Chapter 1: Characterization of Distributed Systems

From **Coulouris, Dollimore, Kindberg and Blair**
Distributed Systems: Concepts and Design

Edition 5, © Addison-Wesley 2012

Outlines

- Introduction – Distributed Systems (DS)
- Examples of DS
- Trends in DS
- DS Challenges
- Parallel vs. Distributed Processing
- Parallel vs. Distributed Storage System

Distributed Systems (DS)

Distributed System:

- Multiple computers that are distributed within some geographical area and communicate through a computer network.
- They interact with each other to achieve a common goal.
- Computers in DS communicate with each other through massage passing.

Common Goal: A service for a particular domain.

Figure 1.1 (see book for the full text)
Selected application domains and associated networked applications

Finance and commerce	eCommerce: Such as Amazon and eBay, PayPal, online banking and trading.
The information society	Web information and search engines, ebooks, Wikipedia; social networking: Facebook and MySpace.
Creative industries and entertainment	online gaming, music and film in the home, user-generated content. Examples: YouTube & Flickr
Healthcare	Health informatics, on online patient records, monitoring patients
Education	E-learning, virtual learning environments; distance learning
Transport and logistics	GPS in route finding systems and map services: Google Maps, Google Earth, etc.
Science	The Grid as an enabling technology for collaboration between scientists (storage, computing, … etc)
Environmental management	Sensor technology to monitor earthquakes, floods or tsunamis

Distributed Systems (DS)

- There are several autonomous (independent) computational entities, each one has its own local memory.
- Each entity is called "Node".
- The entities communicate through massage passing.
- Distributed systems is used to solve large computational problems. Each problem is decomposed into small sub-problems and distributed to different nodes to solve the problem.

Distributed Systems (DS)

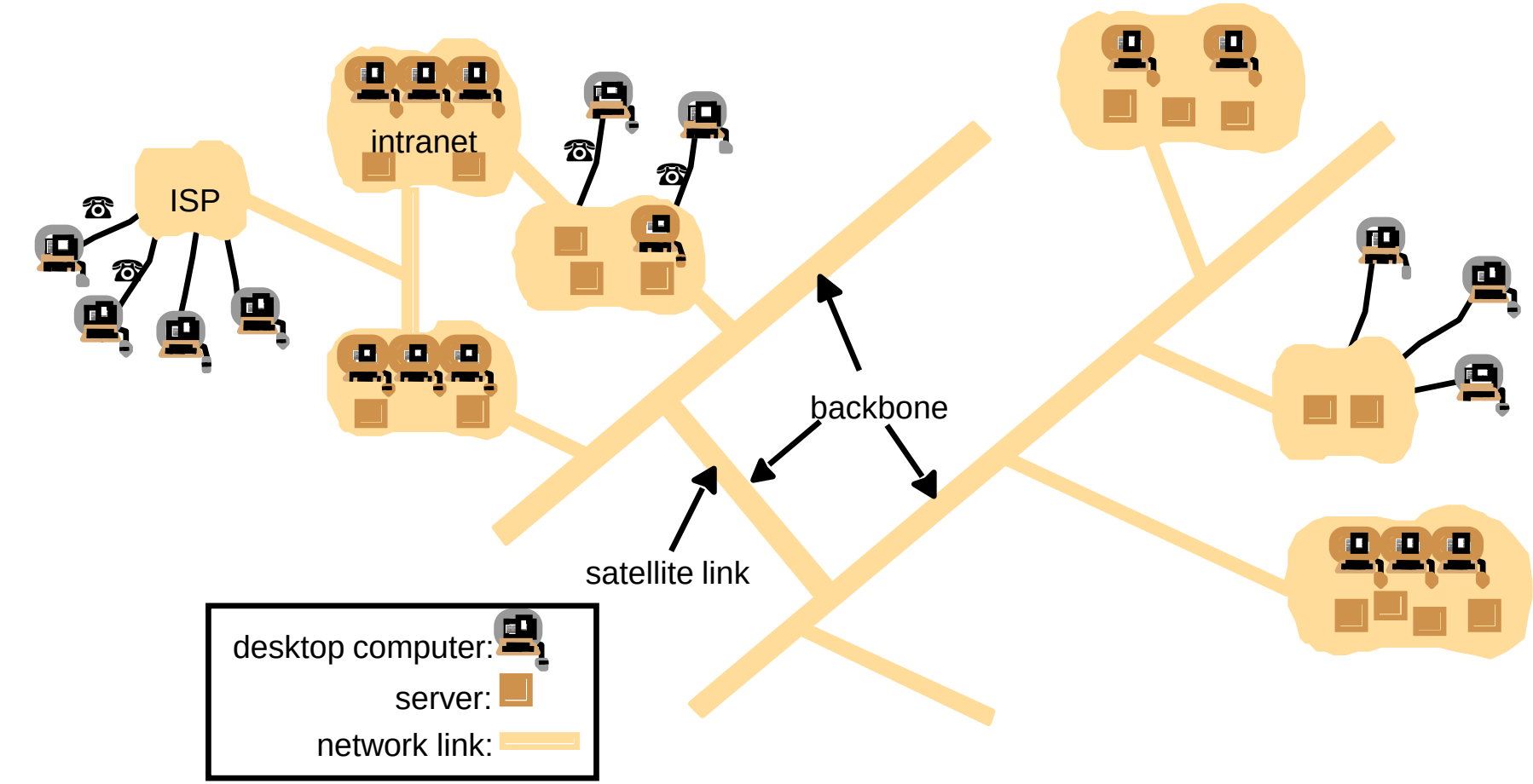
- DS provides high performance (Mainly: Processing and Storage), fault tolerance, reliability, scalability, availability.
- The major aim of DS is to make resource sharing (HW/SW) for more computational and storage power and larger geographical coverage.
- Computer resources: Hardware (HW): Processor, memory, I/O (including storage). Software (SW): OS + distributed programs.

Trends in DS

Pervasive (distributed everywhere) networking and the modern Internet:

1) All computers and computing resources are connected with each other using different network topologies.
2) Each computer/server has processor, OS, HW, and can communicate with other computer/server using massage passing.
3) Internet protocols are used.
4) Computers communicate with each other to request or provide services.

Figure 1.3
A typical portion of the Internet

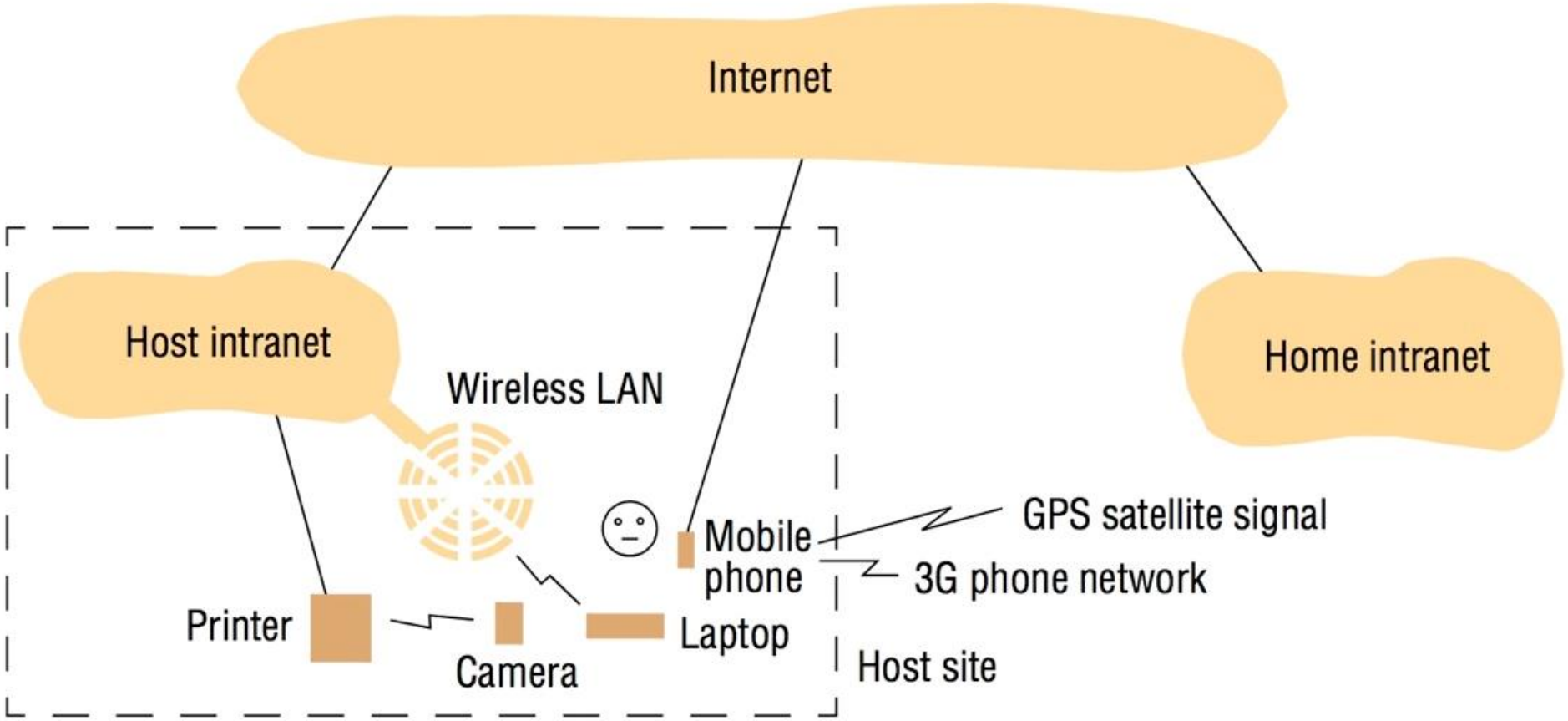


Trends in DS

Mobile and ubiquitous (everywhere) computing:

1) Portable devices (Examples: laptops, smart phones, embedded systems (special purpose)) can join a mobile DS.
2) Computing becomes available everywhere.
3) You can control your home devices remotely.
4) You can join a host network.
5) The system can be scaled up easily.

Figure 1.4
Portable and handheld devices in a distributed system



Trends in DS

Distributed multimedia systems:

- Provides real time multimedia relationship among the communicating entities. Examples include video conferencing, peer to peer IP telephony (e.g., Skype).
- QoS (Quality of Service) is used to determine the maximum delay or latency.

Distributed computing as a utility (cloud computing):

- Infrastructure as a service (IaaS) (Use remote (Disk, Memory, and CPU))

  - Example, execute huge task on remote servers

- Software as a Service (SaaS)

  - Examples, Google docs, gmail, youtube, OneDrive, DropBox

- Platform as a Service (PaaS)

  - Examples, MS Azure, Google Cloud services such as Firebase Push Notifications

Web servers

- Example, Hosting

Figure 1.5
Cloud computing

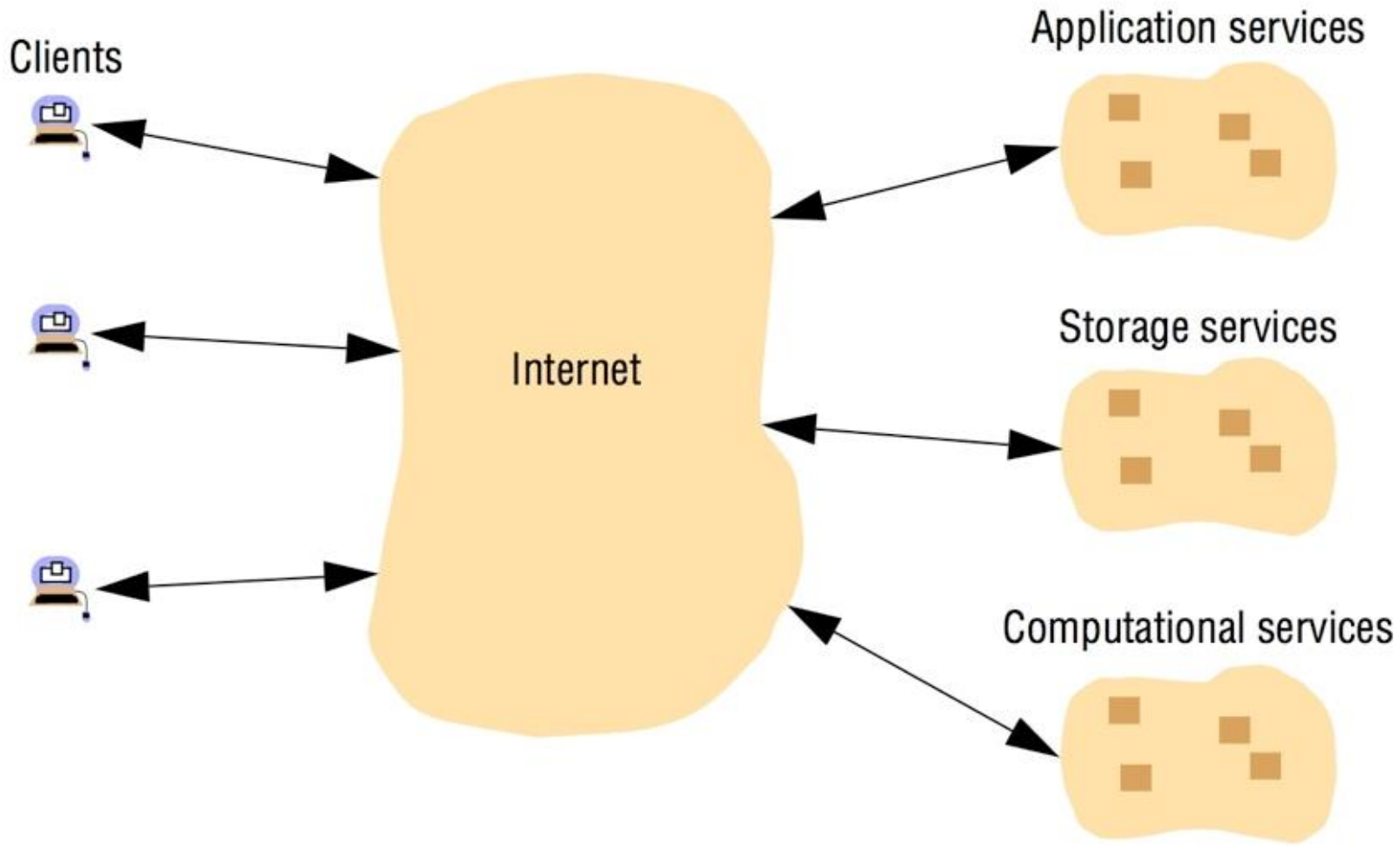


Figure 1.7
Web servers and web browsers

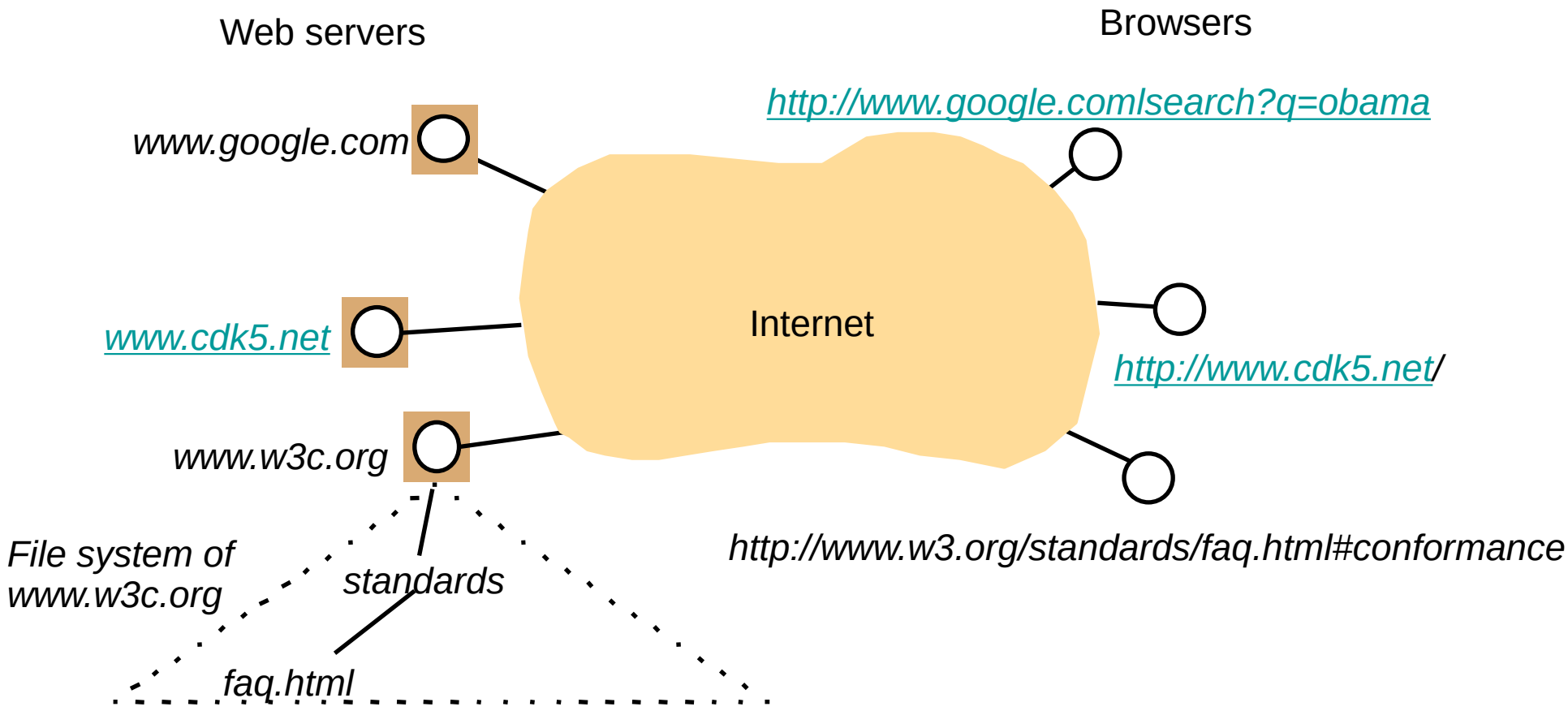


Figure 1.6
Growth of the Internet (computers and web servers)

Date	*Computers*	*Web servers*	*Percentage*
1993, July	1,776,000	130	0.008
1995, July	6,642,000	23,500	0.4
1997, July	19,540,000	1,203,096	6
1999, July	56,218,000	6,598,697	12
2001, July	125,888,197	31,299,592	25
2003, July	~200,000,000	42,298,371	21
2005, July	353,284,187	67,571,581	19

Section 1.5.7
Transparency

***Access transparency*:**

- Enables local and remote resources to be accessed using identical operations.
- The way user uses the system does not let him know if the resource he is accessing is local or remote.

***Location transparency*:**

- Enables resources to be accessed without knowledge of their physical or network location (for example, which building or IP address).
- User cannot tell if the resource he is accessing resides for example in Asia or Europe.

***Concurrency transparency*:**

- Enables several processes to operate concurrently using shared resources without interference between them.
- User think he is using the system by himself, although, other users can be using the system with him.

Section 1.5.7
Transparency

***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.
- User is not aware that there might be several copies of the resource he is using.

***Failure transparency*:**
- Enables the concealment of faults, allowing users and application programs to complete their tasks despite the failure of hardware or software components.
- System recovers from failure, so, user might not notice the failure and he can use the system normally.

***Mobility transparency*:**
- Allows the movement of resources and clients within a system without affecting the operation of users or programs.
- For example, user is accessing a file and he cannot tell that a file is moved from location to another location.

Section 1.5.7
Transparency

***Performance transparency*:**

- Allows the system to be reconfigured to improve performance as loads vary.
- User feels the system is functioning properly although there might be huge load on servers.

***Scaling transparency*:**

- Allows the system and applications to expand in scale without change to the system structure or the application algorithms.
- User feels the system is functioning properly even when number of users increases.

The definition of DS has the following significant consequences:

- **Concurrency:** DS provides resource sharing where several programs may be working concurrently. Coordination between the running programs is an important issue.

- **No global clock:** programs working on a DS coordinate using massage passing. This creates limits to the accuracy of which the computers in the network can synchronize their clock (no single global clock).

- **Independent failure:** Each component of a DS (resources) can fail, leaving the other running.

DS mainly for high processing / storage power

Web search: There are billions of Web Pages on the internet, searching the entire web database is a major challenge. A web search DS can be well designed to provide such service (e.g., Google). It includes:

1) Physical infrastructure that consists of a very large number of networked computers located at data centers all over the world.
2) Well structured distributed storage system to offer fast access to a very large data set.
3) Distributed file system to hold very large files and data required by the searching process and other applications.
4) Synchronization among the distributed nodes.
5) A model that supports the management of a very large parallel and distributed computation among the underlying infrastructure.

Applications such as YouTube, Facebook, Google Maps, DropBox, etc.

DS Challenges

Heterogeneity:

1) Variety of differences in network, computer HW, OS, programming languages, and different developers.
2) Different Internet protocols are used to handle the different network topologies.
3) Data types vary among the programming languages and different HW.
4) Middleware is a SW layer that provides a programming abstraction as well as masking the heterogeneity of the underlying network, HW, OS, and programming languages. Examples, Common Object Request Broker (CORBA) and Java Remote Method Invocation (RMI).

DS Challenges

Openness:

It determines the degree to which resource sharing services can be added and made available for variety of client programs even when using heterogeneous HW and SW (That is, how open the system to interoperating with other systems).

Open system requires:

1) Publishing key interfaces (This way, other systems would know how to interact with the system).

2) Provide uniform communication mechanism and published interfaces for accessing the shared resources. (e.g., Reviewing new protocols and asking for comments from public parties, then adapting the protocols and publishing them. They become uniform in a sense of being standard in industry).

3) Interfaces concept and system calls for abstraction (The way the system work is abstract, and other systems can deal with the system only through interfaces and system calls).

DS Challenges

Security: DS resources are shared and connected through a computer network. They communicate with each other through massage passing.

Security components are:

1) Confidentiality: Protection against disclosure to unauthorized individuals. Attacked by Eavesdropping (التنصت).
2) Integrity: Protection against alteration or corruption. Attacked by massage alteration and replay.
3) Availability: Protection against interference with the means to access the resources. Attacked by Denial of Service (DoS).

DS Challenges

Scalability: Managing the HW and SW resources as the DS scales up.

In particular: reducing cost and controlling the performance loses using appropriate algorithms.

Failure Handling: Resources in a DS partially fail leaving other resources running. This makes it difficult to handle failures.

1) Redundancy is an important technique to handle DS failures. This includes having at least 2 routes between the communicating entities.
2) Storage systems data replication.

DS Challenges

Concurrency: In a DS, several clients (users) and processes attempt to access the shared resources concurrently. Managing concurrency is considered a major challenge (examples on solutions: using semaphores).

Transparency: Concealment (hiding) from the user and the process of the separation of DS components.

- Access Transparency: Access remote resources using same operation.
- Location Transparency: Resources are accessed without the knowledge of their location.

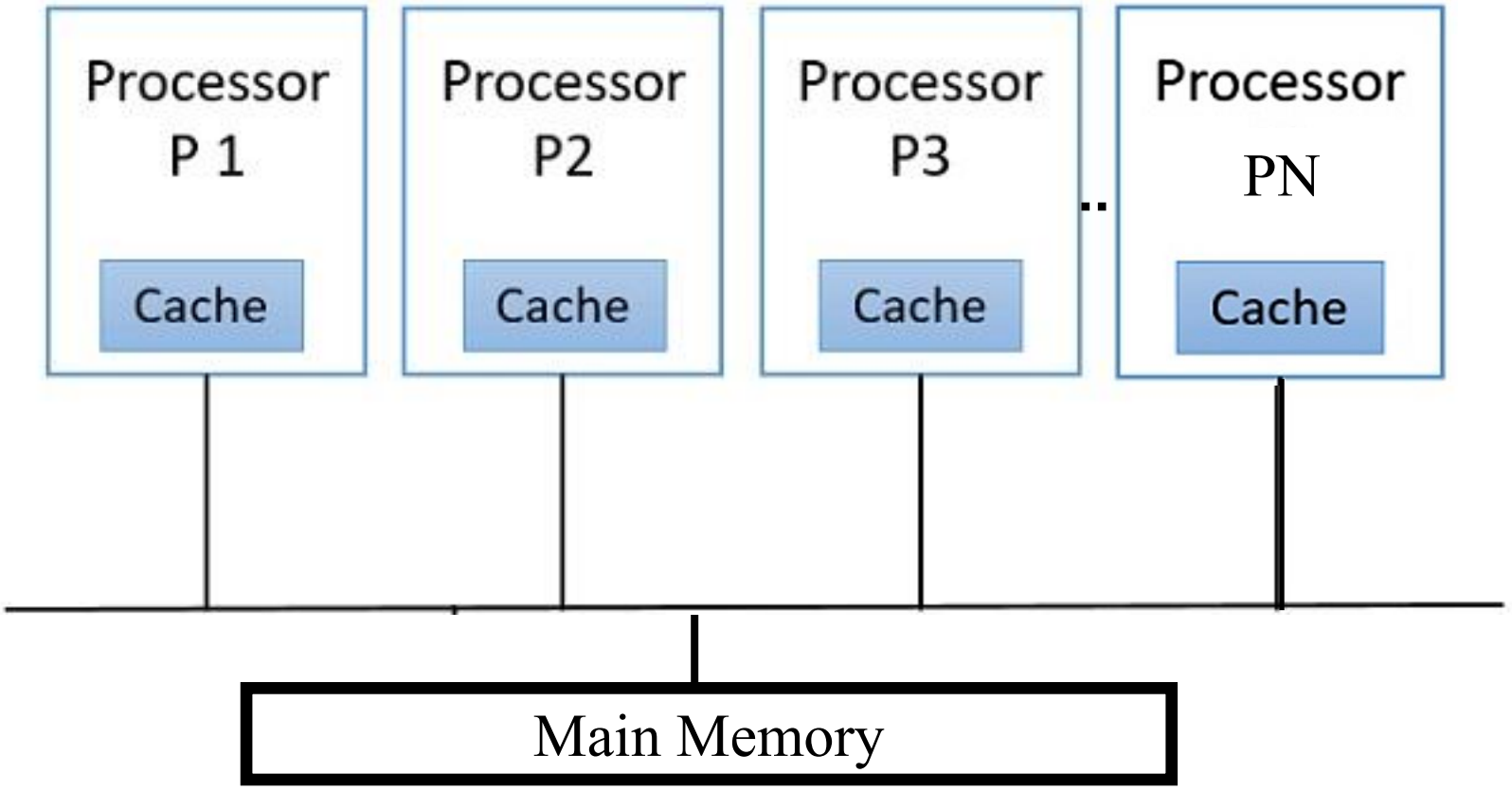
QoS: Internal traffic management to enable a system to meet a deadline especially the real time systems (e.g., multimedia systems, banking systems, etc.).

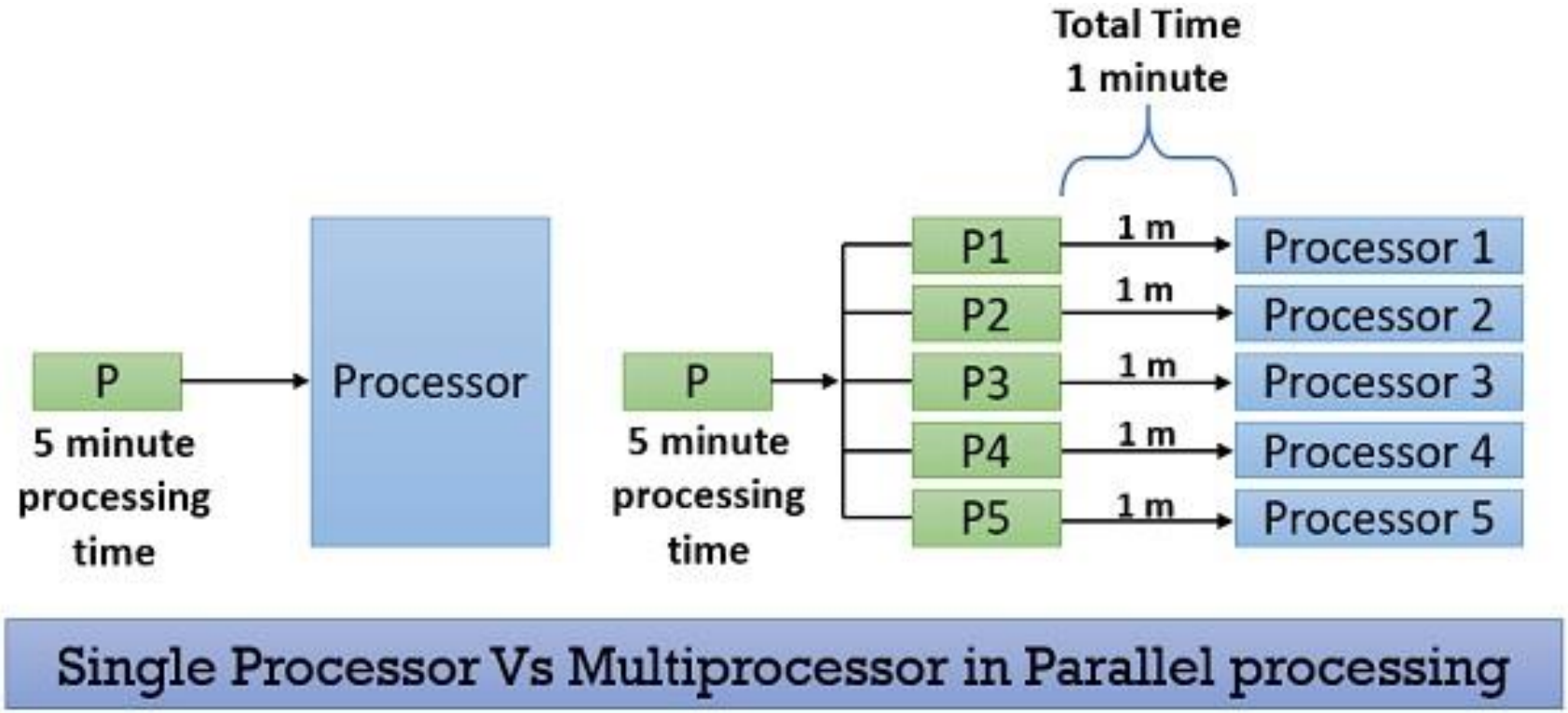
Parallel Processing vs. Distributed Processing

- **Parallel Processing:** All processors have access to a shared memory. Problem is decomposed to sub-problems and each one is assigned to a particular processor. The processor and the memory are in the same node (single node system).

- **Distributed Processing:** Processors and memory chips are distributed into several nodes (Multi-Nodes) . Each processor has its own memory. Problems are decomposed into sub-problems and each one is assigned to a processor at a particular node. Massage passing is used to exchange information.

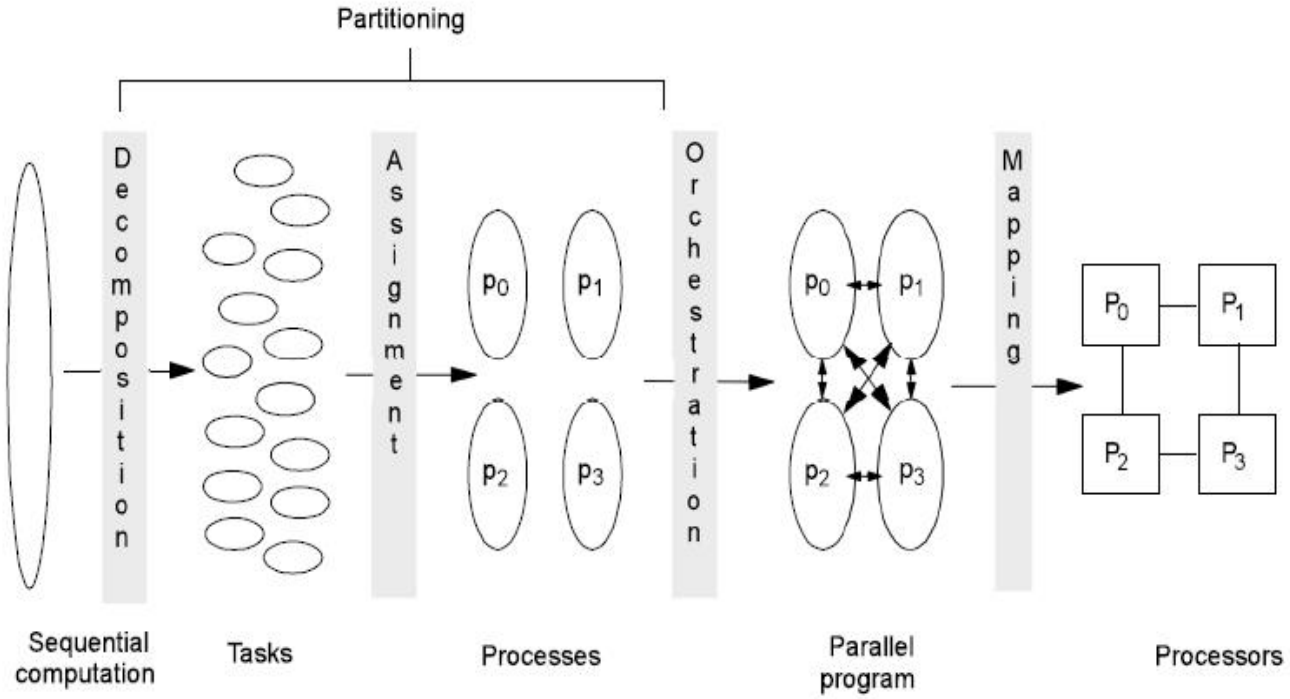
Parallel Processing (Computing) - Single Node



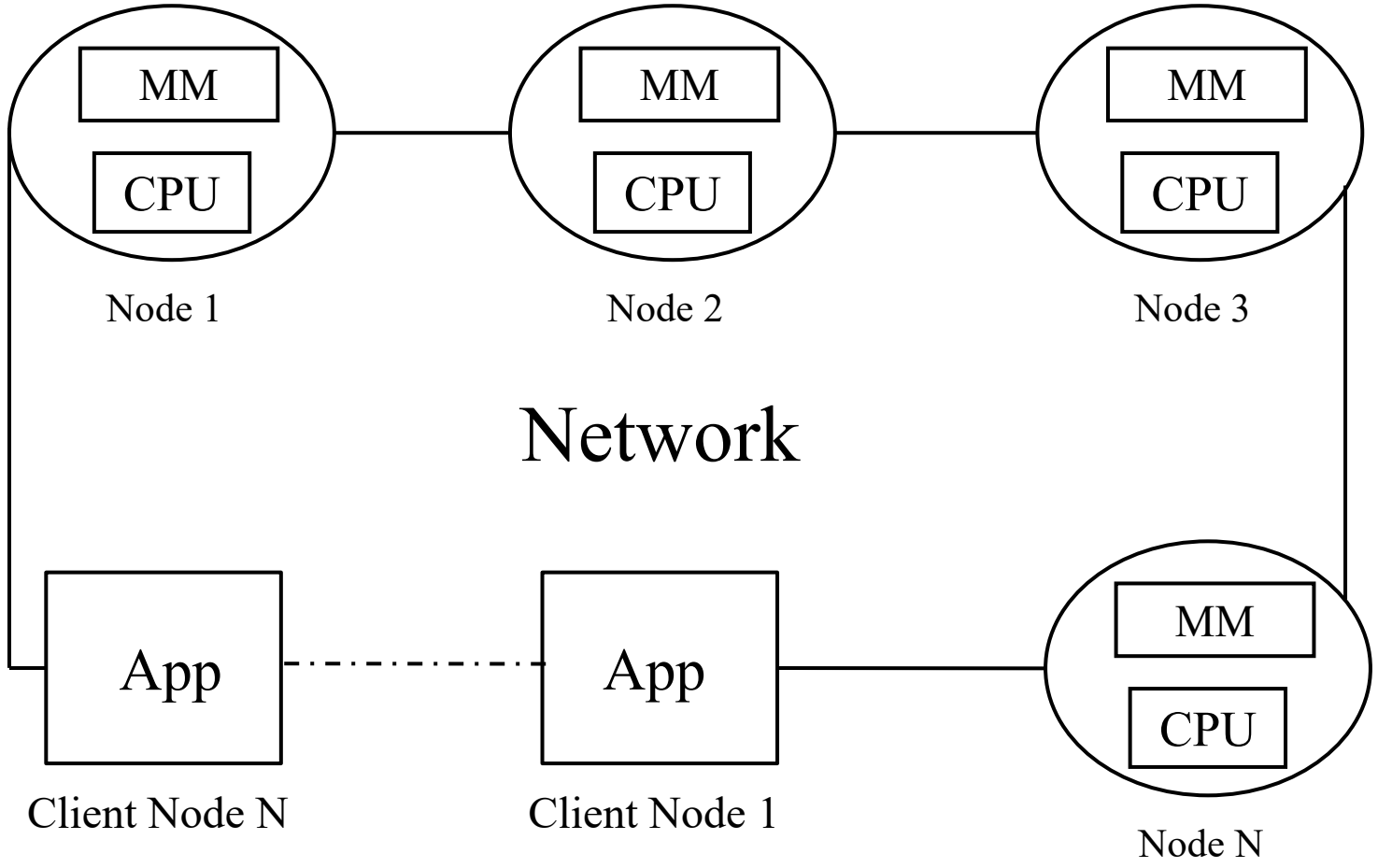


Single Processor Vs Multiprocessor in Parallel processing

Steps in Creating a Parallel Program



Distributed Processing (Computing) - Multi-Nodes



Parallel Storage System vs. Distributed Storage System

Two techniques to parallelize the data:

1) **Data stripping**: Cut the sequential data into segments.
2) **Data replication**: Storing same data in different physical storage devices.

Parallel and distributed storage systems provide high performance (because of parallelism), fault tolerance, reliability, scalability, availability, and high capacity.

Parity Block

- It is an extra block that is stored along with data.
- It helps in error detection.
- It can help in error recovery.
- One type of parity is the even parity (**xor** parity).
- 'Even parity' ensures that the total number of ones is even.

- Suppose data is: 010
- We add a fourth bit as a parity.
- Parity is '1' so that total number of 1's is even.
- So, result is (data and parity): 0101
- Parity bit can also be calculated using **xor** operation
- So, 0 **xor** 1 **xor** 0 = 1

Input 1	Input 2	Output
0	1	1
0	0	0
1	1	0
1	0	1

xor table

Parity Block

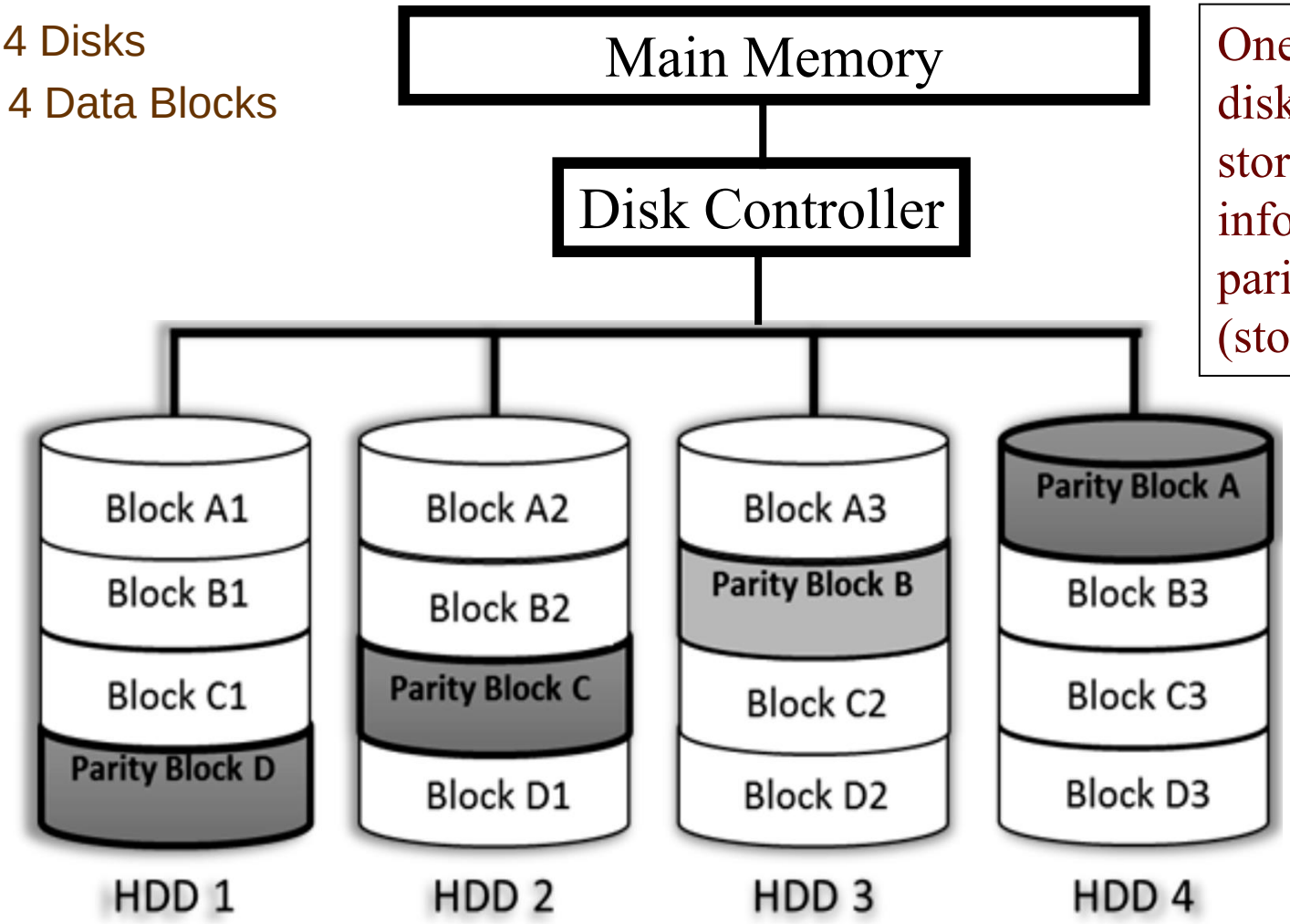
- **Error recovery** example:
  - In the previous example, suppose the second bit is lost
  - 0 ? 0 1
  - Can you tell the content of the second bit?
  - Sure, number of ones must be even, so the second bit must be 1
  - Or use the **xor** operation.
  - 0 xor ? xor 0 = 1
  - So, again the second bit must be 1

Parity Block

- **Error detection** example:
  - In the previous example, suppose the second bit is corrupted and contains 0
  - 0 0 0 1
  - Parity bit is 1
  - But total number of 1s is not even
  - So, we detect there is an error in the data
  - Or, using **xor** operation: 0 xor 0 xor 0 is not 1
  - So, we also detect an error
  - But we cannot detect in this case what bit caused the error

Parallel Storage System – Data Stripping

Ex: 4 Disks
4 Data Blocks

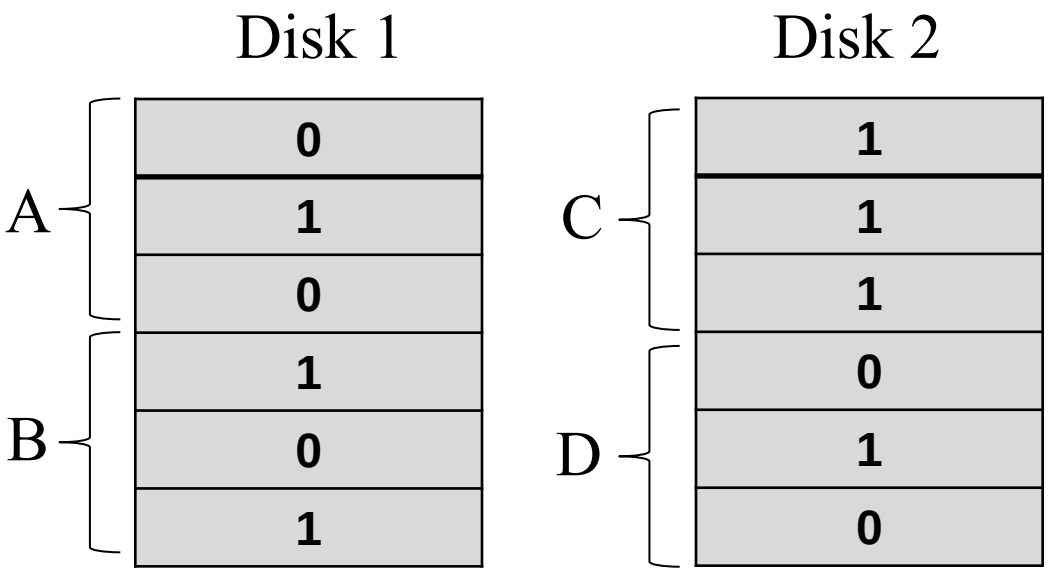


One disadvantage of disk stripping is the storage of extra information such as parity block. (storage space cost)

Parallel Storage System – Data Stripping

- Assume we have 4 disks
- Assume we have 4 data blocks
  - Block A: 010
  - Block B: 101
  - Block C: 111
  - Block D: 010

- Without disk stripping, data is stored sequentially on disk.



If Disk 1 fails, then Blocks A and B are lost.

Parallel Storage System – Data Stripping

- With **disk stripping**, data is stripped across the disks (stored horizontally), and we use a 4th disk for storing parity.

- In this example, we stored parity bits (red color) in different disks. Otherwise, if we store all parity bits in one disk (eg. 4th bit in the 4th disk), then if disk 4 fails, all parity bits are lost.

	Disk 1	Disk 2	Disk 3	Disk 4
A	0	1	0	1
B	1	0	0	1
C	1	1	1	1
D	1	0	1	0

Parallel Storage System – Data Stripping

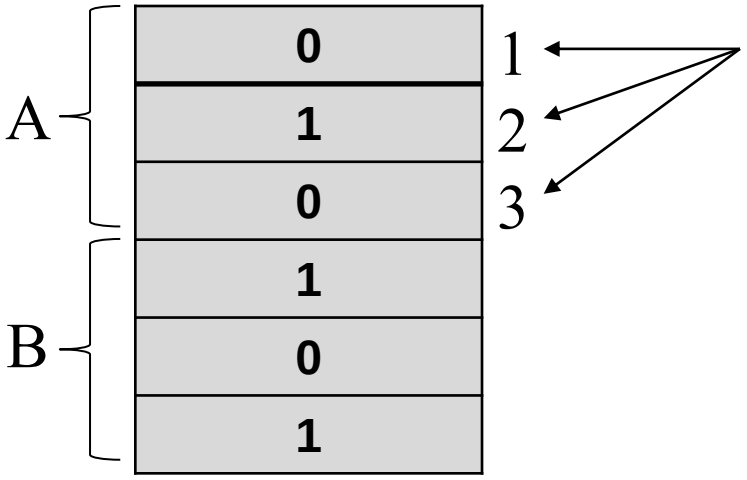
- Now, if for example Disk 1 fails, only part of each block is lost.

- Also, we can recover the contents of Disk 1 using the same parity concept explained before as follows:
  - X1 must be 0 to make total number of 1's even.
  - X2 must be 1 to make total number of 1's even.
  - X3 must be 1 to make total number of 1's even.
  - X4 must be 1 to calculate the parity bit.

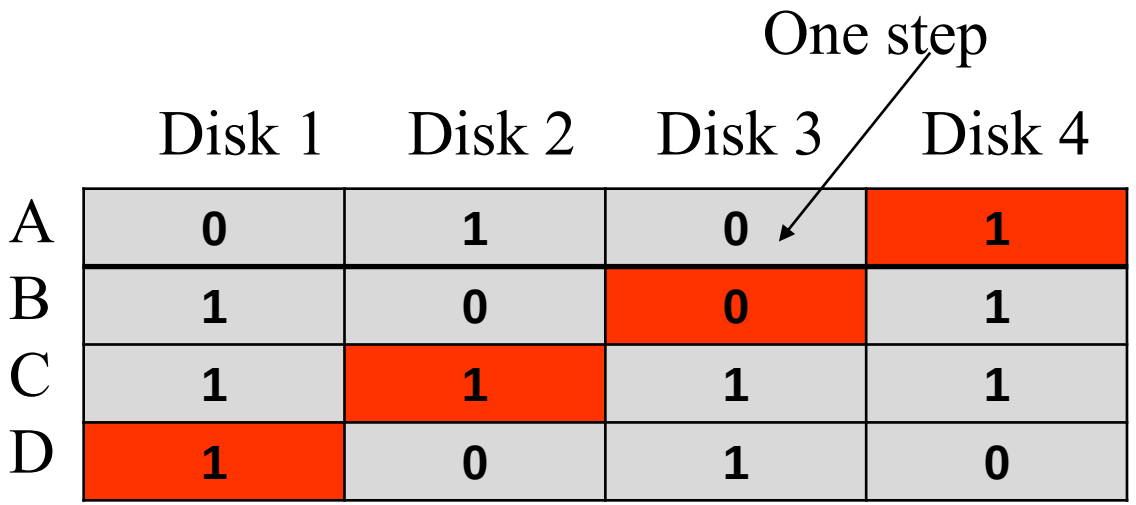
	Disk 1	Disk 2	Disk 3	Disk 4
A	X1	1	0	1
B	X2	0	0	1
C	X3	1	1	1
D	X4	0	1	0

Parallel Storage System – Data Stripping

- **Seek delay is minimized:** Suppose, we want to access the 3rd bit of block "A". If there is no stripping, then read head needs to move from the first bit to the 3rd bit of block "A".



Without Data Stripping



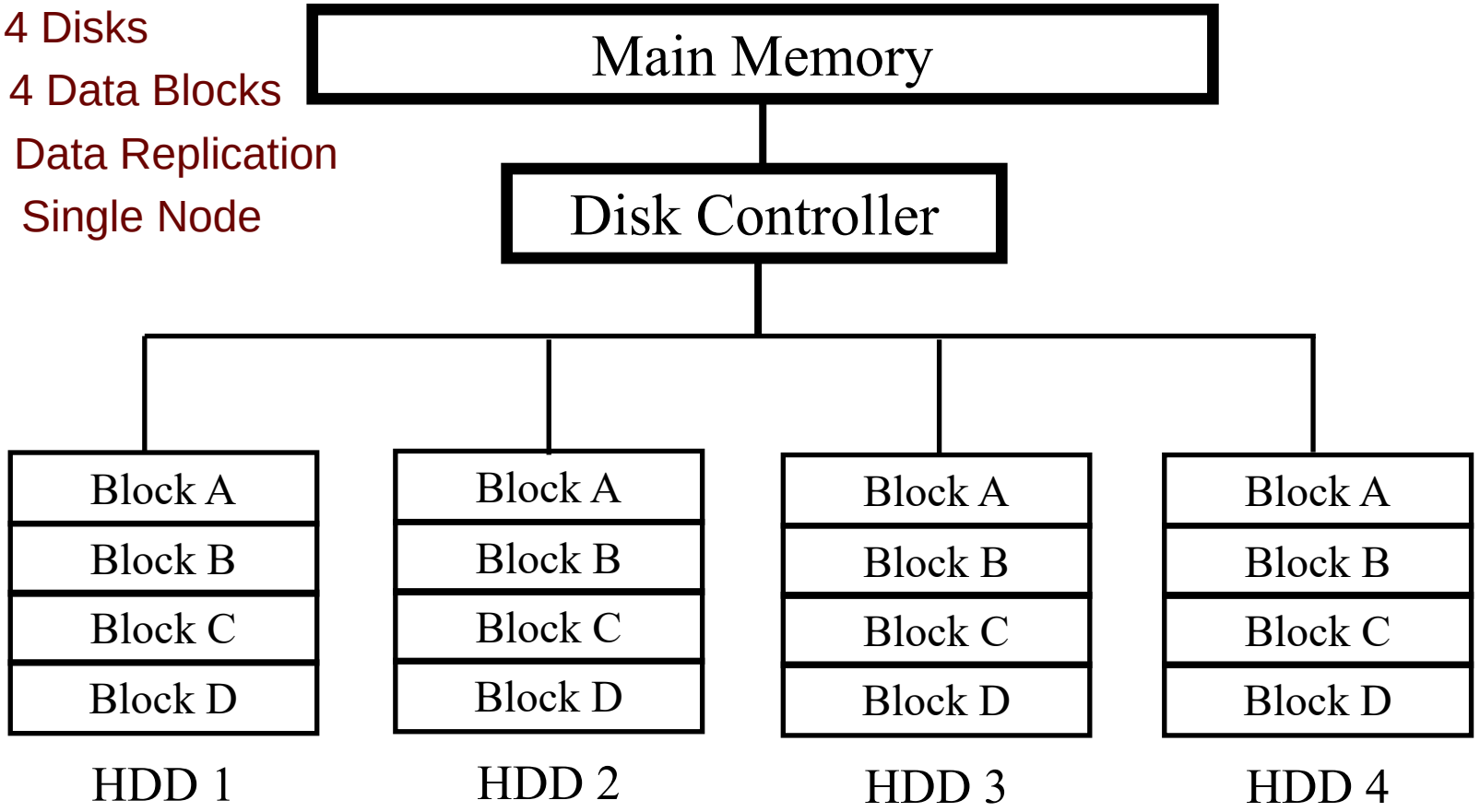
With Data Stripping

Parallel Storage System – Data Stripping

- **Rotational delay is not minimized:** Because even with stripping, disk head will still need to rotate until it reaches the needed block.

Parallel Storage System – Data Replication

Ex: 4 Disks
4 Data Blocks
Data Replication
Single Node



- Pros: 1) Helps in executing parallel tasks 2) No problem if one disk fails.
- Cons: 1) Consuming Storage. 2) Overhead of keeping disks identical.

Distributed Storage System - Multi-Nodes