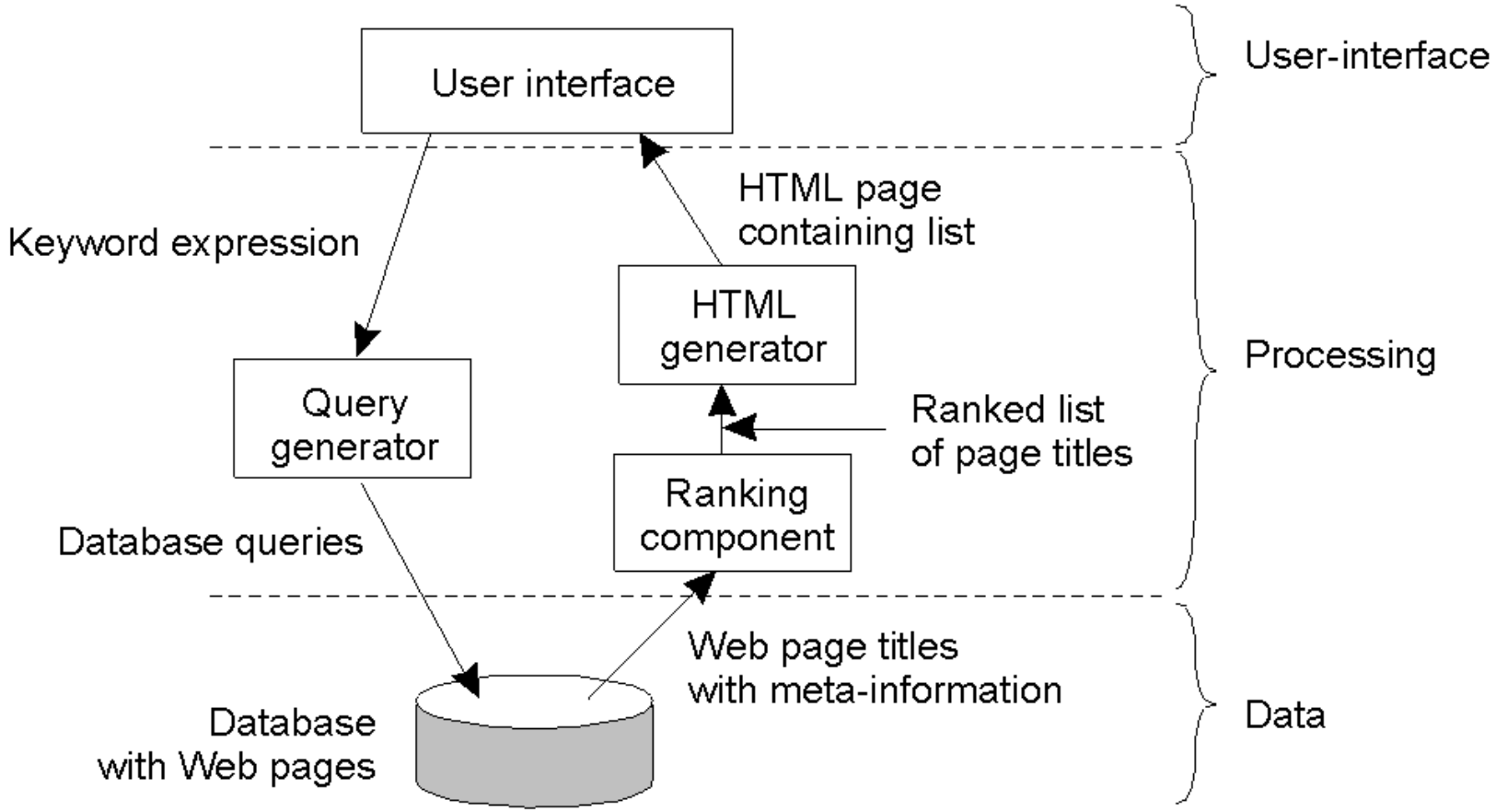
	DOS	**NOS**	**Middleware**
Degree of transparency	High	Low	High
Same OS on all nodes	Yes	No	No
Number of copies of OS	N	N	N
Basis for communication	Messages	Files	Model-specific
Resource management	Global, distributed	Per node	Per node
Scalability	Moderate	Yes	Varies
Openness	Closed	Open	Open

Traditional Application Layering

- **User-interface layer** - Contains the application user interface
- **Processing layer** - Contains the application functions to be invoked through the user interface
- **Data layer** - Contains the data to be accessed through the user interface

Example of Application Layering

- An Internet search engine organized into three layers



Models of Distributed Computation

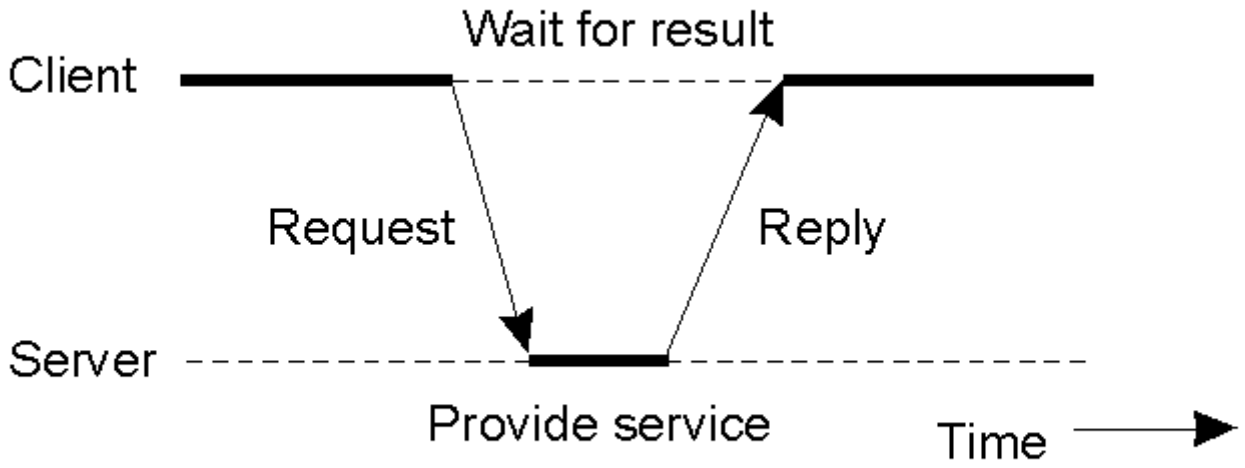
- Client-Server Model
- Peer-to-Peer Model

Client-Server Model

- **Servers:** Generally provide services related to a *shared resource*
  - **Shared file systems, databases, etc.**
  - **Shared linked documents, e.g., Web documents**
  - **Shared applications**
  - **Shared distributed objects**
- **Clients:** Access the servers remotely using
  - **Programming interface that transforms client's local calls to request / reply messages**
  - **Devices such as mobile phones, barcode readers, automated teller machines, etc.**
  - **Computers that provide separate user interfaces for specific services**
  - **Computers that provide an integrated user interface for related services**

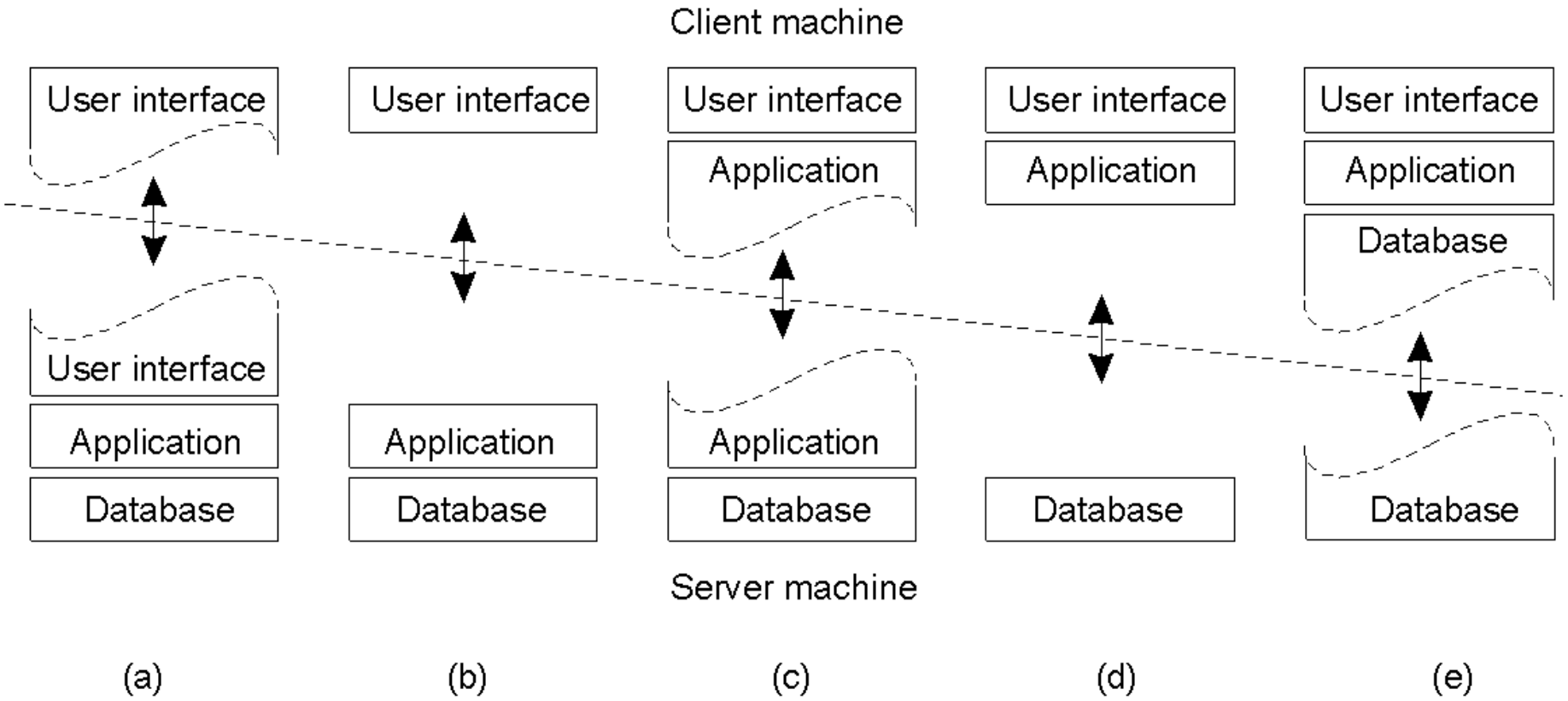
Clients and Servers

- General interaction between a client and a server



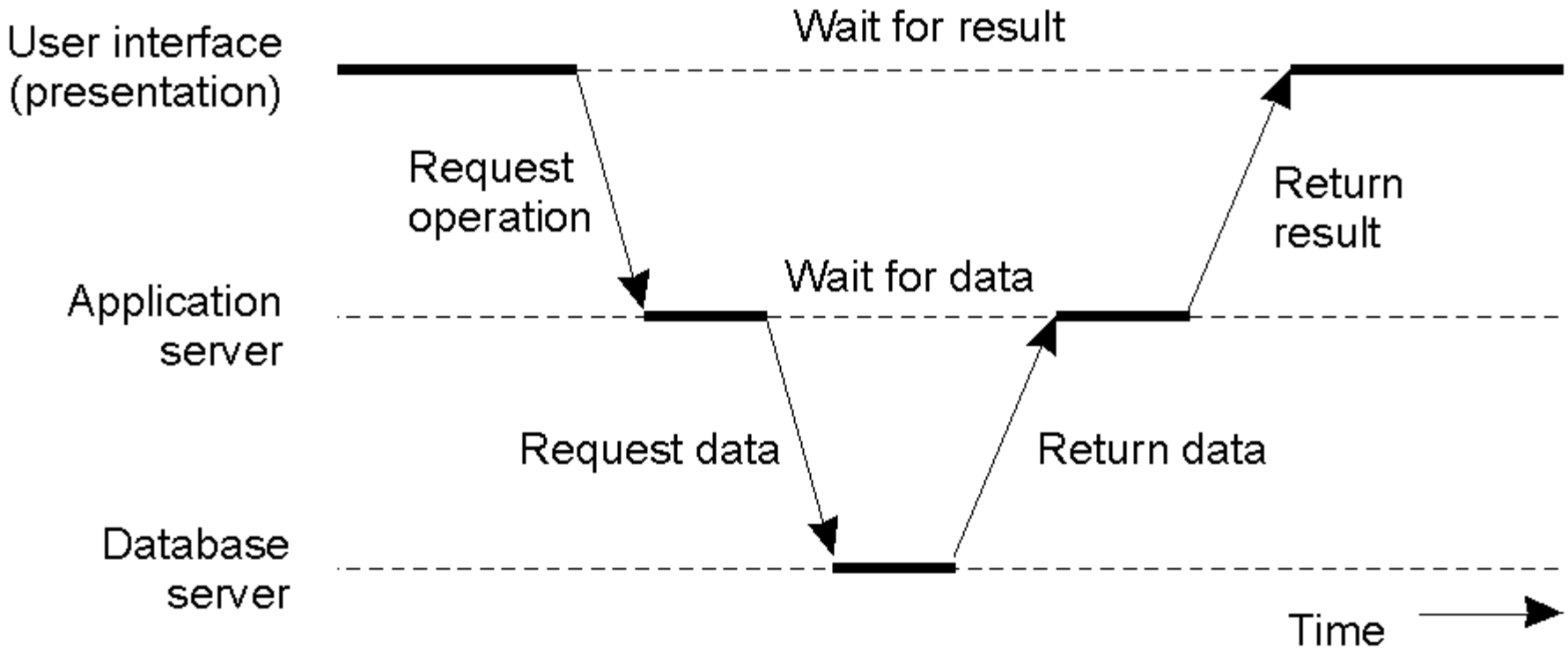
Multi-Tiered Architectures

- Alternative client-server architectures (a) – (e)



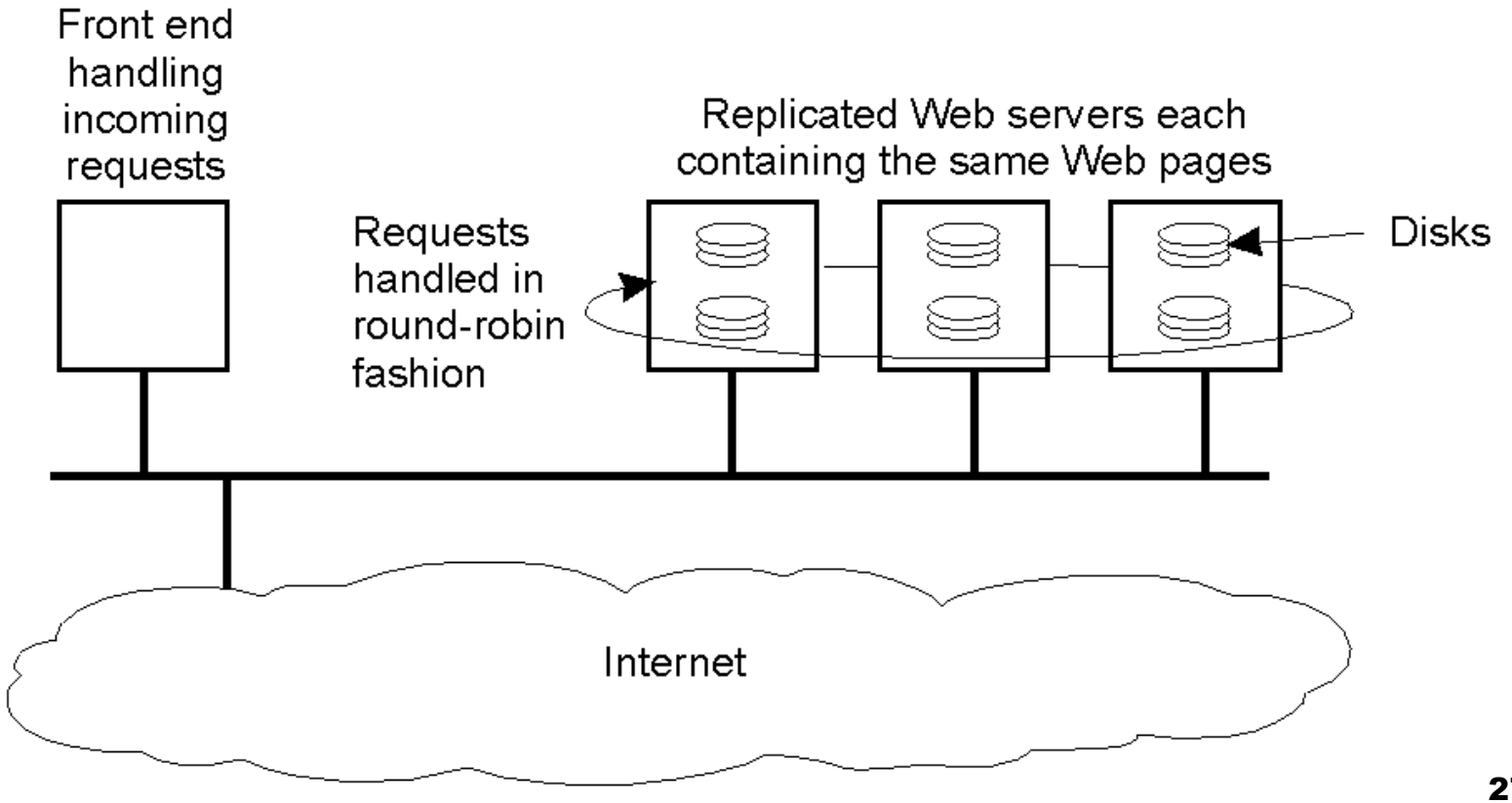
Multi-Tiered Architectures

- An example of a server acting as a client



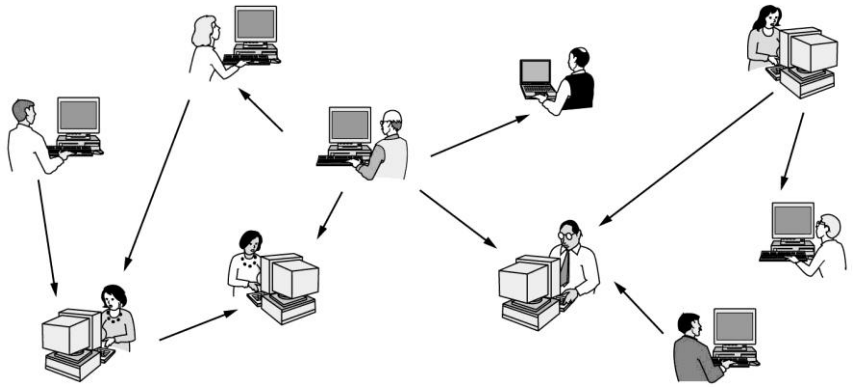
Example Client-Server Architecture

- A client accessing a Web page using a Web server that is replicated for fault tolerance and/or load balancing



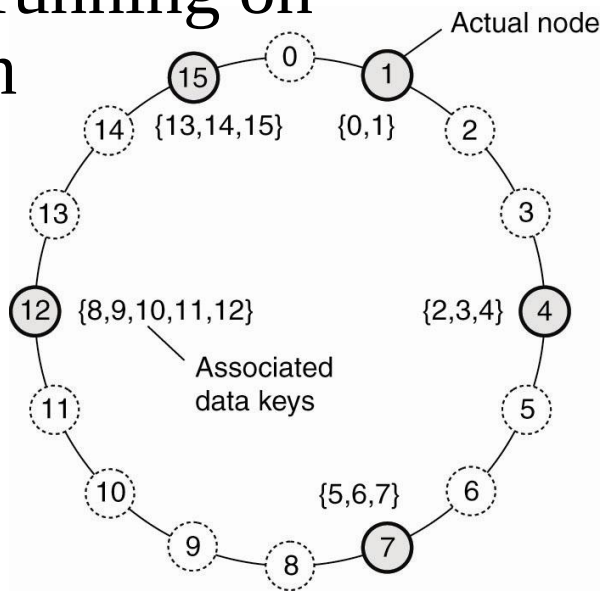
Peer-to-Peer Architecture

- Computers communicate directly with each other
- No notion of client or server (alternatively, every computer is both a client and a server)
- Peer-to-Peer (P2P) architectures are categorized as:
  - **Structured – Based on an overlay network, with a particular topology, e.g., ring, tree, etc. (as shown on Slide 29)**
  - **Unstructured – No associated overlay network (as shown below and on Slides 30-32)**



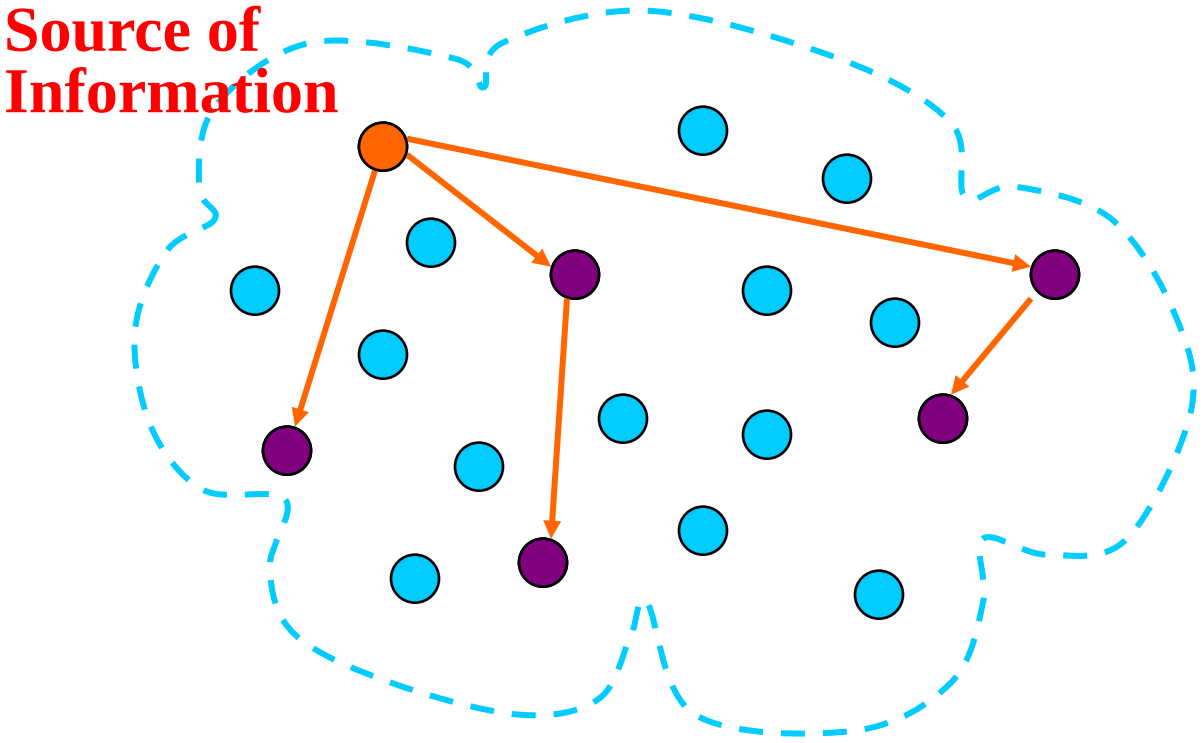
Example Peer-to-Peer Architecture

- **Chord:** A structured peer-to-peer architecture for distributed data storage with a ring topology, using a **Distributed Hash Table (DHT)**
- A data item with key *k* is mapped to the node with the smallest $id \geq k$, referred to as *succ(k)*
- To look up a data item, an app running on a node calls *lookUp(k)* to obtain the network address of *succ(k)*



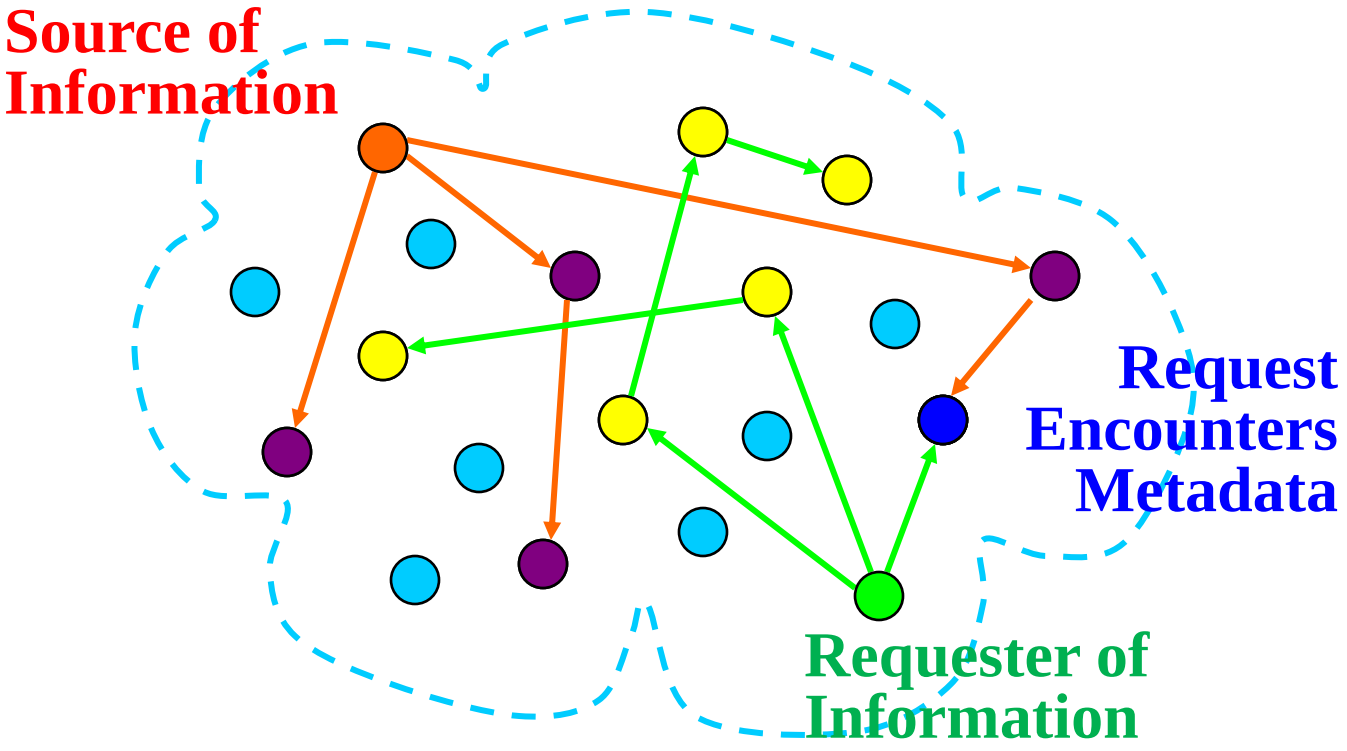
Example Peer-to-Peer Architecture

- **iTrust:** An unstructured peer-to-peer architecture for distribution, search, and retrieval of information
- Source distributes metadata to randomly chosen nodes



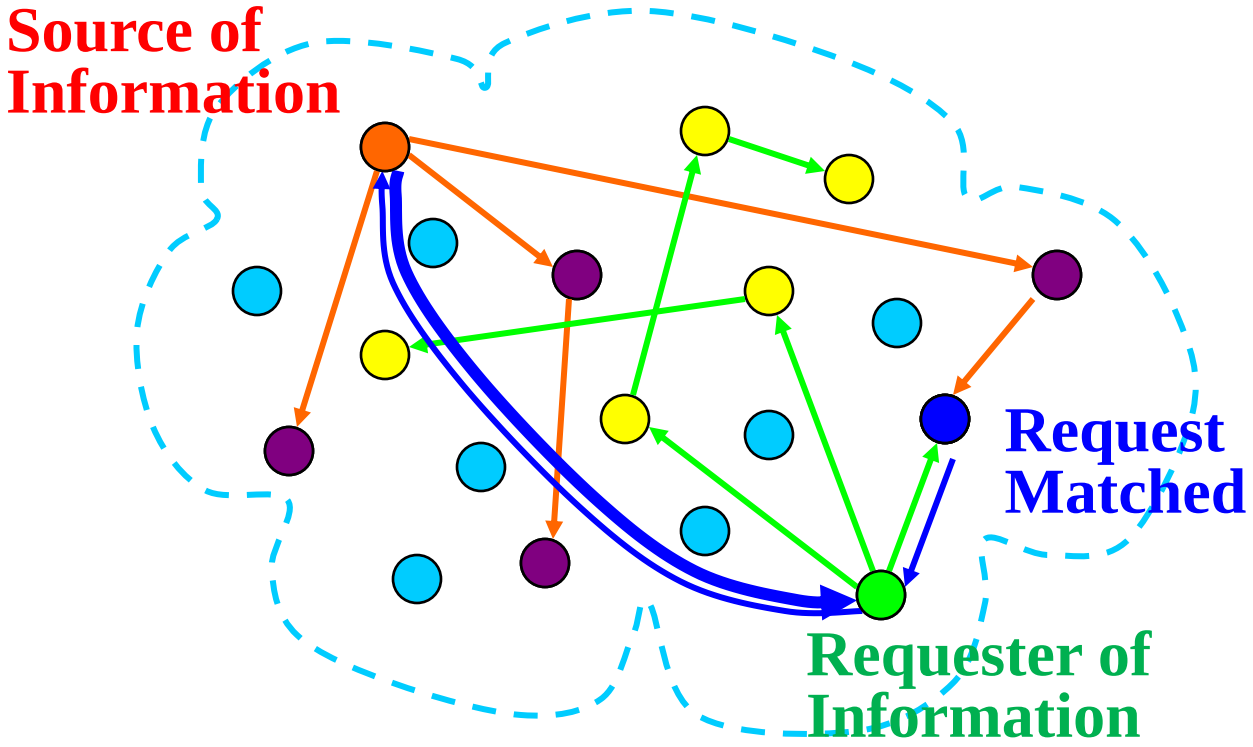
Example Peer-to-Peer Architecture

- Requester distributes request (query) containing keywords to randomly chosen nodes



Example Peer-to-Peer Architecture

- Requester retrieves information from source node



- For more information, see: http://itrust.ece.ucsb.edu