

Introduction, Parallel and distributed systems

CHAPTER 01

Network-centric computing

- Information processing can be done more efficiently on large farms of computing and storage systems accessible via the Internet.
- Cloud computing is a path to utility computing embraced by major IT companies including: Amazon, HP, IBM, Microsoft, Oracle, and others.

Network-centric Content

- Content: any type or volume of media, be it static or dynamic, monolithic or modular, live or stored, produced by aggregation, or mixed.
- The “Future Internet” will be content-centric.
- The creation and consumption of audio and visual content is likely to transform the Internet to support increased quality in terms of resolution, frame rate, color depth, etc..

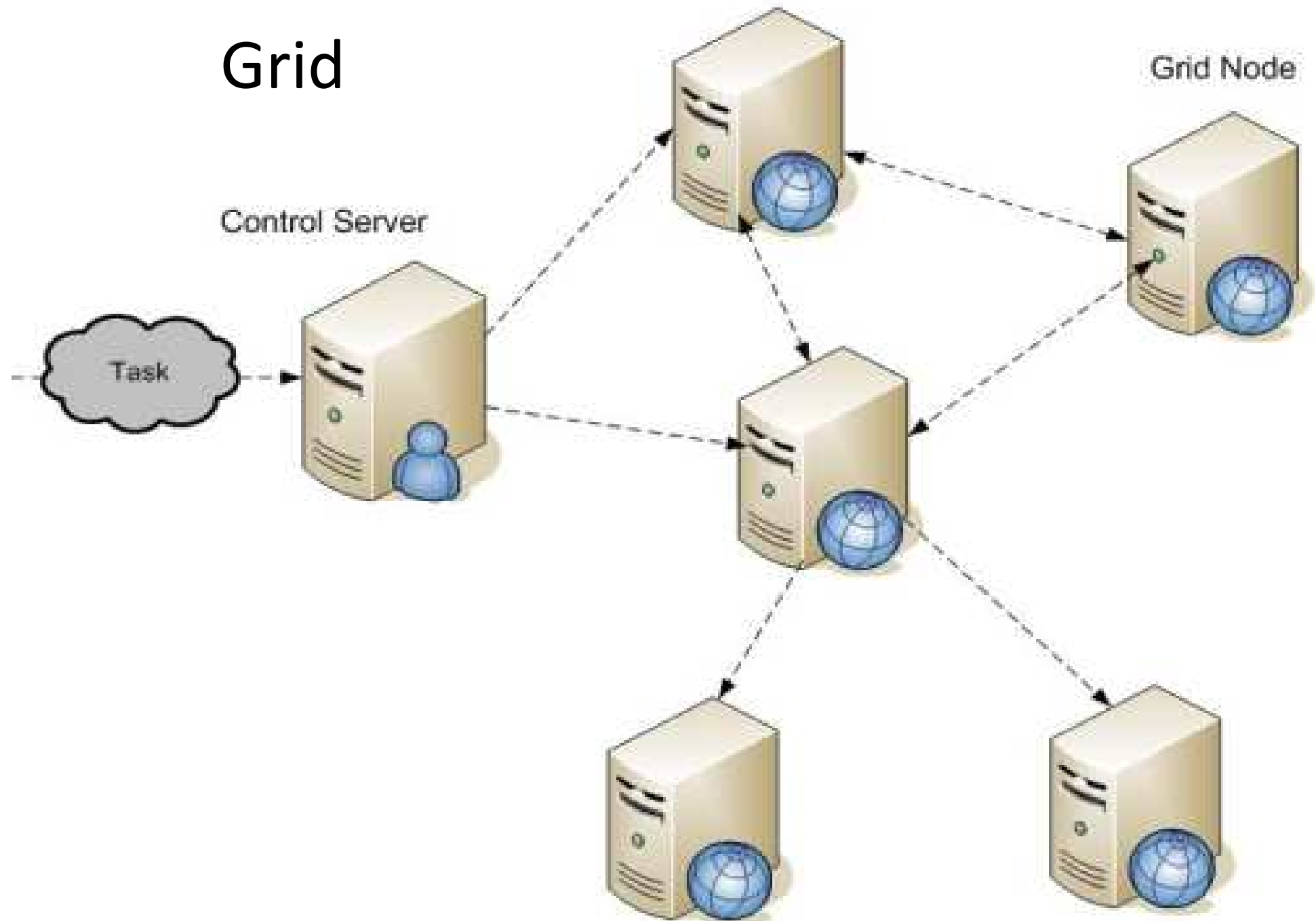
Network-centric computing and content

- Data-intensive: large scale simulations in science and engineering require large volumes of data.
- Network-intensive: Transferring large volumes of data requires high bandwidth networks.
- Low-latency networks for data streaming, parallel computing, computation steering.
- The systems are accessed using thin clients running on systems with limited resources, e.g., wireless devices such as smart phones and tablets.
- The infrastructure should support some form of workflow management.

Evolution of concepts and technologies

- The web and the semantic web
- The Grid - initiated in the early 1990s by National Laboratories and Universities; used primarily for applications in the area of science and engineering.
- Peer-to-peer systems.
- Computer clouds.

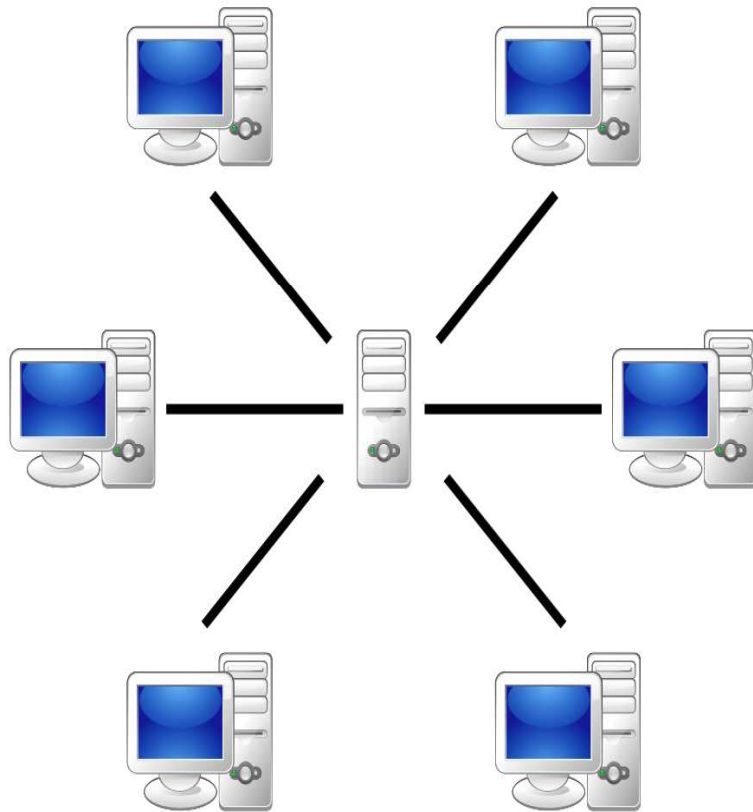
Grid



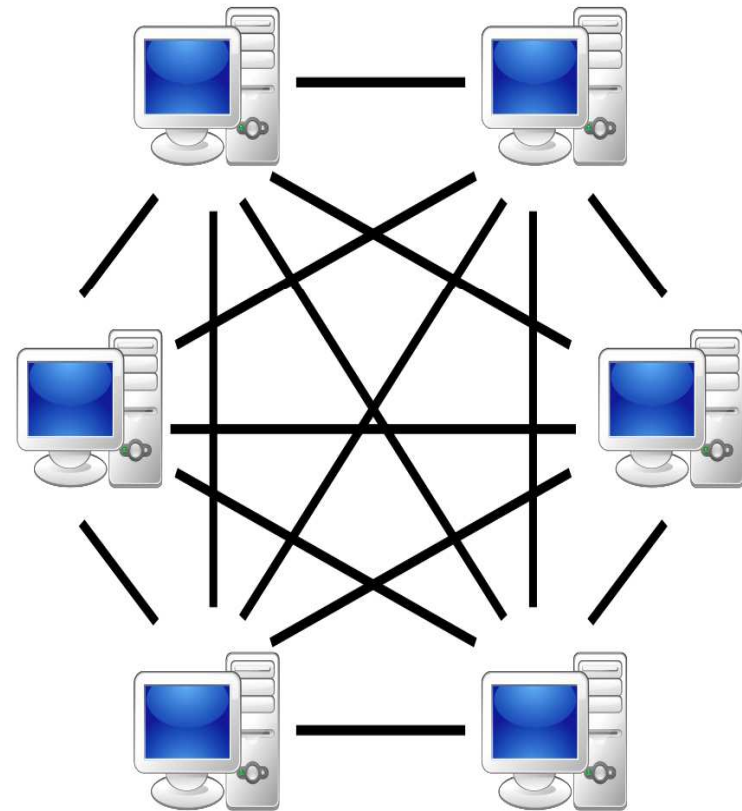
Peer-to-Peer Systems

- In a **P2P** network, the "peers" are computer **systems** which are connected to each other via the Internet.
- Files can be shared directly between **systems** on the network without the need of a central server. In other words, each computer on a **P2P** network becomes a file server as well as a client.

P2P



Server-based



P2P-network

Cloud Computing

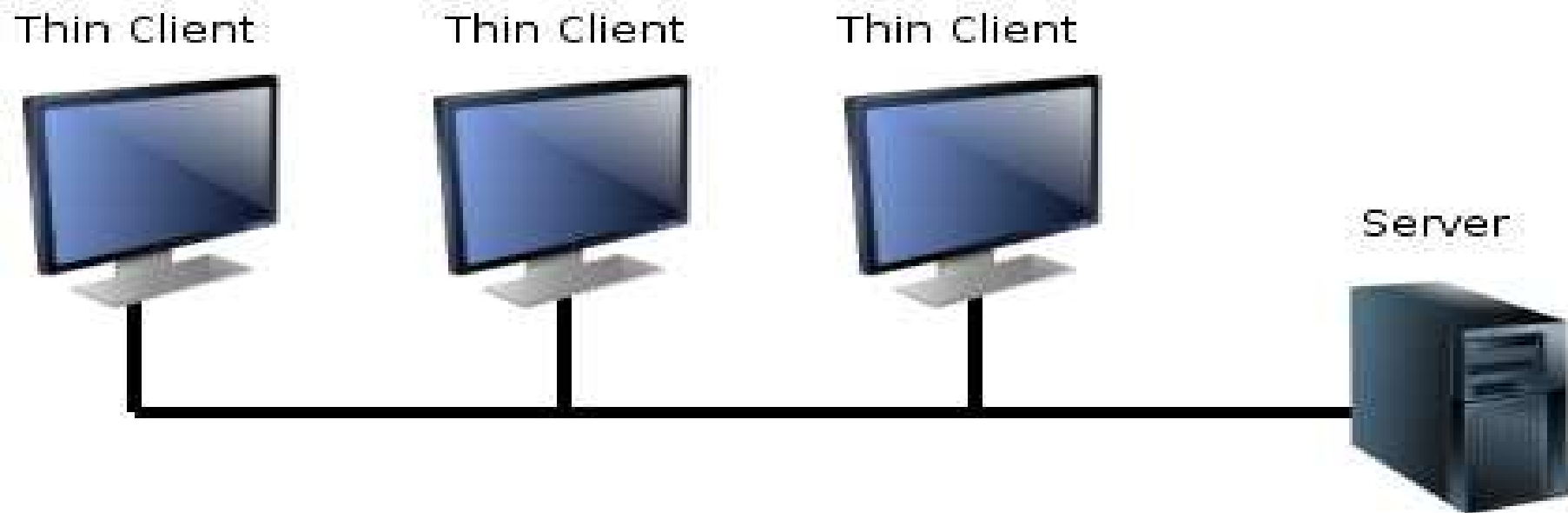
- Uses Internet technologies to offer scalable and elastic services. The term “elastic computing” refers to the ability of *dynamically acquiring computing resources* and supporting a variable workload.
- The resources used for these services can be metered and the *users can be charged only for the resources they used*.
- The maintenance and security are ensured by service providers.

Types of clouds

- Public Cloud - the infrastructure is made available to the general public
- Private Cloud – the infrastructure is operated solely for an organization.
- Community Cloud - the infrastructure is shared by several organizations and supports a community that has shared concerns.
- Hybrid Cloud - composition of two or more clouds (public, private, or community) but bound by standardized technology that enables data and application portability.

Thin Client

- A thin client is a lightweight computer built to connect to a server from a remote location.
- The server does most of the work, which can include crunching numbers and storing information for the thin client.



The “good” about cloud computing

- Resources, such as CPU cycles, storage, network bandwidth, are shared.
- Resources can be aggregated to support data-intensive applications.
- Eliminates the initial investment costs for a private computing infrastructure and the maintenance and operation costs.
- Cost reduction: concentration of resources creates the opportunity to pay as you go for computing.
- Rapid Elasticity: ability of resources to deal with increasing or decreasing demand.
- User convenience.

Benefits

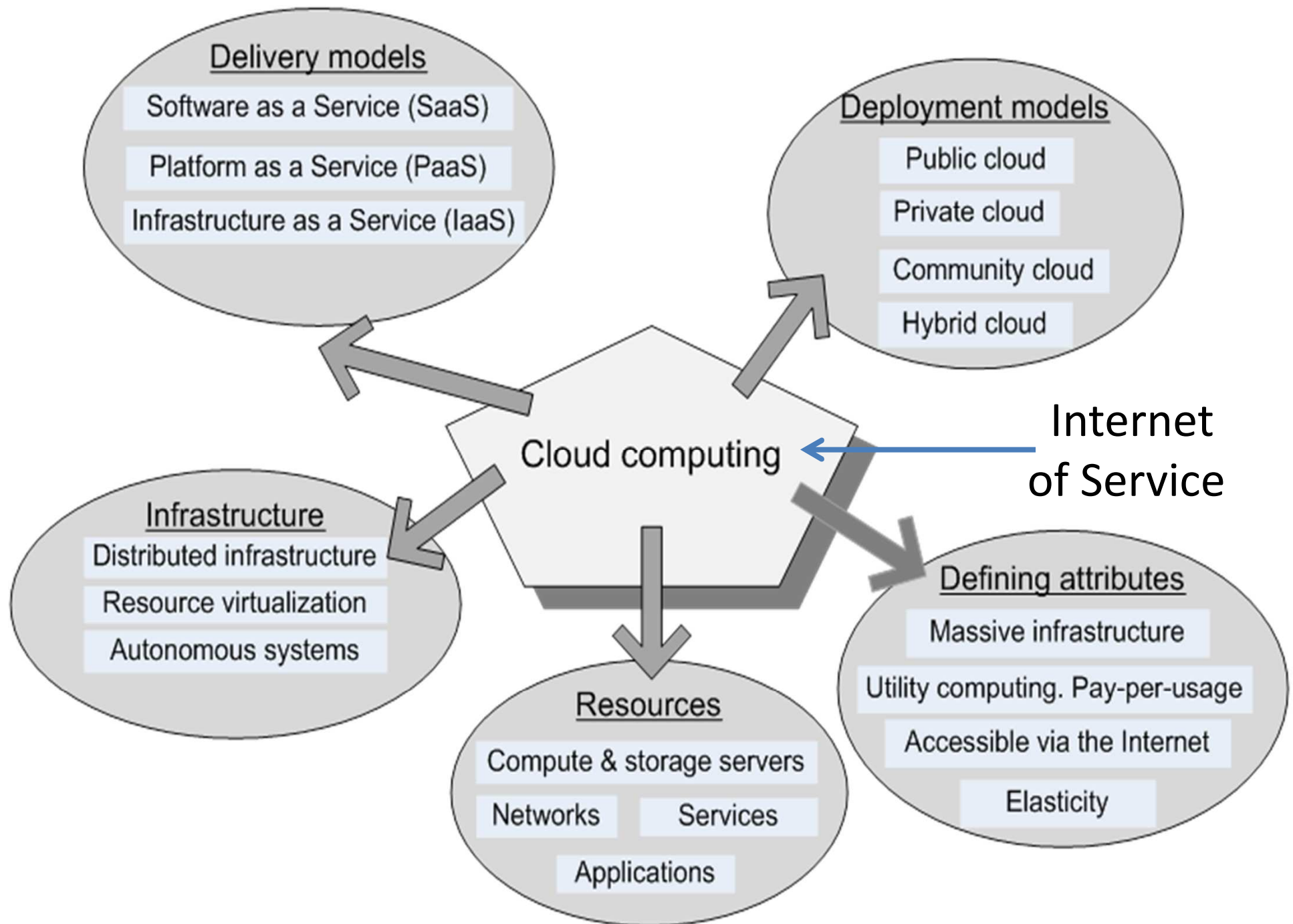
- One can access applications as utilities, over the Internet.
- One can manipulate and configure the applications online at any time.
- It does not require to install a software to access or manipulate cloud application.
- Cloud Computing offers online development and deployment tools, programming runtime environment through **PaaS model**.
- Cloud resources are available over the network in a manner that provide platform independent access to any type of clients.

Benefits

- Cloud Computing is highly cost effective because it operates at high efficiency with optimum utilization. It just requires an Internet connection
- Cloud Computing allows the users to use web services and resources on demand. One can logon to a website at any time and use them.
- Cloud Computing offers load balancing that makes it more reliable.
- Any where and any time access.
- Resource Pooling

Challenges for cloud computing

- Availability of service; what happens when the service provider cannot deliver?
- Diversity of services, data organization, user interfaces available at different service providers limit user mobility; once a customer is hooked to one provider it is hard to move to another (Lock In).
- Data confidentiality and auditability, a serious problem.
- Data transfer bottleneck; many applications are data-intensive.



Cloud delivery models

- Software as a Service (SaaS)
- Platform as a Service (PaaS)
- Infrastructure as a Service (IaaS)

Software-as-a-Service (SaaS)

- This model allows to use software applications as a service to end-users.
- The user does not manage or control the underlying cloud infrastructure or individual application capabilities.
- Services offered include:
- Enterprise services such as: workflow management supply chain, digital signature, customer relationship management (CRM), desktop software, financial management, geo-spatial, and search engine.
- Web 2.0 applications such as: metadata management, social networking, blogs, wiki services, mail and portal services.

Platform-as-a-Service (PaaS)

- provides the runtime environment for applications, development and deployment tools, etc.
- Does not manage or control the underlying cloud infrastructure including network, servers, operating systems, or storage.

Infrastructure-as-a-Service (IaaS)

- provides access to fundamental resources such as physical machines, web server, Internet access and bandwidth provisioning, virtual machines, virtual storage, etc.

Cloud activities

- Service management and provisioning including:
 - Virtualization.
 - Service provisioning.
 - Call center.
 - Operations management.
 - Systems management.
 - QoS management.
 - Billing and accounting, asset management.
 - SLA management.
 - Technical support and backups.

Cloud activities

- Security management including:
 - ID and authentication.
 - Certification and accreditation.
 - Intrusion prevention.
 - Intrusion detection.
 - Virus protection.
 - Cryptography.
 - Physical security, incident response.
 - Access control, audit and trails, and firewalls.

Cloud activities

- Customer services such as:
 - Customer assistance and on-line help.
 - Subscriptions.
 - Business intelligence.
 - Reporting.
 - Customer preferences.
 - Personalization.
- Integration services including:
 - Data management.
 - Development.

Ethical issues

- Paradigm shift with implications on computing ethics:
 - The control is relinquished to third party services.
 - The data is stored on multiple sites administered by several organizations.
 - Multiple services interoperate across the network.
- Implications
 - Unauthorized access.
 - Data corruption.
 - Infrastructure failure, and service unavailability.

Cloud vulnerabilities

- Clouds are affected by malicious attacks and failures of the infrastructure, e.g., power failures. Such events can affect the Internet domain name servers and prevent access to a cloud or can directly affect the clouds:
 - in 2004 an attack at Akamai (Cloud Service) caused a domain name outage and a major blackout that affected Google, Yahoo, and other sites.
 - in 2009, Google was the target of a denial of service attack which took down Google News and Gmail for several days;
 - in 2012 lightning caused a prolonged down time at Amazon.

The path to cloud computing

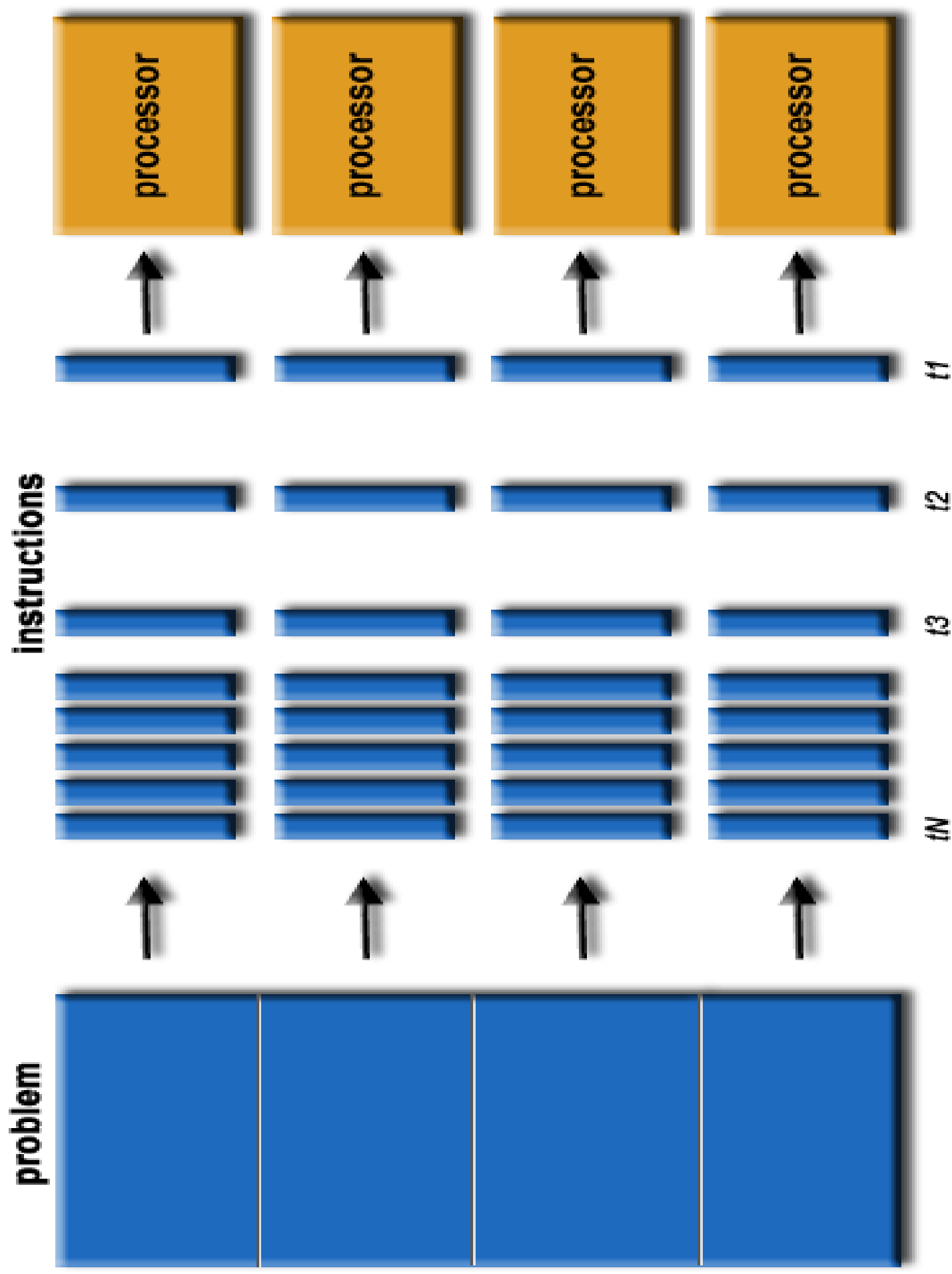
- Cloud computing is based on ideas and the experience accumulated in many years of research in parallel and distributed systems.
 - Cloud applications are based on the client-server paradigm with a relatively simple software, a thin-client, running on the user's machine, while the computations are carried out on the cloud.
 - Concurrency is important; many cloud applications are data-intensive and use a number of instances which run concurrently.
 - Communication is the heart of Cloud computing.

Parallel Computing

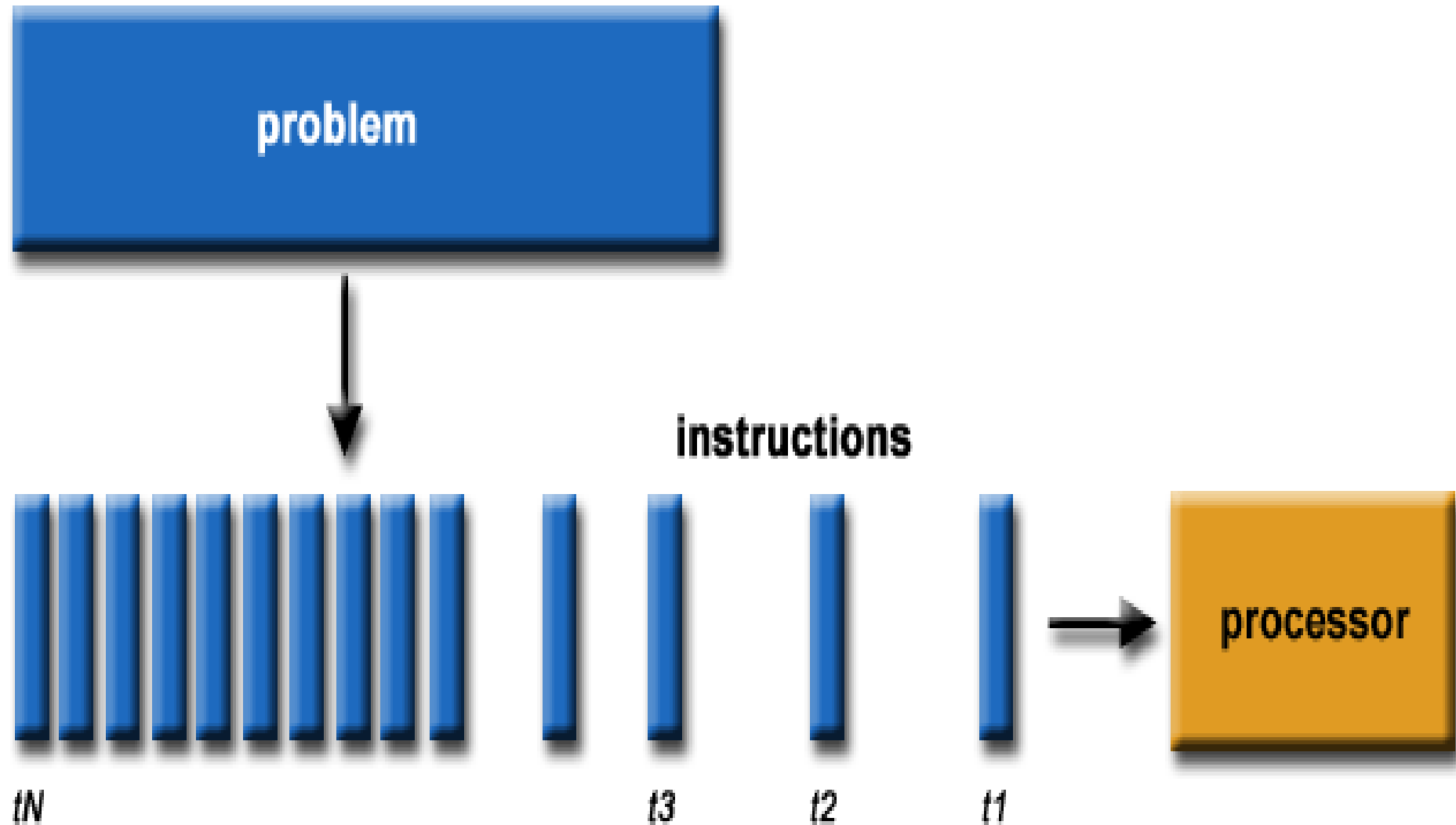
- **Parallel computing** is a type of computation in which many calculations or the execution of processes are carried out simultaneously.
- Large problems are divided into smaller ones, which can then be solved at the same time.
- Parallel computing is a type of computing architecture in which several processors execute or process an application or computation simultaneously.

Parallel Computing

- ***Parallel computing*** is the simultaneous use of multiple compute resources to solve a computational problem:
- A problem is broken into discrete parts that can be solved concurrently
- Each part is further broken down to a series of instructions
- Instructions from each part execute simultaneously on different processors
- An overall control/coordination mechanism is employed



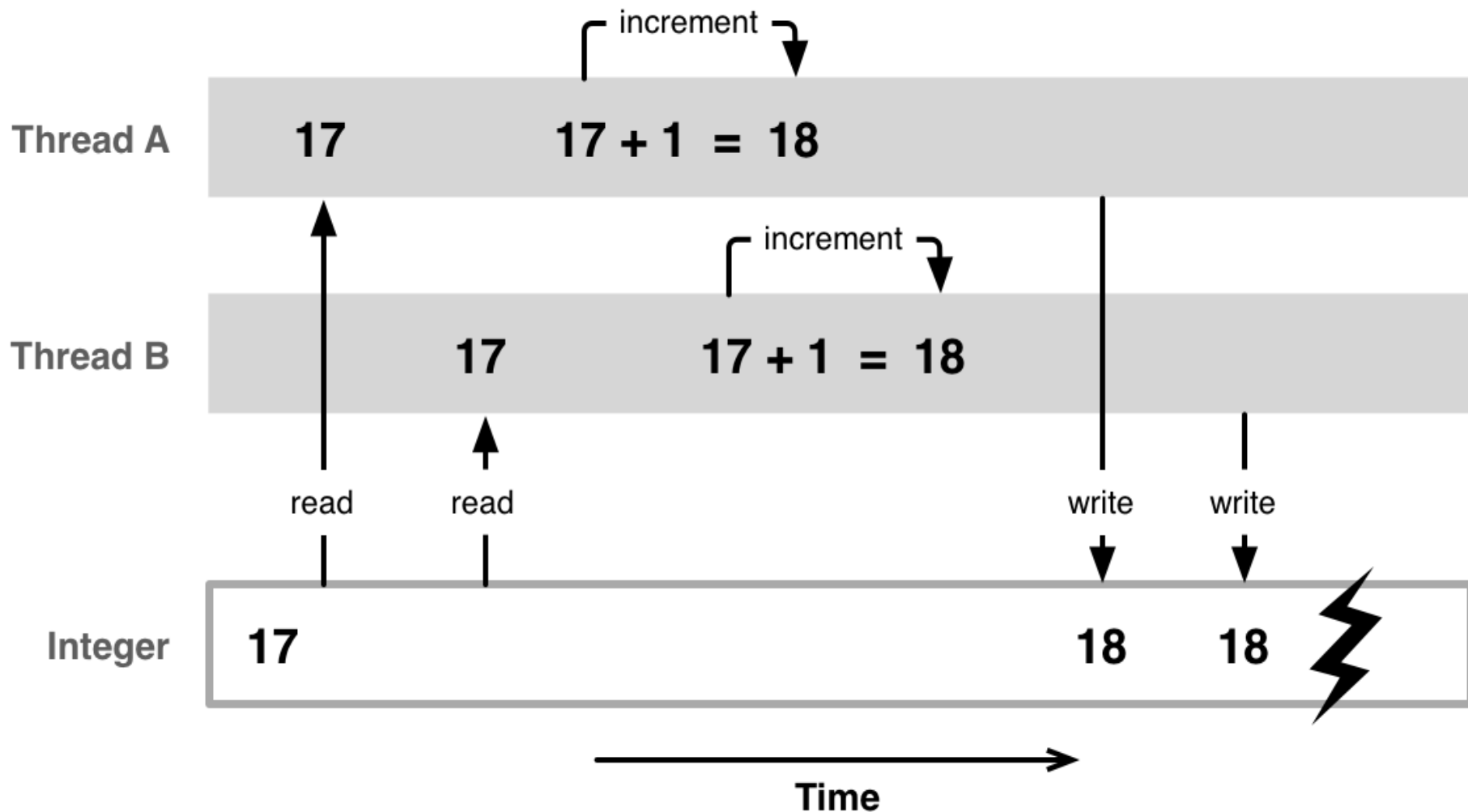
Serial Computing



Concurrency

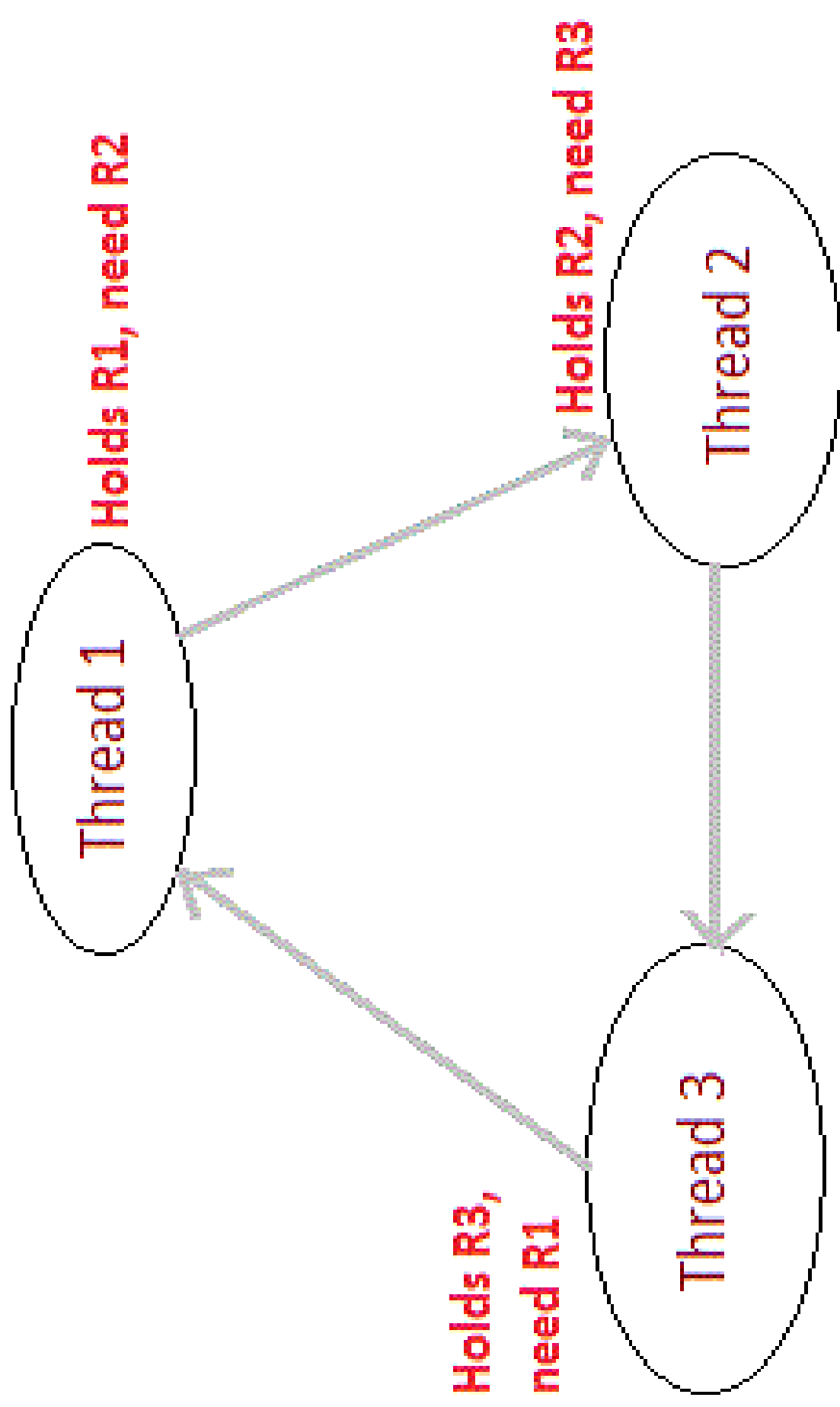
- Concurrent execution can be challenging.
 - It could lead to race conditions,
 - Shared resources must be protected by locks/semaphores /monitors to ensure serial access.
 - Deadlocks and livelocks are possible.

Race Condition



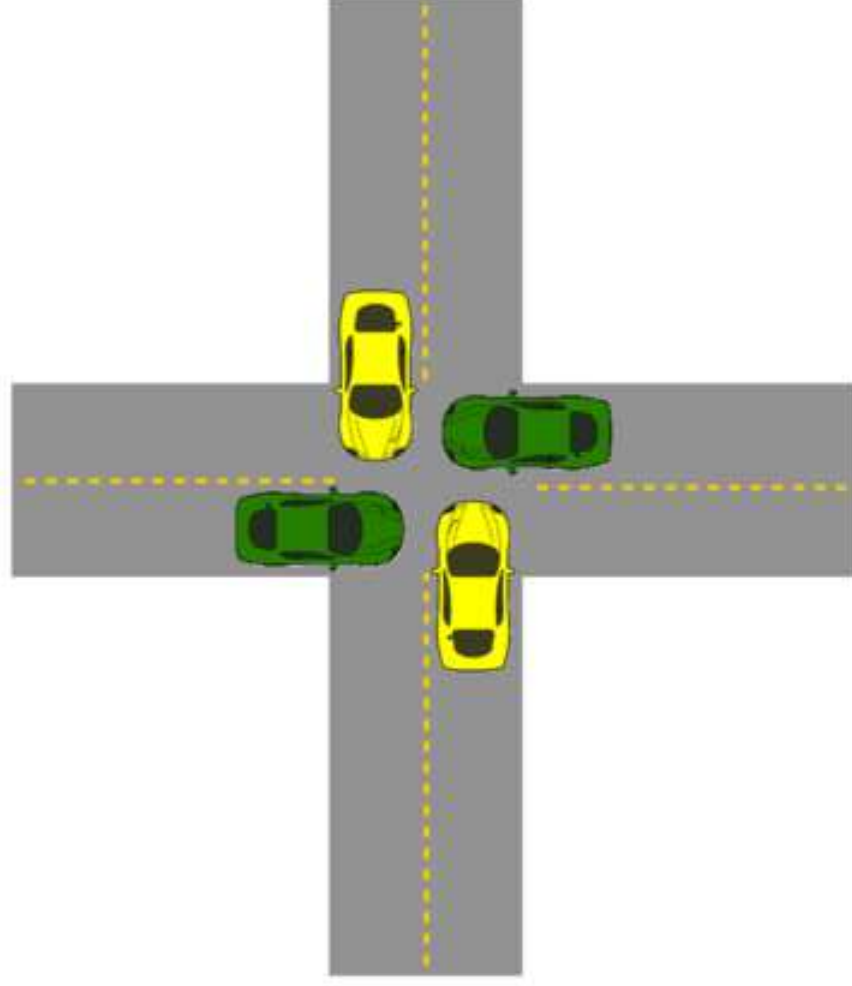
Four Coffman conditions for a deadlock:

- Mutual exclusion – Only one process can use resource at a time.
- Hold and wait – A process holding resources can request more resources.
- No-preemption - The scheduler or a monitor can not force a process/thread holding a resource to relinquish / release it.
- Circular wait - Given the set of n processes/threads $\{P_1, P_2, P_3, \dots, P_n\}$. Process P_1 waits for a resource held by P_2 , P_2 waits for a resource held by P_3 , and so on, P_n waits for a resource held by P_1 .



Deadlock Condition

Livelock



Livelock is a state where a system is executing many operations, but no thread is making meaningful progress.

Can you think of a good daily life example of livelock?

Computer system examples:

Operations continually abort and retry

Livelock

- Two or more processes/threads continually change their state in response to changes in the other processes; then none of the processes can complete its execution.

More Challenges

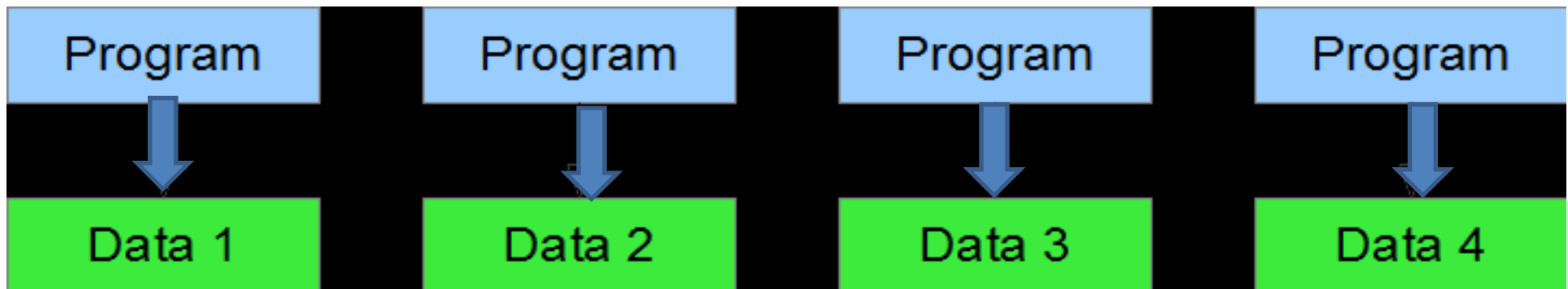
- Discovering parallelism is often challenging and the development of parallel algorithms requires a considerable effort.

Parallelism

- Fine-grain parallelism → relatively small blocks of the code can be executed in parallel without the need to communicate or synchronize with other threads or processes.
- Coarse-grain parallelism → large blocks of code can be executed in parallel.
- The speed-up of applications displaying fine-grain parallelism is considerably lower than those of coarse-grained applications;

Parallelism

- Data parallelism → the data is partitioned into several blocks and the blocks are processed in parallel.
- Same Program Multiple Data (SPMD) → data parallelism when multiple copies of the same program run concurrently, each one on a different data block.



Parallelism levels

- Bit level parallelism. The number of bits processed per clock cycle, often called a word size, has increased gradually from 4-bit, to 8-bit, 16-bit, 32-bit, and to 64-bit.
- Instruction-level parallelism.
- Data parallelism or loop parallelism. The program loops can be processed in parallel.
- Task parallelism. The problem can be decomposed into tasks that can be carried out concurrently.

Parallel computer architecture

Michael Flynn's classification

S I S D Non-parallel Single Instruction stream Single Data stream	S I M D All PUs Single Instruction stream Multiple Data stream
M I S D All PUs Multiple Instruction stream Single Data stream	M I M D All PUs Multiple Instruction stream Multiple Data stream

NOTE

- SISD and SIMD requires single core/processor.
- MISD and MIMD requires several cores/processors.

Distributed System

- A distributed system is a model/software system in which components located on networked computers communicate and coordinate their actions by passing messages.
- The components interact with each other in order to achieve a common goal
- *Distributed computing* also refers to the use of distributed systems to solve computational problems. In *distributed computing*, a problem is divided into many tasks, each of which is solved by one or more computers.

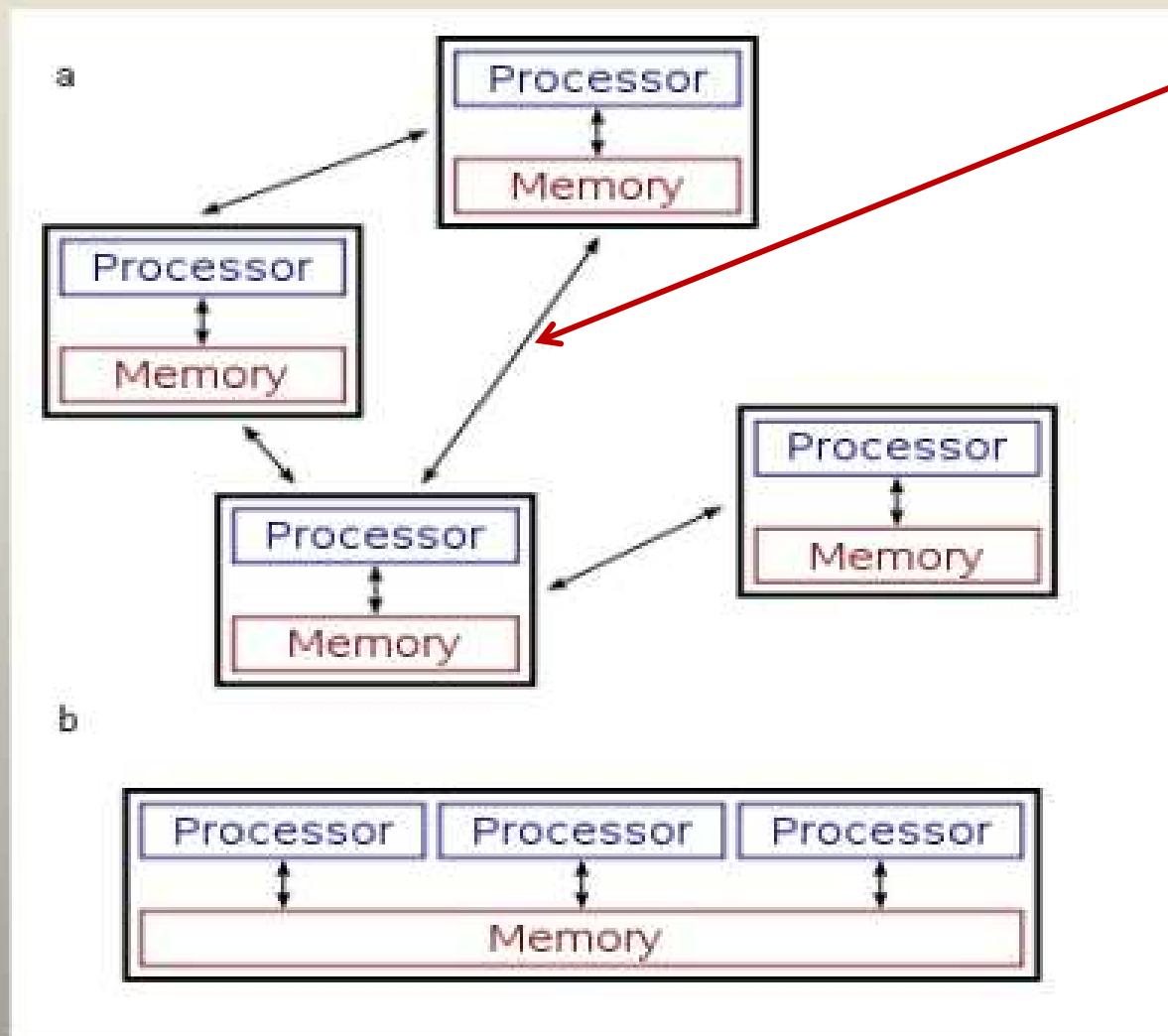
Distributed System

- It is a collection of autonomous computers, connected high speed communication network, having an distribution software called middleware that enables computers to coordinate their activities and to share system resources such a way that the entire n/w appears to its users as a virtual uniprocessor system or single virtual super computer.
- Communication : Message passing.

Characteristics

- The users perceive the system as a single, integrated computing facility.
- The components are autonomous.
- Scheduling and other resource management and security policies are implemented by each system.
- There are multiple points of control and multiple points of failure.
- Can be scaled by adding additional resources.
- Can be designed to maintain availability even at low levels of hardware/software/network reliability.

Parallel and Distributed Computing



Message Passing

(a) Distributed System

(b) Parallel System

Desirable properties of a distributed system

- Access transparency - local and remote information objects are accessed using identical operations.
- Location transparency - information objects are accessed without knowledge of their location.
- Concurrency transparency - several processes run concurrently using shared resources without interference between them.
- Replication transparency - multiple instances of information objects increase reliability without the knowledge of users or applications.

Desirable properties of a distributed system

- Failure transparency - the concealment of faults.
- Migration transparency - the information objects in the system are moved without affecting the operation performed on them.
- Performance transparency - the system can be reconfigured based on the load and quality of service requirements.
- Scaling transparency - the system and the applications can scale without a change in the system structure and without affecting the applications.

Processes, threads, events

- Dispatchable units of work:
 - Process → a program in execution.
 - Thread → a light-weight process.
- State of a process/thread → the information we need to restart a process/thread after it was suspended.
- Event → is a change of state of a process.
 - Local events.
 - Communication events.

Processes, threads, events

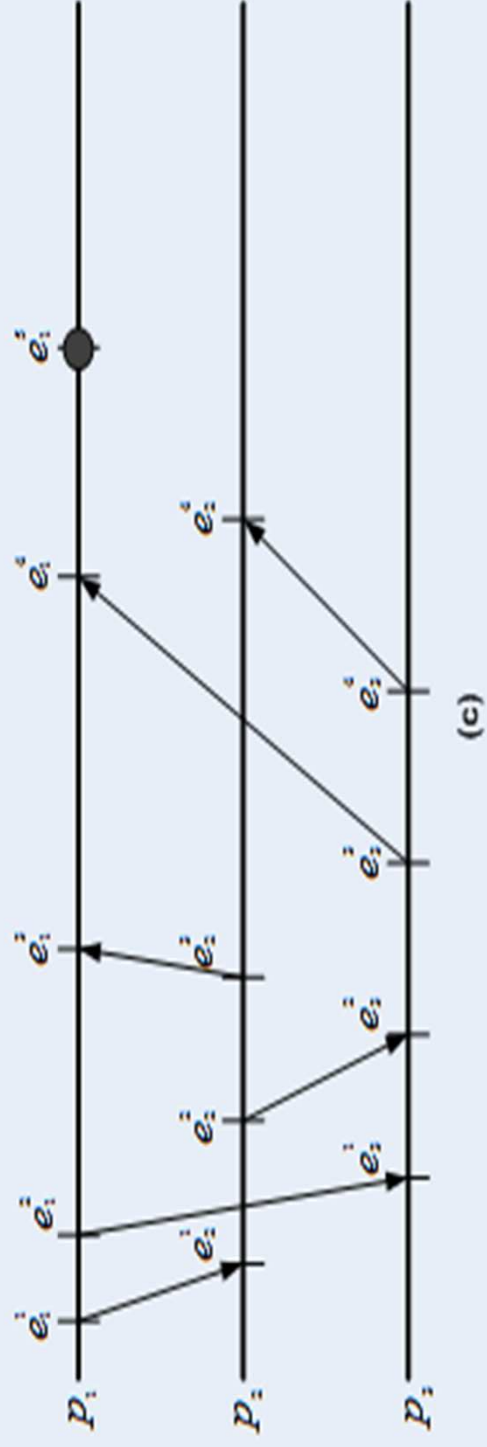
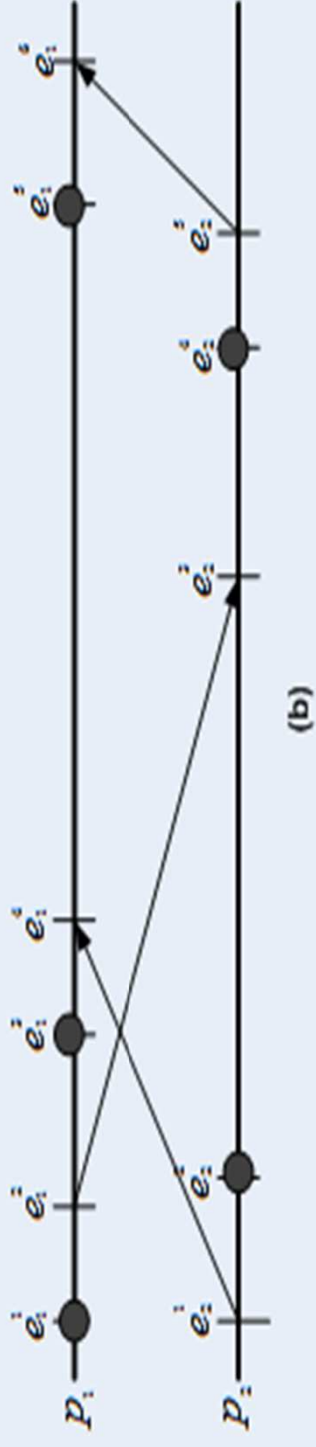
- Process group → a collection of cooperating processes; the processes work in concert and communicate with one another in order to reach a common goal.
- The **global state** of a **distributed system** is the set of local **states** of each individual processes involved in the **system** plus the **state** of the communication channels (Sent/received messages)

Messages and communication channels

- Message → a structured unit of information.
- Communication channel → provides the means for processes or threads to communicate with one another and coordinate their actions by exchanging messages. Communication is done only by means of *send(m)* and *receive(m)* communication events, where *m* is a message.

Space-time diagrams

- Display local and communication events during a process lifetime.
- Local events are small black circles. Communication events in different processes are connected by lines from the send event and to the receive event.



Communication protocols & coordination

- Communication in the presence of channel failures, a major concern. It is impossible to guarantee that two processes will reach an agreement in case of channel failures .
- In practice, error detection and error correction codes allow processes to communicate reliably though noisy digital channels.

Communication protocols & coordination

- Communication protocols implement:
 - Error control mechanisms – using error detection and error correction codes.
 - Flow control - provides feedback from the receiver, it forces the sender to transmit only the amount of data the receiver can handle.
 - Congestion control - ensures that the offered load of the network does not exceed the network capacity.

Time and time intervals

- Process coordination requires:
 - A global concept of time shared by cooperating entities.
 - The measurement of time intervals, the time elapsed between two events.
- Two events in the global history may be unrelated, neither one is the cause of the other; such events are said to be concurrent events.

Time and time intervals

- An isolated system can be characterized by its history expressed as a sequence of events, each event corresponding to a change of the state of the system.
- Global agreement on time is necessary to trigger actions that should occur concurrently.
- Timestamps are often used for event ordering.

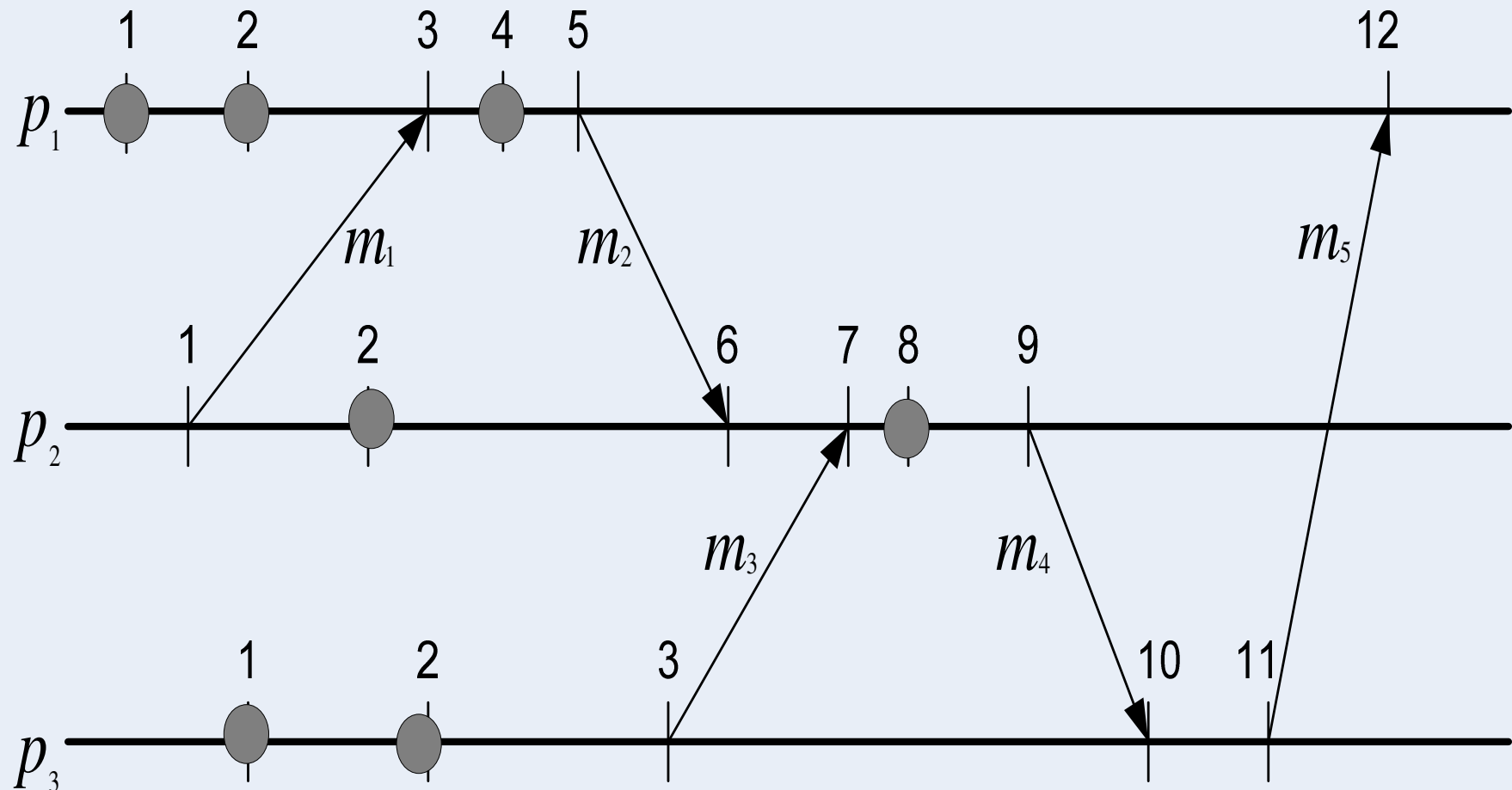
Logical clocks

- **logical clock (LC)** allows global ordering on events from different processes in a distributed systems.
- Logical clock is a monotonically increasing software counter.
- Each process time-stamps each message m it sends with the value of the logical clock at the time of sending:
- $TS(m) = LC(send(m))$.

Logical clocks

- LC1: L_i is incremented before each event is issued at a process P_i : $L_i + 1$;
- LC2:
 - When a process P_i sends a message m , it piggybacks on m the value $t = L_i$.
 - On receiving (m, t) , a process P_j computes $L_j = \max(L_j, t)$ and then applies LC1 before timestamping the event $\text{receive}(m)$

Three processes and their logical clocks

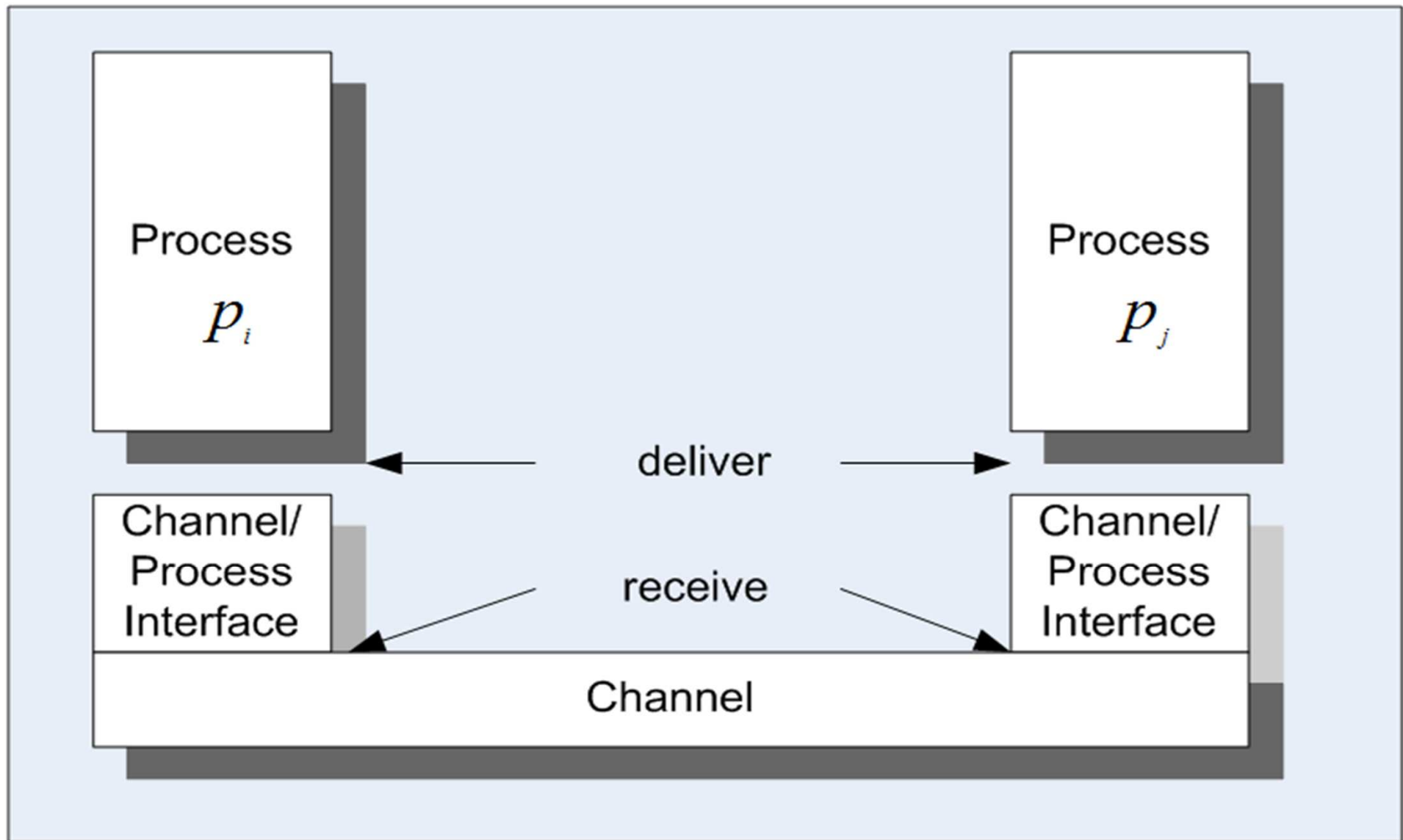


Message delivery rules; causal delivery

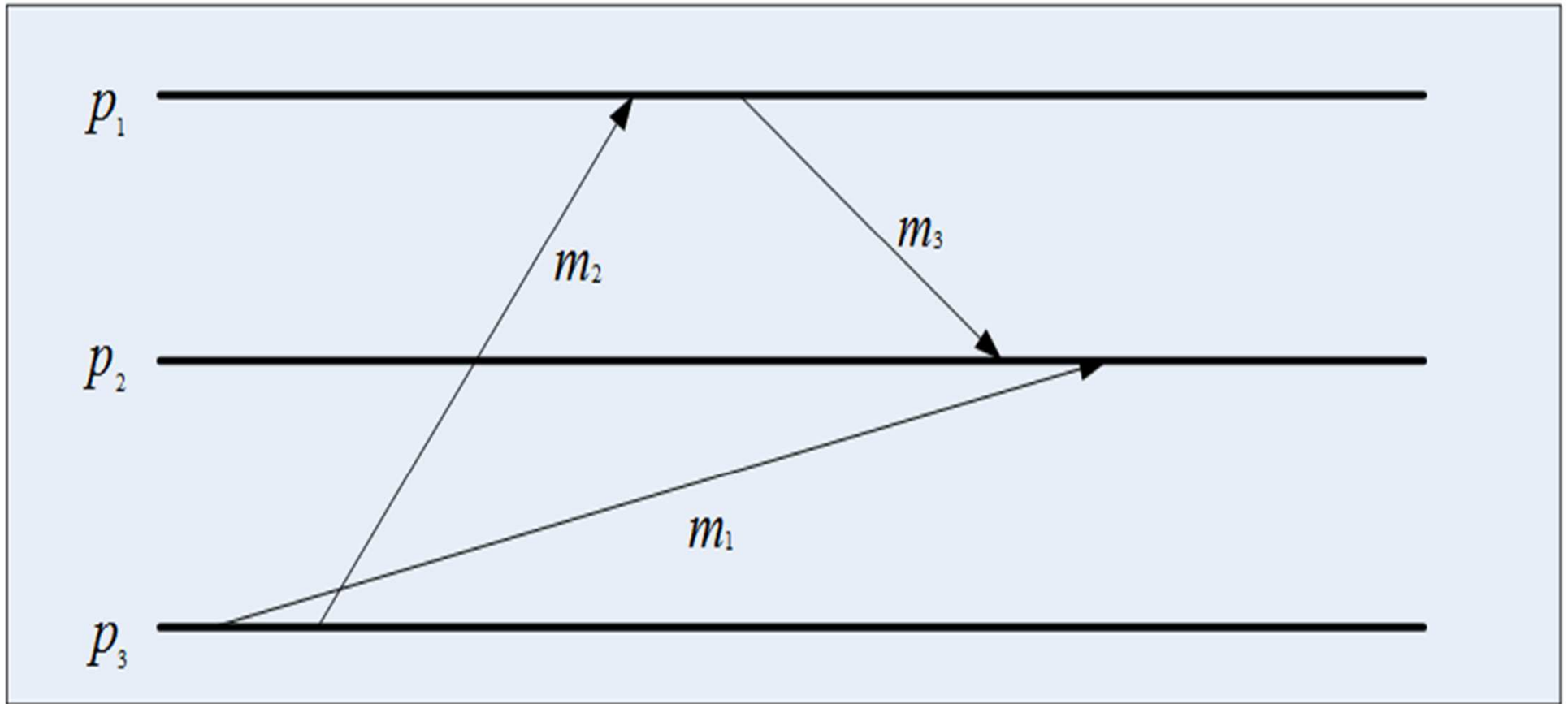
- The communication channel abstraction makes no assumptions about the order of messages; a real-life network might reorder messages.
- First-In-First-Out (FIFO) delivery → messages are delivered in the same order they are sent.
- Causal delivery → an extension of the FIFO delivery to the case when a process receives messages from different sources.

Message delivery rules; causal delivery

- Even if the communication channel does not guarantee FIFO delivery, FIFO delivery can be enforced by attaching a sequence number to each message sent.
- The sequence numbers are also used to reassemble messages out of individual packets.



Message receiving and message delivery are two distinct operations. The channel-process interface implements the delivery rules, e.g., FIFO delivery.



Violation of causal delivery when more than two processes are involved;

Concurrency

- It means, several activities are executed simultaneously.
- It reduce the execution time for data-intensive problems.
- Improve performance - parallel applications partition the workload and distribute it to multiple threads running concurrently.
- Support a variable load and shorten the response time.
- Concurrency is a critical element of the design of system software.

Concurrency

- The kernel of an operating system exploits concurrency for virtualization of system resources such as the processor and the memory Virtualization.
- Concurrency is exploited by application software to speed up a computation and to allow a number of clients to access a service.
- Parallel applications partition the workload and distribute it to multiple threads running concurrently.
- For example, a Web server spawns a new thread when a new request is received; thus, multiple server threads run concurrently

Concurrency

- Distributed applications, including transaction management systems and applications based on the client-server paradigm, use concurrency extensively to improve the response time.
- For example, a Web server spawns a new thread when a new request is received; thus, multiple server threads run concurrently

Atomic actions

- Parallel and distributed applications must take special precautions for handling shared resources.
- **atomic action** An indivisible sequence of primitive operations that must complete without interruption.
- Atomic operation → a multi-step operation should be allowed to proceed to completion without any interruptions and should not expose the state of the system until the action is completed.

Atomicity

- Either all the operations associated with a program unit are executed to completion, or none are performed.
- Ensuring atomicity in a distributed system requires a *transaction coordinator*, which is responsible for the following:
 - Starting the execution of the transaction.

All-or-nothing atomicity

- Either the entire atomic action is carried out, or the system is left in the same state it was before the atomic action was attempted;
- A transaction is either carried out successfully, or the record targeted by the transaction is returned to its original state.

All-or-nothing atomicity

```
start transaction
```

```
{
```

```
    operation 1
```

```
    ...
```

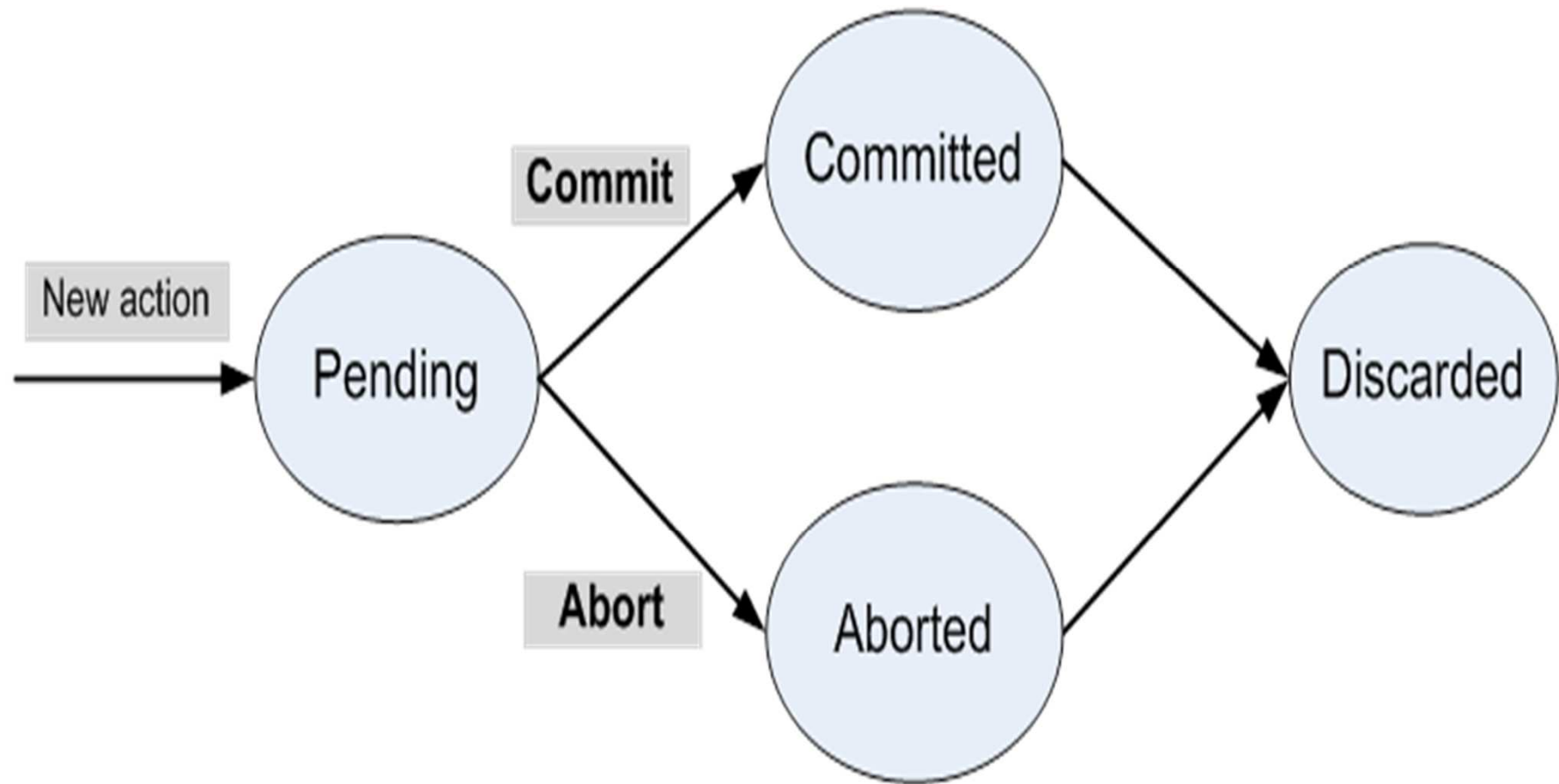
```
    operation n
```

} — all or nothing

```
commit()
```

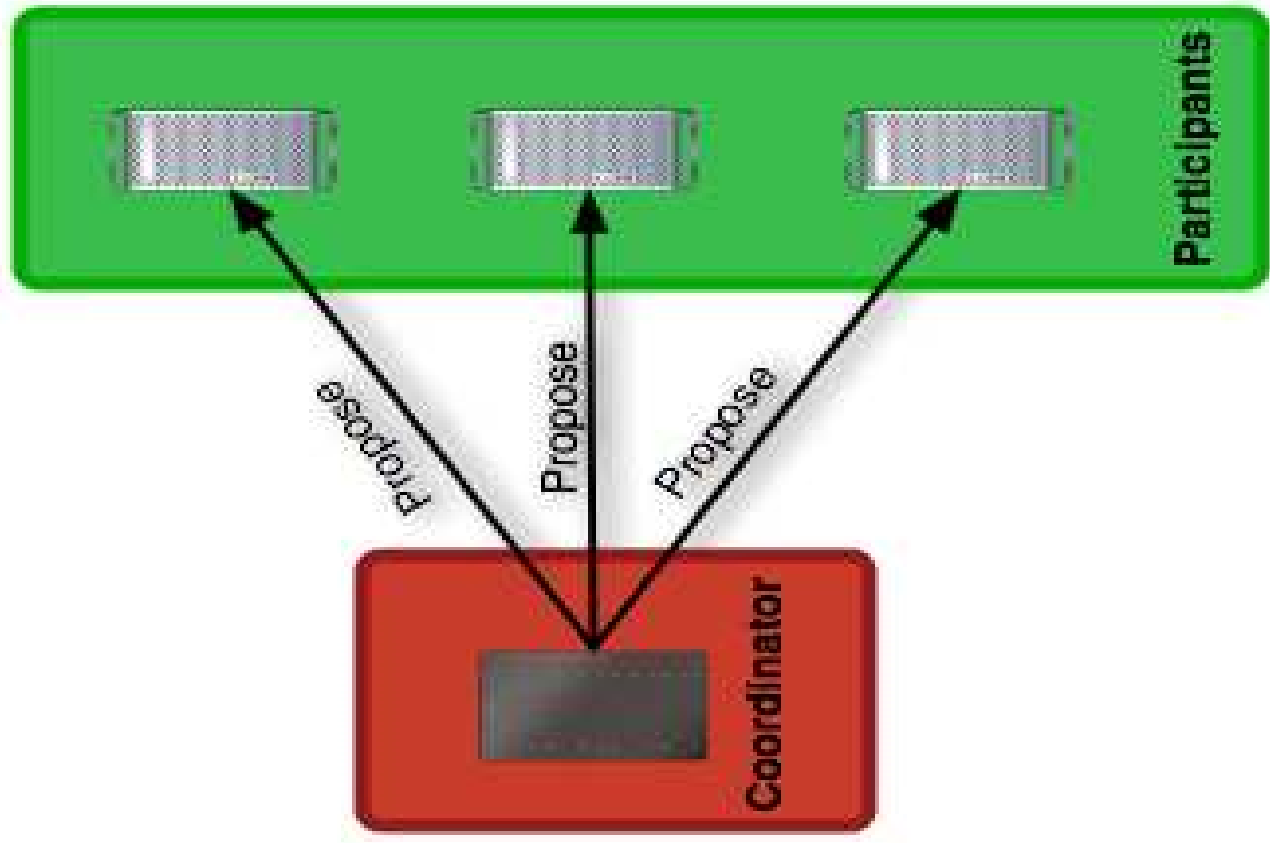
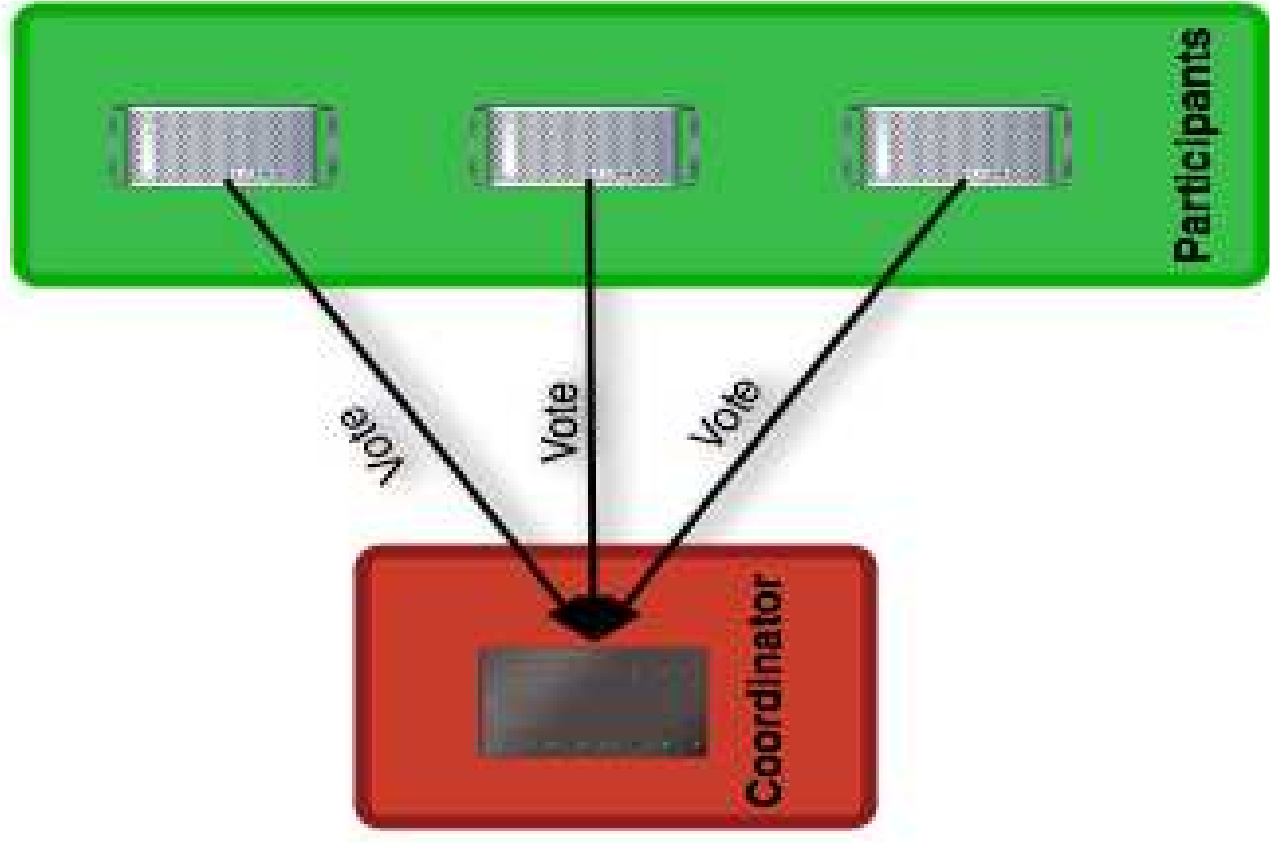
```
}
```

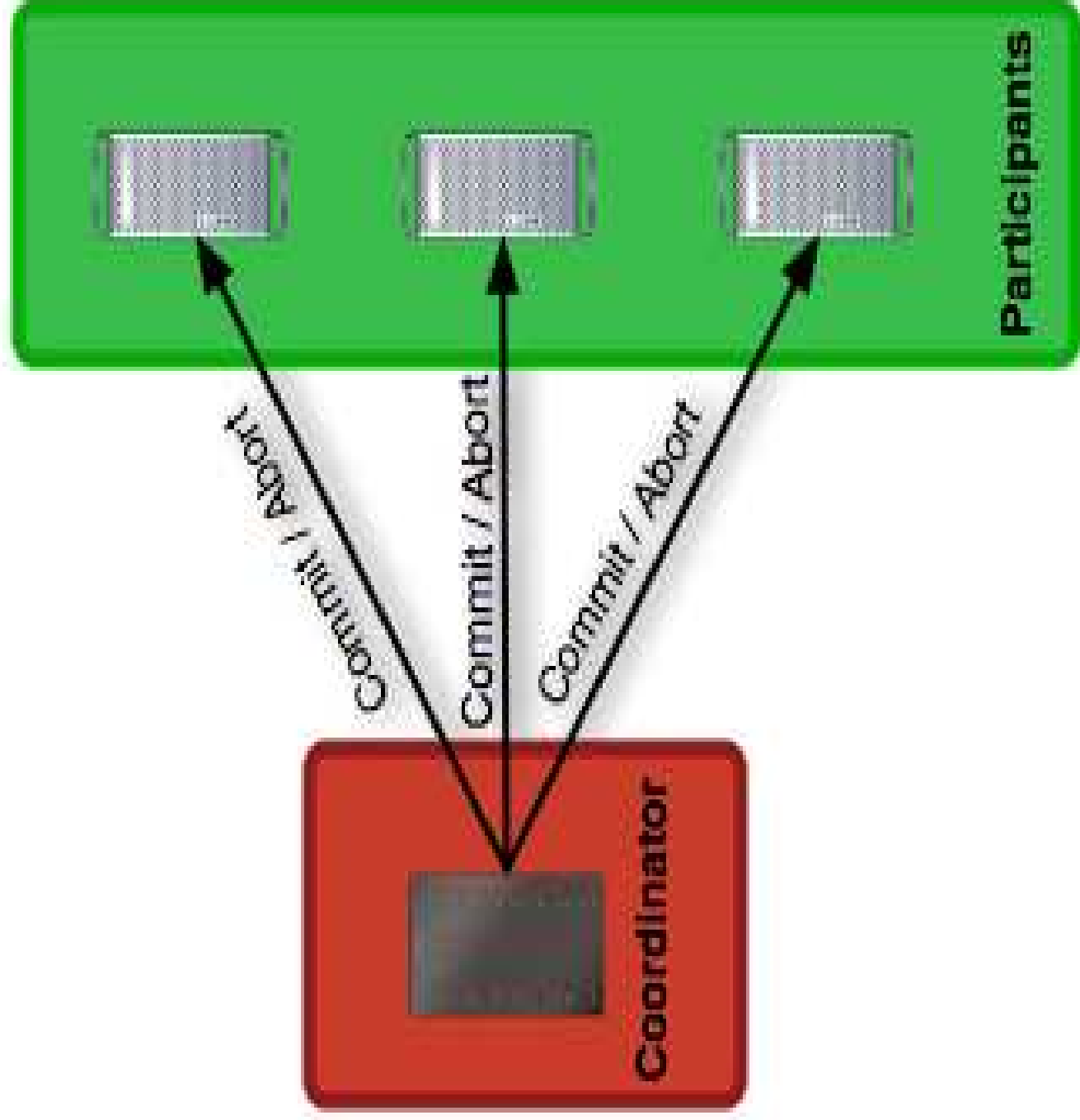
The states of an all-or-nothing action



Consensus protocols

- In a distributed database system, a transaction could be executing its operations at multiple sites. Since atomicity requires every distributed transaction to be atomic, the transaction must have the same fate ([commit](#) or [abort](#)) at every site.
- In case of network partitioning, sites are partitioned and the partitions may not be able to communicate with each other.





Consensus protocols

- Consensus : A general agreement.
- Used for agreeing to one of several alternatives proposed by a number of agents (Processes).
- A fundamental problem in **distributed** computing and multi-agent **systems** is to achieve overall **system** reliability in the presence of a number of faulty processes.
- The consensus requires agreement among a number of processes (or agents) for a single data value

Consensus protocols

- Some of the processes (agents) may fail or be unreliable, so consensus protocols must be fault tolerant.
- The processes must publish their candidate values, communicate with one another, and agree on a single consensus value.

Uses of Consensus protocols

- Deciding whether or not to commit a transaction to a database
- Synchronizing clocks by agreeing on the current time
- Agreeing to move to the next stage of a distributed algorithm (this is the famous *replicated state machine* approach)
- Electing a leader node to coordinate some higher-level protocol.

Consensus protocol requirements

- A consensus protocol correct if and only if:
 - *Agreement* – all nodes in N decide on the same value
 - *Validity* – the value that is decided upon must have been proposed by some node in N
 - *Termination* – all nodes eventually decide

Modularity

- Basic concept in the design of man-made systems.
- **Modularity** is designing a system that is divided into a set of functional units (named modules) that can be composed into a larger application.
- A module represents a set of related concerns.
- It can include a collection of related components, such as features, views, or business logic, and pieces of infrastructure, such as services for logging or authenticating users.

Modularity

- Modules are independent of one another but can communicate with each other in a loosely coupled fashion.
- A complex system/Composite application exhibits modularity i.e. system is made up of components or modules.
- Modules should support being added and removed from the system in a pluggable fashion.
- Hardware and software systems are composed of modules that interact with one another through well-defined interfaces.

Example: Online banking program.

- The user can access a variety of functions, such as transferring money between accounts, paying bills, and updating personal information from a single user interface (UI).
- However, behind the scenes, each of these functions is a discrete module. These modules communicate with each other and with back-end systems such as database servers.
- Application services integrate components within the different modules and handle the communication with the user. The user sees an integrated view that looks like a single application.

Example: Online banking program.

- The user can access a variety of functions, such as transferring money between accounts, paying bills, and updating personal information from a single user interface (UI).
- However, behind the scenes, each of these functions is a discrete module. These modules communicate with each other and with back-end systems such as database servers.
- Application services integrate components within the different modules and handle the communication with the user. The user sees an integrated view that looks like a single application.

Composite Application

Host all the visual components

Shell

Provides the glue between modules and the shell

Application Services

Contain the vertical slices of the application

Module

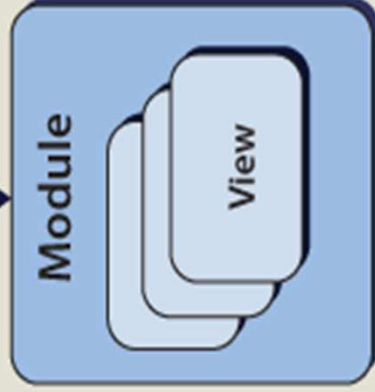
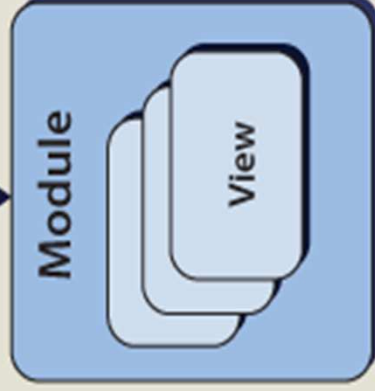
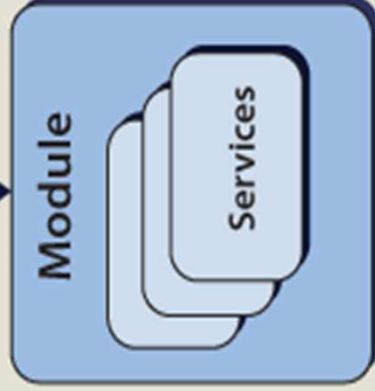
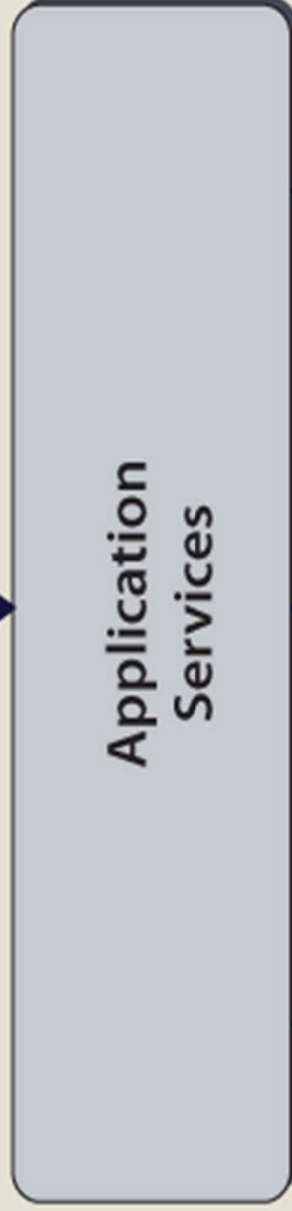
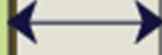
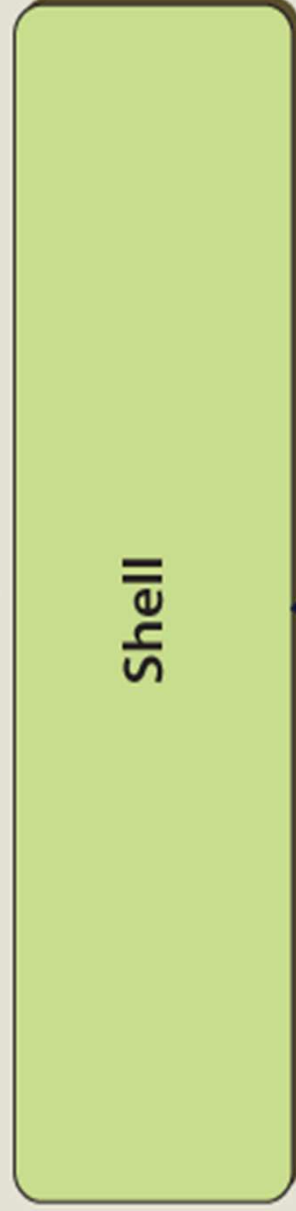
View

Module

View

Module

Services



Modularity supports

- The separation of concerns.
- Specialization.
- Improves maintainability.
- Reduces costs.
- Decrease the development time of a system.

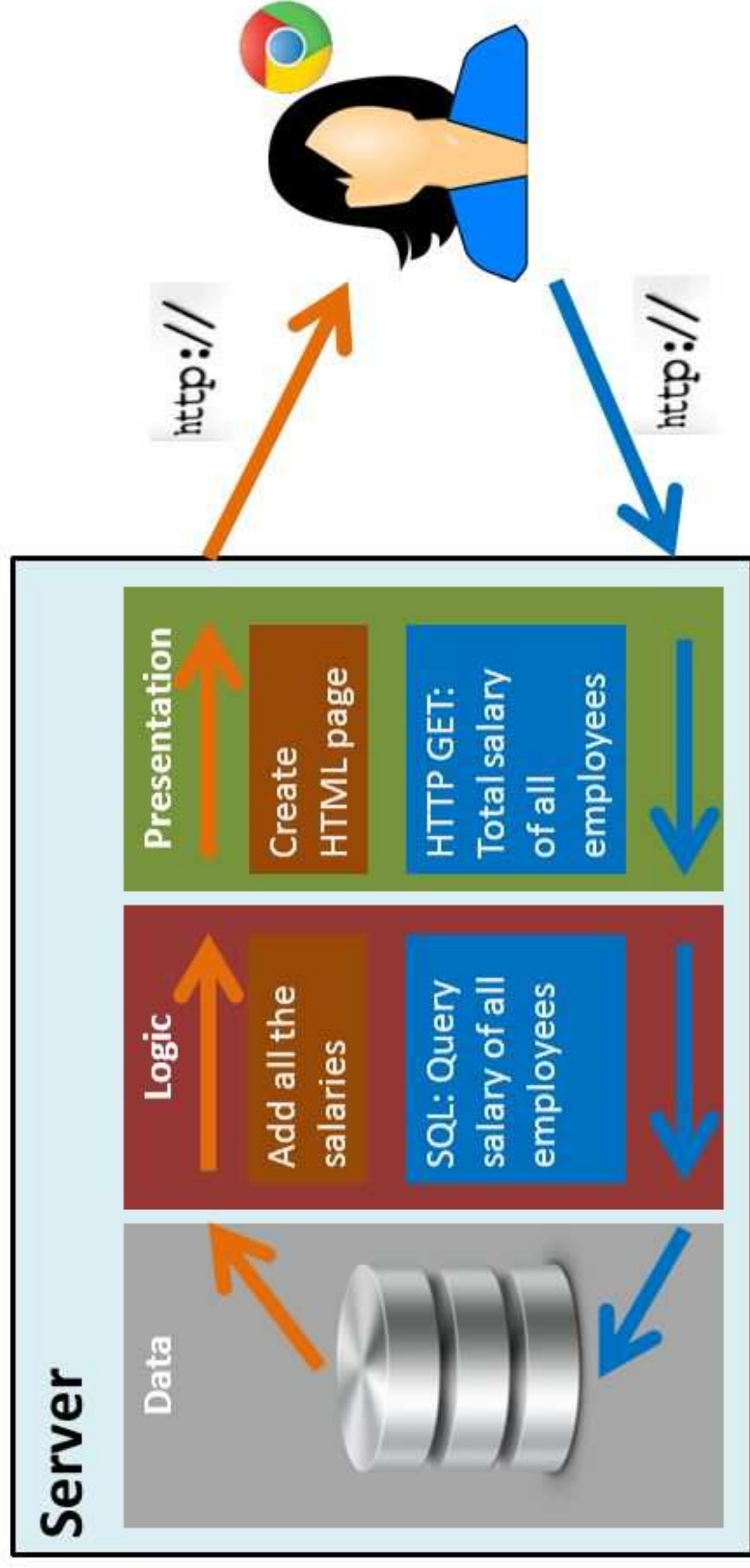
The client-server paradigm

- Based on modularity → the modules are forced to interact only by sending and receiving messages (loosely coupled).
- A more robust design, the clients and the servers are independent modules and may fail separately.
- The servers are stateless, they do not have to maintain state information; the server may fail and then come up without the clients being affected or even noticing the failure of the server.

The client-server paradigm

- An attack is less likely because it is difficult for an intruder to guess the format of the messages or the sequence numbers of the segments, when messages are transported by TCP.

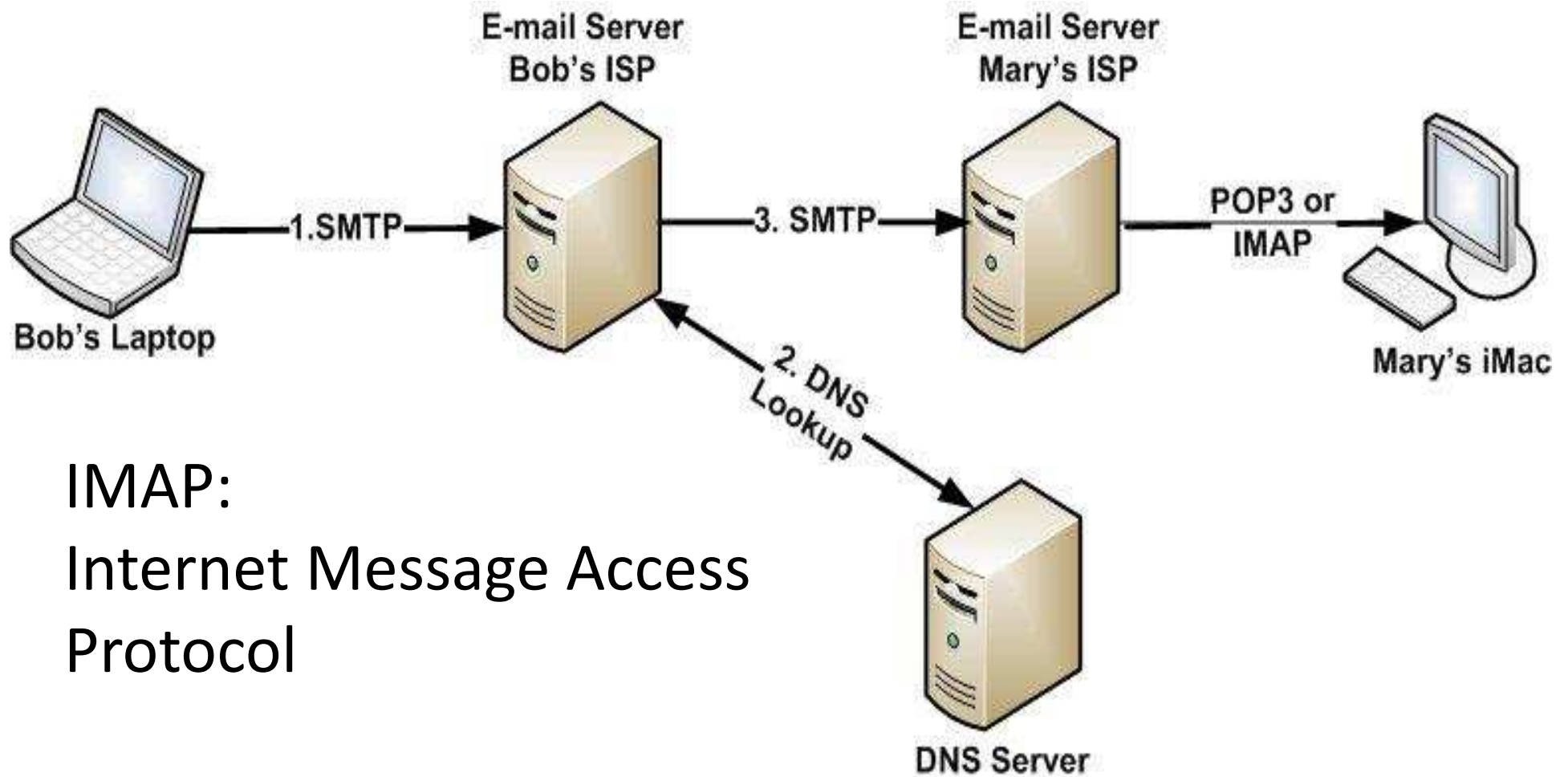
Client-Server: *Three-Tier Server*



Applications of the C-S paradigm

- WWW
- DNS
- E-mail
- Event services and so on.

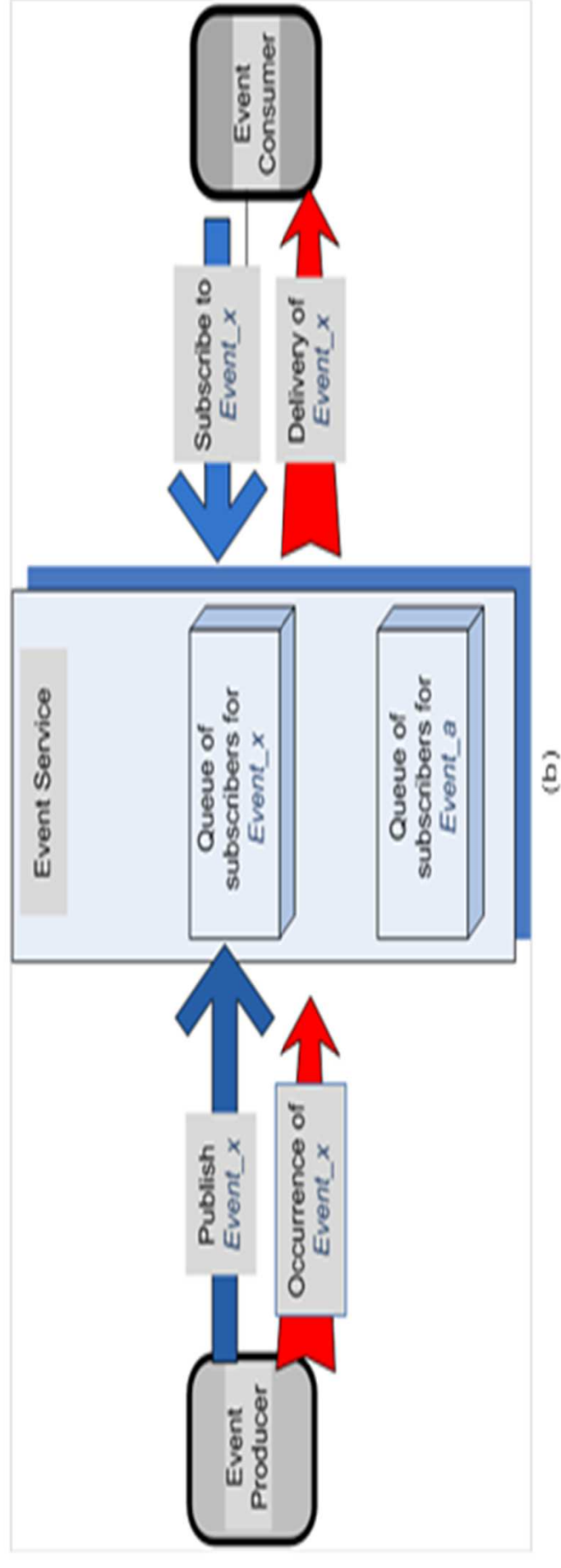
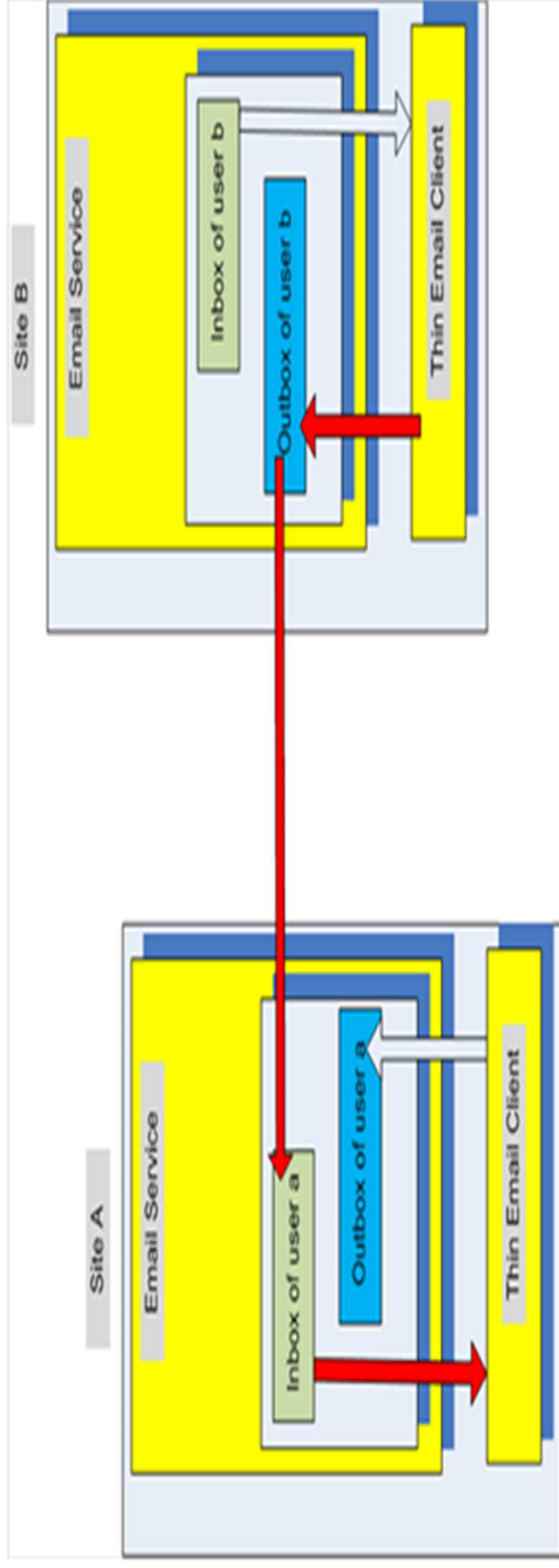
E-mail service



IMAP:
Internet Message Access
Protocol

Event service system

- An event service supports coordination in a distributed system environment.
- The service is based on the publish-subscribe paradigm;
- An event producer publishes events and an event consumer subscribes to events.
- The server maintains queues for each event and delivers notifications to clients when an event occurs.



World Wide Web

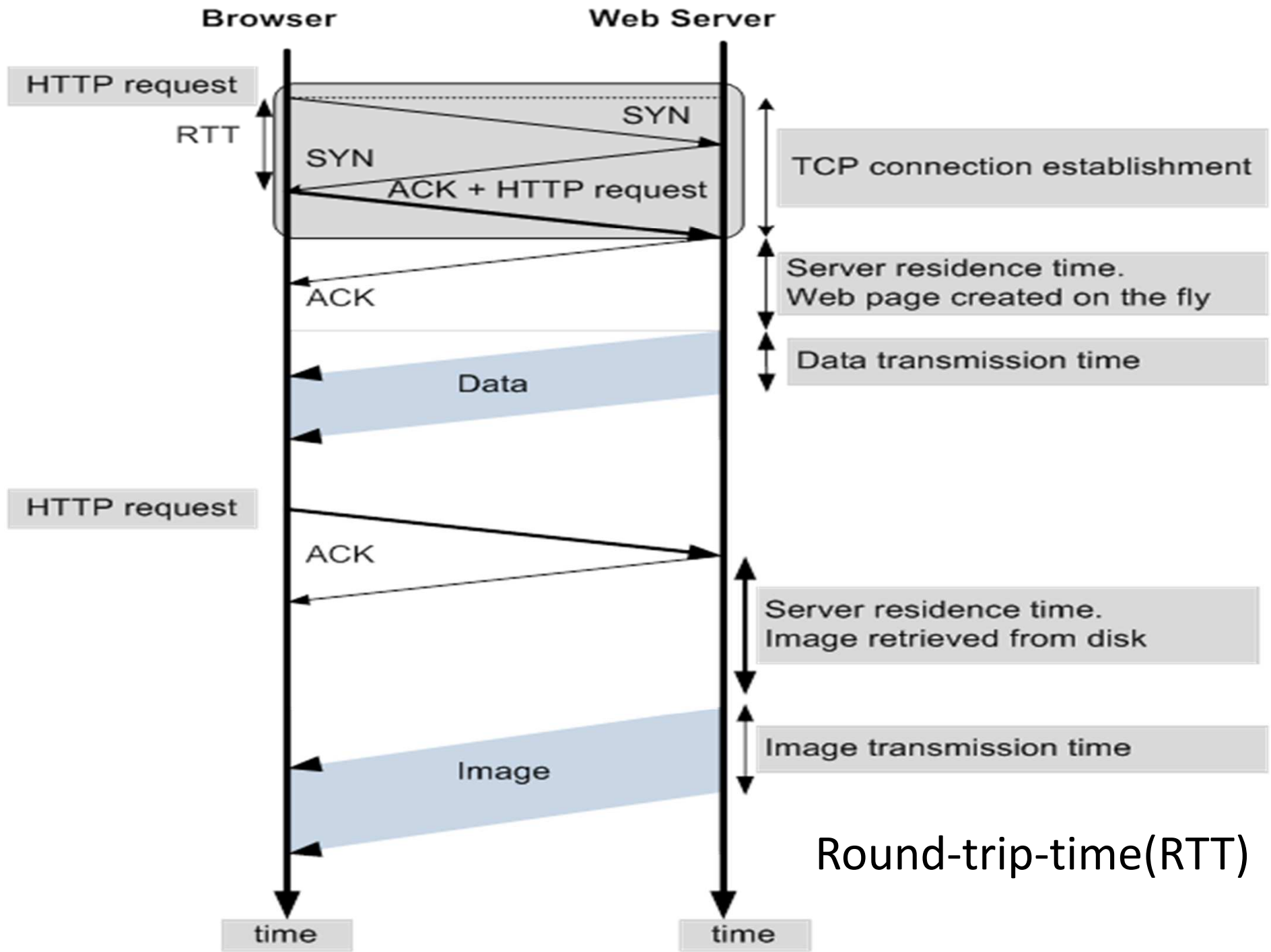
- The Web allows users to access *resources* such as text, images, digital music, video, and any type of information stored in a digital format.
- The information in each Web page is encoded and formatted according to some standard (e.g., GIF, JPEG for images, MPEG for videos, MP3 or MP4 for audio, and so on).

World Wide Web

- The Web is based on a “pull” paradigm; the resources are stored at the server’s site and the client pulls them from the server.
- Some Web pages are created “on the fly”; others are fetched from disk.
- The client, called a *Web browser*, and the server communicate using an application-level protocol Hyper Text Transfer Protocol (HTTP) built on top of the Transport Control Protocol (TCP).

World Wide Web (Image retrieval)

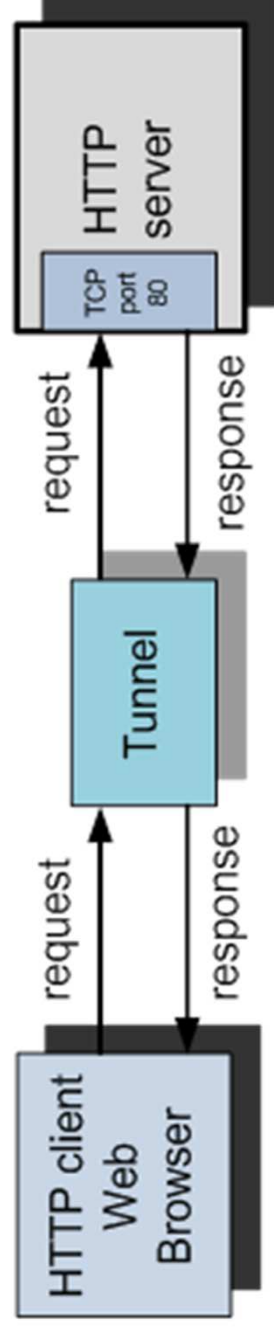
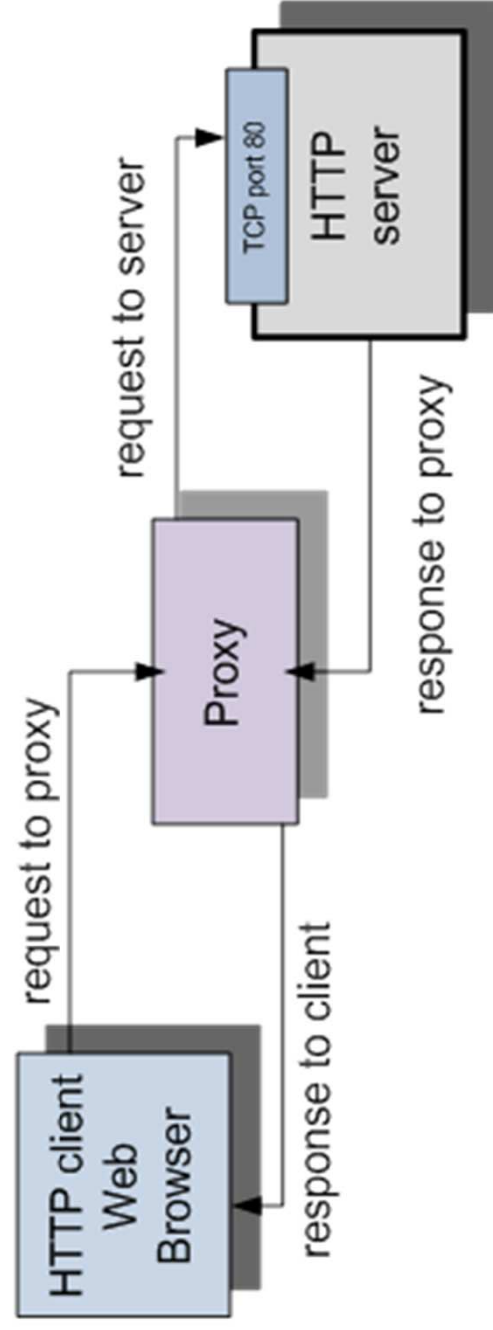
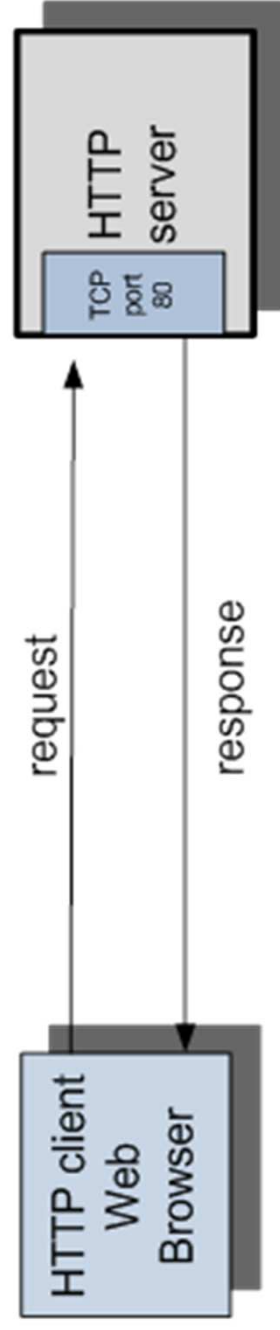
- The three-way handshake involves the three messages exchanged between the client and the server.
- Once the TCP connection is established the HTTP server takes its time to construct the page to respond the first request; to satisfy the second request, the HTTP server must retrieve an image from the disk.
- The response time includes the RTT, the server residence time, and the data transmission time.



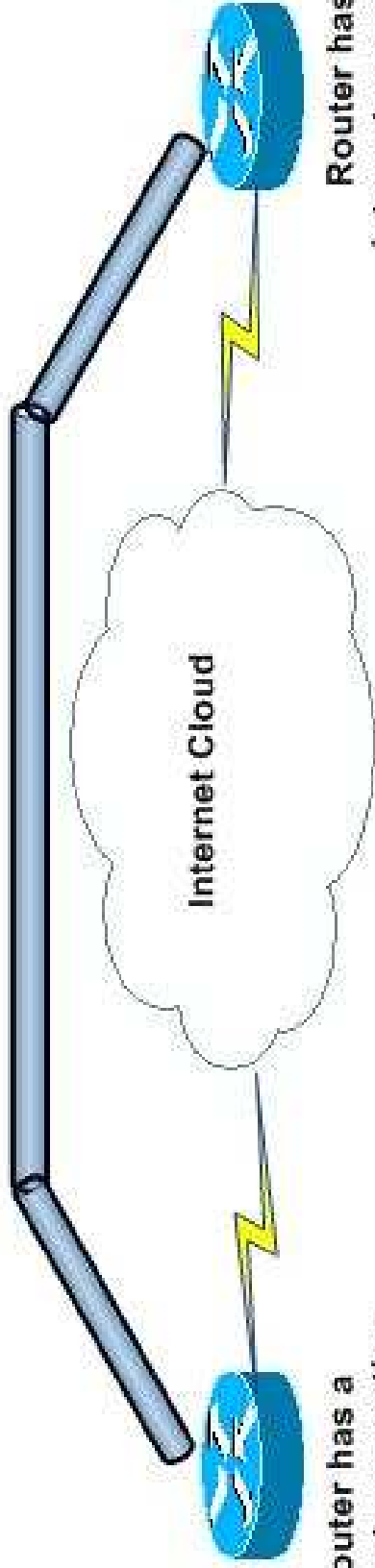
Round-trip-time(RTT)

Types of C-S communication

- A Web client can: (a) communicate directly with the server; (b) communicate through a proxy; (c) use tunneling to cross the network.




Tunnel connecting the
two networks over the
internet cloud



Router has a
internet connection
with ISP ABC.

Router has a
internet connection
with ISP XYZ.



**A bad attitude is like a flat tire:
You can't go anywhere
until you change it.**

www.mwys.net