Chapter 10: Distributed Systems

Based on G. Coulouris' and A. Bernstein's slides on Distributed Systems

Distributed Systems

- Objectives
  - To develop an understanding of what a distributed system is
  - To present the key challenges in distributed systems
  - To overview main concepts and approaches
- Topics
  - Evolution of communication networks and scale
  - Definition of distributed systems
  - Examples of distributed systems
  - Challenges and examples for those
  - Hardware architectures
  - Software concepts
  - Middleware
  - Network interaction types

Evolution of Networks

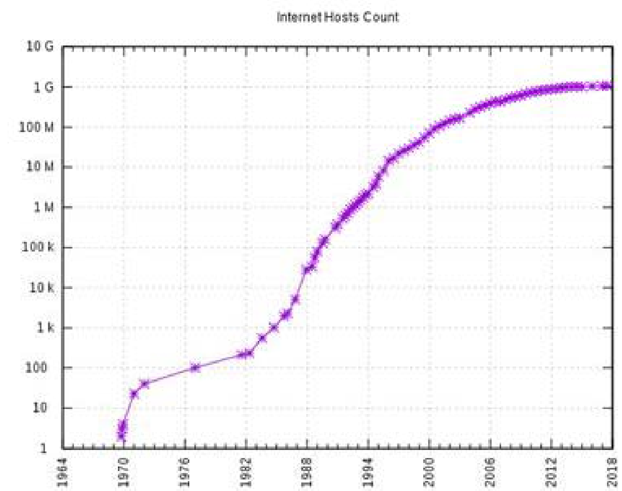
- In early times, computers were standalone devices
- In the late 80's/early 90's, computer networks started
  - Internet brought a revolution to the way of life on the planet!
- Today networking is essential to everyday life
  - Even low budget laptops (One Laptop Per Child project, OLPC, & Lehrplan 21) are designed to be networked
- Shift from standalone computers, to the paradigm of computers communicating, interacting, and collaborating with each other

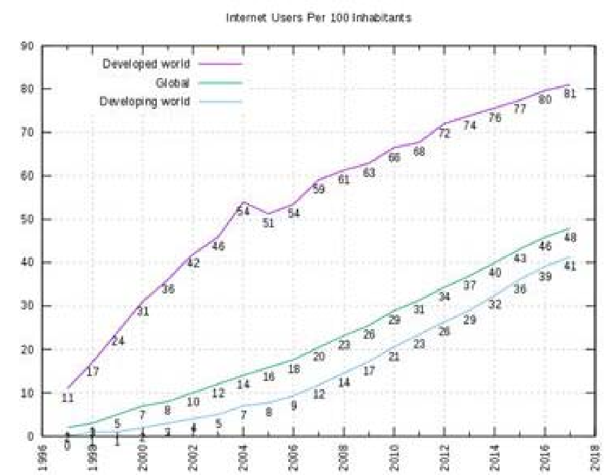

- Challenging to manage Distributed Systems at a large and complex scale

The Scale (1)

- Increasing number of computers
  - Adding devices as for the Internet-of-Things (IoT) ➔ 20+ Billion
- Increasing number of Internet users
- Increasing number of distributed applications

- All needs are increasingly
  - Complex
  - Larger scale
  - Application-specific

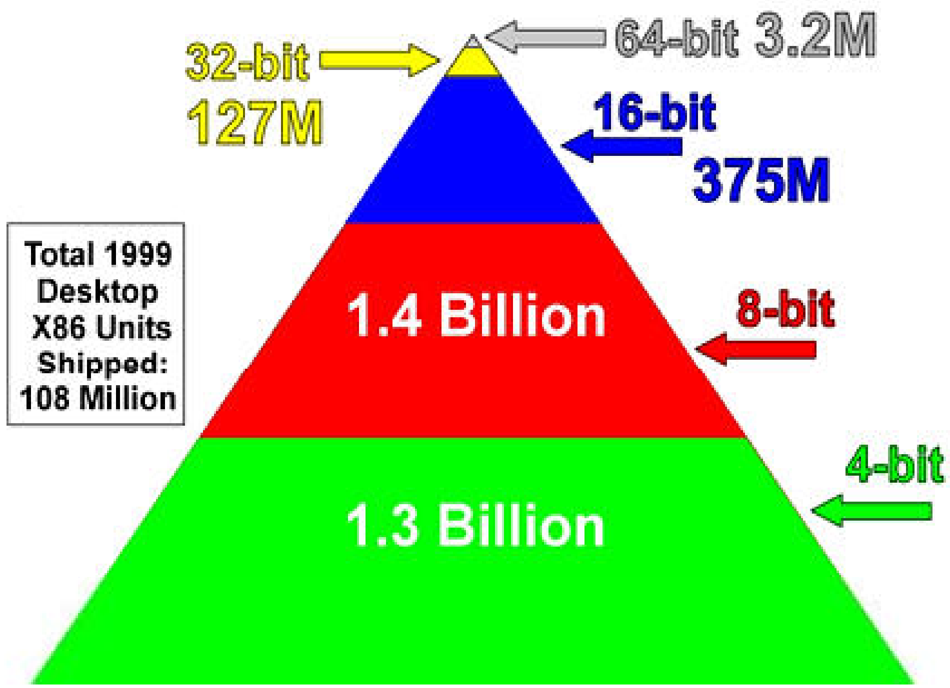




https://en.wikipedia.org/wiki/Global_Internet_usage#Internet_hosts

The Scale (2)

- The real scale is larger



http://www.linuxdevices.com/cgi-bin/printerfriendly.cgi?id=AT9656887918
http://www.embedded.com/1999/9905/9905turley.htm

Internet of Things/ Industry 4.0

http://www07.abb.com/images/librariesprovider20/
Header-image-/contact-industry4-0-industrial-internet.jpg?sfvrsn=1

“The” or “a” Definition

- Distributed Systems come in many flavors!
  - Computers connected by a network and
  - Spatially separated by any distance
- Def.: A collection of independent computers that appears to its users as a single coherent system
  - Hardware: All machines are fully autonomous
  - Software: Users think they deal with a single system
- Consequences
  - Concurrency
  - No global clock
  - Independent failures

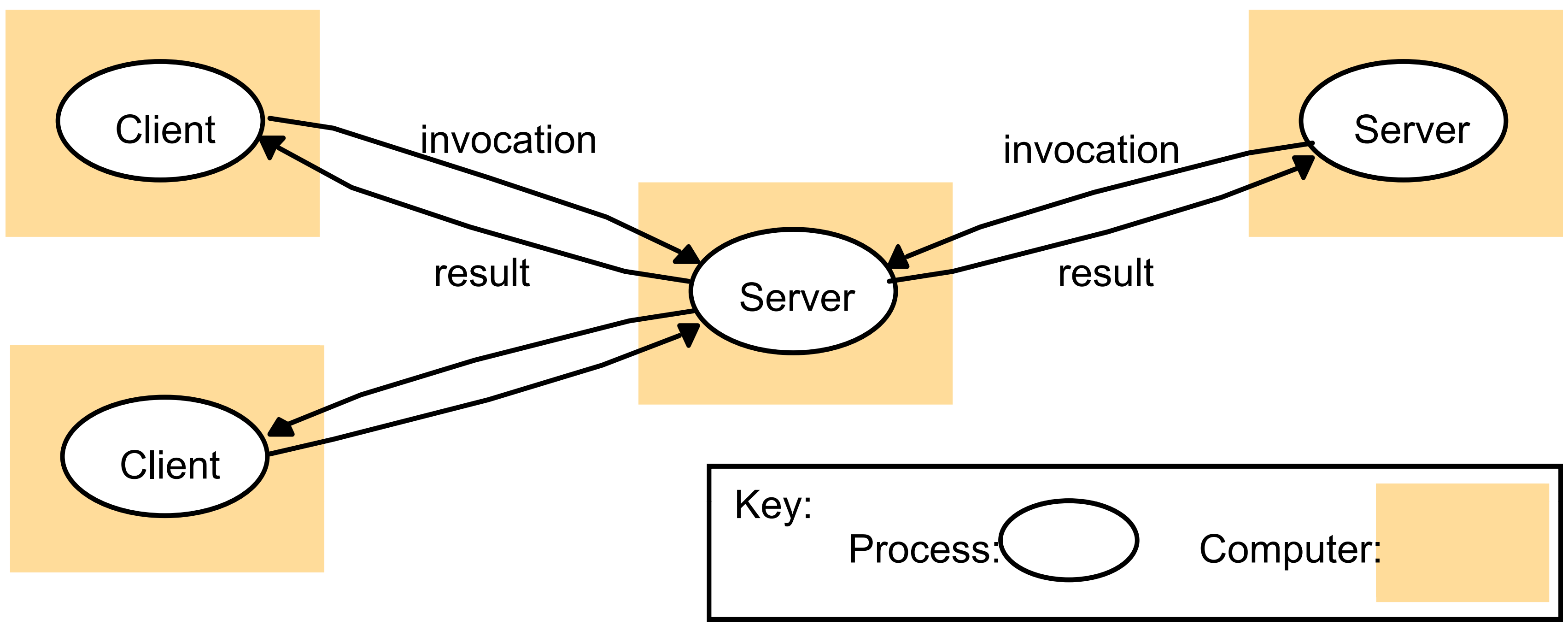
Distributed System Examples

- Client Server Communications
- Intranet
- Internet
- Automation Network
- Personal Network
- Peer-to-Peer Systems
- Cloud Computing

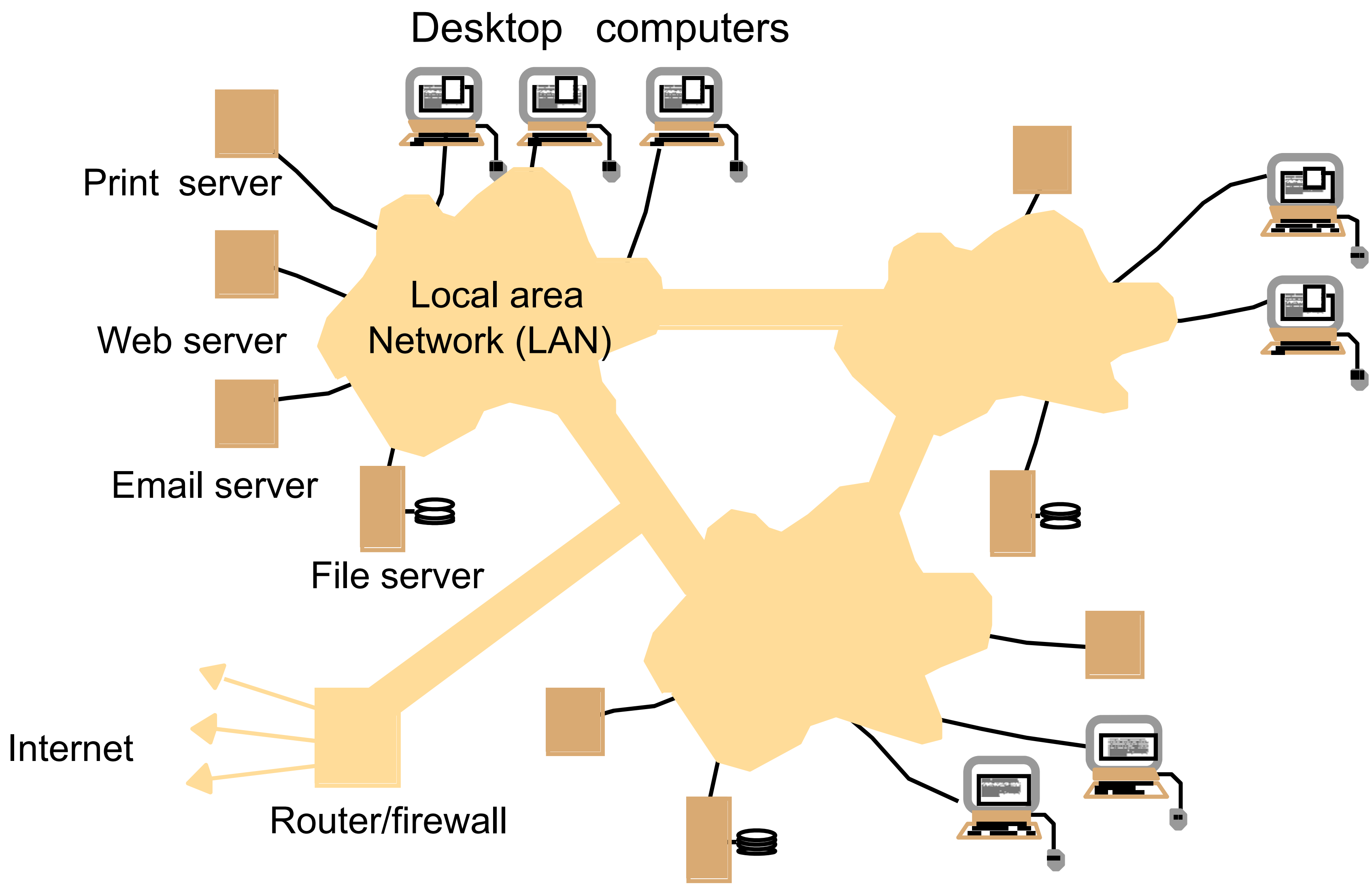
leading to

- Distributed Systems’ Middleware

Client Server Communications



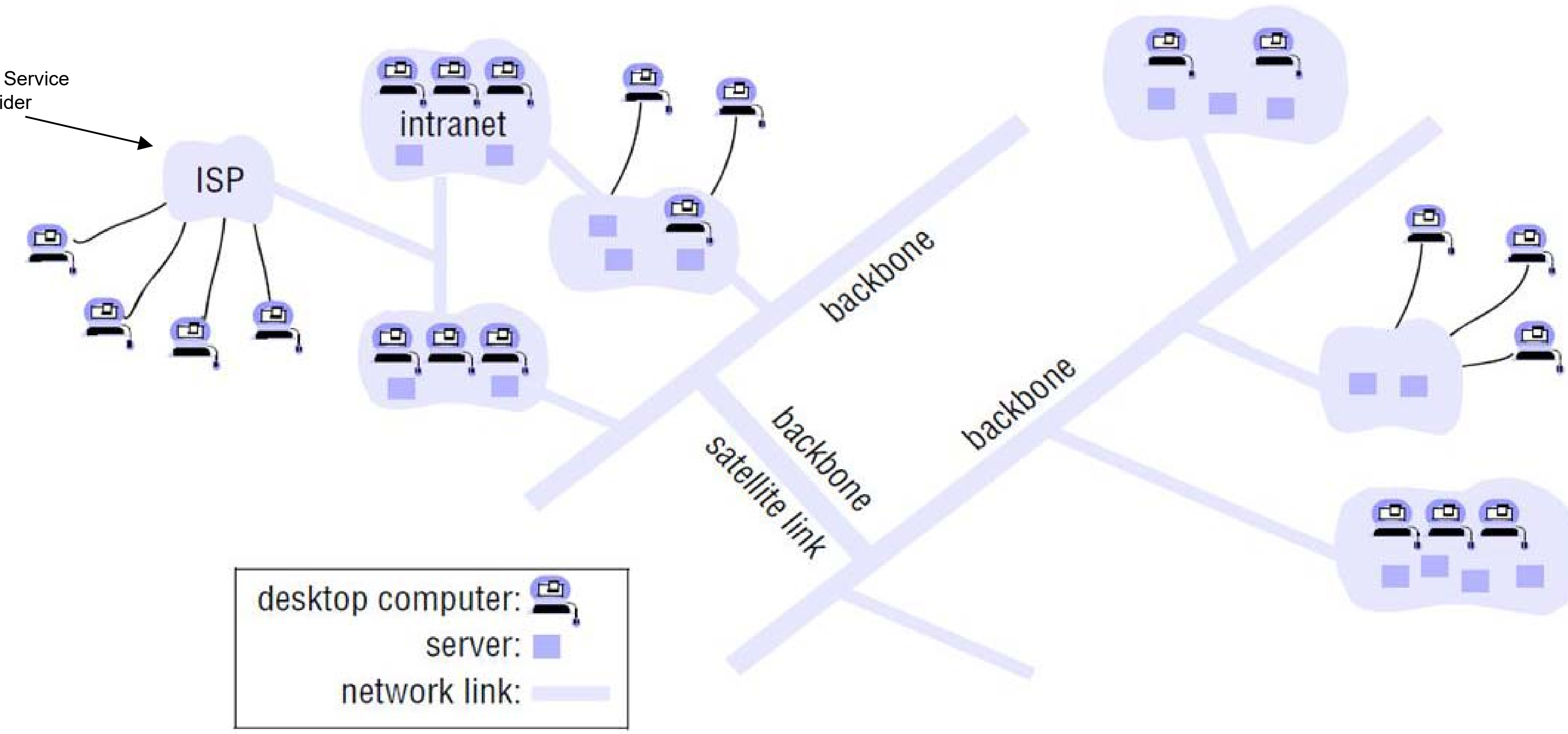
Intranet



Desktop computers
Print server
Web server
Local area
Network (LAN)
Email server
File server
Internet
Router/firewall

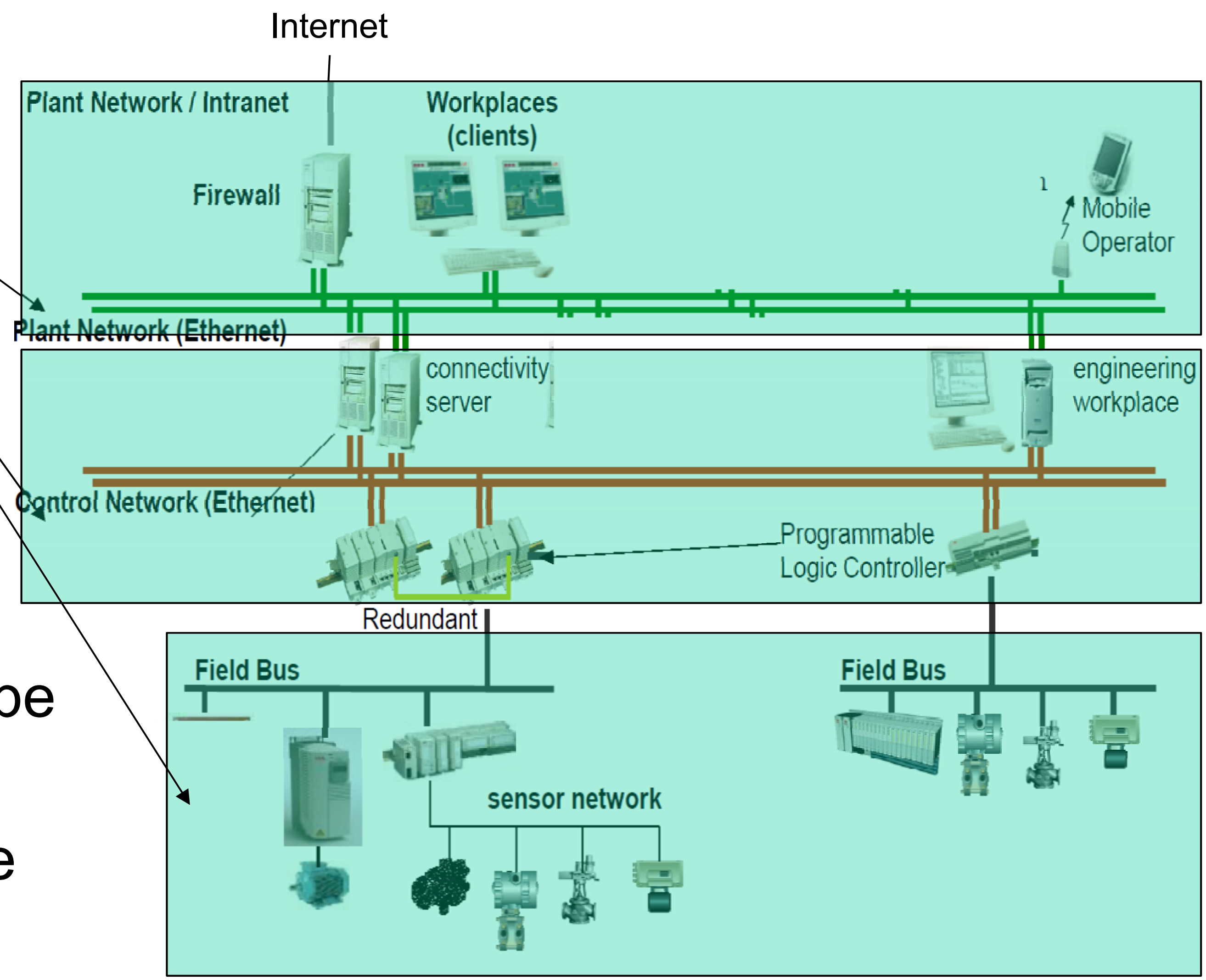
Internet

Internet Service Provider



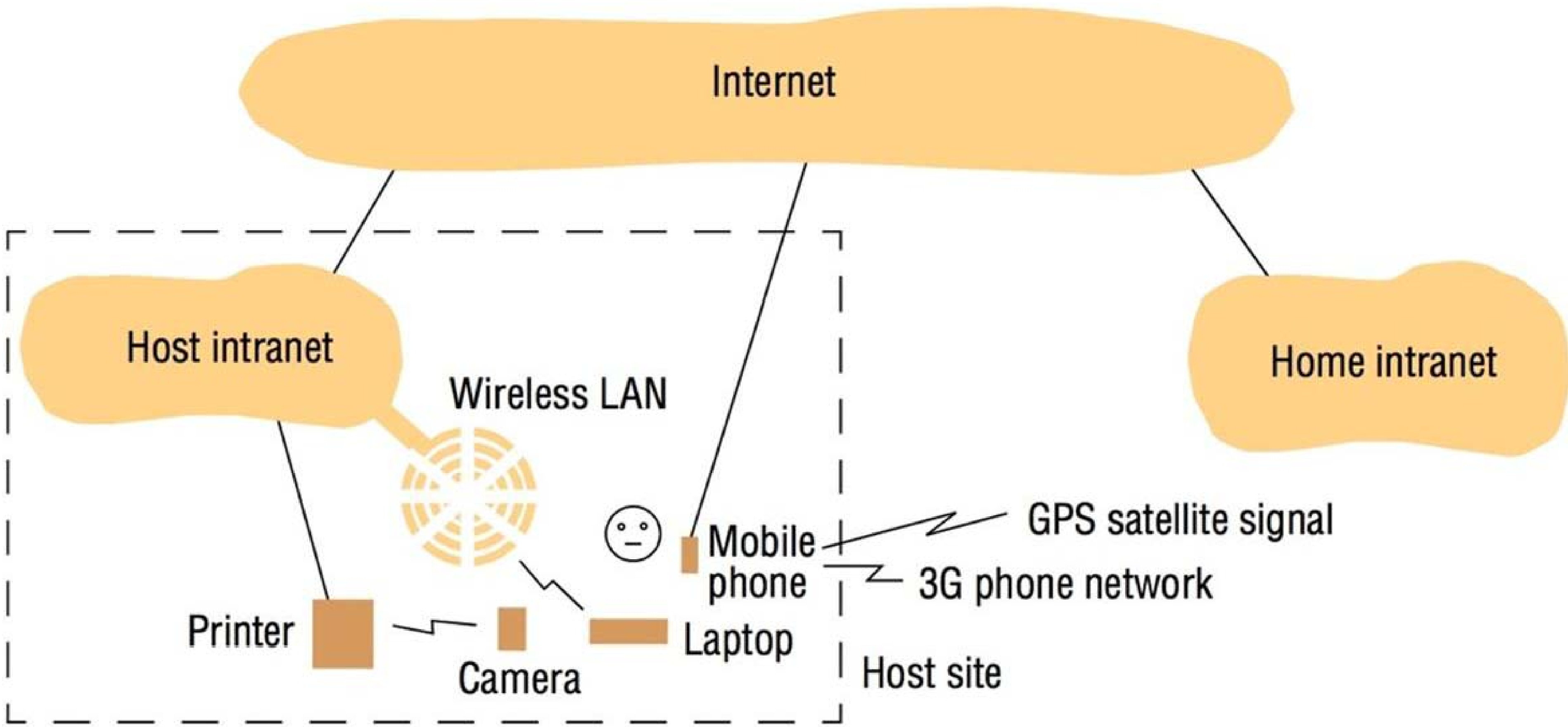
Automation Network

- 3 layers
- Focus on reliability
  - Redundancy
- Real-time (control) network
  - Messages must be delivered in time
  - Special hardware (controllers)

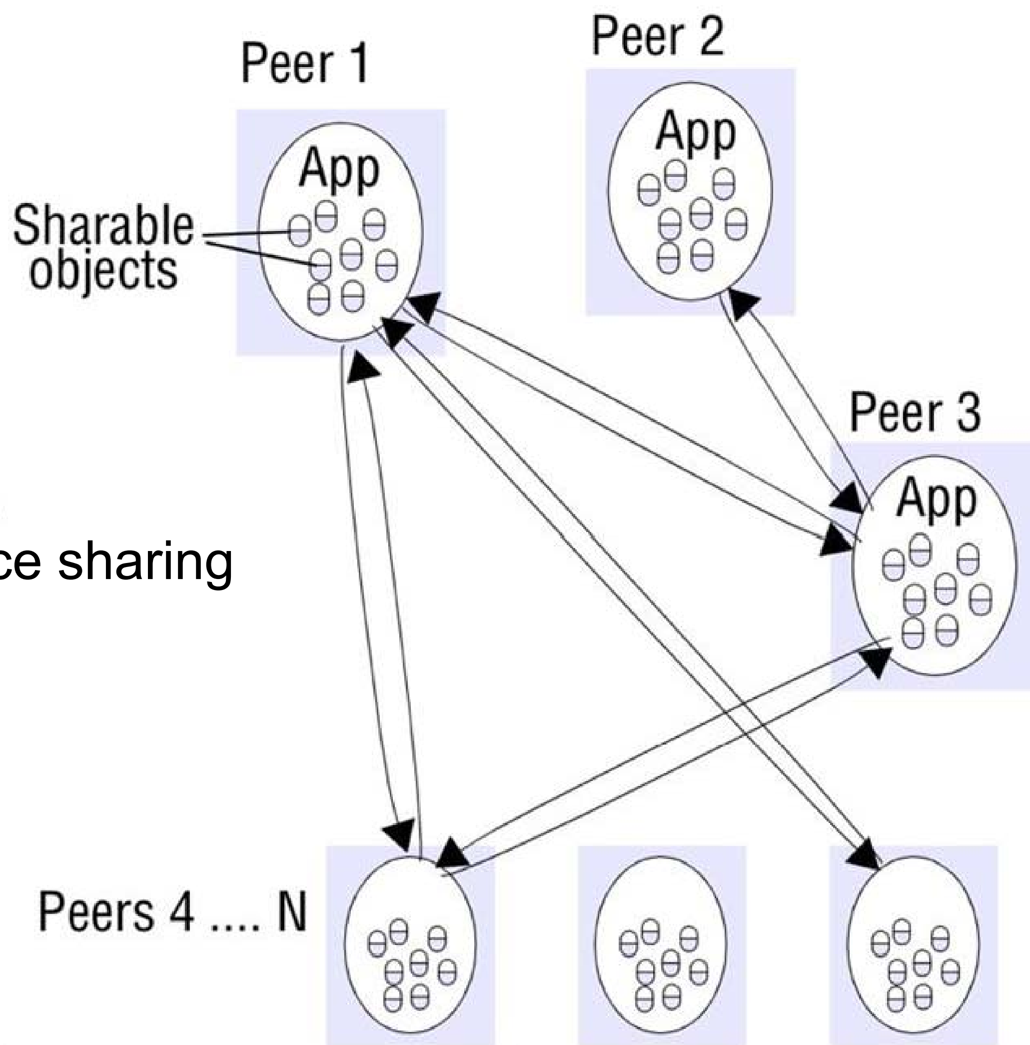


Personal Network

Portable and handheld devices in a distributed system

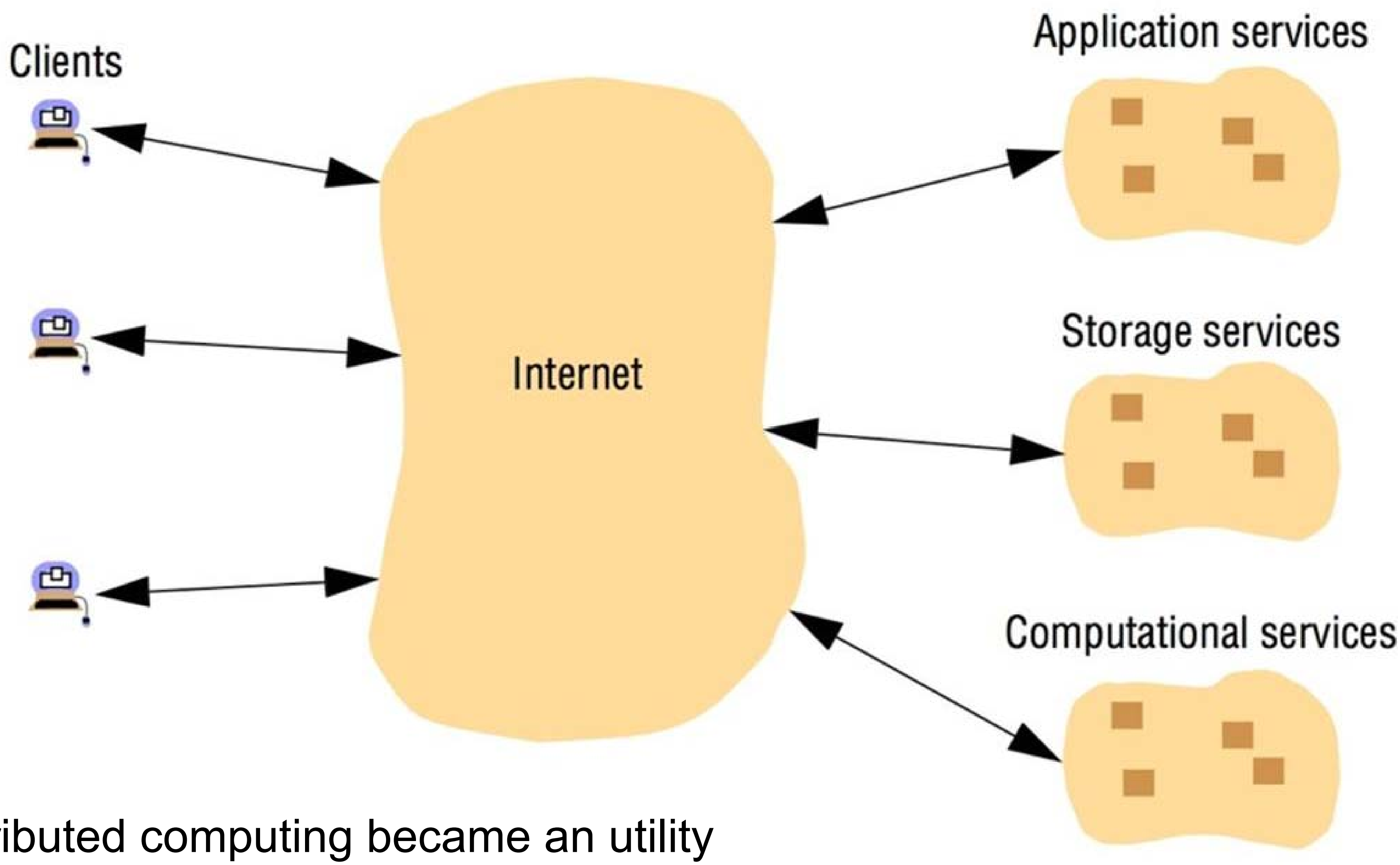


Peer-to-Peer Systems



Distributed resource sharing based on interests

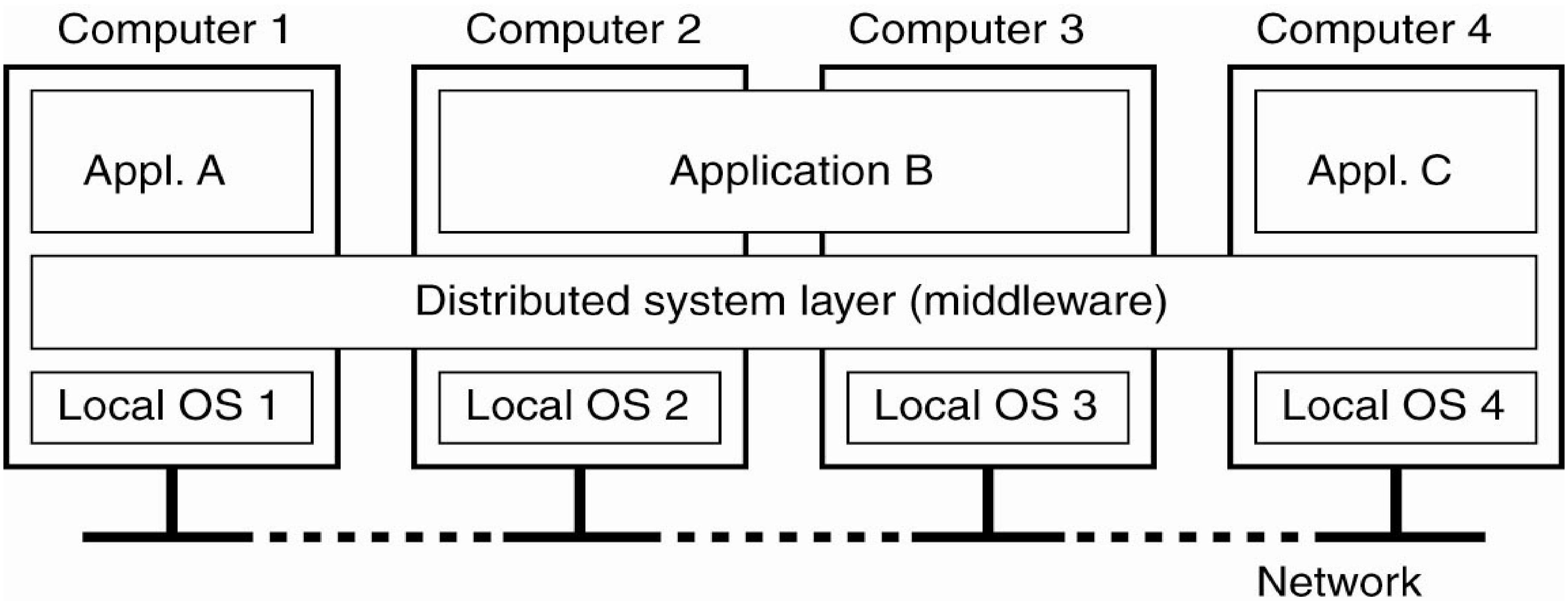
Cloud Computing



Distributed computing became an utility

Distributed System as a Middleware

- Middleware provides similar interfaces to all participants
  - Irrespective of types of OS across multiple machines
  - Irrespective of types of networks, communications used



Key Challenges in Distributed Systems

1. Transparency
   - Single view of the system
   - Hide numerous details
2. Heterogeneity
   - Networks
   - Computers (HW)
   - Operating systems
   - Programming languages
   - Developers
3. Failure Handling
   - Detecting
   - Masking
   - Tolerating
   - Recovery
   - Redundancy
4. Openness
   - Extensibility
   - Publication of interfaces
5. Scalability
   - Controlling the cost of resources
   - Controlling the performance
   - Preventing resources from running out
   - Avoiding performance bottlenecks
6. Security
   - Secrecy
   - Authentication
   - Authorization
   - Non-repudiation

1. Transparency (1)

- Recall of the distributed system definition:
  A collection of independent computers that appears to its users as a single coherent system

- Thus, transparency means
  - A set of computers appears as a single computer to all applications and users
  - Abstractions needed to facilitate application development
  - All details are hidden and dealt with transparently

- Transparency comes in *eight* different flavors

1. Transparency (2)

- Access transparency
  - Enables local and remote resources to be accessed using identical operations
- Location transparency
  - Enables resources to be accessed without knowledge of their physical or network location (for example, which building or IP address)
- Concurrency transparency
  - Enables several processes to operate concurrently using shared resources without interference between them
- Replication transparency
  - Enables multiple instances of resources to be used to increase reliability and performance without knowledge of the replicas by users or application programmers

1. Transparency (3)

- Failure transparency
  - Enables the concealment of faults, allowing users and application programs to complete their tasks despite the failure of hardware or software components
- Mobility transparency
  - Allows the movement of resources and clients within a system without affecting the operation of users or programs
- Performance transparency
  - Allows the system to be reconfigured to improve performance as loads vary
- Scaling transparency
  - Allows the system and applications to expand in scale without change to the system structure or the application algorithms

1. Sample Cases and Problems

- Access transparency
  - Different data representations, e.g., “100” binary
    - 4 decimals in Big Endian by reading from left to right, *e.g.,* Sun Sparc
    - 1 decimal in Little Endian by reading from right to left, *e.g.*, Intel
  - Different conventions
    - *E.g.*, the Linux files system is case-sensitive, Windows’ not
- Location/Migration/Relocation transparency
  - Access to Web page without knowing its IP, physical location
  - Transparent handover when changing wireless access points
- Replication transparency
  - GMail, Facebook, banks maintain their data transparently replicated for increased availability and improved access time

2. Heterogeneity

- *E.g.*, access transparency is dealing with one part of heterogeneity
  - Hiding away details of many technological differences
- Additionally, heterogeneity addresses transparently differences in
  - Performance
  - Capabilities
  - Network connectivity
    - For instance, in a Personal Area Network (PAN), transparently connecting desktop, laptop, watch, and mobile phone
    - Avoiding the assignment of intensive tasks to the mobile phone or a watch

3. Failure Handling

- Another definition of a distributed system (L. Lamport):
  - You know you have one, when the crash of a computer, you've never heard of, stops you from getting any work done." ☺
- Fundamental points in distributed systems
  - Reliability
  - High Availability
- Distributed systems should be failure-transparent
  - A failure on "some" components should not be fatal (or, ideally even detectable) to the applications.
  - *E.g.*, a failure in a bank server should not prevent you from withdrawing money

3. Types of Failures

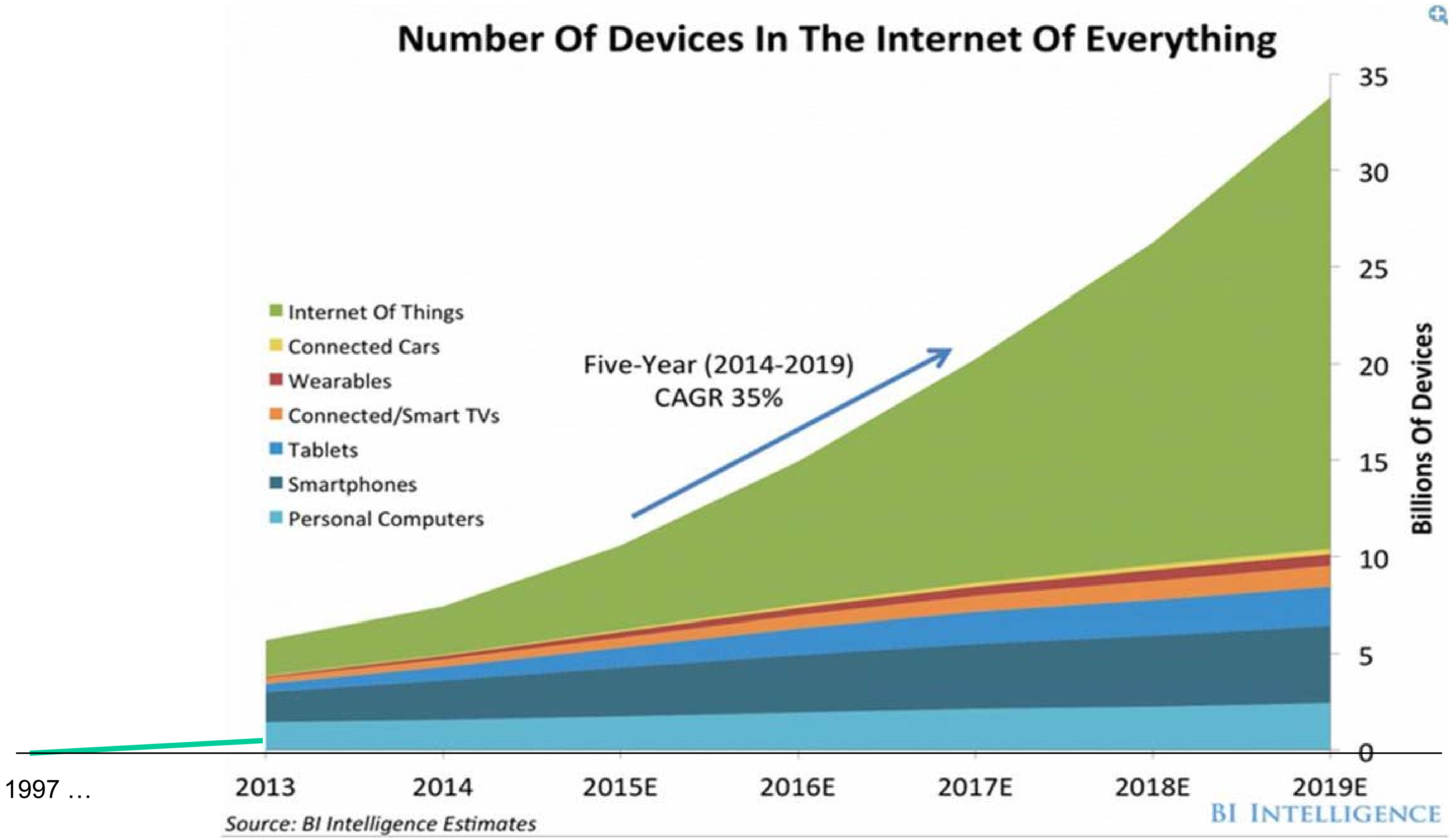
Class of Failure	Affects	Description
Fail-stop	Process	Process halts and remains halted. Other processes may detect this state.
Crash	Process	Process halts and remains halted. Other processes may not be able to detect this state.
Omission	Channel	A message inserted in an outgoing message buffer never arrives at the other end's incoming message buffer.
Send-omission	Process	A process completes a *send*, but the message is not put in its outgoing message buffer.
Receive-omission	Process	A message is put in a process's incoming message buffer, but that process does not receive it.
Arbitrary (Byzantine)	Process or channel	Process/channel exhibits arbitrary behavior: it may send/transmit arbitrary messages at arbitrary times, commit omissions; a process may stop or take an incorrect step.

4. Openness

- Well-defined interfaces
  - *E.g.*, Application Programming Interfaces (API)
  - Well designed
  - Clearly described (publicly)
  - Standardized

- Use of open Interface Definition Language (IDL)
  - Boosting interoperability, portability, and extensibility
  - Boosting modularity
    - Upgrading one module should not affect the rest

5. Scalability (1)



CAGR: Compound Annual Growth Rate

5. Scalability (2)

- Even scalability comes in different flavors
  - Scalability with respect to size
    - *E.g.*, more computers, more users, or more devices
  - Scalability with respect to geography
    - *E.g.*, applicable at different languages, regions, communities
  - Scalability with respect to administration
    - *E.g.,* more active users, more failures, more connections
  - Scalability with respect to load
    - Add/remove resources depending on load

- Effects and impacts
  - Most systems suffer from performance loss when scaling

5. Scalability Limitation

- Problematic dimensions
  - Server performance
  - Network bandwidth
- Solutions
  - Storing data in a distributed manner might reduce overall data volume communicated

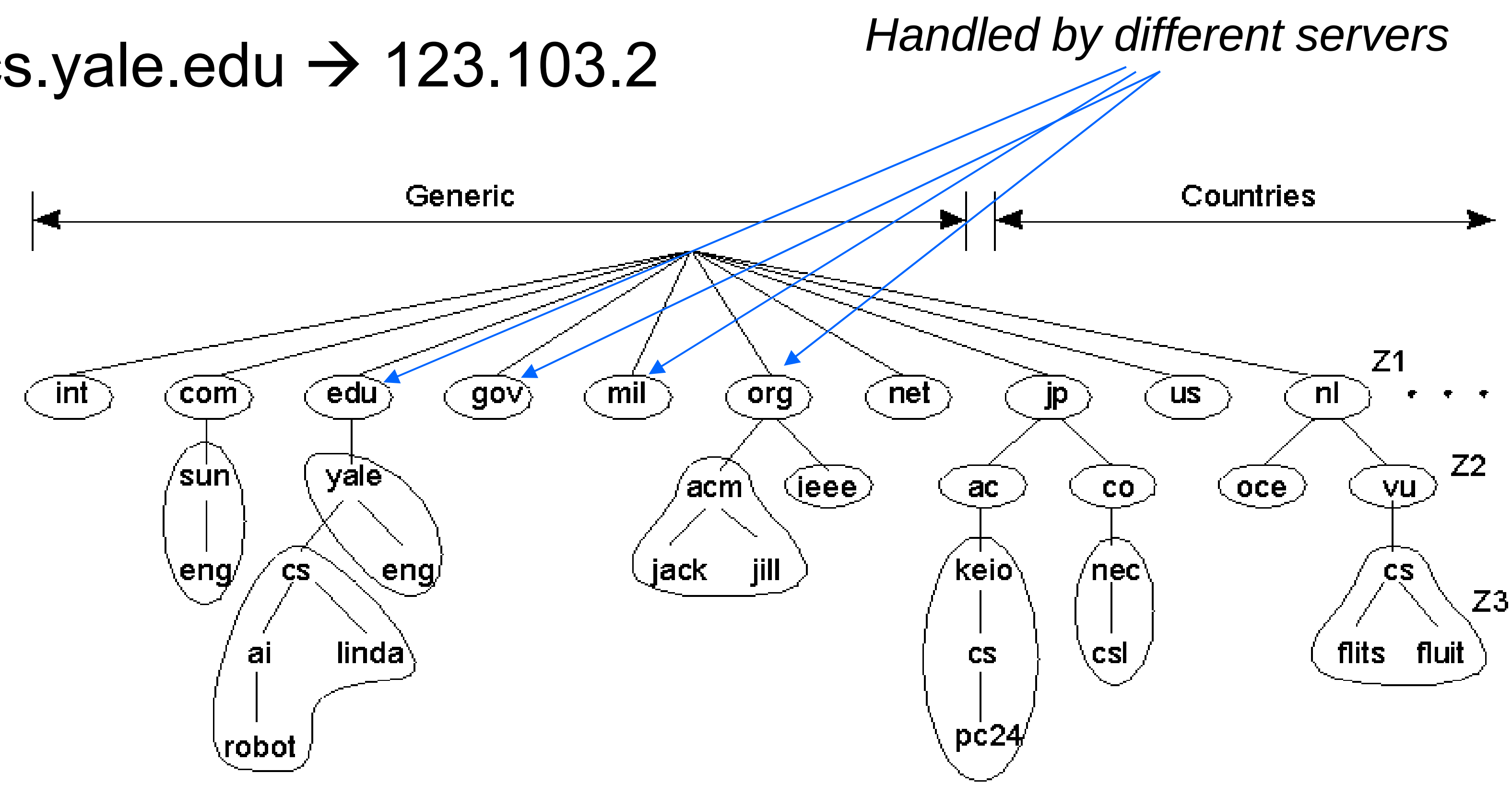
Centralized services	A single server for all users
Centralized data	A single on-line telephone book
Centralized algorithms	Doing routing based on complete information

5. Techniques to Address Scalability

- Alternative approaches
  - Distribution of responsibilities
    - *E.g.*, partition computation, split data on multiple computers
  - Hide communication latencies
    - Avoid waiting for responses, do something else
  - Apply replication techniques
    - *E.g.*, copy data onto multiple servers

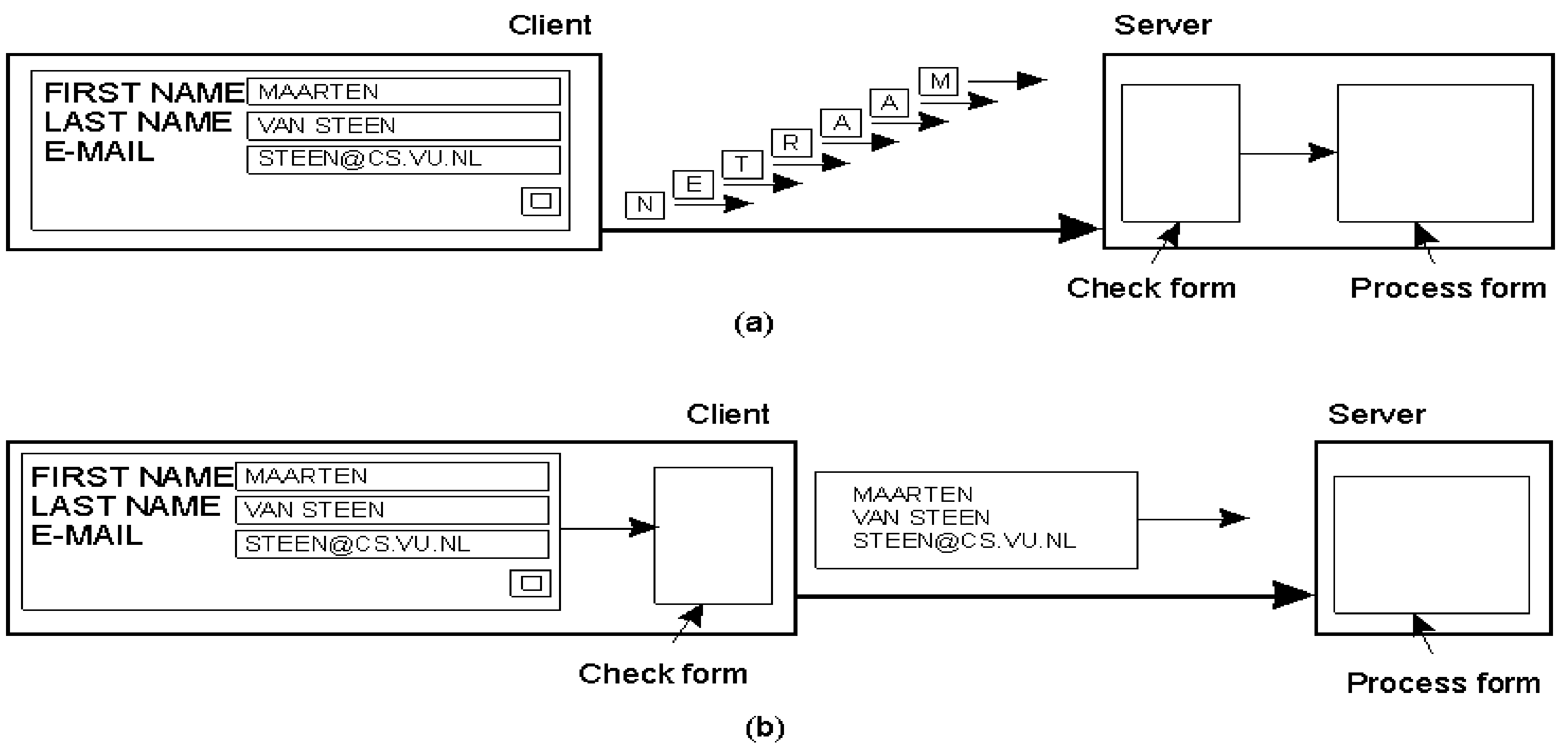
5.1 Distribution of Responsibilities

- Example: Domain Name System (DNS)
  - Server name → IP address
- DNS divided into zones
  - cs.yale.edu → 123.103.2



5.2 Hiding Communication Latencies

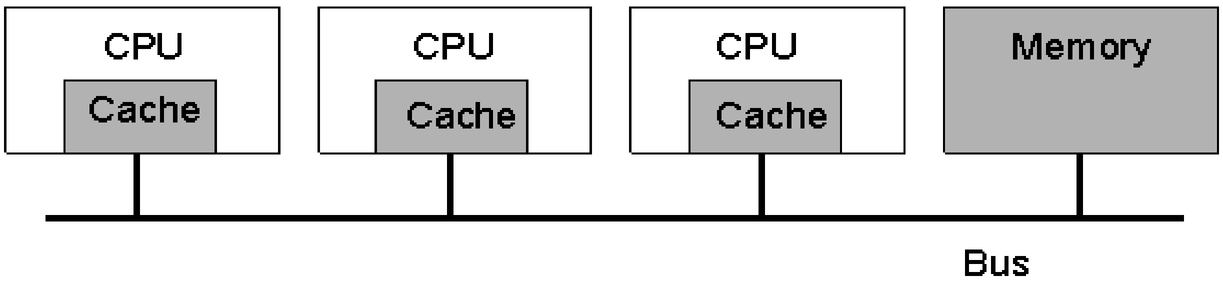
- The difference between letting
  - a server or
  - a client check forms as they are being filled



5.3 Replication Techniques

- A typical example is caching

- Challenge due to replication
  – Maintaining consistency for updates

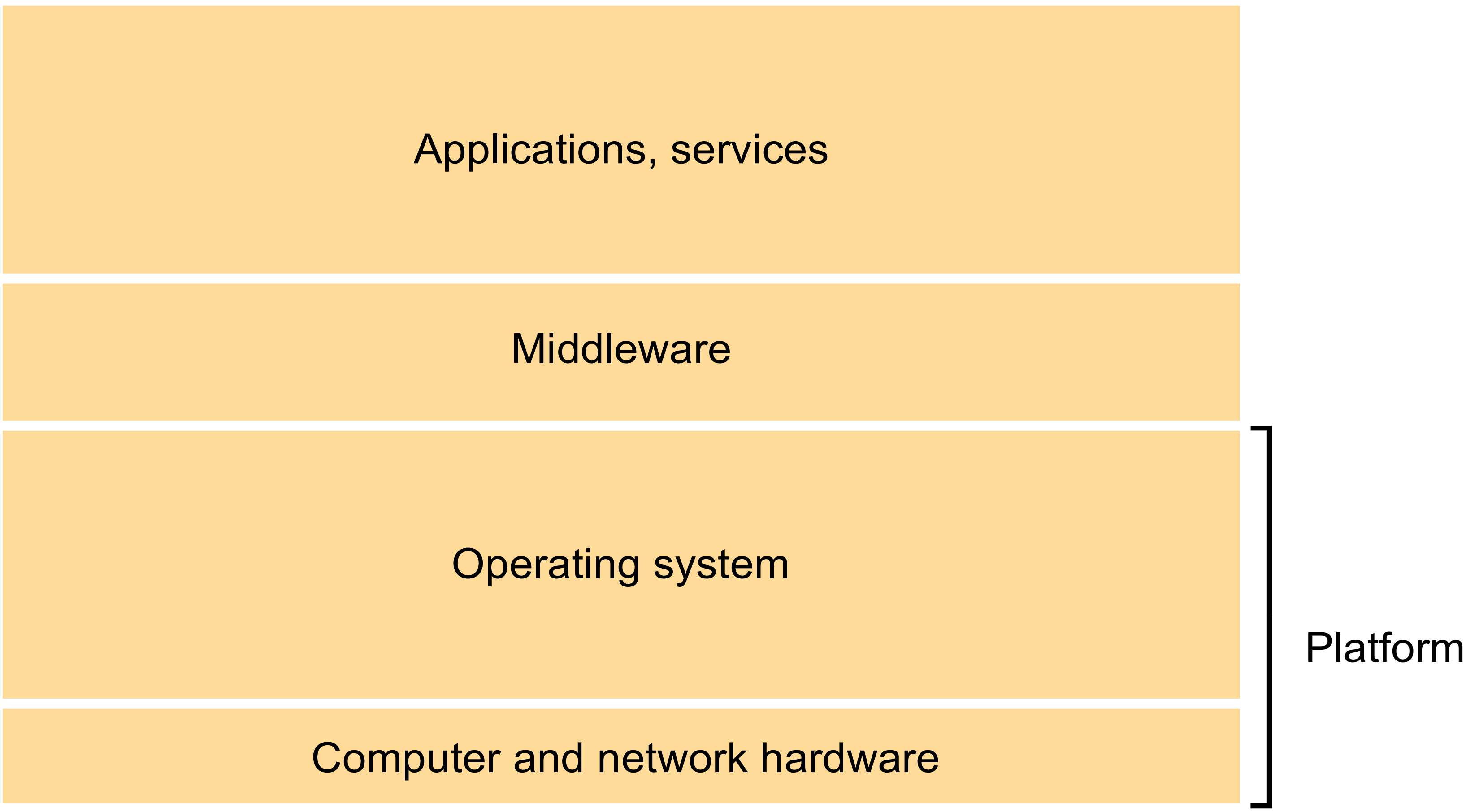


6. Security

- Sensitive information passing through a number of systems, is complex to secure
  - *E.g.*, passwords, credit card data, medical records
- To secure systems against attacks (incomplete)
  - Viruses, worms, (Distributed) Denial-of-Service (DDoS)
  - In general "cybercrime" cases
- Countermeasures (in super short form)
  - Authentication, authorization, encryption, non-repudiation

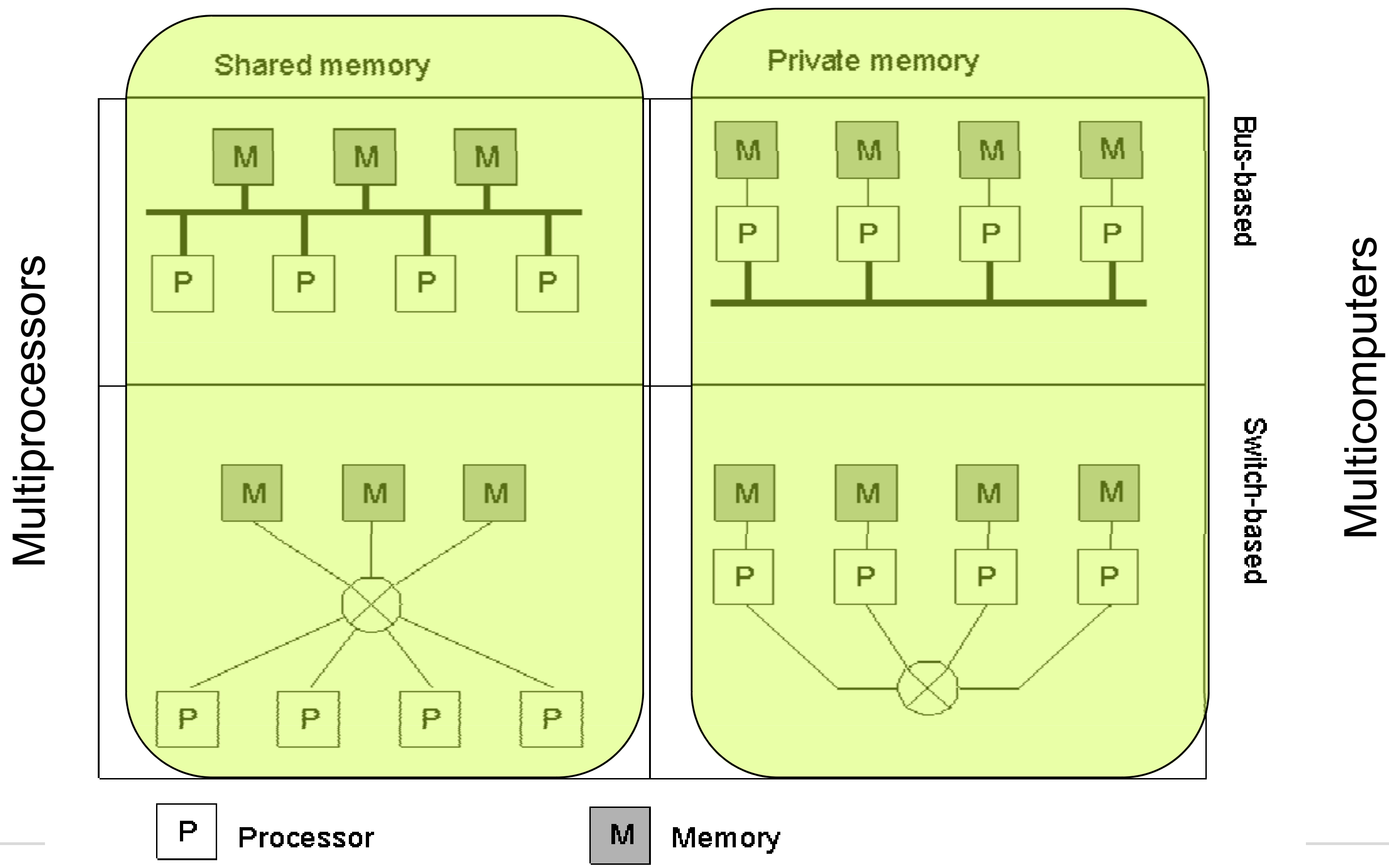
- Security follows the "weakest link" principle
  - Security measures are defined by a risk assessment

Software and Hardware Layers



Hardware Concepts

- Different basic organizations and memories in distributed computer systems

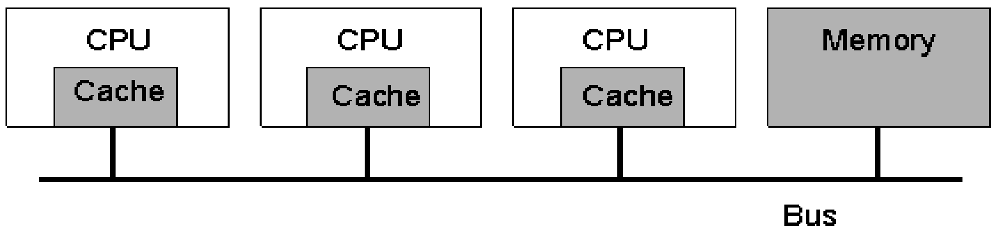


ifi

Bus-based Multiprocessors

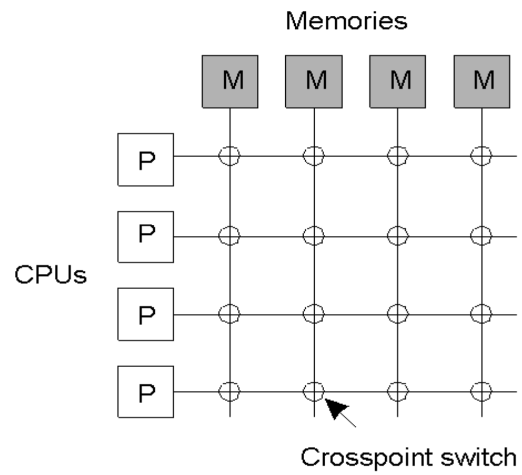
- All processors share the same bus to access memory
  - Advantages
    - Simpler and cheaper construction
  - Drawback
    - Not scalable: with more than a few processors, the bus is saturated

- Caches introduced to prevent bus saturation
  - Consistency problems
  - Need to invalidate other caches after every write

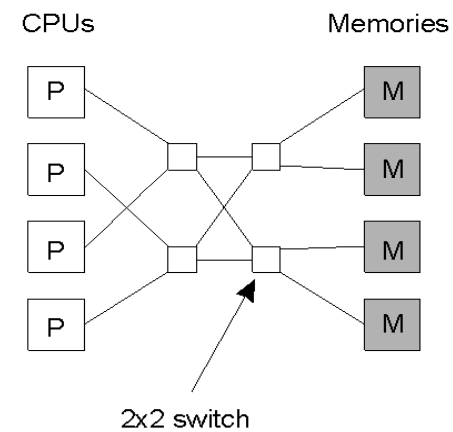


Switch-based Multiprocessors

- Concurrent memory access by multiple processors
  - Although not all combinations useful
- Advantage
  - Increased concurrency → gives speed
- Drawback
  - Delay due to many switches, expensive linkage, and fast crosspoint switches



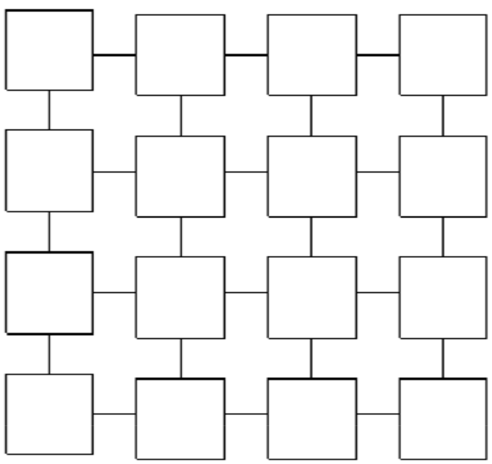
(a)

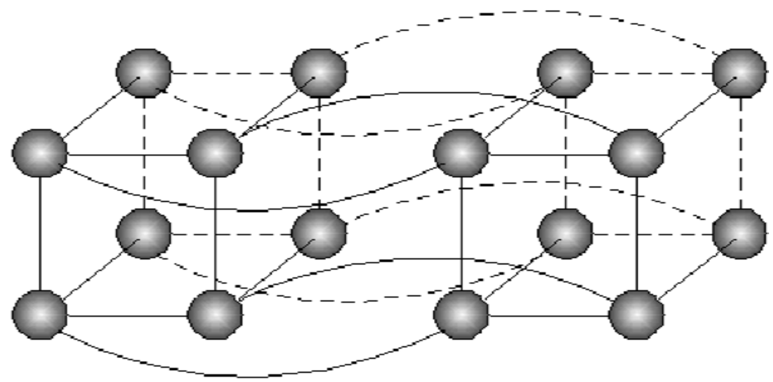


(b)

Switch-based Multi-computers

- Different topologies of interconnection

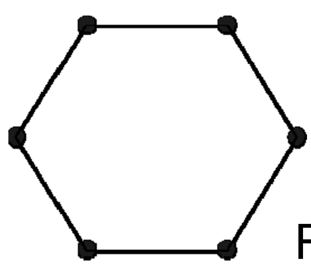




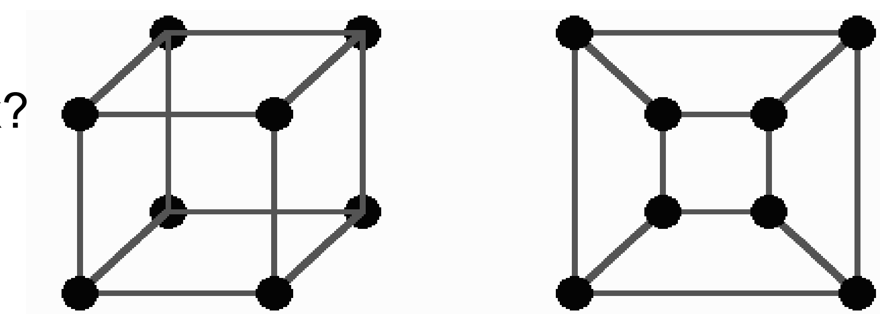
Grid topology

Hypercube topology

Routing: More Hops?
Structure: More Complex?



Rings are also grids

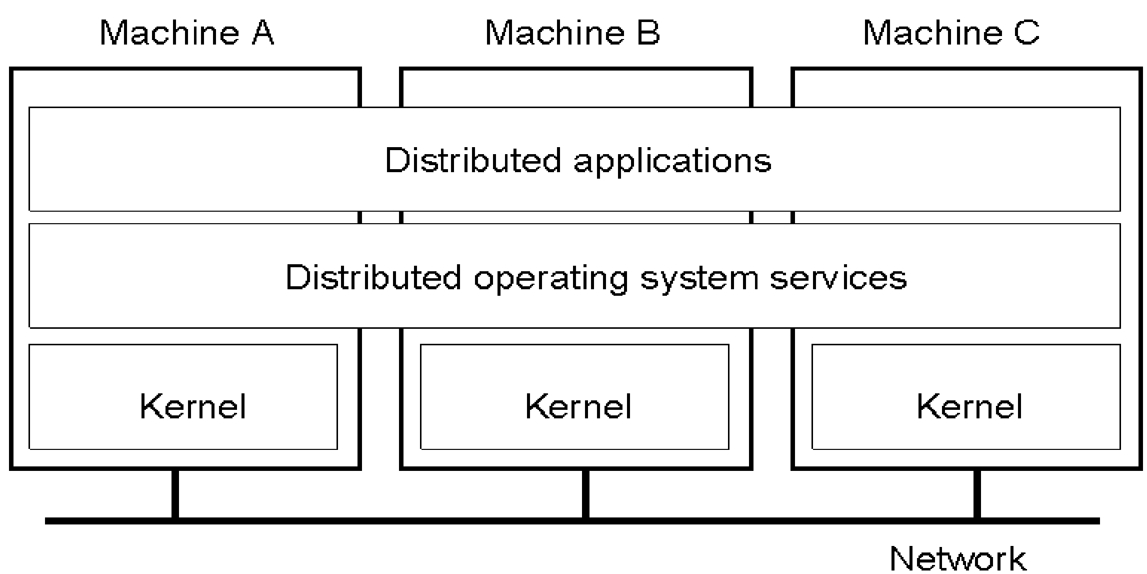


Software Concepts

System	*Description*	*Main Goal*
Distributed Operating System (DOS)	Tightly-coupled operating system for multi-processors and homogeneous multicomputers	Hide and manage hardware resources
Network OS (NOS)	Loosely-coupled operating system for heterogeneous multicomputers (LAN and WAN)	Offer local services to remote clients
Middleware	Additional layer atop of NOS implementing general-purpose services	Provide distribution transparency

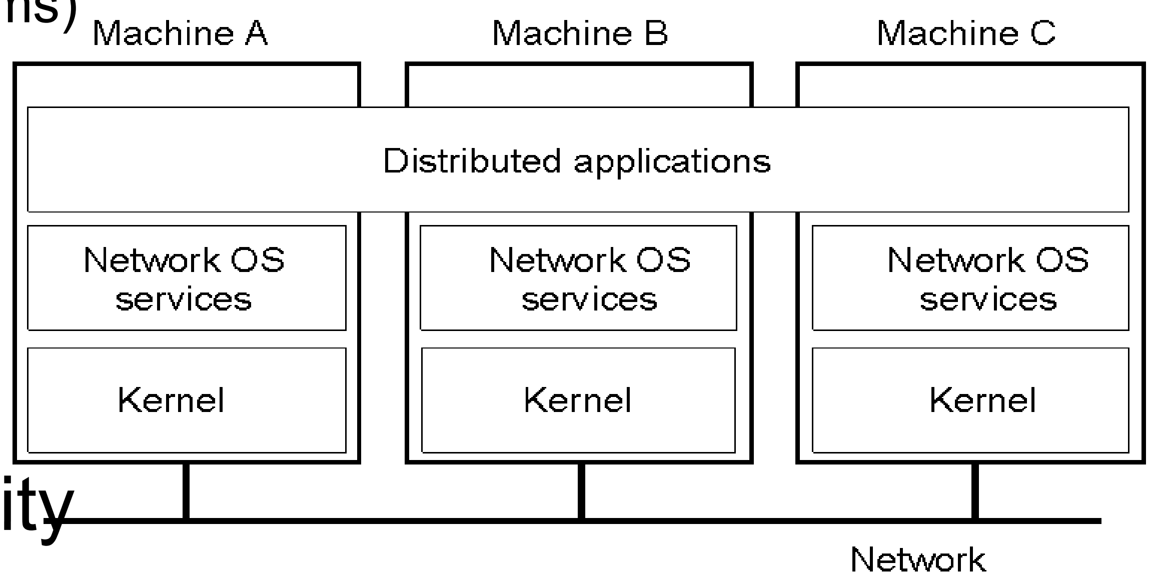
Distributed Operating System (OS)

- Tightly coupled and detailed control
  - Transparent task allocation to a processor
  - Transparent memory access (Distributed Shared Memory)
  - Transparent storage
- Provides complete transparency and a single view of the system
  - Requires multi-processors or homogeneous multi-computers



Network OS

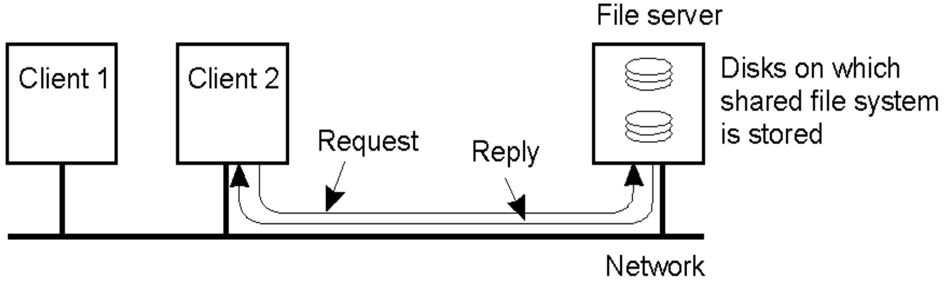
- Loosely coupled and less control
  - Provides services such as
    - rlogin (remote login)
    - ftp (file transfer protocol)
    - scp (secure copy)
    - NFS (network file systems)
- Not transparent
- No single view of the system
- Very flexible with respect to heterogeneity and participation



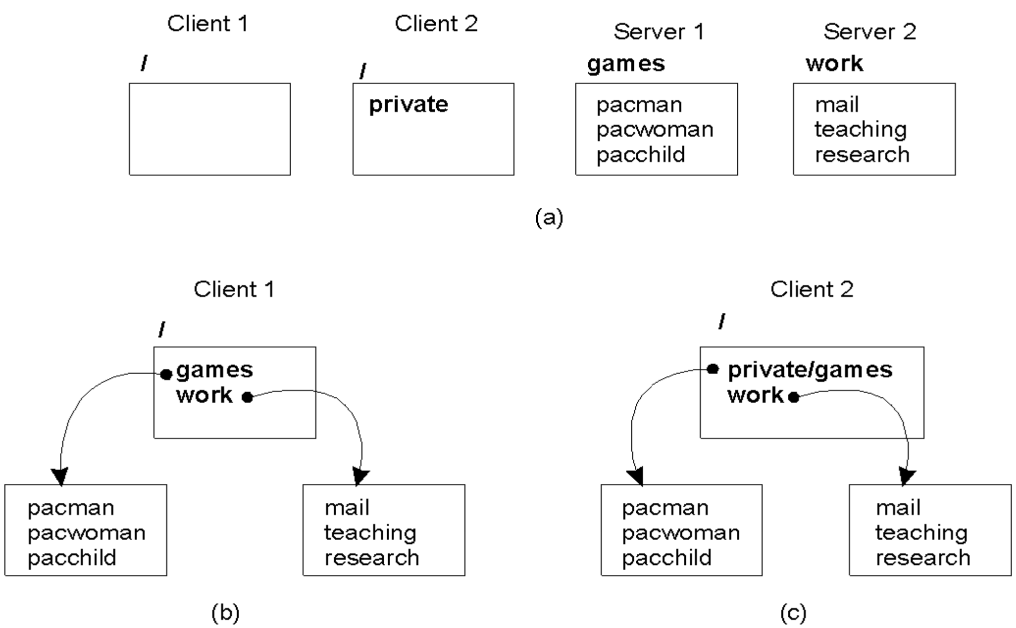
Network OS Instances

- Client/Server architecture

Two clients and a server in a network OS

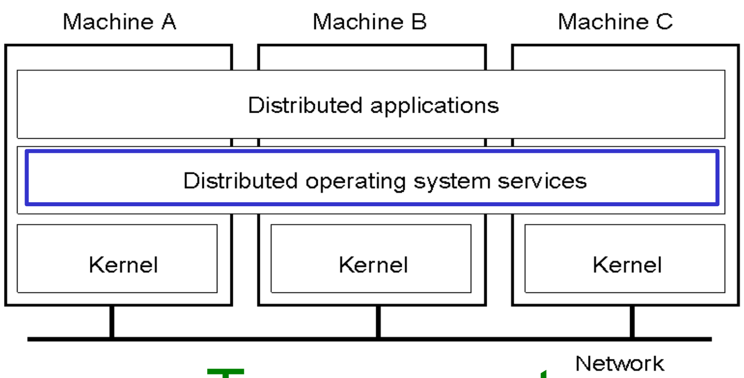


Different clients may mount servers in different places



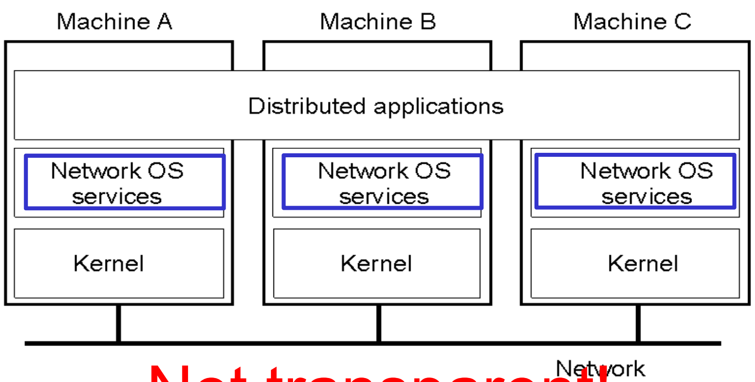
No transparency!

Comparing Distributed OS/Network OS



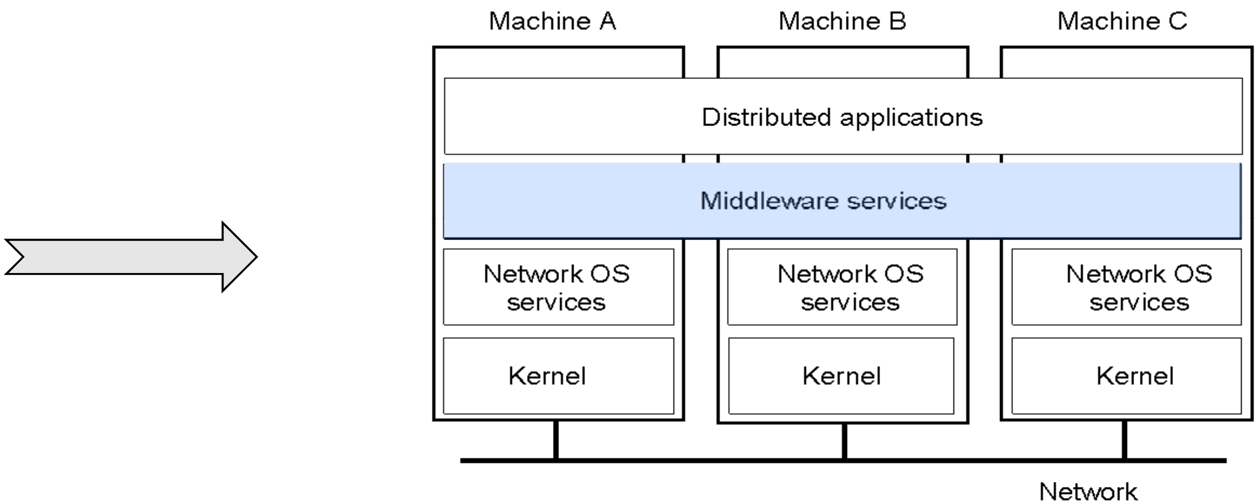
Transparent

Non-autonomous computers!



Not transparent!

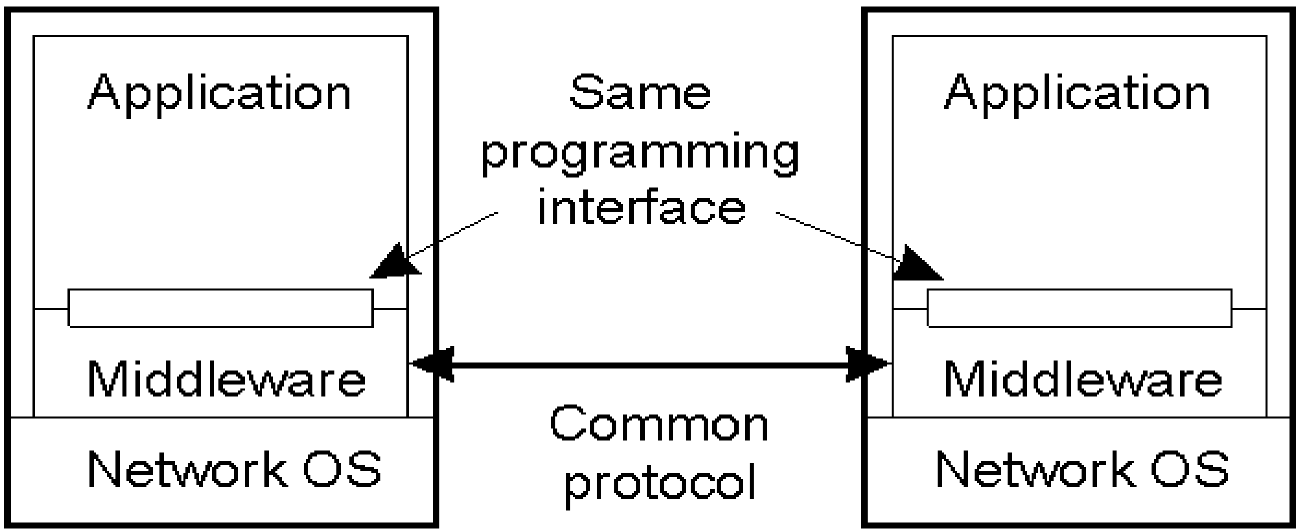
Autonomous computers



Combination of the advantages, while omitting the drawbacks.

Middleware

- In an open system the same protocols are used by each middleware layer and they offer the same interfaces to applications
  - Openness, transparency, (scalability)

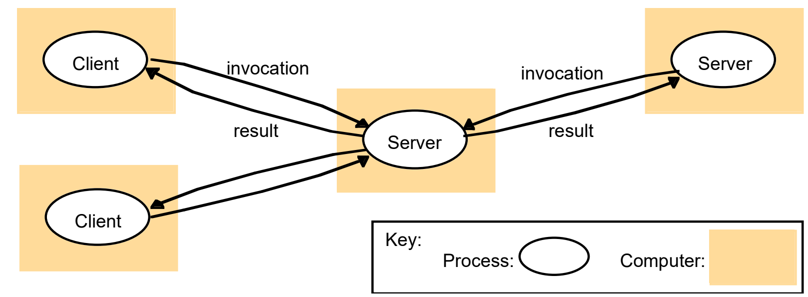


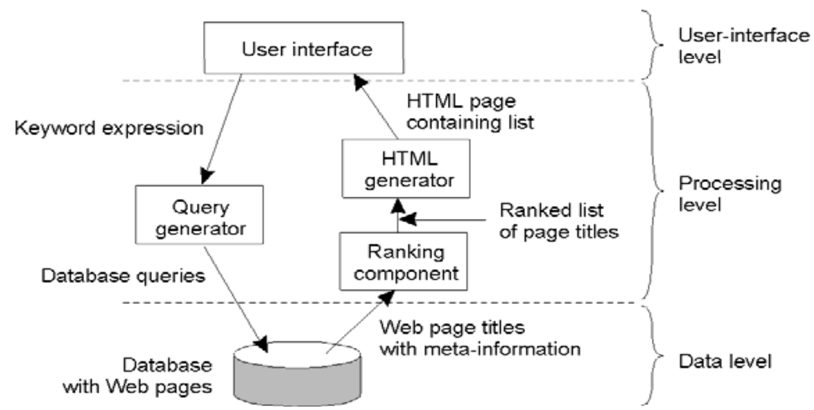
Comparing All Three Alternatives

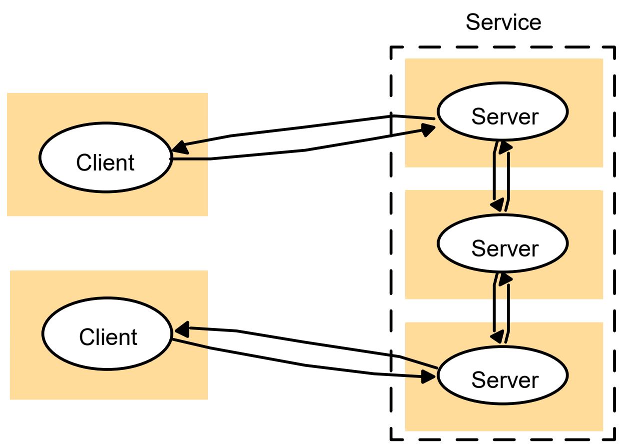
Item	*Distributed OS*		*Network OS*	*Middleware-based OS*
	Multiproc.	*Multicomp.*		
Degree of transparency	Very High	High	Low	High
Same OS on all nodes	Yes	Yes	No	No
Number of copies of OS	1	N	N	N
Basis for communication	Shared memory	Messages	Files	Model specific
Resource management	Global, central	Global, distributed	Per node	Per node
Scalability	No	Moderately	Yes	Varies
Openness	Closed	Closed	Open	Open

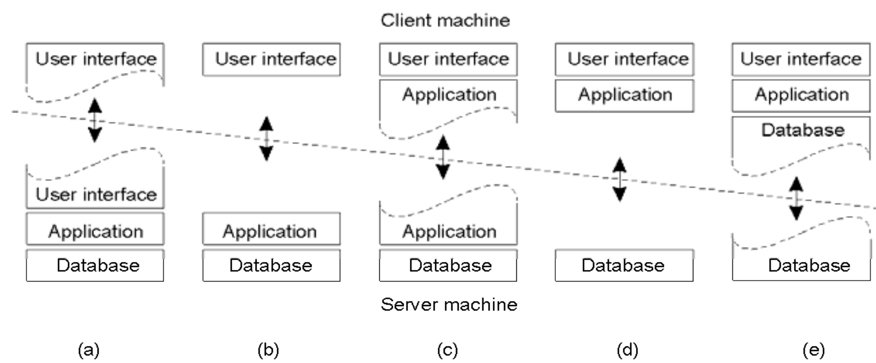
Network Interaction Types (1)

- Client/Server
- 3-tier network application
- Multi-tiered architectures
- Cluster of servers









Network Interaction Types (2)

- Web proxy server
- Code mobility
- Thin clients
- Overlay networks