INFS803
Cloud Computing

Distributed and Cloud Computing Infrastructure

Topics

- *Distributed computing and middleware*
- *The client-server model*
- *Inter-process communication*
- *Synchronous forms of middleware*
- *Asynchronous forms of middleware*
- *Message Oriented Middleware*
- *Enterprise application & e-Business integration*
- *Internet and WWW*
- *Cloud computing infrastructure at Amazon*

Distributed Computing

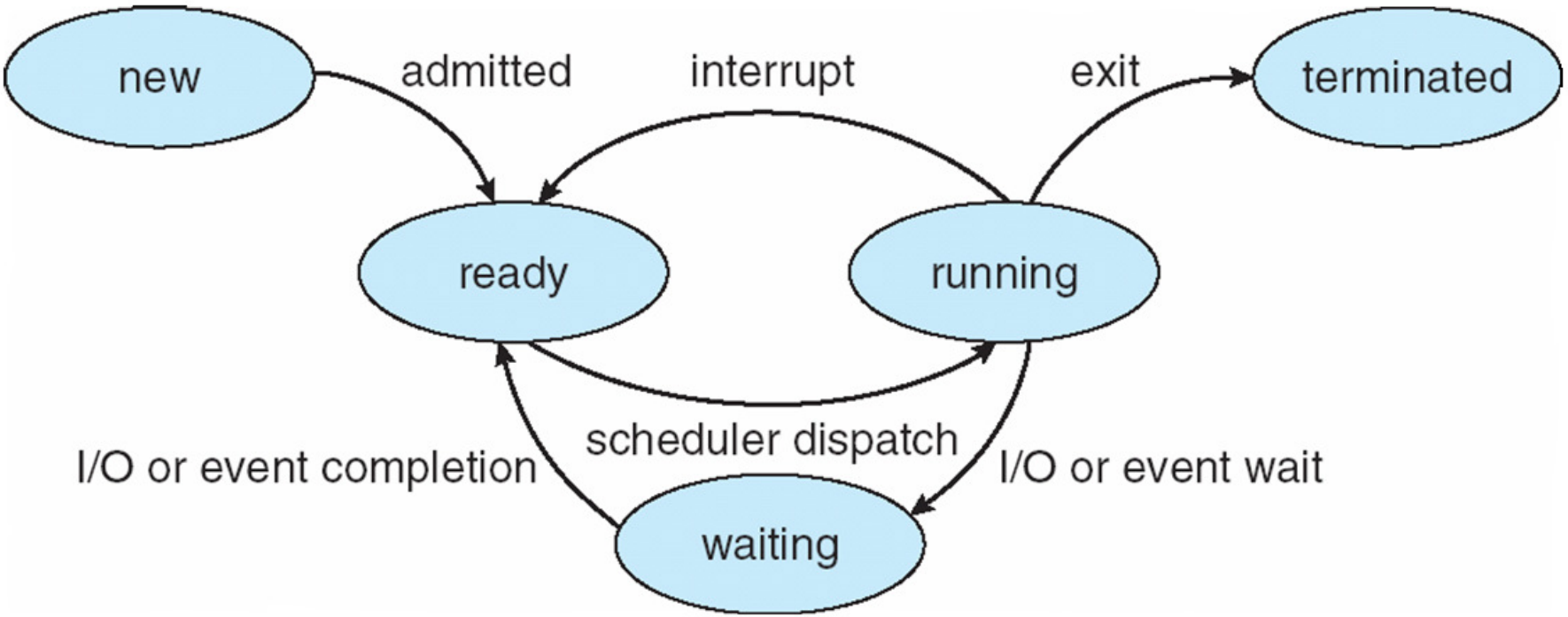
- A distributed system is characterized as a collection of heterogeneous networked computers, which communicate & coordinate their actions by passing messages.
  - Distribution is transparent to the user so that the system appears as a single integrated facility.

- One important characteristic of a distributed system is that processes are not executed on a single processor but rather span a number of processors.
  - This requires inter-process communication mechanisms.

Processes in a computer

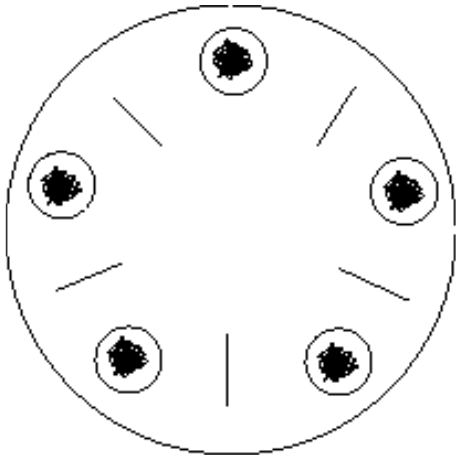
- A process is a program in execution. It is a unit of work within the system. Program is a **passive entity**, process is an **active entity**.
- Process needs resources to accomplish its task
  - CPU, memory, I/O, files
  - Initialization data
- Process termination requires reclaim of any reusable resources
- Process executes instructions sequentially, one at a time, until completion
- Typically system has many processes
  - Concurrency by multiplexing the CPUs among the processes / threads

Process lifecycle



Process Management Activities

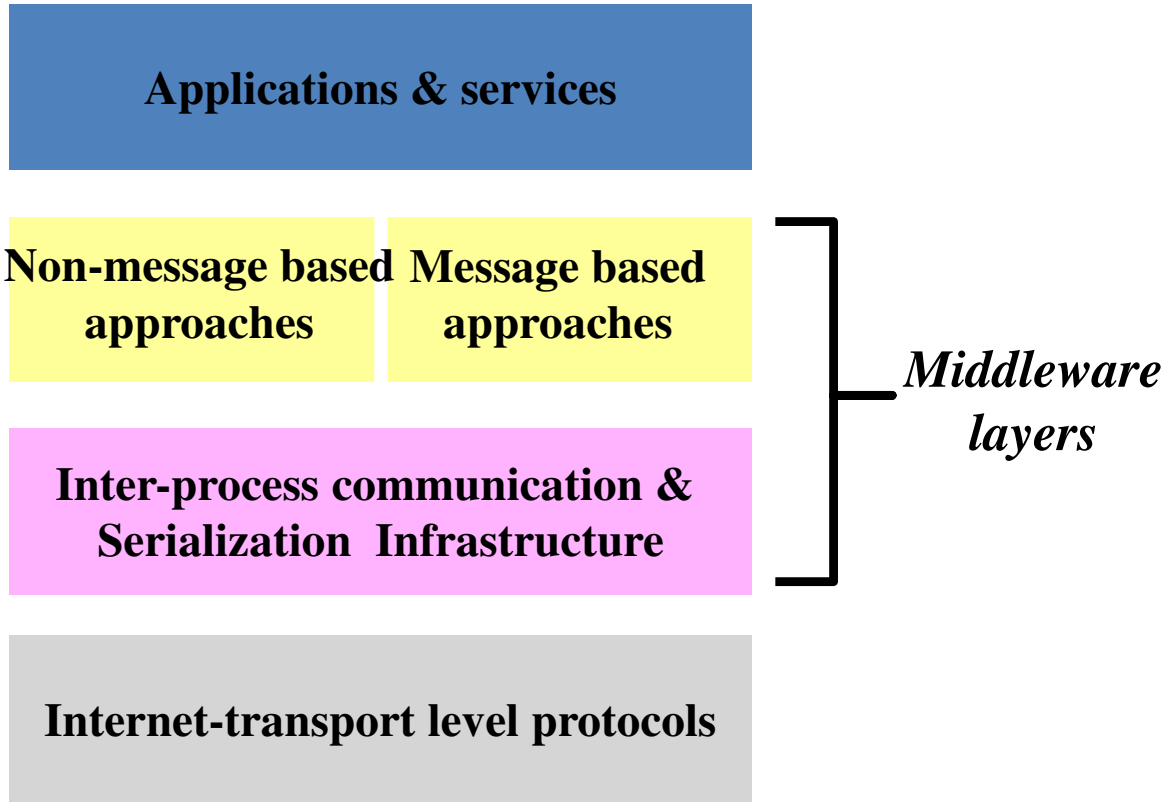
- Creating and deleting both user and system processes
- Suspending and resuming processes
- Providing mechanisms for process synchronization
- Providing mechanisms for process communication
- Providing mechanisms for deadlock handling



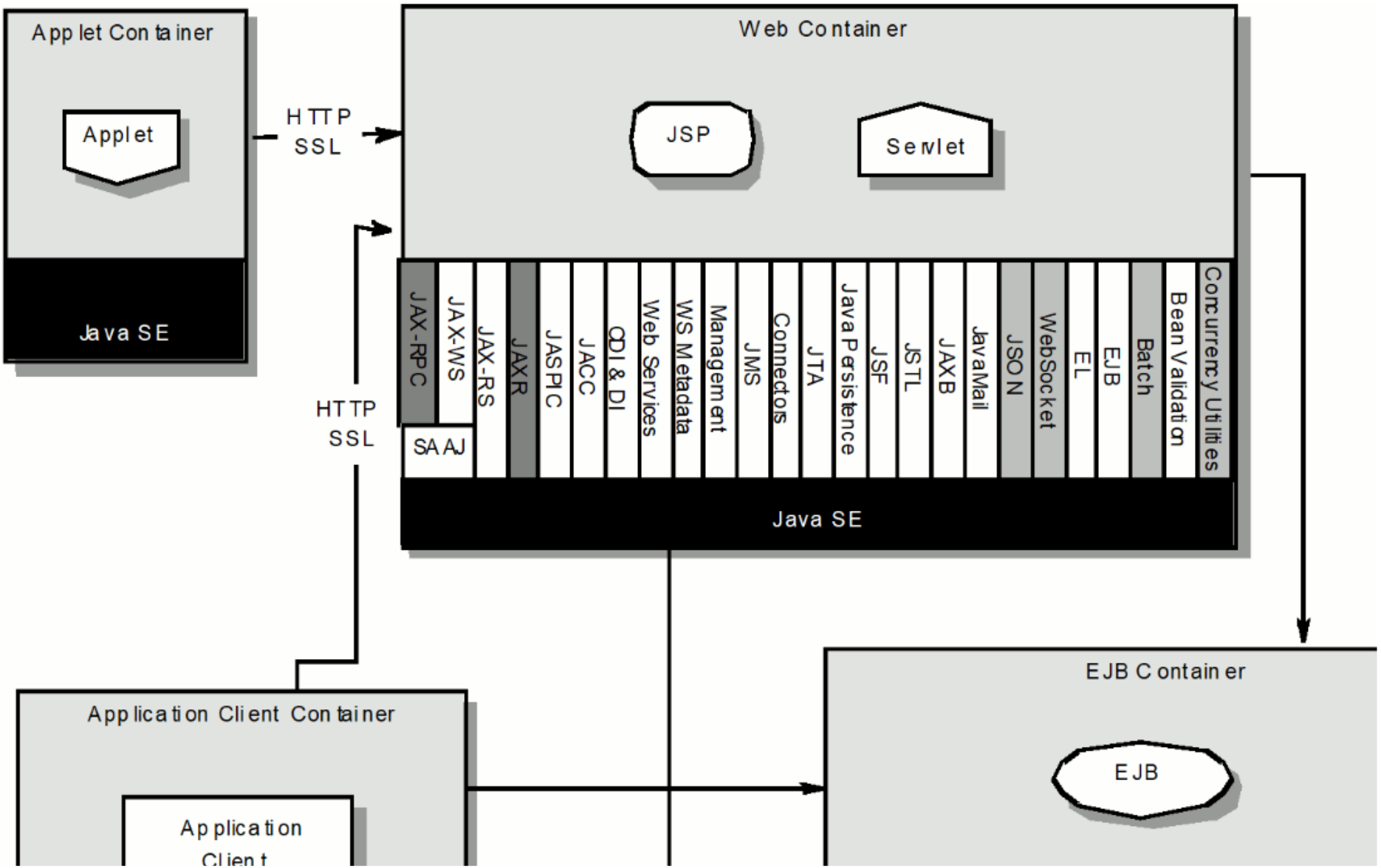
5 philosophers problem

Middleware

- Middleware provides a functional set of interfaces to allow an application to:
  - locate applications transparently across the network;
  - shield software developers from low-level, tedious and error-prone platform details;
  - provide a consistent set of higher-level abstractions that are much closer to application requirements;
  - leverage previous developments and reuse them;
  - provide services such as reliability, availability, authentication and security.



Example: Java EE/Jakarta EE



Middleware is part of PaaS

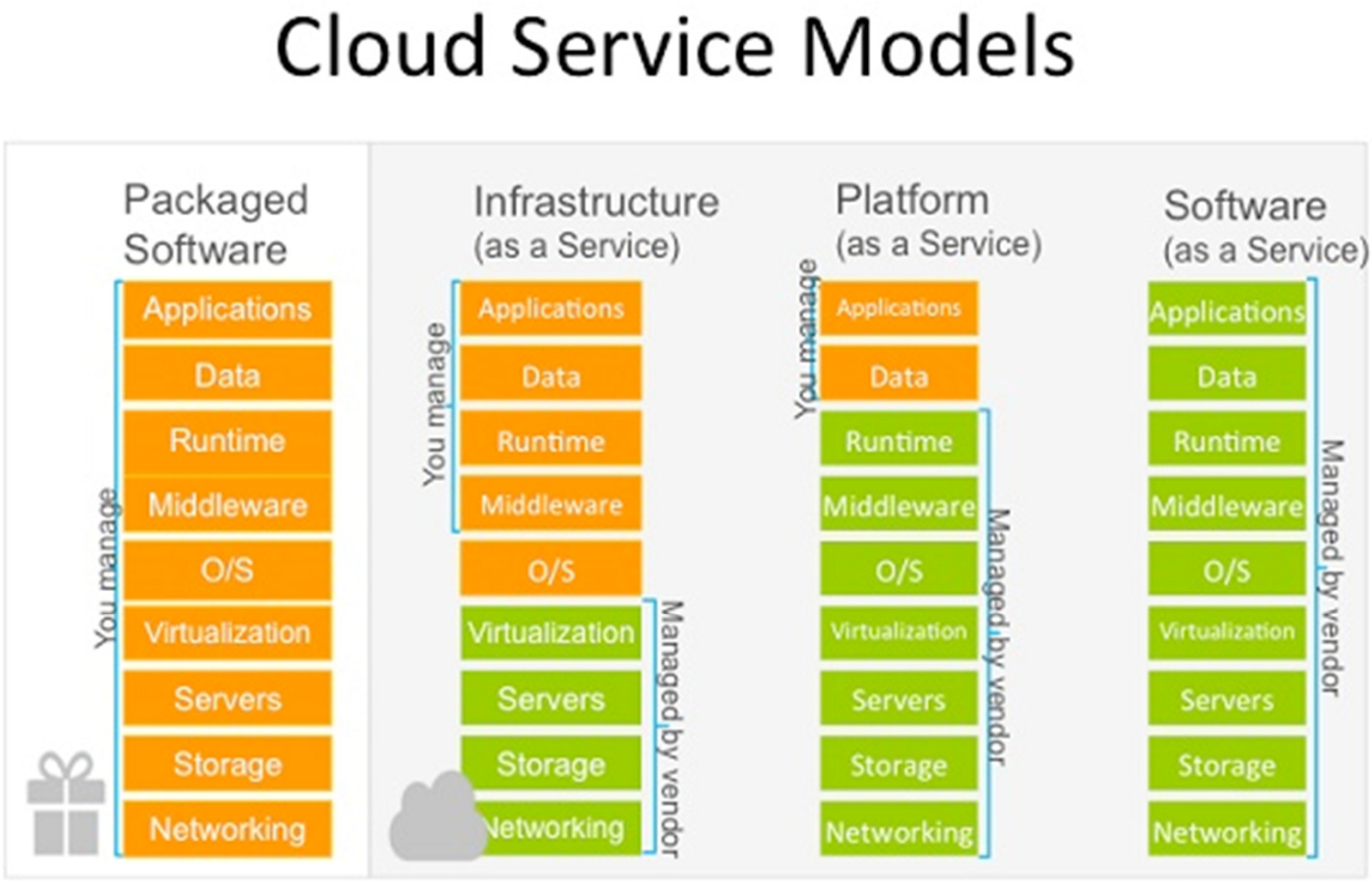
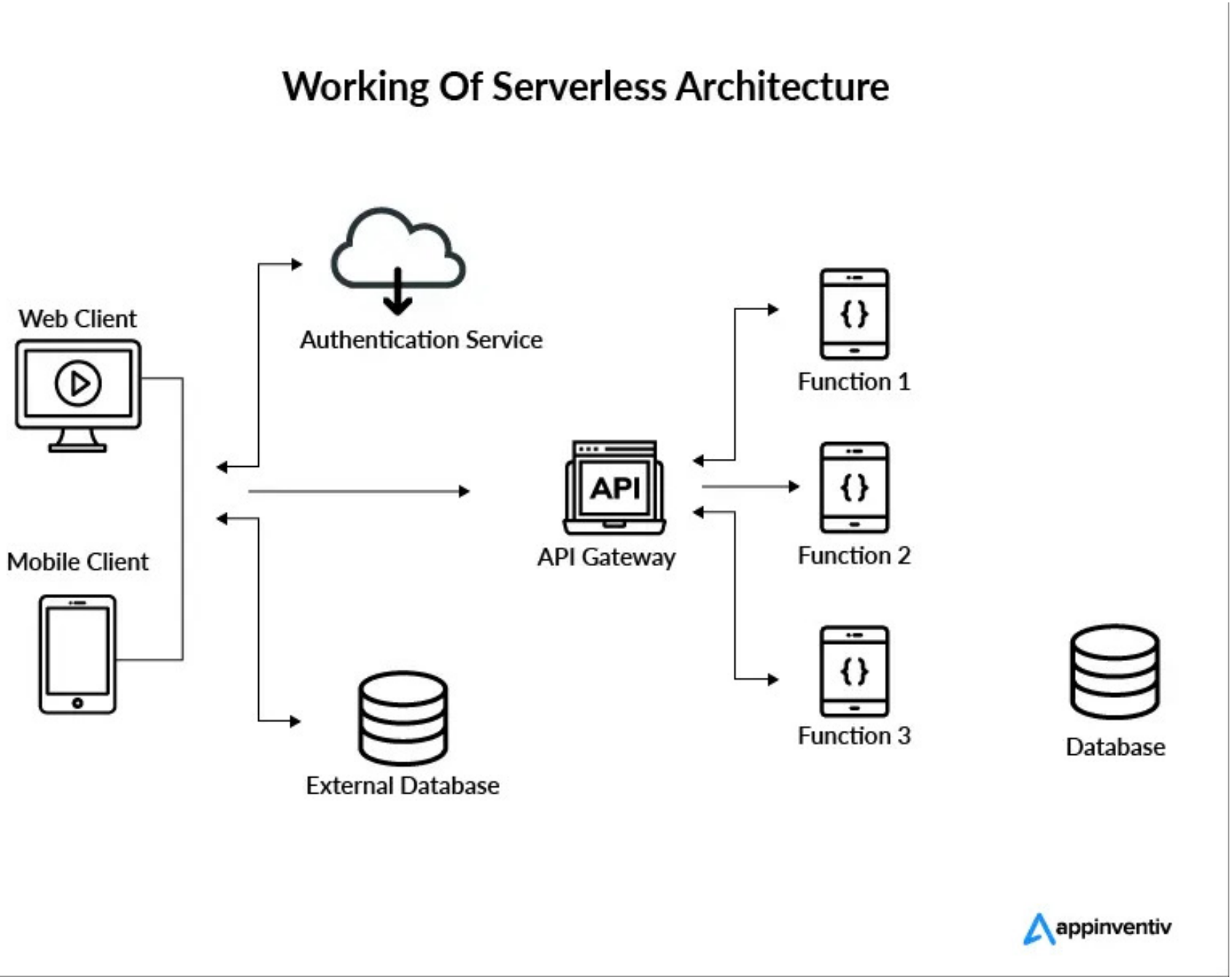


Figure 1. **Source: Microsoft Azure**

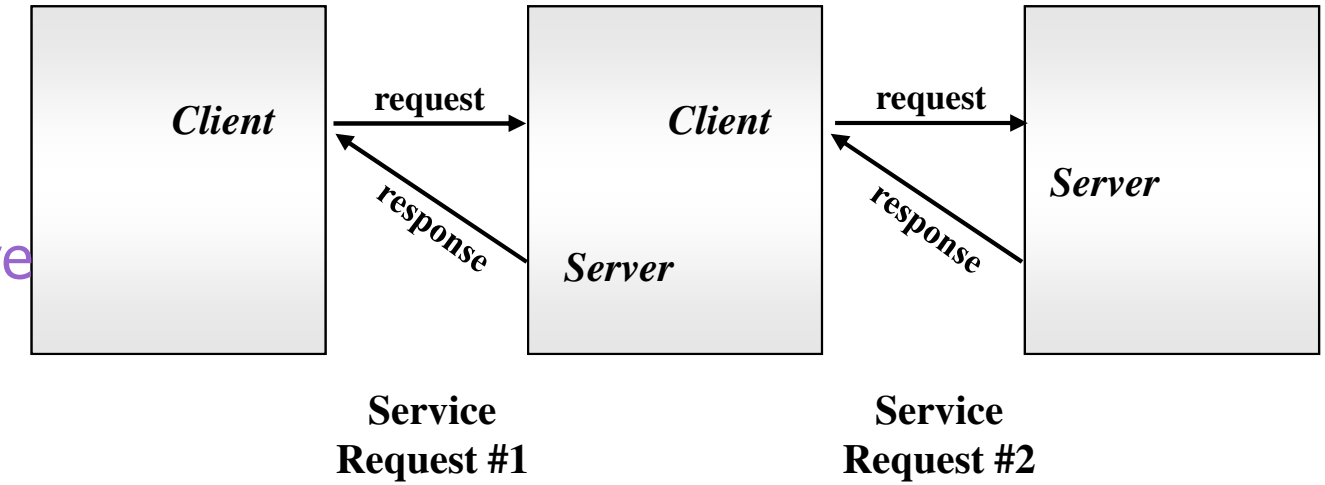
Recent trend



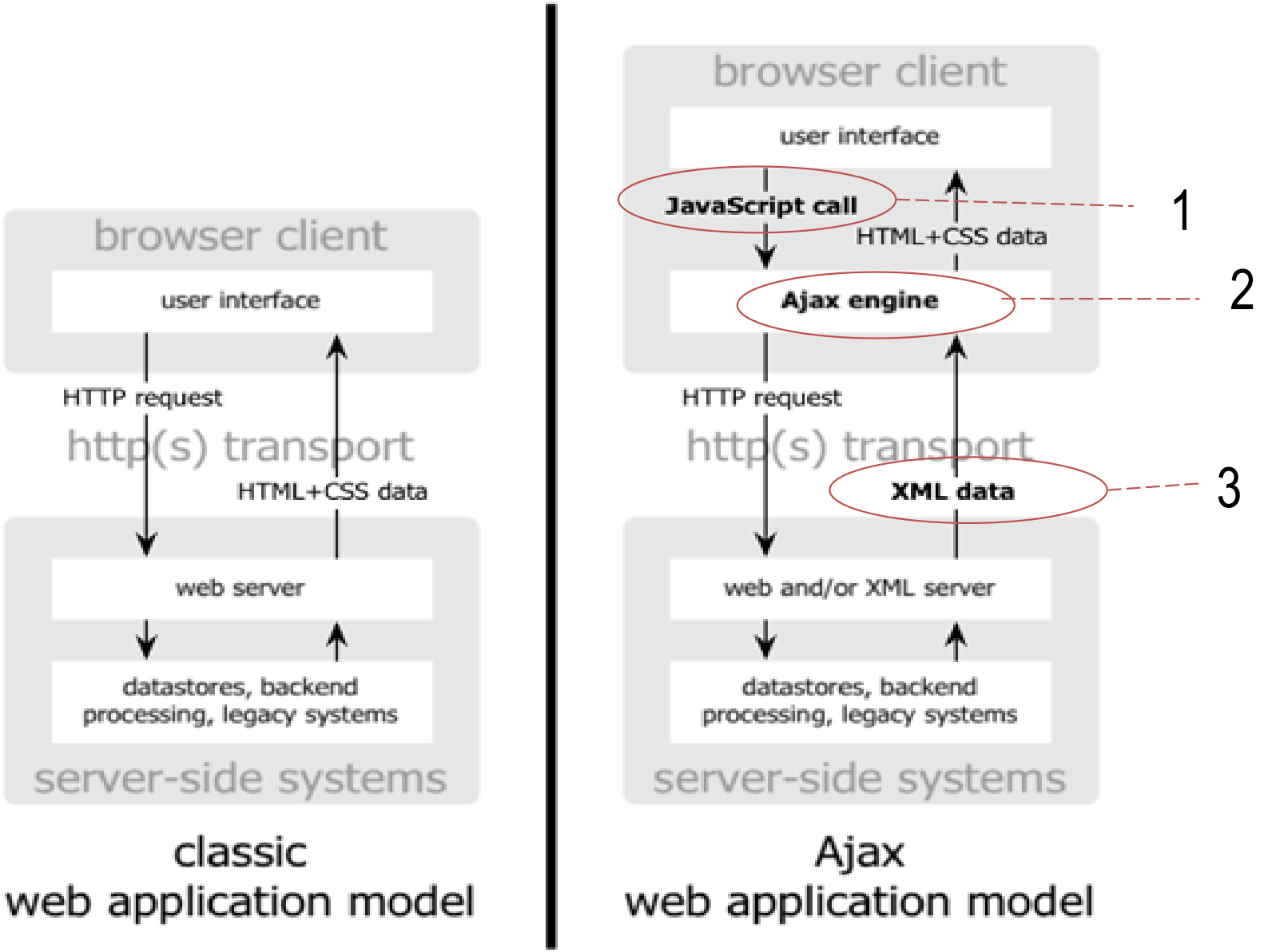
Client-server model

- A client/server is an architecture in which processing and storage tasks are divided between two classes of network members, clients & servers.
- Client/server involves client processes (service consumers) requesting service from server processes (service providers). Servers may in turn be clients of other servers.
  - The client machine runs software & applications that are stored locally. The client makes requests to servers and is also responsible for the user interface.
  - Some of the applications may be stored and executed on the server, but most of it is on the client. The server also provides the data for the application.

How to improve Interactivity?

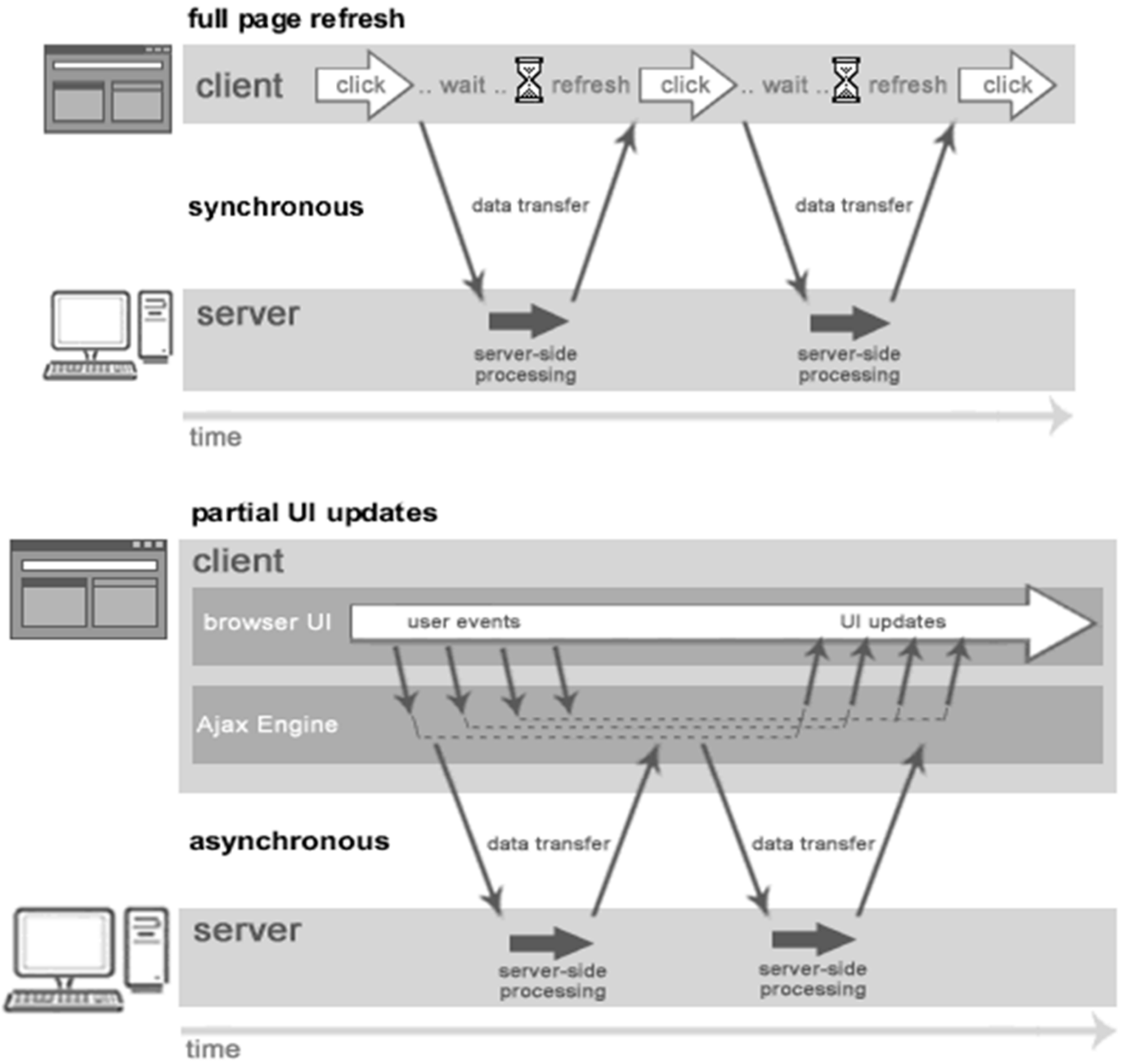


Ajax vs. Classic client-server Model



http://www.adaptivepath.com/publications/essays/archives/000385.php

Ajax vs. Classic Communication Models



Group Discussion

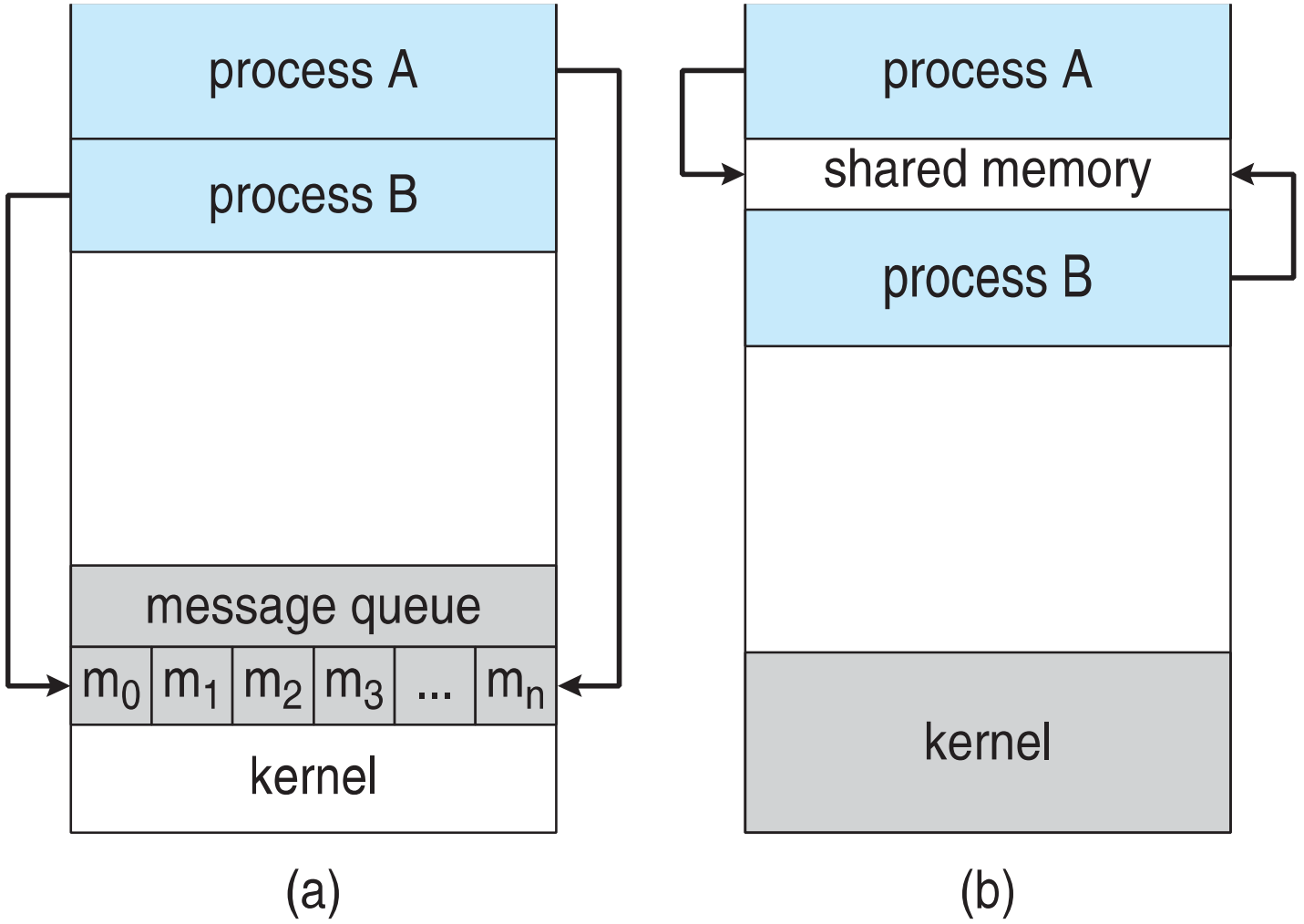
- Describe the distributed/cloud computing environment of an organization that you are familiar with (e.g., your organization). You may visually describe its architecture.

Topics

- *Inter-process communication*

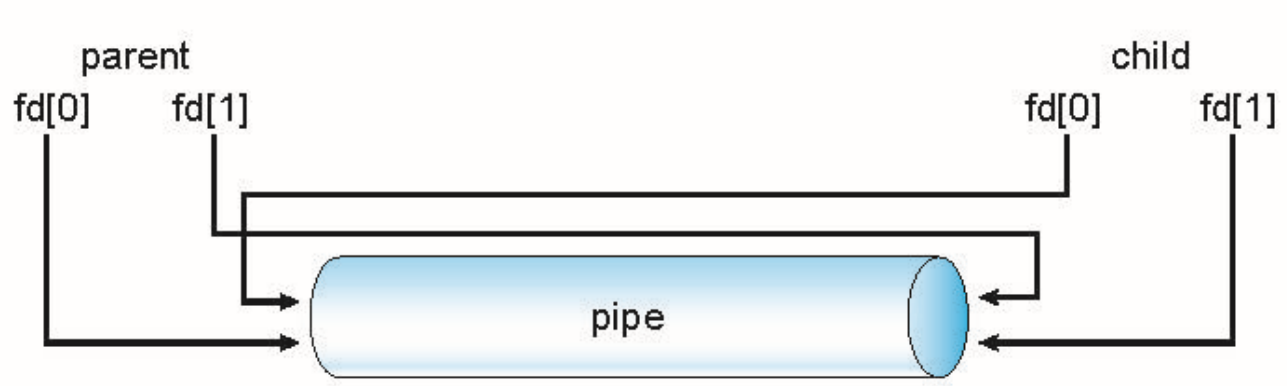
IPC Models

(a) Message passing. (b) shared memory.



UNIX Pipes

- Ordinary Pipes allow communication in standard producer-consumer style
- Producer writes to one end (the **write-end** of the pipe)
- Consumer reads from the other end (the **read-end** of the pipe)
- Ordinary pipes are therefore unidirectional
- Require parent-child relationship between communicating processes

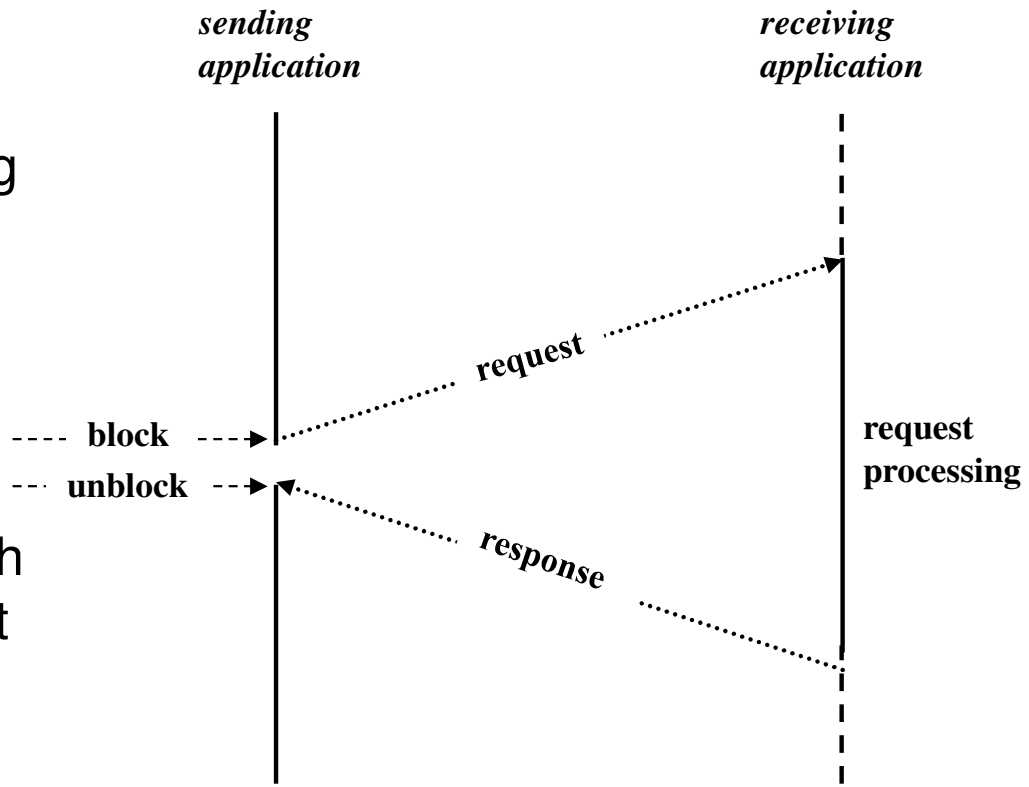


Messaging

- Distributed systems & applications communicate by exchanging messages. Messaging enables high-speed, asynchronous, program-to-program communication with reliable delivery.
- Message passing between a pair of processes is supported by two message communication operations: *send* and *receive*, defined in terms of destinations and messages.
- *Marshalling (serialization)* is the process of taking any form of structured data items and breaking up so that it can be transmitted as a stream of bytes over a communications network in such a way that the original structure can be reconstructed easily on the receiving end.
- *Unmarshalling (deserialization)* is the process of converting the assembled stream of bytes on arrival to produce an equivalent form of structured data at the destination point.

Synchronous & asynchronous messaging

- There are two basic modes of message communication:
- *synchronous* communication which is synchronized between two communicating application systems, which must both up and running.
  - Execution flow at the client's side is interrupted to execute the call.
- *asynchronous* communication where the caller employs a send and forget approach that allows it to continue to execute after it sends the message.
  - Here an application sends a request to another while it continues its own processing activities.

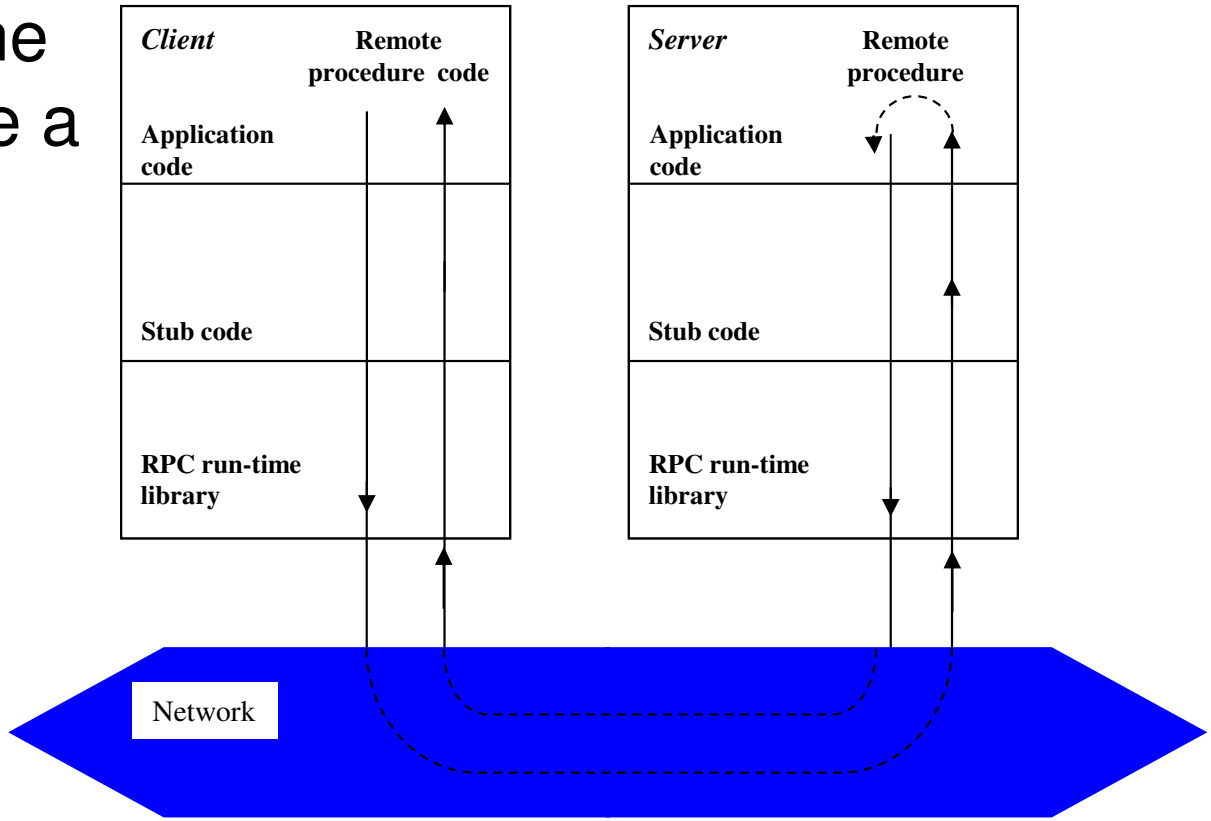


Topics

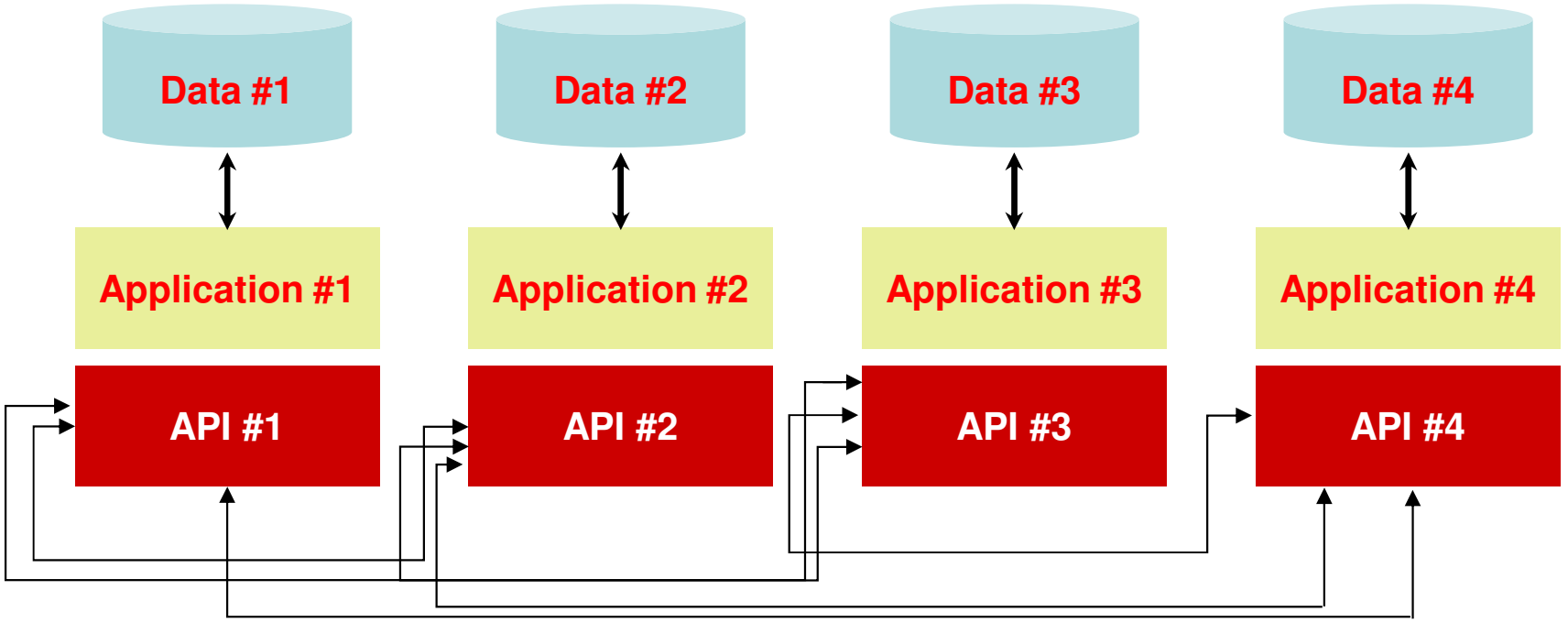
- *Synchronous forms of middleware*

Remote procedure calls

- RPC is a basic mechanism for inter-program communication where the application elements use a request/wait-for-reply (synchronous) model of communication.

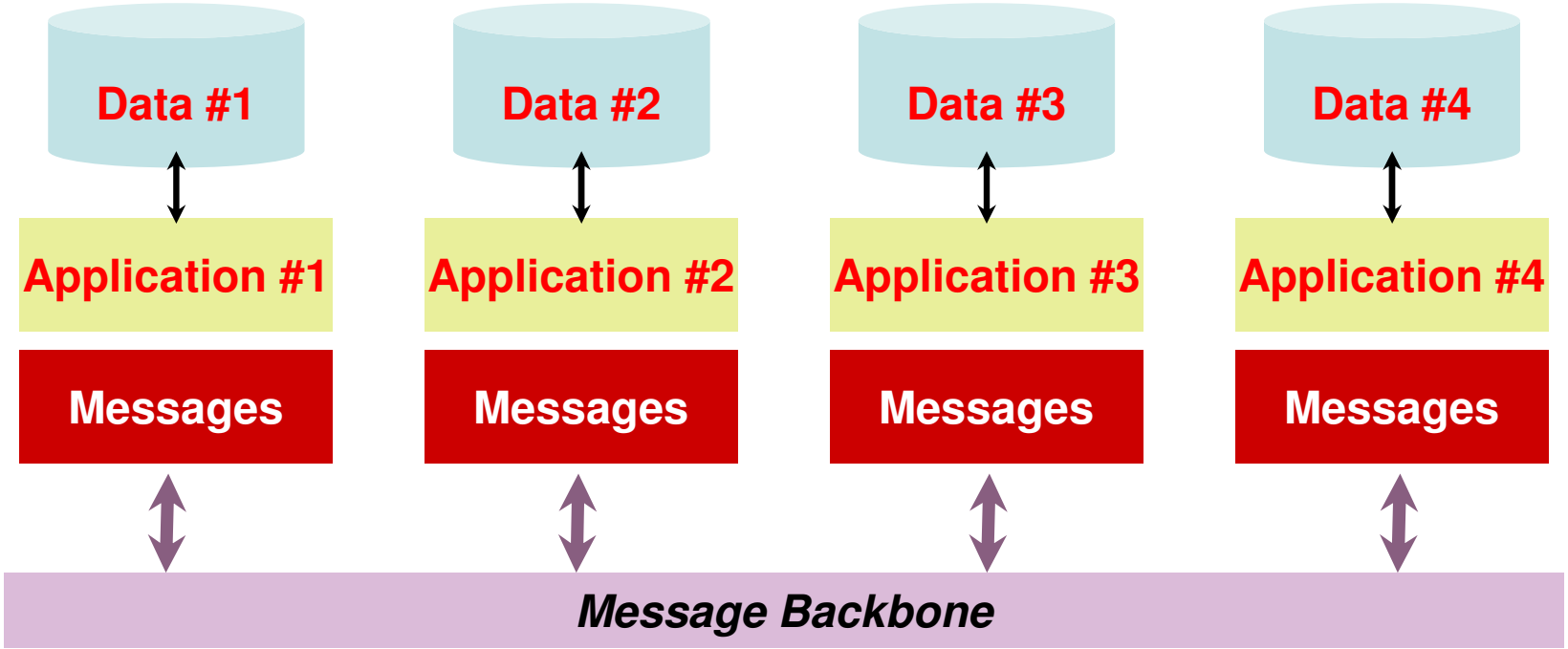


- RPC-style programming leads to *tight coupling* of interfaces and applications.
- In an RPC environment each application needs to know the intimate details of the interface of every other application – the number of methods it exposes & the details of each method signature it exposes.



Asynchronous communication

- Asynchronous communication promotes *loose coupling* in which an application does not need to know the intimate details of how to reach and interface with other applications.
- Each participant in a multi-step business process flow need only be concerned with ensuring that it can send a message to the messaging system.

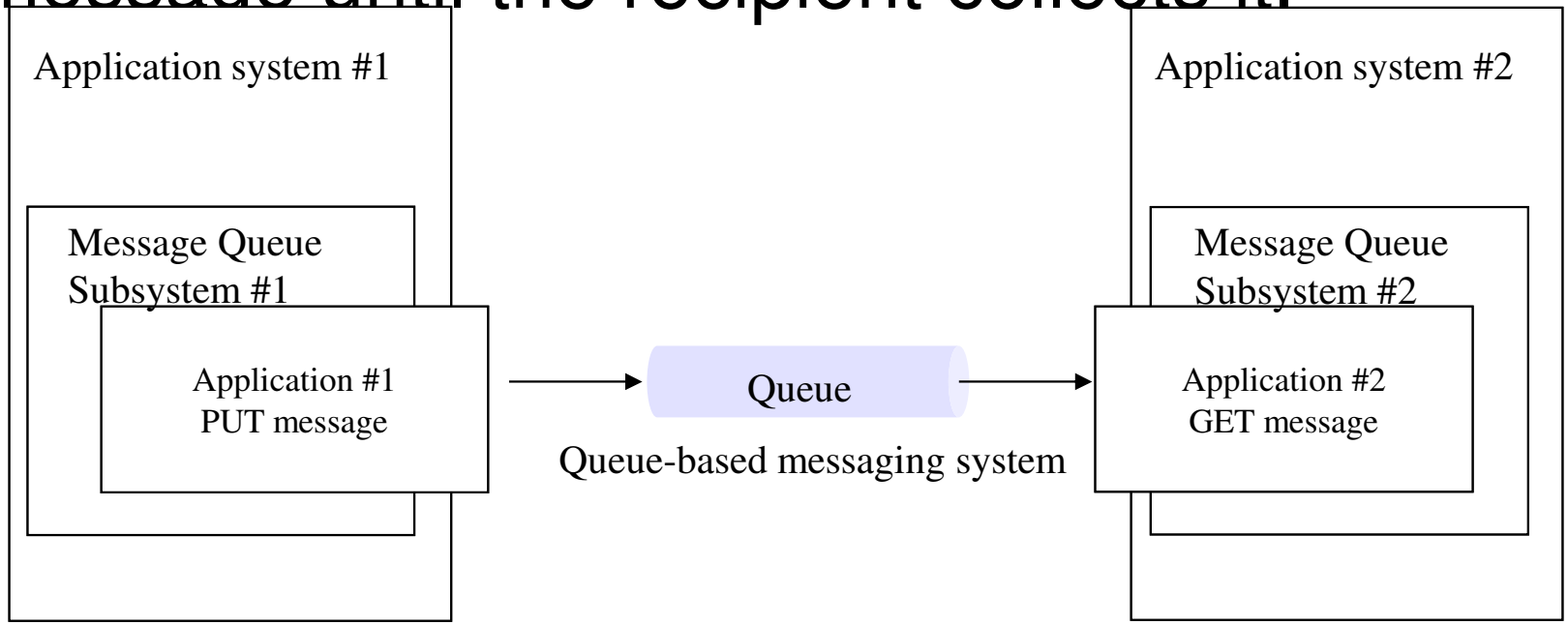


Topics

- *Asynchronous forms of middleware*

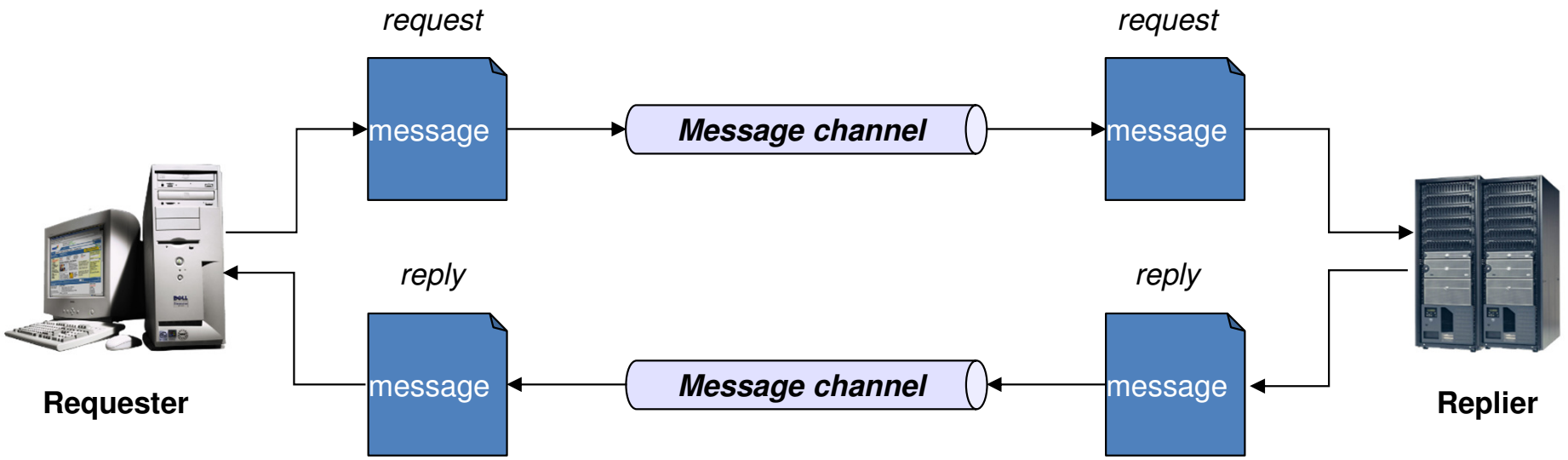
Store and forward messaging

- With the store and forward queuing mechanism, messages are placed on a virtual channel called a message queue by a sending application and are retrieved by the receiving application as needed.
  - The queue is a container that can keep hold of message until the recipient collects it.



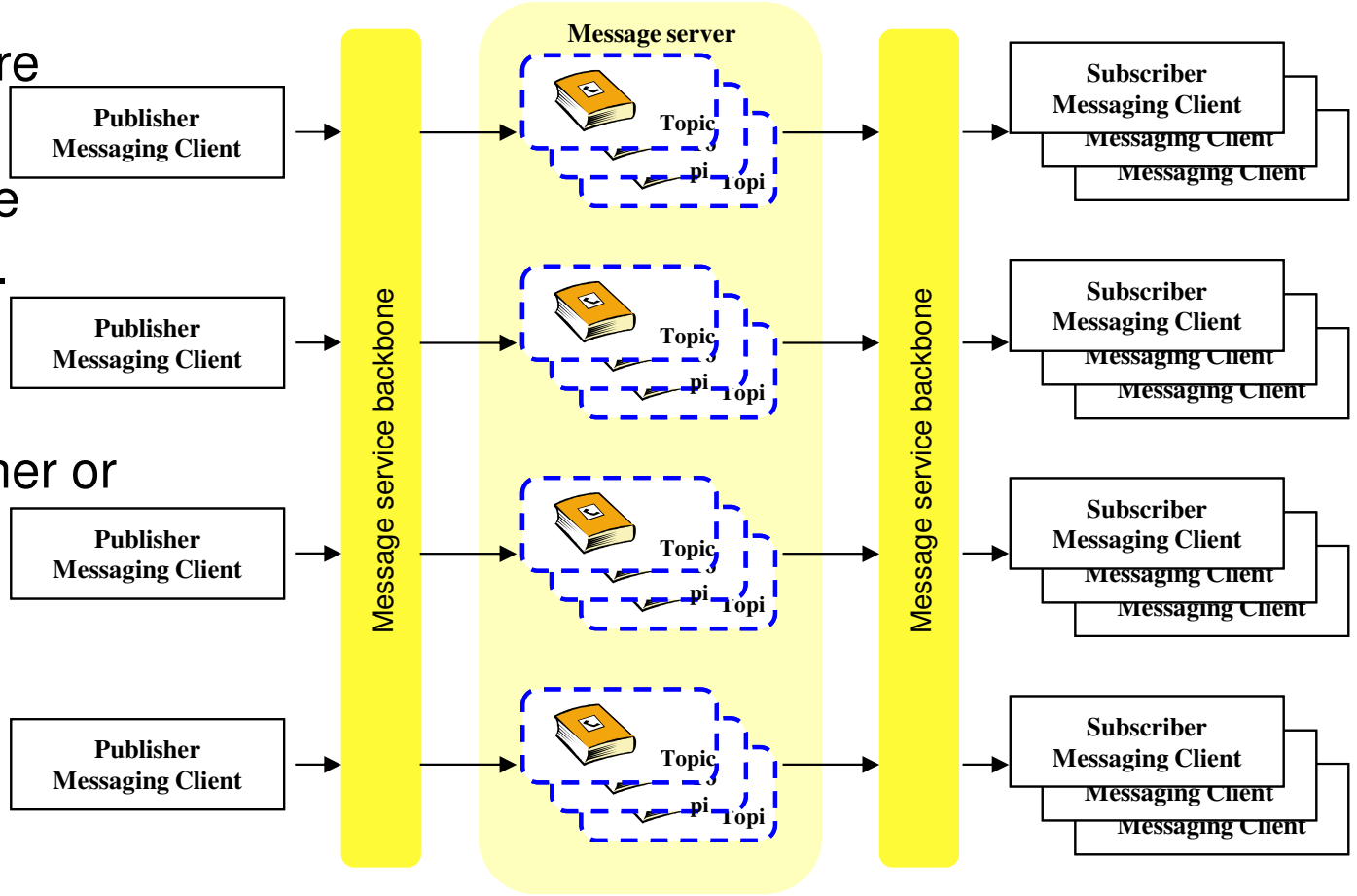
Asynchronous request/reply messaging

- Most asynchronous messaging mechanisms follow the “fire-and-forget” messaging principle where the sending application can conduct its work as usual once a message was asynchronously sent.
  - The sending application assumes that the message will arrive safely at its destination at some point in time.
  - This mode messaging does not preclude the necessity to perform request/reply operations.



Publish/Subscribe Messaging

- The application that produces information publishes it and all other applications that need this type of information, subscribe to it.
  - Messages containing the new information are placed in a queue for each subscriber by the publishing application.
  - Each application may have a dual role: it may act as a publisher or subscriber of different types of information.

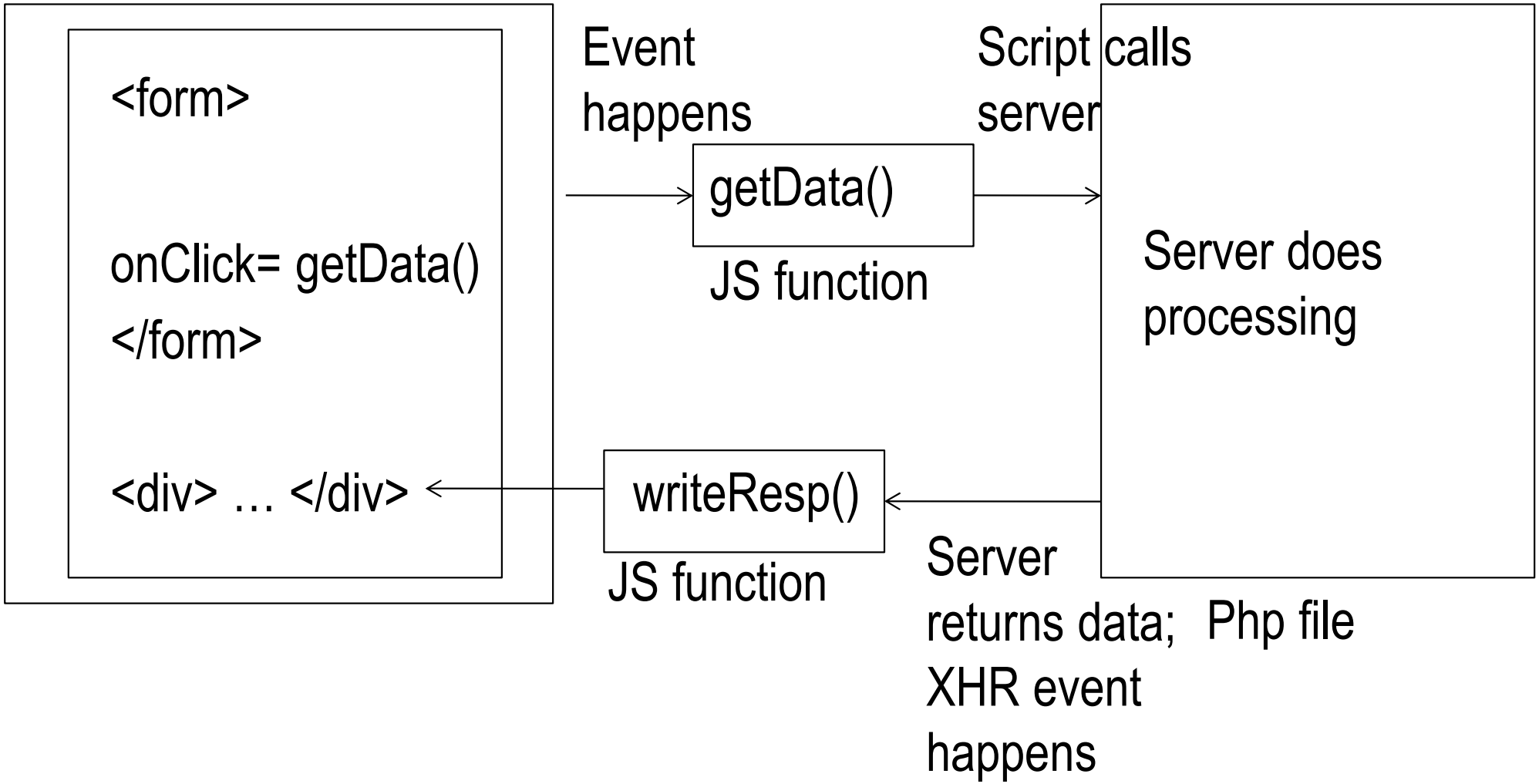


Event-driven processing mechanisms

- The asynchrony, heterogeneity, & inherent loose coupling that characterize modern applications in a wide-area network requires event notification mechanisms.
- Event notification offers a many-to-many communication and integration facility. Clients in an event-notification scheme are of two kinds:
  - objects of interest, which are the producers of notifications, &
  - interested parties, which are the consumers of notifications.
- A client can act as both an object of interest and an interested party. An event notification service typically realizes the publish/subscribe asynchronous messaging scheme.

Ajax event model

the model is event-driven,

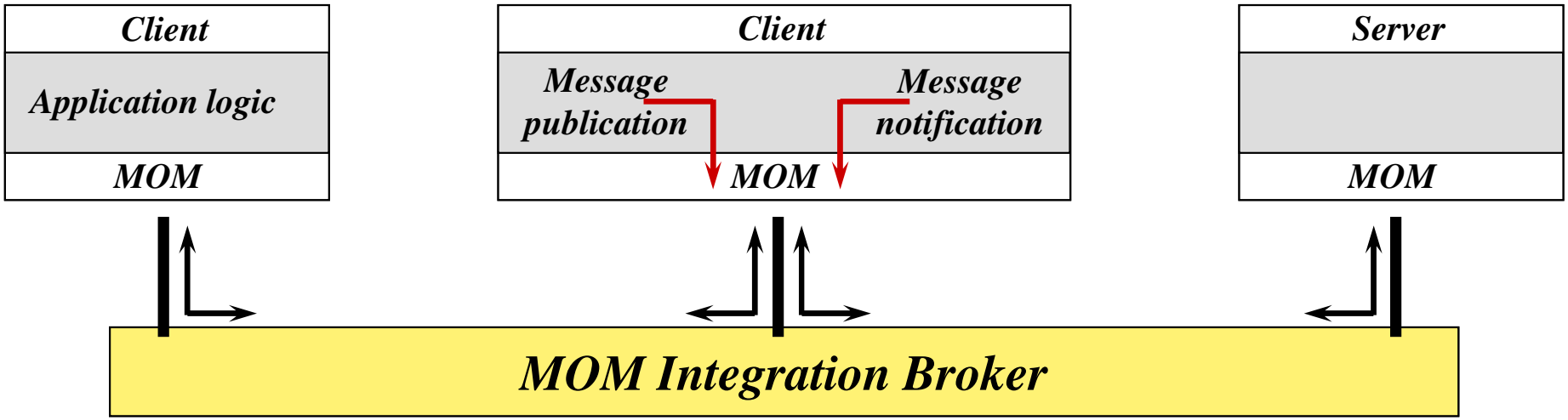


Topics

- *Message Oriented Middleware*

Message-oriented Middleware

- MOM is an infrastructure that involves the passing of data between applications using a common communication channel that carries self-contained messages.
- Messages are sent and received asynchronously.
- The messaging system (*integration broker*) is responsible for managing the connection points between clients & for managing multiple channels of communication between the connection points.



Message-oriented Middleware (cntd)

- MOM provides the following functions:
  - event-driven processing, i.e., the publish/subscribe model.
  - reliability and serialization of messages.
  - subject-based (textual) names and attributes to abstract from physical names and addresses.
  - multiple communications protocols, e.g., store & forward, request/reply, publish/subscribe.
- An integration broker is an application-to-application middleware service capable of one-to-many, many-to-one & many-to-many message distribution.
  - It records & manages the contracts between publishers & subscribers of messages.
- An integration brokers provides the following functions:
  - message transformation, business rules processing, routing services, naming services, adapter services, repository services, events & alerts.

Topics

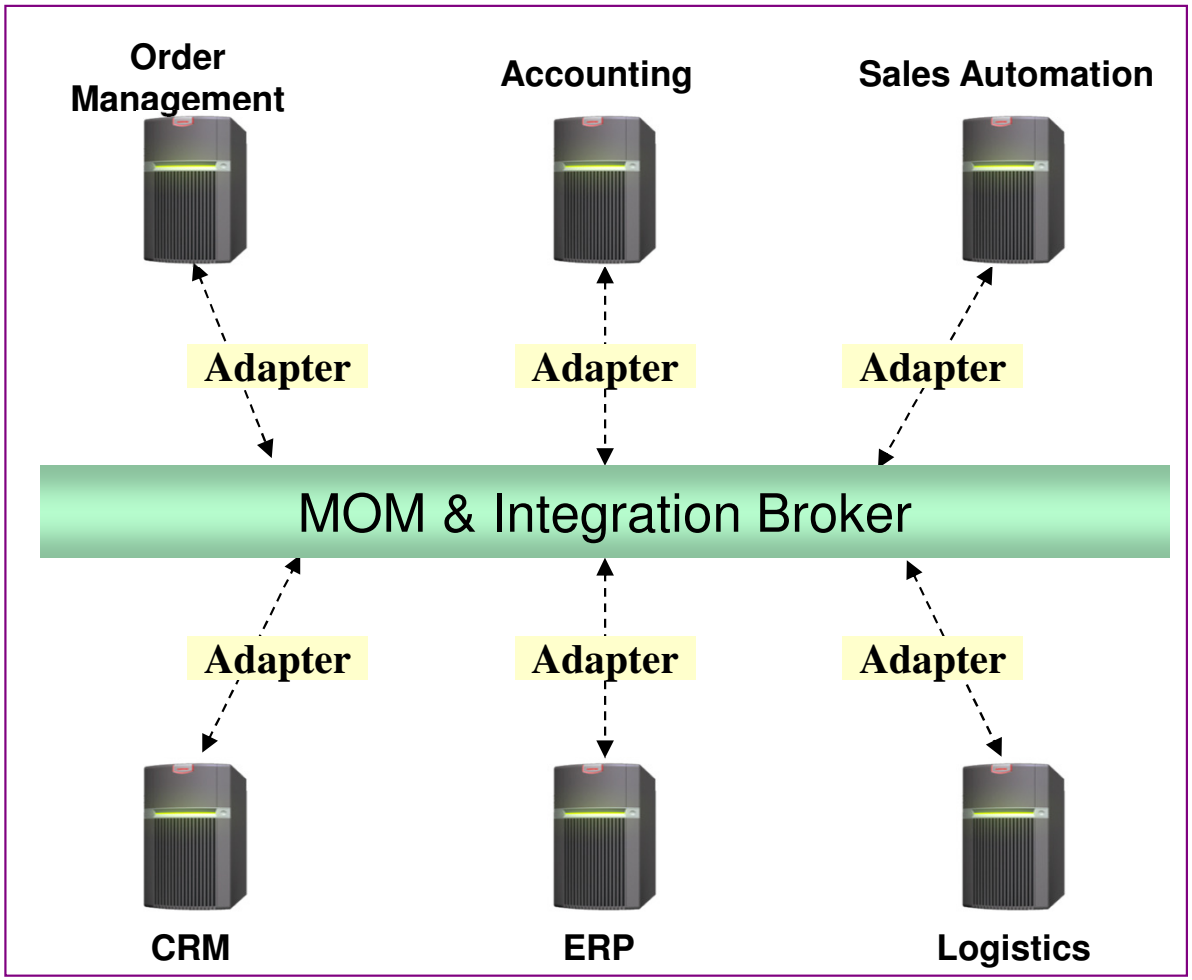
- *Enterprise application & e-Business integration*

Enterprise Application Integration (EAI)

- EAI has emerged to help organizations eliminate islands of data & automation & integrate diverse custom & package applications (including legacy).
- The objective of EAI is to transform an organization's internal applications into a cohesive corporate framework.
- EAI enables applications throughout the enterprise to integrate seamlessly in the form of business processes.
- The internal applications in an enterprise that EAI attempts to integrate are called Enterprise Information Systems. These include:
  - Custom applications
  - Legacy & database applications
  - Enterprise Resource Planning systems
  - Customer Relationship Management systems
  - Transaction systems

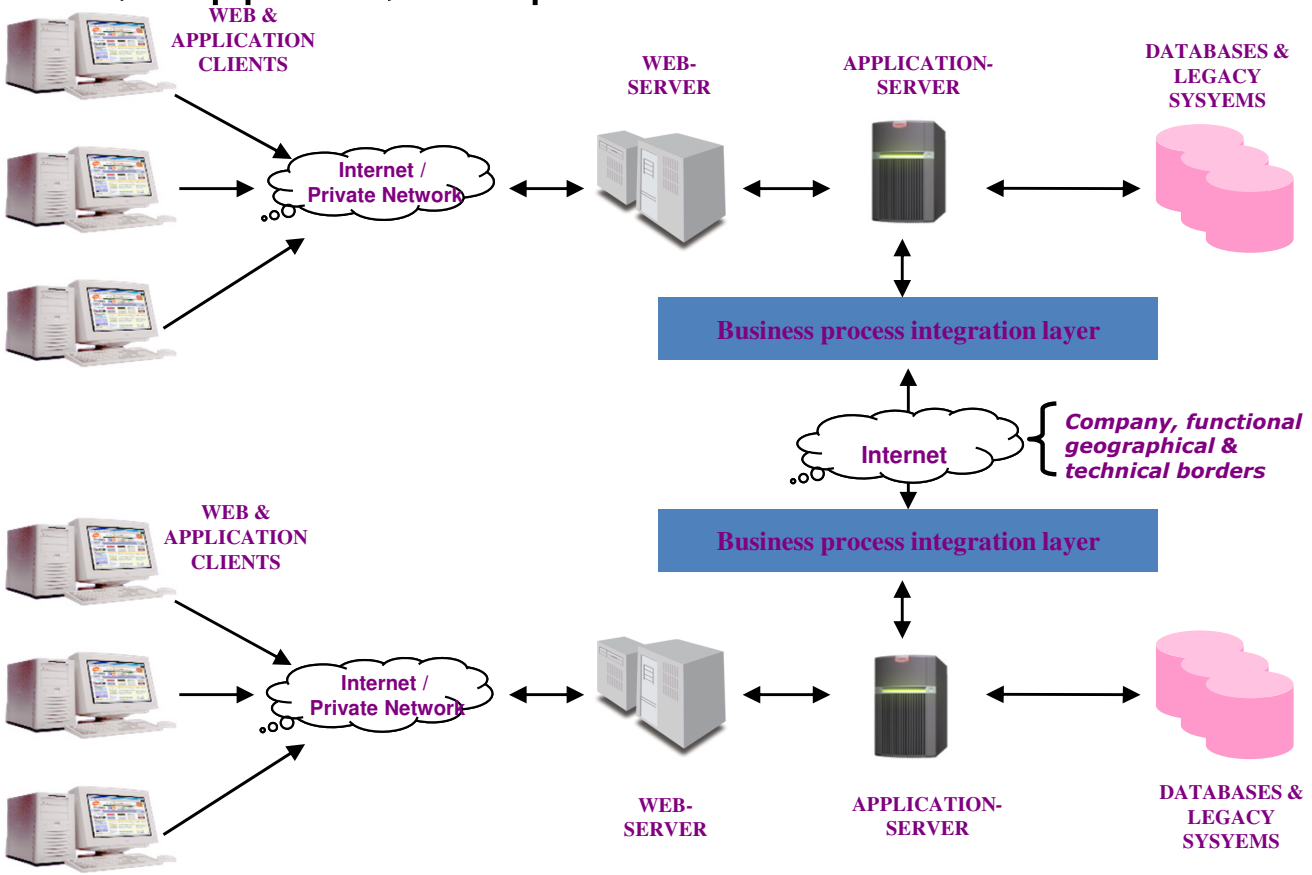
EAI(cntd)

- EAI uses a fast, robust communications backbone with integration broker technology, business process workflow, facilities & tools.
  - Integration brokers are used for message process flow & are responsible for brokering messages exchanged between two or more applications.
  - They provide the ability to
    - transform,
    - store & route messages
    - apply business rules &
    - respond to events.



e-Business integration

- e-Business integration solutions grow on the back of successful internal EAI solutions and provide the capability to link together disparate processes between trading partners.
  - systems internal to an enterprise are able to interact with those of customers, suppliers, and partners.

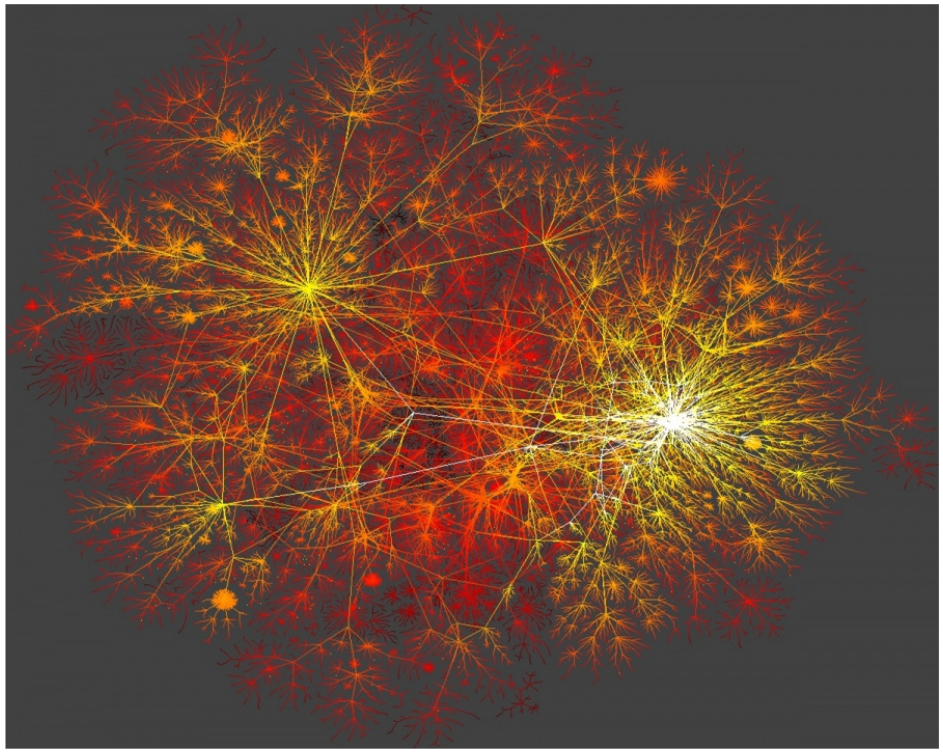


Topics

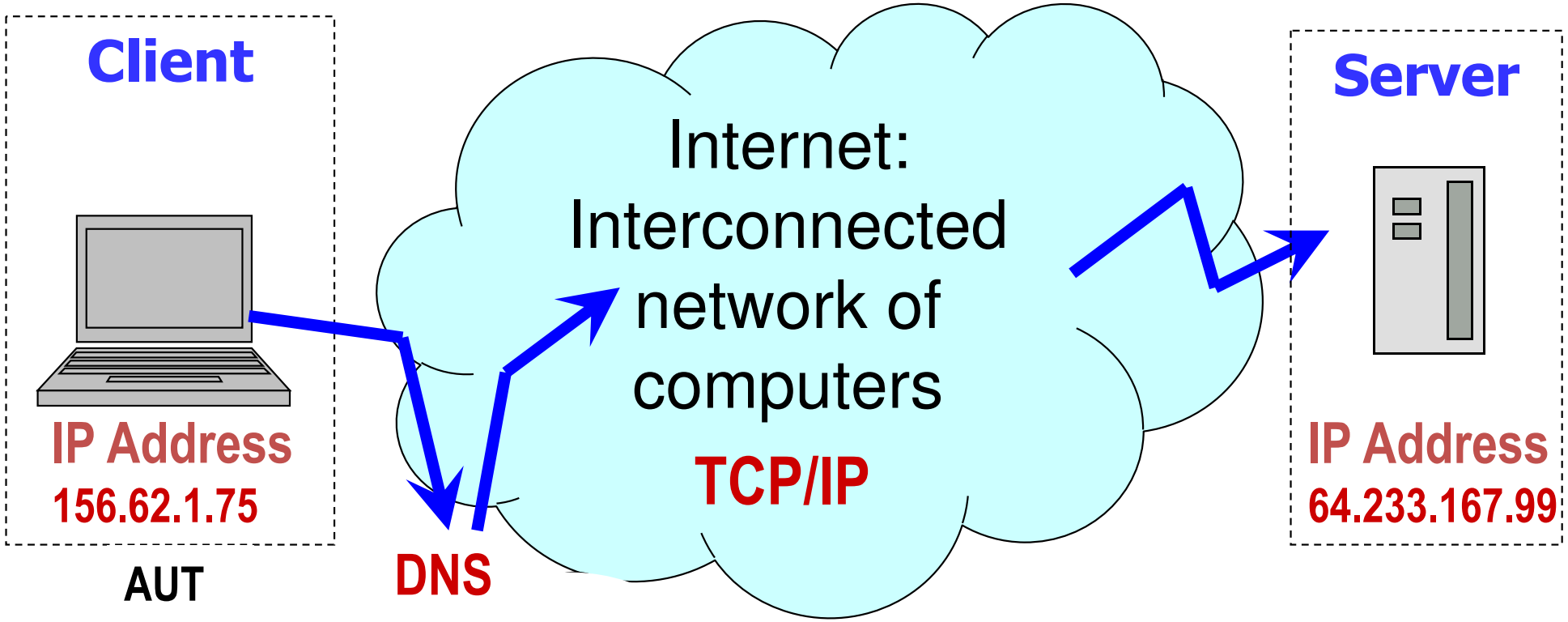
- *Internet and WWW*

Internet and WWW

- Who invented WWW?
- What is WWW?
  - To answer this question, we must know what is Internet?



An overview of Internet

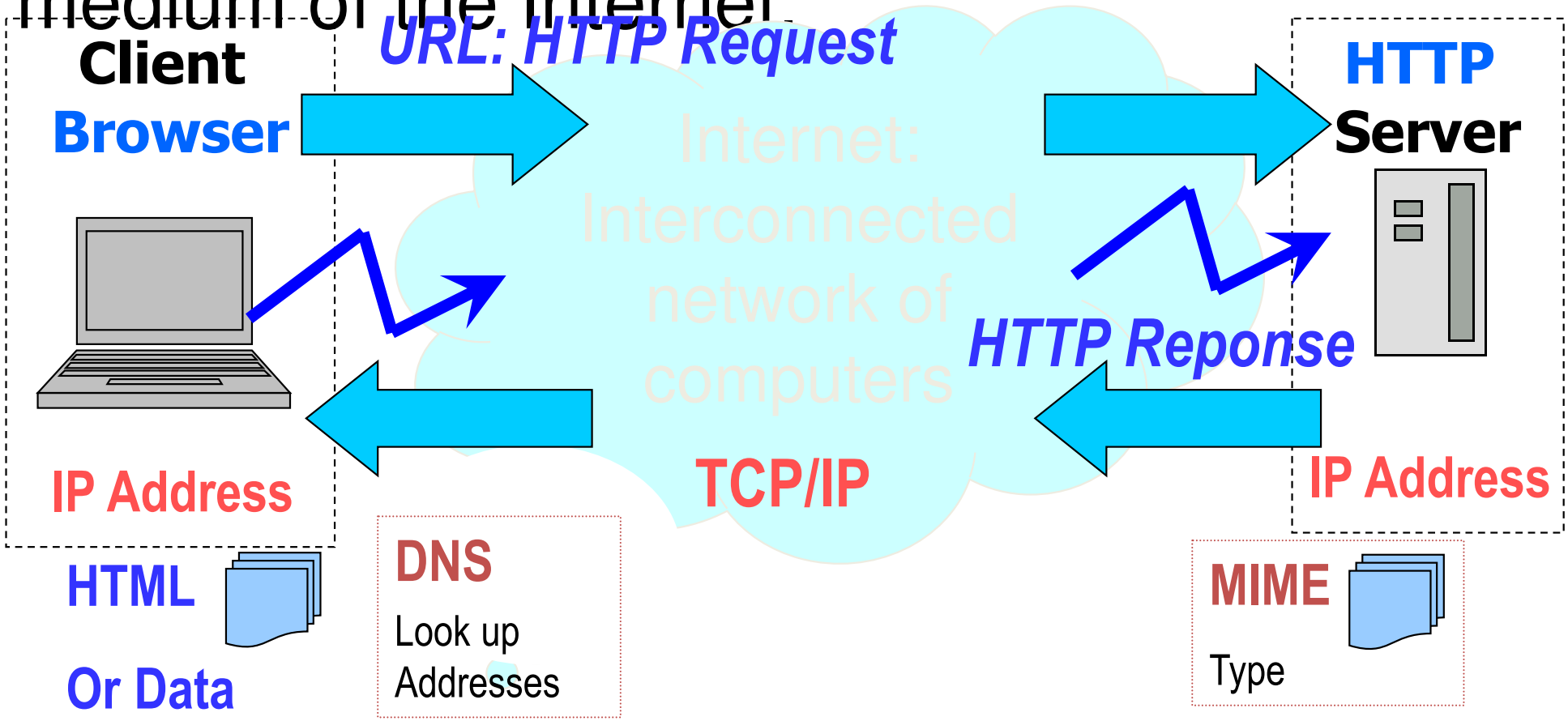


TCP *end-to-end reliable message transfer*

IP *handles the delivery*

The WWW – What is it?

- is a way of accessing information over the medium of the Internet



- Uses the **HTTP Protocol**

The Web – How did it started?

- In 1990 Sir Tim Berners-Lee (European Organization for Nuclear Research) authored a document outlining fundamentals of the web
  - The ability of links to cross machine boundaries (URIs)
  - "A simple, common protocol for exchanging hypertext documents" (HTTP)
  - A common document mark-up language (HTML)

Typical HTTP Methods: GET

- HTTP GET

  Look inside: http://websniffer.cc/

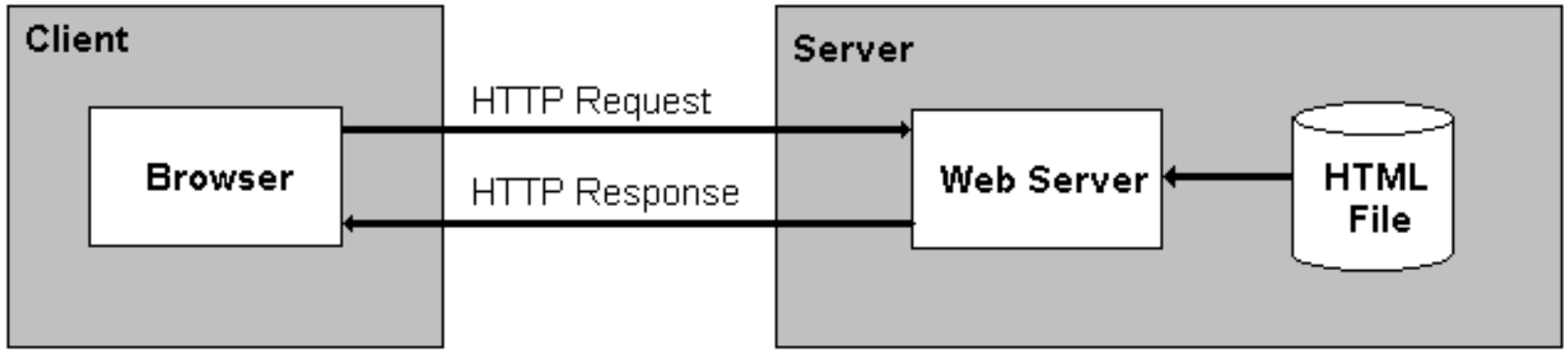
  – A GET request retrieves data from a web server by specifying parameters in the URL portion of the request.

- Typical get request initial line:

```
GET /path/to/file/index.html
HTTP/1.1
```

- Sec

```
Hos
```



View HTTP Request and Response Headers

For more information on HTTP see RFC 2616

HTTP(S)-URL: https://www.aut.ac.nz/

Request type: GET ▾ HTTP version: HTTP/1.1 ▾ User agent: Web-Sniffer ▾

Share This Query

Share this query on a website or a forum by copying the following link:
https://websniffer.cc/?url=https://www.aut.ac.nz/

We have some interesting information about the domain **aut.ac.nz** and the IP addres

- **aut.ac.nz**
- **156.62.238.90**

HTTP Request Header

Connect to **156.62.238.90** on port **443** ... ok

```
GET / HTTP/1.1
User-Agent: WebSniffer/1.0 (+http://websniffer.cc/)
Host: www.aut.ac.nz
Accept: */*
Referer: https://websniffer.cc/
Connection: Close
```

HTTP GET with parameter

https://www.google.co.nz/?gfe_rd=cr&ei=Gp7XVrKmOcbu8wfUqZW4BQ&gws_rd=ssl#q=AUT

Query string: key-value pairs separated by '&'

gfe_rd=cr
ei=Gp7XVrKmOcbu8wfUqZW4BQ
gws_rd=ssl#q=AUT

HTTP Response

- Initial line is status line such as
  - `HTTP/1.0 200 OK`
  - `HTTP/1.0 404 Not Found`
- 200 and 404 are examples of **status codes**
- Message body will contain the requested resource if it was found

Typical HTTP Methods: POST

- HTTP POST
  - E.g., login into Blackboard/gmail
- The POST method is used when you want to send some data to the server, for example, file update, form data, etc.
- Sample post request:

```
POST /path/script.cgi HTTP/1.1
Host: www.google.com
User-Agent: HTTPTool/1.1
Content-Type: application/x-www-form-urlencoded
Content-Length: 32

home=Cosby&favourite+flavour=flies
```

Compare GET and POST

	GET	POST
Restrictions on data length	Yes, when sending data, the GET method adds the data to the URL; and the length of a URL is limited (maximum URL length is 2048 characters)	No restrictions
Restrictions on data type	Only ASCII characters allowed	No restrictions. Binary data is also allowed
Security	GET is less secure compared to POST because data sent is part of the URL	POST is a little safer than GET because the parameters are not stored in browser history or in web server logs
BACK button/Reload	Harmless	Data will be re-submitted (some browser will alert)
Cached	Can be cached	Not cached
Encoding type	application/x-www-form-urlencoded	application/x-www-form-urlencoded or multipart/form-data (for binary data)

HTTP protocol is stateless

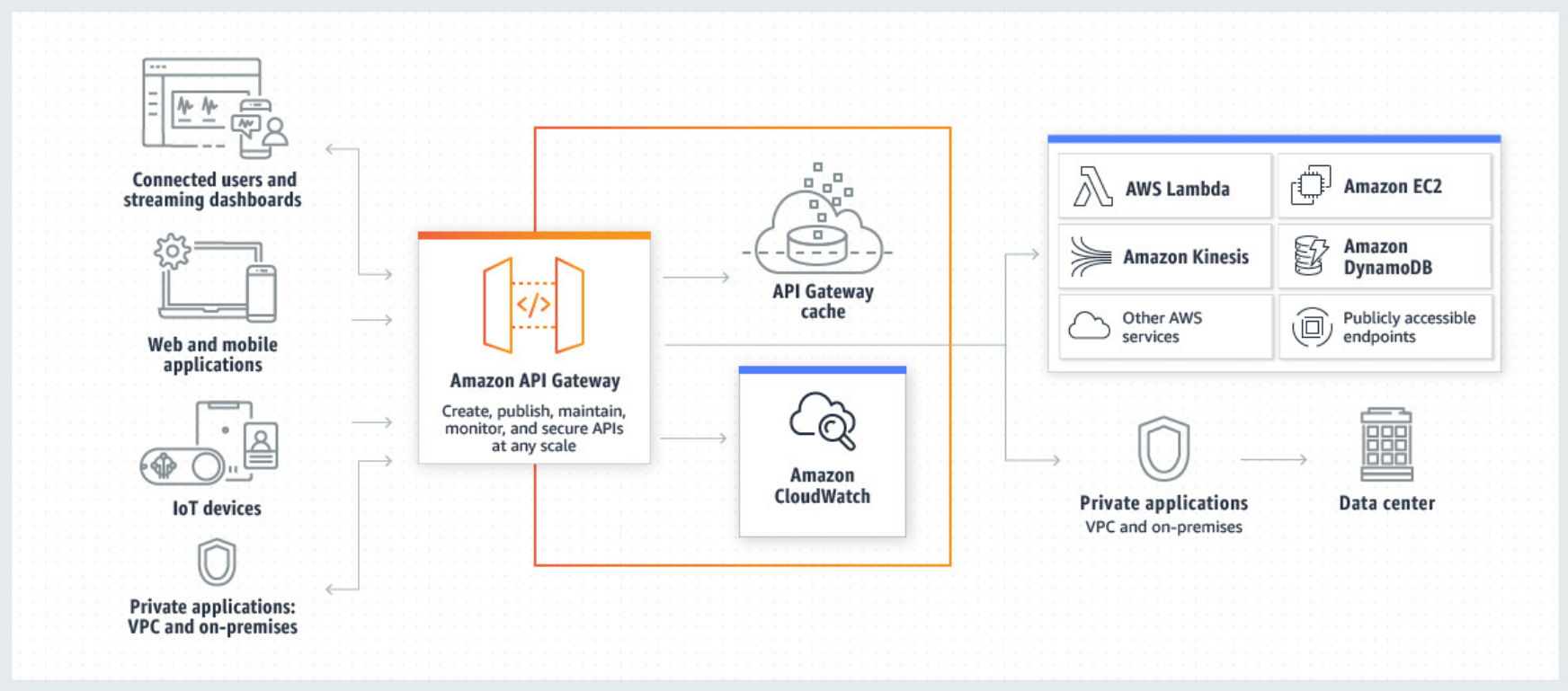
Out of performance concern

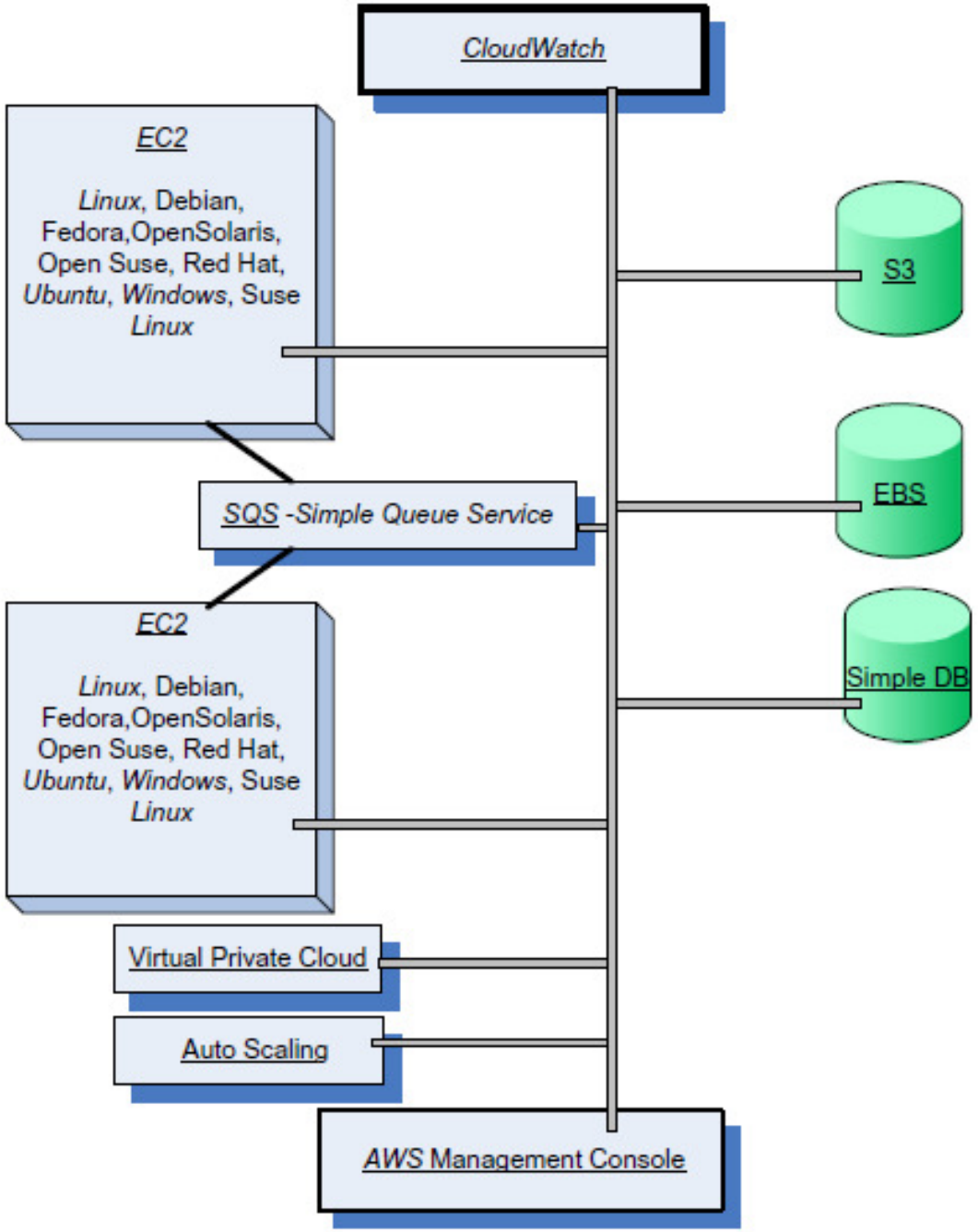
Cloud computing infrastructure at Amazon

Amazon was the first provider of cloud computing; it announced a limited public beta release of its Elastic Computing platform called EC2 in August 2006

Elastic Compute Cloud (EC2) is a Web service with a simple interface for launching instances of an application under several operating systems

A user can interact with EC2 using API Gateway




CloudWatch
EC2
Linux, Debian, Fedora,OpenSolaris, Open Suse, Red Hat, Ubuntu, Windows, Suse Linux
S3
EBS
SQS -Simple Queue Service
Simple DB
EC2
Linux, Debian, Fedora,OpenSolaris, Open Suse, Red Hat, Ubuntu, Windows, Suse Linux
Virtual Private Cloud
Auto Scaling
AWS Management Console

- Simple Queue Service (SQS) is a hosted message queue. SQS is a system for supporting automated workflows; it allows multiple Amazon EC2 instances to coordinate their activities by sending and receiving SQS messages. Any computer connected to the Internet can add or read messages without any installed software or special firewall configurations\

- A virtual private cloud (VPC) is a virtual network dedicated to your AWS account. It is logically isolated from other virtual networks in the AWS Cloud. You can launch your AWS resources, such as Amazon EC2 instances, into your VPC.

- Amazon CloudWatch monitors your Amazon Web Services (AWS) resources and the applications you run on AWS in real time. You can use CloudWatch to collect and track metrics, which are variables you can measure for your resources and applications

- Amazon EC2 Auto Scaling helps you ensure that you have the correct number of Amazon EC2 instances available to handle the load for your application. You create collections of EC2 instances, called Auto Scaling groups. You can specify the minimum number of instances in each Auto Scaling group, and Amazon EC2 Auto Scaling ensures that your group never goes below this size.

References

- Handbook of Cloud Computing, Springer