CSE- 321
Software Engineering

Lecture : 14
Software design

Fahad Ahmed
Lecturer, Dept. of CSE
E-mail: fahadahmed@uap-bd.edu

Broker Architectural Style

Broker Architectural Style

Broker Architectural Style is a middleware architecture used in distributed computing to coordinate and enable the communication between **registered servers and clients**. Here, object communication takes place through a middleware system called an **object request broker(ORB).**

- Client and the server do not interact with each other directly. Client and server have a direct connection to its proxy which communicates with the mediator-broker.

- A server provides services by registering and publishing their interfaces with the broker and clients can request the services from the broker statically or dynamically by look-up.

- **CORBA** (Common Object Request Broker Architecture) is a good implementation example of the broker architecture.

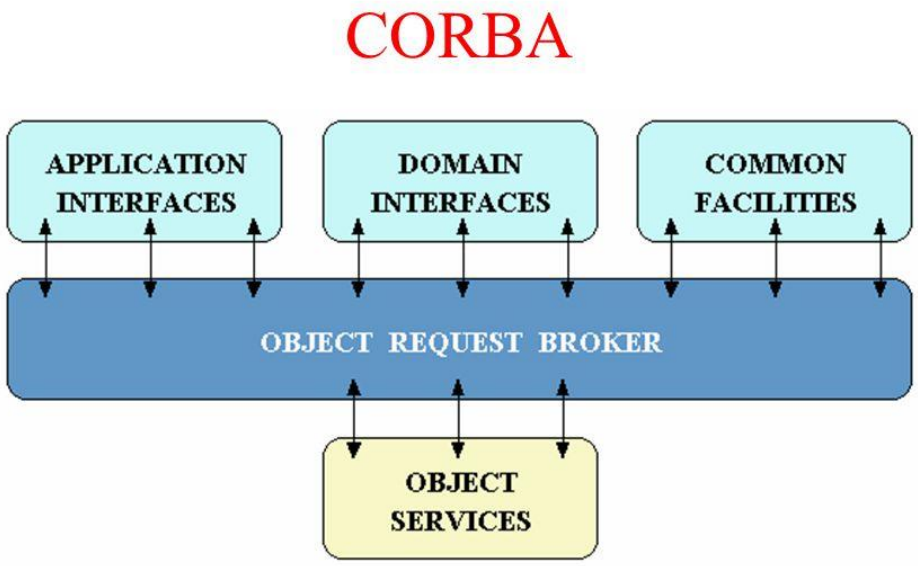
Broker implementation in CORBA

CORBA, or Common Object Request Broker Architecture, is a standard architecture for distributed object systems. It allows a distributed, heterogeneous collection of objects to interoperate.

CORBA is a standard defined by the OMG (Object Management Group). It describes an architecture, interfaces, and protocols that distributed objects can use to interact with each other.

Part of the CORBA standard is the Interface Definition Language (IDL), which is an implementation-independent language for describing the interfaces of remote objects.

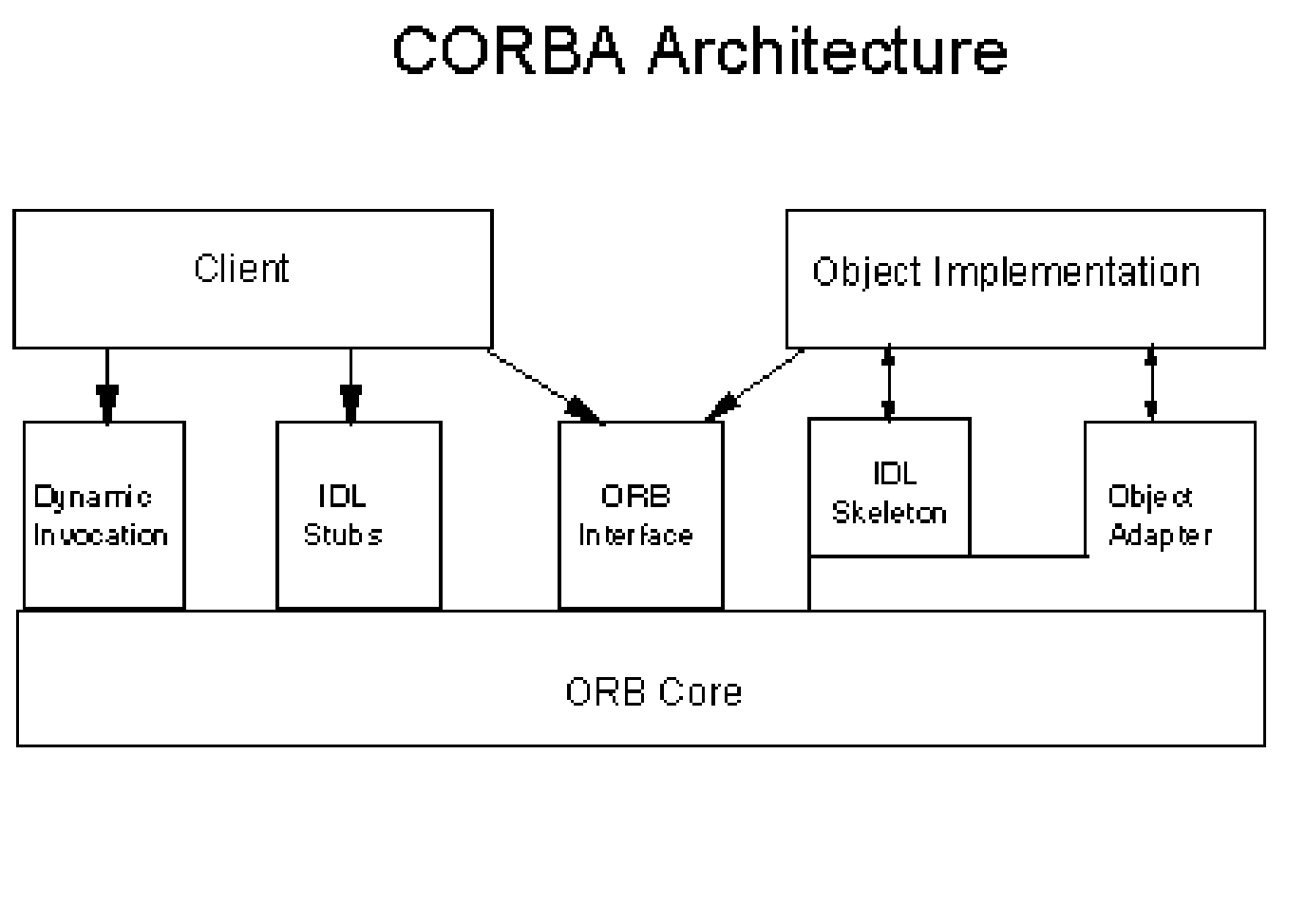
CORBA describes a messaging mechanism by which objects distributed over a network can communicate with each other irrespective of the platform and language used to develop those objects.



OMG Reference Model architecture

Broker implementation in CORBA

The ORB manages the interactions between clients and object implementations (Server).



Broker implementation in CORBA

Client Side Architecture

The client side architecture provides clients with interfaces to the ORB and object implementations. In consists of the following interfaces:

Dynamic Invocation - This interface allows for the specification of requests at runtime. This is necessary when object interface is not known at run-time. Dynamic Invocation works in conjunction with the interface repository.

IDL Stub - This component consists of functions generated by the IDL interface definitions and linked into the program. The functions are a mapping between the client and the ORB implementation. Therefore ORB capabilities can be made available for any client implementation for which there is a language mapping. Functions are called just as if it was a **local object**.

ORB Interface - The ORB interface may be called by either the client or the object implementation. The interface provides functions of the ORB which may be directly accessed by the client (such as retrieving a reference to an object.) or by the object implementations. This interface is mapped to the host programming language. The ORB interface must be supported by any ORB.

ORB core - Underlying mechanism used as the transport level. It provides basic communication of requests to other sub-components.

Broker implementation in CORBA

Implementation Side Architecture

The implementation side interface consists of the ORB Interface, ORB core and IDL Skeleton Interface defined below:

IDL Skeleton Interface - The ORB calls method skeletons to invoke the methods that were requested from clients. Object Adapters (OA) - Provide the means by which object implementations access most ORB services. This includes the generation and interpretation of object references, method invocation, security and activation. The object adapter actually exports three different interfaces: a private interface to skeletons, a private interface to the ORB core and a public interface used by implementations. The OA **isolates the object implementation from the ORB core.**

Requests

The client requests a service from the object implementation. The ORB transports the request, which invokes the method using object adapters and the IDL skeleton.

The client has an object reference, an operation name and a set of parameters for the object and activates operations on this object. The Object Management Group / Object Model defines each operation to be associated with a controlling parameter, implemented in CORBA as an object reference. The client does not know the location of the object or any of the implementation details. The request is handled by the ORB, which must locate the target object and route the request to that object. It is also responsible for getting results back to the client.

Broker implementation in CORBA

Object Adapters

Object Adapters (OA) are the primary ORB service providers to object implementations. OA have a public interface which is used by the object implementation and a private interface that is used by the IDL skeleton.

Example services provided by OA's are:

- Method invocation (in conjunction with skeleton),
- Object implementation activation and deactivation,
- Mapping of object reference to object implementations,
- Generation of object references, and
- Registering object implementations, used in locating object implementations when a request comes in.

ORB Interface Operations

The ORB provides operations on references and strings. Two operations for converting an object reference to strings and back again. An **is_nil** operation for testing whether an object reference is referencing no object. A duplicate operation allows for a duplicate reference to the same object to be created. A release operation is provided to reclaim the memory used by an object reference.

Broker implementation in CORBA

Object Services

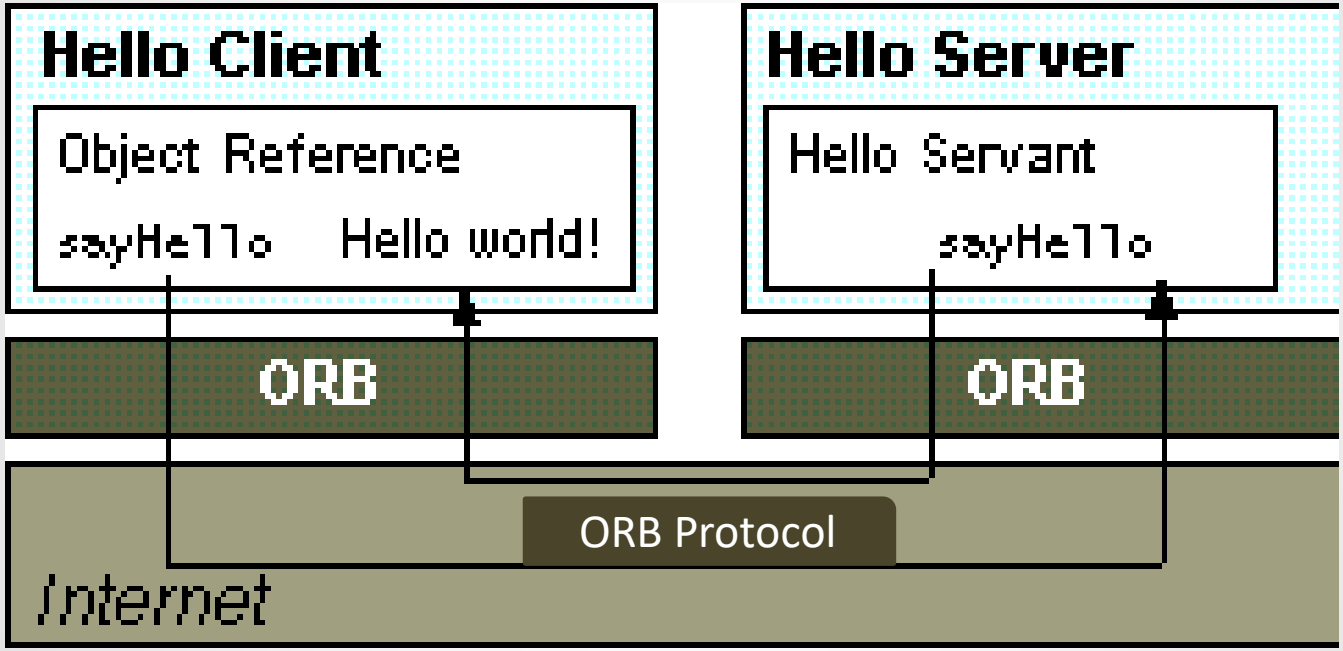
Object services refer to fundamental services provided by their own objects that support common interactions between other objects. The services follow the OMG Common Object Services Specification (COSS) guidelines.

Current object services include:

- **Event Management services** which refer to the asynchronous communication of different CORBA objects with one another.
- **Naming Object services** that maps a human-readable name (string) to an object relative to its context.
- **Persistent services** assure that an object (CORBA developed) outlives its creator. It allows an objects state to be retained at a given point in time. This feature is used when a host crashes or is rebooted.
- **Externalization services** collect objects and transport them as serial objects.
- **Life-cycle services** determine the methods for an object creation and termination.
- **Concurrency services** provide for distributed locks on a given object.
- **Relationship Objects** are used for CORBA object modeling and graphing.
- **Transaction object services** allow the sharing of a transaction among multiple objects.

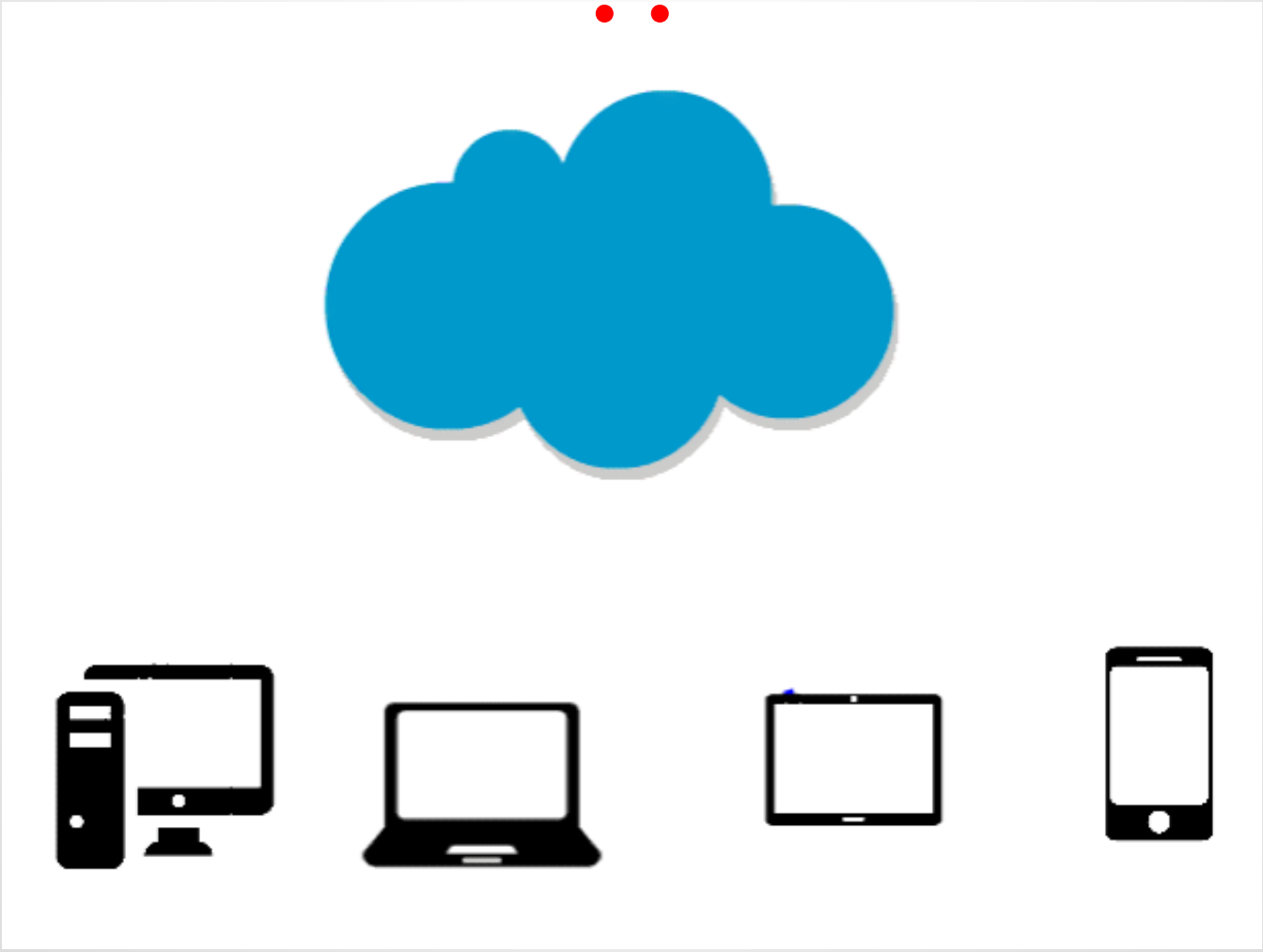
Broker implementation in CORBA

Describe this architecture based on ORB style and mention its all objects services?



Cloud based Software Engineering

Cloud computing ??



Cloud computing

The term **Cloud** refers to a **Network or Internet**. In other words, we can say that Cloud is something, which is present at remote location. Cloud can provide services over public and private networks, WAN, LAN or VPN.

Cloud computing is the on-demand availability of computer system resources, especially data storage and computing power, without direct active management by the user.

Cloud computing (also called simply, the cloud) describes the act of storing, managing and processing data online - as opposed to on your own physical computer or network.

Cloud computing is the delivery of different services through the Internet. These resources include tools and applications like data storage, servers, databases, networking, and software.

Cloud computing is the delivery of **on-demand computing services over the internet** on a **pay-as-you-go basis**.

Cloud computing

Types of Cloud Computing

Public Cloud: Multi-tenant environment with pay-as-you-grow scalability
A public cloud is built over the Internet and can be accessed by any user who has paid for the service. Public clouds are owned by service providers and are accessible through a subscription.

Many public clouds are available, including Google App Engine (GAE), Amazon Web Services (AWS), Microsoft Azure, IBM Blue Cloud, and Salesforce.com's Force.com.

Private Cloud: Scalability plus the enhanced security and control of a single-tenant environment
A private cloud is built within the domain of an intranet owned by a single organization. Therefore, it is client owned and managed, and its access is limited to the owning clients and their partners. Its deployment was not meant to sell capacity over the Internet through publicly accessible interfaces.

Cloud computing: Cost Model

Types of Cloud Computing

Hybrid Cloud: Connect the public cloud to your private cloud or dedicated servers - even in your own data center.

Private clouds can also support a hybrid cloud model by supplementing local infrastructure with computing capacity from an external public cloud.
For example, the Research Compute Cloud (RC2) is a private cloud, built by IBM, that interconnects the computing and IT resources at eight IBM Research Centers scattered throughout the United States, Europe, and Asia. A hybrid cloud provides access to clients, the partner network, and third parties.

Cloud computing

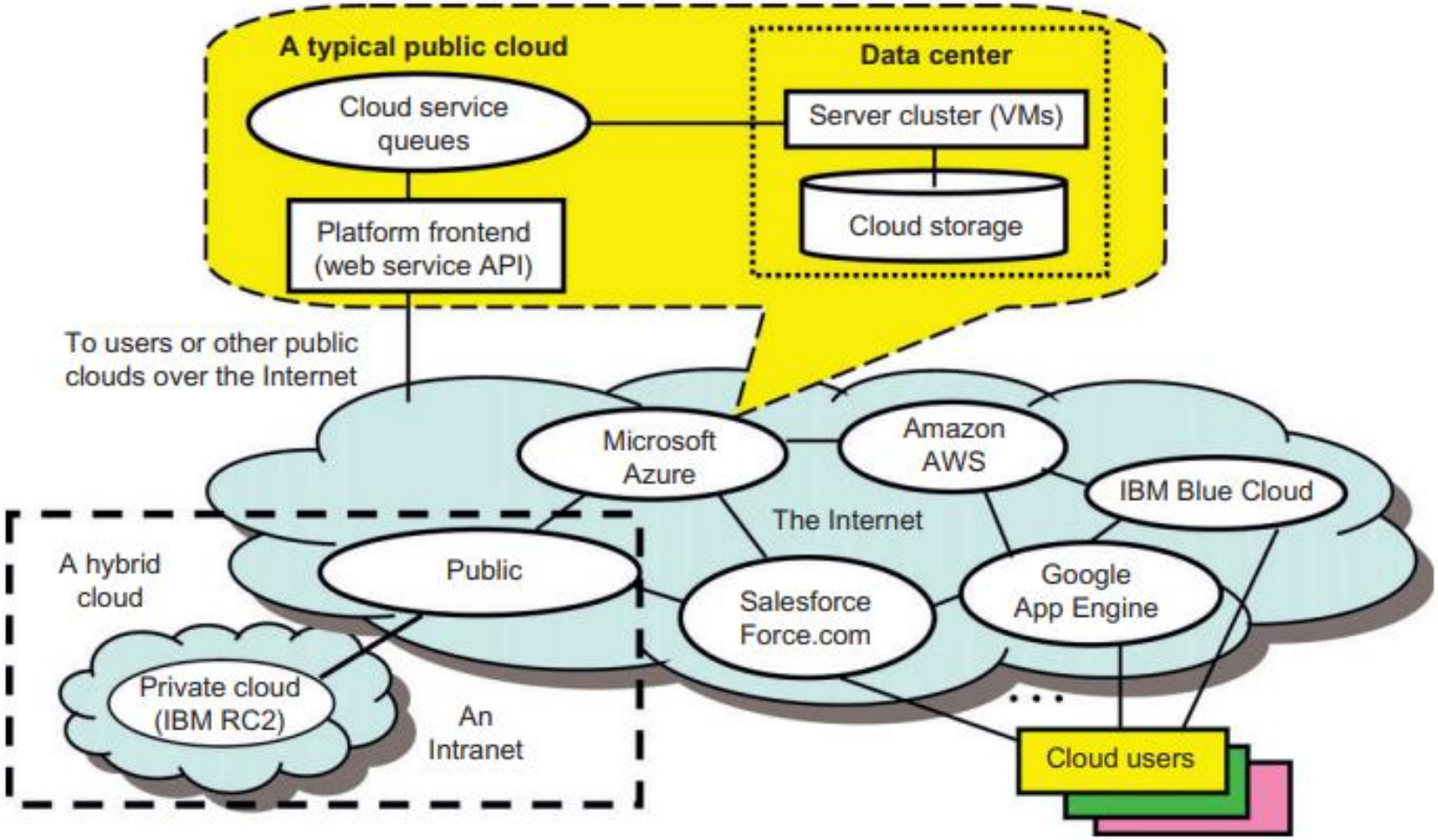


FIGURE 4.1

Public, private, and hybrid clouds illustrated by functional architecture and connectivity of representative clouds available by 2011.

Cloud computing : Cost

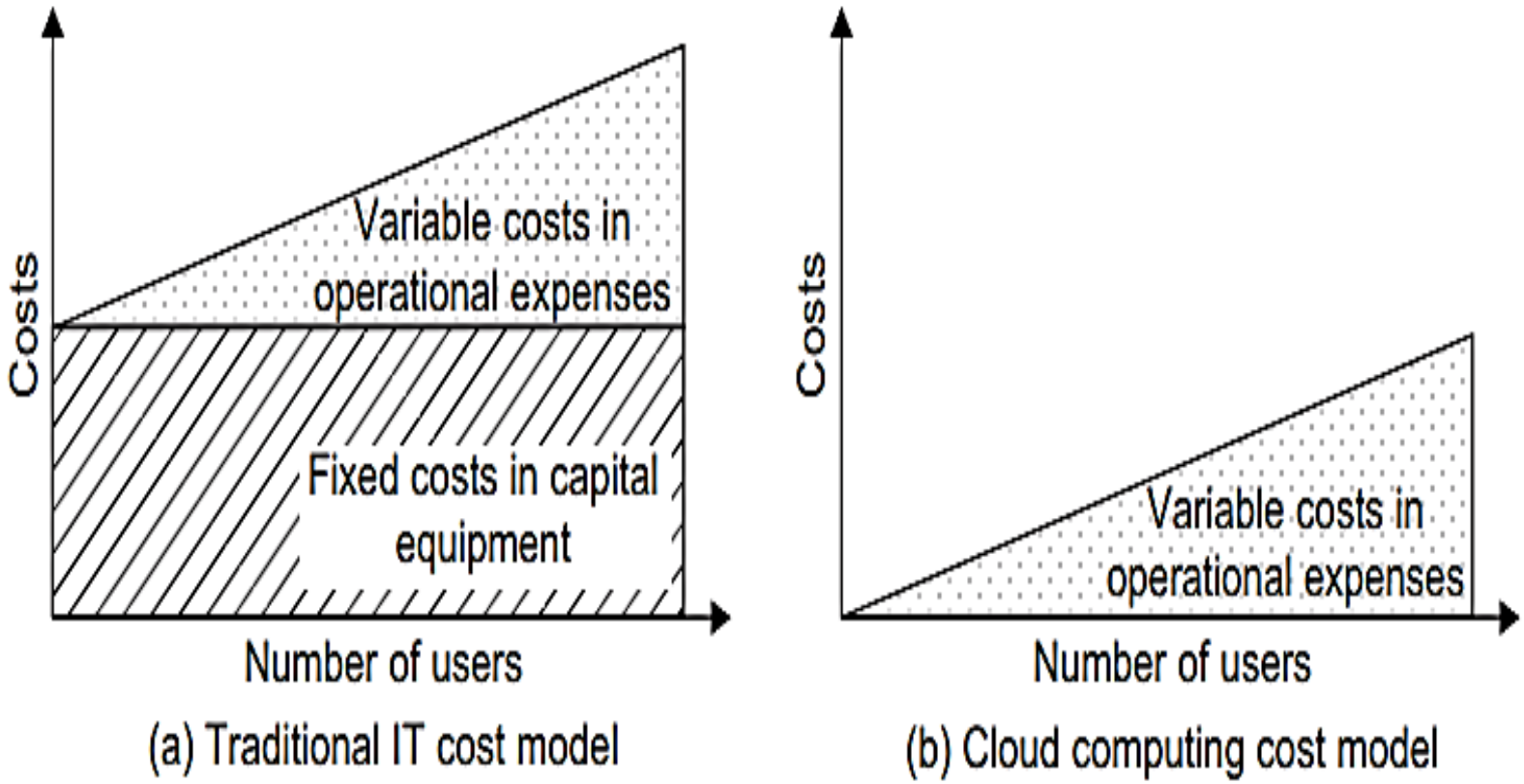


FIGURE 4.3

Computing economics between traditional IT users and cloud users.

Service Models

Cloud computing is based on service models. These are categorized into three basic service models which are –

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

Cloud computing

Infrastructure as a service (IaaS)

- Most basic cloud service model
- Cloud providers offer computers, as physical or more often as virtual machines, and other resources. This model allows users to use virtualized IT resources for computing, storage, and networking.
- Virtual machines are run as guests by a hypervisor, such as Xen or KVM.
- Cloud users deploy their applications by then installing operating system images on the machines as well as their application software.
- Cloud providers typically bill IaaS services on a utility computing basis, that is, cost will reflect the amount of resources allocated and consumed.

Examples of IaaS include: Amazon Cloud Formation (and underlying services such as Amazon **EC2**), Rackspace Cloud, Terremark, and Google Compute Engine

***A hypervisor is computer software, firmware or hardware that creates and runs virtual machines*

Cloud computing

Platform as a service (PaaS)

- Cloud providers deliver a computing platform typically including operating system, programming language execution environment, database, and web server.
- Application developers develop and run their software on a cloud platform without the cost and complexity of buying and managing the underlying hardware and software layers.

- Examples of PaaS include: Amazon Elastic Beanstalk, Cloud Foundry, Heroku, Force.com, EngineYard, Mendix, Google App Engine, Microsoft Azure and OrangeScape.

Cloud computing

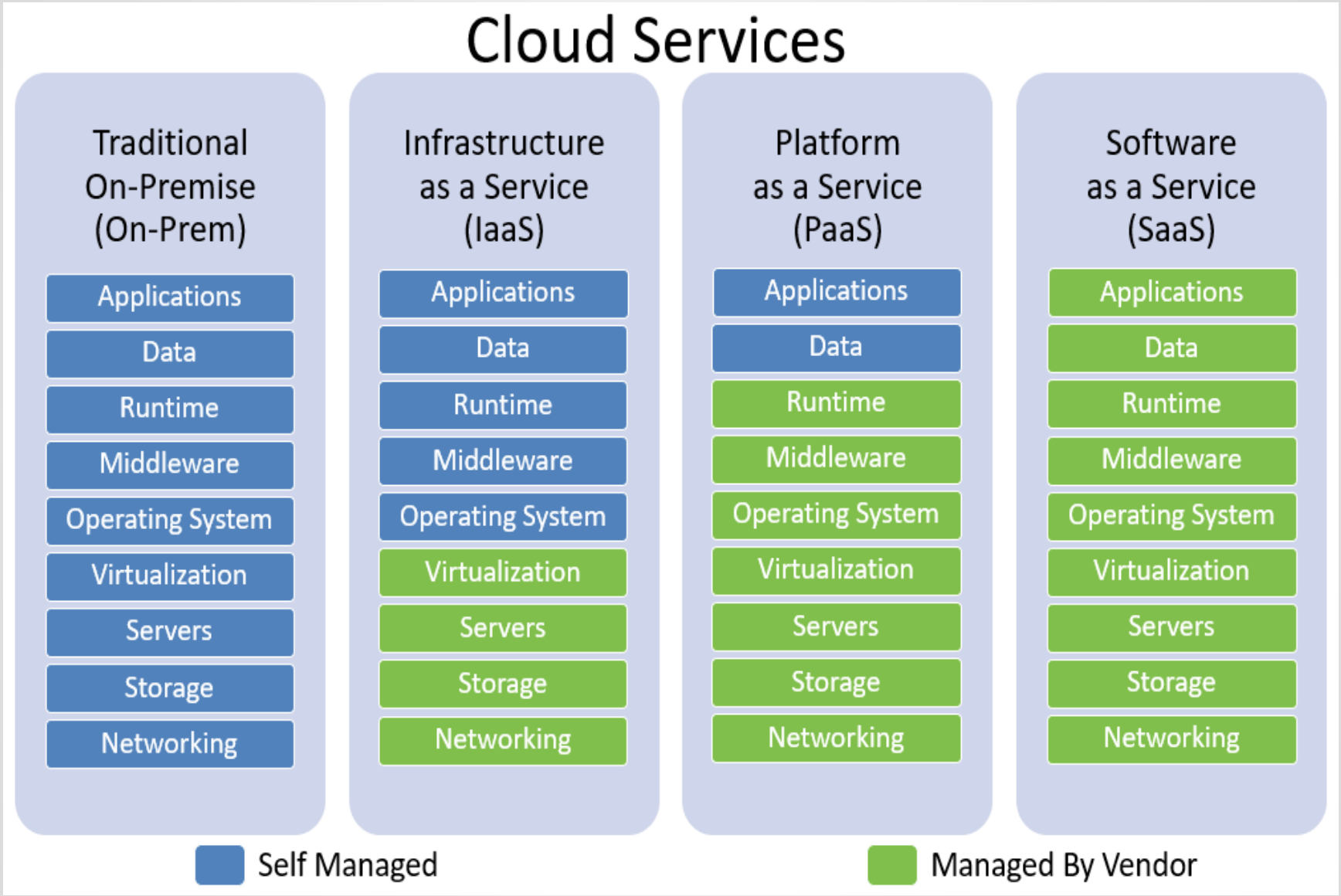
Software as a service (SaaS)

- Cloud providers install and operate application software in the cloud and cloud users access the software from cloud clients.

- The pricing model for SaaS applications is typically a monthly or yearly flat fee per user, so price is scalable and adjustable if users are added or removed at any point.

- **Examples of SaaS include**: Google Apps, innkeypos, Quickbooks Online, Limelight Video Platform, Salesforce.com, and Microsoft Office 365.

Cloud computing Services



Service-Oriented Architecture

Service-Oriented Architecture (SOA)

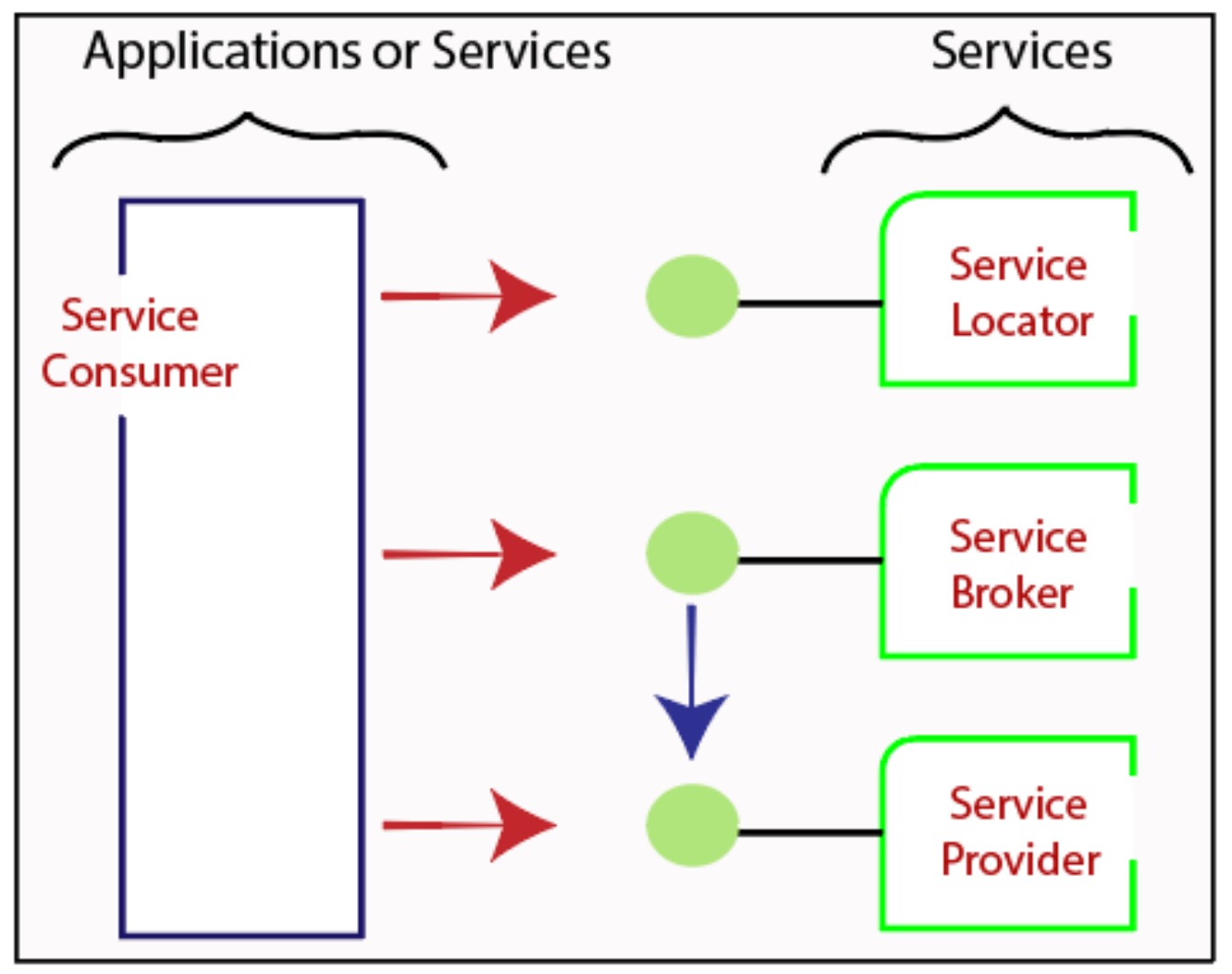
Service-Oriented Architecture (SOA) is a style of software design where services are provided to the other components by application components, through a communication protocol over a network.

There are two major roles within Service-oriented Architecture:

Service provider: The service provider is the maintainer of the service and the organization that makes available one or more services for others to use. To advertise services, the provider can publish them in a registry, together with a service contract that specifies the nature of the service, how to use it, the requirements for the service, and the fees charged.

Service consumer: The service consumer can locate the service metadata in the registry and develop the required client components to bind and use the service.

Service-Oriented Architecture (SOA)



Service-Oriented Architecture (SOA)

•**Services -** The services are the logical entities defined by one or more published interfaces.

•**Service provider -** It is a software entity that implements a service specification.

•**Service consumer -** It can be called as a requestor or client that calls a service provider. A service consumer can be another service or an end-user application.

•**Service locator -** It is a service provider that acts as a registry. It is responsible for examining service provider interfaces and service locations.

•**Service broker -** It is a service provider that pass service requests to one or more additional service providers.

Service-Oriented Architecture (SOA)

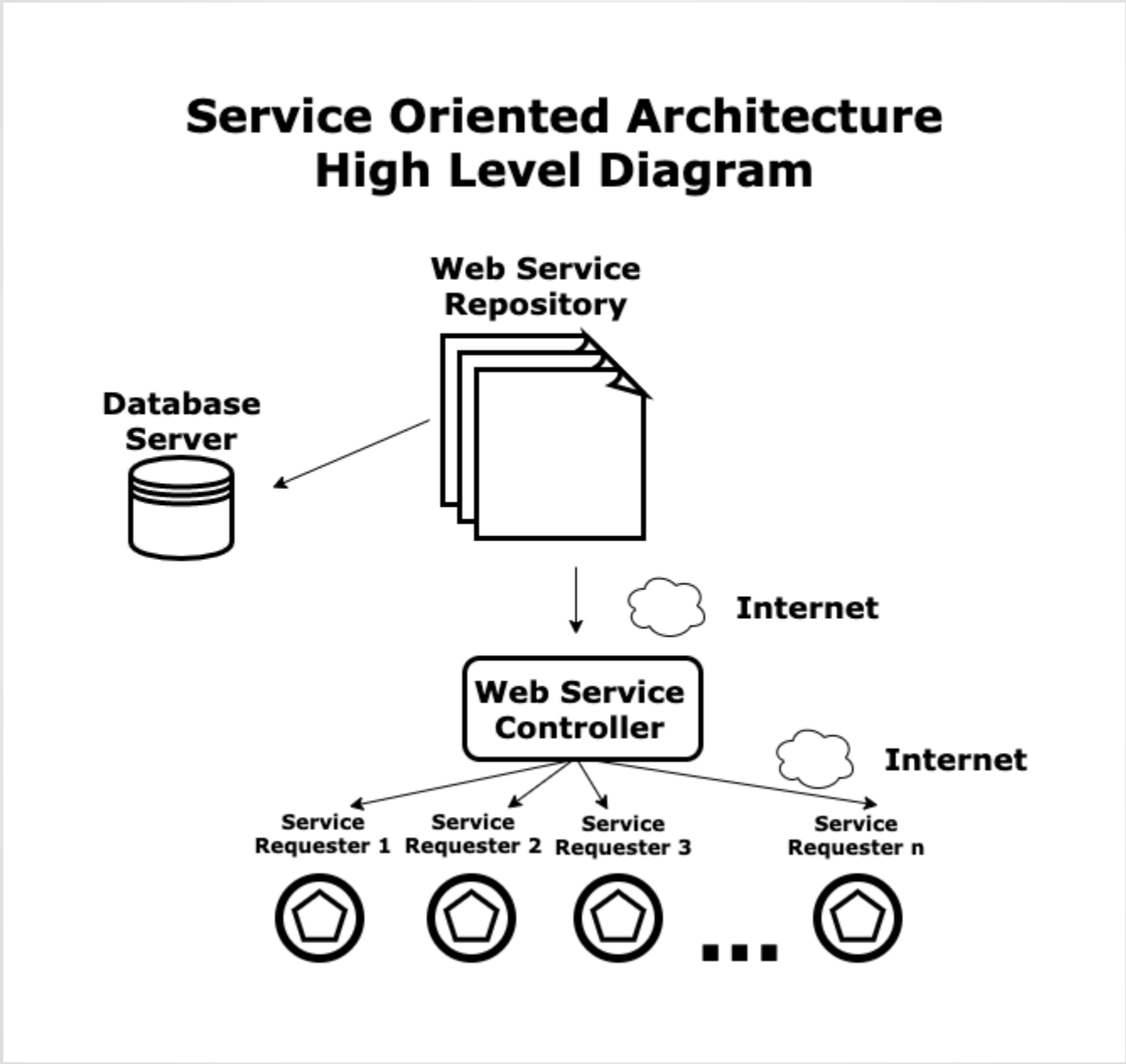
Why to use SOA?

- SOA is widely used in market which responds quickly and makes effective changes according to market situations.
- The SOA keep secret the implementation details of the subsystems.
- It allows interaction of new channels with customers, partners and suppliers.
- It authorizes the companies to select software or hardware of their choice as it acts as platform independence.

Features

- SOA uses interfaces which solves the difficult integration problems in large systems.
- SOA communicates customers, providers and suppliers with messages by using the XML schema.
- It uses the message monitoring to improve the performance measurement and detects the security attacks.
- As it reuses the service, there will be lower software development and management costs.

Service-Oriented Architecture (SOA)



Service-Oriented Architecture (SOA)

A standard client-server architecture has these parts:

Web service repository: This is a library of web services built to serve external requests for information. The served information is usually a little piece of information, like a number, a word, some variables, etc.

Web service controller: This module communicates the information in the web service repository with the service requesters. When an **external service requester calls** a certain function from the web service repository, the web service controller **interprets the call** and looks for the function in the web server repository. Then it executes the function and **returns a value to the requester**.

Service-Oriented Architecture (SOA)

Database server: This server contains the tables, indexes, and data managed by the core application. Searches and insert/delete/update operations are executed here.

Service requesters: These are external applications that request services from the web service repository through the internet, such as an organization requesting flight information from an airline or another company asking the package carrier for the location of a package at a given moment.

Note that service-oriented architecture is **created by the companies providing the service** - not the ones consuming it, and this is done to simplify the connections to possible clients.

Service-Oriented Architecture (SOA)

Advantages of SOA:

- **Service reusability:** In SOA, applications are made from existing services.Thus, services can be reused to make many applications.
- **Easy maintenance:** As services are independent of each other they can be updated and modified easily without affecting other services.
- **Platform independent:** SOA allows making a complex application by combining services picked from different sources, independent of the platform.
- **Availability:** SOA facilities are easily available to anyone on request.
- **Reliability:** SOA applications are more reliable because it is easy to debug small services rather than huge codes
- **Scalability:** Services can run on different servers within an environment, this increases scalability

Service-Oriented Architecture (SOA)

Disadvantages of SOA:

- **High overhead:** A validation of input parameters of services is done whenever services interact this decreases performance as it increases load and response time.
- **High investment:** A huge initial investment is required for SOA.
- **Complex service management:** When services interact they exchange messages to tasks. the number of messages may go in millions. It becomes a cumbersome task to handle a large number of messages.

Case-study: Solving a Business Problem With Service- Oriented Architecture

GlobalWeather is a global company that sells information about weather conditions. Here are some important facts:

- GlobalWeather is based in the U.S. and has more than 700 employees.
- GlobalWeather sells weather information to companies about countries, regions, cities, or even precise locations defined by latitude and longitude.

- GlobalWeather clients are all over the world and usually consume information using the internet. This is especially useful for transport companies: airlines, bus services, and logistics companies that have a truck fleet to distribute merchandise over a region. Additionally, any companies that coordinate outdoor events.

- Clients need information in real-time.

Case-study: Solving a Business Problem With Service- Oriented Architecture

What's the Business Problem?

GlobalWeather sells information to clients in real-time, but it cannot let all clients access their internal systems because of security constraints.

How can we solve this problem?

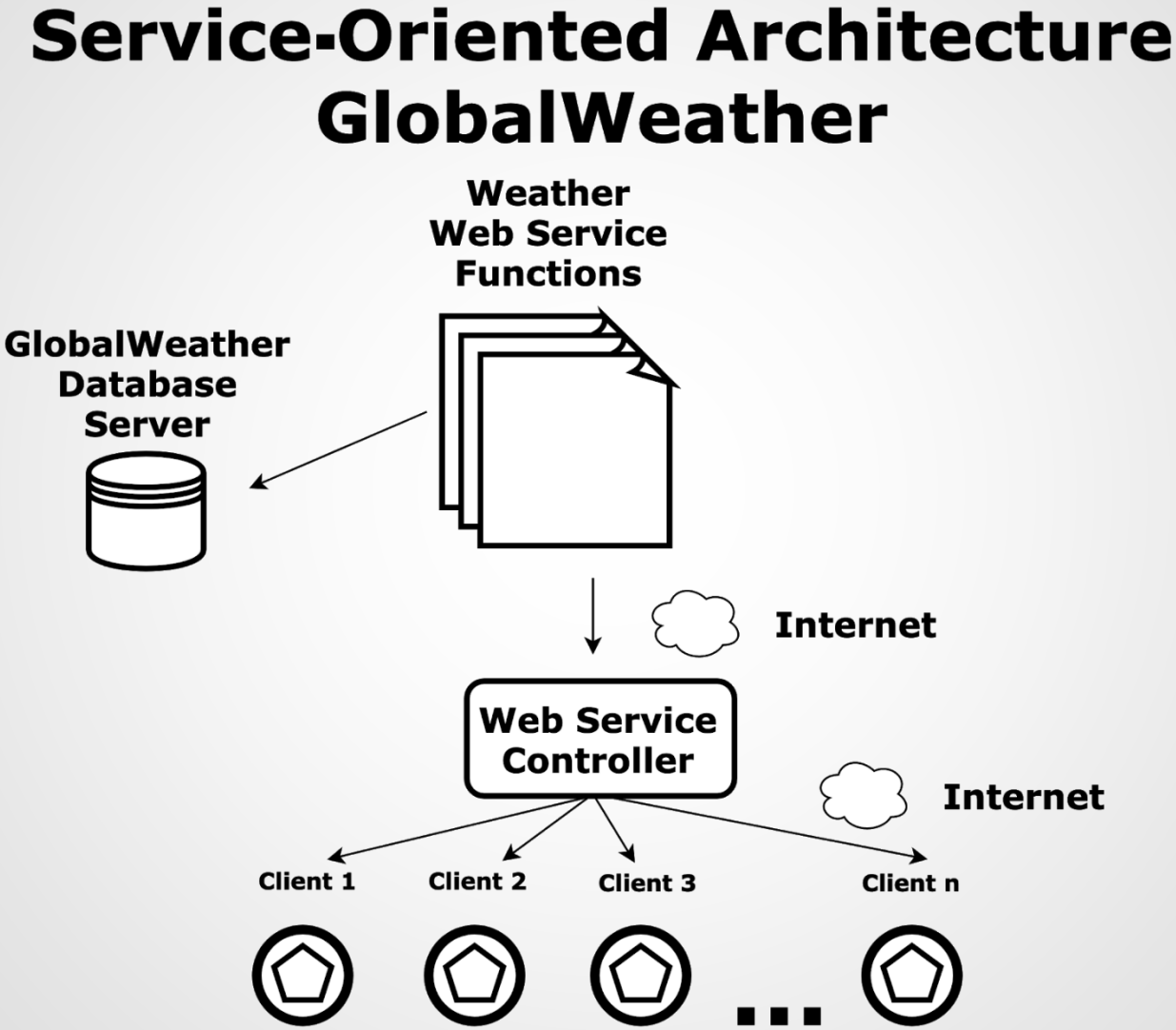
Let's look at the facts:

GlobalWeather has **thousands of clients.** To buy its services, each client needs to connect to GlobalWeather systems and get information manually. It is very **complex and time- consuming,** especially since the information requested is usually a little piece of information that can be transmitted via the internet.

Additionally, many clients are **websites that need to connect** to the information feed and quickly display weather information in real-time. For example, an airline that is selling a ticket from city A to city B wants to display the weather in city B once the client buys that ticket.

This situation is the right situation for a service-oriented architecture: GlobalWeather can build a **library of functions** and implement it as web services that their clients consume once they pay. This ensures a speedy, real-time service without allowing clients to access internal servers.

What's the Solution?



Case-study: Solving a Business Problem With Service- Oriented Architecture

Component of the system:

•**Weather web service functions:** This is a library of web services built to serve external websites with weather information such as rain, wind direction and speed, atmospheric pressure, etc.

•**Web service controller:** This module communicates the web service repository with the service requesters. When an external service requester calls a certain function (for instance, flight information, weather information, package status) from the web service repository, the web service controller interprets the call and looks for the function in the web server repository. Then it executes the function and returns a value to the requester.

•**Clients:** These are external applications that request services from the web service repository through the internet. These requests are coded into the requester's application according to a certain syntax published by the service provider.

•**GobalWeather database server:** This server contains the tables, indexes, and data managed by the system.

Case-study: Home Work

You work at a major airport administration department in your country. You have been asked to develop a software system to give the status of a certain flight that is in the air to some possible requesters: airlines, travel sites, hotel sites, etc.

There is a central system at the airport control area that works in real-time: it receives information for the radars and aircrafts as the flights progress.

Now, it's your job to create an architecture for them!

Here are some **key questions** you can ask yourself to get started:

I. Many flights are in the air at a certain moment. How can you get their flight status from the central system?
II. How are you going to pass this information to the external requesters?
III. Why does the airline need this system? Who is going to consume this information?

Can you produce an architecture diagram for this business setting?

Communication protocols of SOA

Communication protocols

Services communicate using established rules that determine data transmission over a network. These rules are called communication protocols. Some standard protocols to implement SOA include the following:

- Simple Object Access Protocol (SOAP)
- RESTful HTTP
- Apache Thrift
- Apache ActiveMQ
- Java Message Service (JMS)

You can even use more than one protocol in your SOA implementation.

Class Test

Class Test-03: 19-March-2023

Lecture: 12, 13

Class Test-04:

Lecture:

Thanks to All