

“Alexandru Ioan Cuza” University of Iasi  
Faculty of Computer Science



Course  
Master  
I+II

# Integration in Distributed Systems

Lenuța Alboiae  
[adria@info.uaic.ro](mailto:adria@info.uaic.ro)

# Summary

- Integration – as a key concept
- Integration and/or Interoperability
- Integration contexts
- Service Oriented Architecture
- Evolution of Integration Technologies
- ESB - Enterprise Service Bus
- Technology mapping: SOA, ESB, BPM
- Integration Through Cloud Eye
- Integration Roadmap

# Integration

- Continuous Integration [DMG07]
  - coined by Kent Beck, as part of Extreme Programming [Bec04]
  - methodology employed in software development intended to increase software quality and also to adapt dynamically to the client requirements
  - refers actually to a first step in a Continuous Delivery process, which assures that the work-in-progress code belonging to a project is continually integrated with code belonging to others developers from the same project

# Integration

- “connecting computer systems, companies or people” [HW03]
- “Integration services are detailed design and implementation services that link application functionality (custom software or package software) and/or data with each other or with the established or planned IT infrastructure. Specific activities might include project planning, project management, detailed design or implementation of application programming interfaces, Web services, or middleware systems.” [Garl]
- Integration in our discussion has consistent tangents with Enterprise Service Bus (ESB), Integration Platform as a Service (iPaaS) or Data Integration.

# Integration and/or Interoperability

- interoperable systems, there is an environment in which sharing the information is performed without the need of any middleware
  - Issues: a(n) change/upgrades of a party can cause interoperability issues or even the end of communication
- interoperability implies general principles of communication rather than a technical specification
- integration area ensures a full functional systems that provides backwards and forwards compatibility with various versions of the product, a reliable data transfer, automated maintenance et.al.



interoperability offers an exchange and integration offers a full functionality

# Integration contexts

- Integration in a context (e.g. enterprise, applications or services in cloud or on-premise) suffers changes in accordance with the needs which appear in time: new services/extra applications, new technologies, new mechanisms of storage provided by a third party.
- Instruments:
  - ESB (necessary for Application to Application(A2A) or Service Oriented Architecture Systems (SOA)),
  - Business Process Execution Language (BPEL) for process integration
  - Integration platform as a service (iPaaS) (used for A2A or SOA Services in cloud)
  - Internet of Things (IoT Systems - Channels)
  - Application programming interface management (API management) (used for microservices)

# Service Oriented Architecture

- *Service Oriented Architecture* is a design methodology and architecture aimed at maximizing the reuse of multiple services (possibly implemented on different platforms and using multiple programming languages)
  - Services share a formal contract – contains details (and additional metadata) such as their capabilities, interfaces, policies, supported protocols
  - Services must be *loosely coupled*
  - *Abstracting* functionality and technology
  - *Reuse* principle promotes the idea of service as a “repeatable value”. When a business logic is built, reusing it decreases the time of creation
  - Service *Discoverability*
  - Services can be composed to provide other services – *composability*
  - Services are individually useful – they are *autonomous*.
    - =>predictable performance, independence of the service, flexibility, scalability, ability to be easily replaced, fault tolerance and, of course, reuse
  - Service *stateless*

# Service Oriented Architecture

- Open Group SOA Integration Maturity Model (OSIMM) proposed seven levels that should be respected by a SOA application in order to reach “maturity level”
  - E.G. a point-to-point integration is considered an integration service with 2 value (Service Integration Maturity Model - SIMM = 2) or a hub-and spoke architecture corresponds to SIMM =3
- Obs. For small-scale systems, redundancy or even refactoring and consolidation may not be so expensive, for large companies these can spell total failure

# Service Oriented Architecture

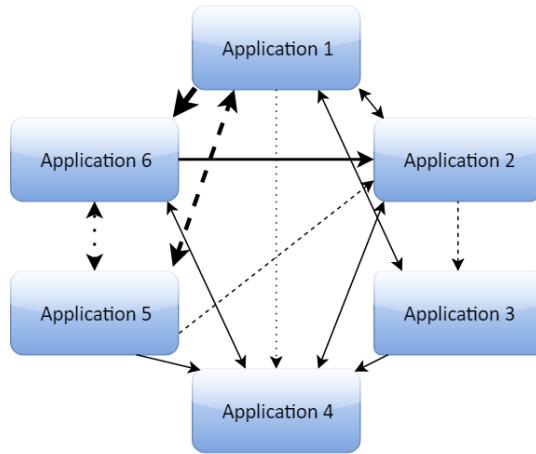


Figure: Point to point integration

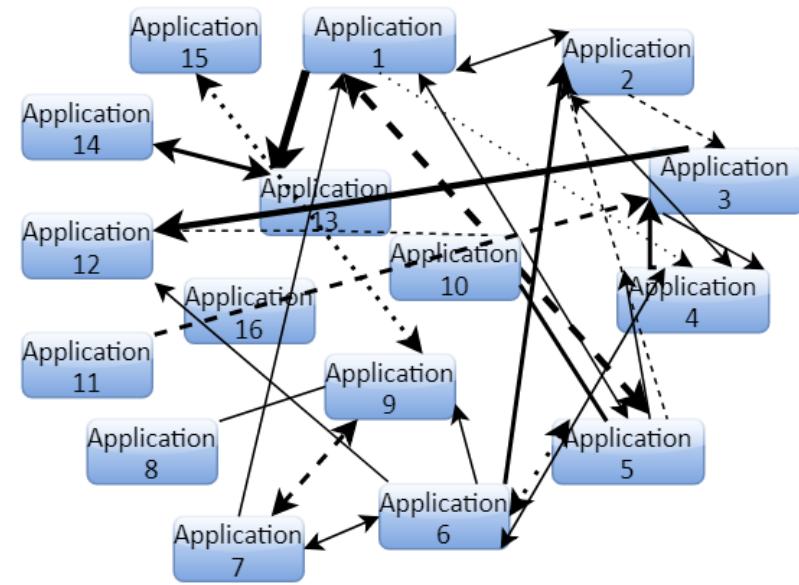


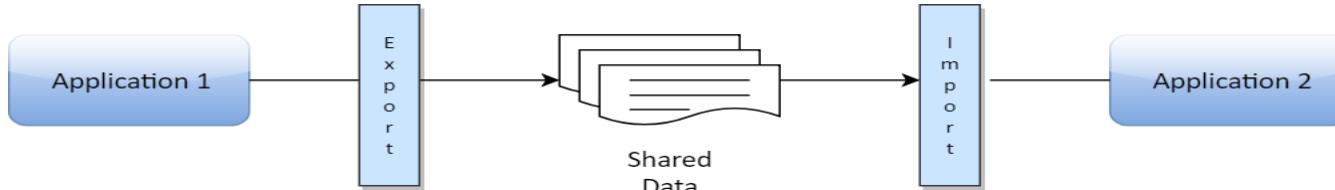
Figure: Point to point explosion in number of connections

# Evolution of Integration Technologies

- In any Enterprise, business processes need an integration layer that allows communication among various applications.
- Example: in an ERP (Enterprise Resource Planning) system, if components such as CRM (Customer Services), Human Resource, Production (PLM), Sales et.al. cannot communicate, than internal processes cannot be automated, ultimately leading to inefficiency. Therefore, the integration solutions for an Enterprise are known as Enterprise Application Integration [Cla15].
- Mechanisms for integration have been proposed [HW03]:
  - *File Transfer*
  - *Shared database*
  - *Remote procedure invocation scenario*
  - *Integration through messages*

# Evolution of Integration Technologies

**File Transfer:** an application A produces and exports a file that is imported and processed by a B application



In this scenario, there are some issues that should be discussed:

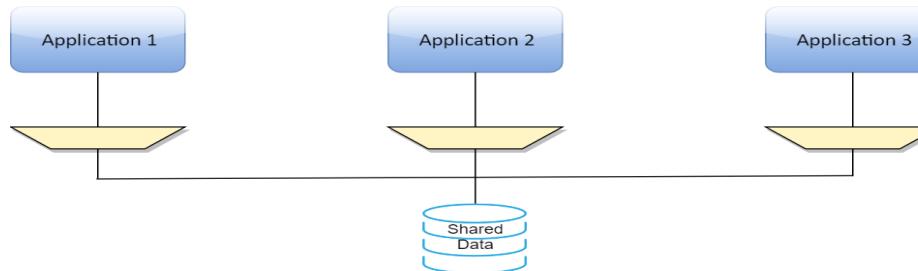
- when it is the "right" moment to export/import the file, which also implies the transfer frequency. Whenever you have to deal with a large amount of data, some processing power is needed making such a decision not so obvious.
- there should be a convention in place regarding the file name, its unicity and its format (XML, JSON)
- there should be rules regarding "time-of-life" for a file and when it is the right moment to delete/update
- there may be situations in the B application when multiple components need some parts of the imported file. In this case, some synchronization mechanisms are required.

All these aspects imply that a massive amount of work is required from developers; there are some benefits linked to this method:

- No external integration tools are necessary
- Applications are decoupled: changes inside the applications are allowed, on condition the data is provided in the same format
- The integrators do not need to know details regarding applications' architecture or implementation details

# Evolution of Integration Technologies

**Shared database:** multiple applications store data in a shared database



The benefits of this approach are:

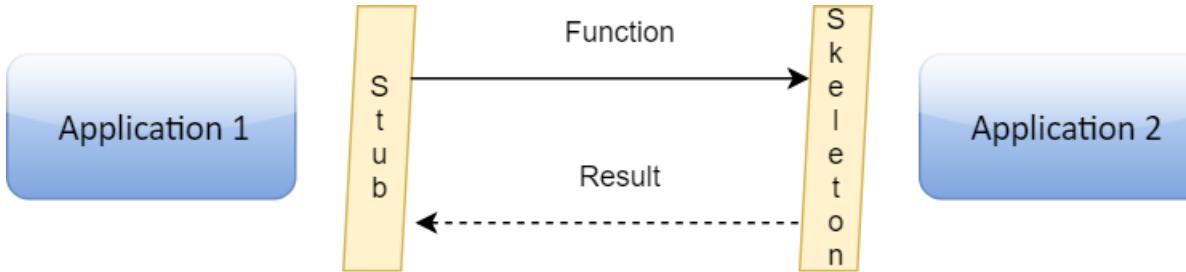
- Inconsistencies due to faulty synchronisation mechanisms are avoided through database transactions
- Semantic dissonance (data appears to be the same but has different meaning) is avoided through the existence of data schema for SQL-based relational databases;

There are, however, some issues that may affect an Enterprise's business process:

- Designing a database for multiple applications is difficult
- This architecture is prone to bottlenecks (if many reads/writes operations are needed) and deadlocks (if an application performs locks, the rest will not be able to perform operations on data)
- The system extension (what happens when a new Application is needed?)
- A change in one application determines changes in the database affecting the other applications. Therefore, in this model *unencapsulated data* [HW03] are a drawback to maintain or integrate future applications.

# Evolution of Integration Technologies

- **Remote Procedure Invocation:** is based on the encapsulation principle and interfaces in order to assure integration



This approach has advantages such as:

- Due to encapsulation, applications can expose multiple interfaces to the same data. Thus, a 1 to n type relation with the clients is possible with this model
- From the implementation point of view, developers are accustomed with procedure calls and there are many standards for RPC (Remote Procedure Call): CORBA, COM(Component Object Model), .NET Remoting, Java RMI. Also, Web Services [AB06] with associated standards comply with this paradigm. Usually, these are HTTP, presenting more flexibility from the communication perspective (e.g. easy to get through firewalls).

Some issues brought by this approach are linked to some fundamental factors that should be considered for integration:

- the main weakness of this model is determined by the fact that remote calls are slower and packages can be lost in networks. Furthermore, if there are multiple applications communicating through RPC mechanism, it is possible for one failure to trigger failures in cascade.
- applications are somehow tightly coupled, because they should know details about their interfaces; any change determines other changes

# Evolution of Integration Technologies

- **Messaging** allows “to transfer packets of data frequently, immediately, reliably, and asynchronously, using customizable formats” [HW03].

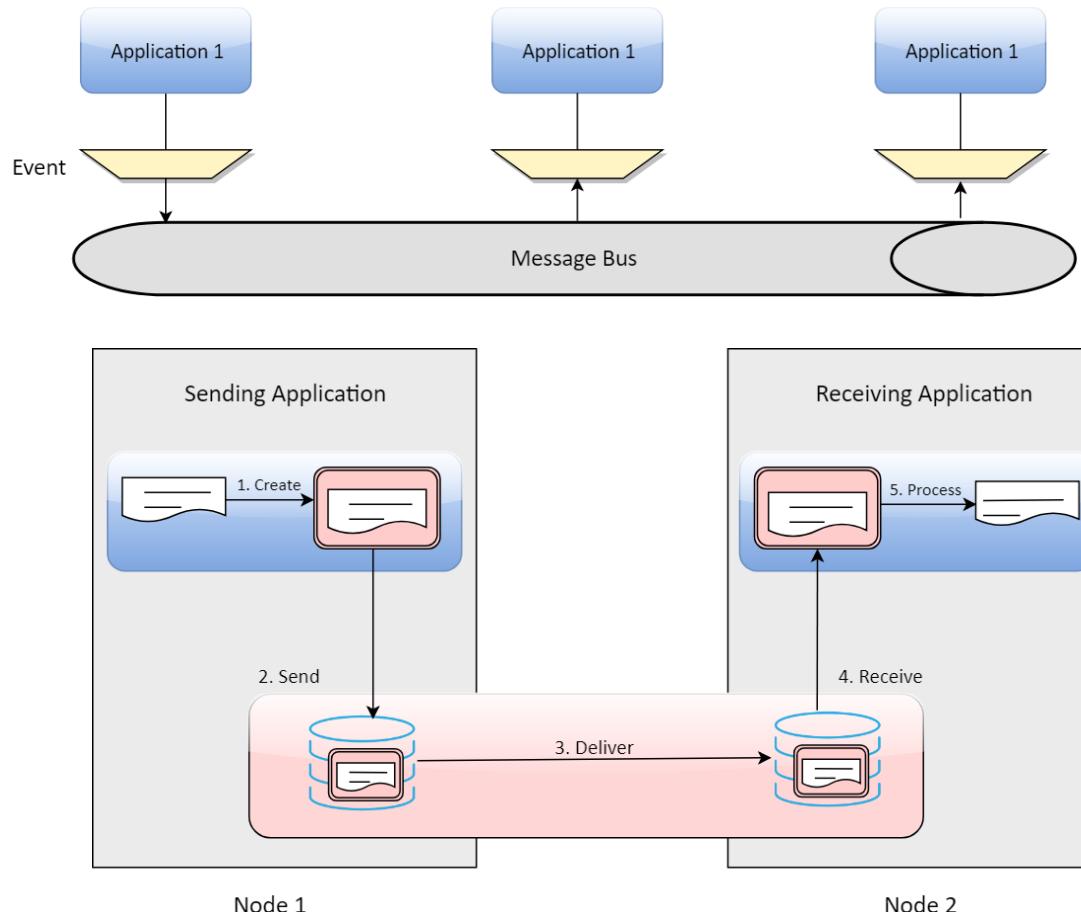


Figure: Integration through messages

# Evolution of Integration Technologies

***Messaging Systems*** or Message-oriented middleware(MOM) advantages

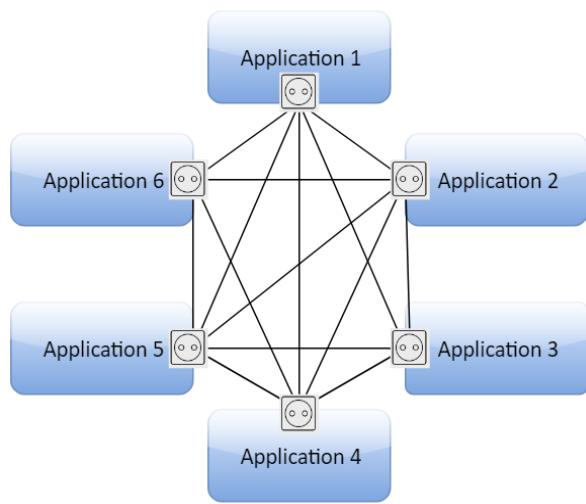
- the asynchronous mechanism changes the approach when you plan to design distributed applications (e.g. the communication - the applications may not be up in the same moment, the remote call can take some time et.al.)
- decoupling from File Transfer is achieved by the fact that messages can be transformed, broadcasted to various receivers without the transmitter and recipient knowing it; therefore the integration level is separated by the application development

## Issues

- event messages are sent fast while we still have the networks lagging, therefore some updates may not perform quite simultaneously
- testing and debugging in an asynchronous environment is not an easy feat
- some “glue code” in usual cases cannot be avoided in order to fit everything
- from the programmatic point of view, the employed mechanism is not familiar to the programmers any more, so there is an increased level of complexity.

# Evolution of Integration Technologies

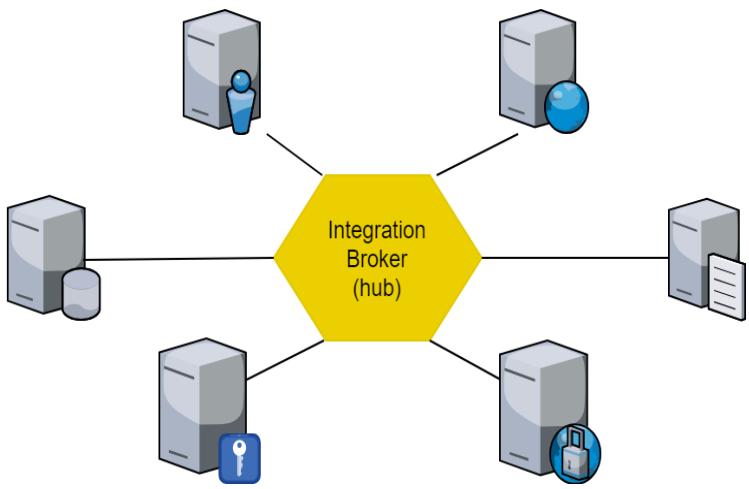
- All these solutions have been used to ensure integration and interoperability between disparate systems of an Enterprise, therefore we reach the Enterprise Application Integration.
- Topologies that were used in EAI started from *Point-to-Point Integration, Integration Broker model* and evolved to Enterprise Service Bus solutions.
- In **Point-to-Point Integration** or **Vertical integration** [SW93] manner, there are special components (called connectors) that are in front of the application intended to interoperate with others. This connector is implemented for each application pair. This approach is also known as *hub-and-spoke* topology.



- No scalability (each adapter/connector is bundled to a specific data format or transmission mechanism, any attempt to add new applications implies individual implementation of a connector)
- Reusing some parts to create new services is not possible
- This mesh network implies a huge maintenance effort

# Evolution of Integration Technologies

- **Integration Broker** or **Star Integration** model comes with a central engine/central hub or broker that provides message flow, message transformation or message routing.
- In comparison with point-to-point, this model performs decently in case of loose coupling (e.g. there may be situations when the sender does not know the receiver's endpoint) and asynchronous communication is encouraged
- The broker manages any message format, therefore from the application perspective, there is no need for a canonical format.



- Having a central point is obviously leading to scaling problems. The system can maintain its state in a shared database and it is possible to scale if the central hub is for example clustered – but is not a flexible architectural alternative
- Heavy load of transactions can lead to bottleneck
- Difficulty of implementation and administration of a central hub, in case you intend to integrate products belonging to different companies (heterogeneity).

# Evolution of Integration Technologies

- Taking all these drawbacks into account, it was necessary for a system to offer more functionality than previously mentioned (e.g. transaction security, error handling mechanisms) or to entirely avoid the issues by a central integration logic, and therefore Enterprise Service Bus or Horizontal Integration [SW93] have become reality.
- Service bus is different from message buses because it supports patterns such as publish/subscribe, request/reaction, request/reply and also SOA concepts like contracts. Examples of such service buses are: Apache CXF, Apache Camel, Windows Azure service bus, MuleESB, WSO2, Talend ESB, SwarmESB (UAIC)

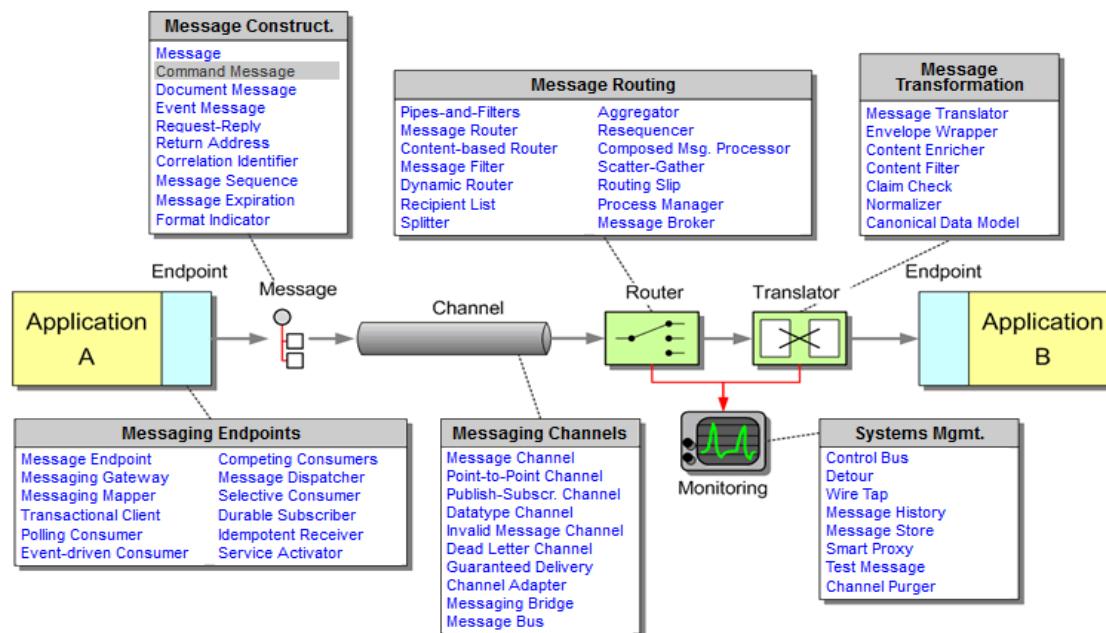


Figure: Messaging patterns [HW03]

# Evolution of Integration Technologies

Example of such patterns:

- *Message Channel* represents a logical channel among applications where (one writes and the other one reads that piece of information)
- *Message Bus pattern* is a “connecting middleware” between applications similar to a communication bus in a computer that allows communication between memory, I/O devices and CPU.
- In a *Message Broker pattern*, the received messages from multiple senders reach a central point that controls the flow and route messages to the correct channel. Using a MessageBroker as central point leads us to the *hub-and-spoke* architectural style.
- *Messaging Endpoint* contains code for application and messaging system. Therefore, applications know little about the message path, format or underlying communication details.
- For an *Event-Driven Consumer* or asynchronous receiver, receiving a message causes/triggers the receiver to perform an action. For example RabbitMQ consumers are using this event-driven pattern.
- *Control Bus* uses the same messaging mechanism used by applications, but separate channels are used for management control messages.
- *Message Sequence* allows transmitting large amounts of data in form of chunk data that have a unique identification field.

# Evolution of Integration Technologies

Example of such patterns:

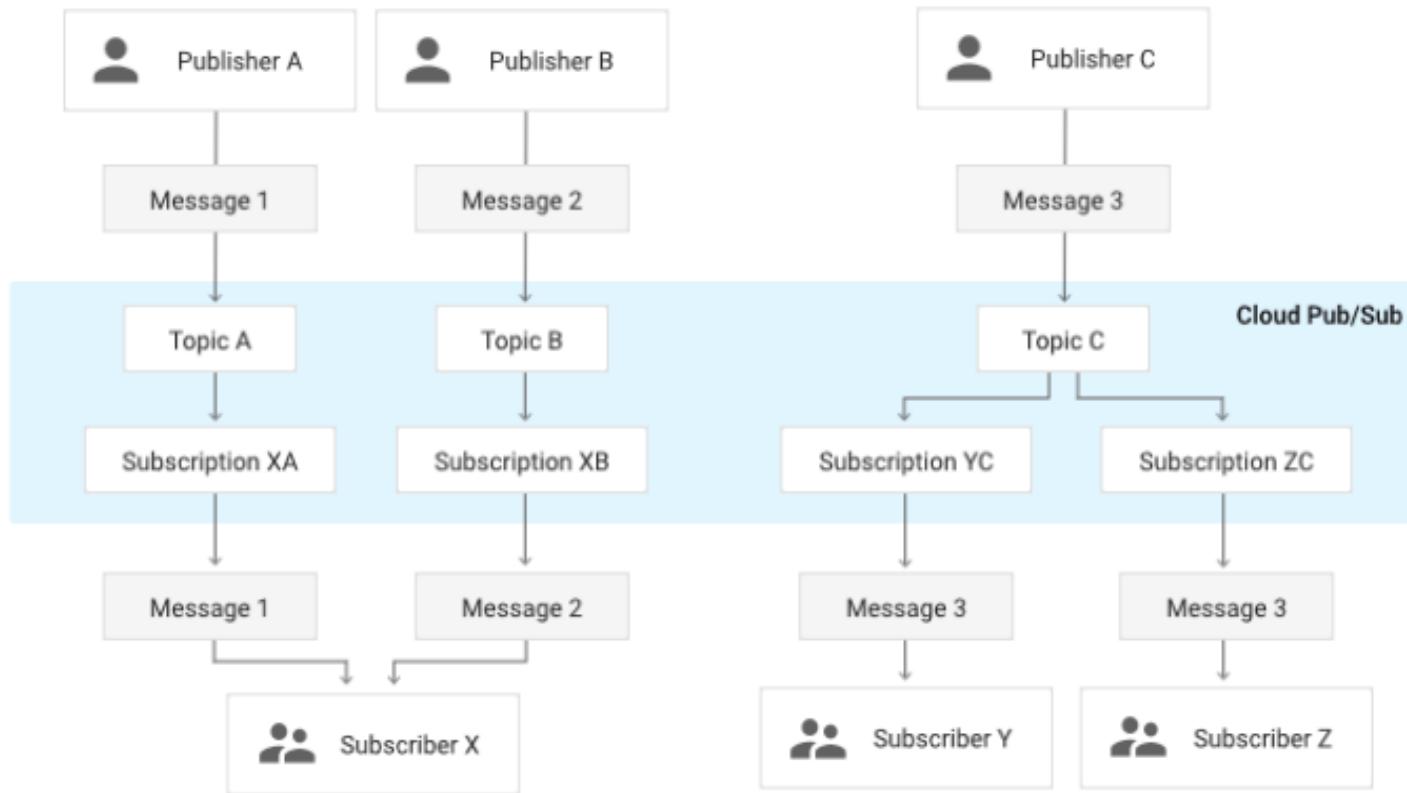


Figure: Google Cloud Pub/Sub [GooglePS]

# Evolution of Integration Technologies

- In a *Publish-Subscriber pattern*, one input channel can split in multiple output channels that has just one corresponding subscriber that consumes a message, which will be deleted. There are many flavors of Publish-Subscriber pattern, for example in Google Cloud Pub/Sub, they offer this mechanism but there are also features as Topics and Subscriptions [GooglePS].
- An application (the sender) publisher sends a message to a *topic*. The receiver creates a *subscription* to a topic in order to receive messages.

The detailed steps, from an implementation point of view are:

- Sender creates a topic in Google Pub/Sub service and sends messages to that topic; a message has a payload and attributes that characterize the payload; the messages are stored persistently until an acknowledgment is received
- The Pub/Sub service forwards the message to all subscriptions, individually, irrespective of the mechanism used by the receiver ( Pub/Sub *pushing* operation, or receiver/subscriber *pulling*)
- The subscriber sends acknowledgment for the received message and the Pub/Sub will delete the message

For Google Cloud Pub/Sub this asynchronous communication that decouples senders and receivers, can be one-to-many (fan-out), many-to-one (fan-in), and many-to-many. In [GooglePS], some basic uses of this Pub/Sub pattern are depicted: balancing workloads in network clusters (multiple tasks can be distributed among multiple Google Compute Engine instances) or logging to multiple systems (an instance can write logs in a database, these logs can be interrogated latter by other applications).

# Evolution of Integration Technologies

Example of such patterns:

- A *Message Router* consumes a message from one MessageChannel, without changing the content, and sends MessageChannel to other(s)
- *Pipe-and-Filters pattern* allows receiving messages, performing various operations on it (filtering operations such as: decryption, deduplication et.al.) and these steps are connected through pipes
- *Command Messages* is a pattern that allows to invoke a procedure in another application. In SOAP protocol [AB06], a request SOAP message can be considered a Command Message
- *Envelope Wrapper* pattern performs an encapsulation of application data in an envelope that is compliant with the messaging infrastructure. A good example of this pattern use is SOAP messages

**ESB also includes the functionality of a bus service and may well constitute the base for a Service Oriented Architecture (SOA). It provides fundamental services to develop more complex architectures, for complexity hiding, data access streamlining, routing messages, transforming data, executing transactions, orchestrating services or enabling “publish-subscribe” mechanisms, etc. Usually, ESB functionalities are packaged as single products**

# Service Oriented Architecture

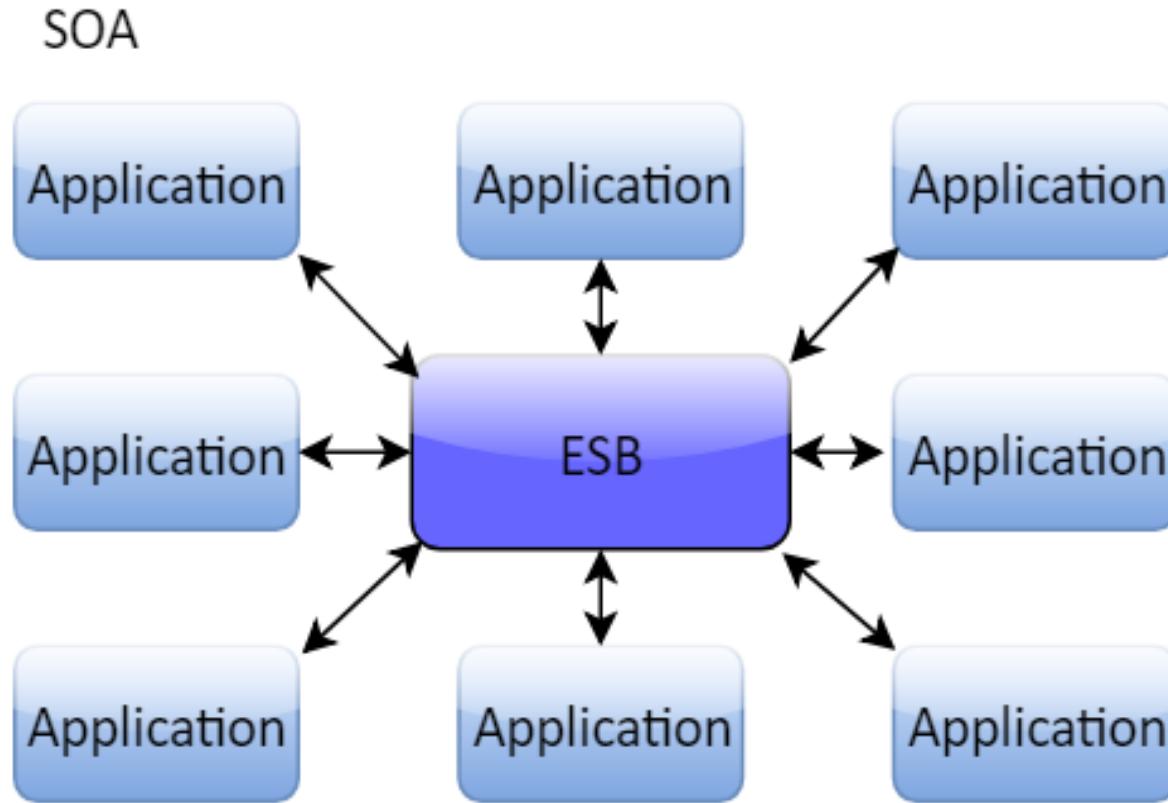
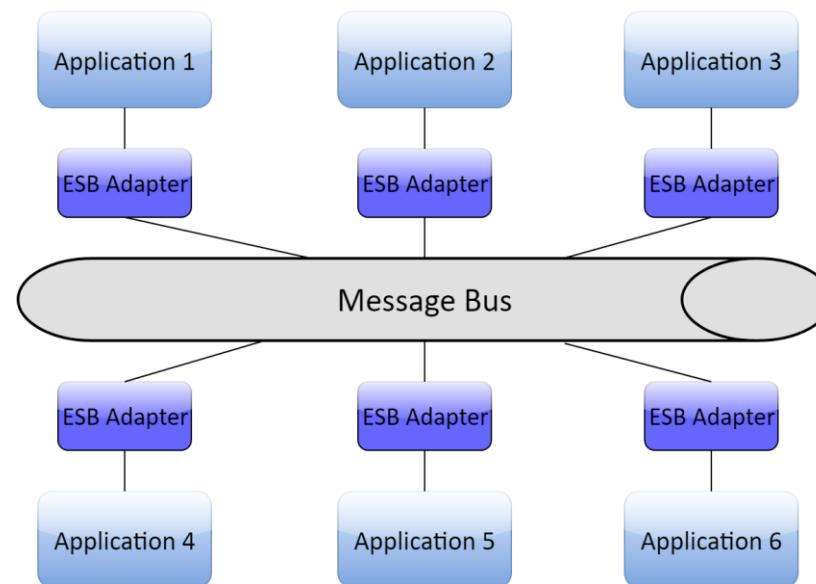


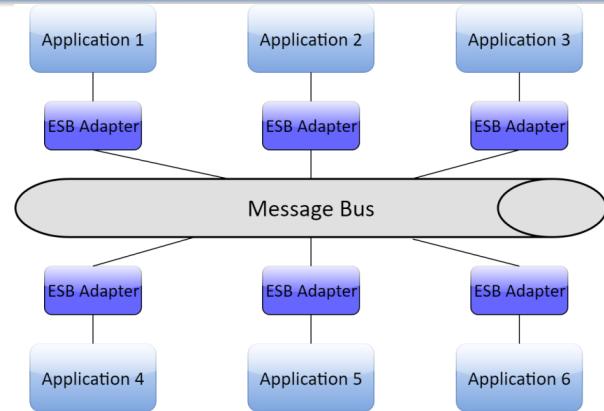
Figure: Using ESB in a SOA architecture

# ESB - Enterprise Service Bus

- The notion of “Enterprise Service Bus” was introduced by Gartner Group in 2002 and thoroughly presented in the book “The Enterprise Service Bus” by David Chappell [CHA09].
- ESB has the same goal like the previously integration models and represents a collection of principles, rules and features that create the environment for an architectural layer providing a bus based integration among various applications/services



# ESB - Enterprise Service Bus



- Each application has an adapter that assures communication among applications using bus functionalities.
- The adapter in an ESB is a “smart” one and includes many features: transformation, marshalling/unmarshalling, security et.al. From this point of view an ESB is different from a *broker model*.
- Therefore, in an ESB there is a good level of decoupling, applications being able to communicate without having knowledge about others on the bus or about particular characteristics of one application (e.g. message format et.al.)
- In an implementation, a service bus can be deployed in multiple ways [RD08]

# ESB

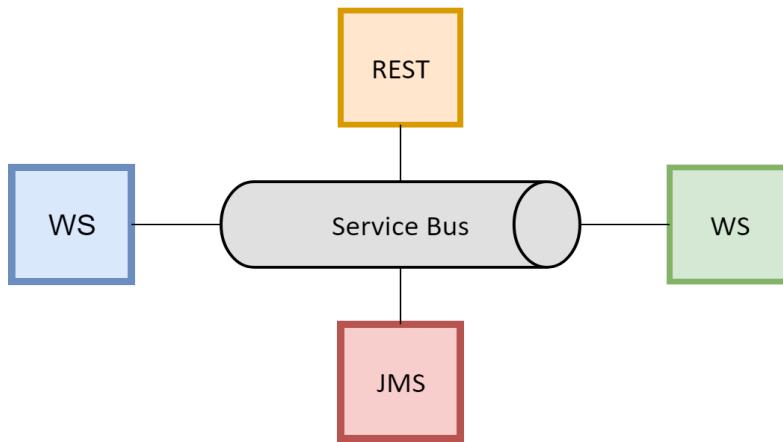


Figure: ESB as a coordinator for services integration

- The model from Figure resembles point-to-point or hub-and-spoke model, but in an ESB implementation, the adapters are smart connectors. An adapter uses two queues for inbound/outbound messages and offers many features (e.g. transformation, security et.al.).
- The older integration models used in EAI, were implemented as a monolithic stack but in an ESB we have coordinated service interaction.
- This model is easy to manage (program, debug) but the bus is bottleneck and we have limited scalability (just scale up solution is possible in this case).

# ESB

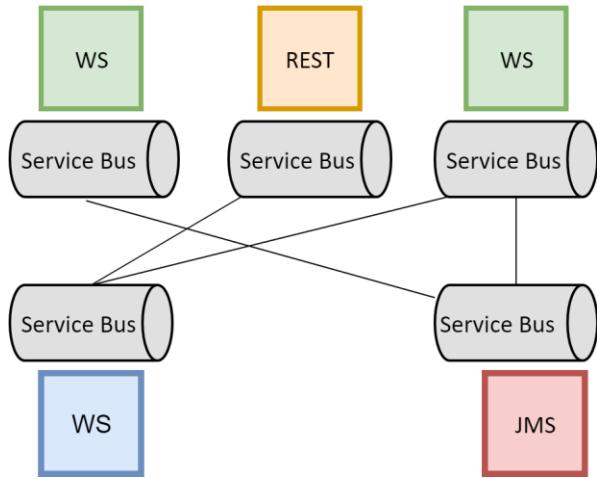


Figure: A P2P network of ESBs  
In this case services have their own bus, and these are connected in a peer-to-peer network.

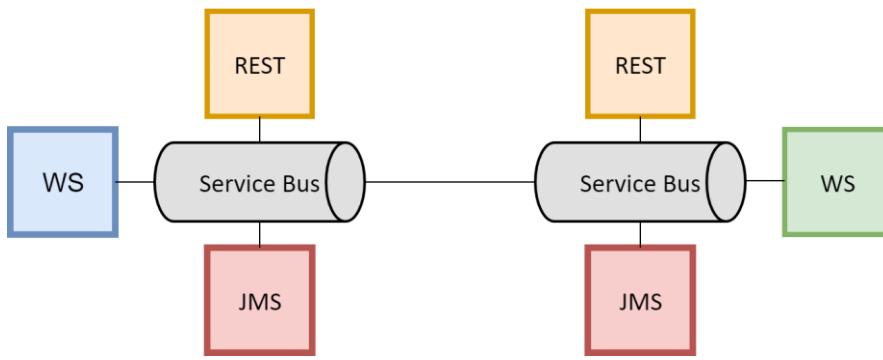


Figure: ESB in a Federated setup  
a service bus serves a limited number of services and multiple buses are organized in a “federated setup”. Even a service is tight to one bus, it can communicate with services connected to other buses from the federation.

Obs. In both cases we have scalability but high complexity for configure and debug.

# ESB Functionalities

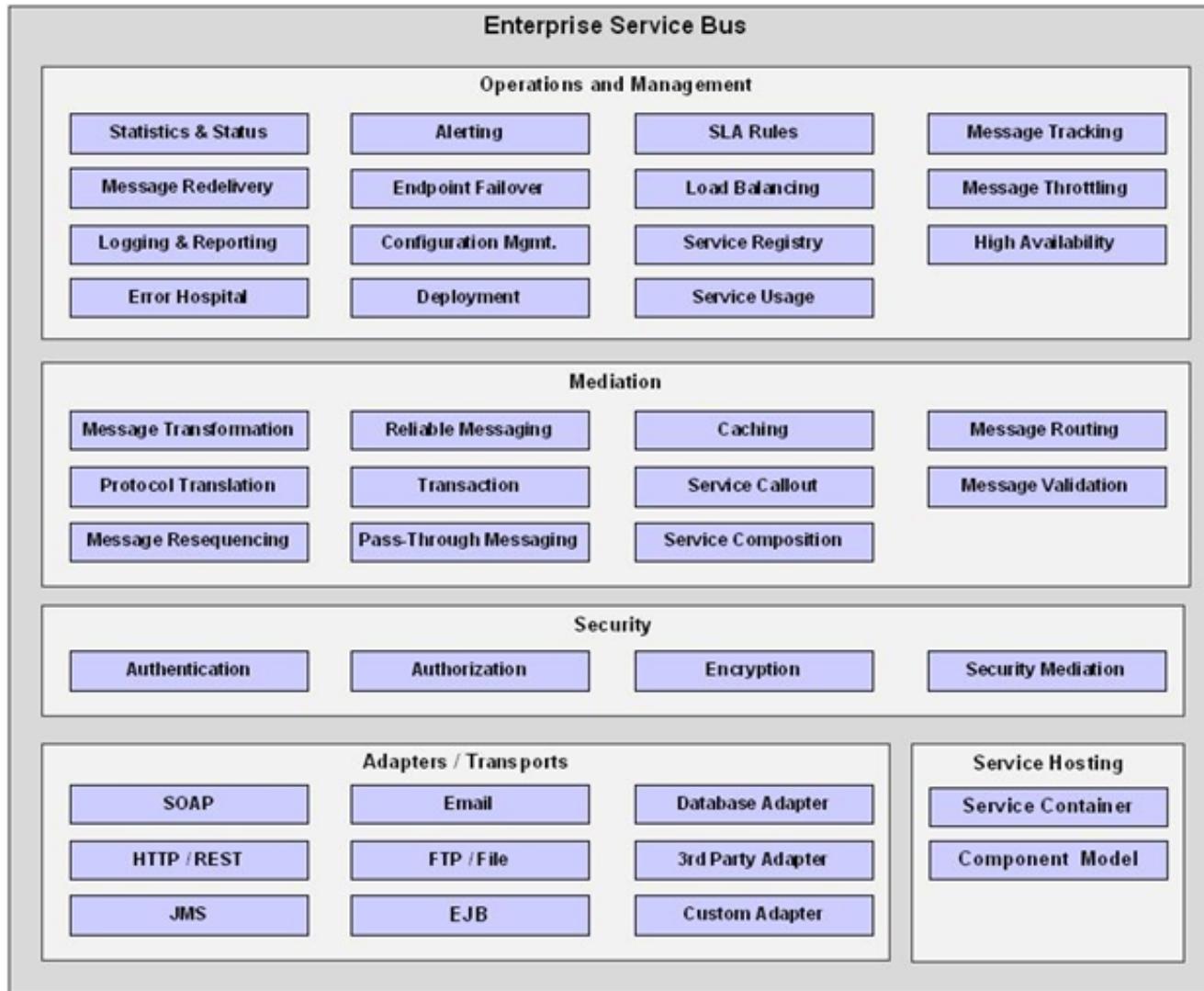


Figure: Functions of an ESB [KMNSST13]

# ESB Functionalities

## Operations and Management

The services from this category allow reliable operations and management of an ESB.

- *Statistics & Status* provides ESB services' statistics such as: the number of system errors, the minimum and maximum response time or statistics link to the number of processed messages.
- *Alerting* provides a mechanism to send warning messages through different channels.
- *SLA Rules* represent rules that may be conceived based on the data provided by the running *Statistics & Status* components. This enables SLA (Service Level Agreement) monitoring and measuring. Any SLA breach is communicated using the warning component.
- *Message tracking* enables the possibility to follow the messages sent by ESB and it should be activated whenever required to reduce the system load.
- *Message Redelivery* ensures that the messages not immediately processed are re-sent automatically after a previously defined period of time. The number of tries and the interval between them are also configurable.
- *Endpoint Failover* enables the possibility to specify a service provider to be called automatically in case the main provider is not available.
- *Load Balancing* is a component enabling load distribution to several equivalent services providers.

# ESB Functionalities

## Operations and Management

- *Message Throttling* is a component allowing to set the maximum messages load to be sent to a specific service provider on a given amount of time. This manner, the overloading of service provider and ESB waiting queue is avoided. This component enables the prioritisation of messages, in order to process first the higher priority messages.
- *Logging & Reporting* enables the messages' recording, in order to be displayed later. It may also provide audit functions.
- *Configuration Management* makes it possible to modify a running ESB configuration without loss of integrity. A history log of these changes may be stored in order to reset a service to a previous state, at any given moment.
- *Service Registry* enables recording and management of an ESB services.
- *High Availability* ensures that the services provided by the ESB run safely regardless of the state of the server they are running on.
- *Error Hospital* is the destination of messages unable to be processed after multiple tries. Here, they may be visualized, corrected if necessary and reprocessed.
- *Deployment* provides the possibility to install an automated service on ESB.
- *Service Usage* monitors and registers the services used by the user, in order to be paid. It's similar with the *metered billing* used in Cloud Computing.

# ESB Functionalities

## Mediation

The services from this category correspond to operational components employed for the implementation of messages' flux in an ESB.

- *Message Routing* enables the messages to be routed to a certain service depending on their content. It may also split a message so it can be sent to several services, or it may combine various messages in a single one.
- *Message Transformation* enables the conversion of a message from one format to another (e.g. text in binary, CSV in XML/JSON etc.).
- *Service Callout* enables the possibility to access other services from an ESB messages flux. A service may be a Web Service, but the ESB may call directly a code locally installed on the ESB, such as a method of a Java class.
- *Reliable Messaging* supports message transfer using queues or WS-\* standards, such as WS-ReliableMessaging.
- *Protocol Translation* enables the transition from one communication protocol to another, such as TCP/IP to HTTP or HTTP to WS.
- *Message Validation* ensures that the messages are valid. In case of a message in XML format, this means it verifies that the message is well formed and complies with a XML schema or with a WSDL [AB06].

# ESB Functionalities

## Mediation

- *Message Exchange Pattern (MEP)* enables various models: synchronous or asynchronous, allows one direction calls or *publish/subscribe* mechanism.
- *Result Cache* enables the possibility to save to a *cache* the returns of a service call, in order to allow subsequent service calls to receive the answer from the *cache*, without the need for the service to be called again. This is applicable mainly to static data that rarely change.
- *Transaction* enables atomic operations in an ESB: a message is completely processed or not at all. There is no possibility for the message to be partially processed.
- *Message Resequencing* enables the messages that are together but not in the proper order to be reordered. Such a component comprises an internal *buffer* that processes the messages until the complete sequence is available and may be sent.
- *Pass-Through Messaging* ensures efficient message transmission to the ESB. This service may be especially useful if the ESB is employed for services virtualization and the messages are sent from the services user to their provider without modification. In this case, the ESB does not process the message, simply transmitting it along.

# ESB Functionalities

## Security

The services from this category enable security through its various components, both on transport level and on message level.

- *Authentication* authenticates services consumers whenever they access ESB services and verifies the ESB authentication for services providers.
- *Authorization* provides a user or role authorization system in order to get access to services.
- *Security Mediation* provides support for interactions communicating outside security domains, by converting authorization data from one domain to the corresponding specific data from other security domains.
- *Encryption/Decryption* provides support to encrypt and decrypt the content of a message.

# ESB Functionalities

## Adapters/Transport

- This module includes adaptors required to connect ESB provided services through the *Hosting Services*. An ESB provides a predefined set of adaptors, but enables the developers to create their own adaptors depending on customer's specific requirements

## Service Hosting

This module enables services to be installed and run directly on ESB and it is usually required in case the ESB is based on an application server.

- *Service Container* provides one or more containers wherein the services may be installed and managed.
- *Component Model* provides a model of abstract components, such as Java EJB, Java Spring Framework or Microsoft COM+, depending on how the services are developed ([KMNSST13],[IBM-ESB], [IBM09], [Mule], [ESBPr]).

# ESB in practical scenarios

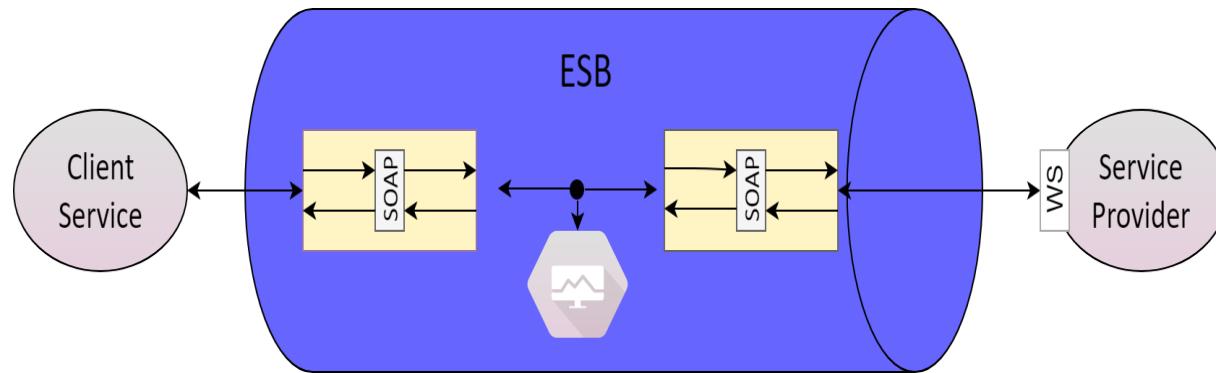


Figure: Decoupling and Monitoring

- Let there be a certain client and service provider. The latter provides a web service for the client to employ. In such a straightforward case the client may connect directly to the web service through the interface it provides. However, **an ESB may be employed to hide the service interface to the client**. As such, should there be any modifications of the web service, the ESB may be easily set up to meet the client's requirements without the need for modifications on the client side. Furthermore, the ESB property to monitor the activity inside it for **statistics purposes** – such as the **number of service calls per period of time, compliance to SLA**, etc.- may be used as well.

# ESB in practical scenarios

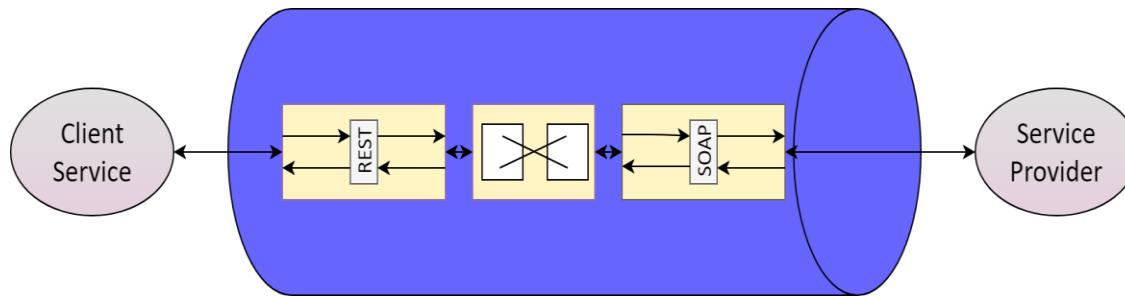


Figure: Services interoperability

- Let there be a certain client and service provider. The latter knows how to communicate through SOAP and the client through REST. Usually, the client should create a **proxy able to communicate** through SOAP and to answer through REST. In this case, if the service provider changes the interface, the client must modify his proxy in his turn, in order to use the new interface. On the other hand, if an ESB is to be used, the client is not required to modify his proxy in case the service provider changes the interface, but rather the ESB should be reconfigured to transform the provider's result in a known format of the client. Thus, the client is not to worry about modification on the provider's side, while the provider must not notify all his customers whenever he implements **modifications**.

# ESB in practical scenarios

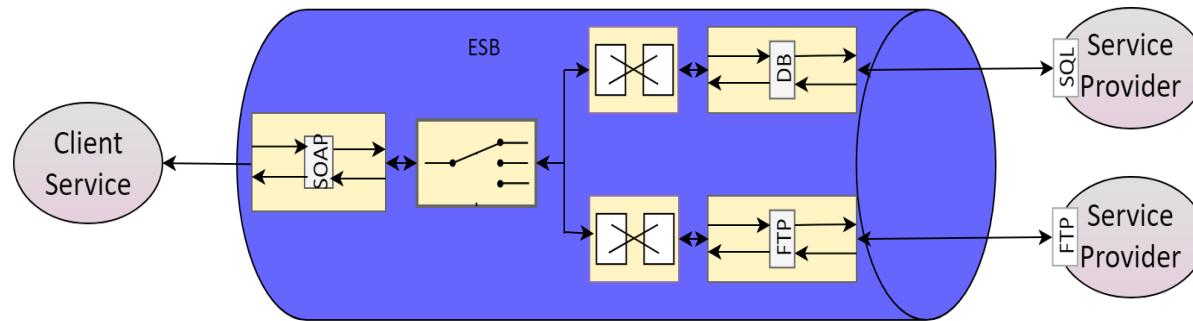


Figure: General Integration Mechanism

- There can be cases when similar services can be supplied using different protocols. We can have critical client applications requiring certain services, therefore **without an ESB the client is forced to implement two or multiple interrogation modules or the provider is forced to keep up multiple interfaces**. An ESB can behave as a **mediator** among clients and various service providers, and therefore not only the Client will not be affected, but also the provider interfaces will remain the same. There are situations when you have no direct control/contact with the service provider therefore an ESB can come with the adapter and the provided services can be used. Furthermore, other services compatible with other protocols can be integrated with the ESB, and changes will remain at this level only

# ESB in practical scenarios

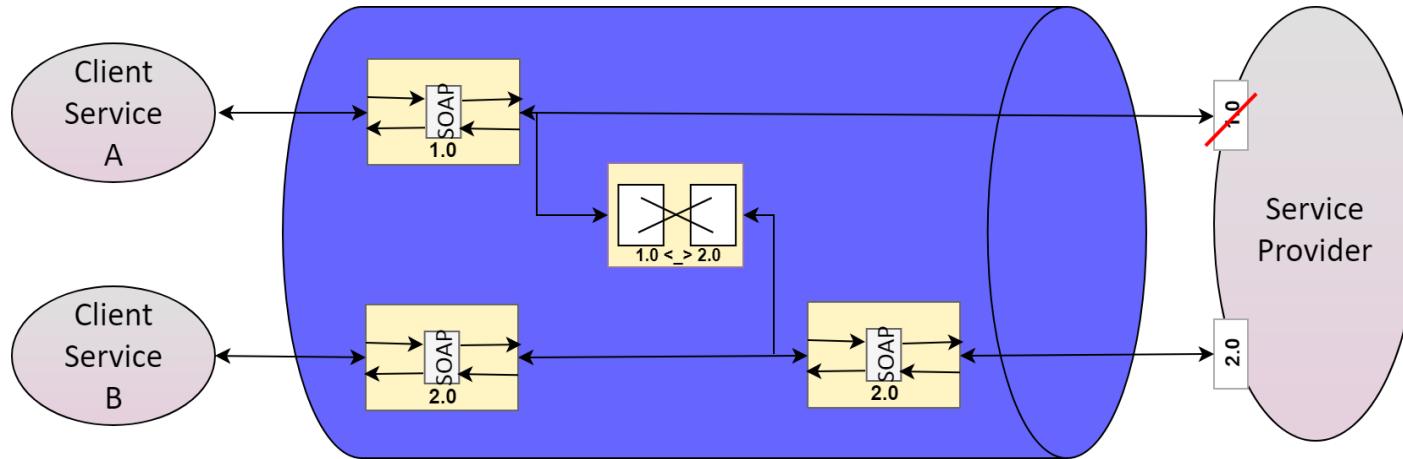


Figure: Versioning

- As any software system, services can suffer changes in time and new versions compatible with the new standards can appear. An **ESB can be used to perform the transformation from one version to another version**. Due to resource consumption, keeping up multiple versions of the same service is almost impossible, therefore an ESB can be used for this purpose.

# Technology mapping: SOA, ESB, BPM

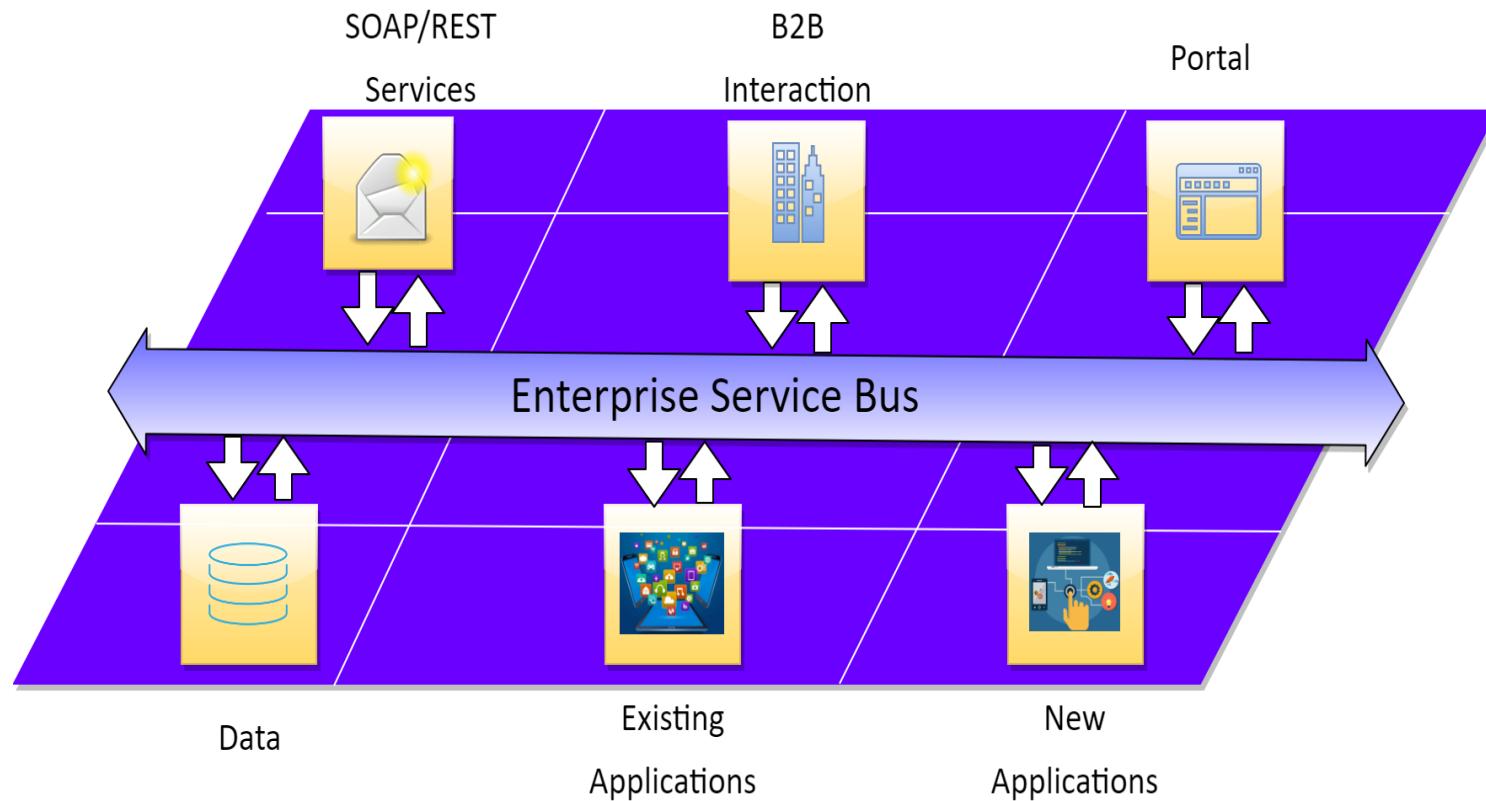


Figure: ESB as a part of SOA and BPM landscape

# Technology mapping: SOA, ESB, BPM

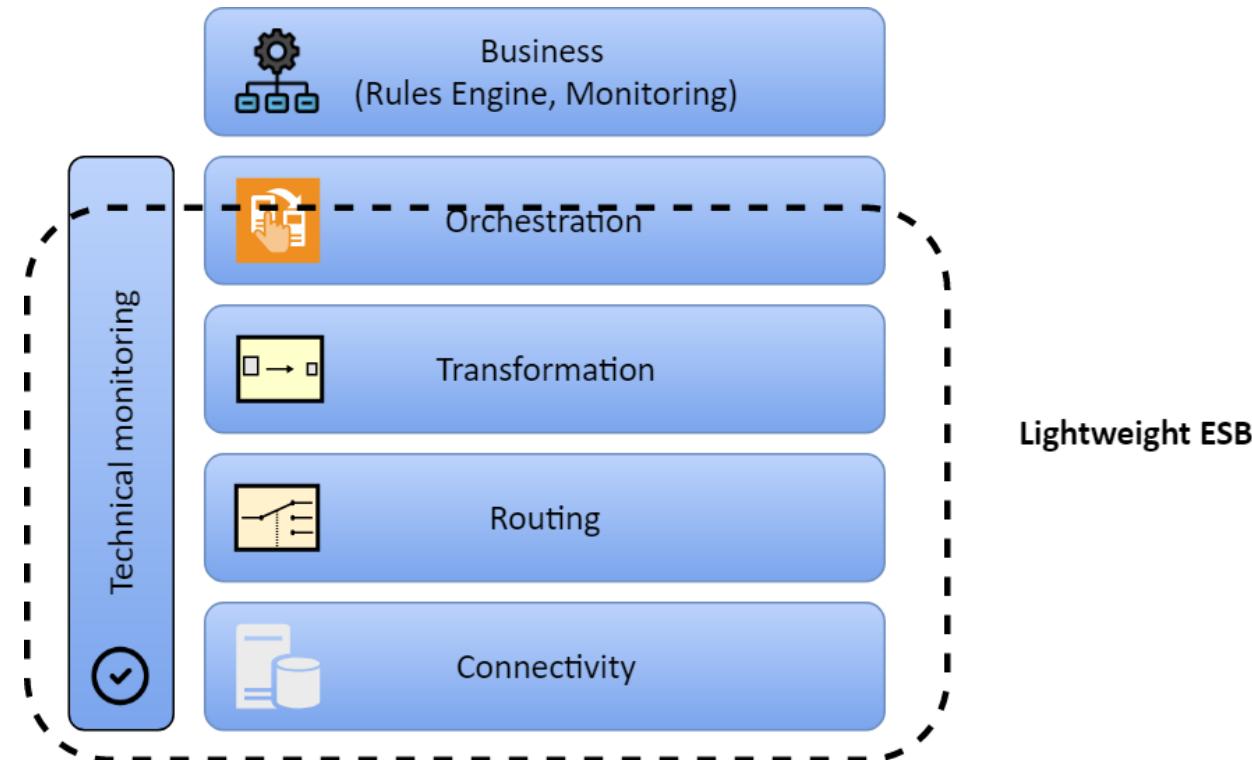
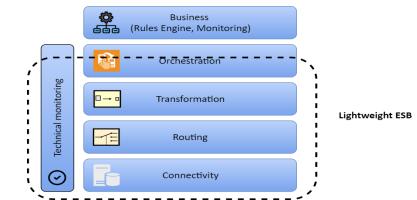


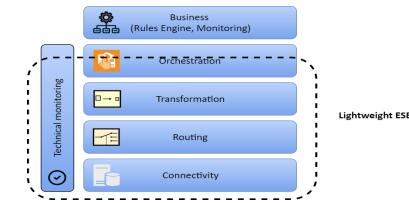
Figure: Stack layers for an lightweight ESB

# Technology mapping: SOA, ESB, BPM

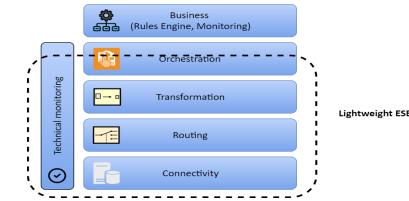


- **Business Monitoring** refers to services as aggregation, analysis and presentation (desirable in real time) information about inside activities that involves the owner, but also customers or business partners. At this level we have a dashboard that provides information about activities, performances, correlation among business processes or possibility to integrate with other monitoring systems.
- At the **connectivity** level we can use a MessageBroker pattern that will assure the messages flow. The MessageBroker can be embedded in the ESB or can be external.
- The **routing** level contains mechanisms which determine what messages must be sent and where. This level can be provided using some protocol translator, a router module with usual functionality, and all configuration can be done using Spring-XML or a DSL (Domain Specific Language) [Fow10] in our case. At this level, some messaging patterns or others newly created over the existing ones can be used.
- At the **transformation** level there are no specific rules, you can meet “no business at the integration level” (and the Canonical Data Model pattern can be used) or this decision depends on Enterprise and contexts.

# Technology mapping: SOA, ESB, BPM



- **Orchestration level** includes two sublevels: **technical orchestration** and **business orchestration** [Fer11].
- In [Rot12] orchestration is defined as a pattern: “*Orchestration—Make business processes agile and adaptable while using services based on the Request/Reply or Request/Reaction interaction patterns*”
- *Technical orchestration* sublevel has direct link with the levels below and its implementation “derives from the technical design of the flow (complex routing, errors management, etc.).”
- *Business orchestration* sublevel offers the base for Business Rules Engine and therefore, it is the bottom-up implementation of the business processes.
- When sophisticated business process definitions are necessary, the Business Orchestration sublevel may contain implementation for stateful processes (which can span over long periods of time) or may contain support for parallel executions and eventually merging mechanisms based on conditions. In this context, paradigms like SOA (Service Oriented Architecture) emerge and play an essential role in building bridges between business and IT departments.



# Technology mapping: SOA, ESB, BPM

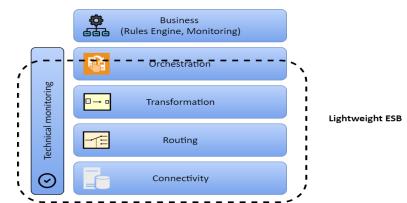
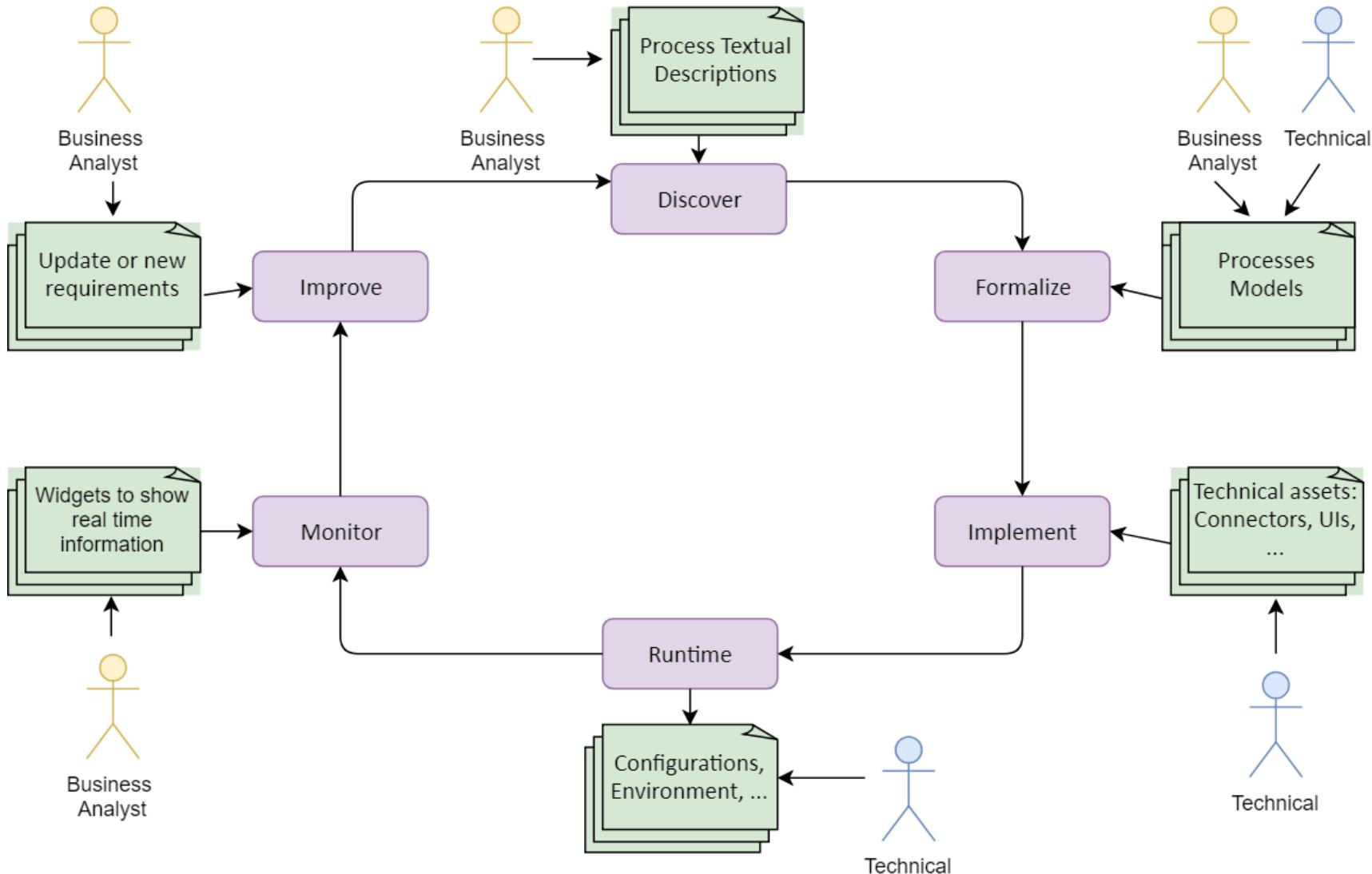
- It's well known that "processes run on services" [Jen12] and therefore BPM (Business Process Management) and SOA coexist.
- Gartner describes in [GarBPM17] Business Process Management as "a discipline that uses various methods to discover, model, analyze, measure, improve, and optimize business processes. A business process coordinates the behavior of people, systems, information, and things to produce business outcomes in support of a business strategy. Processes can be structured and repeatable or unstructured and variable. Though not required, technologies are often used with BPM."

In [Sal11] main concepts linked to BPM are defined and we are presenting them below:

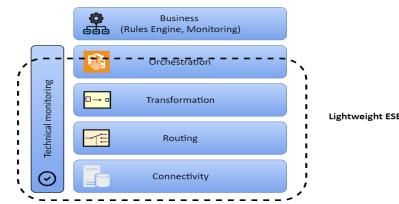
- *Process* - "a set of ordered actions that tends to transform an input to a desired output"
- *Business* - "relative to a domain, company or a scoped scenario, where certain rules and actions need to be applied to achieve a goal"
- *Business goal* - "objective to be satisfied inside the business scope that gives to the company a profit/benefit in some way"
- *Business process* - "Sequence of actions performed by humans and systems to achieve a business goal"

BPM contains some stages that are intended for Business analyst or for Technical staff, which are involved in a "forever" iterative cycle

# Technology mapping: SOA, ESB, BPM



# Technology mapping: SOA, ESB, BPM



- “BPM is a way of building operational solutions and SOA is a thought model that helps decompose complex problems into well-defined and reusable components.”[Jen12]
- From top to down view, a BPM offers an upper integration level, because it facilitates interaction between systems and human actors.
  - Use case: analysis of an enterprise environment seen from two different perspectives
- From a software architect’s/engineer’s perspective, in order to create an environment that offers services decoupling, component reutilization, no visible details regarding communication, a combination of ESB and SOA is a successful solution. But if there is a need for a new application in the company (or an integration with a legacy application), there is a good chance that a new application will be created for end users.
- From the engineer’s point of view there is no problem, but for end users this means: multiple logins (there may be multiple users’ accounts or the need to provide the password twice), or performing the same task in different systems. Also, for new end users there are many things to learn and trainings are time consuming.
- BPM offers to users (end/new users, managers) a proper environment with a Unified User Interface to do a specific task and for developers an integration environment, to coordinate systems and human interactions, to develop or add new functionalities in a unified way

# Technology mapping: SOA, ESB, BPM

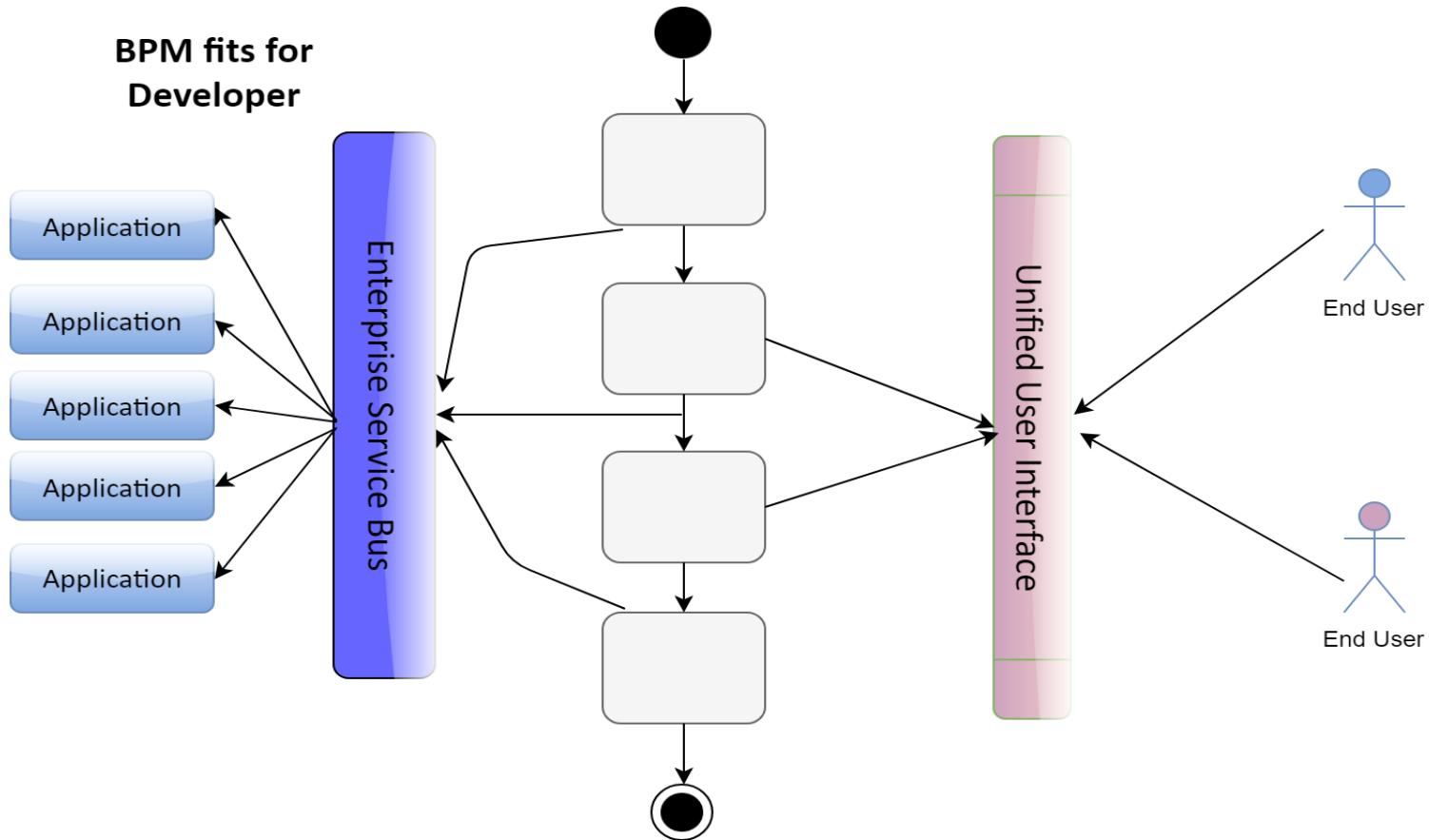
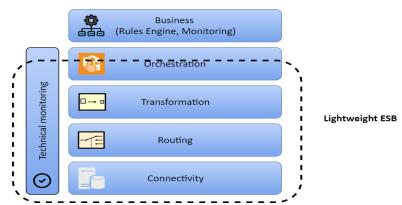


Figure: Mapping of ESB and SOA and BPM

# Integration Through Cloud Eye

- SOA and ESB were pertinent solutions for integration, but the high cost of implementation has determined many organisations to use old point-to-point EAI patterns.
- The software systems as a whole tend towards XaaS and a new era for application programming interface (APIs), and all these imply a reposition of the integration process in an enterprise (both at business and at technological level).
- The integration technologies in cloud, sometimes called SMAC (Social, Mobile, Analytics, Cloud) even if they benefit from previous technologies, are prone to suffer radical transformations.
- We are not talking anymore just about intra-organisational integration (where an ESB can play a perfect role), instead we talk also about inter-organizational integration.

# Integration Through Cloud Eye

- It is possible to use an ESB for the inter-organizational integration, but there are some drawbacks [Jos14]:
  - Installing integration environments involves many supplementary environments, leading to lost time required for configuration and installation
  - Imagine that something is changed at one endpoint. In an ESB necessary changes are required at levels that belong to flows touching the endpoint
- **iPass Models**
- Gartner provides in [GariPaaS17] a precise definition for iPaaS (Integration Platform as a Service: it “is a suite of cloud services enabling development, execution and governance of integration flows connecting any combination of on premises and cloud-based processes, services, applications and data within individual or across multiple organizations”.
- iPaaS is still at the beginning of its evolution, and Gartner mentioned in 2017 different classes of integration providers, which probably do not cover all features an iPaaS can offer: B2B integration, cloud integration, ESB and SOA infrastructure [MuleiPaaS17].

# Integration Through Cloud Eye

- In the world of everything as a service (XaaS) even integration is a Service on cloud and that means there is no capital expenditure but just operational costs for these services' clients.
- To allows a rapid integration, iPaaS providers preconfigure and build connectors for well known SaaS endpoints. Also some pre-packaged integration templates are provided in order to reduce overall integration effort.
- Models for integration supported by an iPaaS
  - On-Premise - Cloud Integration
  - Cloud - Cloud Integration
  - In the case of mobile device management (MDM), integration deployment can happen on cloud in order to allow access to various software (applications, web services, databases). We are mainly in a cloud-cloud integration model.
  - In the Internet of Things for integration case we deal with high-volumes of data and a proper management can be done with proper resources that can be found in cloud, therefore in most cases, this kind of integration happens at cloud level

# Integration Through Cloud Eye

- The integration endpoints are on-premise and therefore the integration deployment (flow/tasks) happens on-premise

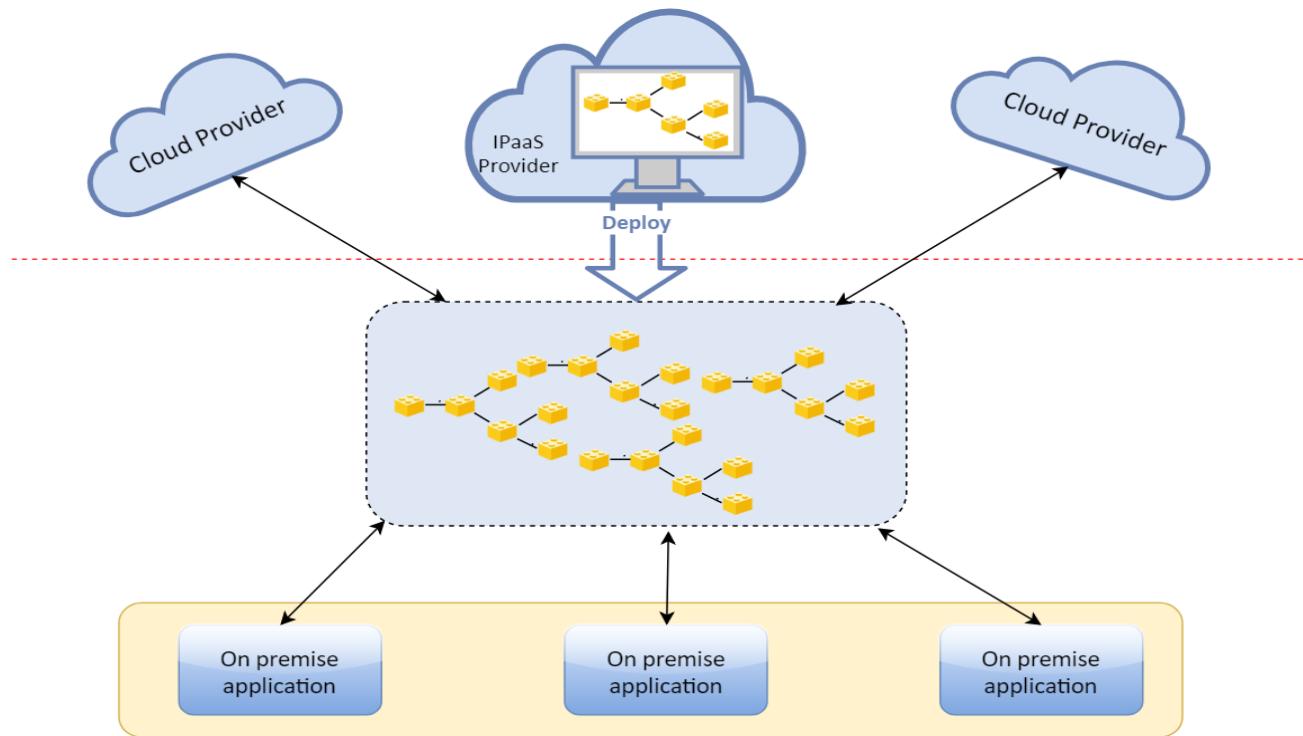


Figure: On-Premise - Cloud Integration

# Integration Through Cloud Eye

- The integration endpoints are on cloud and therefore the integration deployment happens on cloud

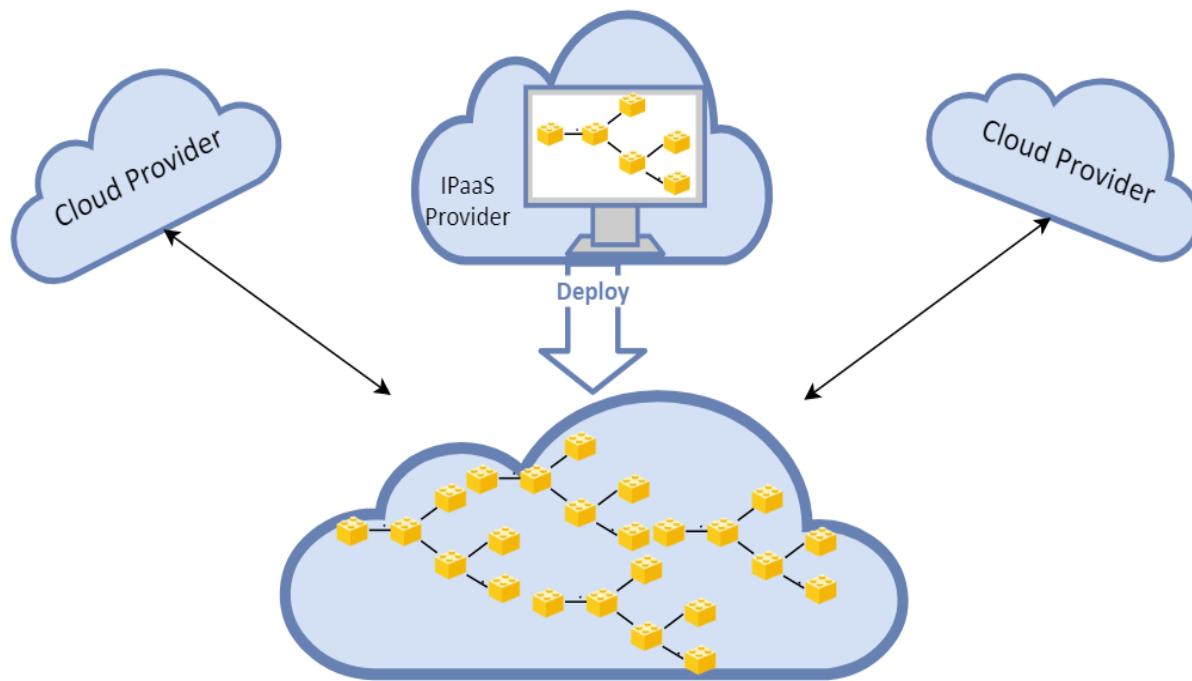


Figure: Cloud - Cloud Integration

# Integration Through Cloud Eye

iPaaS comes with advantages, that are characteristics in general for SaaS:

- No capital expenditure for an integration infrastructure
- No installation, versions, upgrade to take care of or manual processes
- No maintenance costs
- Existence of pre configured connectors/adapters and drag and drop tools may ease the integration process
- High availability, load balancing
- Security, reliability for data transfers
- Takes advantage of cloud associated technologies: Dynamic scaling, containers, resource management

The limitations of iPaaS are somehow similar with those specific to cloud systems: running on someone else's hardware, vendor lock-in, issues regarding security, privacy, governance

# Integration Through Cloud Eye

These factors can be considered as essential issues when someone wants to design and develop an iPaaS system [GPGTIW17]

- **Degree of "cloudiness"** - means that aspects specific to cloud such as tenant isolation, elasticity, dynamic scaling, self-service, billing mechanism should be provided
- **Enterprise worthiness** - aspects such us high availability, disaster recovery, technical support and secure access should be offered
- **Functional completeness** - represents the range of services provided by the iPaaS: “core integration capabilities (multiprotocol support and bridging; multiple data/message delivery styles; data/message validation, transformation and routing), adapters, data quality, development tools, administration, monitoring and management environment, support for secure communication, governance/API management, and community collaboration/crowdsourcing facilities”
- **Openness** - an iPaaS system should offer extensibility, support for open standards and integration with on-premise application or on-premise integration platforms

# Integration Through Cloud Eye

These factors can be considered as essential issues when someone wants to design and develop an iPaaS system [GPGTIW17]

- **Productivity** - an iPaaS platform should make available documentation, examples and a repository with instruments that enable rapid integration for various contexts
- **Ease of operation** - this aspect contains two types of functionalities. The first category refers to the fact that an iPaaS system should provide a HUI interface or at least a command line interface to ease integration operations. The second category refers to instruments that assure backward compatibility in order to accomplish an integration flow.
- **Citizen integrator support** - as we have seen for ESB, there are users without technical skills but with well developed business skills. An iPaaS platform is supposed to provide support to a suite of “non-coding” tools.
- **Versatility** - an iPaaS should offer support for emerging integration requirements that many times provoke changes at Business level with effects at technical level

# Integration Through Cloud Eye



Figure: iPaaS Gartner Magic Quadrant  
[GPGTIW17]

# Integration Through Cloud Eye

Representatives solution:

- **DellBoomi** offers the AtomSphere platform that supports many types of integration and some of them are in trend with new evolving technologies: IoT integration, MDM (Mobile Device Management), EDI Services (Electronic Data Interchange), API management, B2B integration. Boomi offers a visual design interface that hides complexity and allows developers to develop web services or various SOA architectures. Also, it provides the possibility to build integration using pre-built connectors and processes and friendly drag-and-drop tools.
- In [Sel16], a cloud-to-cloud integration example is presented, in which two SaaS applications (Oracle EBS and Salesforce CRM) are connected through AtomSphere. This solution was chosen by A10Networks, a well known company in networking and security. Because they experienced fast business growth they chose Oracle E-Business Suite as ERP solution and for CRM they used Salesforce solution. Before integration, the data had been manually introduced twice. Measurement showed that after integration errors had dropped by 90% and working hours for staff implied in those operations decreased by 40%.

# Integration Through Cloud Eye

Representatives solution:

- **IBM** has three solutions for iPaaS designed for different users categories:
  - IBM App Connect Professional focuses on ad-hoc and citizen integrator
  - IBM Integration Bus on Cloud is addressed to specialist integrators
  - IBM API Connect “provides capabilities to manage and secure the integration flows when exposed as APIs”
- **Microsoft** provides Azure Logic Apps (from the middle of 2016) as their iPaaS stand-alone solution and also Microsoft Flow addressed to citizen integrators.
- Microsoft Flow is built on top of Azure Logica Apps, which is in turn based on other services offered in Azure: BizTalk Server, SQL Databases, SharePoint, API Management and Service Bus and their intentions are to integrate with IoT Suite, Machine Learning, Microsoft Dynamics that will increase the range of functionalities.
- What should be noted is a very wide range of connectors that cover protocol connectors, Azure Services and Power APP Connectors, SaaS connectors, B2B or EDI connectors, Hybrid connectors [Cru17]

# Integration Through Cloud Eye

Representatives solution:

- **MuleSoft** offers Anypoint iPaaS solution providing: API management, on-premise and SaaS integration, B2B integration using EDI. Anypoint Exchange is a huge repository that contains connectors, templates, APIs available to perform integration in their iPaaS that is layering in AWS. Also, their solution is available as Docker image therefore, it can be employed for on-premise or hybrid environment.
- In [Sel16], a use case of Anypoint iPaaS is presented by Splunk , a company that monitors, collects and analyzes data to offer digital transformation [WBM14] services. Because of the business growth, a new level of efficient internal operation was needed. Therefore, they used Mule Solution to integrate Salesforce and Netsuite. The quote-to-cash [Har16] process has experienced a rapidity of time from minutes to real-time operations.

# Integration Through Cloud Eye

Representatives solution:

- Oracle offers two solutions for iPaaS Oracle Integration Cloud Service (ICS) for ad hoc integrators and Oracle SOA Cloud Service for specialist integrators.
- Oracle iPaaS solutions are integrated with: “Oracle Self Service Automation for citizen integrators, Oracle Process Cloud Service for improved orchestration, Oracle Real-Time Integration Business Insight for business activity monitoring, Oracle API Platform Cloud Service for API management, Oracle Managed File Transfer Cloud Service for managed file transfer and Oracle IoT Cloud Service for IoT integration”
- We present below some partial results of a comparison performed in [CapiPaaS17], that take into account “application and integration middleware” that are not iPaaS solutions, but can be/are already part of an iPaaS.

The screenshot shows a comparison chart titled "BizTalk Server vs IBM WebSphere vs Oracle Data Integrator vs SnapLogic". The chart lists four products with their respective logos, developer, star ratings, and review counts. Each product has a "View Profile" button below it.

Product	Developer	Rating	Reviews
BizTalk Server	Microsoft	4.5/5	(3)
IBM WebSphere	IBM	4.5/5	(7)
Oracle Data Integrator	Oracle	4.5/5	(1)
SnapLogic	SnapLogic	4.5/5	(1)

# Integration Through Cloud Eye

## iPaaS architecture and services

- iPaaS is composed by two main levels:
  - top level designated to technical users and business users.
  - the second level can be viewed as background integration platform services for: data formats (XML, JSON) and standards support (Health Level Seven Standards [HL7], Electronic Data Interchange For Administration, Commerce and Transport [EDIFACT] et.al.), data transformation, connectors/adaptors for on-premise packaging application or for SaaS, routing, top operations (orchestration, monitoring, management, integration with various tools, APIs management et.al.).
- As one may notice, from the structural point of view, there are similarities with an ESB system: we have on top instruments for business users and technical specialists and on subsequent levels there are integration mechanisms.

# Integration Roadmap

- In the light of Cloud Computing new technologies and challenges, integration has new dimensions.
- The first forms of integration were referring to the process of replicating data to another cloud (cloud-cloud integration) or replicated some data from/to on-premise systems (cloud-on-premise integration).
- “data” has become step by step the central concept around which technologies gravitate
- In [Lin14] David Linthicum oversaw the evolution of data integration technologies

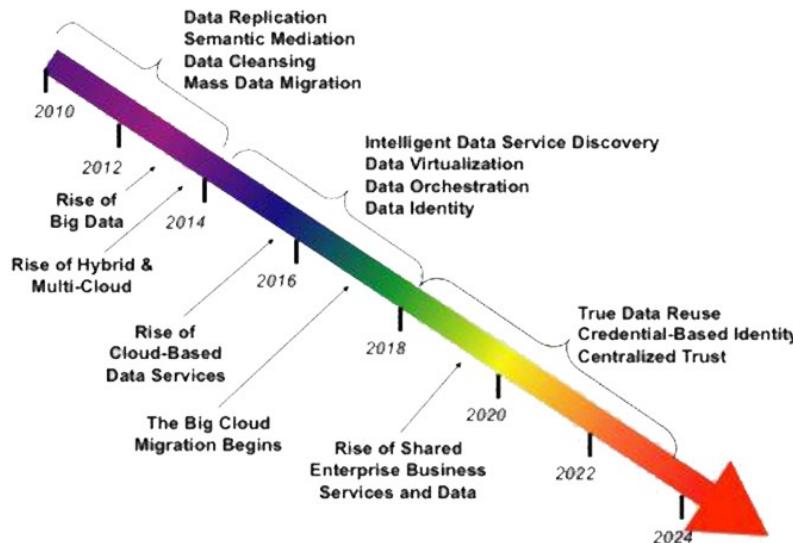
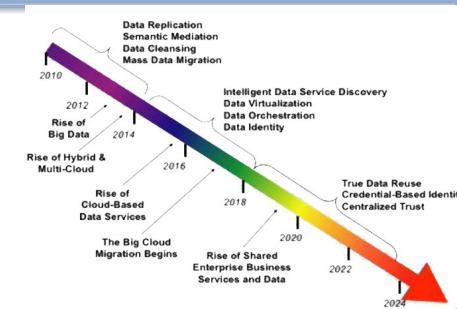


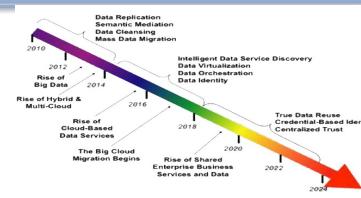
Figure: Data Integration evolution [Lin14]

# Integration Roadmap



- In the 2010-2014 period there is an increased evolution on the cloud level technologies, while on the data level we encounter: “*data replication, semantic mediation, data cleansing and mass data migration.*”
- Enterprises began to have hybrid or multi-cloud architectures, and that allowed them, in order to offer maximum benefits for business processes, to move data (replicate, storage) from cloud-to-cloud, from enterprise-to-cloud, from enterprise-to-enterprise in private-clouds et.al.
- In this period, the semantic web technologies were viewed as solutions to improve customer satisfaction and from a technological perspective to assure a more automate mediation. Ontologies were also used in order to simplify operation for services reuse.

# Integration Roadmap

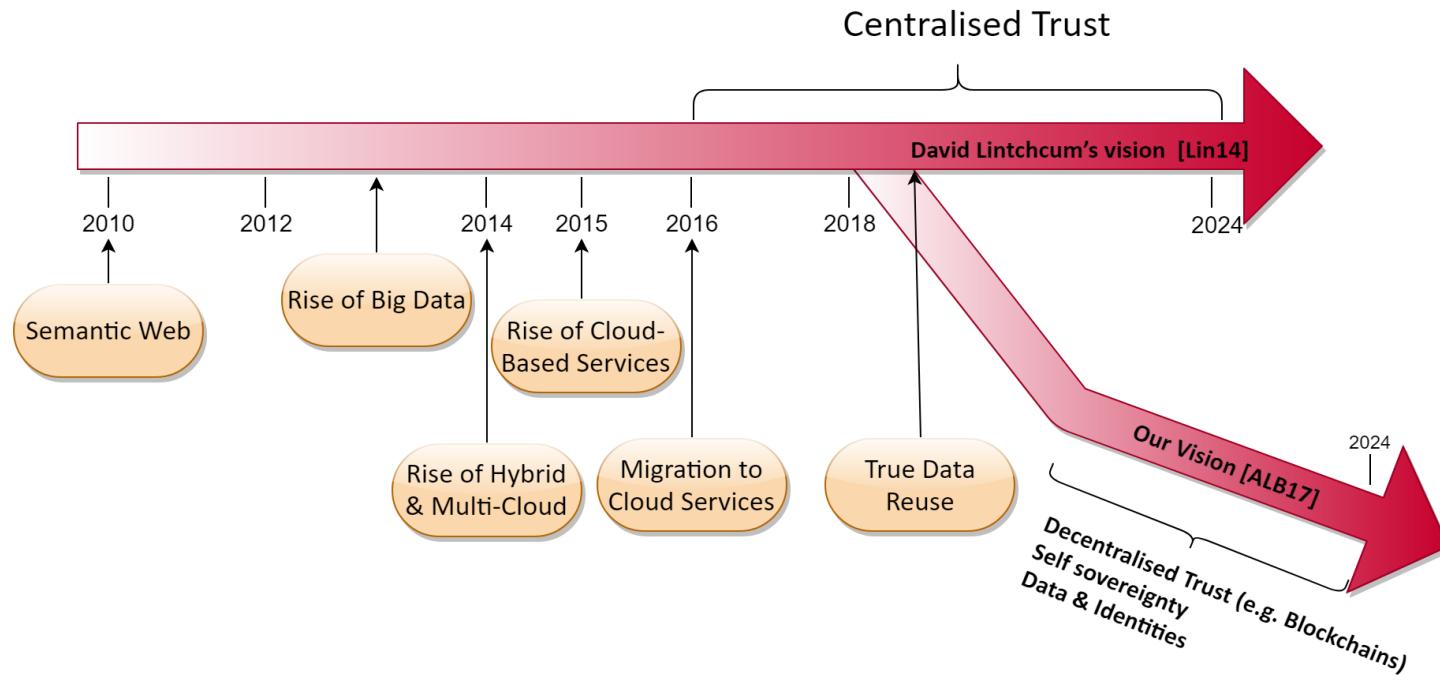


David Linthicum envisions, and the technological reality partially confirms his point of view, the cloud based architectures becoming essential parts in an Enterprise and as such, there is need for [Lin14]:

- **Intelligent Data Service Discovery:** integration technology needs to automatically find (discover, identify details about the API) and provide data services from cloud or non-cloud environments. We can understand this as a generalisation of UDDI [AB06] plus smart technologies (such as Semantic Web based) where Enterprises can benefit by all available data.
- **Data Virtualization** - comes with the concept as data viewed as service itself; it is not a new concept, and does not imply back-end changes; instead it “redefines how the database is made visible and consumed by other systems”[Lin14]
- **Data Orchestration** - has influences from service orchestration and refers to various mechanisms to combine data and therefore to fulfill the needs for multiple solutions
- **Data Identity** - refers to the mechanism that allows to control who and when to consume the data
- The **rise of shared Enterprise business services and data** is viewed as a real path for data reuse (“repurpose data sets”) that can bring benefits to many new or existing enterprises. You can envision a company who can access databases providing them the ability to correlate and avoid future potential losses.
- Through ***credential-based identity and centralized trust***, David Linchcum envisions the existence of techniques that “can validate both the data (structure and content), as well as humans and machines that would like to view or manipulate the data”.

# Integration Roadmap | PrivateSky Vision

- Our vision regarding the level of centralization takes into account the recent development from the blockchain technology area. In accordance with our vision from PrivateSky project [PrivateSky16] we believe in Decentralised Trust (using Blockchain based mechanisms or other underlying technologies (Self Suveranity Data & Identities [Alb17])



# Summary

- Integration – as a key concept
- Integration and/or Interoperability
- Integration contexts
- Service Oriented Architecture
- Evolution of Integration Technologies
- ESB - Enterprise Service Bus
- Technology mapping: SOA, ESB, BPM
- Integration Through Cloud Eye
- Integration Roadmap

# Bibliography

- [AB06] Lenuta Alboiae, Sabin Buraga, Web Services (in Romanian), Polirom Publishing House, ISBN 973-681-522-6 Iasi, 320 pages (2006)
- [Bec04] Kent Beck, Extreme Programming Explained: Embrace Change, Publisher Addison-Wesley Professional, 224 pages, ISBN-10: 0321278658, 2nd edition (2004)
- [DMG07] Paul Duvall, Steve Matyas, Andrew Glover, Continuous Integration - Improving Software Quality and Reducing Risk, ISBN-13: 978-0321336385 (2007)
- [Garl][Online] Available: <https://www.gartner.com/it-glossary/integration/>
- [Fow14][Online] Martin Fowler, Microservices - a definition of this new architectural term, <https://martinfowler.com/tags/microservices.html> (2014)
- [Jos14][Online] Maneesh Joshi, Why Buses Don't Fly in the Cloud: Thoughts on ESBs. Available: <http://insights.wired.com/profiles/blogs/why-buses-don-t-fly-in-the-cloud-thoughts-on-esbs#axzz4z9L9T8Pc> (2014)
- [Mule][Online] MuleESB. Available: <http://www.mulesoft.org/>
- [MuleIPaaS17] [Online] What is iPaaS? Gartner Provides a Reference Model,. Available: <https://www.mulesoft.com/resources/cloudbase/what-is-ipaas-gartner-provides-reference-model> (2017)
- [GPGTIW17][Online] Keith Guttridge, Massimo Pezzini, Elizabeth Golluscio, Eric Thoo, Kimihiko Iijima, Mary Wilcox, Magic Quadrant for Enterprise Integration Platform as a Service. Available: <https://www.gartner.com/doc/reprints?id=1-3X0Y452&ct=170403&st=sb> (2017)
- [Cru17][Online] Paco de la Cruz, Microsoft Azure Integration Platform as a Service (iPaaS). Available: <https://blog.mexia.com.au/azure-integration-platform-as-a-service-ipaas> (2017)

# Bibliography

- [WBM14][Online] George Westerman, Didier Bonnet and Andrew McAfee, The Nine Elements of Digital Transformation. Available: <https://sloanreview.mit.edu/article/the-nine-elements-of-digital-transformation/> (2014)
- [Har16][Online] Anika Harvey, What is Quote-to-Cash? Available: <https://www.salesforce.com/blog/2016/09/what-is-quote-to-cash.html> (2016)
- [CapiPaaS17][Online] Integration Software. Available: <https://www.capterra.com/integration-software/compare/86590-131695-1310-83934/BizTalk-Server-vs-Enterprise-Mashups-vs-IBM-WebSphere-vs-SnapLogic> (2017)
- [Lin14][Online] David S. Linthicum, The next generation of data integration technology for the changing cloud market. Available: <https://gigaom.com/2014/09/01/the-next-generation-of-data-integration-technology-for-the-changing-cloud-market/> (2014)
- [Alb17] Alboiae L., Towards a smart society through personal assistants employing executable choreographies, At 26th International Conference on Information Systems Development, Cyprus, 6-8 September (2017)
- [PrivateSky16] PrivateSky Project - “Dezvoltare experimentală în parteneriat public privat pentru crearea de platforme cloud autohtone cu caracteristici avansate de protecție a datelor”, POC 2014-2020, axa prioritără 1, acțiunea 1.2.3 , Tip proiect – Parteneriate pentru transfer de cunoștințe, Contract 13/01.09.2016, SMIS 106611, ID P\_40\_371, Proiect co-finanțat din Fondul European de Dezvoltare Regională prin Programul Operațional Competitivitate 2014-2020: <https://profs.info.uaic.ro/~ads/PrivateSky/>

# Summary

- History & Evolution
- Distributed Systems
  - Definitions
  - Characteristics
  - Use
  - Relation with multiprocessor parallel systems
  - Flynn taxonomy
  - Concepts
  - Classifications



# Questions?

# **Computing Platforms Evolution**

---

**Prof. Univ. Dr. ALBOAIE LENUȚA**

# Content

---

- ❖ Platforms for Monolithic Applications
- ❖ Web Platforms (Web 1.0, Web 2.0, Web 3.0, Web 4.0, ...)
- ❖ Enterprise Integration Platforms (MOM, ESB)
- ❖ Cloud Computing Platforms
- ❖ Mobile Platforms
- ❖ Edge/Fog Computing Platforms (IoT, 5G)
- ❖ Self Sovereign Cloud Platforms (Blockchain , IPFS, PrivateSky EDFS)
- ❖ Personal Assistant Platforms

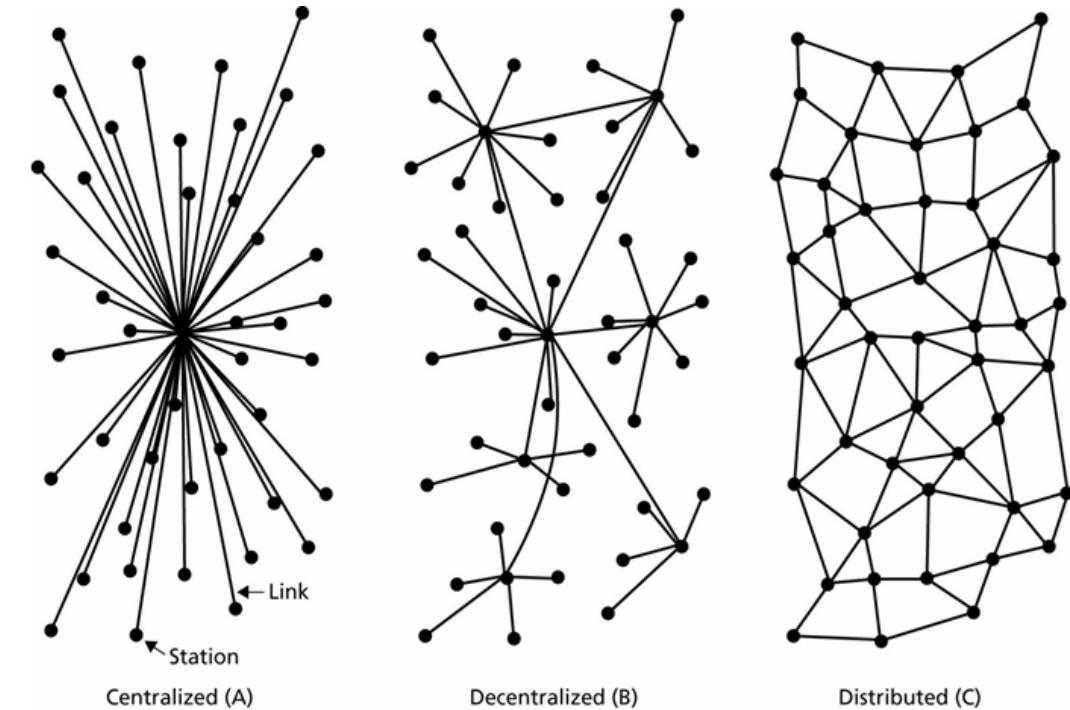


## Analyzed Factors:

- Architecture
- Integration
- Privacy

**“Integration** services are detailed design and implementation services that link application functionality (custom software or package software) and/or data with each other or with the established or planned IT infrastructure.”

**“Privacy** - the right of individuals to keep information about themselves from being disclosed to others”



[Baran, P. (1964). *On Distributed Communications*, Memorandum RM-3420-PR. Santa Monica, Calif.]



# Platforms for Monolithic Applications

---

- ❖ **Examples:** Application Servers (ASP, Tomcat), DataBases (SQL Databases)
- ❖ **Networking:** sometimes LAN
- ❖ **Programmability:** client/server (RPC), CORBA, DCOM
- ❖ **Architecture:** mostly Client/Server based on custom binary, text or XML protocols
- ❖ **Centralisation:** centralised
- ❖ **Privacy:** not a big issues
- ❖ **Integration:** Information silos



# Platforms for Monolithic Applications | Integration

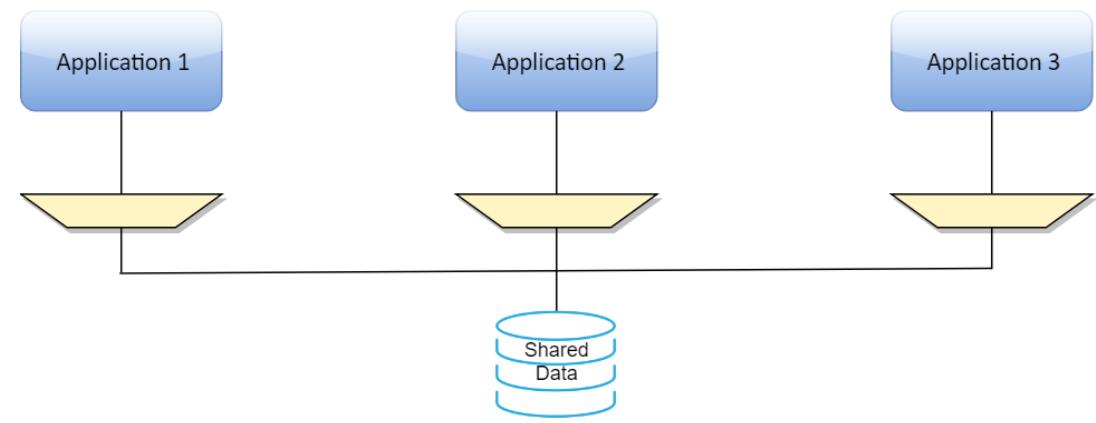
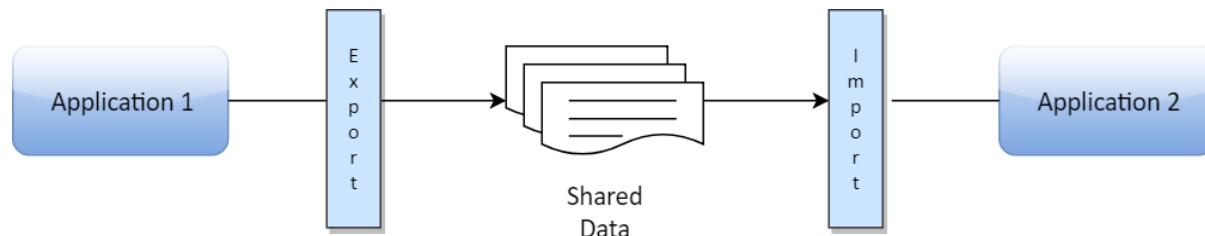


Figure: Integration through file transfer

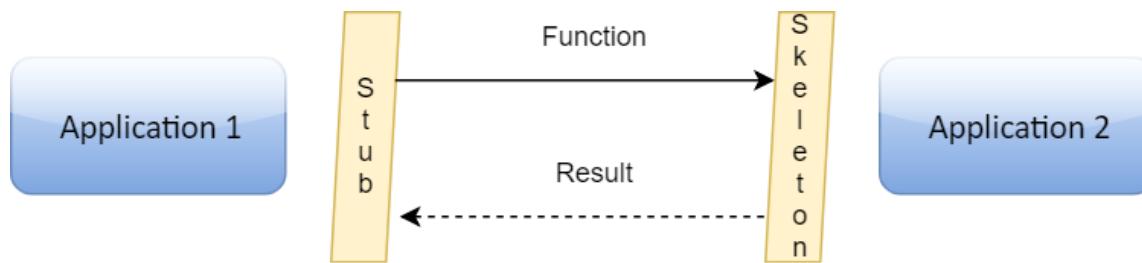


Figure: Remote procedure invocation scenario

Figure: Integration through shared database



# Web Platforms | Web 1.0

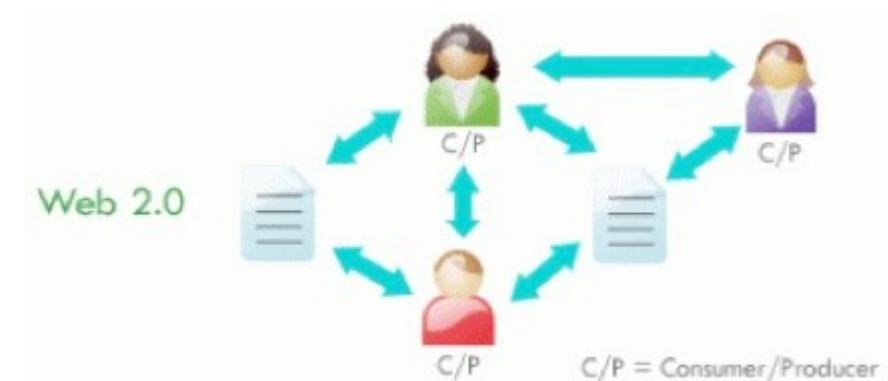
- ❖ **Examples:** static pages
- ❖ **Characteristic:** systems with permanent connection to Internet
- ❖ **Programmability:** HTTP, HTML, URI, CGI
- ❖ **Networking:** Client/Server based on HTTP
- ❖ **Centralisation:** centralised
- ❖ **Integration:** content is served from the server's file system
- ❖ **Privacy:** not a big issues





# Web Platforms | Web 2.0

- ❖ **Examples:** social networks, electronic commerce, mashups,...
- ❖ **Programmability:** HTTP, HTML, CSS, Ajax, Java Script, Web Services (REST/JSON, SOAP/XML), Web APIs
- ❖ **Networking:** Mostly Client/Server based on HTTP
- ❖ **Centralisation:** centralised
- ❖ **Privacy:** issues
- ❖ **Integration:** web services, data marts, SOA



[<http://airccse.org/journal/ijwest/papers/3112ijwest01.pdf>]

# SOA

- ❖ *Service Oriented Architecture* is a design methodology and architecture aimed at maximizing the reuse of multiple services (possibly implemented on different platforms and using multiple programming languages)[Erl06]

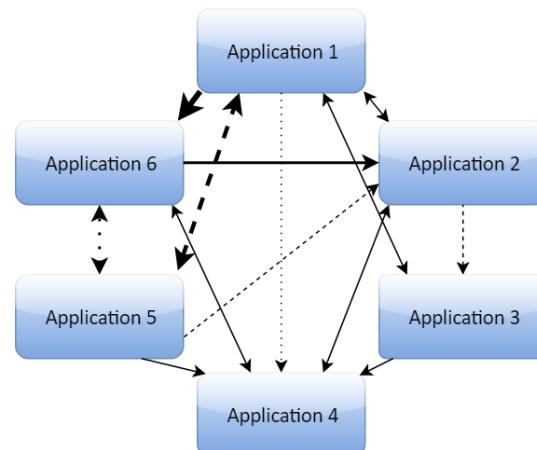


Figure : Point to point integration

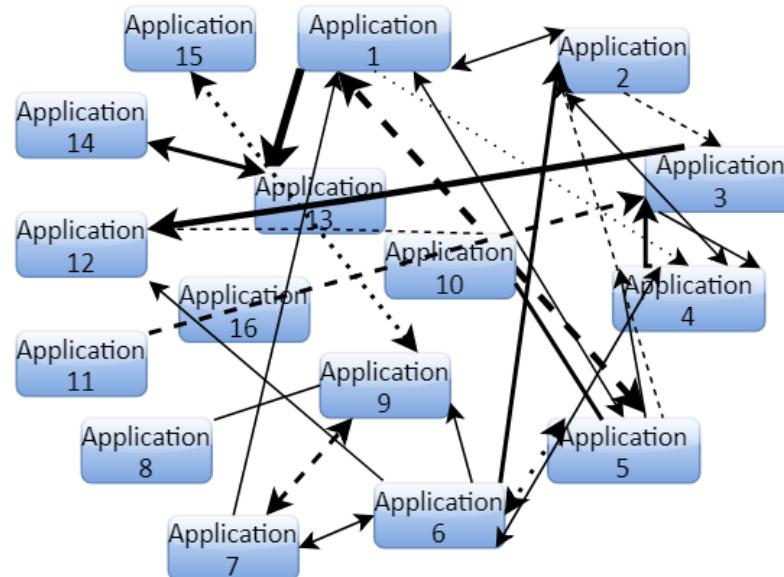


Figure. Point to point explosion in number of connections

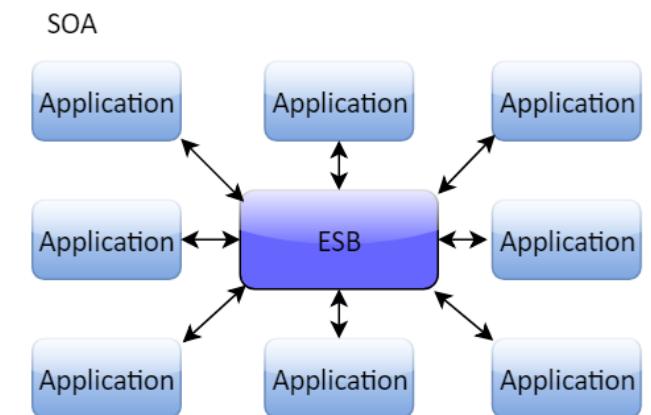


Figure: Using ESB in a SOA architecture



# ESB(Enterprise Service Bus)

---

- ESB - represents a collection of principles, rules and features that create the environment for an architectural layer providing a bus based integration among various applications/services
- ESB is viewed as a distributable backbone upon which an enterprise can build various SOA architectures or microservices oriented architectures

Roles examples:

- ❖ General Integration Mechanism
- ❖ Decoupling and Monitoring
- ❖ Services interoperability
- ❖ Versioning



# Enterprise Integration Platforms

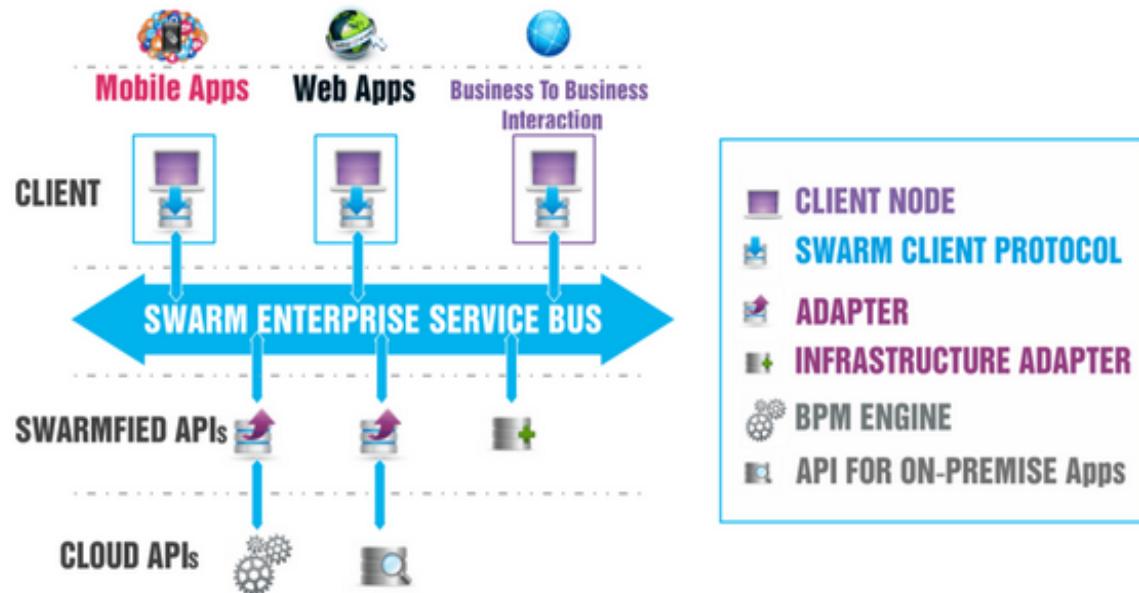
---

- ❖ **Examples:** Mule ESB, Talend ESB,..., Swarm ESB
- ❖ **Programmability:** orchestration technologies or alternatives
- ❖ **Networking:** Mostly Client/Server based on HTTP or Swarm Communication paradigm
- ❖ **Integration:** MOM, ESB with orchestration, ESB with choreographies
- ❖ **Centralisation:** centralised
- ❖ **Privacy:** not a big issue



# Swarm Communication

---



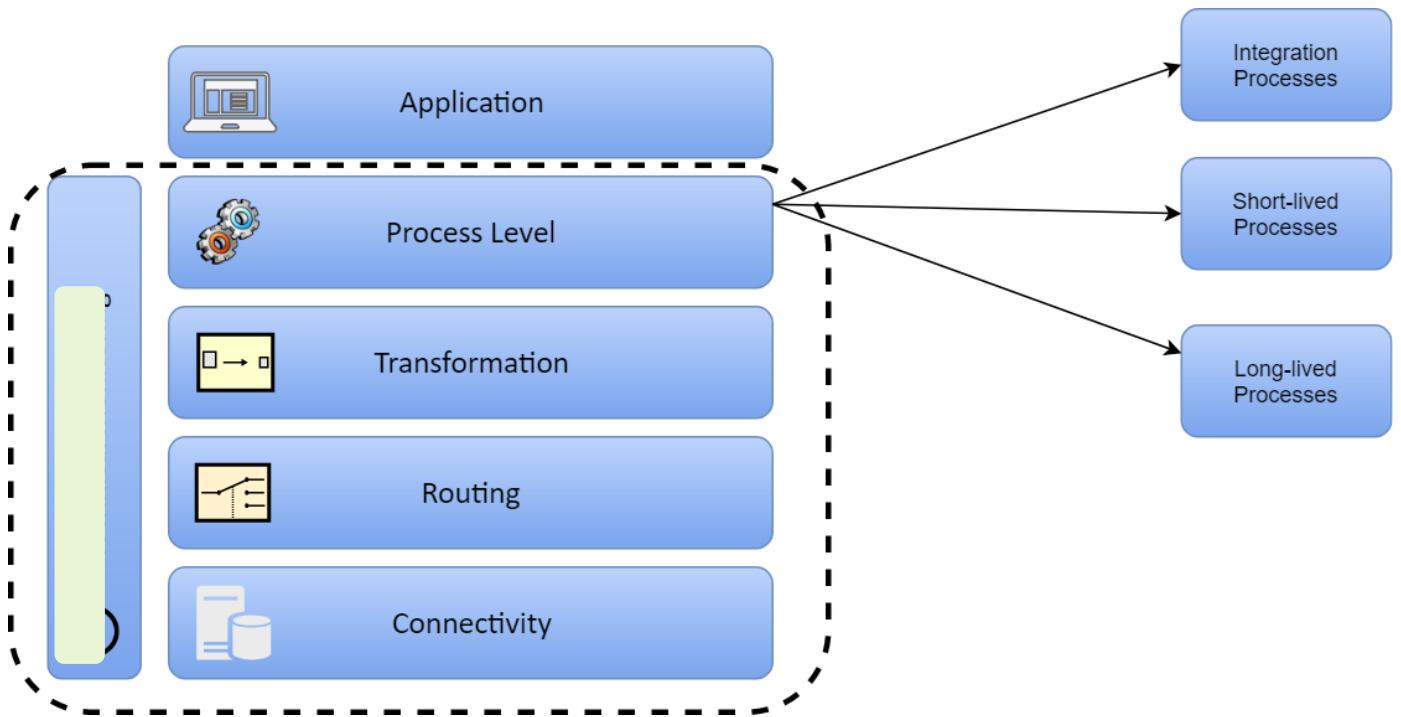
## Swarm Communication:

- **Nodes** (client nodes, adapter nodes, infrastructure nodes)
- **Swarm messages**
- **Swarm phases**

Lenuta Alboiae, Sinica Alboiae, and Panu Andrei. Swarm Communication - a Messaging Pattern proposal for Dynamic Scalability in Cloud. At 15th IEEE International Conference on High Performance Computing and Communications (HPCC 2013). Zhangjiajie, China, November 2013.



# Swarm ESB



# Swarm ESB - a lightweight ESB

- **connectivity** level
  - **routing** level
  - **transformation** level
  - **process** level



# Executable Choreographies

---

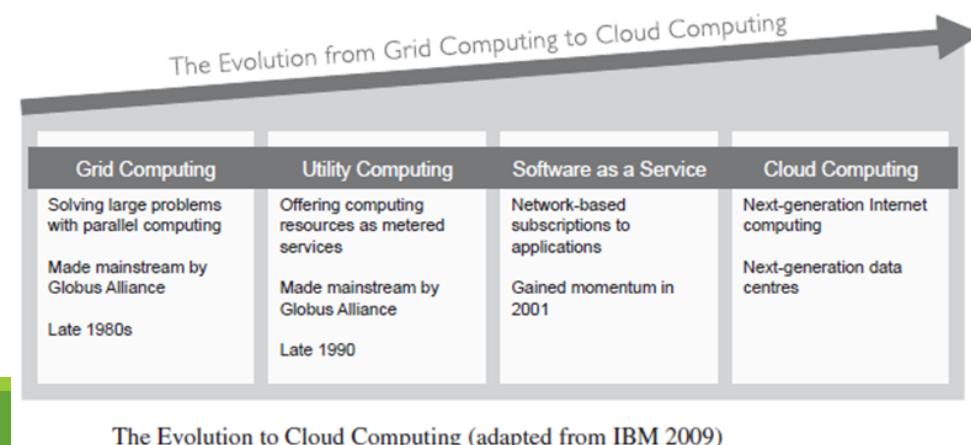
- Choreography is the formal description (non-executable choreography) or the concrete description (executable choreography) of a contract or process which involves the interaction of several systems belonging to departments or autonomous organisations, with different interests (e.g. regarding data privacy and security)
  - Executable choreographies
    - mechanism to describe in code the contracts among several organizations
    - can be implemented using Swarm communication



# Cloud Computing Platforms

Convergence: Grid Computing, Web Services, Utility Computing, SaaS

- Grid computing assures a scalable, flexible, robust and reliable physical infrastructure
- Web Services/SOA assures personalization of services and APIs enable programming access to physical infrastructure through abstract interfaces
- Utility Computing and SaaS – illustrate the increasing trend towards external deployment and sourcing of computing and applications





# Cloud Computing Platforms

---

- “We argue that Cloud Computing not only overlaps with Grid Computing, it is indeed evolved out of Grid Computing and relies on Grid Computing as its backbone and infrastructure support. The evolution has been a result of a shift in focus from an infrastructure that delivers storage and compute resources (such is the case in Grids) to one that is economy based aiming to deliver more abstract resources and services (such is the case in Clouds).” [Fos08]
- “... Cloud computing is a new model of computing fueled by the shift of control from end-users towards increasingly centralized services providers.”[PS12]



# Cloud Computing Platforms

---

- ❖ **Examples:** AWS, Azure, Google Cloud
- ❖ **Programmability:** APIs, microservices, NoSQL, serverless
- ❖ **Networking:** Mostly Client/Server based on REST/JSON, Web Sockets
- ❖ **Centralisation:** centralised control with potential distributed topologies
- ❖ **Integration:** orchestration, API Management, iPaaS
- ❖ **Privacy:** society is being aware of the risk and reacts with GDPR, technical solutions are starting to be proposed

# Cloud Computing Platforms | SOA & Microservices in the APIs world

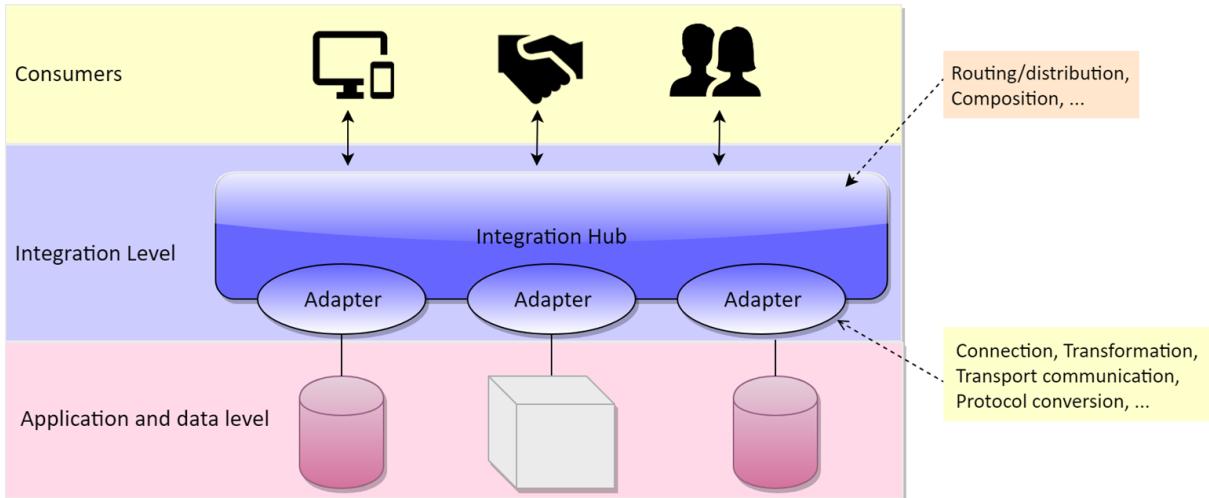


Figure: Standard Enterprise Architecture

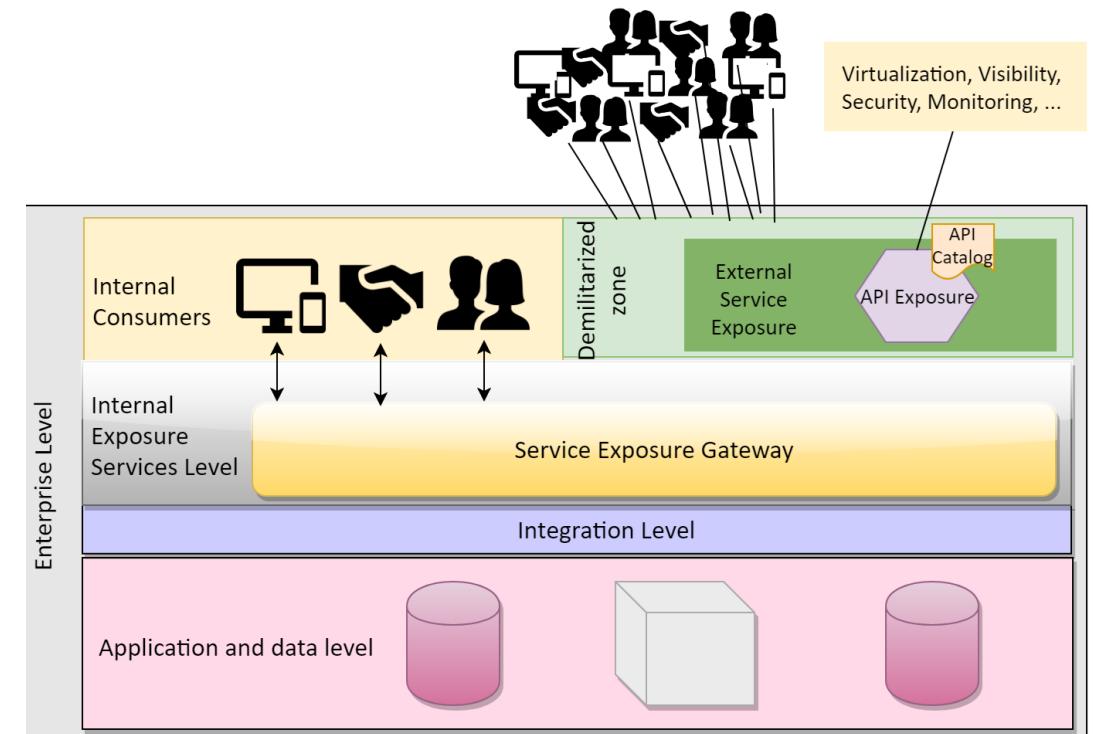
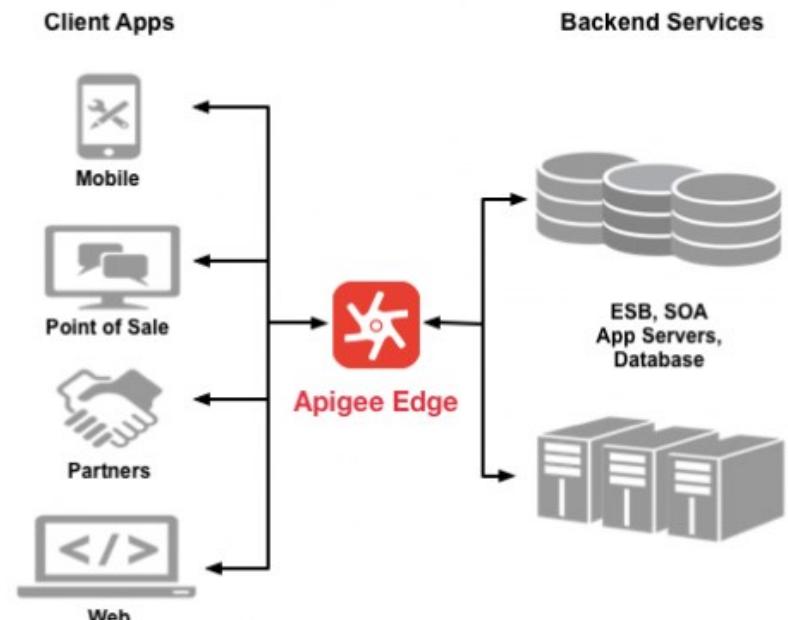
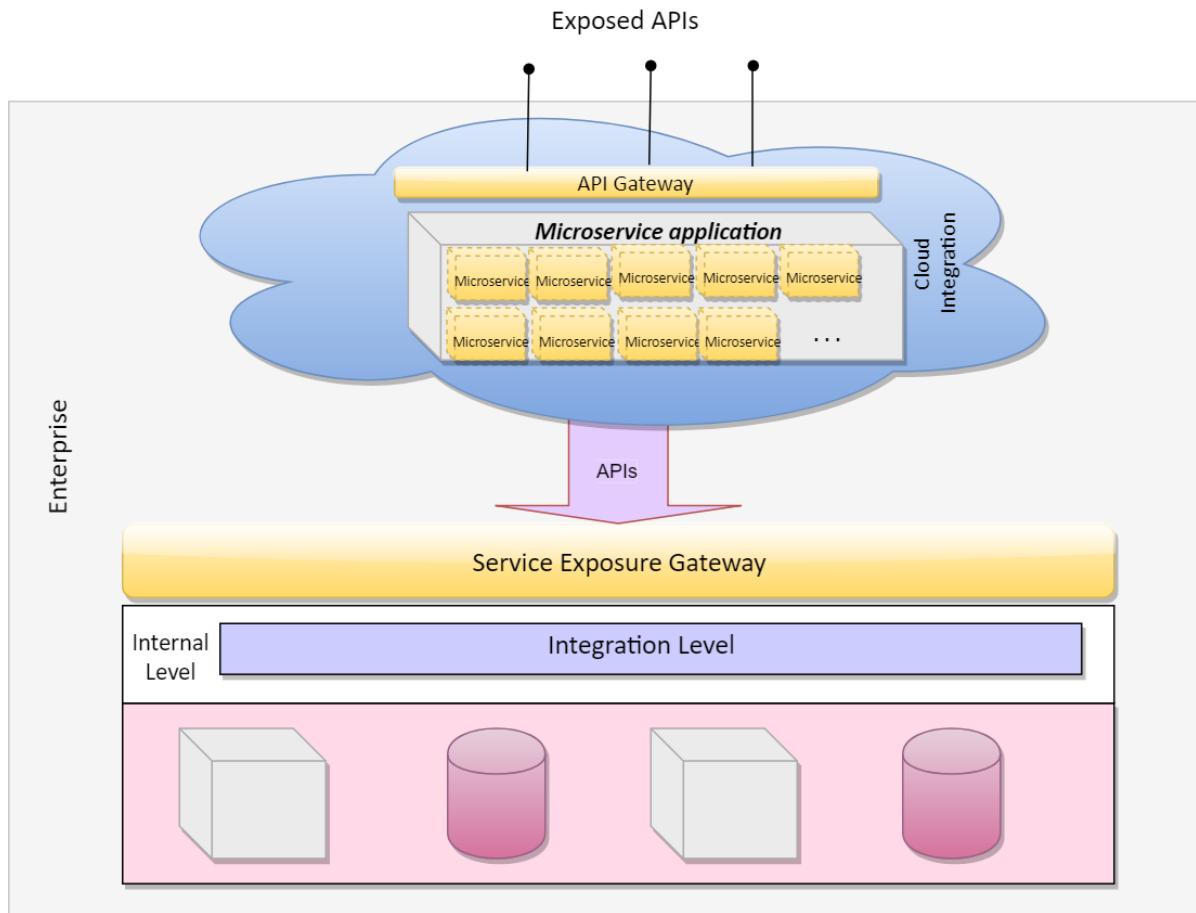


Figure: Exposing Services inside/outside of the enterprise

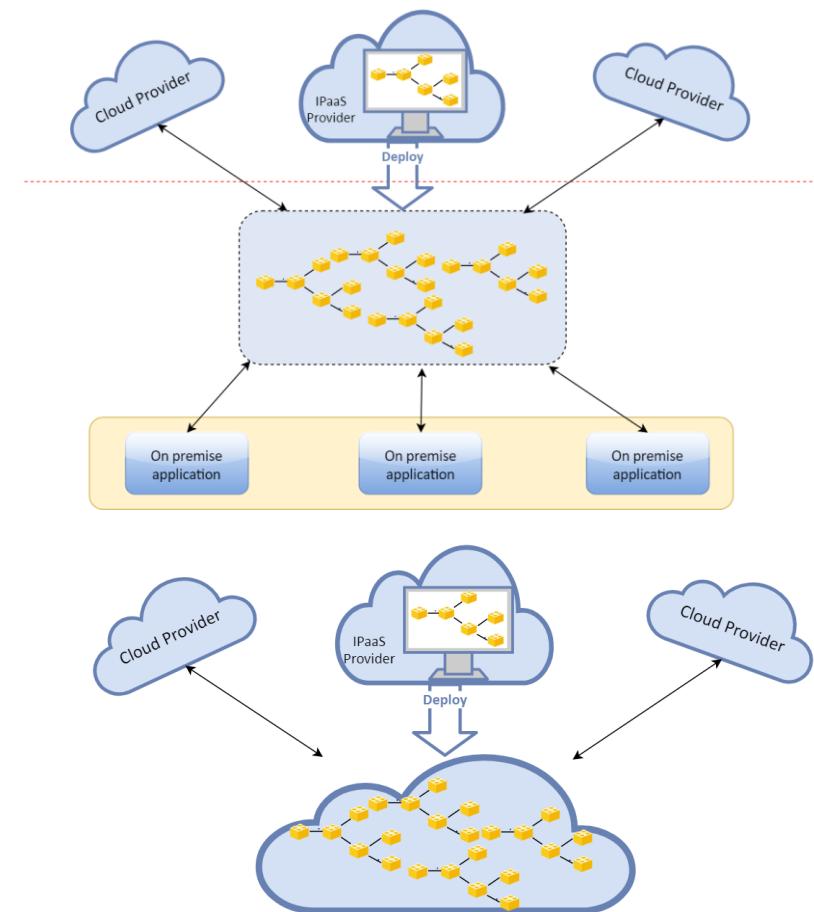
# Cloud Computing Platforms | SOA & Microservices in the APIs world



Example: Apigee handling request for various backend services

# Cloud Computing Platforms | iPaaS

- there is a trend to expose integration functionalities as REST and SOAP services
- iPaaS (Integration Platform as a Service: it “is a suite of cloud services enabling development, execution and governance of integration flows connecting any combination of on premises and cloud-based processes, services, applications and data within individual or across multiple organizations” (Gartner, 2017)



# Cloud Computing Platforms | Serverless

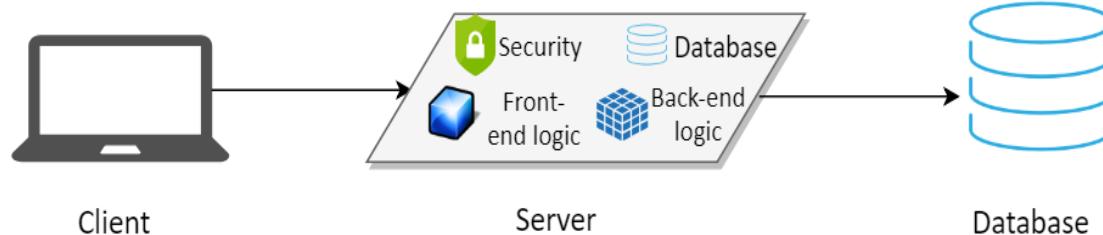


Figure: Traditional architectures

"The microservice community favours an alternative approach: smart endpoints and dumb pipes. Applications built from microservices aim at being as decoupled and as cohesive as possible - they own their own domain logic and act more as filters in the classical Unix sense - receiving a request, applying logic as appropriate and producing a response. These are **choreographed** using simple RESTish protocols rather than complex protocols such as WS-Choreography or BPEL or orchestration by a central tool." [Fow14]

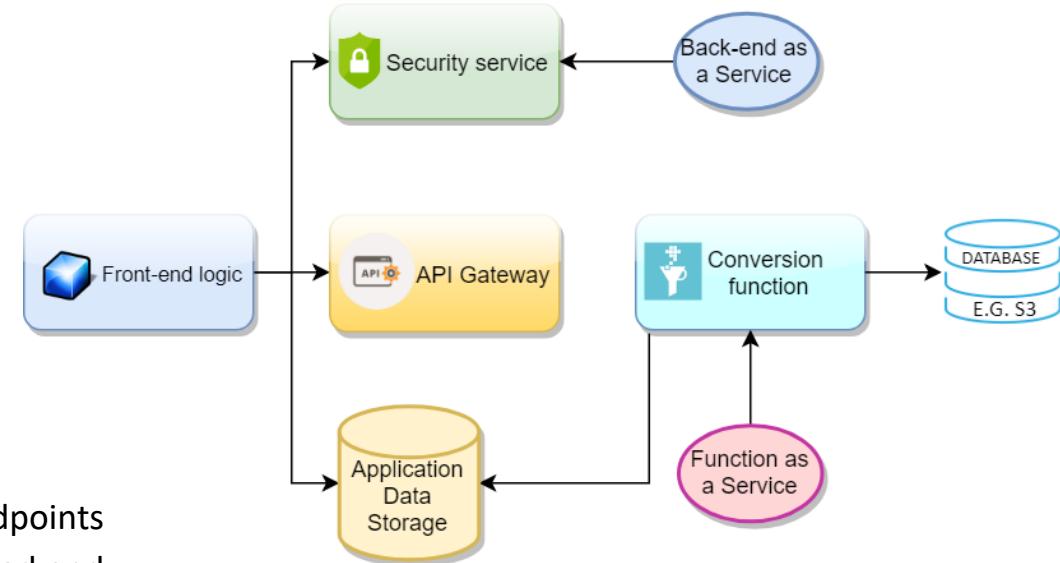


Figure: Serverless Architecture example

# Cloud Computing Platforms | Privacy

---

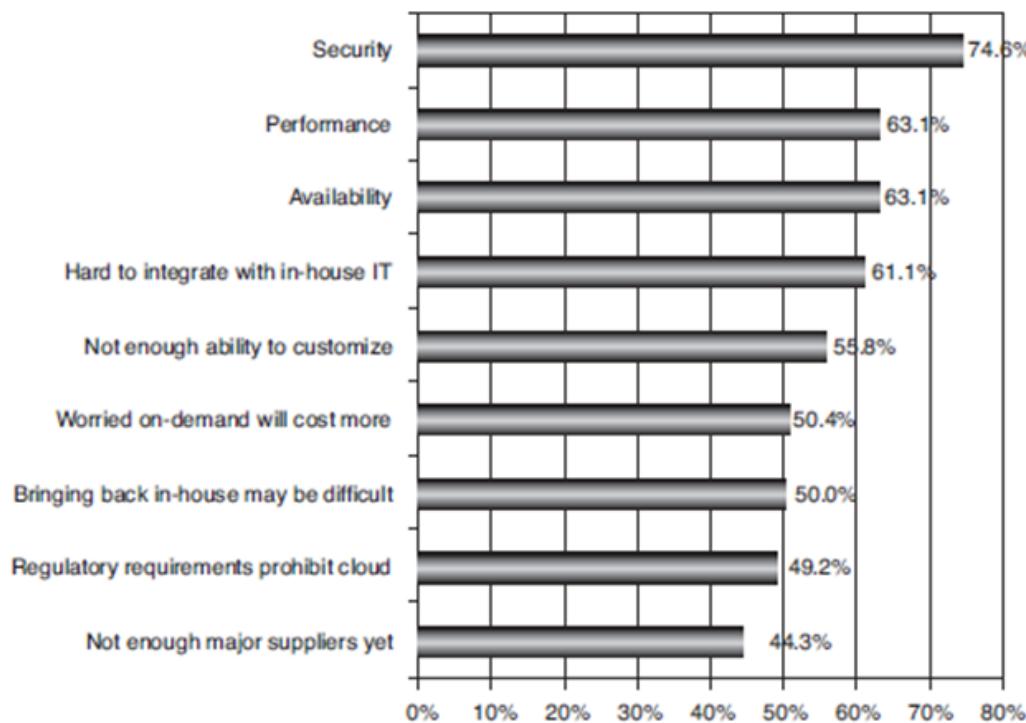
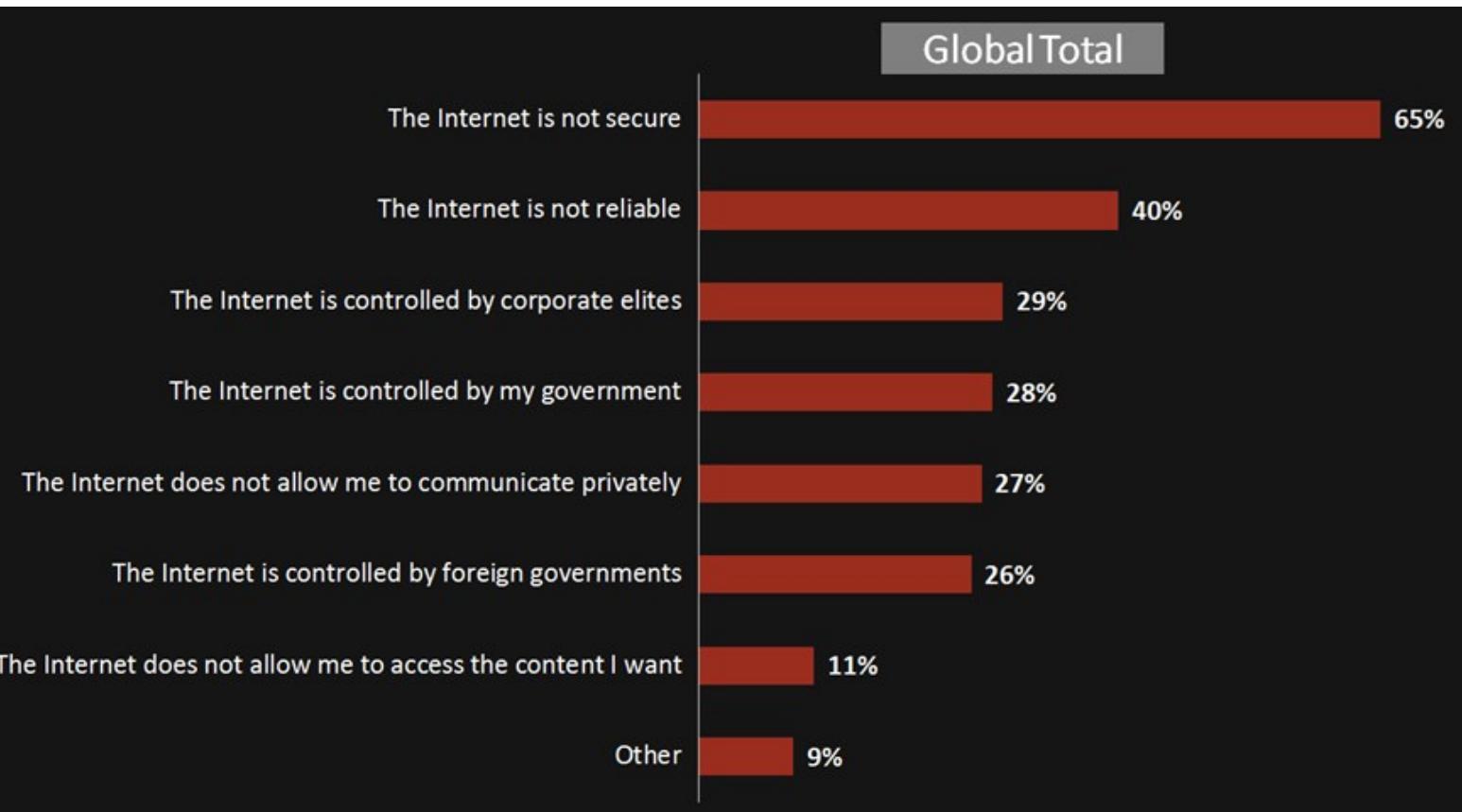


Figure 4.1 Results of IDC's 2009 annual survey of enterprise IT executives regarding concerns about adopting cloud computing for their core applications. Security has remained the top concern for the past three years.

- “But how can anyone trust the cloud?  
–This puts the fate of your company in the hands of third parties!”

# Cloud Computing Platforms | Privacy



Centre for International Governance Innovation (CIGI) - > 24,225 utilizatori de Internet din 24 de tari; Decembrie 23, 2016 - Martie 21, 2017

❖ **Privacy:** society is being aware of the risk and reacts with GDPR, technical solutions are starting to be proposed

# Cloud Computing Platforms | Privacy

- Latest trend in approaching privacy revolves around the Privacy By Design principles (PbD) and their legal interpretation (e.g. GDPR - EU General; Data Protection Regulation)[Cav12]
  - Interpretation of Privacy by Design from a Technical perspective

Proactive not reactive; Preventative not  
remedial

Privacy as the default setting

Privacy embedded into design

Full functionality – positive-sum, not zero-sum

Visibility and transparency – keep it open

End-to-end security – full lifecycle protection

Respect for user privacy – keep it user-centric

obtaining valid consent  
preserve quality of data  
data minimization (obtain only the required data)  
reaction to breaches  
the right to be forgotten

**Data Sovereignty** - the ability of the user to have full control over his data and the entities to which it is shared or revoked

**Data self-sovereignty principle (DSSP)** - any private data is **stored** and handled in ways that preserve ownership information and any access that the data is directly accessible only with the consent of a user or a legally authorized entity



# Personal Assistant Platforms

---

**Characteristic:** reduced role of graphical user-interface

**Examples:** Siri, Google Alo, WeChat, others

**Programmability:** we propose executable choreographies

**Networking:** P2P

**Centralisation:** research area ( decentralised ...)

**Privacy:** research area

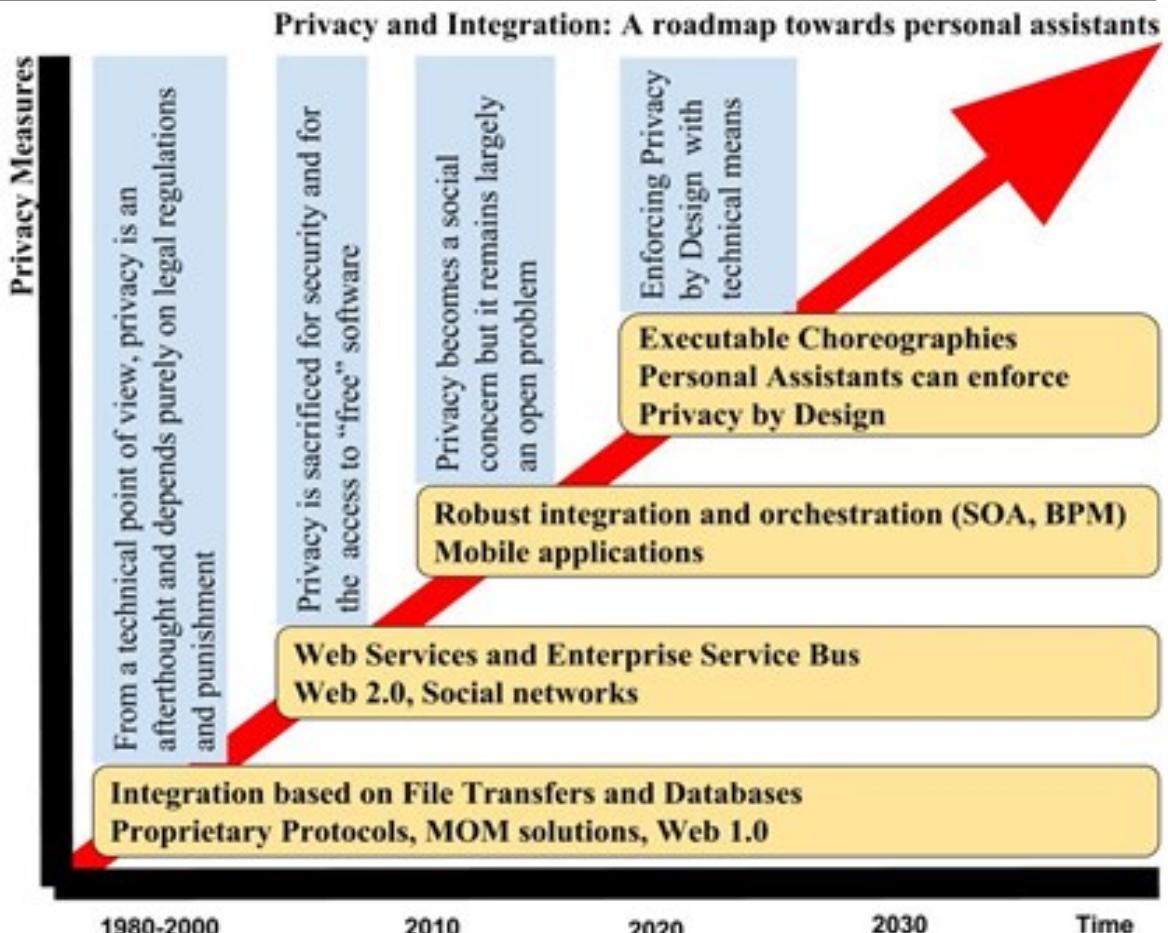


# Personal Assistant Platforms

- Next BIG Platforms
- Sharing Private Data with your Personal Assistants?

From a Risk to Protectors of Privacy

- Implementation: using DSSP and choreographies
- Automates the consent of the user regarding its private data





# Self Souverain Cloud Platforms

---

**Examples:** Bitcoin, Ethereum, IPFS, PrivateSky EDFS

**Programmability:** migrates towards P2P, executable choreographies

**Networking:** P2P

**Centralisation:** decentralised or even distributed

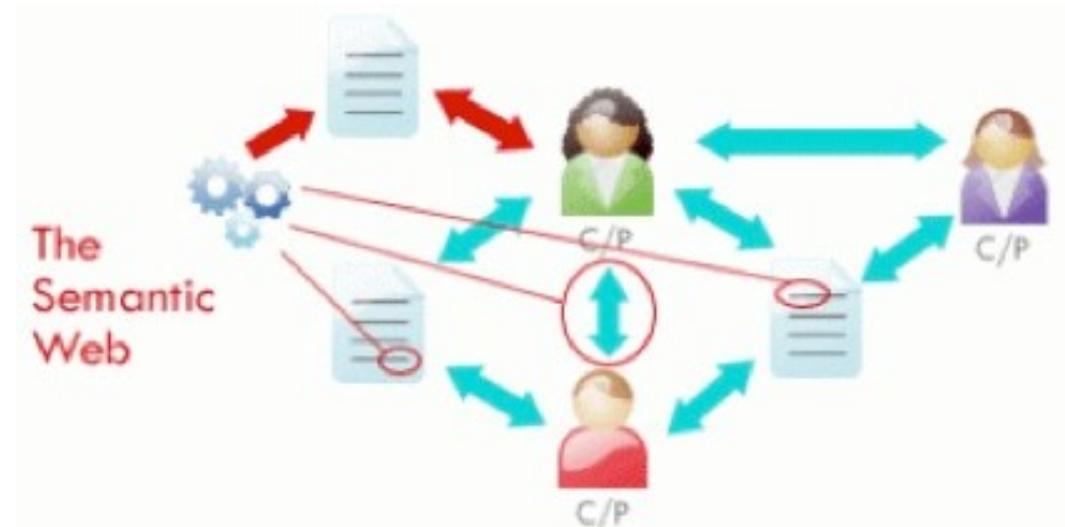
**Integration:** new approaches (e.g. decentralised identities)

**Privacy:** decentralised technical solutions in place, active research area



# Web Platforms | Web 3.0

- ❖ **Examples:** Semantic Web applications
- ❖ **Programmability:** HTTP, HTML5, RDF, OWL, SPARQL
- ❖ **Networking:** Mostly Client/Server based on HTTP
- ❖ **Centralisation:** centralised
- ❖ **Integration:** semantic mediation
- ❖ **Privacy:** research area



[<http://airccse.org/journal/ijwest/papers/3112ijwest01.pdf>]

# Web Platforms | Web 3.0

---

- Semantic Web technologies: potential solution for data integration
- One of the main goals was to allow integration of disparate data sources or semantic mediation

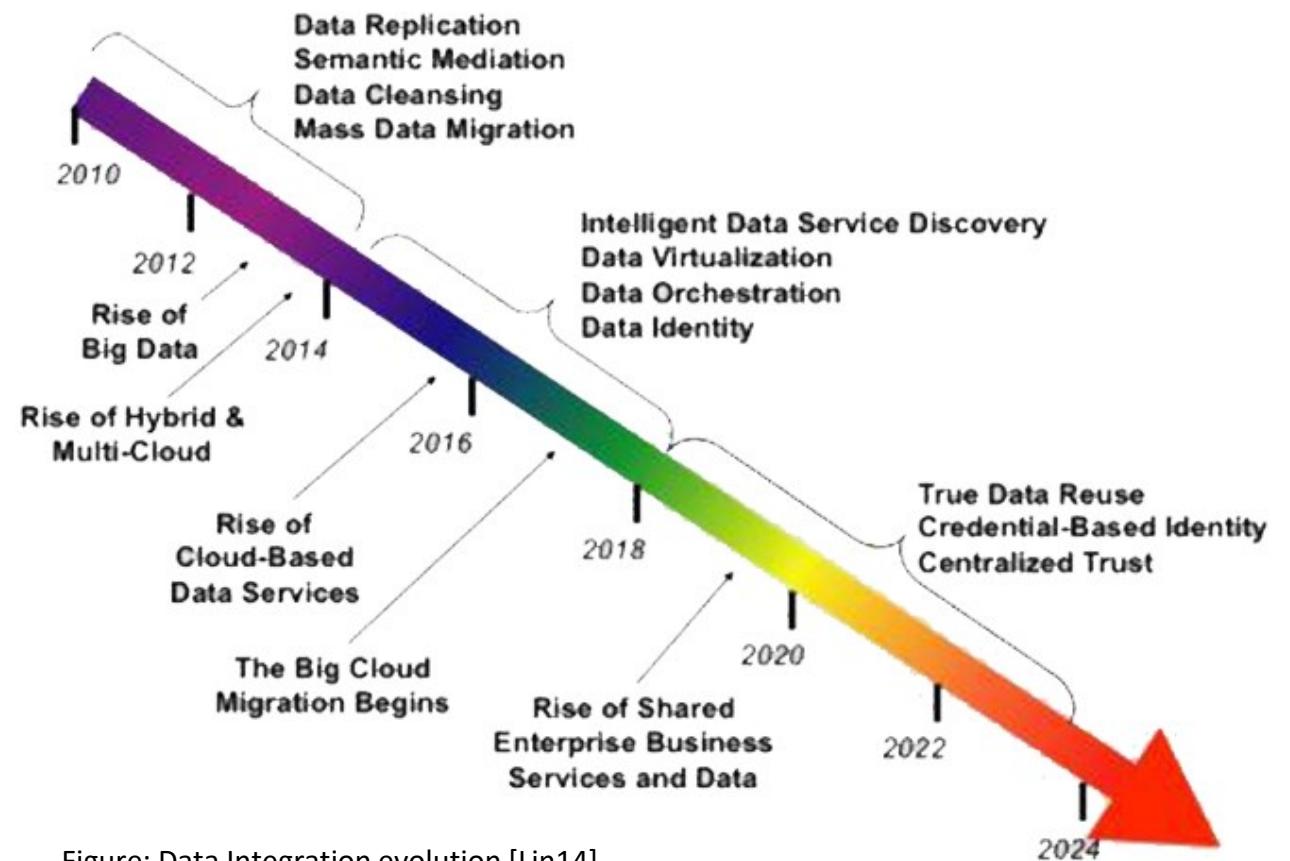


Figure: Data Integration evolution [Lin14]



# Mobile Platforms or Web 4.0

---

- ❖ **Characteristic:** mobile connection to the Edge/Fog Computing & Cloud Platforms <=> “virtual world in real-time” [DMCA13]
- ❖ **Examples:** iOS, Android
- ❖ **Programmability:** variants of enterprise technologies modified for smaller screens, IoT protocols
- ❖ **Networking:** Mostly Client/Server based on REST/JSON, 5G (SDN)
- ❖ **Centralization:** centralized ... decentralized
- ❖ **Integration:** web services
- ❖ **Privacy:** society is being aware of the risks and reacts with GDPR, one of most important challenge for 5G



# Fog/Edge Computing

- Fog Computing and Edge Computing appeared to address the need for optimisation of Cloud Computing systems by processing at the “edge of the network”, as close as possible to the data source
- In Edge Computing the **actual processing** is made by the IoT devices (*large volume real-time data processing*)
- In Fog Computing facilitates the operation of **compute, storage and networking services** between end devices and cloud computing data centers

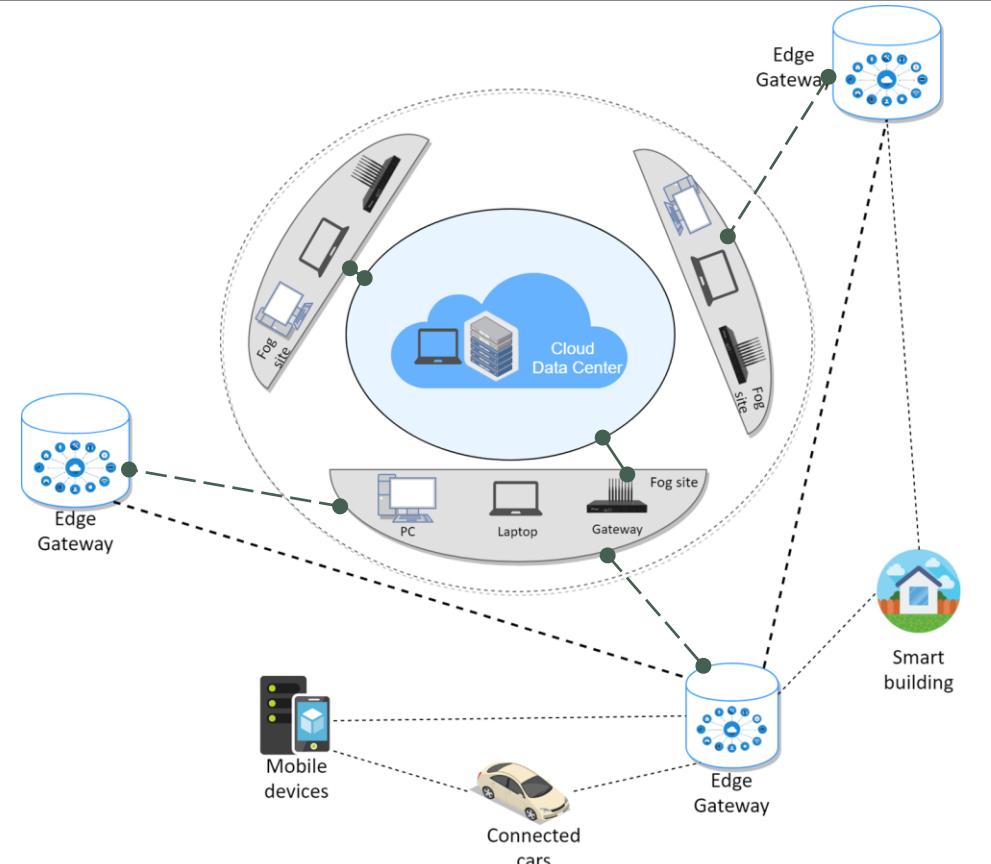
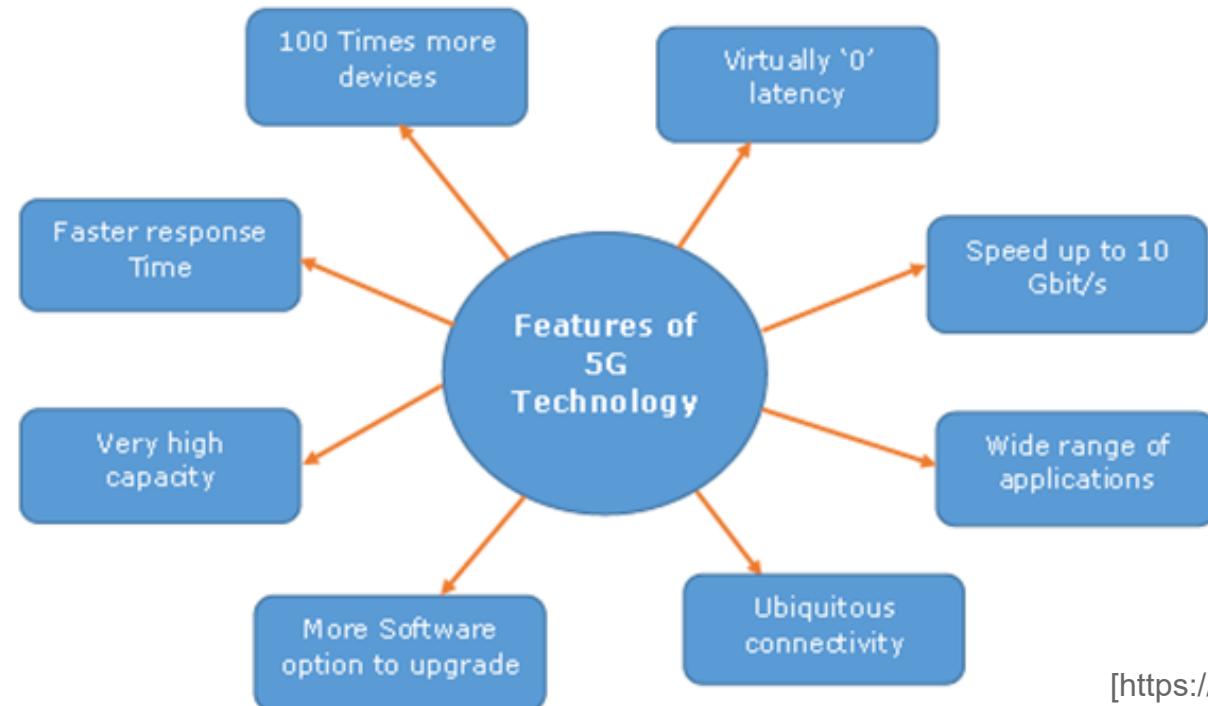


Figure: Relation: Cloud Computing- Fog Computing - Edge Computing

# Mobile Platforms or Web 4.0|5G

---

- ❖ Ultra-high speed (till 10Gbps), it is potential enough to change the meaning of a cell phone usability <=> laptop functionalities



[<https://www.tutorialspoint.com/5g>]

# 4G versus 5G

- High increased peak bit rate
- Larger data volume per unit area (i.e. high system spectral efficiency)
- High capacity to allow more devices connectivity concurrently and instantaneously
- Lower battery consumption
- Better connectivity irrespective of the geographic region, in which you are
- Larger number of supporting devices
- Lower cost of infrastructural development
- Higher reliability of the communications

**Obs. 5G will initially operate in conjunction with existing 4G networks**

# 4G versus 5G

Obs. 5G will initially operate in conjunction with existing 4G networks

- When a 5G connection is established, the User Equipment (or device) will connect to both the 4G network to provide the control signaling and to the 5G network to help provide the fast data connection by adding to the existing 4G capacity.
- Where there is limited 5G coverage, the data is carried as it is today on the 4G network providing the continuous connection. Essentially with this design, the 5G network is complementing the existing 4G network.

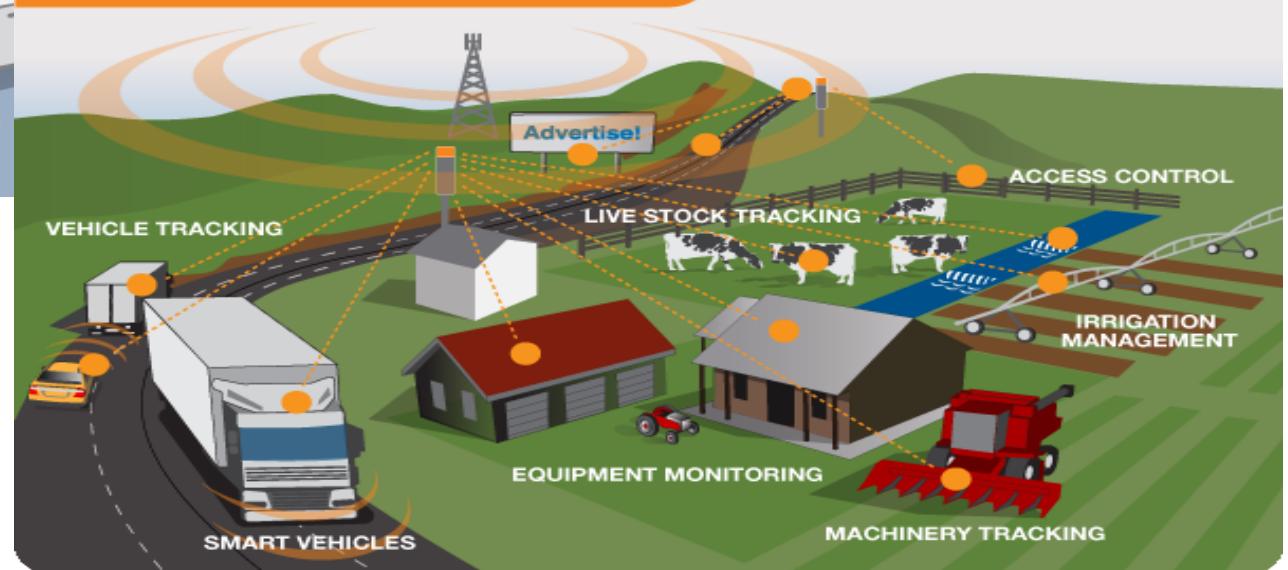


# 5G

## 5G CONNECTING THE COMMUNITY



## CONNECTED FARMS



[<http://www.emfexplained.info/?ID=25916>]

# 5G Architecture

---

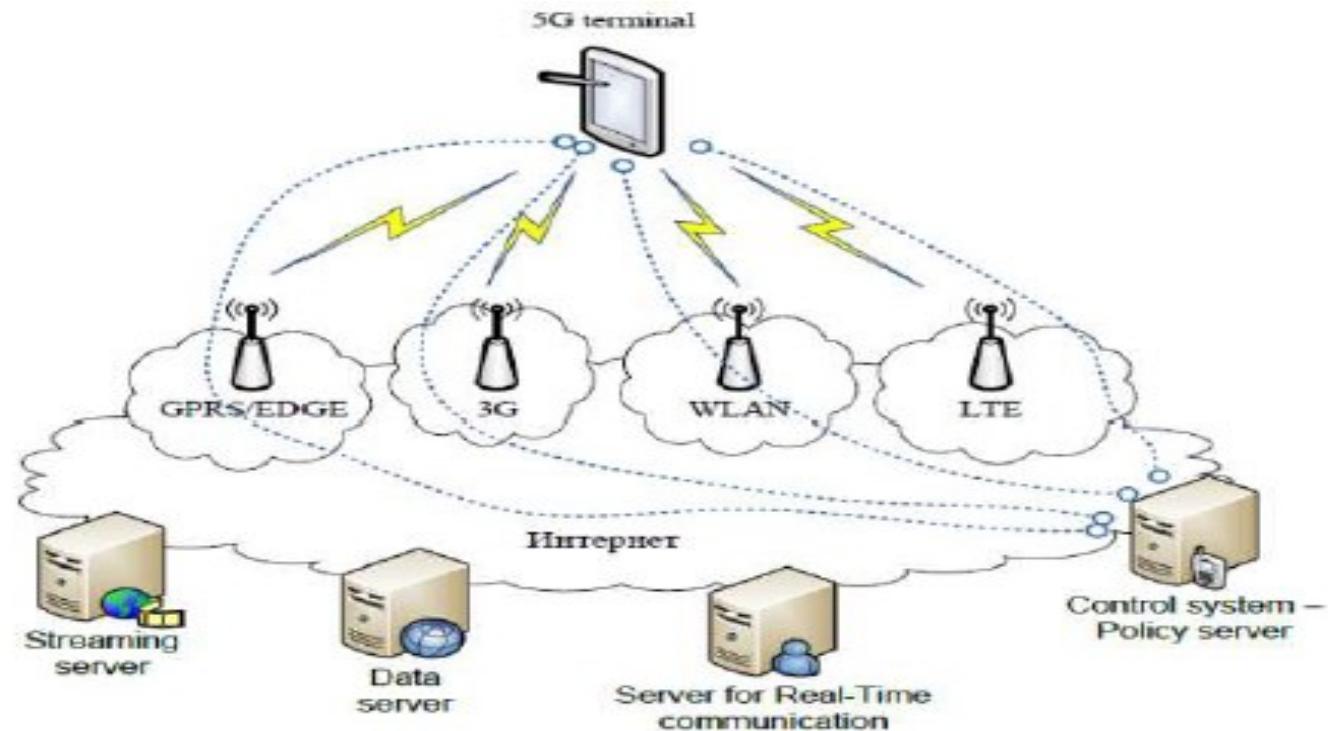
5G uses radio waves or radio frequency (RF) energy to transmit and receive voice and data connecting our communities

## Cognitive radio technology

- ability of devices to identify their geographical location as well as weather, temperature
- acts as a transceiver (beam) that perceptively can catch and respond radio signals in its operating environment; it promptly distinguishes the changes in its environment and hence respond accordingly to provide uninterrupted quality service
- the system model of 5G is entirely IP based model designed for the wireless and mobile networks
- The system comprising of a main user terminal and then a number of independent and autonomous radio access technologies

# 5G Architecture

Cognitive radio technology



# 5G Architecture

---

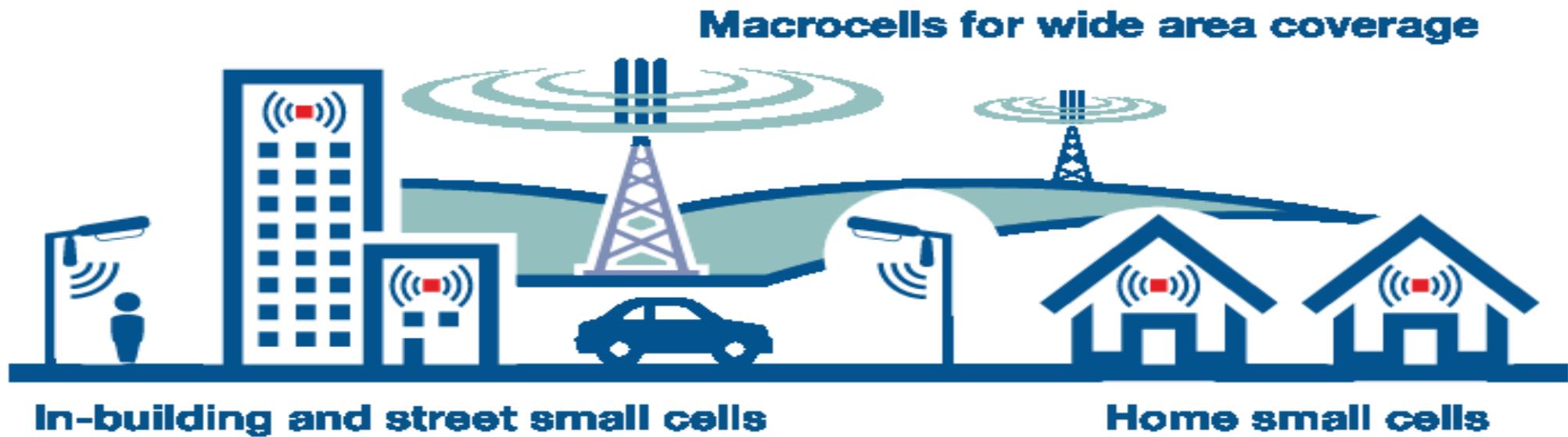
A mobile network has two main components: **Radio Access Network** and **Core Network**

**The Radio Access Network** - consists of various types of facilities including *small cells, towers, masts* and *dedicated in-building and home systems* that connect mobile users and wireless devices to the main core network.

- **Small cells** will be a major feature of 5G networks particularly at the new millimeter wave (mmWave) frequencies where the connection range is very short.
  - To provide a continuous connection, small cells will be distributed in clusters depending on where users require connection which will complement the macro network that provides wide-area coverage.
- 5G macro cells will use **MIMO** (multiple input, multiple output) antennas that have multiple elements or connections to send and receive more data simultaneously.
  - The benefit to users is that more people can simultaneously connect to the network and maintain high throughput.
  - MIMO antennas are often referred to as ‘Massive MIMO’ due to the large number of multiple antenna elements and connections however the physical size is similar to existing 3G and 4G base station antennas.

# 5G Architecture

- A mobile network has two main components, the ‘Radio Access Network’ and the ‘Core Network’.
- ~~The Radio Access Network~~ Small cells will be a major feature of 5G networks particularly at the new millimetre wave



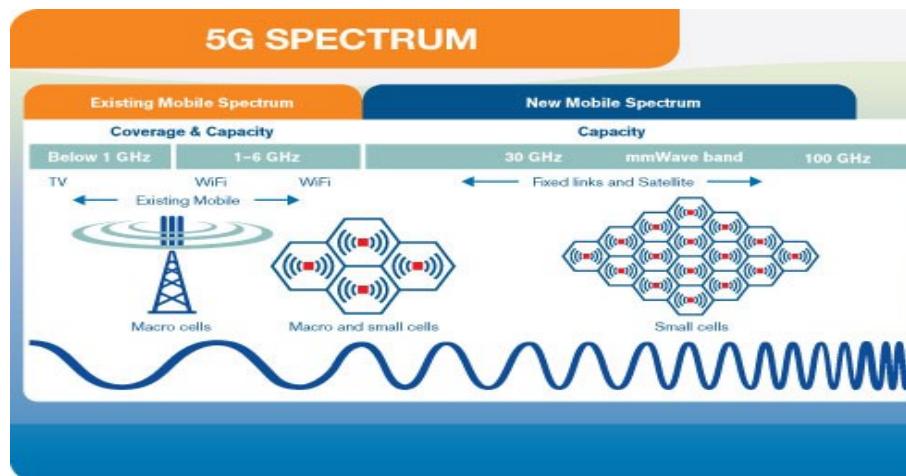
- ~~BETTER CONNECTION - ALWAYS CONNECTED~~
- 5G networks are designed to work in conjunction with 4G networks using a range of macro cells, small cells and dedicated in-building systems.

# 5G Architecture

A mobile network has two main components, the '**Radio Access Network**' and the '**Core Network**'.

## INCREASED SPECTRUM – GREATER CAPACITY, MORE USERS AND FASTER SPEED

- Initial frequency bands for 5G are proposed around 600-700 MHz, 3-4 GHz, 26-28 GHz and 38-42 GHz which will add significantly more capacity compared to the current mobile technologies. The additional spectrum and greater capacity will enable more users, more data and faster connections. It is also expected that there will be future reuse of existing low band spectrum for 5G as legacy networks decline in usage and to support future use cases.
- The increased spectrum in the millimeter (mm) Wave band above 30 GHz will provide localized coverage as they only operate over short line of sight distances. Future 5G deployments may use mmWave frequencies in bands up to 86 GHz.



# 5G Architecture

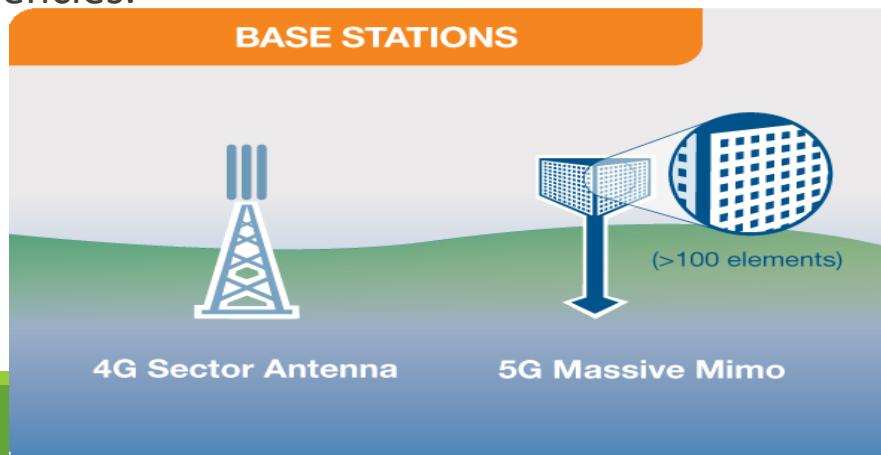
---

A mobile network has two main components, the '**Radio Access Network**' and the '**Core Network**'.

**MASSIVE MIMO** - multiple element base station - greater capacity, multiple users, faster data

**The Radio Access Network:** 5G macro cells will use MIMO

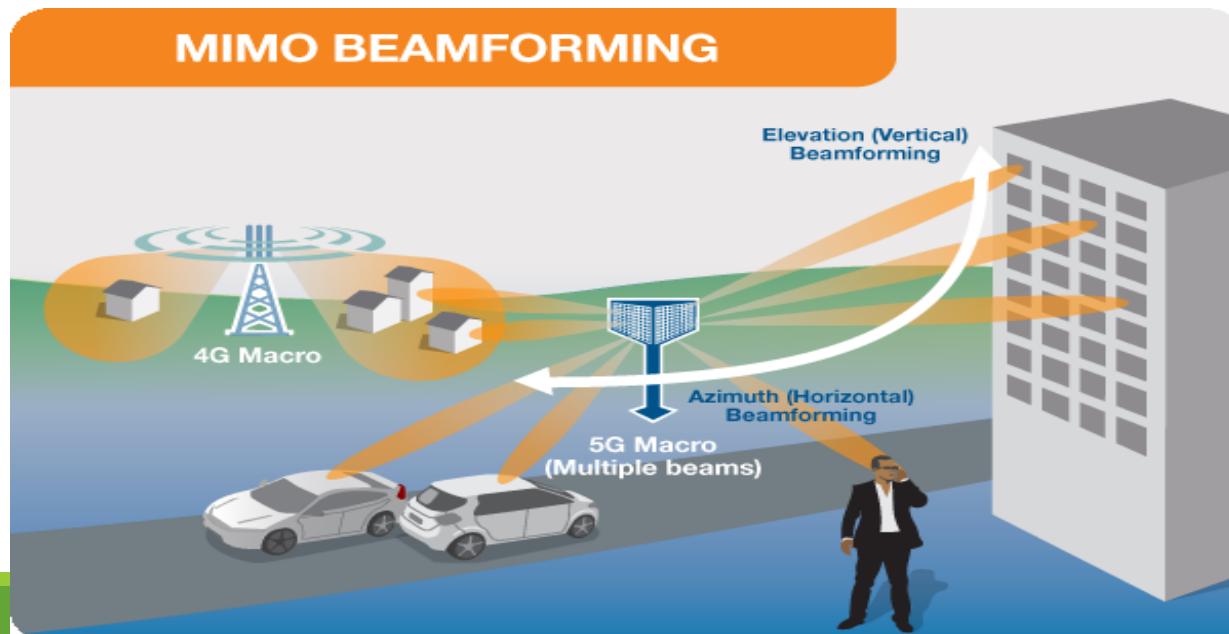
- The overall physical size of the 5G Massive MIMO antennas will be similar to 4G, however with a higher frequency, the individual antenna element size is smaller allowing more elements (in excess of 100) in the same physical case
- 5G User Equipment including mobile phones and devices will also have MIMO antenna technology built into the device for the mmWave frequencies.



# 5G Architecture

A mobile network has two main components, the '**Radio Access Network(RAN)**' and the '**Core Network (CN)**'.

**MIMO - Beam Steering:** Beam steering is a technology that allows the Massive MIMO base station antennas to direct the radio signal to the users and devices rather than in all directions. The beam steering technology uses advanced signal processing algorithms to determine the best path for the radio signal to reach the user. This increases efficiency as it reduces interference (unwanted radio signals).



# 5G Architecture

---

A mobile network has two main components, the ‘Radio Access Network’ and the ‘**Core Network**’.

**The Core Network** - is the mobile exchange and data network that manages all of the mobile voice, data and internet connections. For 5G, the ‘core network’ is being redesigned to better integrate with the internet and cloud based services and also includes distributed servers across the network improving response times (reducing latency).

- **Network Slicing** – enables a smart way to segment the network for a particular industry, business or application. For example emergency services could operate on a network slice independently from other users.
- **Network Function Virtualization (N VF)** - is the ability to instantiate network functions in real time at any desired location within the operator’s cloud platform. Network functions that used to run on dedicated hardware for example a firewall and encryption at business premises can now operate on software on a virtual machine. N VF is crucial to enable the speed efficiency and agility to support new business applications and is an important technology for a 5G ready core.

# 5G Architecture

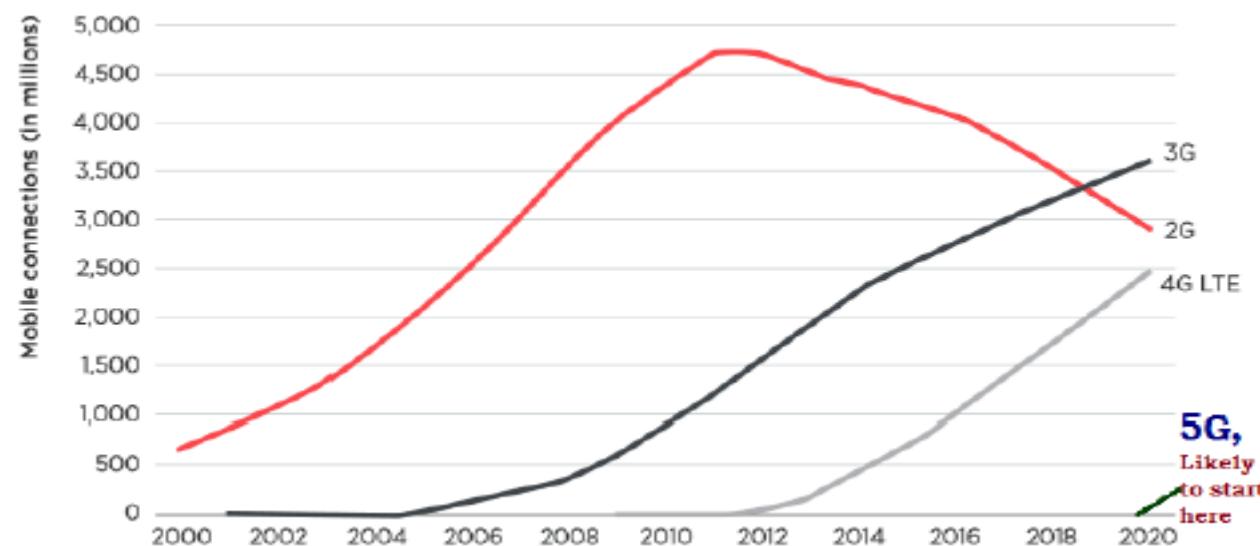
- **LOWER LATENCY** <= significant advances in mobile device technology and mobile network architecture (RAN and CN).
- 

Technology	Response time (milliseconds)
4G - LTE systems	20-30 ms
5G - enhanced mobile broadband	4-5 ms
5G - URLLC (Ultra Reliable Low Latency Communications) systems	1 ms

- **Core Network Changes** With the redesigned core network, signalling and distributed servers, a key feature is to move the content closer to the end user and to shorten the path between devices for critical applications. Good examples are video on demand streaming services where it is possible to store a copy or 'cache' of popular content in local servers, so the time to access is quicker.
- **Radio Access Network Changes** - Low latency and high reliability over the air interface requires new radio techniques to minimise the time delays through the radio within a few TTIs (time transmit intervals) along with robustness and coding improvements to achieve high degrees of reliability (e.g. one message is delayed or lost in every billion).

# 5G Time Period Required

it is expected that the time period required for the 5G technology development and its implementation is about five years more from now (by 2019)



researchers are anticipating that this technology will be in use until 2040s

# Applications of 5G

---

It will make unified global standard for all.

Network availability will be everywhere and will facilitate people to use their computer and such kind of mobile devices anywhere anytime.

Because of the IPv6 technology, visiting care of mobile IP address will be assigned as per the connected network and geographical position.

Its application will make world real Wi Fi zone.

Its cognitive radio technology will facilitate different version of radio technologies to share the same spectrum efficiently.

Its application will facilitate people to avail radio signal at higher altitude as well

# Applications of 5G

---

In comparison to previous radio technologies, 5G has following advancement –

- Practically possible to avail the super speed i.e. 1 to 10 Gbps.
- Latency will be 1 millisecond (end-to-end round trip).
- 1,000x bandwidth per unit area.
- Feasibility to connect 10 to 100 number of devices.
- Worldwide coverage.
- About 90% reduction in network energy usage.
- Battery life will be much longer.
- Whole world will be in *wifi* zone.

# Important Advantages

---

High resolution and bi-directional large bandwidth shaping.

Technology to gather all networks on one platform.

More effective and efficient.

Technology to facilitate subscriber supervision tools for the quick action.

Most likely, will provide a huge broadcasting data (in Gigabit), which will support more than 60,000 connections.

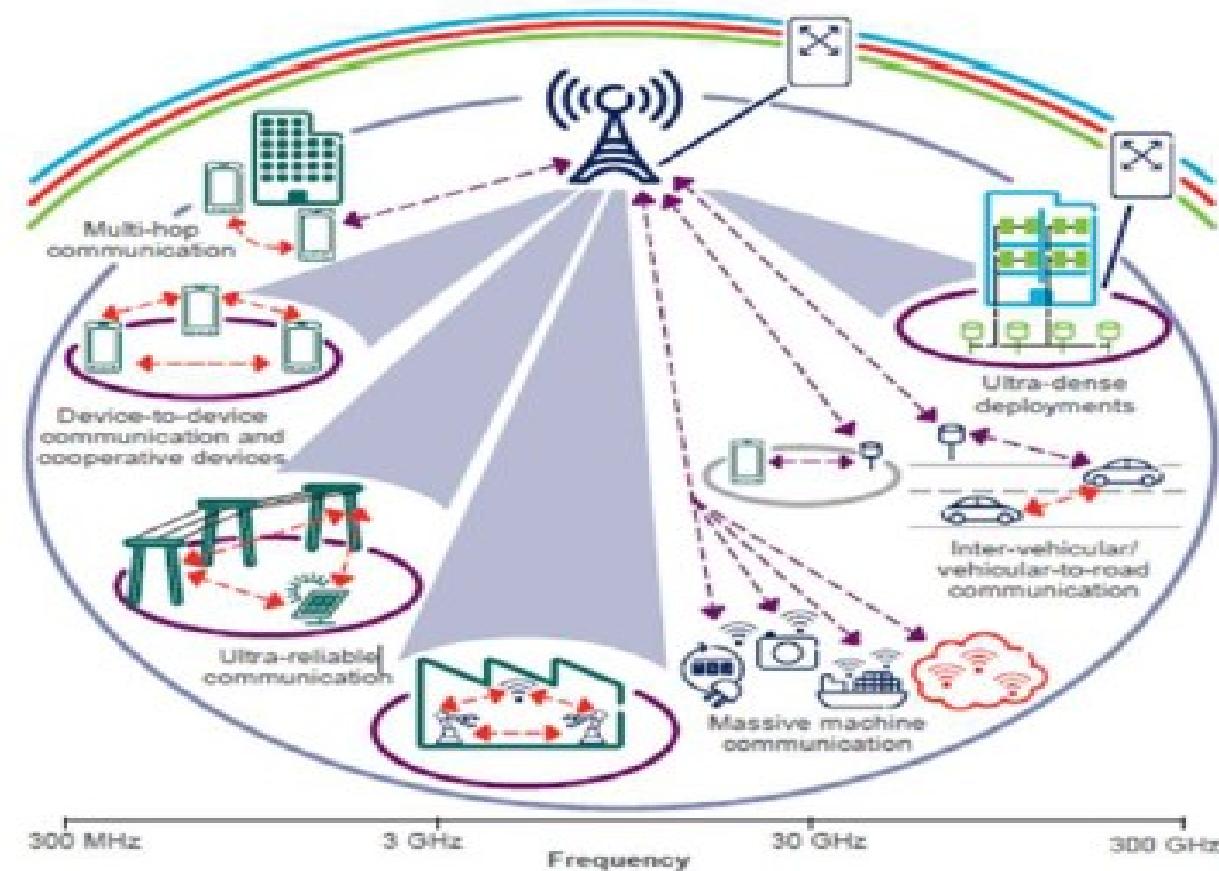
Easily manageable with the previous generations.

Technological sound to support heterogeneous services (including private network).

Possible to provide uniform, uninterrupted, and consistent connectivity across the world.

# Important Advantages

---



# Disadvantages of 5G Technology

---

Technology is still under process and research on its viability is going on.

The speed, this technology is claiming seems difficult to achieve (in future, it might be) because of the incompetent technological support in most parts of the world

Many of the old devices would not be competent to 5G, hence, all of them need to be replaced with new one — expensive deal.

Developing infrastructure needs high cost.

Security and privacy issue yet to be solved.



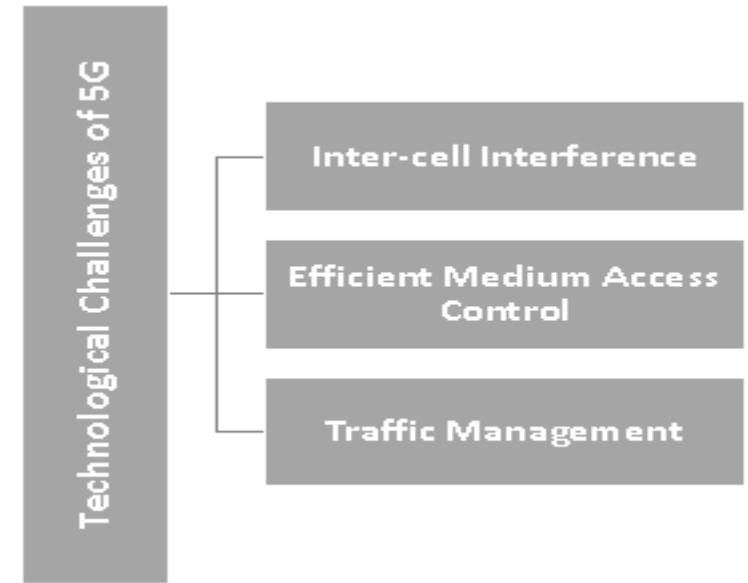
# 5G - Technological Challenges

---

**Inter-cell Interference** – This is one of the major technological issues that need to be solved. There are variations in size of traditional macro cells and concurrent small cells that will lead to interference.

**Efficient Medium Access Control** – In a situation, where dense deployment of access points and user terminals are required, the user throughput will be low, latency will be high, and hotspots will not be competent to cellular technology to provide high throughput. It needs to be researched properly to optimize the technology.

**Traffic Management** – In comparison to the traditional human to human traffic in cellular networks, a great number of Machine to Machine (M2M) devices in a cell may cause serious system challenges i.e. radio access network (RAN) challenges, which will cause overload and congestion.



# 5G – Common Challenges

---

**Multiple Services** – Unlike other radio signal services, 5G would have a huge task to offer services to heterogeneous networks, technologies, and devices operating in different geographic regions. So, the challenge is of standardization to provide dynamic, universal, user-centric, and data-rich wireless services to fulfil the high expectation of people.

**Infrastructure** – Researchers are facing technological challenges of standardization and application of 5G services.

**Communication, Navigation, & Sensing** – These services largely depend upon the availability of radio spectrum, through which signals are transmitted. Though 5G technology has strong computational power to process the huge volume of data coming from different and distinct sources, but it needs larger infrastructure support.

# 5G – Common Challenges

---

**Security and Privacy** – This is one of the most important challenges that 5G needs to ensure the protection of personal data. 5G will have to define the uncertainties related to security threats including trust, privacy, cybersecurity, which are growing across the globe.

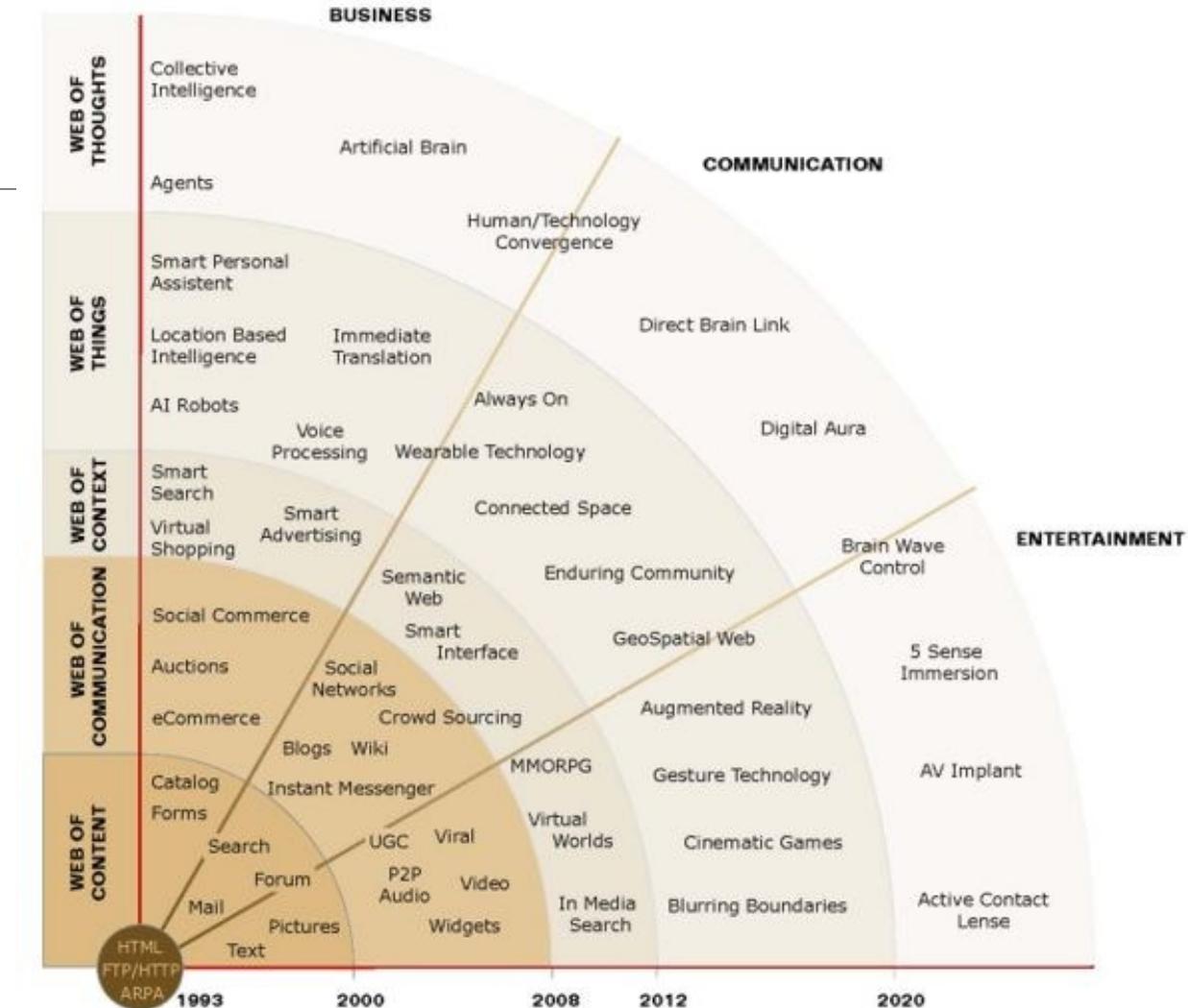
**Legislation of Cyberlaw** – Cybercrime and other fraud may also increase with the high speed and ubiquitous 5G technology. Therefore, legislation of the Cyberlaw is also an imperative issue, which largely is governmental and political (national as well as international issue) in nature.



# Web Platforms Trends

## Web 5.0 - The Telepathic Web (after 2030)

- “Brain implants: ability to communicate with the internet through thought, to think of a question and open up a web page
- Payments will be paid for with a microchip in the brain or the hand and all devices will be connected to the internet” [Web5.0]



# Bibliography

---

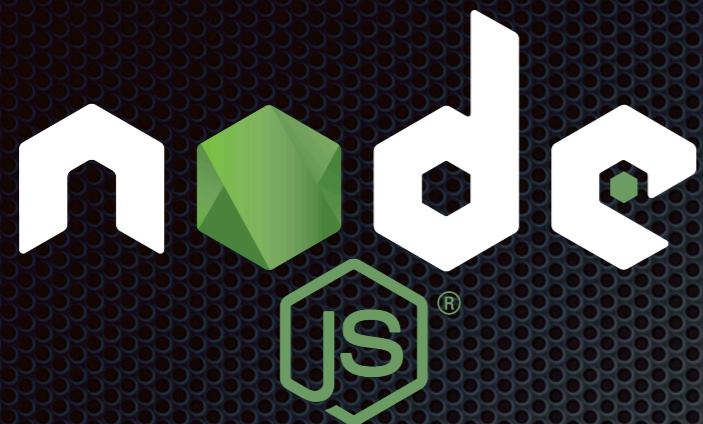
- Tim Berners-Lee – [http://www.youtube.com/watch?v=OM6XIIcm\\_qo](http://www.youtube.com/watch?v=OM6XIIcm_qo)
- [Erl06] Thomas Erl, Service-Oriented Architecture: Concepts, Technology, and Design" (792 pages, Hardcover, ISBN: 0131858580, Prentice Hall/Pearson PTR) (2006)
- [Erl09] T.Erl, SOA Design Patterns. Prentice Hall, 2009
- [Fos08] Ian Foster, Yong Zhao, Ioan Raicu, Shiyong Lu, Cloud Computing and Grid Computing 360-Degree Compared,<https://arxiv.org/abs/0901.0131>
- [PS12] Primavera de Filippi, Smari McCarthy. Cloud Computing : Centralization and Data Sovereignty. European Journal of Law and Technology, University of Warwick, 2012, 3 (2). fffhal-00746065f - <https://hal.archives-ouvertes.fr/hal-00746065>
- [Fow14] Martin Fowler, Microservices - a definition of this new architectural term. Available: <https://martinfowler.com/tags/microservices.html> (2014)
- [Cav 12] Cavoukian, Ann, Privacy by design: origins, meaning, and prospects for assuring privacy and trust in the information era, Privacy protection measures and technologies in business organizations: aspects and standards, IGI Global, pg. 170-208 (2012)
- <https://flatworldbusiness.wordpress.com/flat-education/Previously/web-1-0-vs-web-2-0-vs-web-3-0-a-bird-eye-on-the-definition/>
- Web 5.0: the future of emotional competences in higher education,
- <https://www.postscapes.com/internet-of-things-examples/>
- [Lin14] David S. Linthicum, The next generation of data integration technology for the changing cloud market. Available: <https://gigaom.com/2014/09/01/the-next-generation-of-data-integration-technology-for-the-changing-cloud-market/> (2014)
- [https://www.sdxcentral.com/articles/news/databases-are-shifting-to-the-cloud-says-gartner/2019/07/?hit=472e7469-8ebb-47f9-bd0f-9fba18ab5efb&utm\\_campaign=push&utm\\_content=sdxcentral+breaking+news&utm\\_medium=referral&utm\\_source=onesignal](https://www.sdxcentral.com/articles/news/databases-are-shifting-to-the-cloud-says-gartner/2019/07/?hit=472e7469-8ebb-47f9-bd0f-9fba18ab5efb&utm_campaign=push&utm_content=sdxcentral+breaking+news&utm_medium=referral&utm_source=onesignal)
- [DMCA13] <https://link.springer.com/article/10.1007%2Fs40196-013-0016-5#CR21>
- <https://www.winsystems.com/cloud-fog-and-edge-computing-whats-the-difference/>
- International Journal of Web and Semantic Technology – EVOLUTION OF THE WORLD WIDE WEB: FROM WEB 1.0 TO WEB 4.0 – <http://airccse.org/journal/ijwest/papers/3112ijwest01.pdf>
- [Web5.0] Web 5.0: the future of emotional competences in higher education, <https://link.springer.com/article/10.1007%2Fs40196-013-0016-5#CR21>

“The two most important days in your life are the day you are born  
and the day you find out why.” (Mark Twain)

---

# Thank you!





# Programming in Cloud with Node.js

Georgiana Calancea  
Ioana Bogdan



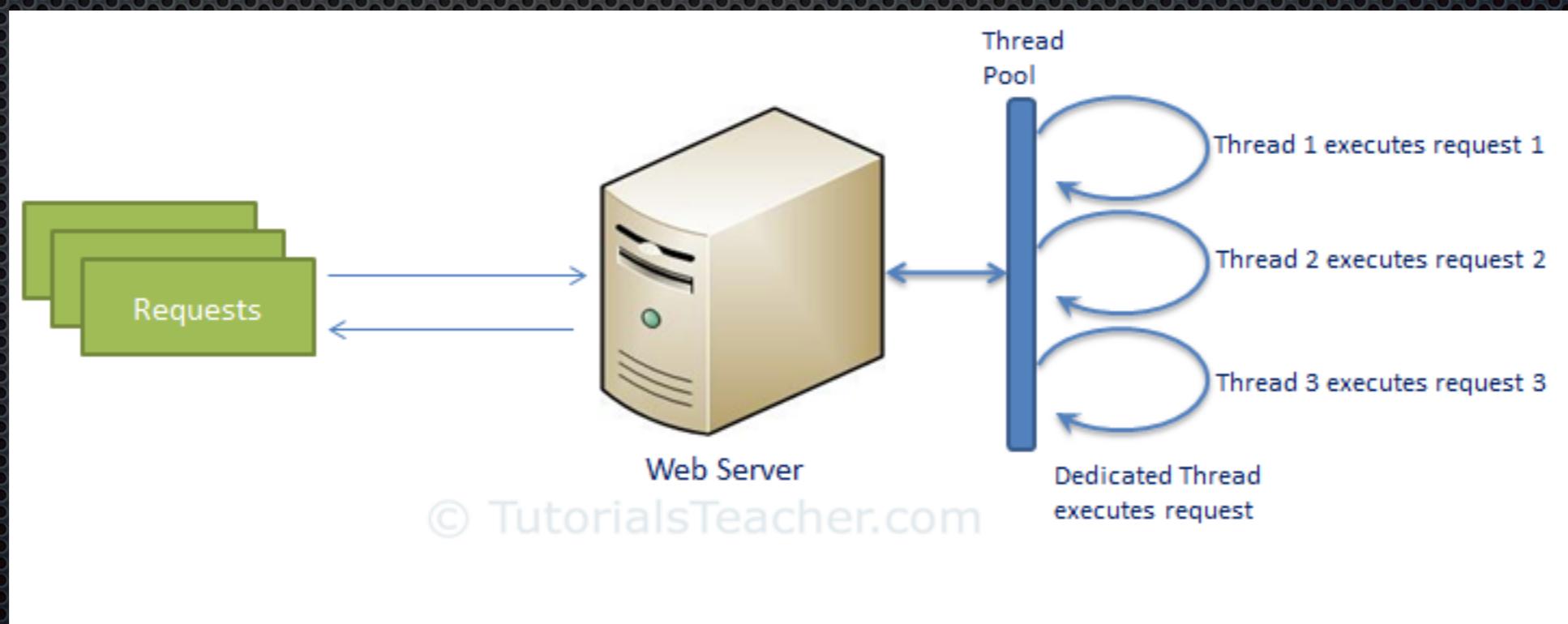
# What is Node.js?



- open-source server side runtime environment built on Chrome's V8 JavaScript engine;
- *event driven, non-blocking (asynchronous) I/O and cross-platform* runtime environment for building highly scalable server-side application using JavaScript;
- used to build different types of applications such as command line application, web application, real-time chat application, REST API server;

# Traditional Web Server Model

- each request is handled by a dedicated thread from the thread pool;
- if no thread is available in the thread pool at any point of time then the request waits till the next available thread;
- a dedicated thread executes a particular request and does not return to thread pool until it completes the execution and returns a response;



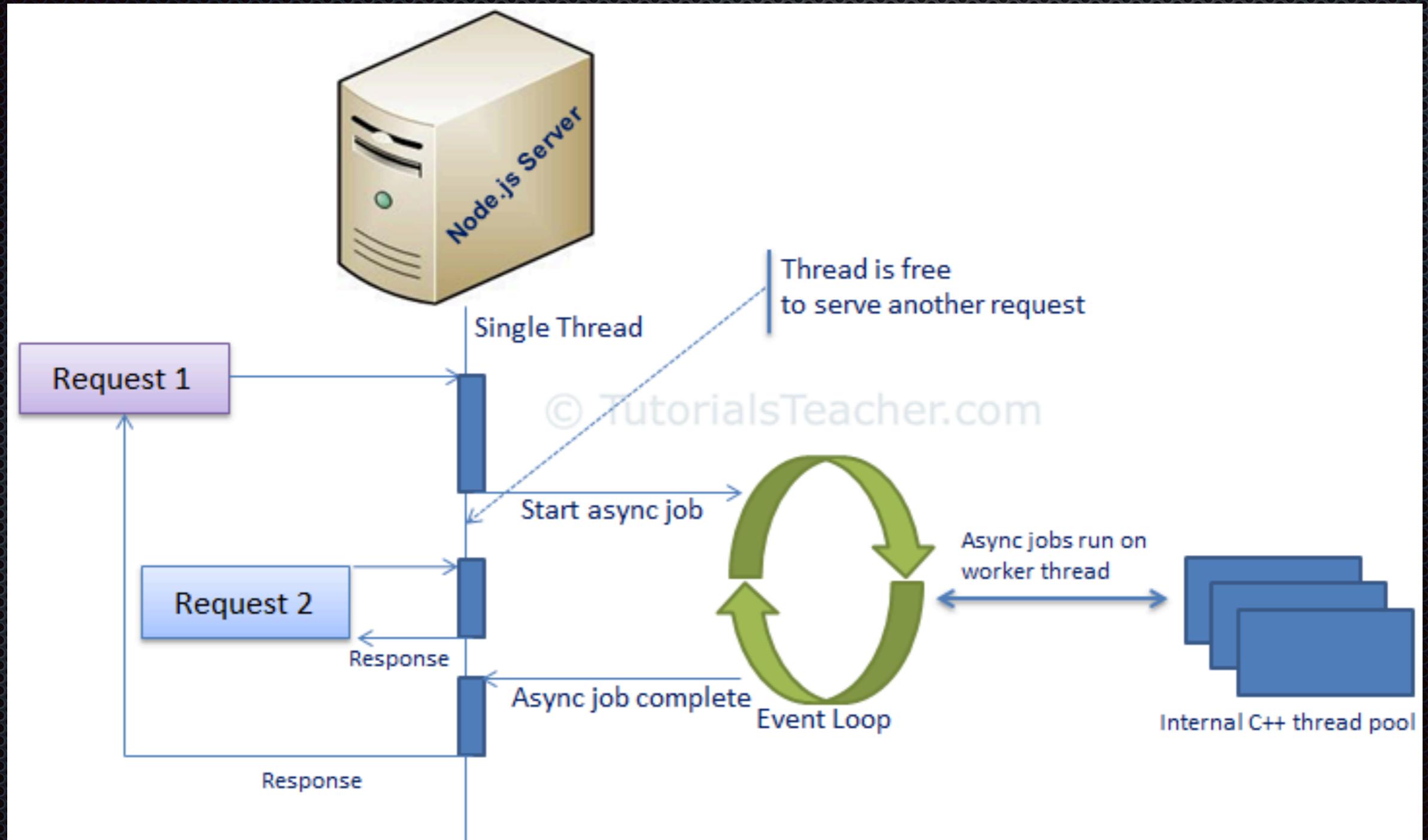
# Node.js Process Model

- runs in a single process and the application code runs in a single thread and thereby needs less resources than other platforms;
- all the user requests to your web application will be handled by a single thread and all the I/O work or long running job is performed asynchronously for a particular request;
- the single thread doesn't have to wait for the request to complete and is free to handle the next request;

# Node.js Process Model

- when asynchronous I/O work completes, it processes the request further and sends the response;
- an event loop is constantly watching for the events to be raised for an asynchronous job and executes the callback function when the job completes;
- Node.js uses in its implementation [libev](#) for the event loop which in turn uses internal C++ thread pool to provide asynchronous I/O;

# Asynchronous Web Server Model



# Setup the dev environment

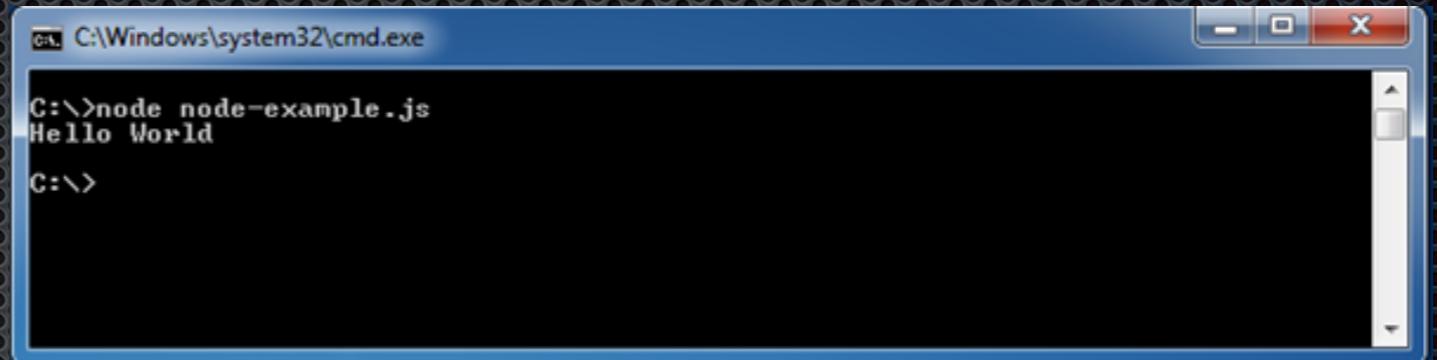
Node.js development environment can be setup in Windows, Mac, Linux and Solaris. The following tools/SDK are required for developing a Node.js application on any platform.

1. Node.js ( download from <https://nodejs.org/en/> or install with Homebrew on MacOS or Linuxbrew on Linux OS )
2. Node Package Manager (NPM) - is bundled with Node.js
3. IDE (Integrated Development Environment) or TextEditor ( you can download Sublime Text 3 from <https://www.sublimetext.com/> )

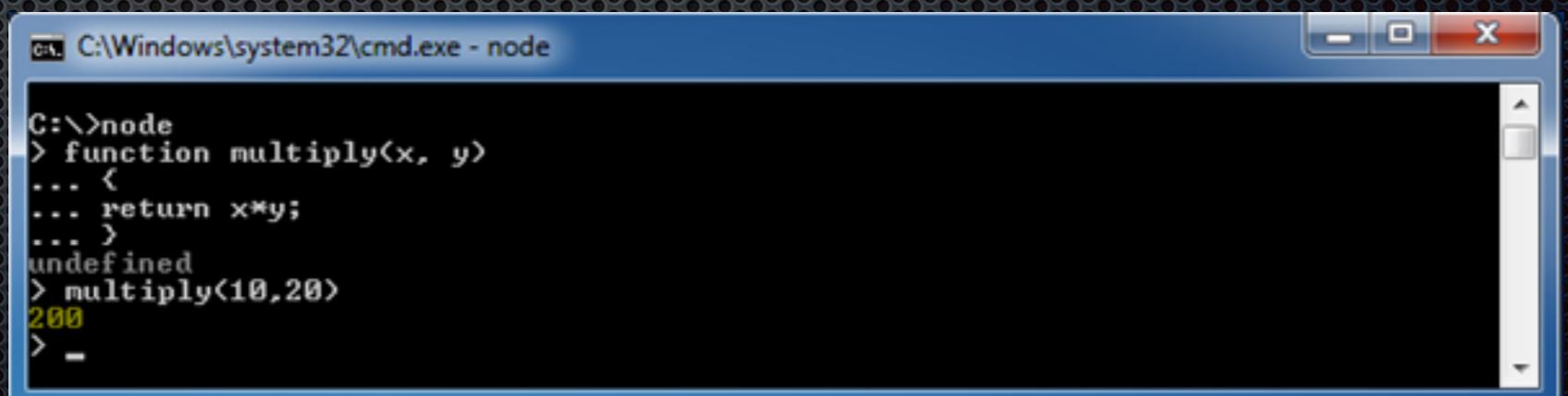
# Node.js Console

```
node-example.js

console.log("Hello World");
```



- Node.js comes with virtual environment called REPL (aka Read-Eval-Print-Loop);
- quick and easy way to test simple code;
- to launch the REPL (Node shell), open command prompt (in Windows) or terminal (in Mac or UNIX/Linux) and type *node*;



# Node.js Basics

## Primitive Types

- String
- Number
- Boolean
- Undefined
- Null
- RegExp



## Object Literal

### Example: Object

```
var obj = {  
  authorName: 'Ryan Dahl',  
  language: 'Node.js'  
}
```

## Functions

### Example: Function

```
function Display(x) {  
  console.log(x);  
}  
  
Display(100);
```

# Node.js Core Modules

- simple or complex functionality organised in single or multiple JavaScript files which can be reused throughout the application;
- each module in Node.js has its own context, so it cannot interfere with other modules or pollute global scope;

Core Module	Description
<a href="#">http</a>	http module includes classes, methods and events to create Node.js http server.
<a href="#">url</a>	url module includes methods for URL resolution and parsing.
<a href="#">querystring</a>	querystring module includes methods to deal with query string.
<a href="#">path</a>	path module includes methods to deal with file paths.
<a href="#">fs</a>	fs module includes classes, methods, and events to work with file I/O.
<a href="#">util</a>	util module includes utility functions useful for programmers.

# Usage of Core Modules

- **require()** function returns an object because http module returns its functionality as an object
- you can use its properties and methods using dot notation e.g. `http.createServer()`.

## Example: Load and Use Core http Module

```
var http = require('http');

var server = http.createServer(function(req, res){
    //write code here
});

server.listen(5000);
```

Import a module using require():

```
var module = require('module_name');
```

# Local Modules

- modules created locally in your Node.js application;
- these modules include different functionalities of your application in separate files and folders;
- **module.exports** is a special object which is included in every JS file in the Node.js application by default
- use **module.exports** or **exports** to expose a function, object or variable as a module in Node.js.

Log.js

```
var log = {
    info: function (info) {
        console.log('Info: ' + info);
    },
    warning:function (warning) {
        console.log('Warning: ' + warning);
    },
    error:function (error) {
        console.log('Error: ' + error);
    }
};

module.exports = log
```

app.js

```
var myLogModule = require('./Log.js');

myLogModule.info('Node.js started');
```

```
C:\> node app.js
Info: Node.js started
```

# Node Packaging Manager

- NPM is a command line tool that installs, updates or uninstalls Node.js packages in your application.
- It is also an online repository for open-source Node.js packages. The node community around the world creates useful modules and publishes them as packages in this repository;
- All the modules installed using NPM are installed under **node\_modules** folder.

```
C:\MyNodeProj> npm install express
```

the command will create an ExpressJS folder under **node\_modules** folder (in the root folder of your project) and will install Express.js there

```
C:\MyNodeProj> npm install express --save
```

it additionally saves the dependency in **package.json**

```
C:\MyNodeProj> npm install -g express
```

the command will install the Express.js dependency globally (NPM installs global packages into /<User>/local/lib/node\_modules folder)

```
C:\MyNodeProj> npm init
```

it starts a command line client that allows the configuration presented in the package.json

```
C:\MyNodeProj> npm update express
```

updates the version for package express

```
C:\MyNodeProj> npm uninstall express
```

uninstalls the package express

## package.json

```
{  
  "name": "NodejsConsoleApp",  
  "version": "0.0.0",  
  "description": "NodejsConsoleApp",  
  "main": "app.js",  
  "author": {  
    "name": "Dev",  
    "email": ""  
  },  
  "dependencies": {  
    "express": "^4.13.3"  
  }  
}
```



How do you uninstall a global module?

How do you update **npm**?

# Callbacks

**Simply put:** A callback is a function that is to be executed after another function (normally asynchronous) has finished executing – hence the name ‘call back’.

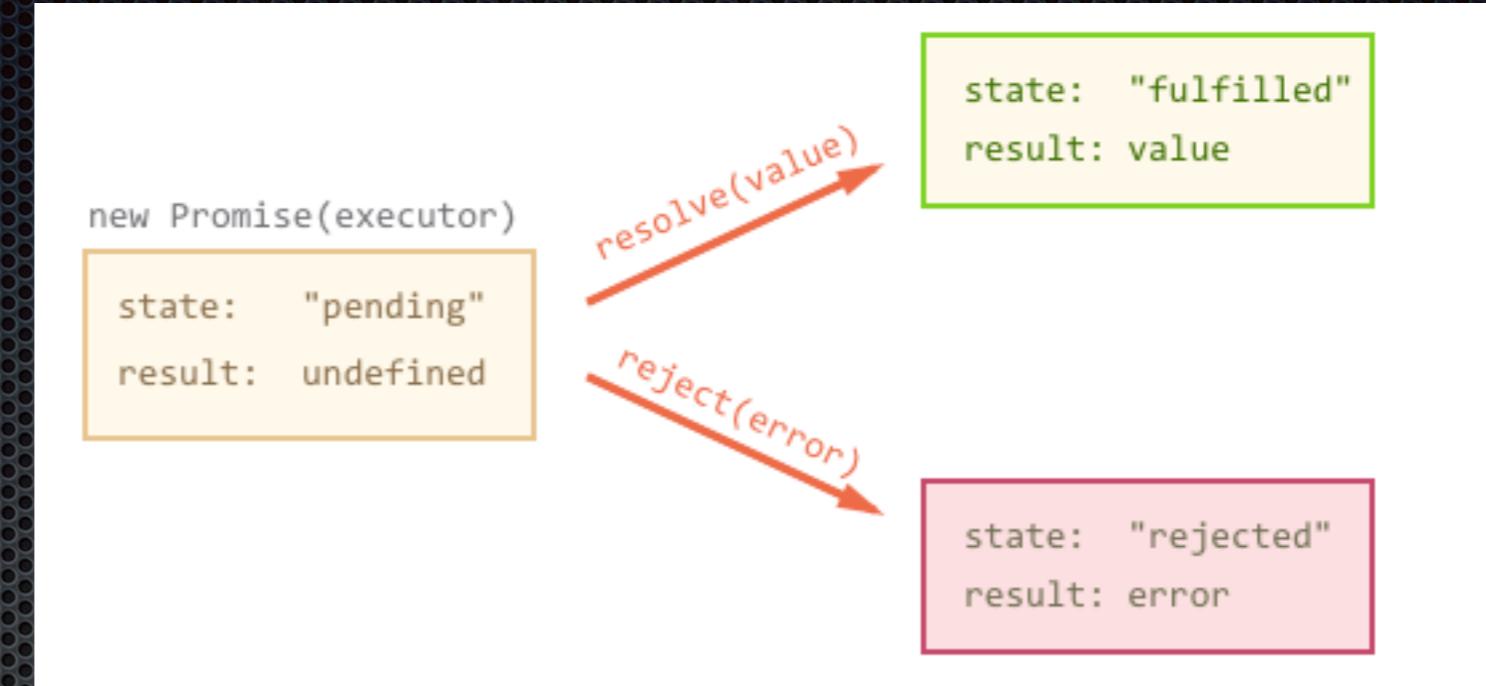
```
function doHomework(subject) {  
  alert(`Starting my ${subject} homework.`);  
}
```



```
function doHomework(subject, callback) {  
  alert(`Starting my ${subject} homework.`);  
  callback();  
}  
  
doHomework('math', function() {  
  alert('Finished my homework');  
});
```



# Promises



"Imagine you are a **kid**. Your mom **promises** you that she'll get you a **new phone** next week."

You *don't know* if you will get that phone until next week. Your mom can either *really buy* you a brand new phone, or *stand you up* and withhold the phone if she is not happy.

That is a **promise**. A promise has 3 states. They are:

1. *Pending*: You *don't know* if you will get that phone
2. *Fulfilled*: Mom is happy, she buys you a brand new phone
3. *Rejected*: Your mom is not happy, she withdraws the phone

# Promises Example

```
/_ ES5 _/
var isMomHappy = false;

// Promise
var willIGetNewPhone = new Promise(
  function (resolve, reject) {
    if (isMomHappy) {
      var phone = {
        brand: 'Samsung',
        color: 'black'
      };
      resolve(phone); // fulfilled
    } else {
      var reason = new Error('mom is not happy');
      reject(reason); // reject
    }
  }
);
```

```
/_ ES5 _/
...

// call our promise
var askMom = function () {
  willIGetNewPhone
    .then(function (fulfilled) {
      // yay, you got a new phone
      console.log(fulfilled);
      // output: { brand: 'Samsung', color: 'black' }
    })
    .catch(function (error) {
      // oops, mom don't buy it
      console.log(error.message);
      // output: 'mom is not happy'
    });
};

askMom();
```

Promise Definition

Promise Usage

# Node.js File System

Node.js includes **fs** module to access physical file system.

The fs module is responsible for all the asynchronous or synchronous file I/O operations.

## Reading File

Use `fs.readFile()` method to read the physical file asynchronously.

### Signature:

```
fs.readFile(fileName [,options], callback)
```

### Parameter Description:

- **filename:** Full path and name of the file as a string.
- **options:** The options parameter can be an object or string which can include encoding and flag. The default encoding is utf8 and default flag is "r".
- **callback:** A function with two parameters err and fd. This will get called when `readFile` operation completes.

## Writing File

Use `fs.writeFile()` method to write data to a file. If file already exists then it overwrites the existing content otherwise it creates a new file and writes data into it.

### Signature:

```
fs.writeFile(filename, data[, options], callback)
```

### Parameter Description:

- `filename`: Full path and name of the file as a string.
- `Data`: The content to be written in a file.
- `options`: The options parameter can be an object or string which can include encoding, mode and flag. The default encoding is `utf8` and default flag is "`r`".
- `callback`: A function with two parameters `err` and `fd`. This will get called when write operation completes.

## Open File

Alternatively, you can open a file for reading or writing using `fs.open()` method.

### Signature:

```
fs.open(path, flags[, mode], callback)
```

### Parameter Description:

- `path`: Full path with name of the file as a string.
- `Flag`: The flag to perform operation
- `Mode`: The mode for read, write or readwrite. Defaults to 0666 readwrite.
- `callback`: A function with two parameters `err` and `fd`. This will get called when file open operation completes.

## Important method of fs module

Method	Description
fs.readFile(fileName [,options], callback)	Reads existing file.
fs.writeFile(filename, data[, options], callback)	Writes to the file. If file exists then overwrite the content otherwise creates new file.
fs.open(path, flags[, mode], callback)	Opens file for reading or writing.
fs.rename(oldPath, newPath, callback)	Renames an existing file.
fs.chown(path, uid, gid, callback)	Asynchronous chown.
fs.stat(path, callback)	Returns fs.stat object which includes important file statistics.
fs.link(srcpath, dstpath, callback)	Links file asynchronously.
fs.symlink(destination, path[, type], callback)	Symlink asynchronously.
fs.rmdir(path, callback)	Renames an existing directory.
fs.mkdir(path[, mode], callback)	Creates a new directory.
fs.readdir(path, callback)	Reads the content of the specified directory.
fs.utimes(path, atime, mtime, callback)	Changes the timestamp of the file.
fs.exists(path, callback)	Determines whether the specified file exists or not.
fs.access(path[, mode], callback)	Tests a user's permissions for the specified file.
fs.appendFile(file, data[, options], callback)	Appends new content to the existing file.

The following example opens an existing file and reads its content.

### Example: File open and read

```
var fs = require('fs');

fs.open('TestFile.txt', 'r', function (err, fd) {
    if (err) {
        return console.error(err);
    }

    var buffr = new Buffer(1024);

    fs.read(fd, buffr, 0, buffr.length, 0, function (err, bytes) {
        if (err) throw err;

        // Print only read bytes to avoid junk.
        if (bytes > 0) {
            console.log(buffr.slice(0, bytes).toString());
        }
    });

    // Close the opened file.
    fs.close(fd, function (err) {
        if (err) throw err;
    });
});
});
```



Write a script in Node.js that reads a file, modifies the content, saves it in another location and deletes the old file.

# Node.js Web Server

- Node.js provides capabilities to create your own web server which will handle HTTP requests asynchronously;
- you can use Apache to run a Node.js web application but it is recommended to use Node.js web server;

server.js

```
var http = require('http'); // 1 - Import Node.js core module

var server = http.createServer(function (req, res) { // 2 - creating server

    //handle incomming requests here..

});

server.listen(5000); //3 - listen for any incoming requests

console.log('Node.js web server at port 5000 is running..')
```

```
C:\> node server.js
Node.js web server at port 5000 is running..
```

# Handling HTTP Requests

## server.js

```
var http = require('http'); // Import Node.js core module

var server = http.createServer(function (req, res) {    //create web server
  if (req.url == '/') { //check the URL of the current request

    // set response header
    res.writeHead(200, { 'Content-Type': 'text/html' });

    // set response content
    res.write('<html><body><p>This is home Page.</p></body></html>');
    res.end();

  }
  else if (req.url == "/student") {

    res.writeHead(200, { 'Content-Type': 'text/html' });
    res.write('<html><body><p>This is student Page.</p></body></html>');
    res.end();

  }
  else if (req.url == "/admin") {

    res.writeHead(200, { 'Content-Type': 'text/html' });
    res.write('<html><body><p>This is admin Page.</p></body></html>');
    res.end();

  }
  else
    res.end('Invalid Request!');

});

server.listen(5000); //6 - listen for any incoming requests

console.log('Node.js web server at port 5000 is running..')
```

- The `http.createServer()` method includes **request** and **response** parameters which are supplied by Node.js.
- The **request object** can be used to get information about the current HTTP request e.g., url, request header, and data.
- The **response object** can be used to send a response for a current HTTP request.

# Express.js

- web application framework for Node.js. It provides various features that make web application development fast and easy which otherwise takes more time using only Node.js;
- based on the Node.js middleware module called **connect** which in turn uses **http** module. So, any middleware which is based on connect will also work with Express.js;
- easy to configure and customize;
- allows you to define routes of your application based on HTTP methods and URLs.
- includes various middleware modules which you can use to perform additional tasks on request and response;
- easy to integrate with different template engines like Pug;
- allows you to define an error handling middleware;
- easy to serve static files and resources of your application;
- allows you to create REST API server;
- easy to connect with databases such as MongoDB, Redis, MySQL



## Install Express.js

You can install express.js using npm. The following command will install latest version of express.js globally on your machine so that every Node.js application on your machine can use it.

```
npm install -g express
```

## app.js: Express.js Web Server

```
var express = require('express');
var app = express();

// define routes here..

var server = app.listen(5000, function () {
  console.log('Node server is running..');
});
```



Run the above example using **node app.js** command and point your browser to <http://localhost:5000>. What will it show?

## Example: Configure Routes in Express.js

```
var express = require('express');
var app = express();

app.get('/', function (req, res) {
  res.send('<html><body><h1>Hello World</h1></body></html>');
});

app.post('/submit-data', function (req, res) {
  res.send('POST Request');
});

app.put('/update-data', function (req, res) {
  res.send('PUT Request');
});

app.delete('/delete-data', function (req, res) {
  res.send('DELETE Request');
});

var server = app.listen(5000, function () {
  console.log('Node server is running..');
});
```

In the above example, `app.get()`, `app.post()`, `app.put()` and `app.delete()` methods define routes for HTTP GET, POST, PUT, DELETE respectively. The first parameter is a path of a route which will start after base URL. The callback function includes `request ↗` and `response` object which will be executed on each request.



Run the above example using **node app.js** command and point your browser to <http://localhost:5000>. What does it show now?

# Handle POST Request in Express

## Body Parser

To handle HTTP POST request in Express.js version 4 and above, you need to install middleware module called [body-parser](#). The middleware was a part of Express.js earlier but now you have to install it separately.

This body-parser module parses the JSON, buffer, string and url encoded data submitted using HTTP POST request. Install body-parser using NPM as shown below.

```
npm install body-parser --save
```

First, create Index.html file in the root folder of your application and write the following HTML code in it.

### Example: Configure Routes in Express.js

```
<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <meta charset="utf-8" />
    <title></title>
</head>
<body>
    <form action="/submit-student-data" method="post">
        First Name: <input name="firstName" type="text" /> <br />
        Last Name: <input name="lastName" type="text" /> <br />
        <input type="submit" />
    </form>
</body>
</html>
```

## app.js: Handle POST Route in Express.js

```
var express = require('express');
var app = express();

var bodyParser = require("body-parser");
app.use(bodyParser.urlencoded({ extended: false }));

app.get('/', function (req, res) {
  res.sendFile('index.html');
});

app.post('/submit-student-data', function (req, res) {
  var name = req.body.firstName + ' ' + req.body.lastName;

  res.send(name + ' Submitted Successfully!');
});

var server = app.listen(5000, function () {
  console.log('Node server is running..');
});
```



Run the above example using **node app.js** command and point your browser to <http://localhost:5000>. Test the application.



... and now for the Cloud part

- the term **Cloud Computing** refers to the on-demand delivery of IT resources via the Internet with pay-as-you-go pricing

**Cloud computing has become the new normal**



- Amazon Web Services (AWS) is a secure cloud services platform, offering compute power, database storage, content delivery and other functionality to help businesses scale and grow.

# ... one success story



- 400,000 people are hosted on any given night
- 10M guests in one year after migration to Cloud solutions
- Started from 24 instances in 2010 and got to over 1000



“ We have a 5 person operations team. ”

AWS allows us to devote our resources and mindshare to the core business.

# An overview of services

- **EC2:** Server configuration and hosting
- **S3:** Data storage and movement
- **CloudFront:** Deliver a better user experience
- **Route 53:** The AWS DNS service
- **Cloudwatch:** Monitor your AWS environment

- **Lambda**: Functions for optimized compute
- **AWS Config**: Infrastructure management
- **Kinesis**: Optimize data flow
- **DynamoDB**: Fast, easy database access

The screenshot shows the AWS Management Console with the 'Services' tab selected. On the left, there's a sidebar with icons for History, S3, EC2, Console Home, Data Pipeline, Compute & Networking, Storage & Content Delivery, Database, Analytics, Deployment & Management, and App Services. The main area displays a grid of AWS services, each with an icon and a name. The services are organized into two columns: a primary column under 'All AWS Services' and a secondary column of related services.

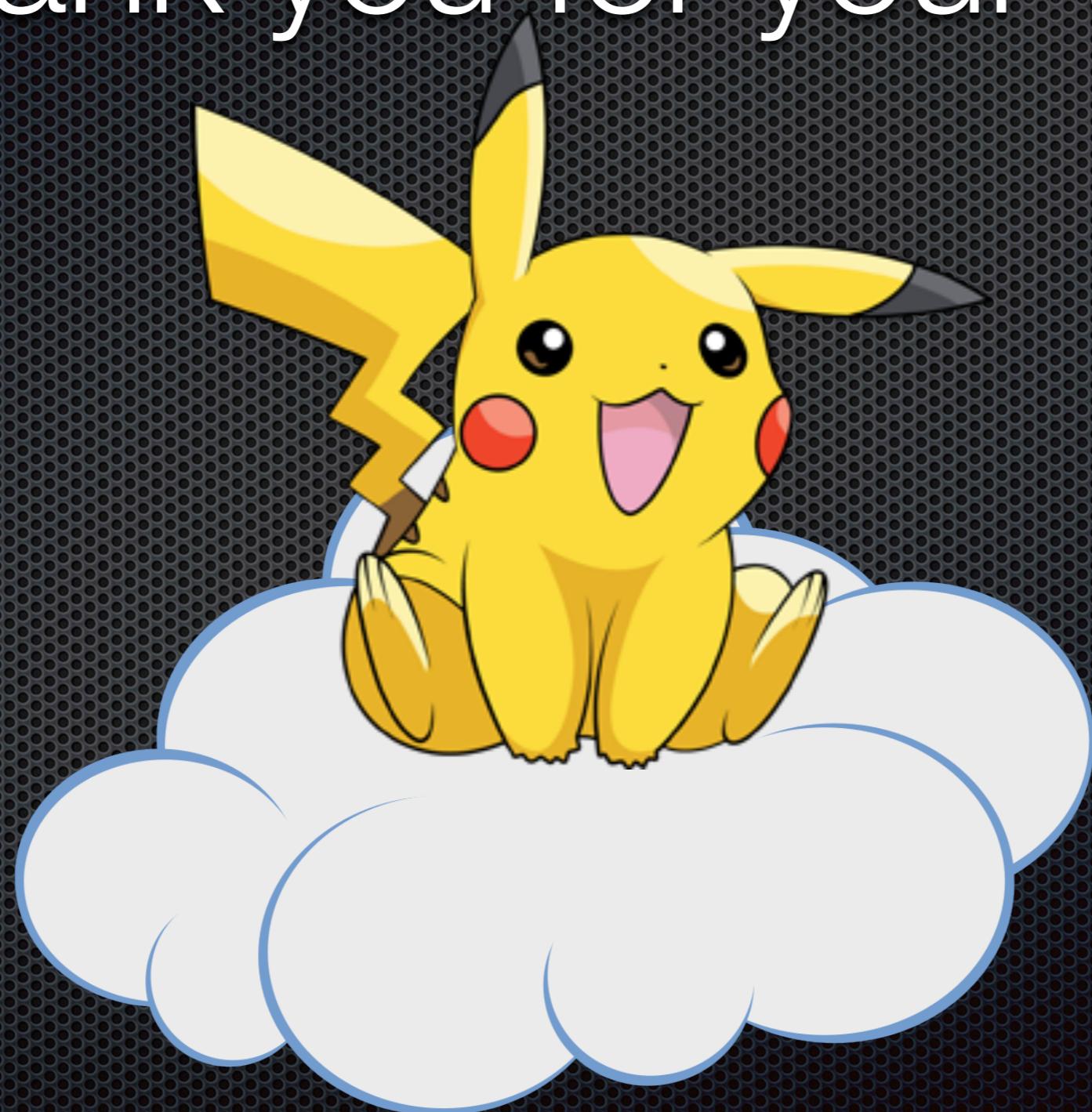
All AWS Services	
CloudFormation	ElastiCache
CloudFront	Elastic Beanstalk
CloudSearch	Elastic MapReduce
CloudTrail	Elastic Transcoder
CloudWatch	Glacier
Data Pipeline	IAM
Direct Connect	OpsWorks

*The almost limitless possibilities of AWS are wrangled down into manageable control screens, showing you how all of your virtual gears are meshing together.*



... wait until you build your first serverless web application

Thank you for your time!

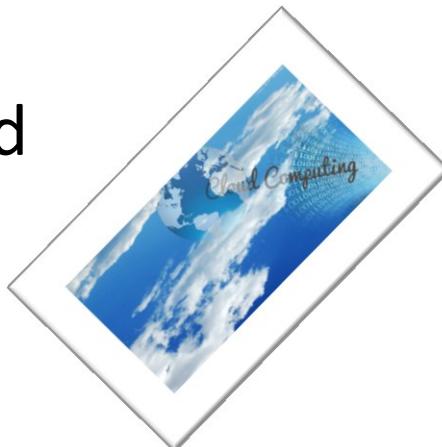


Universitatea "Alexandru Ioan Cuza"  
Facultatea de Informatică

Prof. Dr. Lenuța Alboiae  
adria@info.uaic.ro



# Concurrent and Distributed Programming - Overview -



# Concurrent and Distributed Programming

Course	Laboratory
Distributed Systems: History&Evolution	Client/Server applications
NodeJS - Programming	Client/Server applications
Network: Concepts & Architectures	Client/Server - TCP- C or Node JS
Network Programming	Web Services & APIs
Application Level Protocols	Web Services & APIs
Paradigms: P2P,RPC	Web Services & APIs
Cloud Computing - I	AWS – Cloud Explore
Cloud Computing - II	Google Cloud Explore
CloudComputing - IaaS	Google Cloud Applications
Google App Engine	Google Cloud Applications
Cloud Computing - Business Aspects	Google Cloud Applications
CloudComputing - Overview	
CloudComputing - Security Aspects	

# Concurrent and Distributed Programming

## Bibliography:

- *bibliography is specified for each course*

# Concurrent and Distributed Programming

## Evaluation

$$N = 0.3*T+0.6*L+1$$

### Details:

Nota finală va fi calculată conform prevederilor ECTS

<b>Test Final (T)</b>	Testul final – T – sustinut la finalul semestrului va fi evaluat cu minim 0 si maxim 10 puncte si are ca pondere 30% din nota finală; Este o proba de evaluare pe parcurs si nu poate fi reevaluat in sesiunea de restante. Bonusuri pentru activități suplimentare	30%
<b>Activitate de Laborator (L)</b>	Rezolvarea a 3 teme cu caracter practic, evaluate pe parcursul semestrului, care sunt evaluate cu 20 de puncte fiecare. Evaluarea acestora se va realiza in urma prezentarii orale a solutiilor si a incarcarii codului sursa la adresa indicata in cadrul laboratorului. Toate aceste teme sunt probe de evaluare pe parcurs si trebuie sa respecte termenul de predare specificat in cadrul laboratorului/cursului.  Obs. <ul style="list-style-type: none"><li>O tema poate fi echivalata prin sustinerea in cadrul cursului a unei prezentari de jumata de ora, strans legata de tematica cursului si care a trecut printr-o recenzie anterioara (numai cu acceptul prealabil al titularului de curs)</li><li>O tema poate fi echivalata prin publicarea unei lucrari stiintifice incadrata pe tematica cursului (numai cu acceptul prealabil al titularului de curs)</li></ul> Bonusuri pentru activități suplimentare.	60%

# Concurrent and Distributed Programming

## Conclusions:

- ❑ The course is not for you .....
- ❑ If you are not interested in the topic
- ❑ If you are not ready to do programming
- ❑ If you can not stand the permanent change of vision and technology
- ❑ Otherwise You will be rewarded with a knowledge of an important field in computer science ☺

Universitatea "Alexandru Ioan Cuza"  
Facultatea de Informatică



Questions?

# Distributed Systems

## Prehistory, History, Present, Future

**Lenuta Alboiaie**  
**adria@info.uaic.ro**  
Alexandru Ioan Cuza  
University of Iasi, Romania



- Steps to distributed programming?
- Steps to reach Cloud Computing?



# Prehistory | Origins of the Internet

- The desire to communicate (also remote)
- First instruments: language and speech
- Communication form: written



Writing on rocks, cave walls, papyrus ☺

- Remote communication?
  - Fire signals
  - Morse Code
- 19<sup>th</sup> century: electricity and telegraph

# History & Evolution

1837 Charles Babbage designs “first computer”



1863 Jules Verne:

"photo-telegraphy allowed any writing, signature or illustration  
to be sent faraway - every house was wired"

1891 – Ada Lovelace – first program for Babbage machinery

# History & Evolution

- 1936 Alan Turing – describes Turing machine
- 1945-1985: “computers were large and expensive”
- ... improvements:

**Networking**

**Processors**

**Memory**

**Storage**

**Protocols**

# History & Evolution

Networking

Processors

Memory

Storage

Protocols

- Microprocessor industry (8-bit, 16,32,64 ...) experienced a rapid evolution
- Computers have become
  - Smaller
  - Cheaper
  - Faster
- *“...from machine that cost 10 million dollars and executed 1 instruction per second we have come to machines that cost 1000 dollars and are able to execute 1 billion instructions per second, a price/performance gain of 1013”*

# History & Evolution

Networking

Processors

Memory

Storage

Protocols

Year	Cost (\$/MB)	Capacity (average)
1977	\$32,000	16K
1987	\$250	640K-2MB
1997	\$2	64MB-256MB
2007	\$0.06	512MB-2GB+
2020	...	8GB...->...

[<http://www.cs.rutgers.edu/~pxk/>]

# History & Evolution

Networking

Processors

Memory

Storage

Protocols

- 1977: 310KB floppy drive ~ \$1480
- 1987: 40 MB drive ~ \$679
- 2008: 750 GB drive ~ \$99
- 2016: 1TB drive ~ \$120
- 2020 3-4TB drive ~ \$110
- *“Recording density increased over 60,000,000 times over 50 years”*

[<http://www.cs.rutgers.edu/~pxk/>]

# History & Evolution

## 1961-1972: first communication's attempts using packet-switching

- 1961: Kleinrock – proposed a theoretical model
- 1964: Baran – implemented the communication among US military computers
- 1967: ARPAnet was projected by Advanced Research Projects Agency
- 1969: first operational node ARPAnet, a network formed by 4 computers
- 1972:
  - public demonstration of ARPAnet technologies
  - NCP (Network Control Protocol) – the first host-host protocol
  - First program for electronic mail (e-mail)
  - The sign @ is introduced
  - ARPAnet contains 15 nodes

# History & Evolution

LAN – speed data transfer:

- Original Ethernet: 2.94 Mbps
- **1985**: thick Ethernet: 10 Mbps; 1 Mbps with twisted pair networking
- **1991**: 10BaseT - twisted pair: 10 Mbps
- **1995**: 100 Mbps Ethernet
- **1998**: 1 Gbps (Gigabit) Ethernet
- **1999**: 802.11b (wireless Ethernet) standardized
- **2001**: 10 Gbps introduced
- **2005**: 100 Gbps (over optical link)
- **2012**: 159.2 Gbps



Huge amount of data can be sent among networks

# History & Evolution

**1972-1980:** The *Internetworking* concept appeared. Also, proprietary networks appeared.

- 1974: Cerf si Kahn – proposed a communication protocol entitled TCP(Transmission Control Protocol)
- 1978: TCP/IP protocols stack was standardized via RFC (Request For Comments) documents
- In the late of 70s: proprietary networks stacks appeared: DECnet, SNA, XNA
- 1979: ARPAnet contained 200 nodes

# History | Origins of the Internet

**1980-1990: new protocols, the network number was increasing, Internet**

1983: TCP/IP was used

1982: SMTP (Simple Mail Transfer Protocol) was defined

1983: DNS (translation of host name into IP address and vice versa) appeared

1985: FTP(File Transfer Protocol) protocol appeared

1986: **Internet backbone appeared**

1988: some congestion control mechanisms for TCP were introduced

# History & Present | Origins of the Internet

**1990 ->2000->2020: World Wide Web, new commercial network applications**

Early '90:

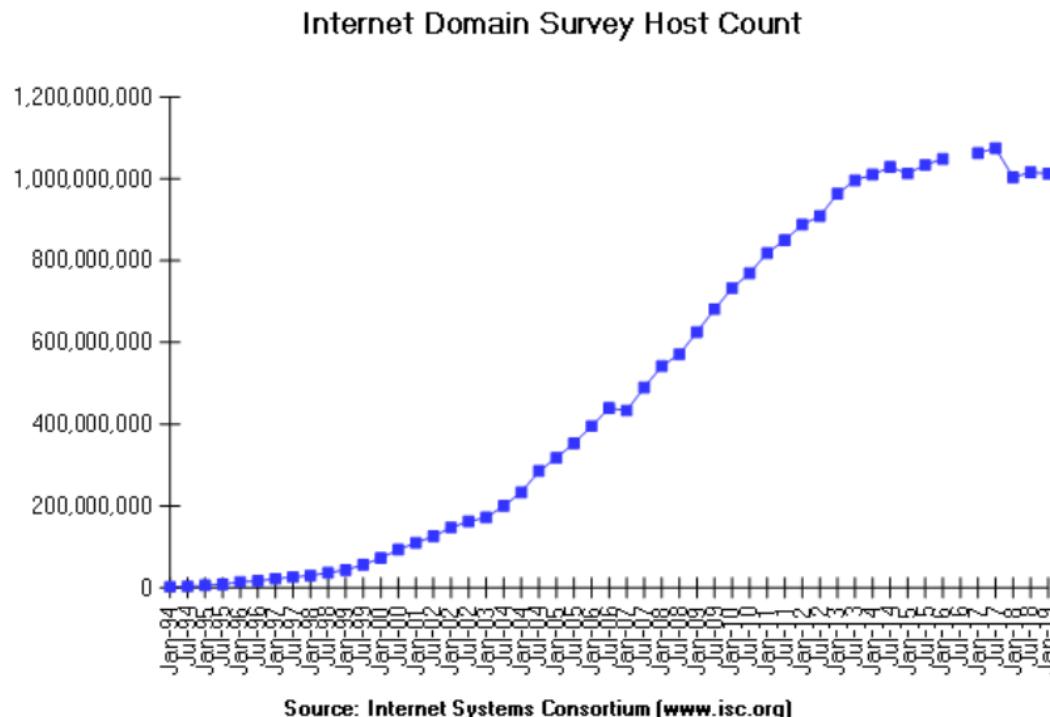
- **World Wide Web**

After the late '90 -> present :

- Global Networks
- Wireless networks
- Commercial applications of the Internet
  - instant messaging, P2P file sharing
  - social networks, video-streaming,
  - cloud-computing, edge/fog computing

...

# Present



**Figure. Hosts number form January 1994 till January 2019**

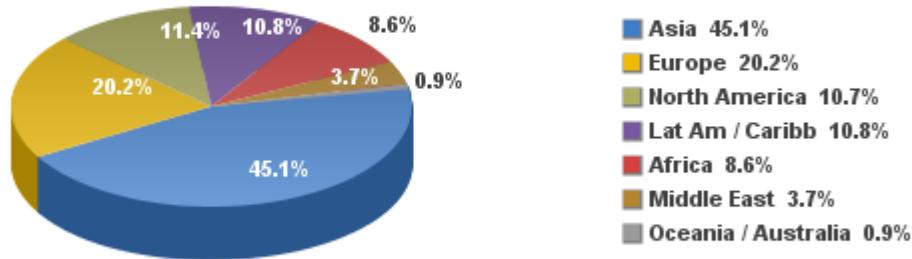
Source: <https://www.isc.org/network/survey/>

# Past

## WORLD INTERNET USAGE AND POPULATION STATISTICS December 31, 2013

World Regions	Population (2014 Est.)	Internet Users Dec. 31, 2000	Internet Users Latest Data	Penetration (% Population)	Growth 2000-2014
Africa	1,125,721,038	4,514,400	240,146,482	21.3 %	5,219.6 %
Asia	3,996,408,007	114,304,000	1,265,143,702	31.7 %	1,006.8 %
Europe	825,802,657	105,096,093	566,261,317	68.6 %	438.8 %
Middle East	231,062,860	3,284,800	103,829,614	44.9 %	3,060.9 %
North America	353,860,227	108,096,800	300,287,577	84.9 %	177.8 %
Latin America / Caribbean	612,279,181	18,068,919	302,006,016	49.3 %	1,571.4 %
Oceania / Australia	36,724,649	7,620,480	24,804,226	67.5 %	225.5 %
<b>WORLD TOTAL</b>	<b>7,181,858,619</b>	<b>360,985,492</b>	<b>2,802,478,934</b>	<b>39.0 %</b>	<b>676.3 %</b>

### Internet Users in the World Distribution by World Regions - 2013 Q4



Source: Internet World Stats - [www.internetworldstats.com/stats.htm](http://www.internetworldstats.com/stats.htm)

Basis: 2,802,478,934 Internet users on Dec 31, 2013

Copyright © 2014, Miniwatts Marketing Group

Tabel. INTERNET  
USAGE STATISTICS  
- The Internet Big  
Picture

Figura. Internet Users in the  
World by Regions  
December 2013 | Source:  
<http://www.internetworldstats.com/>

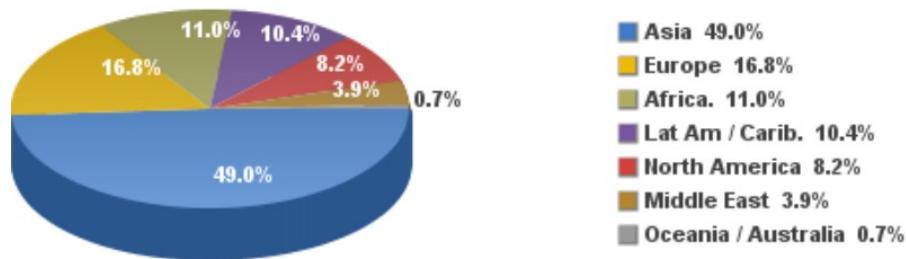
# Present

## WORLD INTERNET USAGE AND POPULATION STATISTICS JUNE 30, 2018 - Update

World Regions	Population (2018 Est.)	Population % of World	Internet Users 30 June 2018	Penetration Rate (% Pop.)	Growth 2000-2018
Africa	1,287,914,329	16.9 %	464,923,169	36.1 %	10,199 %
Asia	4,207,588,157	55.1 %	2,062,197,366	49.0 %	1,704 %
Europe	827,650,849	10.8 %	705,064,923	85.2 %	570 %
Latin America / Caribbean	652,047,996	8.5 %	438,248,446	67.2 %	2,325 %
Middle East	254,438,981	3.3 %	164,037,259	64.5 %	4,894 %
North America	363,844,662	4.8 %	345,660,847	95.0 %	219 %
Oceania / Australia	41,273,454	0.6 %	28,439,277	68.9 %	273 %
<b>WORLD TOTAL</b>	<b>7,634,758,428</b>	<b>100.0 %</b>	<b>4,208,571,287</b>	<b>55.1 %</b>	<b>1,066 %</b>

Table. Internet usage statistics relative to the number of people in the main geographic areas of the world

### Internet Users in the World by Regions - June 30, 2018



Source: Internet World Stats - [www.internetworldstats.com/stats.htm](http://www.internetworldstats.com/stats.htm)

Basis: 4,208,571,287 Internet users in June 30, 2018

Copyright © 2018, Miniwatts Marketing Group

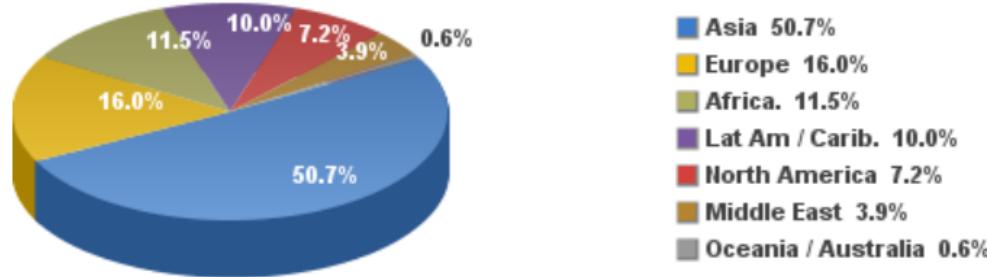
Figure. Internet users in the world - geographically divided June 2018 | Source: <http://www.internetworldstats.com/>

# Present

WORLD INTERNET USAGE AND POPULATION STATISTICS 2019 Mid-Year Estimates					
World Regions	Population (2019 Est.)	Population % of World	Internet Users 30 June 2019	Penetration Rate (% Pop.)	Growth 2000-2019
Africa	1,320,038,716	17.1 %	522,809,480	39.6 %	11,481 %
Asia	4,241,972,790	55.0 %	2,300,469,859	54.2 %	1,913 %
Europe	829,173,007	10.7 %	727,559,682	87.7 %	592 %
Latin America / Caribbean	658,345,826	8.5 %	453,702,292	68.9 %	2,411 %
Middle East	258,356,867	3.3 %	175,502,589	67.9 %	5,243 %
North America	366,496,802	4.7 %	327,568,628	89.4 %	203 %
Oceania / Australia	41,839,201	0.5 %	28,636,278	68.4 %	276 %
WORLD TOTAL	7,716,223,209	100.0 %	4,536,248,808	58.8 %	1,157 %

Table. Internet usage statistics relative to the number of people in the main geographic areas of the world

## Internet Users Distribution in the World - Mid-Year 2019



Source: Internet World Stats - [www.internetworldstats.com/stats.htm](http://www.internetworldstats.com/stats.htm)

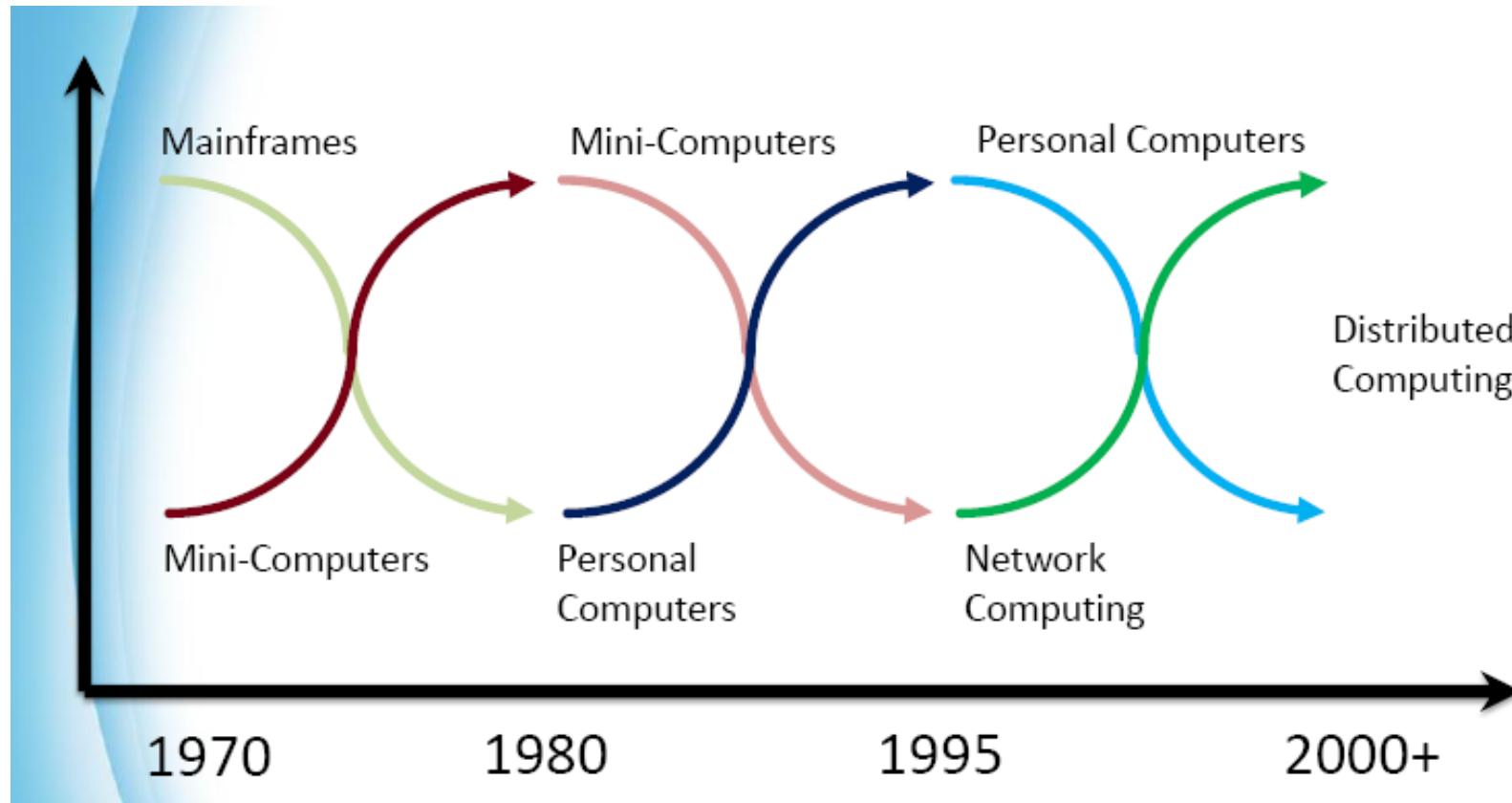
Basis: 4,536,248,808 Internet users in June 30, 2019

Copyright © 2019, Miniwatts Marketing Group

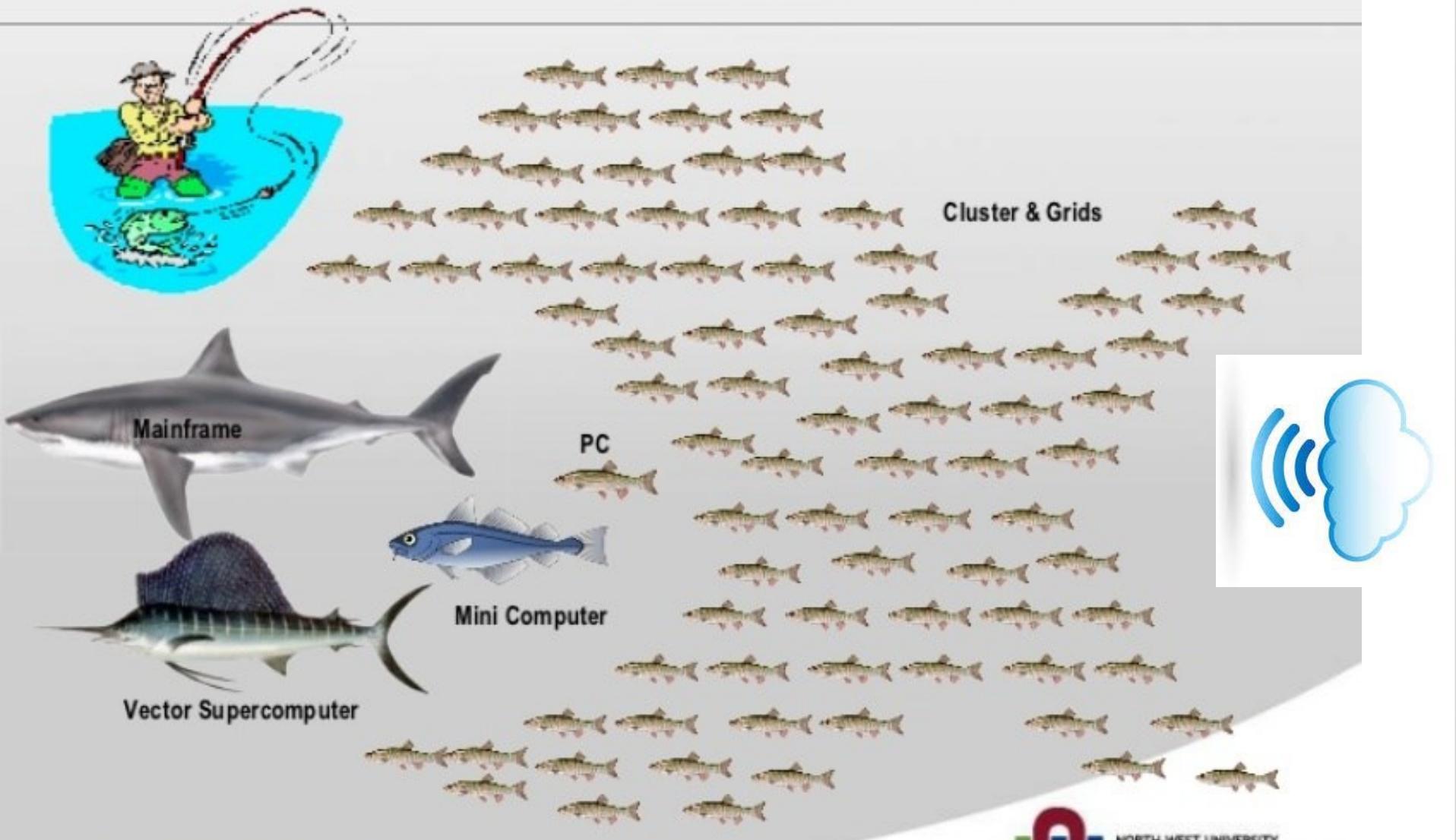
Figure. Internet users in the world - geographically divided  
February 2020 | Source:  
<http://www.internetworldstats.com/>

# Istorie & Evolutie

\* From supercomputers to workstations that can be connected together

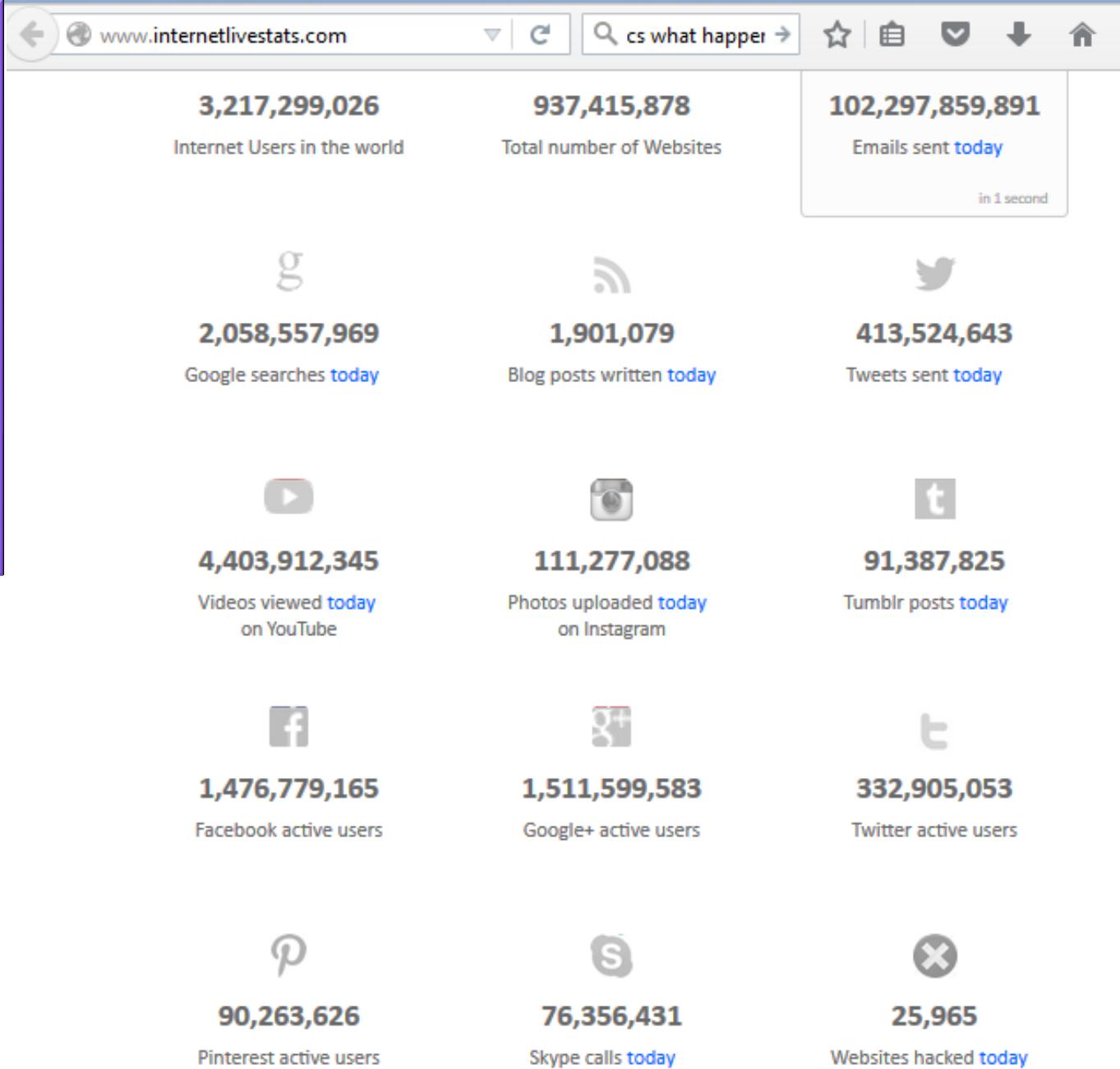


# The New World Order



Source 2006 UC Regents

# 2015



# 2018

**4,034,785,862**

Internet Users in the world

**1,917,268,622**

Total number of Websites

**59,436,723,979**Emails sent [today](#)**1,459,891,645**Google searches [today](#)**1,380,072**Blog posts written [today](#)**171,765,400**Tweets sent [today](#)**1,583,632,120**Videos viewed [today](#)  
on YouTube**18,230,030**Photos uploaded [today](#)  
on Instagram**30,000,050**Tumblr posts [today](#)**2,324,502,983**

Facebook active users

**641,046,211**

Google+ active users

**339,328,724**

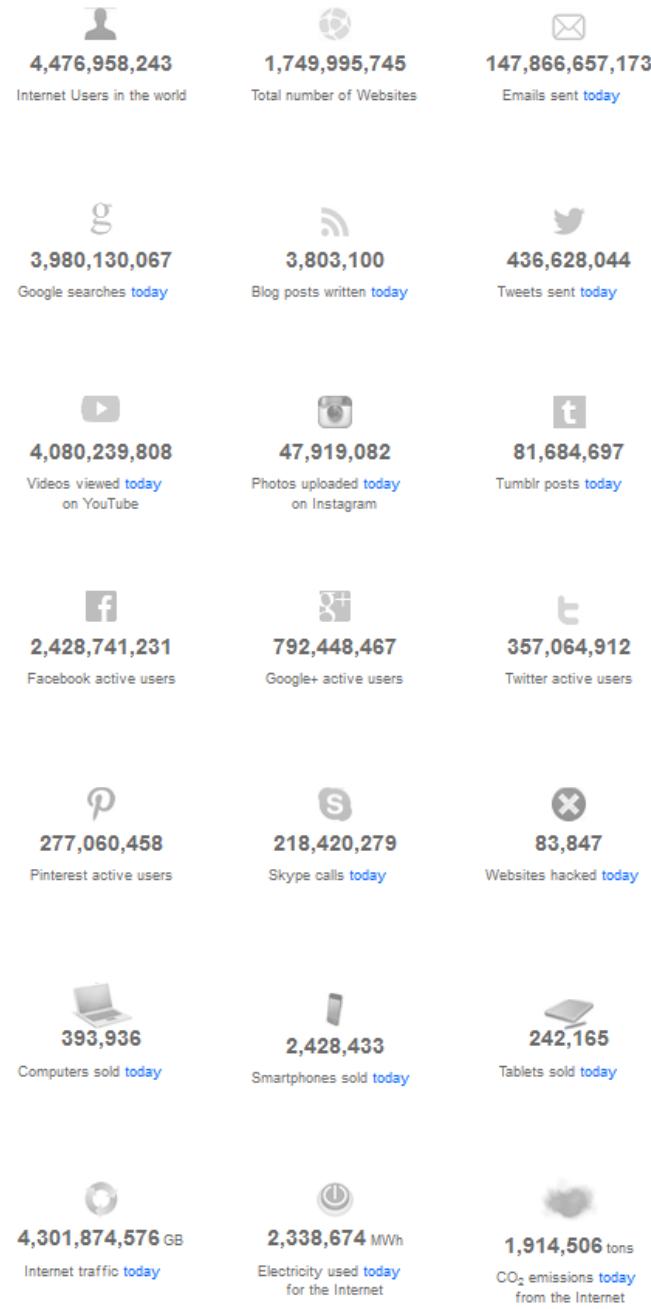
Twitter active users

**255,464,399**

Pinterest active users

**70,823,093**Skype calls [today](#)**26,170**Websites hacked [today](#)**171,532**Computers sold [today](#)**1,002,546**Smartphones sold [today](#)**110,882**Tablets sold [today](#)

# 2020



# Bibliography

- Internet Society: <http://www.isoc.org>
- FNC – Federal Networking Council: [http://www.nitrd.gov/fnc/Internet\\_res.html](http://www.nitrd.gov/fnc/Internet_res.html)
- InternetWorld Stats: <http://www.internetworldstats.com>
- Internet Systems Consortium: <http://www.isc.org>
- Internet Assigned Numbers Authority (IANA): <http://www.iana.org/>
- Cisco Systems: <http://www.cisco.com>
- <http://www.nethistory.info>
- <http://www.caida.org/>
- [http://news.netcraft.com/archives/web\\_server\\_survey.html](http://news.netcraft.com/archives/web_server_survey.html)
- <http://www.swivel.com/>
- <https://www.facebook.com/Grovo/info> :)
- <http://www.internetlivestats.com/>
- <http://www.worldometers.info/>
- <http://www.internetsociety.org/globalinternetreport/?gclid=CN6a2Y2wqMgCFUfkwgodZ8gL9A>

# Bibliography

- <https://www.facebook.com/50447527854/videos/10154755074917855/>
- <http://my.ss.sysu.edu.cn/courses/cloud/>
- <http://blogs.idc.com/ie/?p=730>
- <http://www.slideshare.net/woorung/trend-and-future-of-cloud-computing>
- <http://ganglia.sourceforge.net/>
- <http://www.focus.com/briefs/top-10-cloud-computing-trends/>
- <http://cacm.acm.org/magazines/2010/4/81493-a-view-of-cloud-computing/fulltext>
- <https://www.edutopia.org/discussion/use-cloud-based-technologies-classroom>
- <https://talktechwithme.wordpress.com/2012/10/17/blooms-revised-technology-taxonomy/>
- <http://www.teachthought.com/the-future-of-learning/trends-shifts/10-specific-examples-of-emerging-educational-technologies/>
- <http://www.hongkiat.com/blog/future-classroom-technologies/>
- <http://theregister.co.uk>
- <https://www.youtube.com/watch?v=KVoRVIg7U40&list=PLE2083A5E6D403628>

# Summary

## History & Evolution





# Network: Concepts&Architectures

**Lenuta Alboiae**  
**adria@info.uaic.ro**  
Alexandru Ioan Cuza  
University of Iasi, Romania



# Content

- Concepts and fundamental notions
  - Concepts
  - Definitions
  - Computer networks – necessity and use
  - Classification
  - Topologies
  - Components
- Structure of Computer Networks
- Network Architecture Models (OSI, TCP/IP)
- TCP/IP Model
- ISO/OSI versus TCP/IP

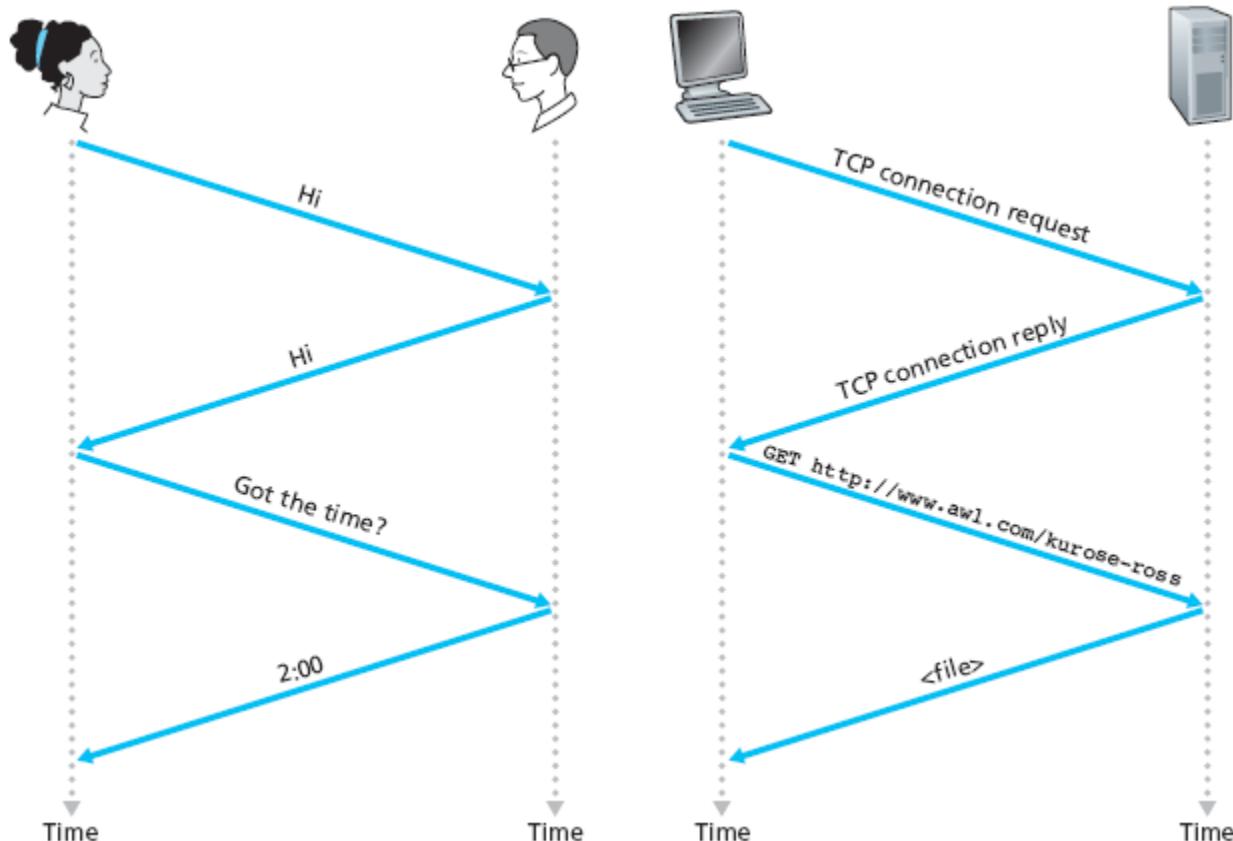
Hardware &  
Software  
Aspects

# Concepts

- **Information:** anything that can be represented using bits
- **Resource:** generic term that can signify data, equipment et. al.
- **Package:** a way of storing data
- **Link:** a connection among network members
- **Node:** a computer from the network which has an address
- **Protocol:** rules used to communicate
- **Communication:** information exchanges between network nodes

# Concepts

- **Protocol**



*A protocol defines the format and the order of messages exchanged between two or more communicating entities, as well as the actions taken on the transmission and/or receipt of a message or other event.*

**Figure. Protocol**

[Computer networking : a top-down approach  
James F. Kurose, Keith W. Ross]

# Computer Network

- **Definitions:**
  - Interconnected collection of autonomous computers
  - A network may be defined recursively as two or more nodes physically connected, as well as two or several networks connected through one or more nodes.
- **Aspects:**
  - **Hardware:** connect computers from a physical point of view
  - **Software:** Protocols – specify services provided by the network

# Computer Network

## Necessity:

- Resource sharing (physical, data)
- Reliability
- Reduced costs
- Impact in real life:
  - Remote information access
  - Interactive entertainment
  - E-Commerce
  - ...

# Networks types- classification

- After the spatial arrangement:
  - PAN (Personal Area Network)
  - LAN (Local Area Network)
  - MAN (Metropolitan Area Network)
  - WAN (Wide Area Network)
  - Internet

# Network types- classification

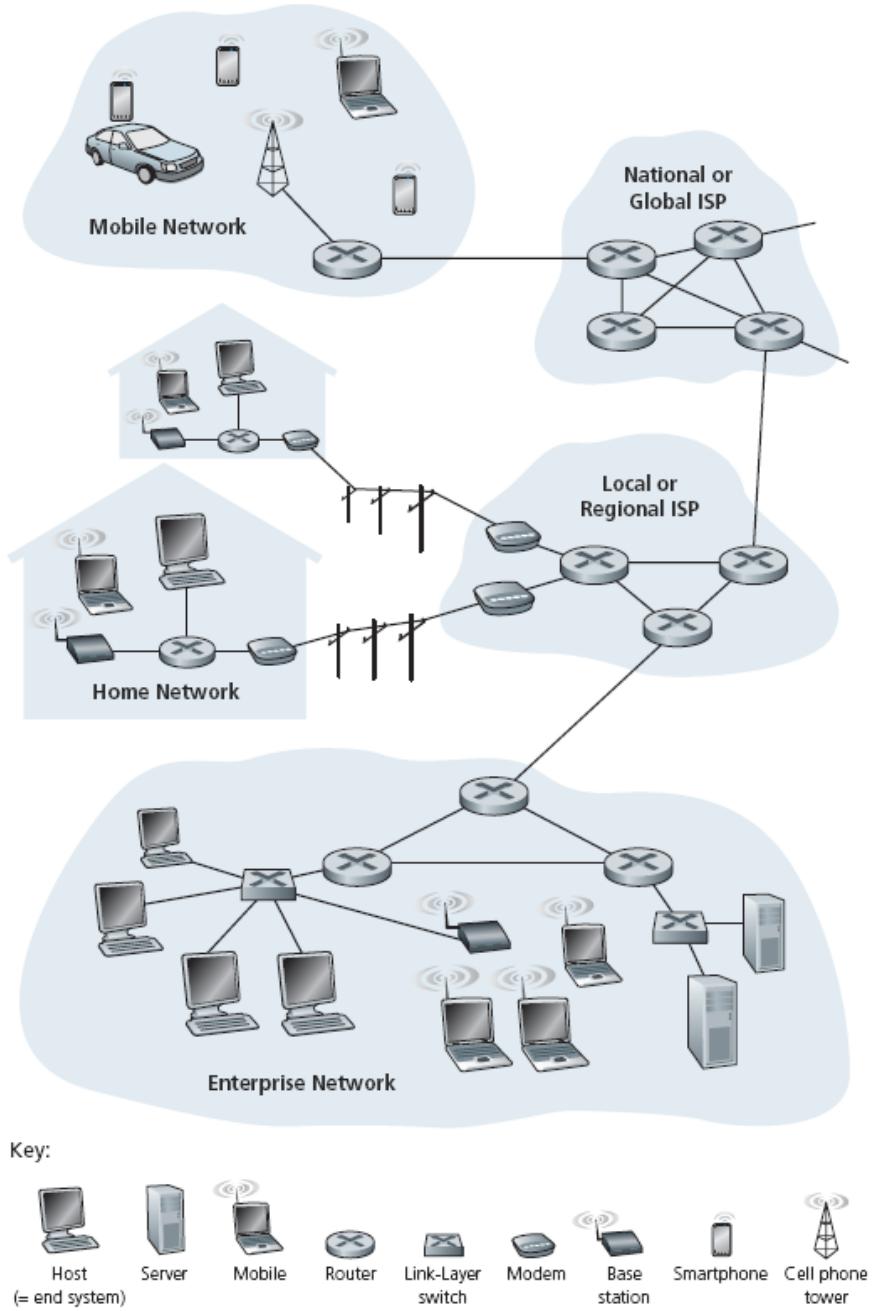
- After the spatial arrangement :

Distanța Interprocesor	Procesoare localizate în aceeași/același:	Exemple
1 m	Metru pătrat	PAN
10 m	Cameră	
100 m	Clădire	LAN
1 km	Campus	
10 km	Oraș	MAN
100 km	Țară	
1000 km	Continent	WAN
10.000 km	Planetă	Internet

**Figure. Classification after spatial arrangement**

[conform Computer Networks, 2010 – Andrew S. Tanenbaum, et. al.]

# Network types-classification



**Figure. Some pieces of the Internet**  
[Computer networking : a top-down approach  
James F. Kurose, Keith W. Ross]

# Networks types- classification

- Depending on transmission technology:
  - Broadcast networks (one channel to communicate)
    - *broadcast, multicast*
  - *Point-to-point* networks
    - *unicast*

# Network topologies

**Physical topology:** the way computers are connected in the network

**Logical topology:** the way in which data are transferred from one computer to other

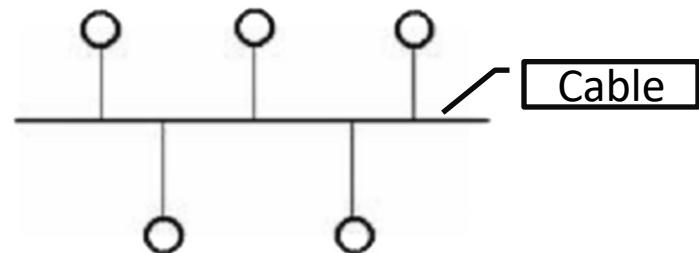
**Possible physical topologies for:**

- Broadcast networks - LAN
  - *Bus*
  - *Ring*
- *Point-to-point networks*
  - *Star*
  - *Ring*
  - *Tree*
  - Mesh

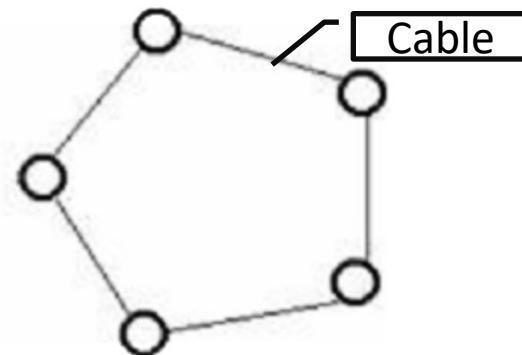
# Network topologies

## Broadcast networks - LAN

- *Magistrala (bus)*



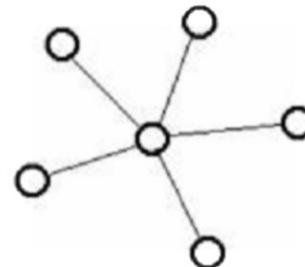
- *Inel (ring)*



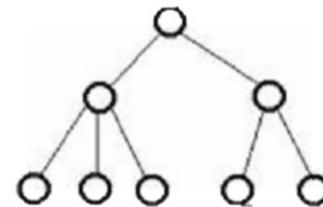
# Network topologies

## *Point-to-point* networks

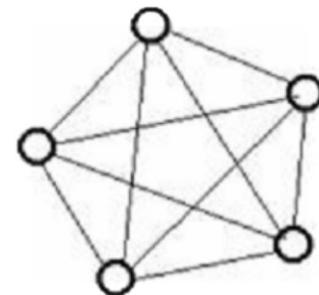
- *Star*



- *Tree*



- *Mesh*



# Networks types- classification

- Depending on the hardware technology (and software) used for interconnection:
  - Networks using **wired transmission medium**
  - Networks using **wireless transmission medium** (future course)

# Networks types- classification

- Depending on the components:
  - **Homogeneous:** the computer networks use similar configurations and protocols
    - Example: A network using Microsoft Windows via TCP/IP
  - **Heterogeneous:** the network contains different types of computers, operating systems and/or different protocols
    - Example: a LAN that connects a smart phone with an Android and an Apple Machintosh computer

# Components

*Host* – it's a computational system connected to the Internet

*Hub Network* – a device (often a signal booster) used to connect multiple devices => *network segment*



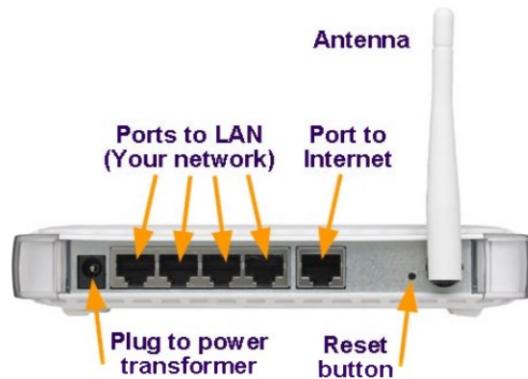
*Switch* - a device which filters network packets and resends them



Leonard-Kleinrock -> IMP  
(Interface Message Processor)  
1969

# Components

- *Router* – device providing connectivity between networks, perform routing packets between these networks



- *Bridge* – device that connects two LANs or two segments of the same LAN
- *Gateway* – is a connection point of two networks that use different base protocols
- *Repeater* - device that receives signals that it rebroadcast at a higher level or higher power, so that the signal can cover large areas without degrading its quality

# Computers Networks Structure

- Computer Network Structure – **stack levels**
  - Functionality :
    - **Interface:** ensure communication between two consecutive levels
    - **Service:** functionality provided by a level

Result: reducing design complexity
  - The principle of communication: **transmitter sends at n level what the recipient receives at the n level**
  - **Protocol** – Rules and conventions through which the communication takes place

# Example: link among - levels, protocols and interfaces

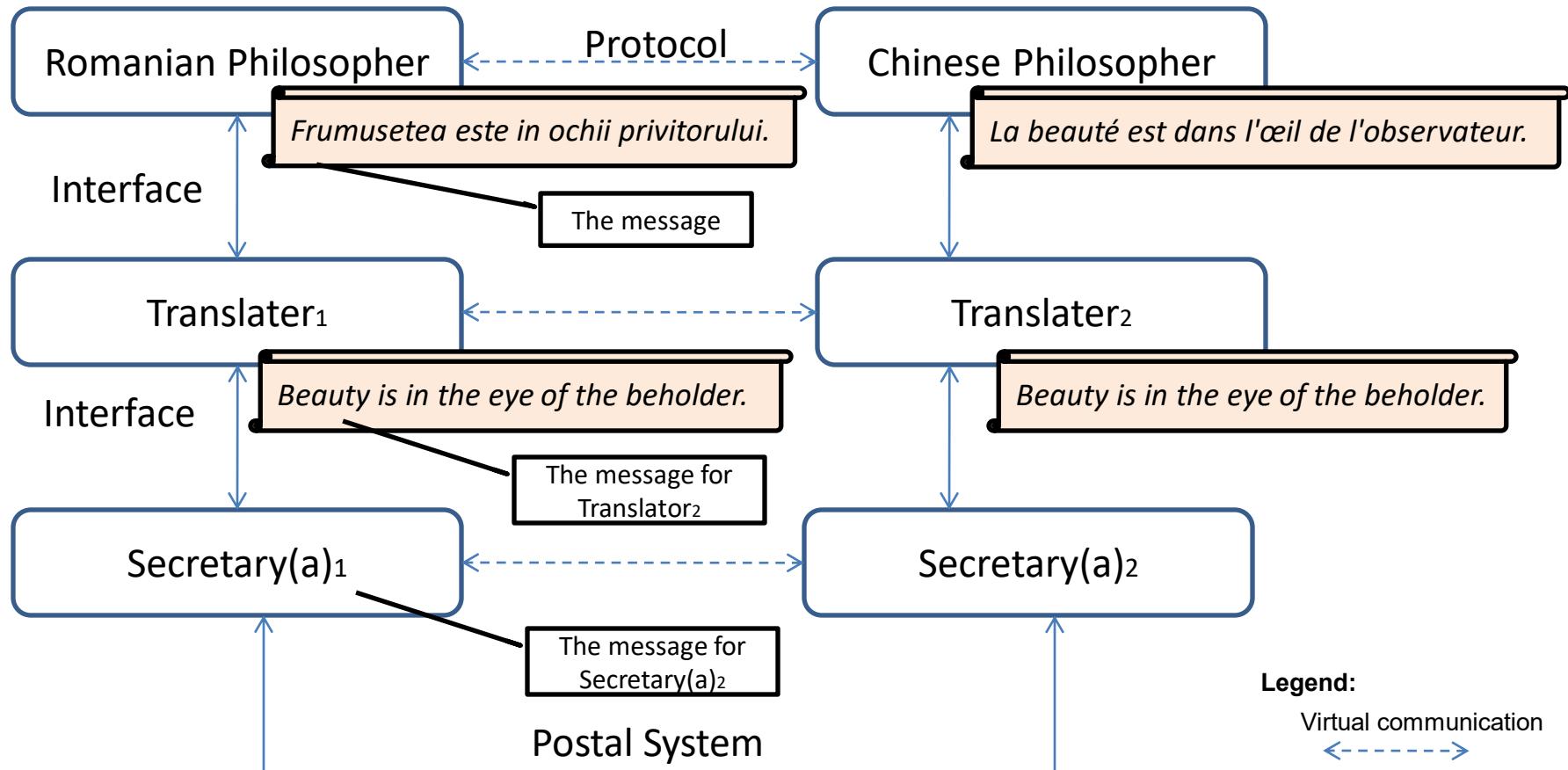
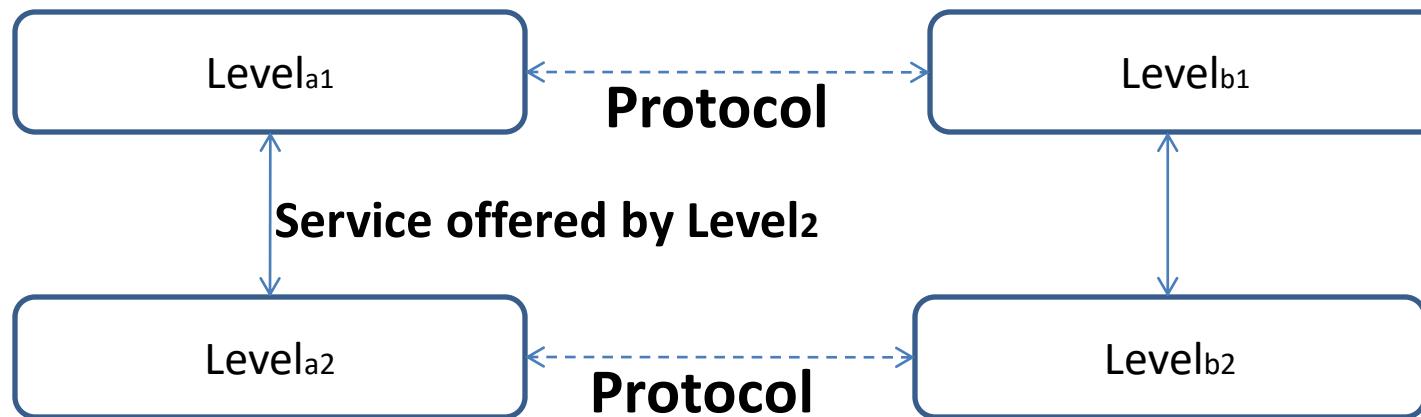


Figure: Architecture philosophie – translator - secretary

# Aspects regarding levels design

- Specifying the service is performed by a set of primitives (operations) available to the one who uses the service
- Service!= Protocol



# Aspects regarding levels designs

- Services types
  - *Connection-oriented*
    - Communication requires a connection
    - Similar to a telephone service
  - *Connectionless*
    - Communication does not require a connection
    - Similar to postal service

# Aspects regarding level design

- **Network architecture:** the set of levels and protocols
  - Architecture specification must provide sufficient information for programs or equipment intended, in order to offer the specific protocols
- **Protocols stack:** list of protocols (on all levels) used by a particular system

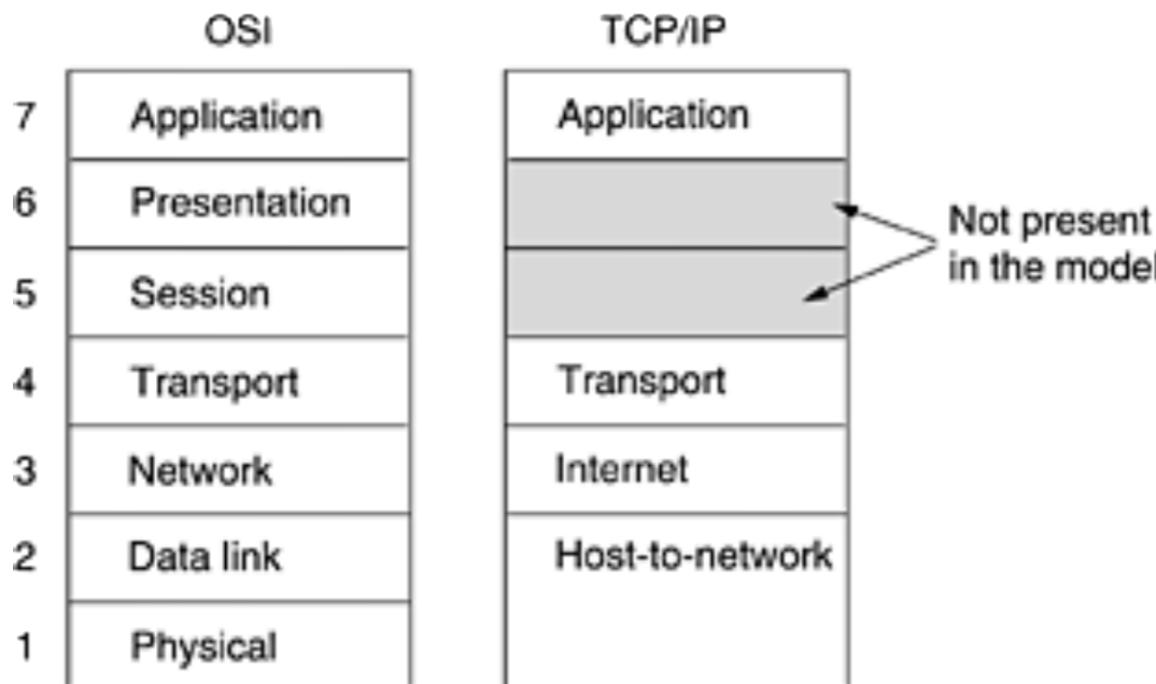


# Aspects regarding levels design

- Each level must identify transmitters & receivers through an *addressing mechanism*
- Data transfer rules identification
  - simplex communication
    - Example: TV
  - half-duplex communication
    - Example: "walkie-talkie"
  - full-duplex communication
    - Example: telephone

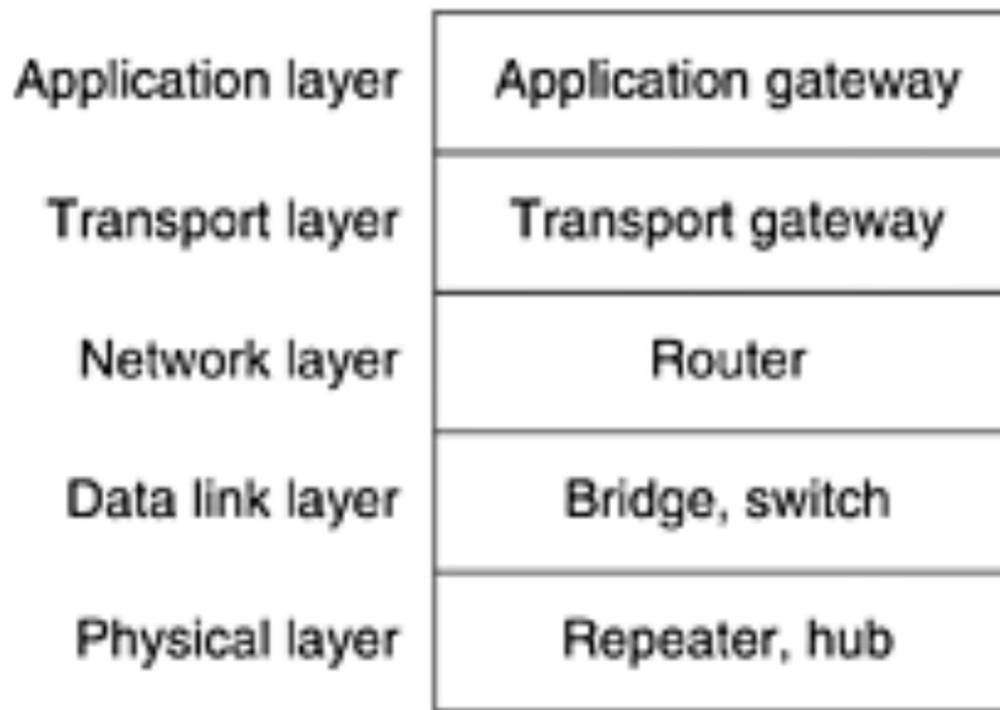
# Reference models for network architecture

- ISO/OSI (*International Standard Organization/ Open System Interconnection*)
- TCP/IP (*Transmission Control Protocol/ Internet Protocol*)



[Computer Networks, 2010 – Andrew S. Tanenbaum, et.al.]

# Network Architecture - Equipment

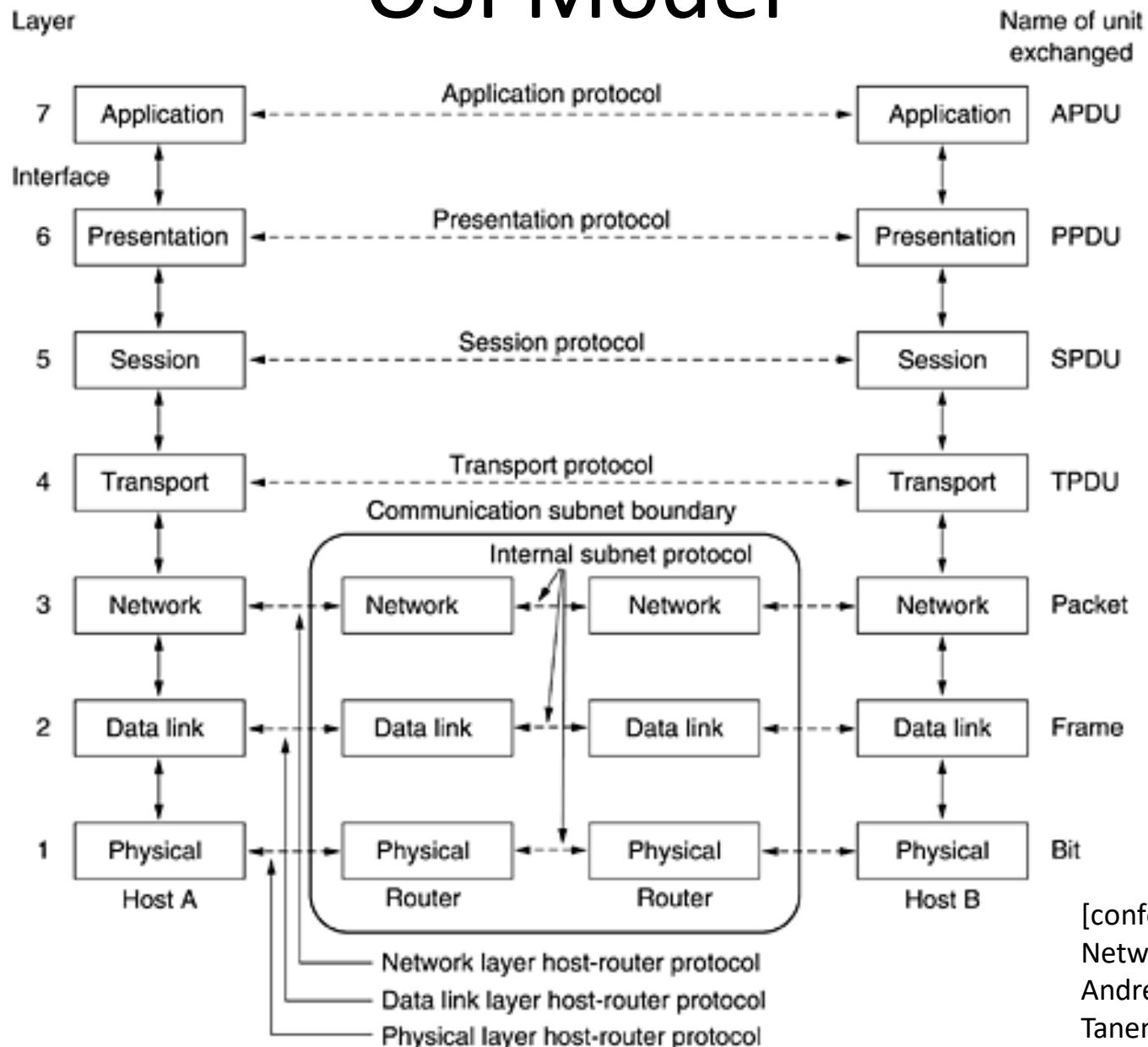


**Figure: Devices and appropriate levels**

# OSI Model- motivation

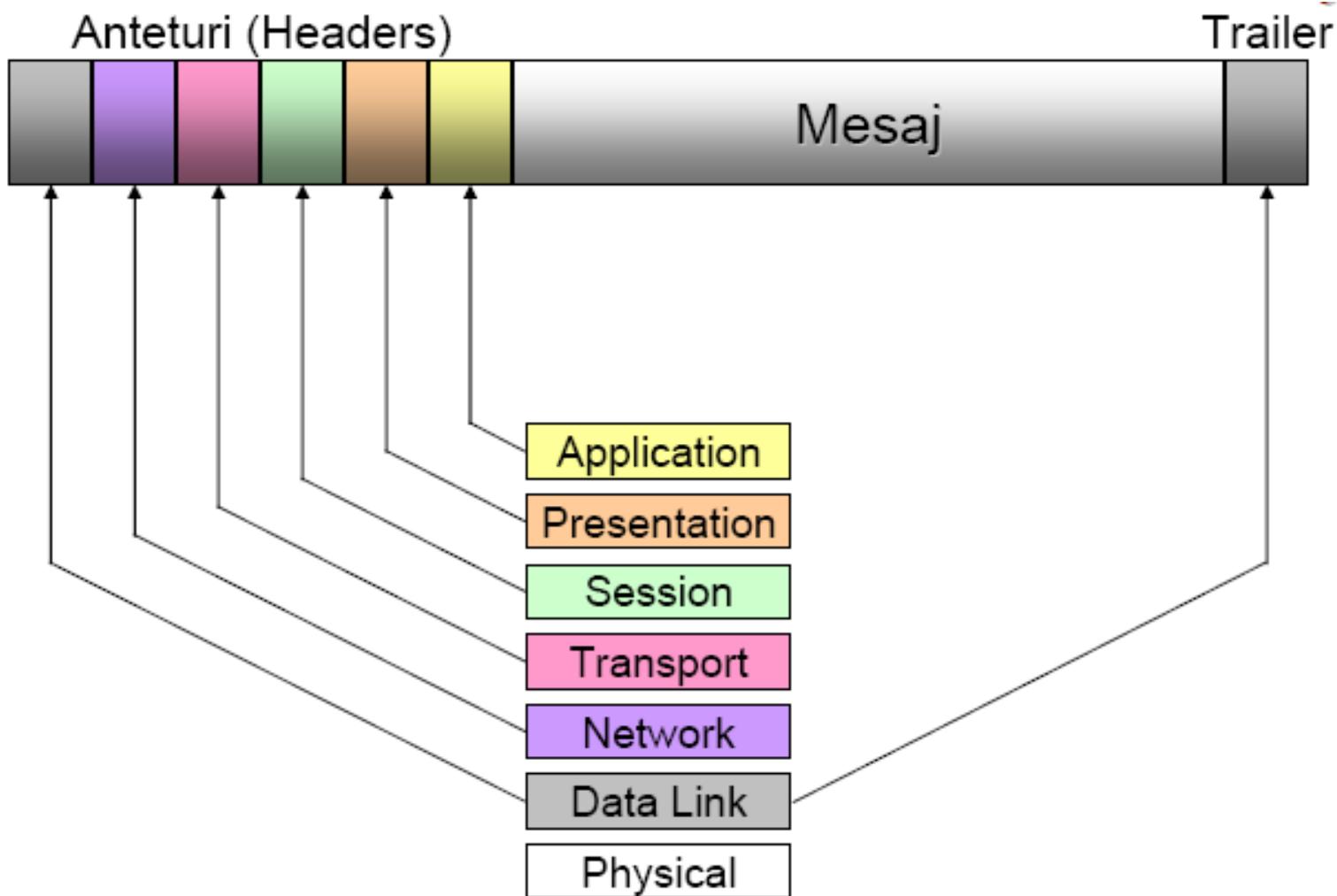
- The need for a different level of abstraction => to create a new level
  - Obs. The number of levels must be optimal, therefore each level has different functions and the whole architecture is functional
- A level has a clear role; a level function must take into account protocols that are standardized at international level
- Minimizing the flow of information between levels is accomplished through good boundary levels => Levels can be modified and implemented independently
- Each level offers services for superior level (using services from previous levels)
- “*peer*” levels of different systems communicate via a protocol

# OSI Model



[conform Computer Networks, 2010 – Andrew S. Tanenbaum, et.al.]

# OSI Model – message structure

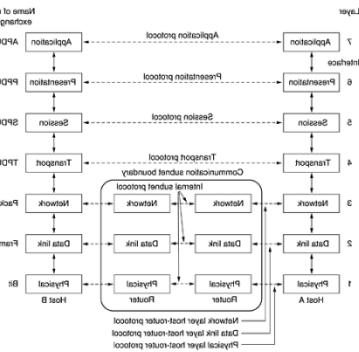


[Retele de calculatoare – curs 2007-2008, Sabin Buraga]

# OSI Model – structure

- Physical Level
- Data Link Level
- Network Level
- Transport Level
- Session Level
- Presentation Level
- Application Level

# OSI Model



## Physical Level: data transmission medium

Role: ensure that the sequence of bits transmitted from the transmitter reaches the receiver

### – Transmission media:

- Wired (twisted pair, coaxial cable, optical fiber)
- Wireless (electromagnetic spectrum - radio, microwave, infrared, ...) -> next course



# OSI Model

- **Physical Level:**

Data transmission:

- Analog (continuous values)
  - Example: telephone systems
- Digital (discrete)
  - Example: computers

Data conversion from analog to digital and vice versa:

- Modem: digital data are transmitted in analog format
- Codec (coder/decoder): analog data are transmitted in digital format

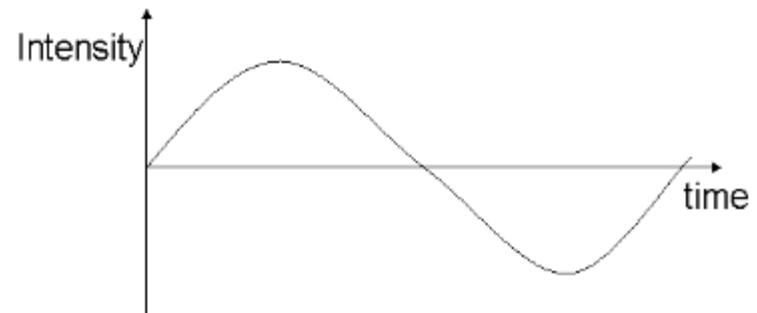


Figure. Analog Signal

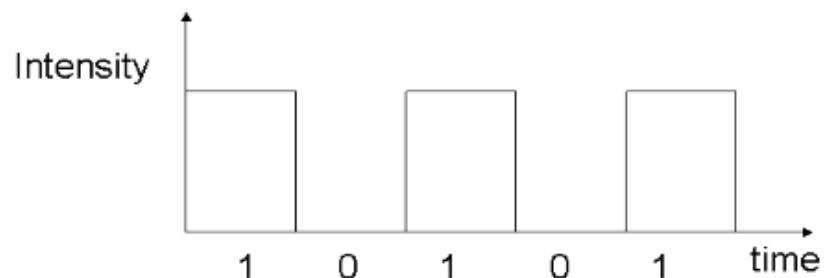


Figure. Digital Signal

# OSI Model

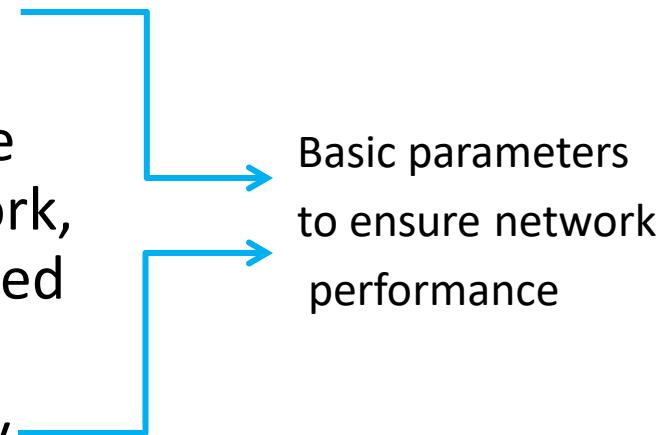
- **Physical Level - aspects**

- **Bandwidth**: the number of bits that can be transmitted over the network in a given period of time (data transfer speed)

- Usually expressed in *bits/seconds*

- **Latency**: represents the maximum time required for a bit to propagate in a network, from one end to another and it is expressed in units of time

- **RTT(Round Trip Time)** – the necessary time for a bit to cross from one end to the other and back to the environment



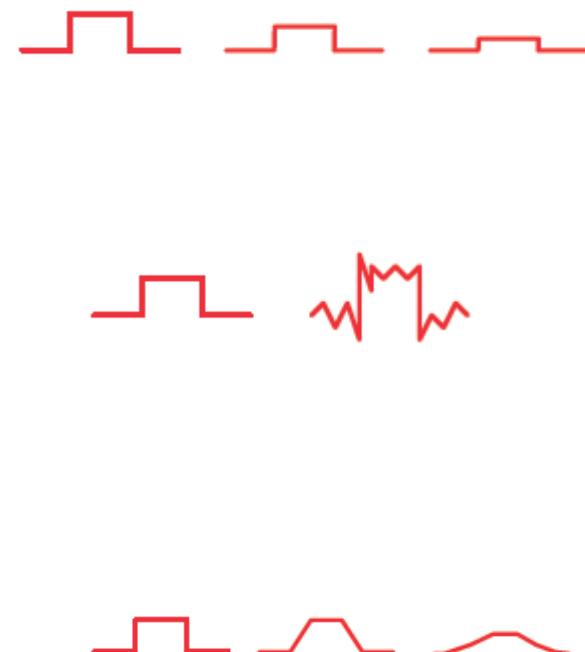
Basic parameters  
to ensure network  
performance

# OSI Model

- **Physical Level - aspects**

Modification suffered by signals during propagation:

- **Attenuation:** energy loss during signal propagation through a transmission medium
- **Noise:** signal change caused by external factors (e.g. lightning, other electronic equipment, etc.)
  - Diaphony = noise from the signal transmitted by a neighbouring transmission medium
- **Distortion-** is a deterministic change of a signal



# OSI Model

- **Physical Level - conclusions**

Offers transportation services, on which we can identify a number of possible problems

- Data can be altered / destroyed due to the noise
- If the destination cannot process the data in the right time, some will be lost
- If the same transmission medium is used by multiple transmitters, packages may alter each other
- It is less expensive to build logical connections to share the same physical medium, than create independent physical links



A new level?

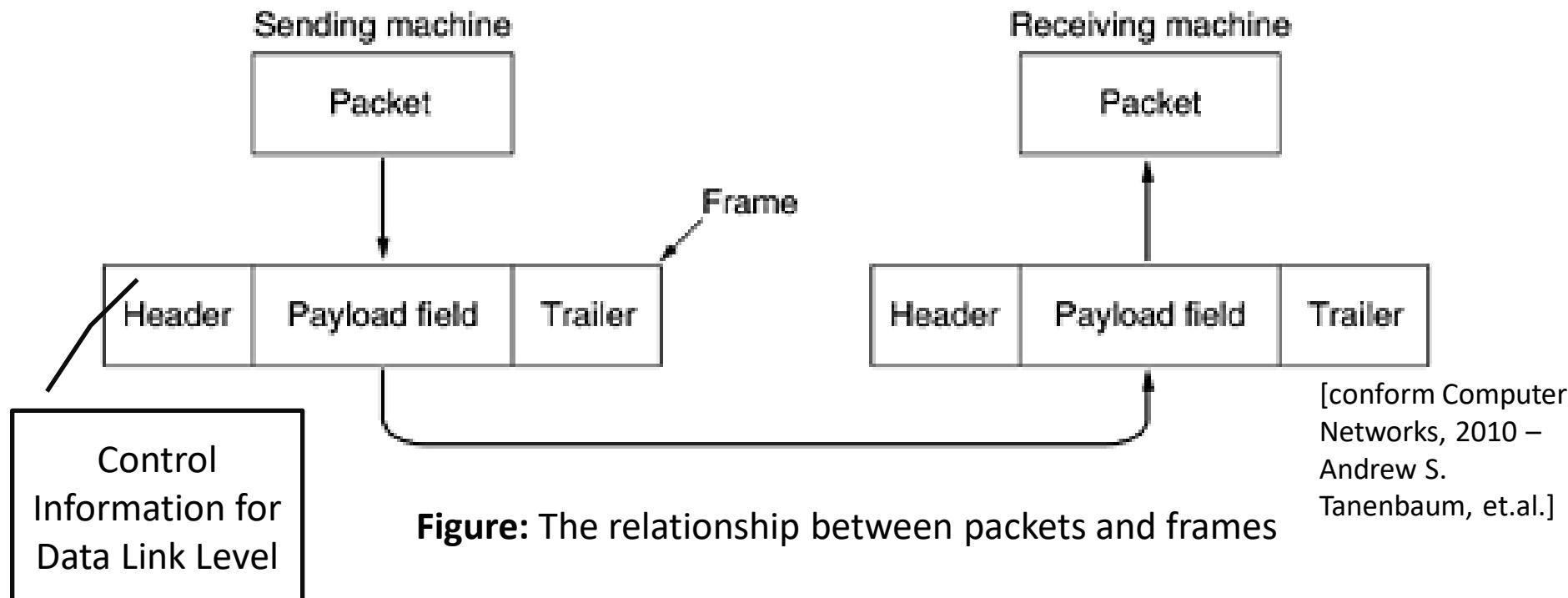
# OSI Model

- **Data Link Level:**
  - Offers:
    - mechanisms to detect and correct errors
    - regulatory mechanisms for dataflow
    - control mechanism for media access
  - Services at the network level
  - The data unit used at this level is called *frame*

# OSI Model

- **Data Link Level:**

- The data are encapsulated in *frames*
- Analogy: *frame*= digital envelope



# OSI Model

- **Data Link Level:**
  - It provides services at the network level
    - Unconfirmed connectionless service
      - » The transmitter sends independent frames to the receiver without waiting for any confirmation
      - » A lost frame is not recovered
    - Confirmed connectionless service
      - » The sent frames are confirmed
      - » The frames are not sent in order
    - Confirmed connection-oriented services
      - » A connection is established before the transmission
      - » Frames are numbered to keep the right order

# OSI Model

- **Data Link Level:**
  - Divided into two sublevels:
    - **LLC (Logical Link Control)**
      - Role: Provides an independent view of the medium at a superior level
    - **MAC (Medium Access Control)**
      - Role: Used to determine who is to transmit into *multi-access channel*

# OSI Model

- **Data Link Level:**

## MAC (Medium Access Control)

- Context of the problem: the same physical environment is used by more emitters (uniquely identified by a physical address or MAC address) operating simultaneously, for example:
  - Half-duplex transmission between entities that use the same physical environment for both directions
  - communication by radio when there are stations that emit on the same wavelength (Wireless Ethernet - IEEE 802.11, Bluetooth, etc.)

# OSI Model

- **Data Link Level:**

## MAC (Medium Access Control)

- Strategies:

- **Static allocation**

- » FDM (Frequency Division Multiplexing)

- » TDM (Time Division Multiplexing)

- Accepting the possibility of collisions and retransmitting packets affected by collisions - **dynamic allocation**

- Collision = data is simultaneously transmitted

- General mechanism: a station that has data to send, transmit them immediately; if the collision appears, the resend action is performed

# OSI Model

- **Data Link Level:**

Medium Access Control – **protocols:**

- ALOHA
  - Pure ALOHA : “*send whenever you want*”
  - Slotted ALOHA
- **CSMA (Carrier Sense - Multiple Access):** protocol with transmission detection (“*free channel before sending?*”)
  - *1-persistent CSMA*
  - ...
  - *p-persistent CSMA*

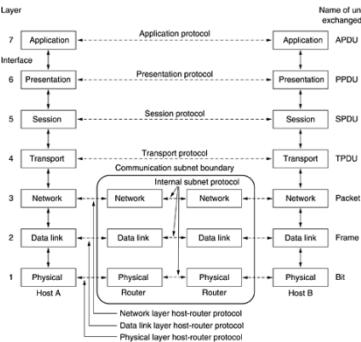
# OSI Model

- **Data Link Level:**
  - Medium Access Control – **protocols:**
    - **CSMA** (Carrier Sense - Multiple Access)
      - CSMA/CD (*CSMA with Collision Detection*)
        - » “*free channel while transmit?*”
        - » Based on Ethernet LAN (IEEE 802.3)
      - **MACA (Multiple Access with Collision Avoidance)**
        - The basis for wireless networks (IEEE 802.11)
      - **MACAW**
        - Improves MACA

Standard	Description
IEEE	
802	Group standards for LAN and MAN
802.2	LLC (Logical Link Control)
802.3	Ethernet (Carrier Sense Multiple Access with Collision Detect (CSMA/CD))
802.3u	Fast Ethernet
802.3z	Gigabit Ethernet
802.11	Wireless (WLAN)
a/b/g/n/ac	
802.15	Wireless PAN ( 802.15.1 Bluetooth, ...)
802.16	Wireless WAN

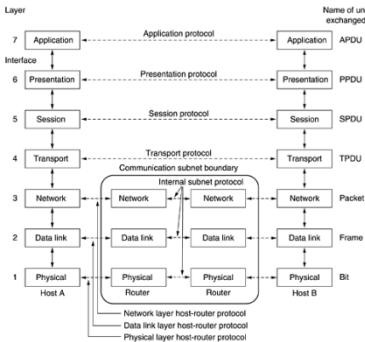
## Medium Access Control – Standards Example

# OSI Model



- **Network Level:**
  - Retrieves packages from the source and transfer them to the destination
  - It provides services to transport level
    - What services?
      - Internet community proposes:
        - » Connectionless services: SEND PACKET, RECEIVE PACKET
        - » Packages (called **datagrams**) are independent and are managed individually
        - » Datagram services are similar to a typical post system

# OSI Model

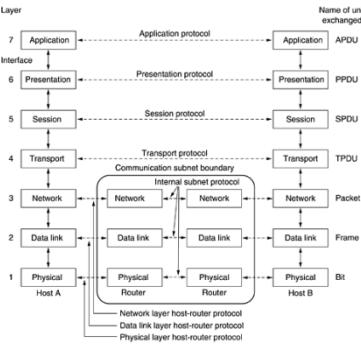


- **Network Level:**
  - Retrieves packages from the source and transfer them to the destination
  - It provides services to transport level
    - What services?
      - Telephone companies propose:
        - » Connection-oriented service – safe services
        - » Before the transfer some negotiations are initiated to establish a connection (VC-virtual circuit)
        - » These services are similar to the telephony system

# OSI Model

- **Network Level:**
  - Used Protocols
    - X.25 (Connection-oriented)
    - IP
  - Problems
    - Protocol conversions and addresses
    - Error control (flow, congestion)
    - Dividing and recomposing packages
    - Security – encryption, *firewall*

# OSI Model



- **Transport level:** it offers safe and cost-effective data transport from the source machine to the destination machine, independent of physical network or networks currently in use

**Services:** provides connection-oriented and connectionless services

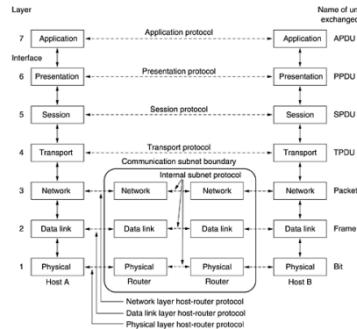


Differences between the transport and network layer?

# OSI Model

- **Transport level:**
  - **Primitives:**
    - LISTEN – it's a blocking operation until a process tries to connect
    - CONNECT – trying to establish a connection
    - SEND – send data
    - RECEIVE – it's a blocking operation until data is received
    - DISCONNECT – connection release
  - **Performance** - quality of service (QoS - Quality of Service): establishing /releasing the connection, error rate, protection, priority, resilience (the probability that a connection shut down because various internal reasons), duplicate packets, flow control

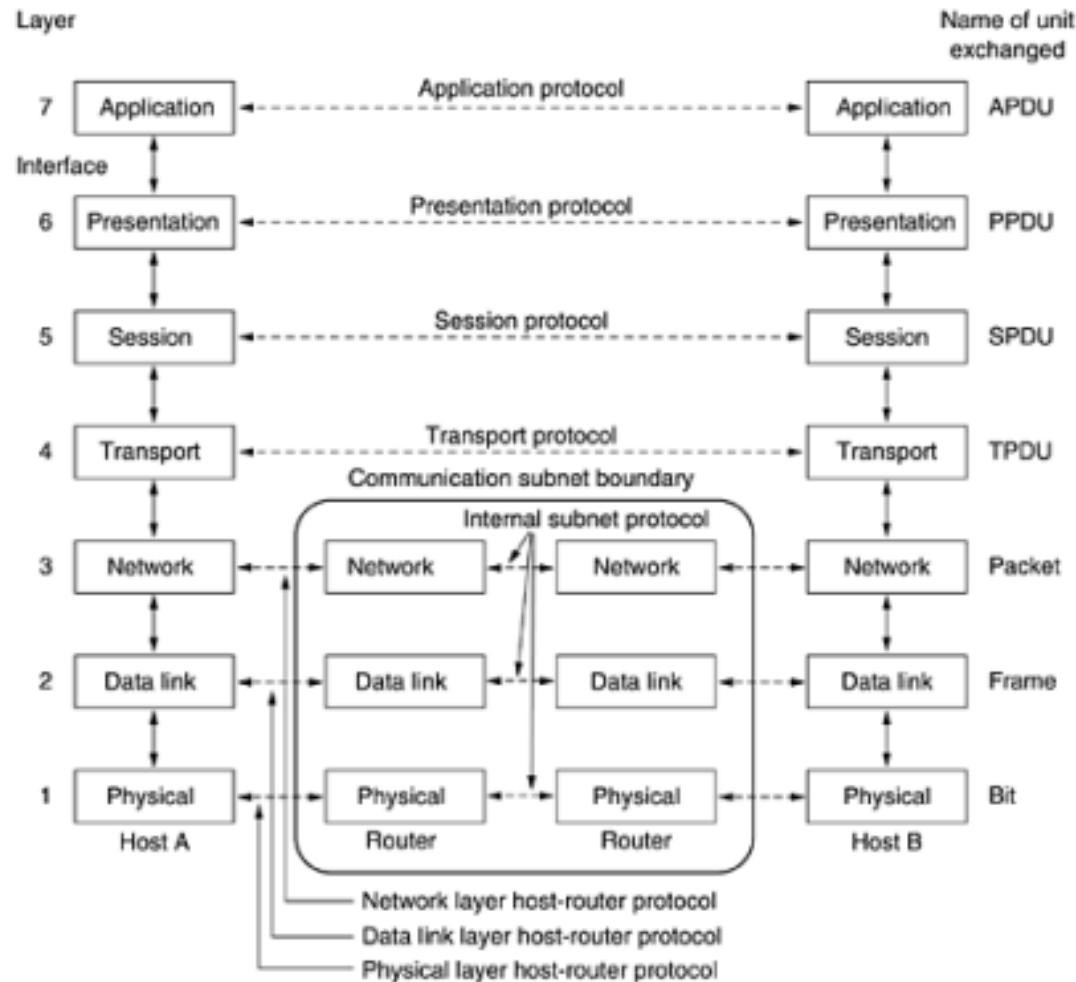
# OSI Model



- **Session level:** refers to problems linked to session settings (dialogue control services, synchronisation etc.)
- **Presentation level:** handle data presentation, codified them into standard format
  - To ensure communication among computers with different representations, the presentation level ensures the conversion of internal data in standardized network representation and vice versa

# Modelul OSI

- **Application level:**  
manage network services: virtual terminal, file transfer, electronic mail, remote execution of applications, ...



# TCP/IP Model

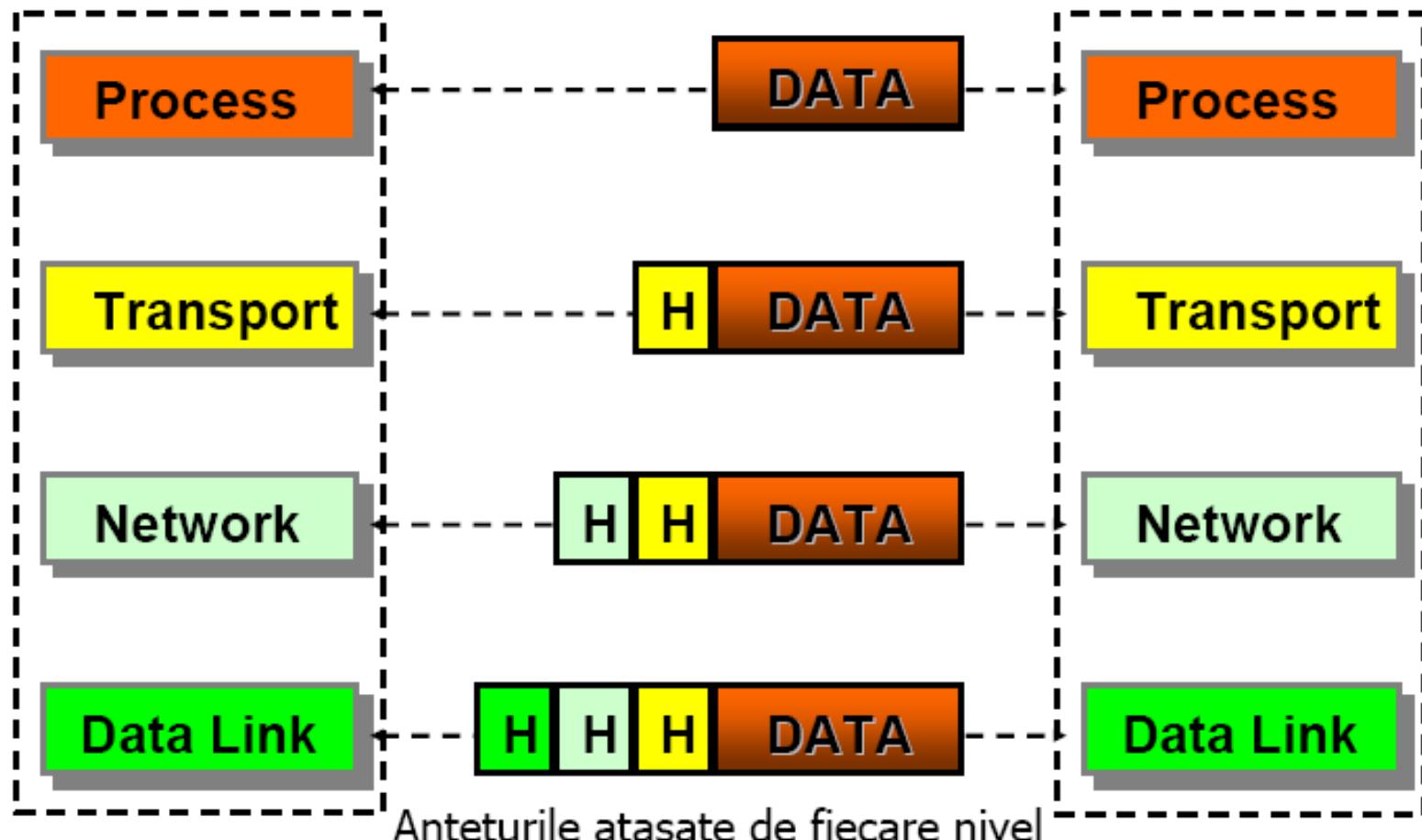
- Terms:
  - end-system – *host*
  - network - provides support for data transfer between end systems
  - internet - collection of networks (interconnected)
  - Sub-network - part of the internet
  - intermediate system - connects two sub-networks

# OSI versus TCP/IP

TCP/IP Model	TCP/IP - Protocols	OSI Model
Application	FTP, Telnet, HTTP, ...	Application
		Presentation
Transport	TCP, UDP, ...	Session
		Transport
Internetwork	IP, ...	Network
Host to Network	Ethernet, ...	Datalink
		Physical

Figure: Overview of models OSI and TCP / IP

# TCP/IP Model



[Retele de calculatoare –  
curs 2007-2008, Sabin Buraga]

# TCP/IP Model

- It provides the ability to interconnect multiple network types
- Network and Transport levels are the kernel of this model
- Successfully implemented over Ethernet (IEEE 802.3) - supported by many implementations of the physical layer (coaxial cable, twisted pair, fiber optic)

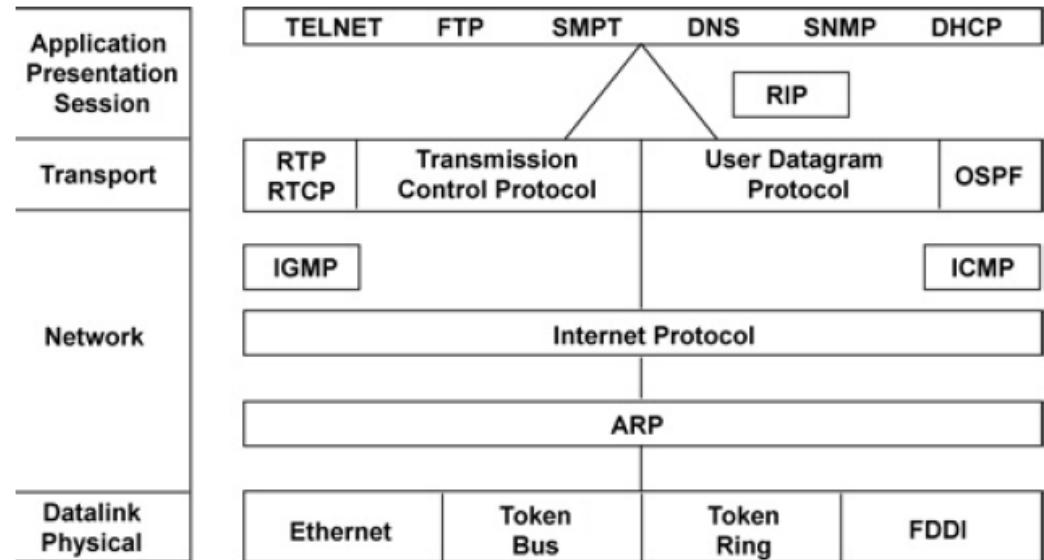
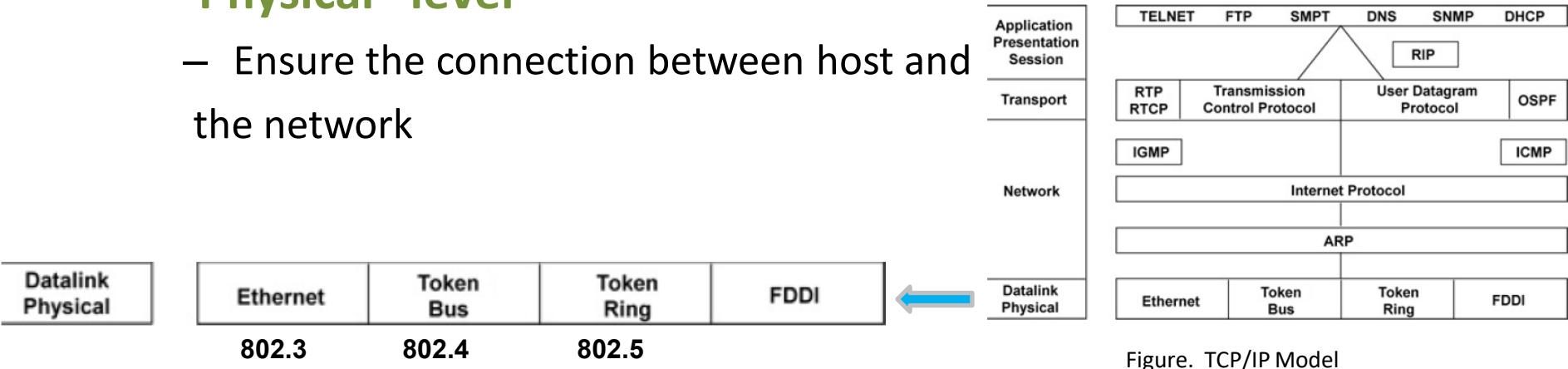


Figure. The TCP / IP - protocols

# TCP/IP Model

- “Physical” level

- Ensure the connection between host and the network



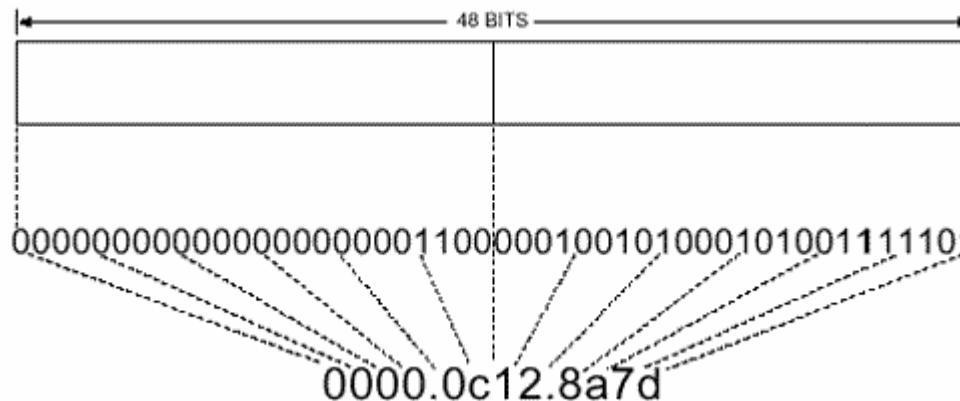
## Ethernet

- It provides multiple access (shared transmission medium) in a network
- Collision Detection: CSMA / CD (Carrier Sense Multiple Access with Collision Detection)
- Each Ethernet interface has a unique address 48 bits: hardware address (MAC) - e.g. C0: B3: 44: 17: 21: 17
  - Addresses are assigned to NIC (Network Interface Card) producers by a central authority

# TCP/IP Model

## Ethernet

- Each interface (board) network has a unique MAC address (some operating systems allow it to be modified by software)



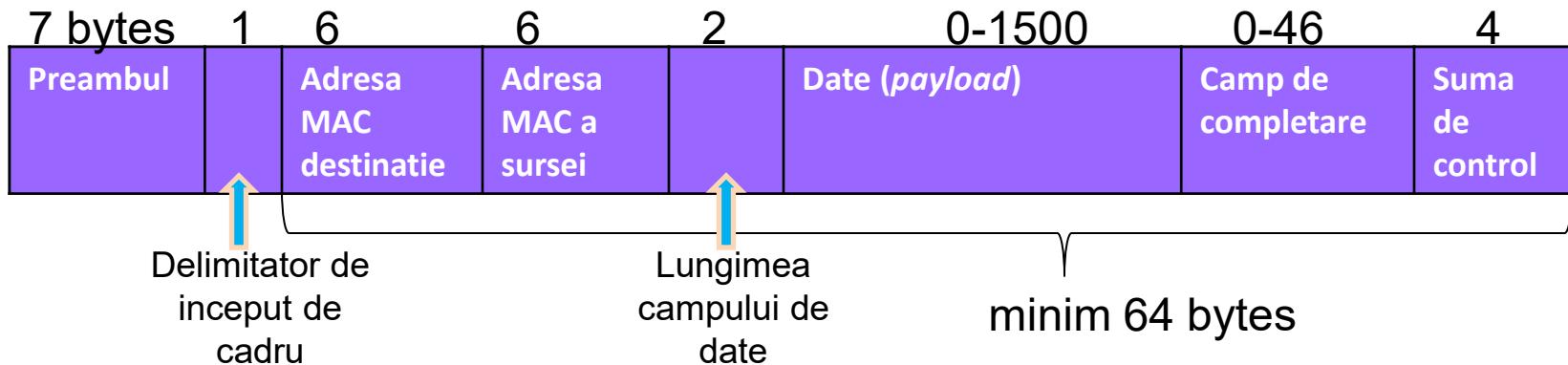
- The first 24 bits identify the manufacturer

[ conform Retele de calculatoare –  
curs 2007-2008, Sabin Buraga]

# TCP/IP Model

## Ethernet

- A frame format:



- Broadcast: the address has all bits set to 1
- Each network interface inspecting the destination address in each frame
- If the destination address does not match with the hardware address or the broadcast address, then the frame is ignored

# TCP/IP Model

**Ethernet** – standards (examples):

- 10 BASE5: 10 Mbps using thick coaxial cable (Thick Ethernet)- 1980
- 1BASE5: 1 Mbps using two Ethernet cables (Unshilded Twisted Pair)
- 10BASE-T: 10Mbps using 2 pairs UTP– 1990
- 10BASE-FL: 10 Mbps optical fiber with point-to-point link
- 10BASE-FB: 10Mbps backbone with optical fiber
- 100BASE – FX: 100MBps CSMA/CD with two optical fiber, full duplex
- ... etc

# TCP/IP Model

## Ethernet versus Fast Ethernet

	<b>Ethernet</b>	<b>Fast Ethernet</b>
Viteza	10 Mbiti/s	100 Mbiti/s
Protocolul MAC	CSMA/CD	CSMA/CD
Diametrul retelei	2.5 km	205 m
Topologie	Magistrala, stea	Stea
Tip cablu	Coax, UTP, fibra	UTP, fibra
Standard	802.3	802.3u
Cost	c	2*c

[conform Retele de calculatoare –  
curs 2007-2008, Sabin Buraga]

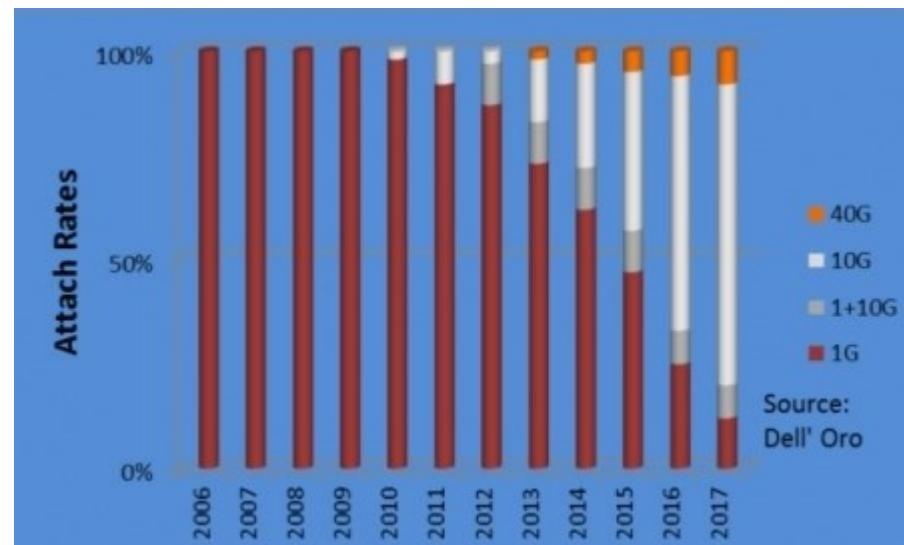
# TCP/IP Model

- **Gigabit Ethernet**

- Implementations for both copper wires (802.3ab), and fiber (802.3z)
- The difference from other Ethernet implementations is at physical level

- **10 Gigabit Ethernet**

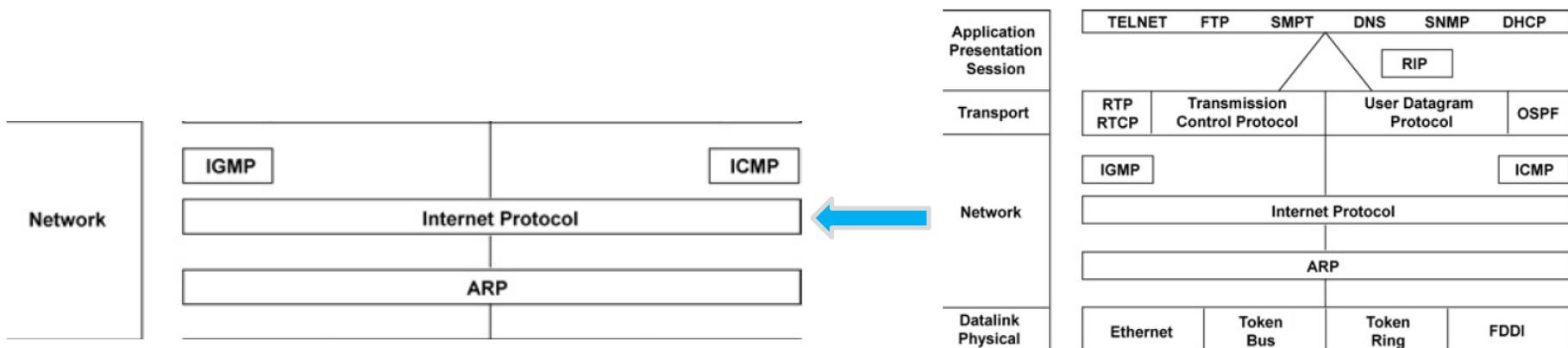
- Implementations for fiber (802.3ae)
- Operates at distances of 40km (useful for MAN and WAN)
- Frame format is similar to other implementations of Ethernet



[<http://www.networkcomputing.com/networking/will-2014-be-the--year-of-10-gigabit-ethernet/a/d-id/1234640?>]

# TCP/IP Model

- **Network Level**
  - It allows hosts to emit a packet in any network; packages travel independently up to destination



- Highlights:
  - routing packets
  - congestion avoidance

# TCP/IP Model

- **Network Level**
  - Level design aimed at achieving the following objectives:
    - The services offered are independent from the technology used (e.g. routers)
    - Provide transport level services, which allow it to operate independently of number, type and topology
    - It provides a unique mechanism to address LANs and WANs

# TCP/IP Model

- Network Level
  - IPv4
  - IPv6
  - Routing
    - OSPF(*Open Shortest Path First*) – RFC 1131
    - BGP(*Border Gateway Protocol*) – RFC 1105
  - Multicast:
    - IGMP (*Internet Group Management Protocol*) – RFC 1112, 1054
  - Control:
    - ICMP (*Internet Control Messages Protocol*) - RFC 792,777
    - SNMP (*Simple Network Management Protocol*) – RFC 1157
    - ICMPv6

# Network Level | IP

- **Initial Situation**

Before the Internet, only nodes from the same network could communicate with each other

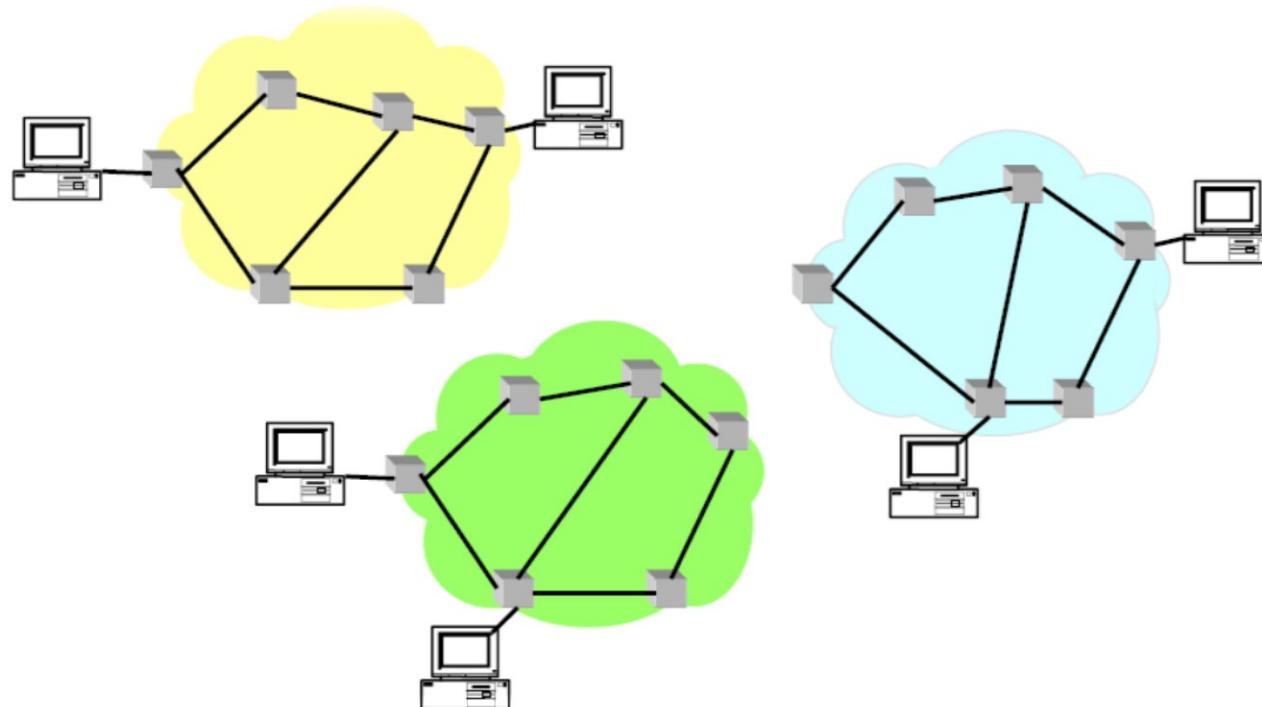
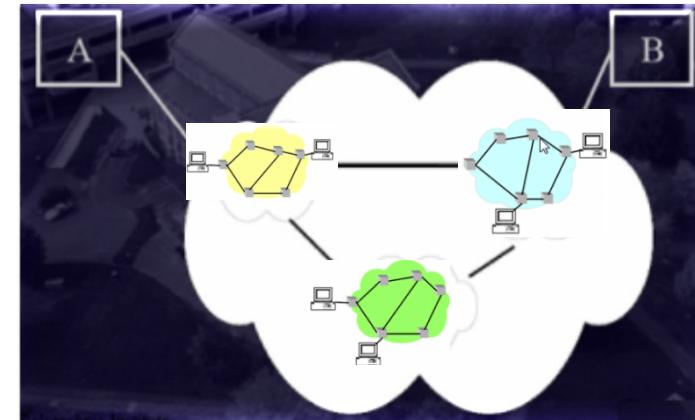


Figure: Individual Network

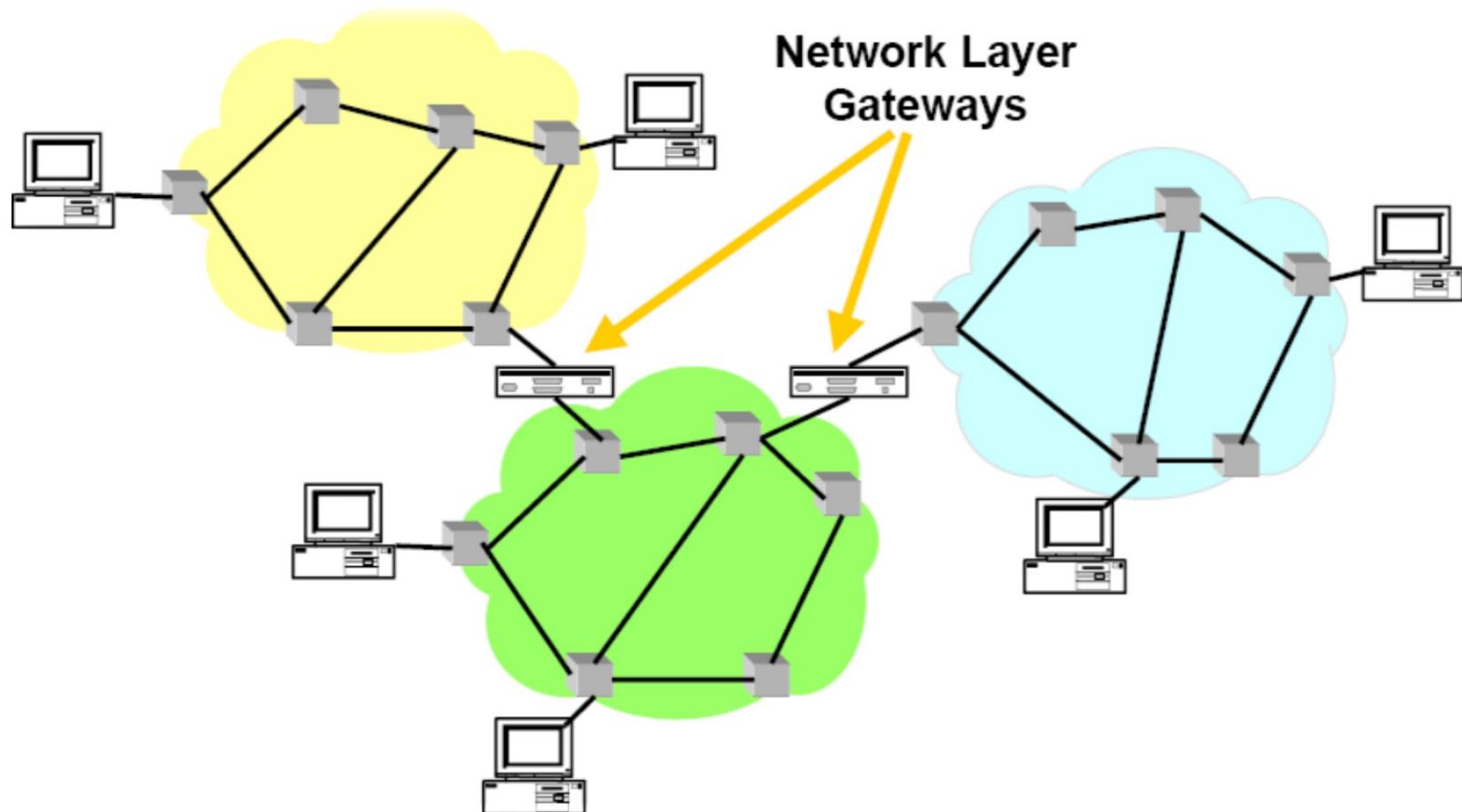
# Considerations

- Problem
  - How to carry packages in a heterogeneous environment?
  - **Heterogeneity**
    - At lower levels: how to make the interconnection of a large number of independent networks?
    - At higher levels: how to provide support for a wide variety of applications?
  - **Scaling:** how could we handle a large number of nodes and applications in such a system of interconnected networks?



# Solution

- IP – Internet Protocol



# Network Level | IP

- IP protocol is used for autonomous systems (AS - Autonomous Systems) in order to interconnect

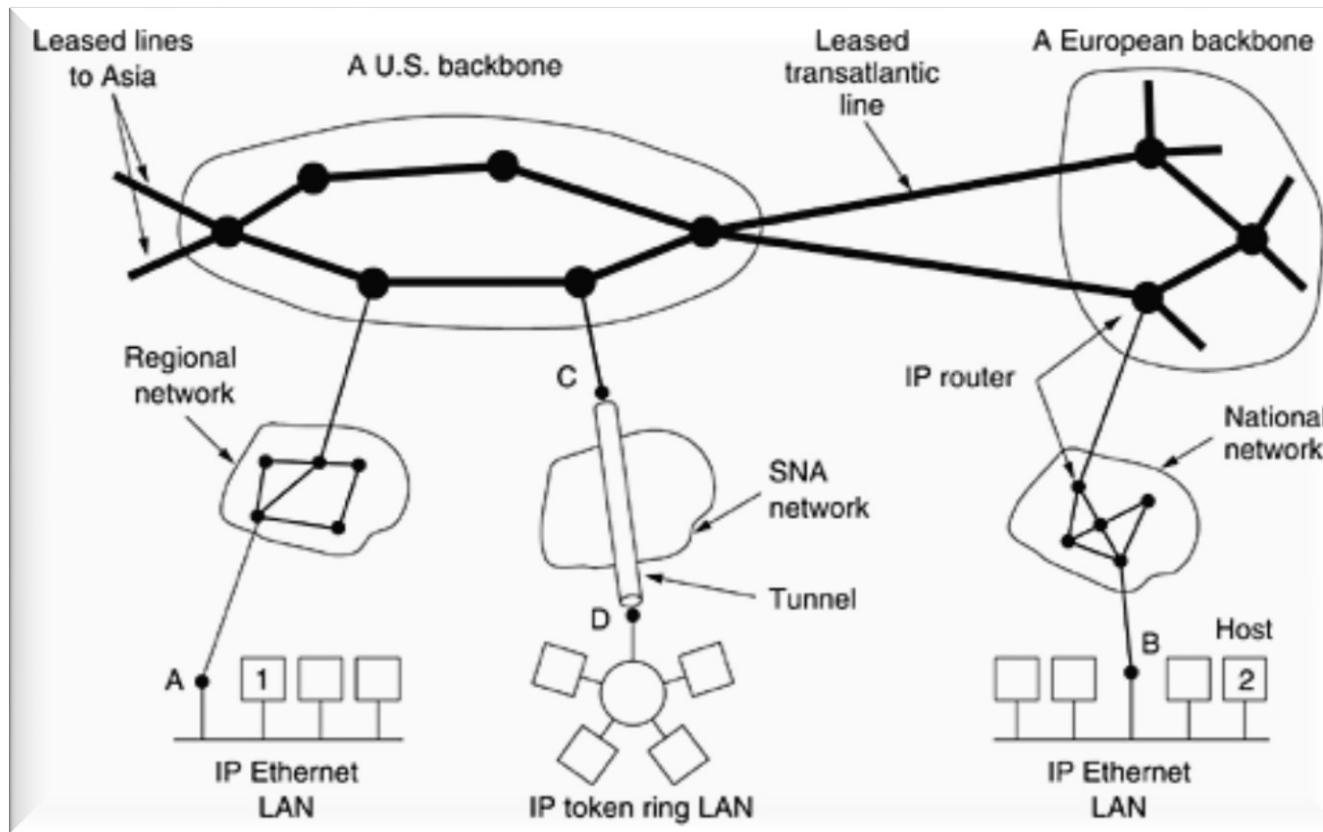


Figure: Internet  
- collection of  
interconnected  
networks

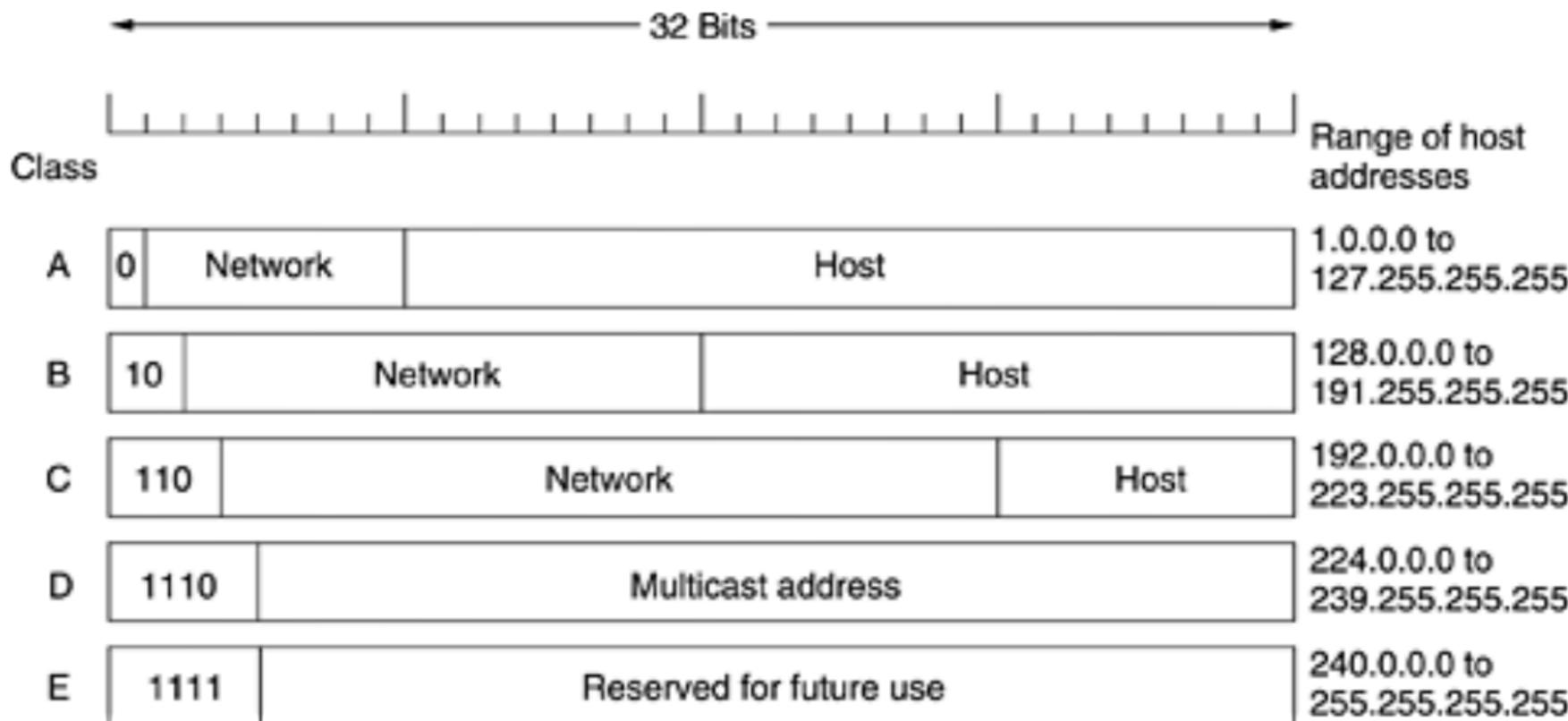
[Computer Networks, 2003  
Andrew S. Tanenbaum]

# IP Protocol

- **IPv4 Addresses**
  - Each IP address includes a **network identifier(NetID)** and a **host identifier(HostID)**
  - Each network interface has a single IPv4 address
  - An IPv4 address has a length of 32 bits
  - Initially (RFC 791) there was a division into network classes: A,B,C,D,E

# IP Protocol

- IPv4 Addresses



[Computer Networks, 2003  
Andrew S. Tanenbaum]

# IP Protocol

- **IPv4 Addresses**

- Class A: 128 possible networks,  $2^{24}$  hosts/network
- Class B:  $2^{14}$  possible networks,  $2^{16}$  hosts/network
- Class C: over 2 million networks, 255 hosts/network
- Network Identifier(NetID) is assigned by a central authority (NIC – *Network Information Center*)
- Host Identifier(HostID) is assigned locally by a network administrator
- Example: 85.122.23.145 – Class A (in decimal notation convention)  
0101 0101 0111 1010 0001 0111 1001 0001
- For IPv6, hexadecimal representation is recommended

# IP Protocol

- **IPv4 Addresses**

- An interface network has assigned a unique IP address
- A host can have multiple NICs, therefore it has multiple IP addresses
- The hosts of the same network have the same network identifier (the same NetID)
- *Broadcast* addresses have HostID's bites equaled to 1
- The IP address in which all HostID's bites are 0 is called a **network address** – refers to the whole network
  - Example: 85.122.23.0 (*network address* for a host such as 85.122.23.145 and 85.122.23.1)
  - 127.0.0.1 – *loopback address (localhost)*

# IP Protocol

## • IPv4 Addresses

- From the address space, some addresses are reserved: (RFC 1918):
  - 0.0.0.0 – 0.255.255.255
  - 10.0.0.0 – 10.255.255.255 (private addresses)
  - 127.0.0.0 – 127.255.255.255 (*loopback* addresses)
  - 172.16.0.0 - 172.31.255.255 (private addresses)
  - 192.168.0.0 - 192.168.255.255 (private addresses)
- Private addresses : addresses that are not accessible to the outside (the "real"Internet), but only in the organization's intranet

# IPv6

- Context:
  - Issues in IPv4 addresses world:
    - The exponential growth of the hosts` number
    - Very large routing tables
    - Complex configurations, more and more users (and increasing)
    - Failure to ensure QoS
  - Pressure from mobile operators

# IPv6

- Objectives for a new protocol:
  - Support for billions of hosts
  - Reducing routing tables
  - Simplifying Protocol
  - Support for mobile hosts
  - Compatibility with the old IP
  - Support for future developments of the Internet
  - RFC 2460, 2553

# IPv6



- 6 June 2012

# IPv6

- Aspects:
  - IPv6 addresses are 16 bytes in length -  $2^{128}$  addresses
  - Note: 16 hexadecimal numbers, 2 digits each, separated by ":"
    - Example: 2001:0db8:0000:0000:0000:0000:1428:57ab
    - If one or more groups of 4 digits is 0000, the zeros may be omitted and replaced (once) with "::"
    - Example: 2001:0db8::1428:57ab
  - To maintain compatibility, public IP addresses are considered a subset of IPv6 address space
  - IPv4 addresses in IPv6 can be written as: 10.0.0.1 -> ::10.0.0.1 or 0:0:0:0:0:A00:1

# IPv6

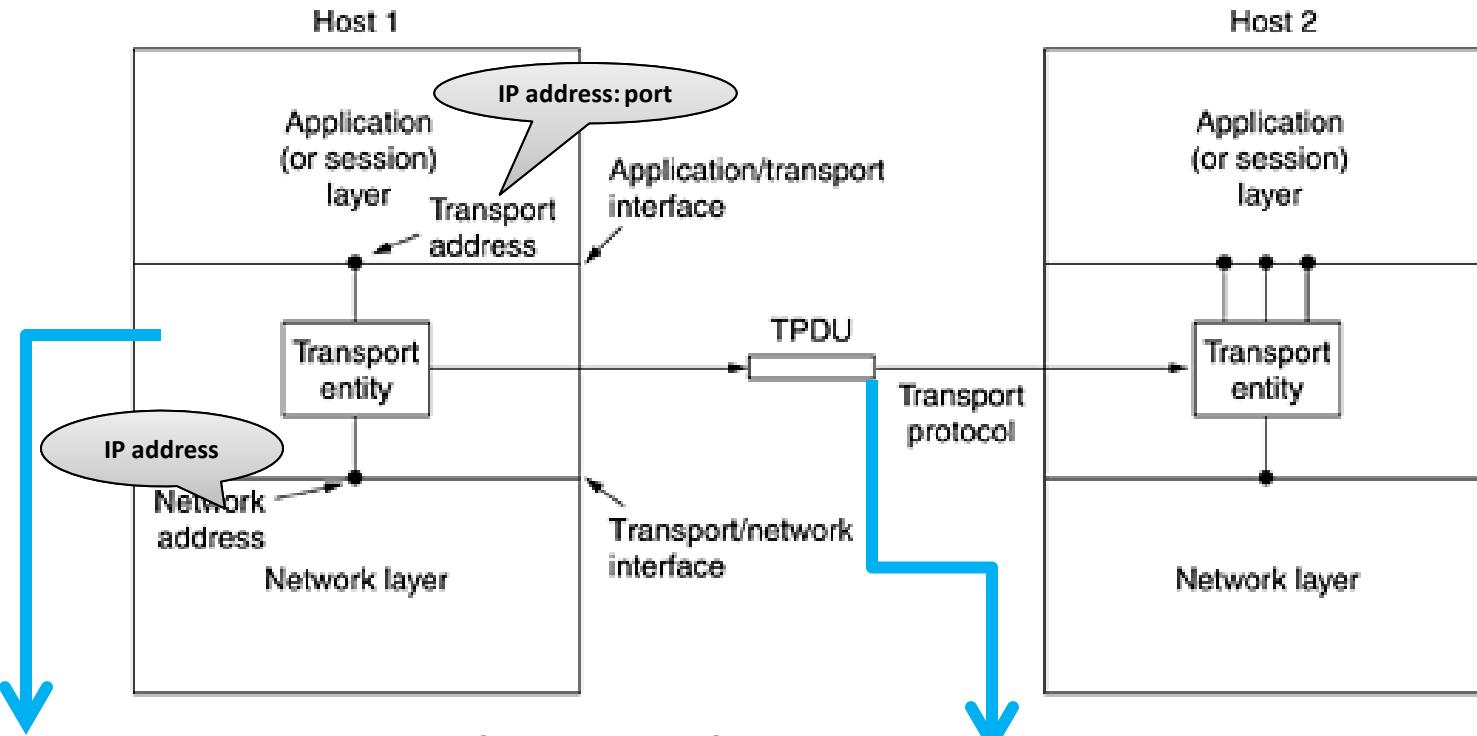
- ICMPv6
  - ICMP provides functions (reporting data transmission, errors, etc.) plus:
    - Neighbor Discovery(*Neighbor Discovery Protocol – NDP*) - Replaces the ARP
    - Multicast listener discovery(*Multicast Listener Discovery*) – replaces IGMP (*Internet Group Management Protocol*)
  - Details in RFC 4443

# TCP/IP Model

- **Transport level**
  - Ensures the realization of communication between the source host and destination host
  - Protocols
    - ***TCP (Transmission Control Protocol)*** - RFC 793,761
    - ***UDP (User Datagram Protocol)*** – RFC 768
    - ***Other Protocols: SCTP (Stream Control Transmission Protocol)*** – RFC 4960, 3286 (2960, 3309);***DCCP (Datagram Congestion Control Protocol)*** – RFC 4340, 4336;

# Transport Level

- The relationship between levels: network-transport-application



**Transport entity:** uses the network services level and provides services to a superior level

**TPDU (Transport Protocol Data Unit)** – the data transmission unit

[conform Computer Networks, 2010  
– Andrew S. Tanenbaum, et.al.]

# Transport Level

- Offers much more reliable services than those at the network layer (e.g. packets lost at the network layer can be detected and the situation can be fixed at the transport level)

## QoS (*Quality of Service*)

- The delay in establishing the connection
- Productivity or transfer rate
- Delay in transfer
- The residual error rate
- Protection
- Priority
- Resilience

# Transport Level

- Provides logical communication between processes running on different hosts (*end-to-end communication*)
  - (Previous Course!) Network level provides logical communication between *hosts*

## PORT

- Add a new dimension of IP addresses from the network level
- It is associated to an application (service) and not a host
- A process can provide more services => can use multiple ports
- A service can correspond to several processes

# Transport Level

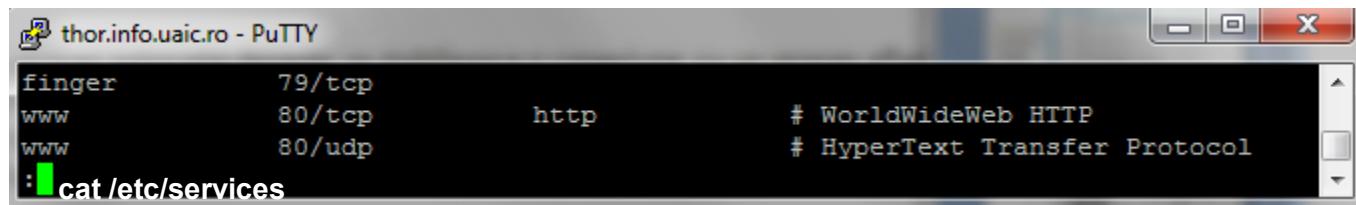
## PORT

- It is a 16-bit number and can be set between 0-65535
  - 1-1024 – reserved values (*well-known*), 1-512 system services (IANA – Internet Assigned Number Authority: RFC 1700)

Examples:

/etc/services : system services are associated with ports

HTTP – 80; SMTP – 25; telnet – 23; SSH - 22



```
thor.info.uaic.ro - PuTTY
finger      79/tcp
www         80/tcp          http      # WorldWideWeb HTTP
www         80/udp          # HyperText Transfer Protocol
: cat /etc/services
```

# Transport Level

## General primitives

- Allow to transport layer users (e.g. application programs) to access services

Primitive	TPDU	Explanation
LISTEN	(nothing)	It blocks until a process tries to connect
CONNECT	CONNECTION REQUEST	Try to establish the connection
SEND	DATA	Send information
RECEIVE	(nothing)	It blocks until it receives sent data
DISCONNECT	DISCONNECTION REQ.	Sent by the party that wants to disconnect

# TCP/IP Model

- **Application Level:**
  - Contains high level protocols
  - SMTP (*Simple Mail Transfer Protocol*) – RFC 5321 (821)
  - POP3(*Post Office Protocol*) – RFC 1081
  - TELNET – RFC 854,764
  - FTP (*File Transfer Protocol*) – RFC 454
  - NFS (*Network File System*) – RFC 1095
  - DNS (*Domain Name System*) – RFC 1034,1035
  - HTTP (*HyperText Transfer Protocol*) – RFC 2616
  - RTP (*Real-time Transport Protocol*) – RFC 3550 (1889)
  - SIP (*Session Initiation Protocol*) – RFC 3261
  - ...etc

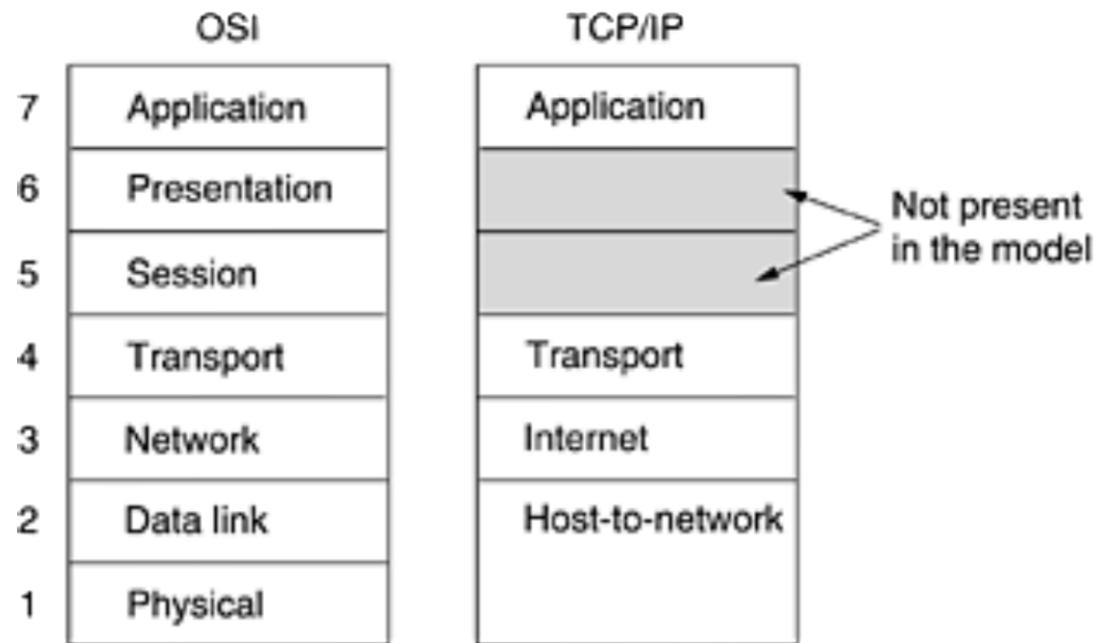
# TCP/IP Model

- Organizations involved in standardization:
  - ISOC – *Internet Society*
  - IAB – *Internet Architecture Board*
  - IETF – *Internet Engineering Task Force*
  - IRTF – *Internet Research Task Force*
  - InterNIC – *Internet Network Information Center*
  - IANA – *Internet Assigned Number Authority*
- **RFC** (*Request For Comments*) documents
  - Edited by Network Working Group (IETF)
  - RFC 1800 (Internet Official Protocol Standards)
  - More details -> [www.ietf.org](http://www.ietf.org)

# OSI versus TCP/IP

- **Similarities:**

- Both are based on a protocol stack
- The layer functionalities are somehow similar
- Both have an application layer on top
- Are based (directly or not) on transport level

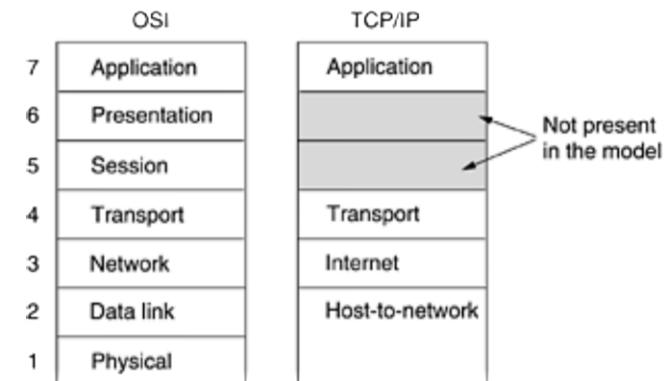


[conform Computer Networks, 2010 – Andrew S. Tanenbaum, et.al.]

# OSI versus TCP/IP

- **Differences:**

- ISO/OSI is a theoretical model; TCP/IP is effective in implementation
- OSI makes explicit the distinction between service, interface and protocol; TCP / IP does not
- ISO / OSI provides protocols that ensure reliable communication (detection and treatment of errors at each level); TCP/IP verifies communication at transport level
- OSI support both types of communication at network level (connectionless and connection oriented); TCP/IP has connectionless services at network level and both types at transport level



[conform Computer Networks, 2010 – Andrew S. Tanenbaum, et.al.]

# Summary

- Computer Networks Structure
- Network Architecture Models (OSI, TCP/IP)
- TCP/IP Model
- ISO/OSI versus TCP/IP

# Questions?

“Alexandru Ioan Cuza” University of Iasi  
Faculty of Computer Science



Course  
Master

# Distributed Systems Architectures

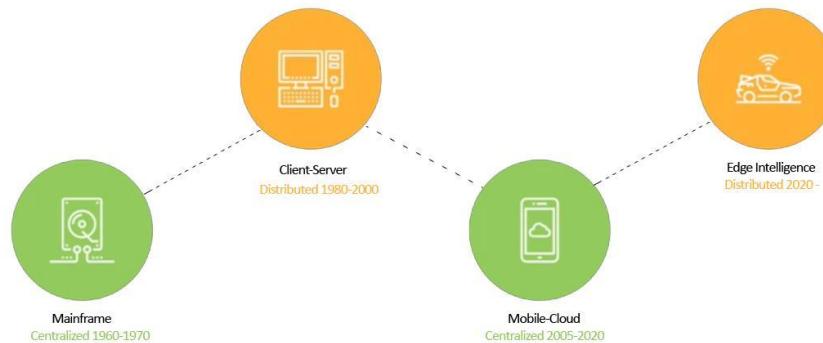
- overview-

**Lenuța Alboiae**  
**adria@info.uaic.ro**

# Summary

- **Distributed Systems Architectures**
  - Client-Server architectures
  - Peer-to-Peer (P2P) architectures
  - Cloud Computing Architectures
  - Emerging Trends
    - ...

Back to the Future



# Vision

Back to the Future

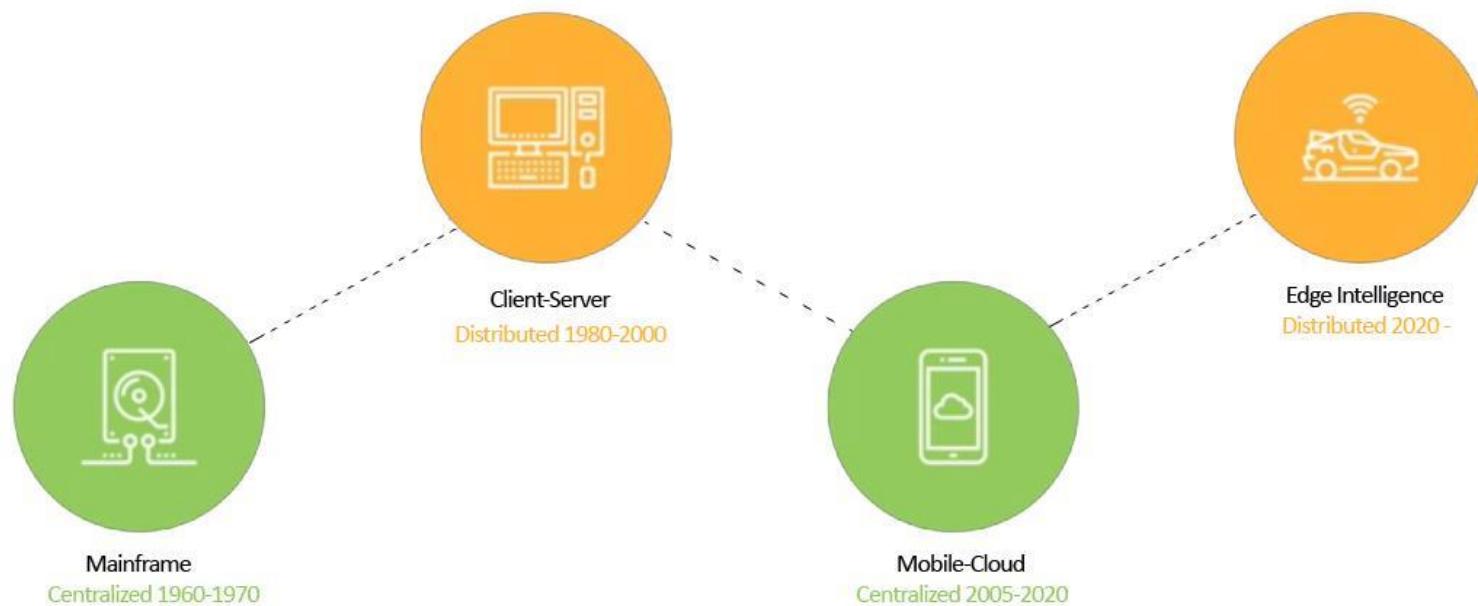


Figure. Vision over centralised and distributed architecture over time [Lev16]

# Application Architectures | Software Layers

- The term software architecture referred originally to the structuring of software as layers or modules in a single computer and recently in terms of services offered and requested between processes located in the same or different computers
- This process- and service-oriented view can be expressed in terms of service layers

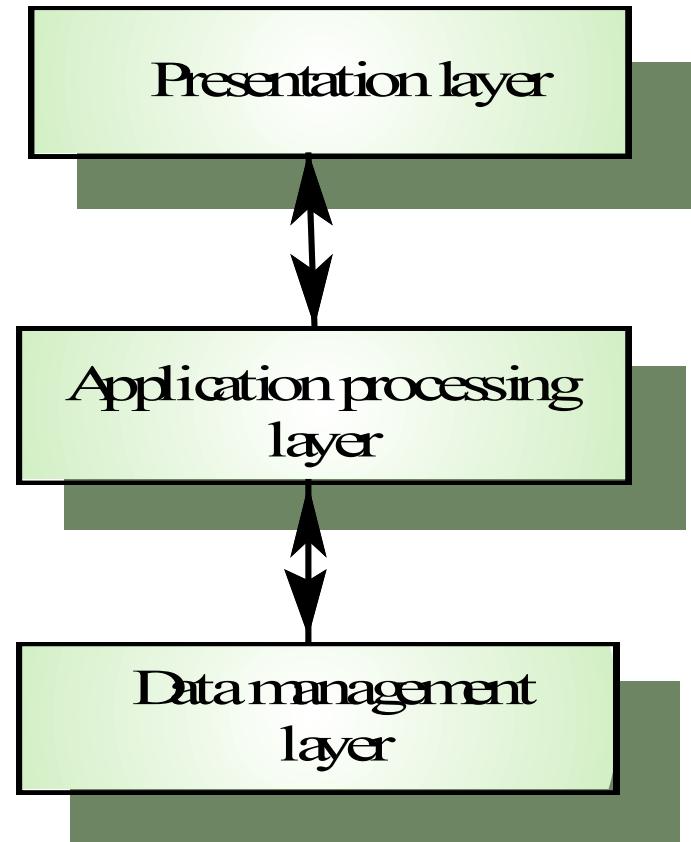
# Application Architectures

## Centralized model

- No network
- Time-sharing system
- Usually a single workstation/PC
- One or several CPUs
- Not easy scalable
- Limiting Factor: CPUs number for the same resources (memory, network, devices)

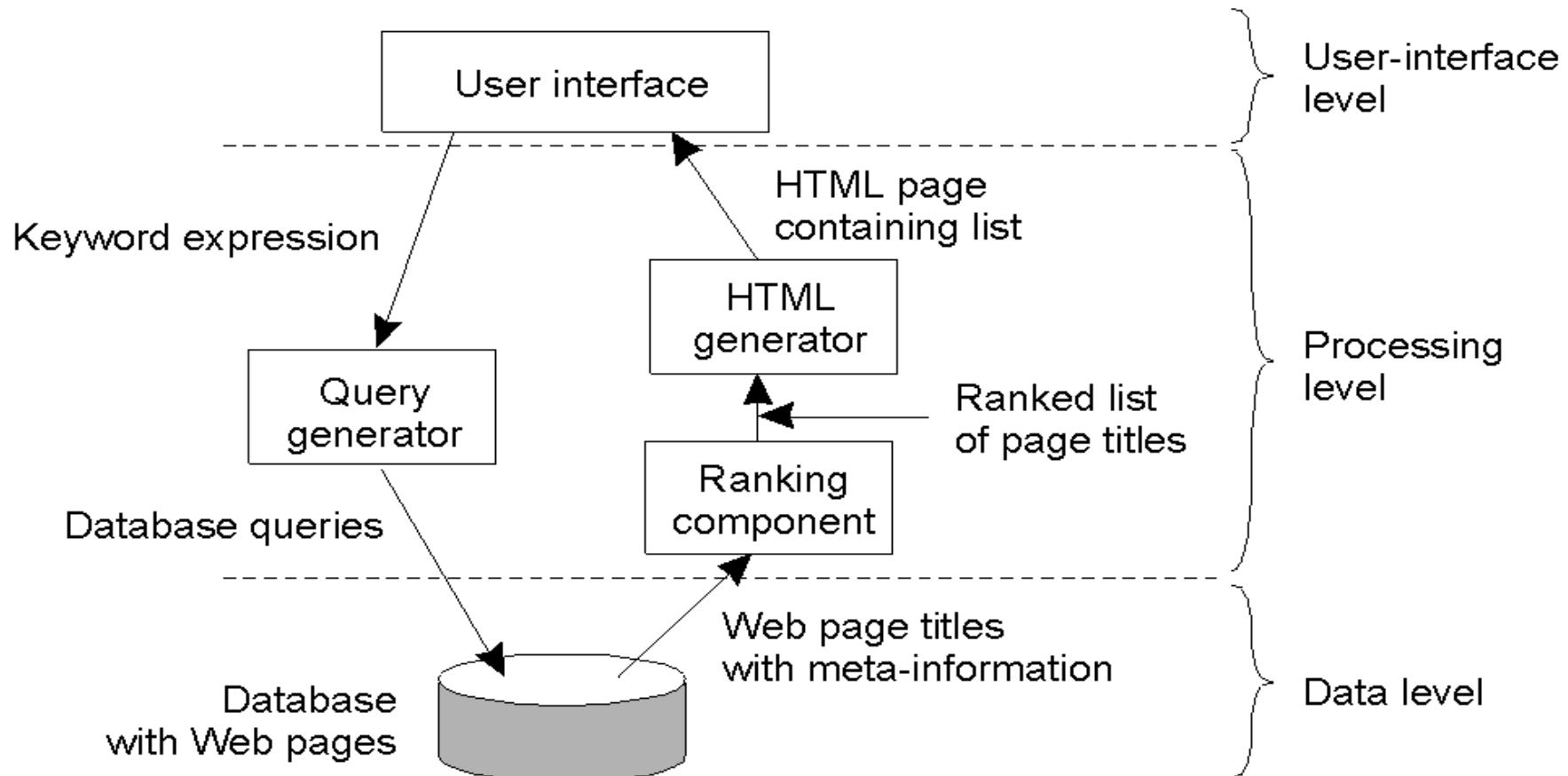
# Layered application architecture

- Presentation layer (UI)
  - Concerned with presenting the results of a computation to system users and with collecting user inputs
- Application processing layer
  - Concerned with providing application specific functionality e.g., in a banking system, banking functions such as open account, close account, etc.
- Data management layer
  - Concerned with managing the system databases



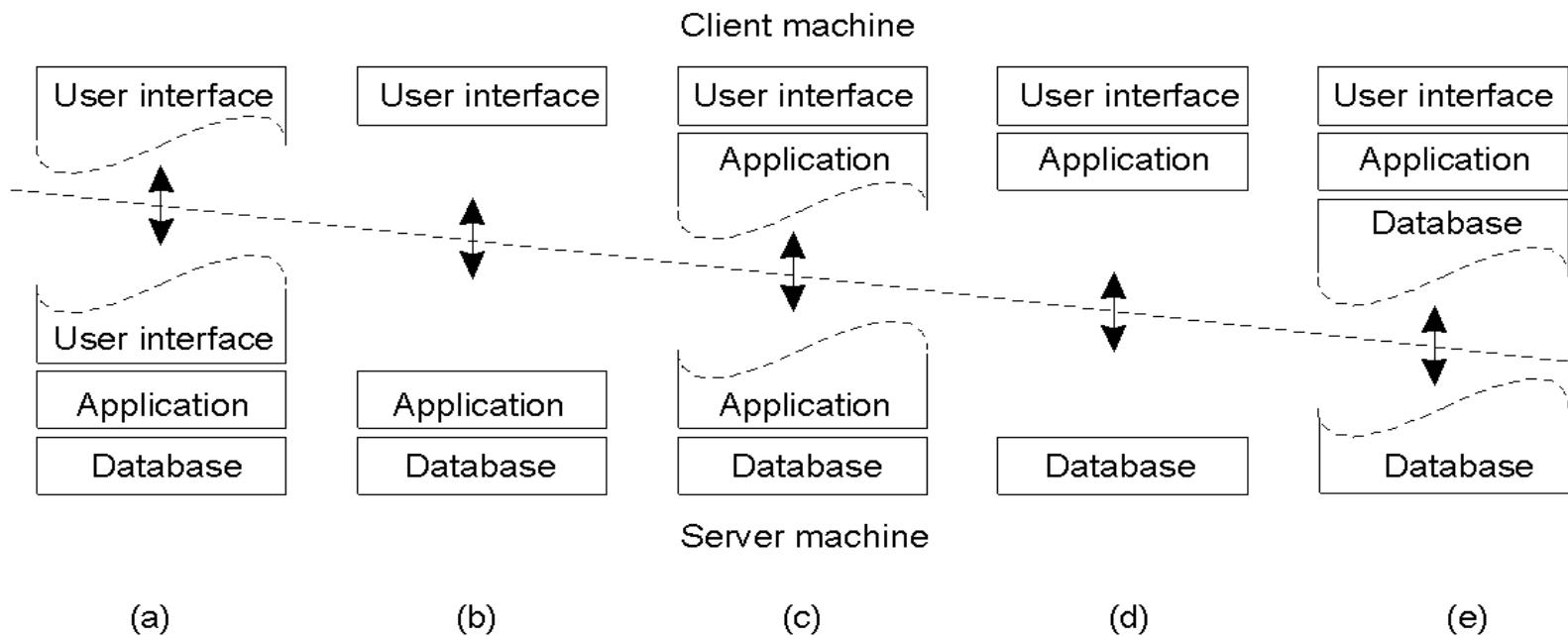
# Application Architectures | Software Layers in DS

- Tiered (multi-tier) architectures | Example: Internet Search Engine – general organization



# Application Architectures | Software Layers in DS

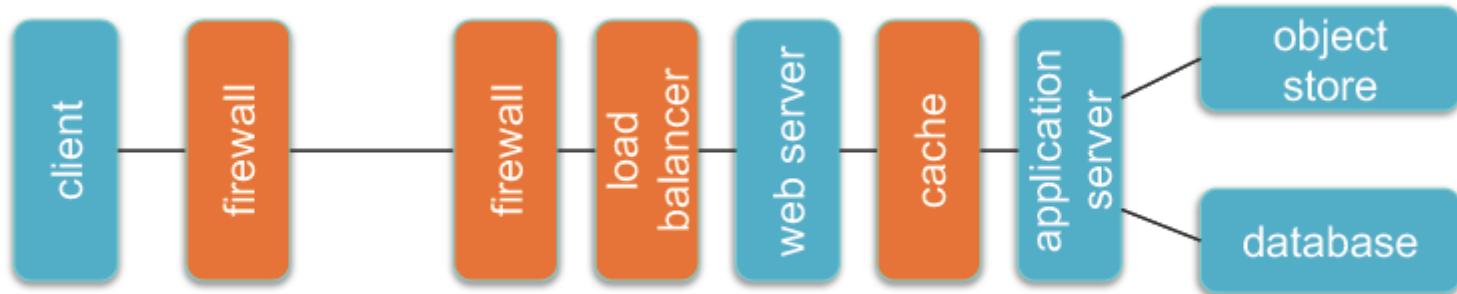
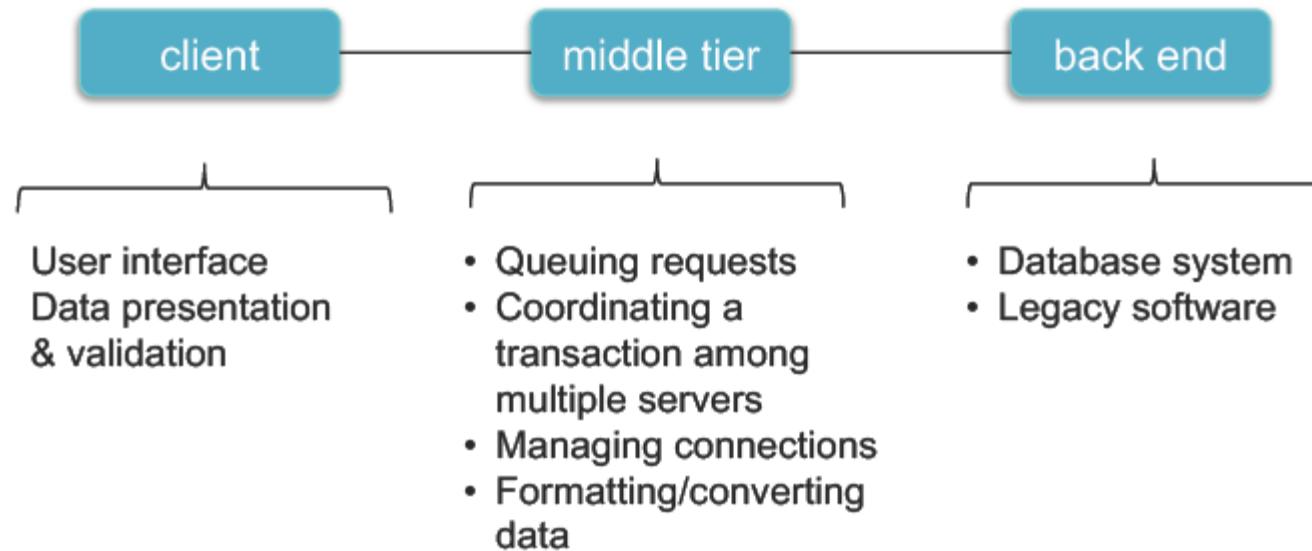
- **Tiered (multi-tier) architectures**
  - Distributed systems analogy to a layered architecture
  - Each tier(layer) runs as a network service
  - “classic” client server architecture is a two-tier model



Alternative client-server organizations (a) – (e).

# Application Architectures | Software Layers in DS

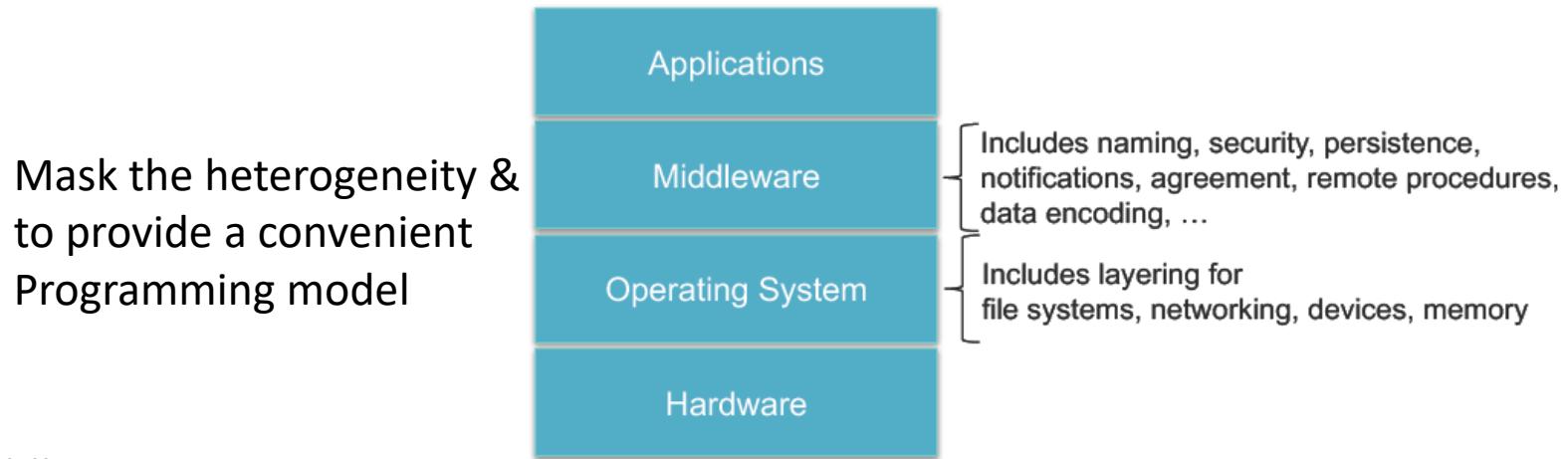
- Tiered (multi-tier) architectures | Examples



# Software Layers in DS

## Layered architectures

- Break functionalities into multiple layers
- Each layer hides implementation details, network abstractions, data encoding et.al.



- Middleware Representation
  - processes or objects interacting with each other to implement communication and resource sharing
- Middleware Function
  - provides building blocks for developing components that can interact in a DS
  - an important goal is to hide the heterogeneity of the underlying platforms from applications

# Application Architectures | Software Layers in DS

- **Middleware | Examples**
  - Earliest middlewares: Sun RPC
  - OO middlewares: CORBA (naming, security, transaction, persistent storage, event notification), DCOM, Java RMI
- **Middleware | Limitations**
  - distributed applications rely entirely on middleware services for communication and data sharing
  - assuring all dependability attributes (availability, reliability, safety, integrity, maintainability) require support at the application level
    - “End-to-end Arguments in System Design”, (Salzer et, al, 1984), ACM  
*“Some communication related functions can be completely and reliably implemented only with the knowledge and help of the application standing at the end points of the communication system. Therefore, providing that function as a feature of the communication system itself is not always sensible.”*

# **Distributed Systems Architectures**

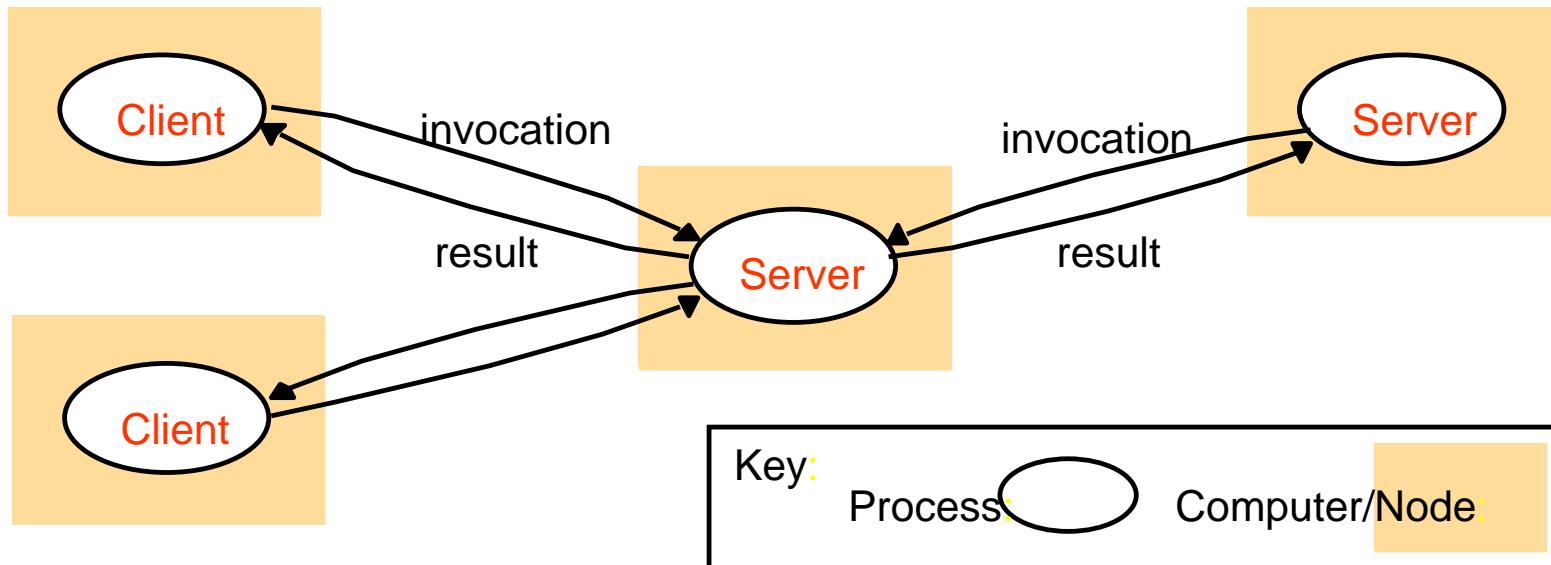
## **Client-Server architectures**

- A server is a program that provide a service.
- A client is a program that uses a resource
- The server is always on and processes requests from clients
- Clients do not communicate with other clients
- Servers may in turn be clients
- Examples: FTP, web, email

# Distributed Systems Architectures

## Client-Server architectures

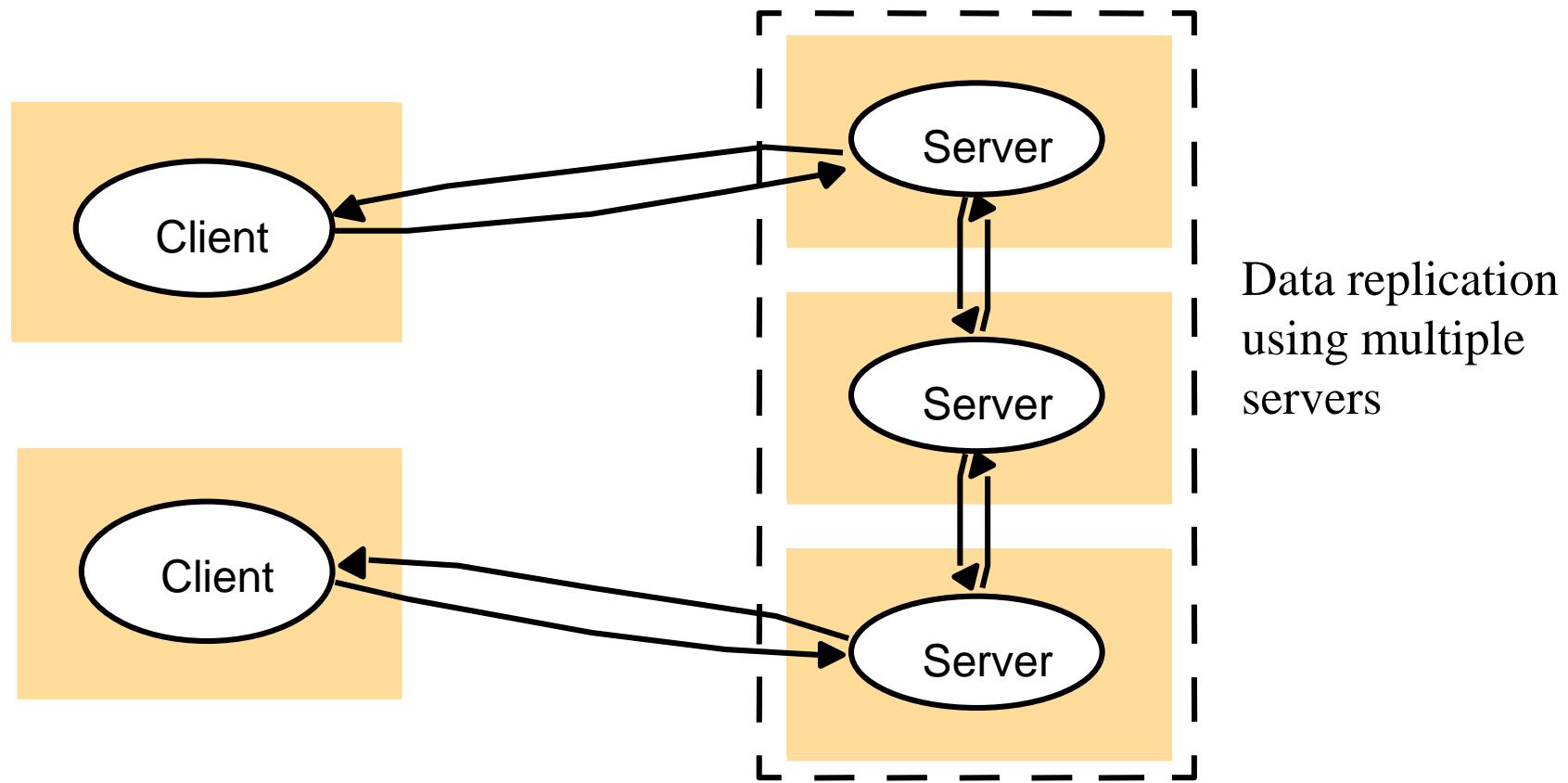
- Servers may in turn be clients



# Distributed Systems Architectures

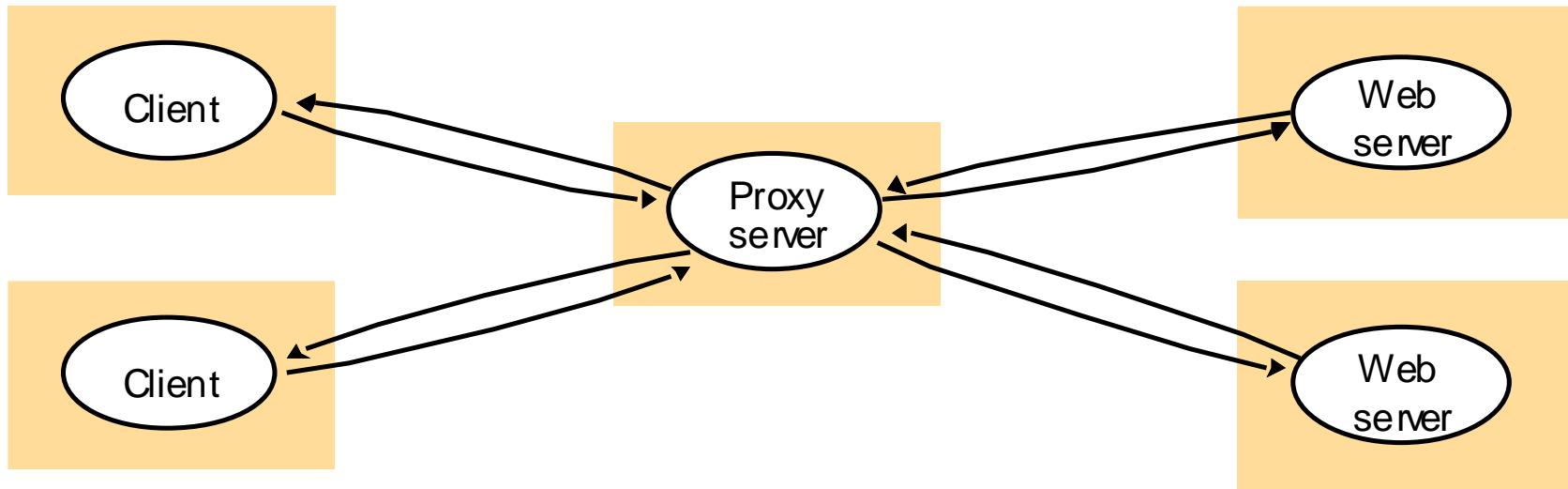
## Client-Server architectures

- A service can be provided by multiple servers



# Distributed Systems Architectures

## Client-Server architectures



### Proxy server:

- improves availability and performance (caching of data)
- Proxy servers may be used to access remote web servers through a firewall

# Distributed Systems Architectures

## Client-Server architectures – Mobile Code

- A running program that travels from one machine to other in a network carrying out task on someone's behalf, eventually returning with the result
  - It is a common practice in distributed systems to require the movement of code or processes between parts of the system, instead of data

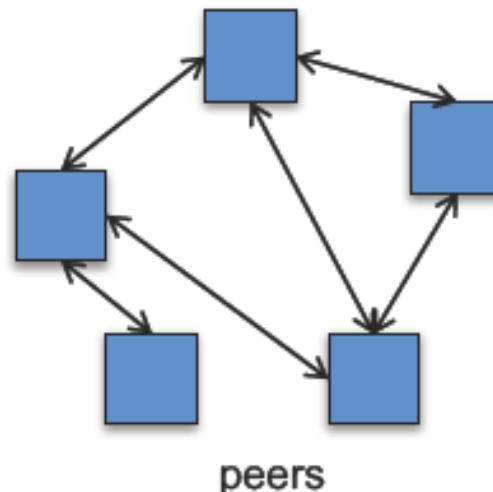
**Examples of code mobility (scripts downloaded over network):** JavaScript, Java applets, ActiveX controls, Flash animations, macros embedded in Microsoft Office

- **Architectural styles:**
  - **Remote evaluation** - A client sends code to a remote machine for execution
  - **Code on demand** - A client downloads code from a remote machine to execute locally. (e.g. Mobility-RPC)
  - **Mobile agents:** Objects or code with the ability to migrate between machines autonomously (e.g. Aglets , Mobile-C, Java Agent Development Framework)
- **A potential security threat**

# Distributed Systems Architectures

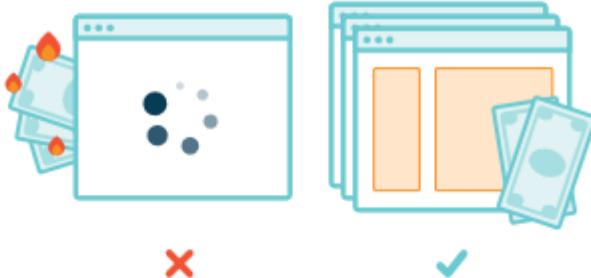
## Peer-to-Peer (P2P) Model

- Code in the peer process maintains consistency and synchronization
- No designated clients or servers
- All play similar roles interacting cooperatively as peers
- Goals examples:
  - Robustness
  - Self-scalability
- See details in Computer Network Course (2nd Year, Faculty Of Computer Science, UAIC)



# IPFS

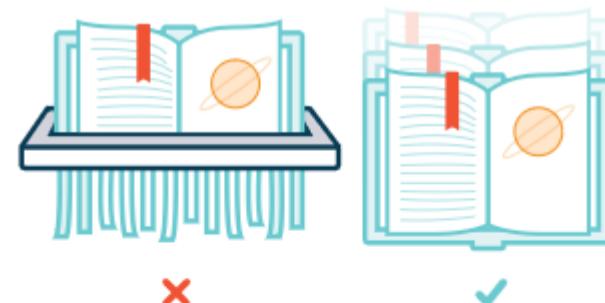
- “ IPFS is the Distributed Web” - <https://ipfs.io/>
  - A peer-to-peer hypermedia protocol to make the web faster, safer, and more open



## HTTP is inefficient and expensive

HTTP downloads a file from a single computer at a time, instead of getting pieces from multiple computers simultaneously. With video delivery, a P2P approach could save 60% in bandwidth costs.

IPFS makes it possible to distribute high volumes of data with high efficiency. And zero duplication means savings in storage.



## Humanity's history is deleted daily

The average lifespan of a web page is 100 days. Remember GeoCities? The web doesn't anymore. It's not good enough for the primary medium of our era to be so fragile.

IPFS provides historic versioning (like git) and makes it simple to set up resilient networks for mirroring of data.

# IPFS

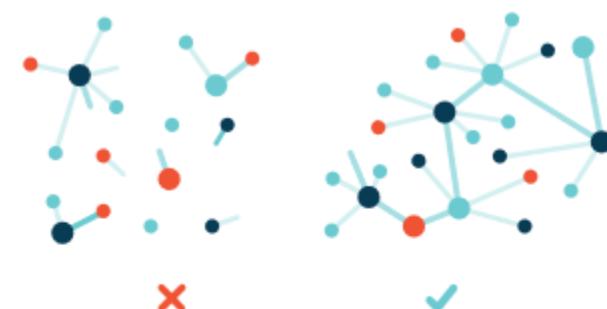
- “ IPFS is the Distributed Web” - <https://ipfs.io/>
  - A peer-to-peer hypermedia protocol to make the web faster, safer, and more open



## The web's centralization limits opportunity

The Internet has been one of the great equalizers in human history and a real accelerator of innovation. But the increasing consolidation of control is a threat to that.

IPFS remains true to the original vision of the open and flat web, but delivers the technology which makes that vision a reality.



## Our apps are addicted to the backbone

Developing world. Offline. Natural disasters. Intermittent connections. All trivial compared to interplanetary networking. The networks we're using are so 20th Century. We can do better.

IPFS powers the creation of diversely resilient networks which enable persistent availability with or without Internet backbone connectivity.

# IPFS

- “ IPFS is the Distributed Web” - <https://ipfs.io/>

Let's take a look at what happens when you add files to IPFS:



Each file and all of the **blocks within it** are given a **unique fingerprint** called a **cryptographic hash**.



Each **network node** stores only content it is interested in, and some indexing information that helps figure out who is storing what.



IPFS **removes duplications** across the network and tracks **version history** for every file.



When **looking up files**, you're asking the network to find nodes storing the content behind a unique hash.



Every file can be found by **human-readable names** using a decentralized naming system called **IPNS**.

- <https://github.com/ipfs/papers/raw/master/ipfs-cap2pfs/ipfs-p2p-file-system.pdf>

# Distributed Systems Architectures

## Cluster/Grid Computing systems architectures

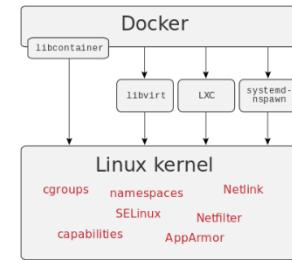
## Cloud Computing: Resources are provided as a network service

- Software as a Service (SaaS): Remotely hosted software  
*Salesforce.com, Google Apps, Microsoft Office 365*
- Infrastructure as a Service (IaaS): Compute + storage + networking  
*Microsoft Azure, Google Compute Engine, Amazon Web Services*
- Platform as a Service (PaaS): Deploy & run web applications without setting up the infrastructure  
*Google App Engine, AWS Elastic Beanstalk*
- Storage : Remote file storage  
*Dropbox, Box, Google Drive, OneDrive, ...*

# Emerging Trends | Containers

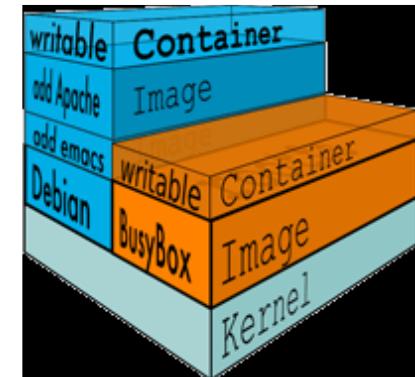
- Containers originate in the *dotCloud* project initiated on March 2013 by Solomon Hyke and colleagues
  - Fedora, Red Hat Enterprise Linux and OpenShift, adopted containers in September 2013
  - October 2014 Microsoft announced the integration of a Docker engine in Windows Server, while in November 2014 Amazon published the integration of a Docker container in EC2 (Amazon Elastic Compute Cloud)
- Docker is a tool that allows virtualization at operating system level, thus allowing the accomplishment of the easy package of an application and its dependencies in a container that can run anywhere on a server.

# Docker

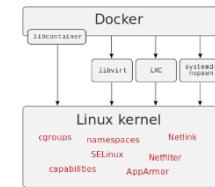


Docker containers wrap up a piece of software in a complete filesystem that contains everything it needs to run: code, runtime, system tools, system libraries – anything you can install on a server. This guarantees that it will always run the same, regardless of the environment it is running in.

*"Docker is a tool that can package an application and its dependencies in a virtual container that can run on any server. This helps enable flexibility and portability on where the application can run, whether on premise, public cloud, private cloud, bare metal, etc."*



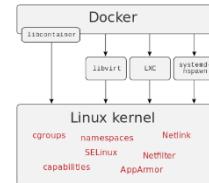
[<https://www.docker.com/what-docker>]



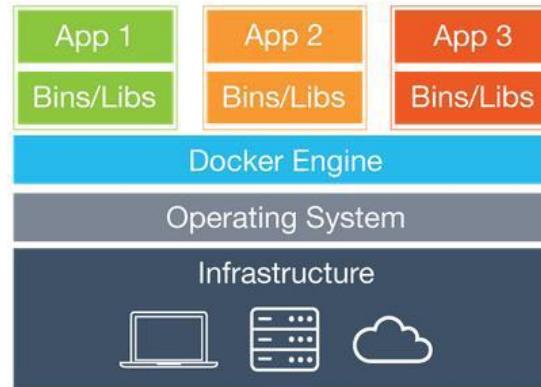
# Docker

- Implements a high level API that enables the creation of containers ensuring isolated processes running
- It is built over the facilities provided by the operating system (initially Linux kernel) => does not require a discrete operating system, as is the case of virtualization
  - => different containers are sharing the same kernel, but are limited by the available resources (CPU, memory, I/O)
- Docker also accesses various virtualization facilities through a series of libraries
- Pros: the containers' creation and management supports seamless work with distributed systems (e.g. multiple applications, distributed tasks may autonomously run on a single or a range of virtual machines)

[<http://sleekd.com/servers/docker-vs-virtualization>]<sup>24</sup>



# Docker



- Fiecare masina virtuala include aplicatiile, librariile si guest OS => zeci de GB
- *Docker containers* - includ aplicatiile si dependentele dar partajeaza kernelul
- Ruleaza la procese izolate in host OS
- Nu sunt legate de o anumita infrastructura

1

# Docker or Virtualization

So how do you go about deciding between VMs and containers anyway? Scott S. Lowe, a VMware engineering architect, suggests that you **look at the "scope" of your work**. In other words if you want run multiple copies of a single app, say MySQL, you use a container. If you want the flexibility of running multiple applications you use a virtual machine.

In addition, containers tend to lock you into a particular operating system version. That can be a good thing: You don't have to worry about dependencies once you have the application running properly in a container. But it also limits you. With VMs, no matter what hypervisor you're using -- KVM, Hyper-V, vSphere, Xen, whatever -- you can pretty much run any operating system. Do you need to run an obscure app that only runs on QNX? That's easy with a VM; it's not so simple with the current generation of containers.

[<http://www.itworld.com>]

# Docker or Virtualization

- They are used together
- Most providers are running *bare-metal virtualization* technologies(e.g. XEN) and Docker that runs over a virtualized instance (e.g. Ubuntu)
- Containers have not replaced the virtualisation mechanism and the two are often used together new services category, namely the Container as a Service (CaaS)
- <http://sleekd.com/servers/docker-vs-virtualization/>
- <http://www.serverwatch.com/server-trends/the-benefits-of-docker-vs.-server-virtualization.html>
- <http://searchservervirtualization.techtarget.com/feature/Docker-containers-virtualization-can-work-in-harmony>
- <https://tech.yandex.com/events/yac/2013/talks/14/>

# Emerging Trends | Containers

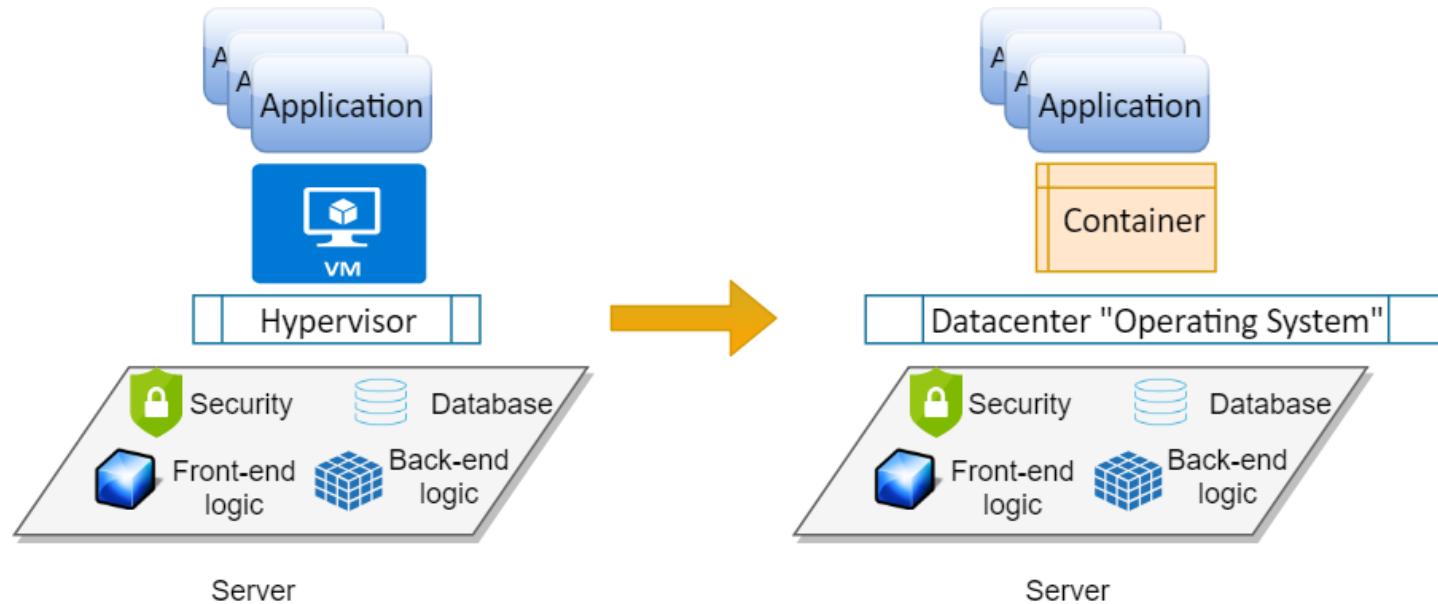


Figure: A Use Case Before and After Containers

- Virtualisation: each machine includes the applications, the libraries and the host operating system
- Docker containers include the applications and their dependencies
- The containers run as independent processes in the host operating system and are not connected by a certain infrastructure

# Emerging Trends | Containers

- The generation and management of containers contribute to easier work with distributed task workflows that may autonomously run on a single machine or on a range of virtual machines
- Containers management (“containers orchestration engines”): Kubernetes, Docker Swarm, Nomad, Mesos
  - Kubernetes allows: automating deployment, scaling, and management of applications that use containers.
  - Features: service discovery (Kubernetes assigns a DNS name and IPs for multiple containers), load balancing, storage orchestration (Kubernetes allows managing of on-premise storage, cloud storage or network storage (NFS, iSCSIm, Cinder et.al.), fail-recovery (Kubernetes restarts/replaces containers when nodes die), Batch execution (with a replacing mechanism for potential failed containers), flexible management (e.g. updating some configuration is performed without rebuilding the application image), horizontal scaling (automatically according to CPU use or manually through a simple command or UI)

# Emerging Trends | Serverless Computing

- Metaphor: “Focus on your application development, not the infrastructure”
  - PaaS ? IaaS?
- Serverless computing is an application execution model
- A serverless system runs in stateless containers that are event-triggered and the deployment represents a specific combination of client-side logic, cloud-hosted remote procedure calls (Function as a Service) and often third-party services (Backend as a Service - BaaS)
- Providers: AWS Lambda, Google Cloud Functions, Azure Functions, IBM OpenWhisk, Oracle Fn Project, Kubeless

# Emerging Trends | Serverless Computing

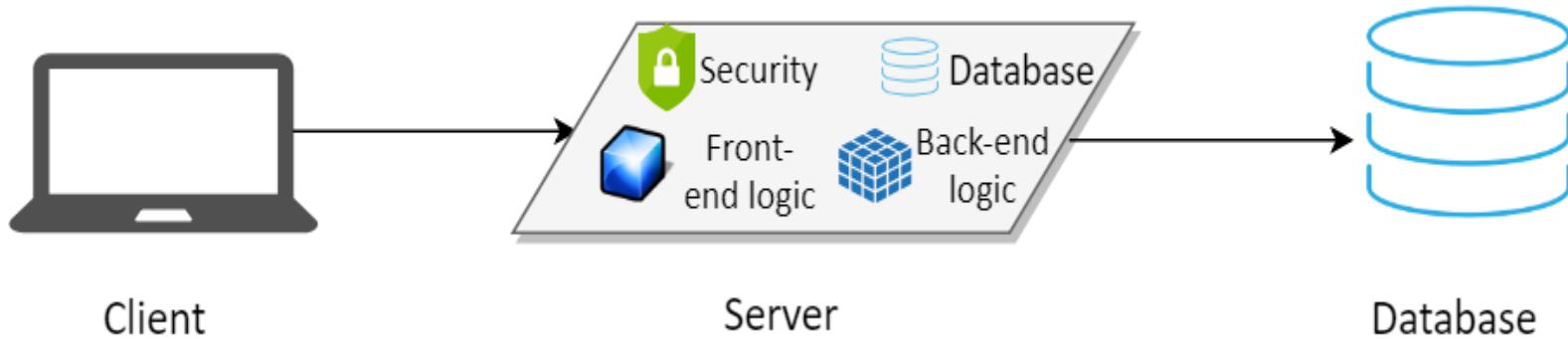


Figure. Traditional architectures

- Example: client desires to watch a film on a tablet from a provider that use the Amazon services.
- Serverless architectures
  - the client will host a part of logic - the front-end logic.
  - for the authentication service, a Backend as a Service supplier may be contracted
  - deliver the film in a database (let's say S3) in an appropriate device format, a FaaS (Function as a Service) may be called. This may be a AWS Lambda Function service that converts the movie file in a format suited to the client's device and places it in a temporary storage area in the geographical proximity of the client.

# Emerging Trends | Serverless Computing

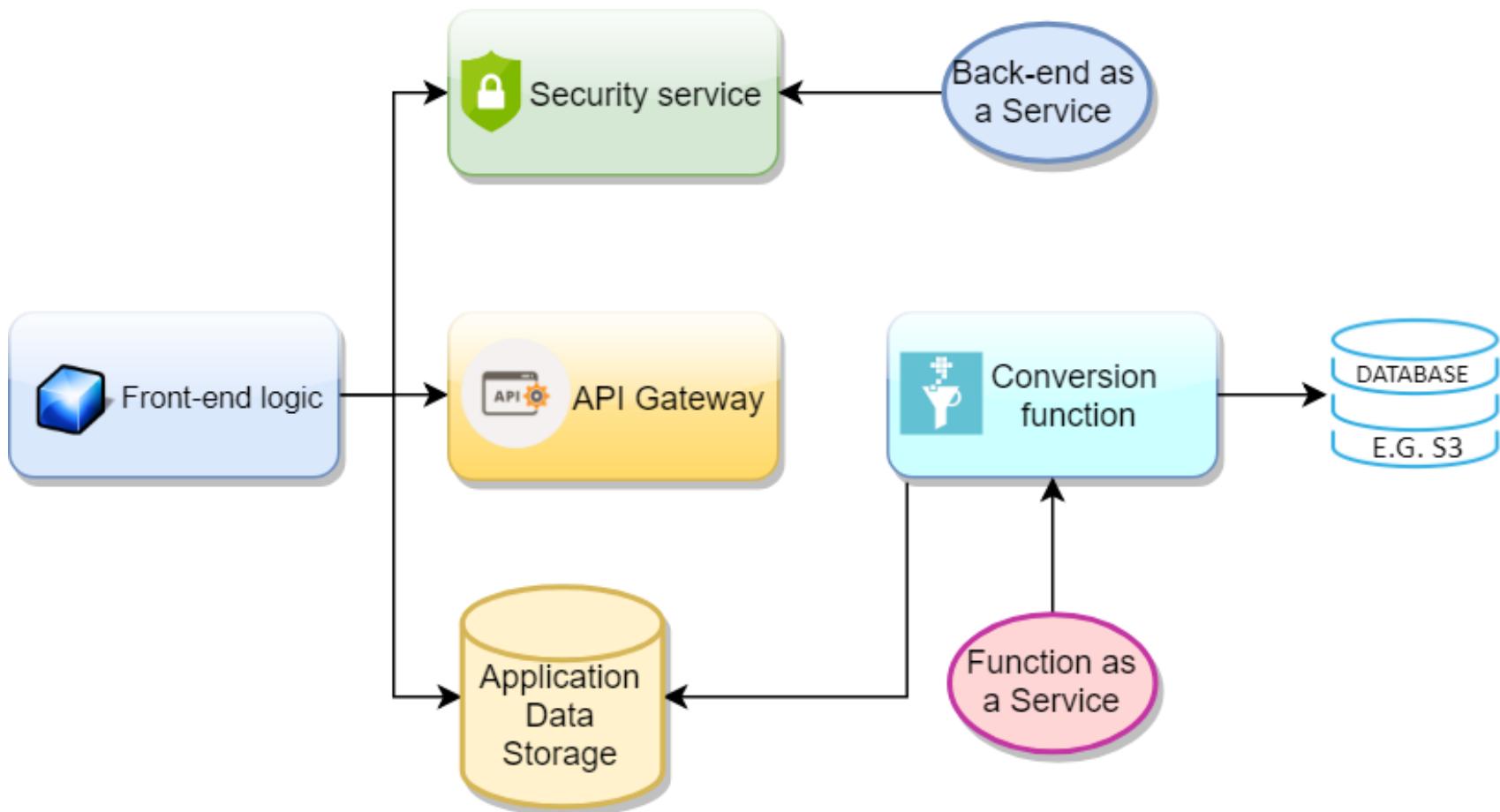
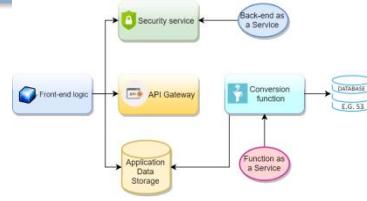


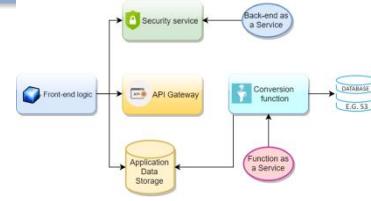
Figure: Serverless Architecture example

# Emerging Trends | Serverless Computing



- FaaS are small, separate, units of logic that have an input and return an output. They are similar with usual functions and have a set of properties:
  - Are managed by a cloud vendor: we have mentioned before that there are FaaS suppliers (AWS Lambda, Google Cloud Functions, Azure Functions) which offer a large range of environments allowing function programming in Node.js, Python, Java, .Net et.al.
  - Are event-triggered: even though they can be called directly, they can be triggered by HTTP requests or notifications
  - Are ephemeral and stateless: after a FaaS runs, it stops automatically. More than two calls of the same FaaS may run on different containers.
  - Assures scalability: due to previous characteristics, multiple containers can be initialised in order to address the current requests.

# Emerging Trends | Serverless Computing



## Pros and cons

- **Pricing:** Serverless is an execution model, meaning no provisioning or not maintaining 24 hours per day a server are needed. Instead, the charge is performed depending on the number of executions.
- **Networking :** In the case of a long-running function that exceeds the timeout limit established or which have variable execution times, the final user experience is lacking because of latency and as such, it is probably best to use traditional architectures.
- **Configuration:** For serverless architectures, it is not necessary to manage production machines or to set up various environments. Pay per execution saves your effort, therefore a serverless architecture is a better solution.
- **Third-party dependencies:** When the application has many dependencies, then a traditional architecture can be used. Otherwise, for simple applications, serverless can be employed.

Although serverless is a start-up technology, without many “best-practice” models available, we have an intuition that it may well represent an important step in the history of conceiving distributed systems architectures.

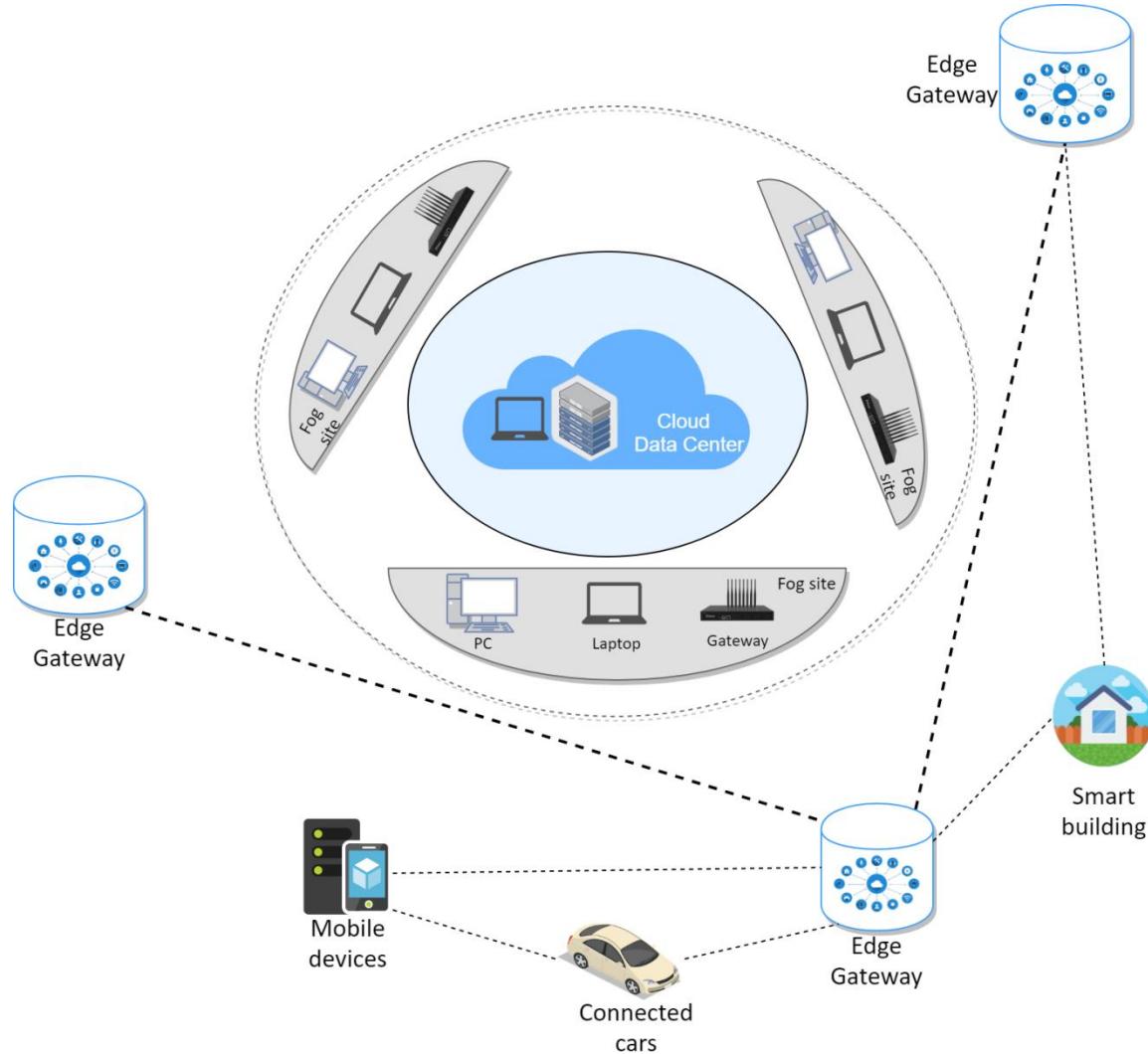
# Emerging Trends | Fog Computing and Edge Computing

- Fog Computing and Edge Computing appeared to address the need for optimisation of Cloud Computing systems by processing at the “edge of the network”, as close as possible to the data source.
- “edge” and “fog” do not stand for a new infrastructure, but rather an idea of conducting local processing using the infrastructure present as near as possible to the data source
- Edge Computing the actual processing is made by the IoT devices
  - Edge computing is a novel trend, and although there are currently no standards for complete integration between IoT devices and API, the idea not to transmit massive amounts of data (which is an expensive and time consumer process)
  - Examples of Edge computing may be observed on IoT level (e.g. mobile devices, traffic lights, motor vehicles or home appliances) or on Industrial Internet of Things level (e.g. automated industrial machines or smart power grid technology)
- Fog Computing, computing resources are placed near data sources

# Emerging Trends | Fog Computing and Edge Computing

- Fog Computing, computing resources are placed near data sources
- Fog Computing plays a bridge role among edge devices and cloud data centers
- According to [FvE16] “The key difference between the two architectures is exactly where that intelligence and computing power is placed”, obviously as close as possible to the data sources:
  - “Fog computing pushes intelligence down to the local area network level of network architecture, processing data in a fog node or IoT gateway.”
  - “Edge computing pushes the intelligence, processing power and communication capabilities of an edge gateway or appliance directly into devices like programmable automation controllers (PACs).”

# Emerging Trends | Fog Computing and Edge Computing



- Figure: Relation: Cloud Computing- Fog Computing - Edge Computing

# Emerging Trends | Fog Computing and Edge Computing

- Number of devices at EDGE Computing level will reach billions, the number of nodes at Fog Computing level will be of millions and, of course, the number of DataCenters will be of thousands [Pub17]
- In this ecosystem, **realtime** is the main characteristic of modern application
- 5G technologies have started to play an essential role in ensuring the necessary connectivity for Edge Computing.
  - comparing 4G with 3G, the main difference represented the speed of transmission,
  - for 5G, besides the speed as high as 10Gbps, they offer low latency and network support for massive increases in data traffic.

More at FII-Orange SpringSchool:

<https://profs.info.uaic.ro/~springschool/>

# Emerging Trends | Fog Computing and Edge Computing

## School Program

	Prelegere	Lector	Interval Orar
Ziua 1	<b>Curs introductiv:</b> Introducere în arhitecturi de comunicatii mobile și IoT	Marius Iordache	10:00 – 15:00
Ziua 2	Arhitecturi de comunicatii în rețele și sisteme 5G: virtualizare, programabilitate, automatizare Rețele cognitive autoscalabile	Marius Iordache	09:00 – 13:00 14:00 – 16:00
Ziua 3	Aplicabilitate in domeniul IoT. Servicii. Cloud Computing, aplicații native de tip cloud	Cătălin Brezeanu	09:00 – 12:00 13:00 – 16:00
Ziua 4	Introducere in securitatea cibernetica aplicata in Cloud Anatomia unui atac cibernetic aplicata in Cloud	Ioan Constantin	09:00 – 12:00 13:00 – 16:00
Ziua 5	Digital Forensics Detectia amenintarilor cibernetice "Zero-Day" folosind algoritmi de Machine Learning	Ioan Constantin	09:00 – 13:00 14:00 – 16:00
Ziua 6	Hackathon: Middleware for Infrastructure Protection	Orange & FII	16:00 – 19:00
	Hackathon: Middleware for Infrastructure Protection	Orange & FII	09:00 – 12:00
	PREMIERE		13:00 – 15:00

<https://profs.info.uaic.ro/~springschool/index.php/school-program/>

# Bibliography

- Andrew S. Tanenbaum, Maarten van Steen, Distributed Systems, Principles and Paradigms, Second Edition, 2007
- Ajay D. Kshemkalyani , Mukesh Singhal , Distributed Computing - Principles, Algorithms, and Systems, © Cambridge University Press 2008
- Gray, J. and Reuter, A. Transaction Processing: Concepts and Techniques. San Mateo, CA: Morgan Kaufmann, 1993.
- [https://en.wikipedia.org/wiki/Parallel\\_computing](https://en.wikipedia.org/wiki/Parallel_computing)
- <https://en.wikipedia.org/wiki/SPMD>
- *Fuggetta, Alfonso; Gian Pietro Picco; Giovanni Vigna (1998). "Understanding Code Mobility". IEEE Transactions on Software Engineering. NJ, USA: IEEE Press Piscataway. 24 (5): 342–361. doi:10.1109/32.685258. ISSN 0098-5589. Retrieved 29 July 2009.*
- *Dr Lawrie Brown. "Mobile Code Security". Australian Defence Force Academy. Retrieved 23 April 2012.*

# Bibliography

- IIT Guwahati, Topics in Distributed Systems, 2013
- Coulouris, Dollimore, Kindberg: Distributed Systems: Concepts and Design, Pearson Education.
- Singhal and Shivaratri: Advanced Concepts in Operating Systems, Tata McGraw Hill
- [https://en.wikipedia.org/wiki/Code\\_mobility#cite\\_note-Understanding\\_Code\\_Mobility-1](https://en.wikipedia.org/wiki/Code_mobility#cite_note-Understanding_Code_Mobility-1)
- <https://howdoesinternetwork.com/2013/jitter>
- Peter Levine, The End of Cloud Computing.  
Available: <https://a16z.com/2016/12/16/the-end-of-cloud-computing/> (2016)
- [Pub17] Moving the Cloud to the Edge,.  
Available:<https://www.pubnub.com/blog/moving-the-cloud-to-the-edge-computing/> (2017)
- [FvE16] Fog Computing vs. Edge Computing: What's the Difference? Available: <https://www.automationworld.com/fog-computing-vs-edge-computing-whats-difference> (2016)

# Summary

- **Distributed Systems Architectures**
  - Client-Server architectures
  - Peer-to-Peer (P2P) architectures
  - Cloud Computing Architectures
  - Emerging Trends
    - ...



# Questions?

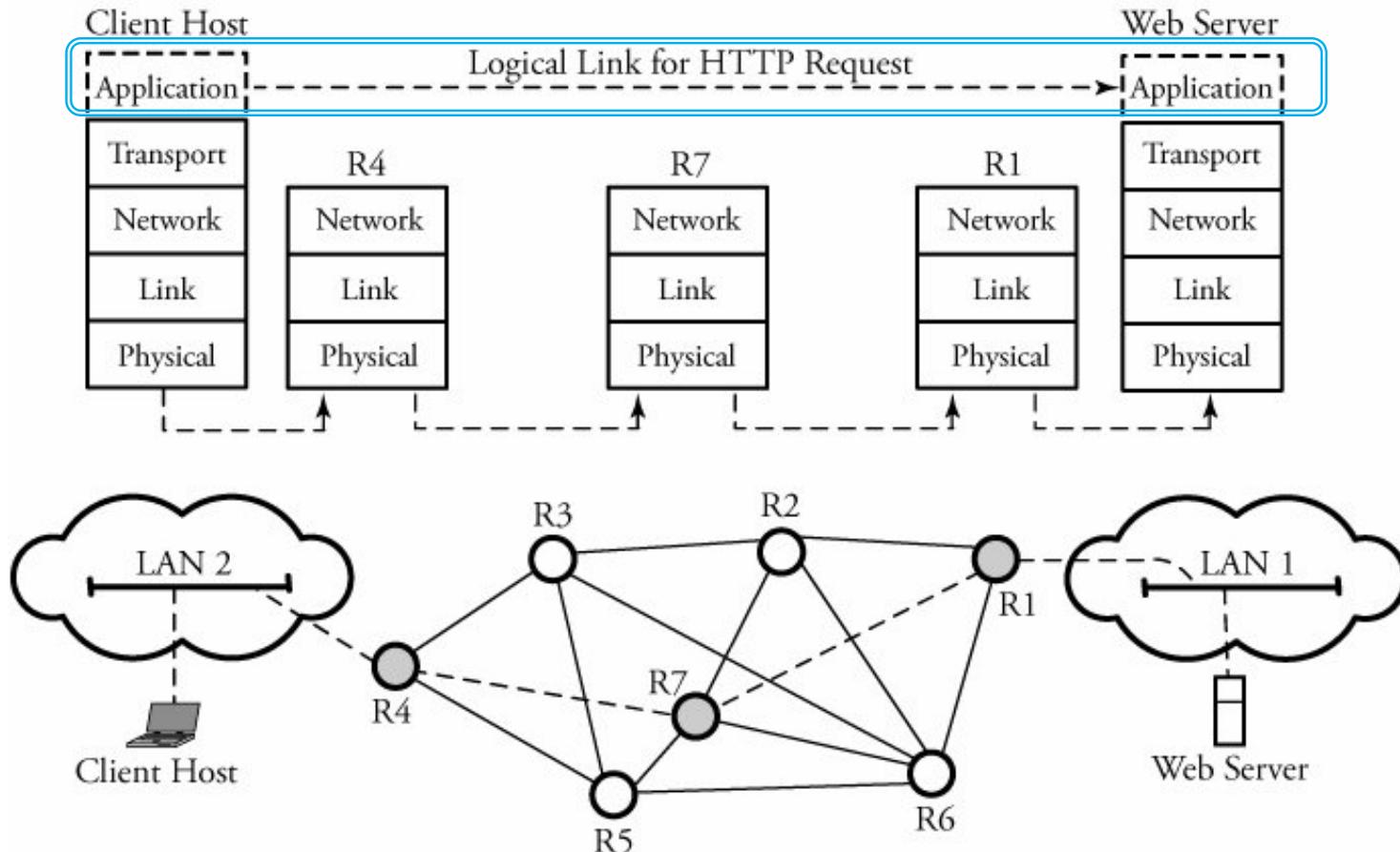
# Nivelul Aplicatie

**Lenuta Alboaie  
adria@info.uaic.ro**

# Cuprins

- **Protocole la nivelul aplicatie**
  - Preliminarii
  - Caracteristici de proiectare
  - Posta Electronica
    - SMTP (Simple Mail Transfer Protocol)
    - POP (Post Office Protocol)
  - Transferul de fisiere
    - TFTP (Trivial File Transfer Protocol)
    - FTP (File Transfer Protocol)

# Preliminarii



**Comunicare intre doua *end-systems***

[Computer and Communication Networks , Nader F. Mir, 2006]

# Preliminarii

Application	Application layer protocol	Underlying transport protocol
e-mail	SMTP [RFC 2821]	TCP
remote terminal access	Telnet [RFC 854]	TCP
Web	HTTP [RFC 2616]	TCP
file transfer	FTP [RFC 959]	TCP
streaming multimedia	HTTP (eg Youtube), RTP [RFC 1889]	TCP or UDP
Internet telephony	SIP, RTP, proprietary (e.g., Skype)	typically UDP

# Preliminarii

- **La nivelul aplicatie sunt puse la dispozitie o serie de servicii :**
  - Terminal la distanta (TELNET, SSH, ...)
  - Posta electronica (SMTP, IMAP, POP, ...)
  - Transferul de fisiere (TFTP, FTP si altele)
  - World-Wide Web (HTTP)
  - Conversatii instantanee (ICQ, XMPP (din Mai 2014 -> nu mai are suport in Google Voice), Hangouts IM, WhatsApp... )
- **Se ofera si protocoale pentru rezolvarea unor sarcini de sistem - /etc/services, /etc/protocols**
  - Sistemul de fisiere in retea (NFS)
  - Conectivitatea cu alte sisteme de fisiere (SMB)
  - Servicii de baze de date (MySQL, PostgreSQL, ..., Hive, ...)

# Caracteristici de proiectare

- Tipuri de protocole in functie de natura datelor transferate
  - Fluxuri de caractere generate de utilizator
    - Folosite pentru aplicatii interactive la distanta (*telnet*, *rlogin*, *IRC*, ...)
    - Traficul este in mare masura compus din date neinterpretate
    - Se pot include sechente de control (i.e. controlul terminalului, coduri de culoare) – coduri ANSI  
(Exemplu: **CSI** *n* **E** -> numit: CNL – Cursor Next Line  
Moves cursor to beginning of the *n*-th (default 1) following line)

# Caracteristici de proiectare

- **Tipuri de protocoale in functie de natura datelor transferate**
  - **Mesaje intrebare/raspuns ASCII**
    - Serverul si clientul vehiculeaza siruri de caractere care pot fi citite si de utilizatori umani (SMTP, FTP, HTTP/1.1, XMPP, SIP, ...)
    - Uzual, sunt compuse din linii de text
  - **Formate binare**
    - Utilizate pentru protocoale de nivel inferior (SNMP – Simple Network Management Protocol) sau de nivel inalt (NFS peste RPC, HTTP/2.0)
    - Apar probleme la reprezentarea datelor (de ex. ordinea octetilor)
  - **Protocoale ad-hoc folosite de aplicatiile (nestandard) scrise de utilizatori**
    - Pot adopta unele dintre tipurile anterioare

# Caracteristici de proiectare

- Cerinte referitoare la proiectarea unui protocol
  - Parametri critici: lungimea numelui comenzilor, marimea *buffer*-elor, modul de adresare
  - Definirea operatiilor permise (e.g., creare, citire, scriere, stergere, actualizare)
  - Raportarea erorilor: coduri de eroare, mesaje
  - Formatul mesajelor: sursa, destinatie, parametri, codificarea datelor, lungime fixa/variabila, ...

# Caracteristici de proiectare

- Scenariul uzual
  - Serverul – citeste coduri de operatii (*opcode-uri*) si raporteaza starea folosind coduri de eroare
  - Clientul – construieste mesaje folosind *opcode-urile* permise
- Moduri de adresare
  - Proces executat pe o singura masina  
adresa (fizica/logica) a masinii: thor.info.uaic.ro
  - Procese execute pe masini diferite:
    - Adrese formate din 2 parti (proces, masina)  
thor.info.uaic.ro:80
    - Adrese ca nr. generate aleatoriu (*universal ID*)
      - fiecare ID trebuie difuzat tuturor

# Caracteristici de proiectare

- Problema fiabilitatii (engl. *Reliability*) comunicarii
  - Reteaua poate pierde mesaje
  - Abordari:
    - Posta clasica (*post-office*)
      - Nu asteapta nici un fel de confirmari
    - *Handshaking* – toate mesajele sunt confirmate
    - Raspuns confirmat (*acknowledged reply*)
      - Se asteapta un raspuns, iar expeditorul raspunsului asteapta confirmarea primirii lui
    - Cerere/raspuns (*request/reply*) – expeditorul asteapta (un timp) venirea raspunsului (e.g. RPC, SOAP)

# E-mail (*Electronic Mail*)

- Protocole bazate pe TCP:
  - **SMTP** (*Simple Mail Transfer Protocol*)
    - RFC 821 (specifica modul de schimb a mail-ului intre doua host-uri)
  - **POP** (*Post Office Protocol*)
    - RFC 1939
  - **POP3S** – varianta securizata a **POP3**
    - A se vedea si: RFC 822 (specificatii privind antetul unui mail), 2049 (specificatii privind documente diferite *de plain text ASCII* ce pot fi continute intr-un email), RFC 974 (standard privind rutarea mailurilor folosind DNS)
    - RFC 822 si 974 -> consolidate in RFC 2821, 2822

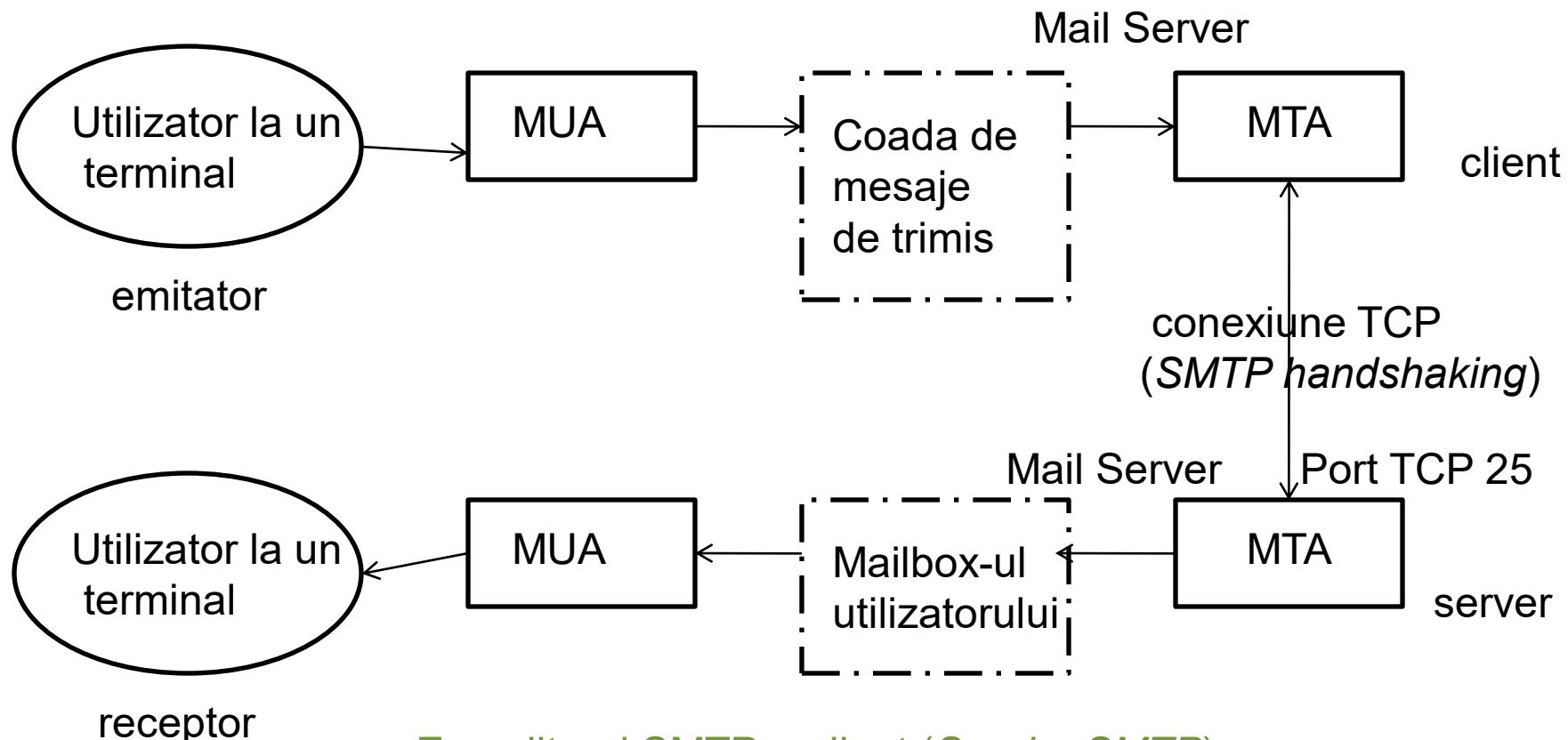
# E-mail

- **Terminologie**

- **Agent utilizator (MUA – Mail User Agent)**: client (local) pentru posta electronica  
Ex: alpine, mutt, Mozilla Thunderbird, Kmail, Outlook etc.
- **Agent de transfer (MTA – Mail Transport Agent)**  
responsabil cu comunicarea cu gazdele la distanta si cu trimiterea/receptionarea de posta  
(client & server) - sendmail, qmail
- **Agent de distributie (MDA - Mail Distribution Agent sau LDA – Local Delivery Agent)** - directioneaza mesajele primite catre casuta postala a utilizatorului; Ex: maildrop, Sieve, procmail
- **Mail exchanger (MX)** – gazda responsabila cu e-mail-urile unui domeniu (masina intermediara)

# E-mail | SMTP

- Utilizat în schimbul de mesaje de postă între serverele de mail (MTA-uri)



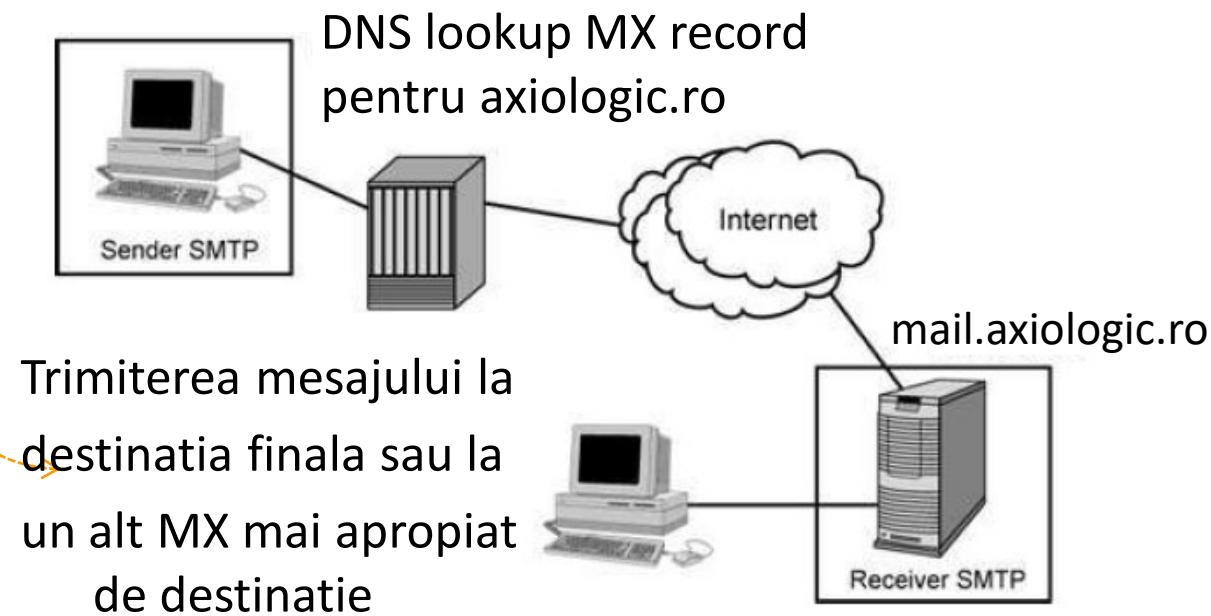
Expeditorul SMTP = client (*Sender SMTP*)

Destinatarul SMTP = server (*Receiver SMTP*)

# E-mail | SMTP

- DNS si e-mail-ul

- Inregistrarea de tip MX din DNS identifica gazda (MX) cu rol de procesare si *forwarding* a mailurilor pentru respectivul domeniu



- Mecanism general:

- Serverul SMTP verifică inregistrarea MX a domeniului specificat în adresa de email (e.g. axiologic.ro pentru adresa b@axiologic.ro) și să zicem că aceasta inregistrare este mail.axiologic.ro.
- Se va trimite acest mail pe serverul SMTP de pe mail.axiologic.ro .

# E-mail

- **Caracteristici**
  - Distinctia dintre **plic** si **continut**
    - **Plicul** incapsuleaza mesajul, contine date necesare pentru transportul mesajului: destinatar, adresa, prioritate, securitate, ...
    - Plicul este folosit pentru dirijarea mesajului la destinatar
    - **Mesajul** din plic contine un **antet** (date de control pentru MUA) si un **corp** (date pentru utilizator)
  - Fiecare utilizator este identificat printr-o adresa de e-mail:  
**cutie\_postala@locatie (cont@adresaInternet)**

# E-mail | SMTP

- Componente:
  - Plic (*envelope*) – folosit de MTA pentru livrare  
Exemplu:  
MAIL From: <adria@info.uaic.ro>  
RCPT to: <adria@info.uaic.ro>
  - Anteturi (*headers*) – folositi de MUA  
Exemplu: Received, Message-ID, From, Date, Reply-To,  
Subject,...
  - Continut –ul mesajului (*body*) -
- Mecanism: MUA preia continutul, adauga anteturi si il transmite la MTA; MTA adauga anteturi, adauga plicul si il trimite la un alt MTA

# E-mail | SMTP

- Campuri de antet utilizate in transportul de e-mail-uri:

Header	Meaning
To:	Email address(es) of primary recipient(s)
Cc:	Email address(es) of secondary recipient(s)
Bcc:	Email address(es) for blind carbon copies
From:	Person or people who created the message
Sender:	Email address of the actual sender
Received:	Line added by each transfer agent along the route
Return-Path:	Can be used to identify a path back to the sender

# E-mail | SMTP

- Campuri de antet utilizate in transportul de e-mail-uri:

Header	Meaning
Date:	The date and time the message was sent
Reply-To:	Email address to which replies should be sent
Message-Id:	Unique number for referencing this message later
In-Reply-To:	Message-Id of the message to which this is a reply
References:	Other relevant Message-Ids
Keywords:	User chosen keywords
Subject:	Short summary of the message for the one-line display

# E-mail | SMTP

## Exemplu

```
thor.info.uaic.ro - PuTTY

Un mesaj din univers:)

X-UID: 22949
X-Keywords:
X-Status:
Status: RO
To: undisclosed-recipients:;
From: anonimus@spacesome.ro
Date: Mon, 6 Dec 2010 10:48:53 +0200 (EET)
Message-Id: <20101206084910.0164CD00C@fenrir.info.uaic.ro>
    for <adria@infoiasi.ro>; Mon, 6 Dec 2010 10:48:53 +0200 (EET)
    by fenrir.info.uaic.ro (Postfix) with SMTP id 0164CD00C
Received: from fenrir.info.uaic.ro (fenrir.info.uaic.ro [127.0.0.1])
Delivered-To: adria@infoiasi.ro
    id B2F69CFB6; Mon, 6 Dec 2010 10:51:35 +0200 (EET)
Received: by fenrir.info.uaic.ro (Postfix)
    for <adria@thor.info.uaic.ro>; Mon, 6 Dec 2010 10:51:37 +0200 (EET)
    by thor.info.uaic.ro (Postfix) with ESMTP id C71932FC61
Received: from fenrir.info.uaic.ro (fenrir.info.uaic.ro [85.122.23.145])
Delivered-To: adria@thor.info.uaic.ro
X-Original-To: adria@thor.info.uaic.ro
    NO_DNS_FOR_FROM shortcircuit=no autolearn=no version=3.2.5
X-Spam-Status: No, score=3.1 required=4.5 tests=BAYES_40,MISSING SUBJECT,
X-Spam-Level: ***
X-Spam-Checker-Version: SpamAssassin 3.2.5 (2008-06-10) on thor.info.uaic.ro
X-Spam-ASN: AS12675 85.122.16.0/20
Return-Path: <anonimus@spacesome.ro>
From anonimus@spacesome.ro Mon Dec 6 10:51:37 2010
```



Anteturi  
nestandard

# E-mail | SMTP

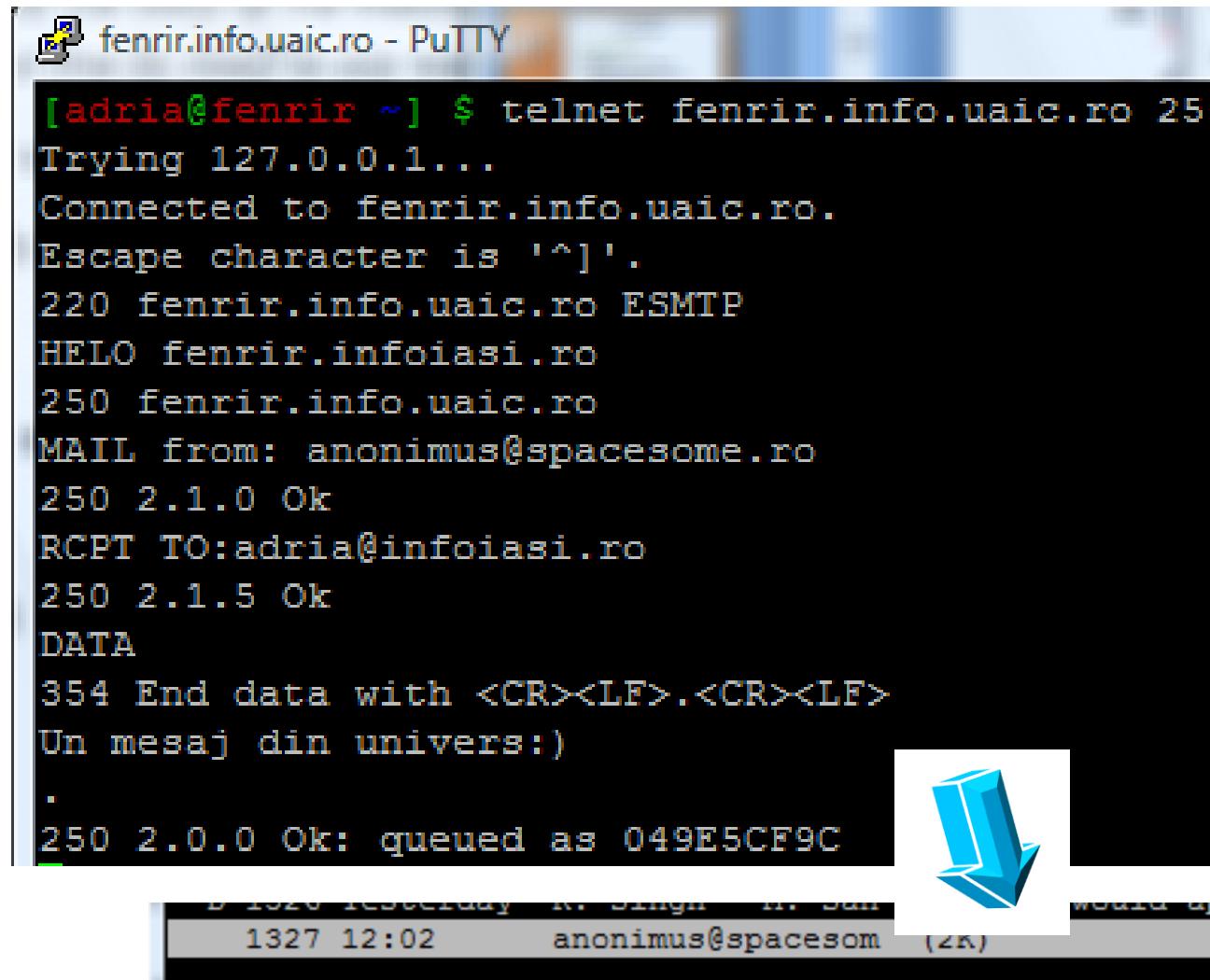
- Comunicarea:
  - Se realizeaza o conexiune TCP intre *Sender SMTP* si *Receiver SMTP* (intre MTA-uri). Obs. Receiver SMTP poate fi destinatia finala sau un intermediar (*mail gateway*)
  - Clientul trimite comenzi SMTP, iar serverul raspunde cu coduri de stare
  - Mesajele de stare include coduri numerice NNN si texte explicative
  - Ordinea comenzilor este importanta
  - Se utilizeaza portul 25

# E-mail | SMTP

- Comenzi uzuale:
  - **HELO**: identifica gazda expeditoare
  - **MAIL FROM**: porneste o tranzactie si identifica orginea e-mail-ului
  - **RCPT TO**: identifica recipientii individuali ai mesajului (adrese de e-mail); pot exista comenzi **RCPT TO**: multiple
  - **DATA** desemneaza o serie de linii text terminate cu \r\n, ultima linie continind doar “”
  - **QUIT**

# E-mail | SMTP

- Exemplu:



```
[adria@fenrir ~] $ telnet fenrir.info.uaic.ro 25
Trying 127.0.0.1...
Connected to fenrir.info.uaic.ro.
Escape character is '^]'.
220 fenrir.info.uaic.ro ESMTP
HELO fenrir.infoiasi.ro
250 fenrir.info.uaic.ro
MAIL from: anonimus@spacesome.ro
250 2.1.0 Ok
RCPT TO:adria@infoiasi.ro
250 2.1.5 Ok
DATA
354 End data with <CR><LF>.<CR><LF>
Un mesaj din univers:)
.
250 2.0.0 Ok: queued as 049E5CF9C
```



# E-mail | SMTP

- Alte comenzi:
  - **VRFY**: permite verificarea validitatii unui *recipient*
  - **NOOP**: forteaza serverul sa raspunda cu un cod de OK (200)
  - **EXPN**: expandeaza un grup de adrese (*alias*)
  - **TURN**: interschimba destinatarul cu expeditorul fara a fi necesara crearea unei noi conexiuni TCP  
(*sendmail* nu suporta aceasta comanda)
  - **RSET** abandoneaza tranzactia curenta

# E-mail | SMTP

- RFC 822: SMTP este limitat la text ASCII pe 7 biti
- RFC 1521: defineste un standard care sa rezolve limitarile anterioare -> MIME (*Multipurpose Internet Mail Extensions*)
  - Standard de codificare a continutului mesajelor non-ASCII
    - Limbi cu accente, cu alfabete non-latine, fara alfabet, mesaje non-textuale
  - Permite atasarea la e-mail a fisierelor de orice tip
  - Se foloseste campul:

**Content-Type: tip/subtip**

Exemplu: Mime-Version: 1.0

Content-Type: TEXT/PLAIN

# E-mail | SMTP

- Tipuri MIME principale:

**application** defineste aplicatiile client  
(**application/executable**)

**text** defineste formatele text  
(**text/plain**, **text/html**)

**image** specifica formatele grafice  
(**image/gif**, **image/jpeg**)

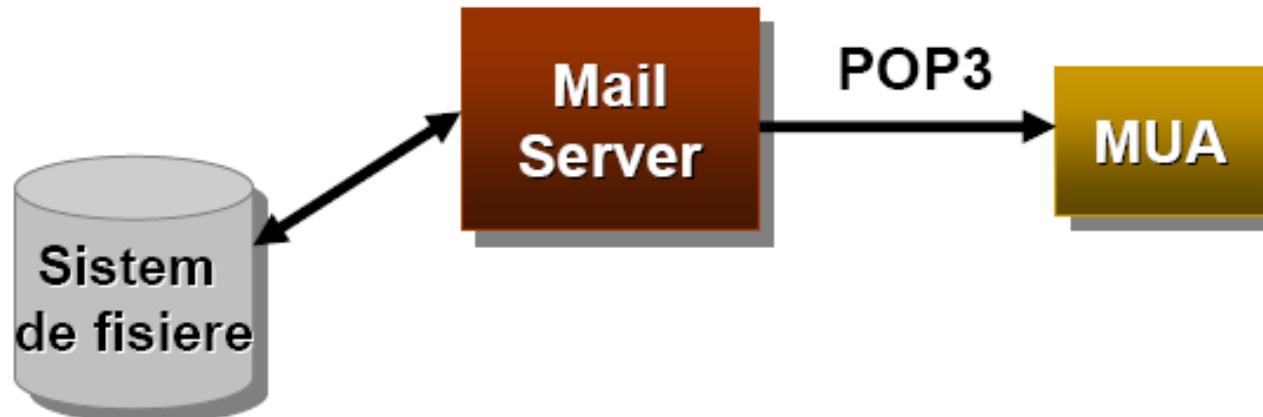
**audio** specifica formatele audio (**audio/basic**)

**video** specifica formatele video (**video/mpeg**)

**multipart** utilizat pentru transportul datelor compuse  
(**multipart/mixed**, **multipart/alternative**)

# E-mail | POP

- POP (Post Office Protocol) – RFC 1939
- Utilizat la transferul de mesaje de pe un server de posta la un MUA – portul 110
- Comenzile si raspunsurile sunt mesaje ASCII
- Raspunsurile incep cu +OK sau -ERR



[Retele de calculatoare –  
curs 2007-2008, Sabin Buraga]

# E-mail | POP

- Comenzi uzuale:
  - **USER** specifica numele de cont
  - **PASS** specifica parola
  - **STAT** furnizeaza numarul de mesaje din cutia postala (*mailbox*)
  - **LIST** afiseaza lista de mesaje si lungimea, cate 1 pe linie
  - **RETR** preia un mesaj
  - **DELE** reseteaza tranzactia, iar orice marcat de stergere este eliminat
  - **QUIT** sterge mesajele marcate si inchide conexiunea

# E-mail | POP Exemplu

```
thor.info.uaic.ro - PuTTY
adria@thor:~$ telnet thor 110
Trying 85.122.23.1...
Connected to thor.info.uaic.ro.
Escape character is '^]'.
+OK POP3 thor.info.uaic.ro 2007b.104 server ready
user adria
+OK User name accepted, password please
pass [REDACTED]
+OK Mailbox open, 1108 messages
stat
+OK 1108 194519602
retr 1108
+OK 1115 octets
Return-Path: <anonimus@spacesome.ro>
X-Spam-ASN: AS12675 85.122.16.0/20
X-Spam-Checker-Version: SpamAssassin 3.2.5 (2008-06-10) on thor.info.uaic.ro
X-Spam-Level: ***
X-Spam-Status: No, score=3.1 required=4.5 tests=BAYES_40,MISSING SUBJECT,
    NO_DNS_FOR_FROM shortcircuit=no autolearn=no version=3.2.5
X-Original-To: adria@thor.info.uaic.ro
Delivered-To: adria@thor.info.uaic.ro
Received: from fenrir.info.uaic.ro (fenrir.info.uaic.ro [85.122.23.145])
    by thor.info.uaic.ro (Postfix) with ESMTP id C71932FC61
    for <adria@thor.info.uaic.ro>; Mon, 6 Dec 2010 10:51:37 +0200 (EET)
Received: by fenrir.info.uaic.ro (Postfix)
    id B2F69C9B6; Mon, 6 Dec 2010 10:51:35 +0200 (EET)
Delivered-To: adria@infoiasi.ro
Received: from fenrir.info.uaic.ro (fenrir.info.uaic.ro [127.0.0.1])
    by fenrir.info.uaic.ro (Postfix) with SMTP id 0164CD00C
    for <adria@infoiasi.ro>; Mon, 6 Dec 2010 10:48:53 +0200 (EET)
Message-Id: <20101206084910.0164CD00C@fenrir.info.uaic.ro>
Date: Mon, 6 Dec 2010 10:48:53 +0200 (EET)
From: anonimus@spacesome.ro
To: undisclosed-recipients:;
Status: O

Un mesaj din univers:)
.
```

Parola necriptata

# E-mail | POP

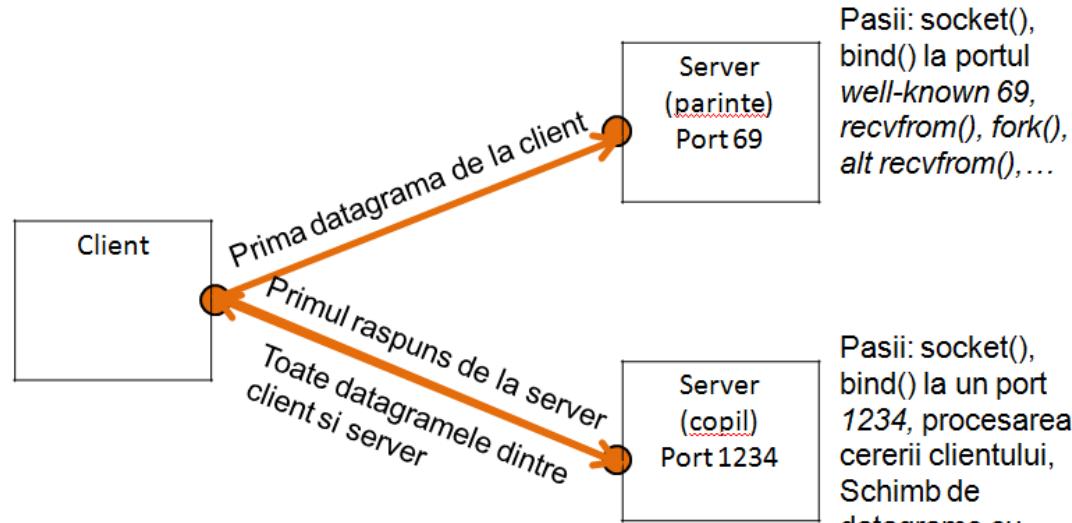
- POP 3 - caracteristici:
  - In general, daca utilizatorul schimba clientul el nu-si mai poate reciti mailurile; Obs: Clienti cu optiunea: '*keep a copy of the email on the server*'
  - Foloseste mecanismul “*download-and-keep*”: copierea mesajelor pe clienti diferiti
  - POP3 este fara stare intre sesiuni
- Alte solutii:

**IMAP (Interactive Mail Access Protocol)** – RFC 1730

- Pastreaza toate mesajele intr-un singur loc: pe server
- Permite utilizatorului sa organizeze mesajele in directoare
- Pastreaza starea “utilizatorului” intre sesiuni
  - Numele directoarelor si maparea dintre ID-urile mesajelor si numele folderului

# Transferul de fisiere | TFTP

- TFTP (Trivial File Transfer Protocol) -> ...Cursul 6 & RFC 1350



- utilizeaza UDP si portul 69
- utilizat deseori la initializarea statiilor de lucru fara disc sau a altor dispozitive
- nu are mecanisme de autentificare si criptare => este utilizat in retele locale
- RFC 1785, 2347, 2348, 2349

# Transferul de fisiere | TFTP

- TFTP (Trivial File Transfer Protocol)

Implementarile TFTP utilizeaza comenzi de tipul:

**Connect <host>** Specifies the destination host ID.

**Mode <ascii|binary>** Specifies the type of transfer mode.

**Get <remote filename> [<local filename>]**  
Retrieves a file.

**Put <remote filename> [<local filename>]**  
Stores a file.

**Verbose** Toggles verbose mode, which displays additional information during file transfer, on or off.

**Quit** Exits TFTP.

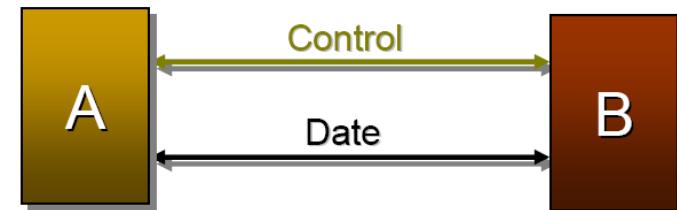
- RFC 1785, 2347, 2348, 2349

[ TCP/IP Tutorial and Technical Overview, IBM, 2006 ]

# Transferul de fisiere | FTP

## FTP – caracterizare

- Folosit atat interactiv, cat si de programe
- Asigura transferul sigur si eficient al fisierelor
- Se bazeaza pe modelul client/server
- FTP utilizeaza doua conexiuni TCP pentru transferul fisierelor:
  - **Conexiune de control**
    - folosita pentru trimitera comenzilor si receptionarea codurilor de stare
    - Conexiunea de control utilizeaza portul 21
  - **Conexiunea de date**
    - folosita pentru transferul efectiv
    - conexiunea de date foloseste portul 20 sau unul aleator ( $P > 1023$ )
    - nu este obligatorie intr-o sesiune FTP



# Transferul de fisiere | FTP

## FTP – caracterizare

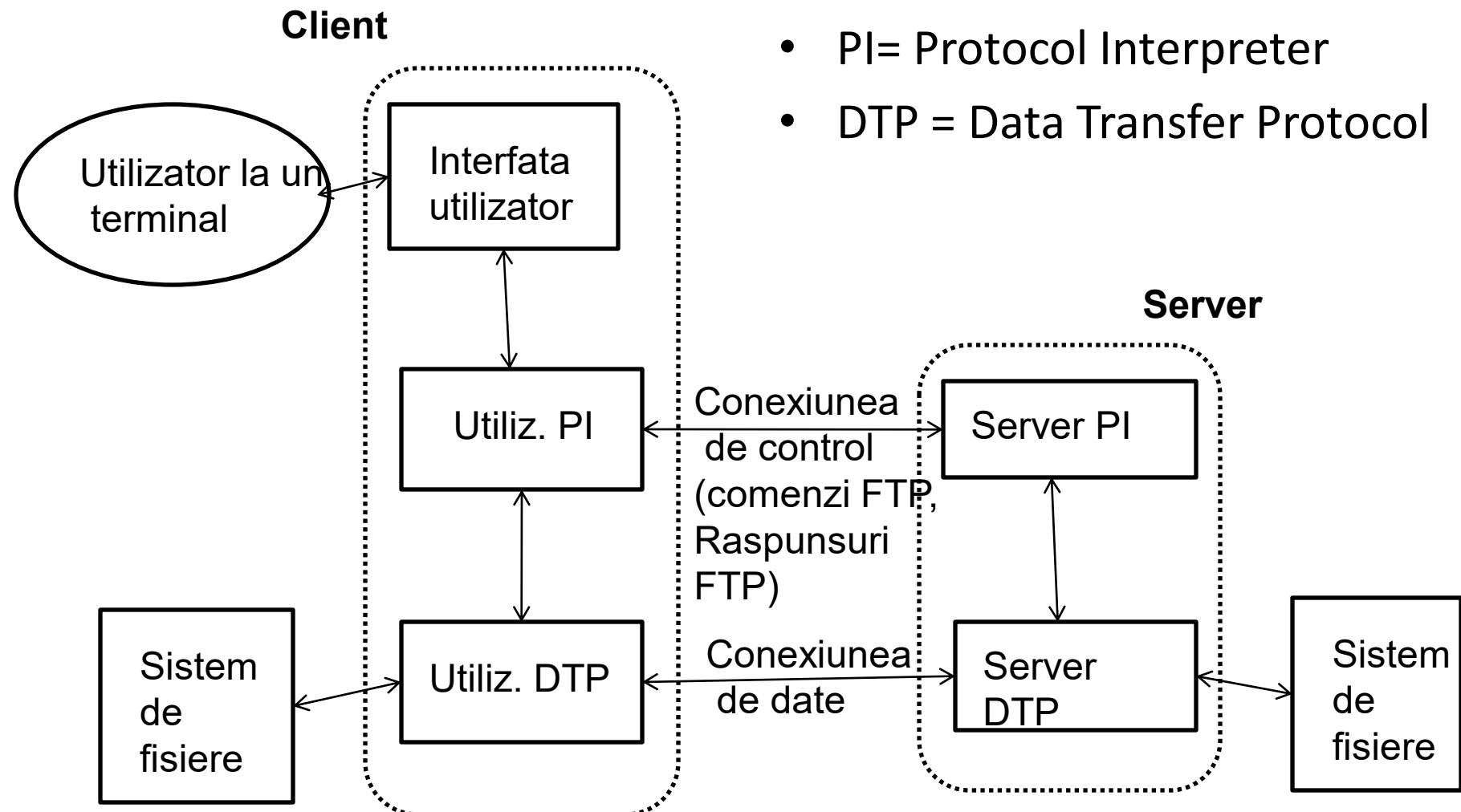
- Comenzile si raspunsurile sunt linii de text
- Obs. (FTP->)file transfer != file access (->NFS)
- Vezi si RFC 956, 1068, 2228, (FTP Security Extensions), 2428 (FTP Extensions for IPv6 and NATs)
- Pentru interactivitate se foloseste protocolul TELNET

Tipuri de acces:

- Anonim (FTP anonymous) – RFC 1635
  - Autentificare cu numele *anonymous* si drept parola o adresa de e-mail
  - Acces public la o serie de resurse (aplicatii, date, multimedia etc.)
- Autentificat
  - Necesa un nume de utilizator existent, insotit de o parola valida
  - Pentru transferul de date in/din contul personal

# Transferul de fisiere | FTP

## FTP- model



# Transferul de fisiere | FTP

## FTP – comenzi (client)

```
adria@thor:~$ ftp
ftp> help
Commands may be abbreviated.  Commands are:

!
$          debug      mdir      qc        send
account    dir        mget      sendport   site
append    disconnect mkdir     put        size
ascii      form       mls       pwd        status
bell       get        mode      quit       struct
binary    glob       mput      quote      system
bye        hash       newer     recv       sunique
case      help       nmap      reget      tenex
cd         idle       nlist     rstatus    tick
cdup     image       ntrans    rhelp      trace
chmod    lcd        open      rename    type
close    ls         prompt   reset     user
cr       macdef    passive  restart  umask
delete  mdelete   proxy    rmdir    verbose
ftp> [green bar]
```

# Transferul de fisiere | FTP

## FTP – comenzi uzuale (client)

Command	Meaning	
Open	Create an FTP connection between the two hosts.	
Close	Close an FTP connection between two hosts.	
Bye	End the FTP session.	
Get	Retrieve a remote file from the remote host.	→ <b>RETR (retrieve)</b>
Put	Store a file on the remote host.	→ <b>STOR (store)</b>
Mget	Get multiple files using wildcards (for example, mget a* fetches all files that begin with the letter "a" in the current directory).	
Mput	Put multiple files on the remote host using wildcards.	
Glob	Enable wildcard interpretation. This is usually on by default.	
Ascii	The file transferred is in ASCII representation (a common default).	
Binary	The file is in image (binary) format (sometimes the default), and is useful for programs and formatted word processing files.	
Cd	Change the directory on the remote host.	
Dir	Get a directory listing from the remote host.	
Ldir	Get a directory listing from the local host.	
Hash	Display hash marks (dots) to show file transfer progress.	

# Transferul de fisiere | FTP

## FTP – comenzi (protocol)

- Comenzi de control al accesului
  - USER *username*, PASS *password*, QUIT, ChangeWorkingDir,...
- Comenzi de transfer a parametrilor
  - PORT, TYPE, MODE
- Comenzi de realizarea a serviciilor FTP
  - RETR *filename*, ABOR, STOR *filename*, LIST, PrintWorkingDir

## Raspunsul de stare

Linie de text continind:

XYZ un cod de stare (utilizat de software) + un mesaj explicativ (destinat oamenilor)

# Transferul de fisiere | FTP

## FTP – codul de stare (xyz)

Prima cifra semnifica:

- 1 replica pozitiva preliminara (“am indeplinit, dar asteapta”)
- 2 replica pozitiva finala (“succes”)
- 3 replica pozitiva intermediara (“am nevoie si de alte informatii”)
- 4 replica negativa tranzitorie (“eroare, incerc iar”)
- 5 replica negativa finala (“eroare fatala”)

# Transferul de fisiere | FTP

## FTP – codul de stare (xyz)

A doua cifra specifica grupuri de functii:

0 erori de sintaxa

1 informare (ajutor, informatii de stare)

2 referitor la conexiuni

3 privitor la autentificarea utilizatorului

4 nespecificat

5 referitor la sistemul de fisiere

# Transferul de fisiere | FTP

## FTP – codul de stare (xyz)

A treia cifra da informatii suplimentare asupra mesajelor de eroare

Exemple:

125 Conexiune deschisa; transfer pornit

200 Comanda OK

226 Transfer complet

331 Nume utilizator OK, se cere parola

452 Eroare la scrierea fisierului

500 Eroare de sintaxa (comanda necunoscuta)

501 Eroare sintaxa (argumente invalide)

221 Goodbye /\*rezultat al comenzii QUIT \*/

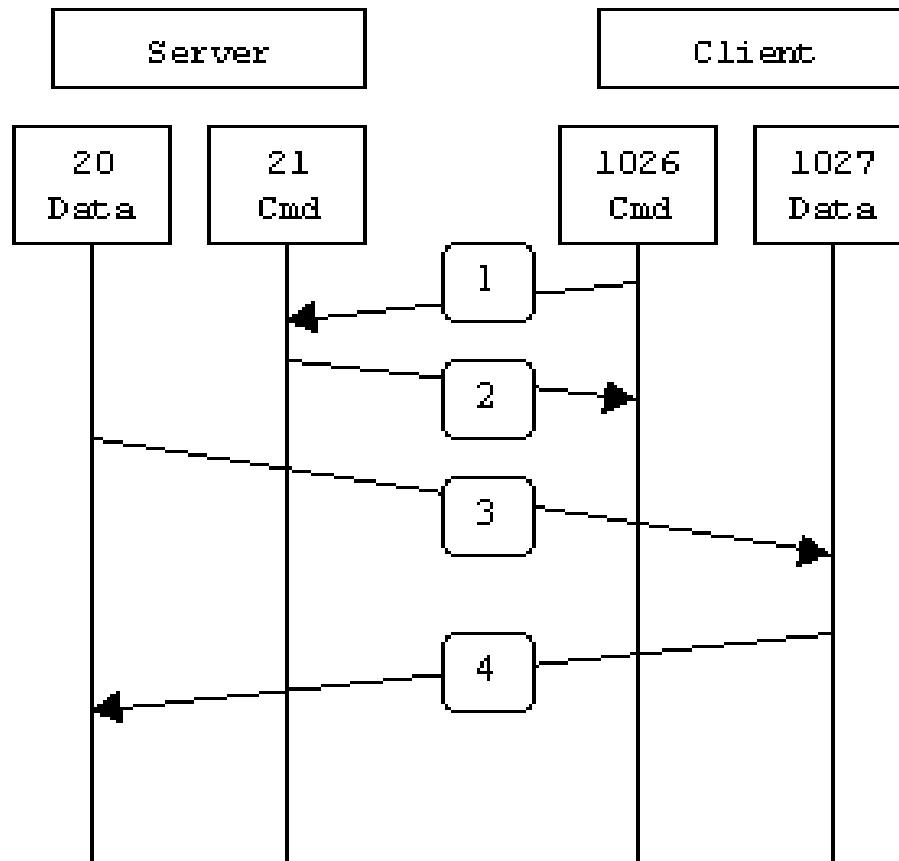
# Transferul de fisiere | FTP

## FTP – Moduri de transfer

- **STREAM**
  - Fisierul este trimis ca un flux de octeti; sfîrșitul transmisiei este indicat de inchiderea normală a conexiunii;
- **BLOCK**
  - Fisierul este transmis ca o serie de blocuri de date precedate de antete continand conțoare și descriptori de bloc (e.g. End of data block)
- **COMPRESSED**
  - Fisierele sunt compresate, conform unui algoritm de compresare (e.g., gzip) și sunt trimise ca date binare

# Transferul de fisiere | FTP

## Active FTP – exemplu

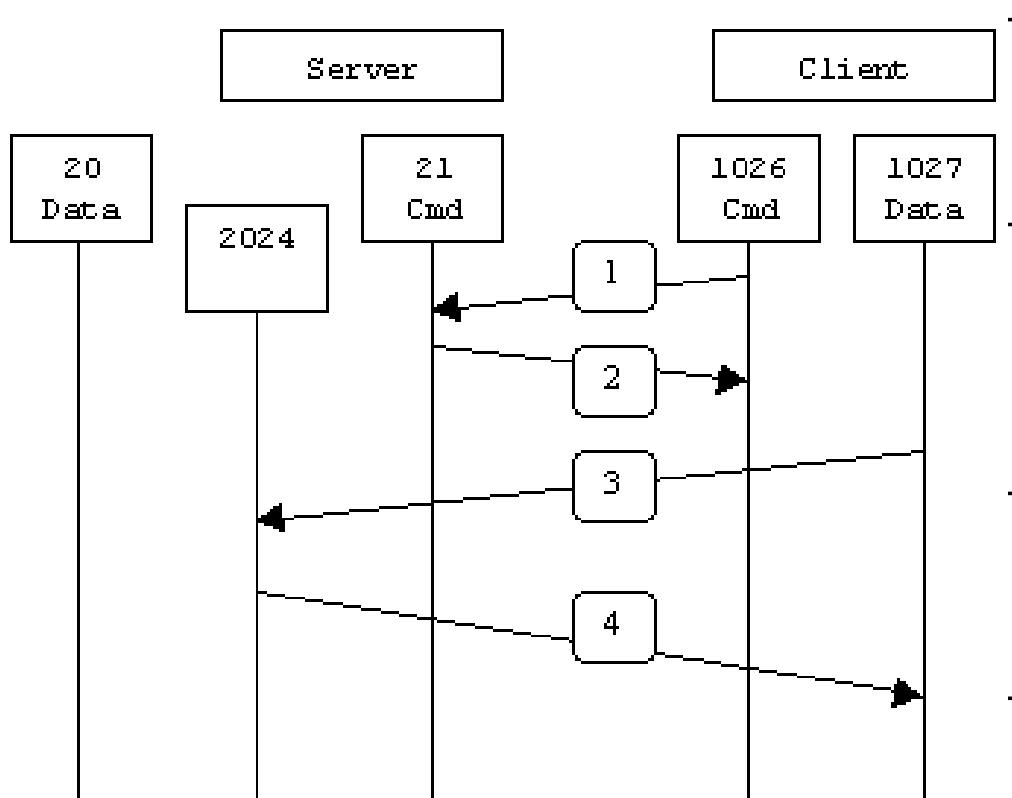


- Clientul se conecteaza la un server (85.122.23.145:**21**) de la un port P > 1023
- Clientul trimite comanda PORT 85.122.23.1.**4.2** ( $4 * 256 + 2 = \textbf{1026}$ ) ce indica Server-ului sa initieze o conexiune cu clientul la portul P+1
- Clientul asculta la P+1 si primeste datele trimise de server prin portul **20**

Obs. Conexiunea initiată de server poate fi interpretată ca un potential atac de firewall-ul clientului

# Transferul de fisiere | FTP

## Pasive FTP – exemplu



- La initierea unei conexiuni FTP clientul foloseste doua porturi ( $P > 1023$  si  $P+1$ )
- Clientul se conecteaza la un server (85.122.23.145:21) de la portul  $P$  si trimite comanda PASV
- Serverul deschide un port  $Ps > 1023$  si trimite comanda PORT  $Ps$  clientului
- Clientul va initia o conexiune (de la portul  $P+1$ ) cu serverul folosind portul primit ( $Ps$ )

# Rezumat

- **Protocole la nivelul aplicatie**
  - Preliminarii
  - Caracteristici de proiectare
  - Posta Electronica
    - SMTP (Simple Mail Transfer Protocol)
    - POP (Post Office Protocol)
  - Transferul de fisiere
    - TFTP (Trivial File Transfer Protocol)
    - FTP (File Transfer Protocol)

# Bibliografie

**Content Networking Fundamentals**, Silvano Da Ros, Publisher: Cisco Press Pub Date: March 30, 2006 Print ISBN-10: 1-58705-240-7 Print ISBN-13: 978-1-58705-240-8 Pages: 576

**Computer and Communication Networks**, Nader F. Mir, Publisher: Prentice Hall Pub Date: November 02, 2006 Print ISBN-10: 0-13-174799-1 Print ISBN-13: 978-0-13-174799-9 Pages: 656

**TCP/IP Tutorial and Technical Overview, IBM, 2006**

**Network + Guide to Networks**, Tamara Dean, 2009



# Intrebari?

# Network Programming

**Lenuta Alboiae**  
**adria@info.uaic.ro**

# Content

- Transport Level
  - Preliminary
  - UDP (User Datagram Protocol)
  - TCP (Transmission Control Protocol)
  - TCP versus UDP
- Network Communication paradigms
  - Client/Server model (TCP, UDP)
  - API for network programming
  - BSD *Socket*
    - Characteristics
    - Primitives
  - TCP client/server model
  - UDP client/server model

# Transport Level

- The most important transport layer protocols:
  - **TCP (Transmission Control Protocol)** – connection-oriented transport protocol;  
RFC 793 (1122), 1323
  - **UDP (User Datagram Protocol)** – connectionless transport protocol;  
RFC 768

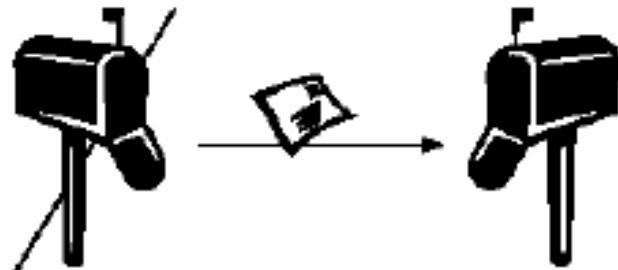
# UDP

- Connectionless transport protocol, unsafe, minimal
- Does not offer any additional quality to services
- There is no negotiation regarding receiving data or data confirmation

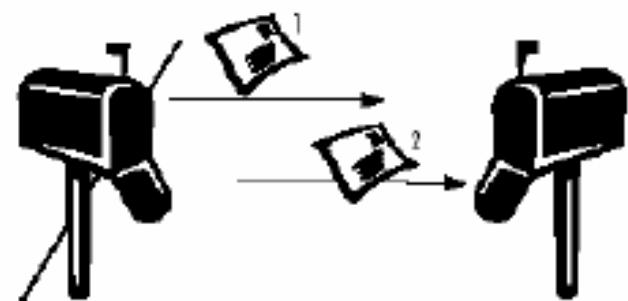
# UDP

- **Analogy:** UDP – similar to post system

➤ Sending a letter



➤ Does not guarantee the receiving order



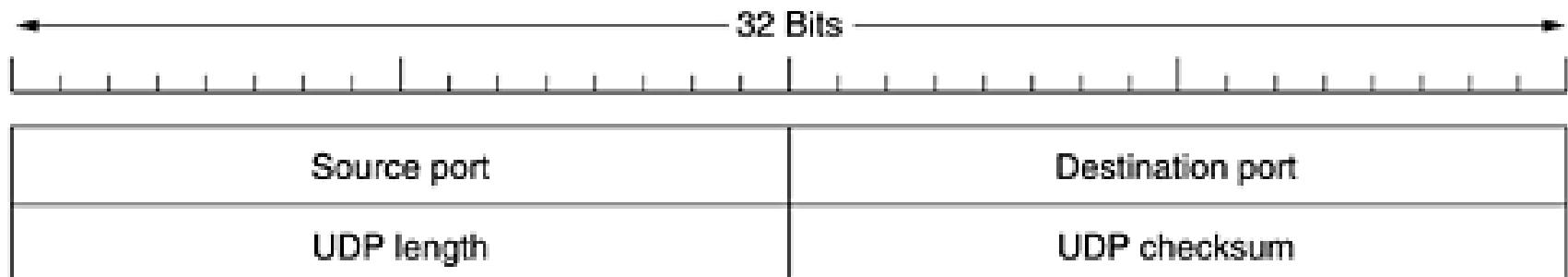
➤ The message may be lost



[conform Retele de calculatoare – curs  
2007-2008, Sabin Buraga]

# UDP

- Uses IP
- Offers communication services between processes using **ports**
- UDP sends packages: header (8 bytes) + content



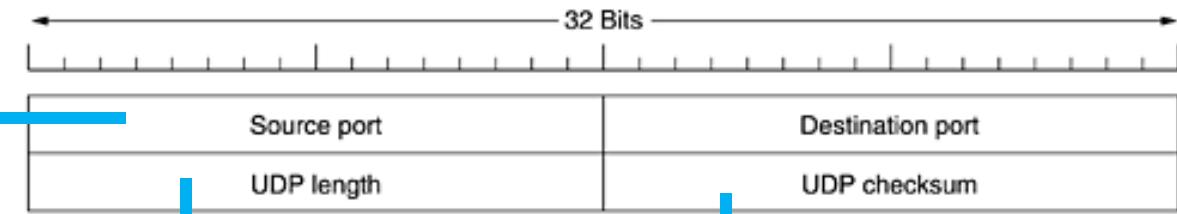
**Figure: UDP Header**

[Computer Networks, 2010 –  
Andrew S. Tanenbaum, et.al.]

# UDP

- *Source port* and *Destination port*
- identify "end-point" from the source and destination machines

Figure: UDP Header



*UDP length* = 8  
bytes +  
content size

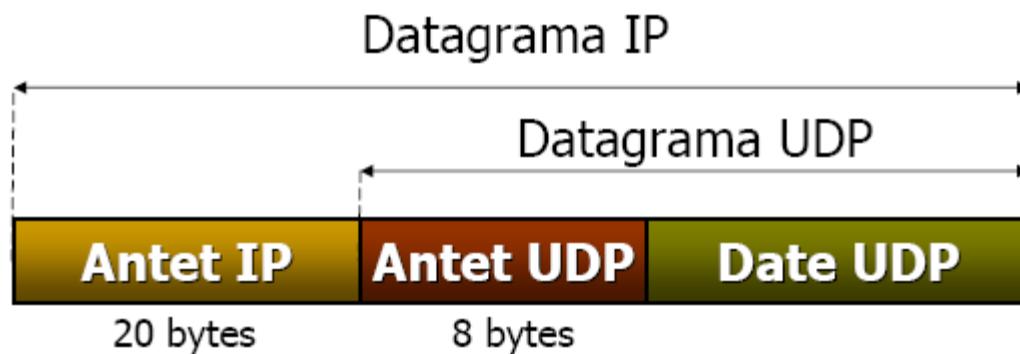
- *UDP checksum* (is not required)

# UDP

- Examples of uses:
  - **DNS** (Domain Name System)
    - Use-case:* A needs the host's IP which has the name [www.info.uaic.ro](http://www.info.uaic.ro)
    - Pas 1. A sends a UDP packet containing hostname: [www.info.uaic.ro](http://www.info.uaic.ro)
    - Pas 2. DNS server sends an UDP packet containing the host's IP address: 85.122.23.146 as response
  - **RPC** (Remote Procedure Call)

# UDP

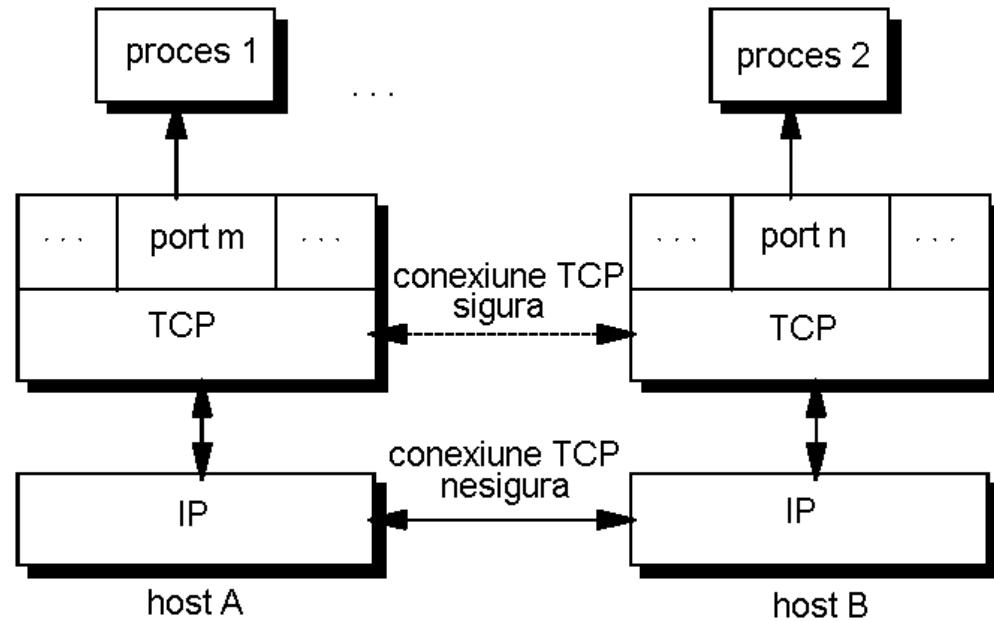
- What doesn't it offer?
  - flow control
  - error control
  - Retransmission of an error package
- What does it offer?
  - Using **ports** expands IP protocol for communication between processes running on different hosts (and not only between host as in the IP case)



[conform Retele de calculatoare – curs  
2007-2008, Sabin Buraga]

# TCP - Transmission Control Protocol

- Transport connection-oriented protocol without information loss
- Aimed at providing maximum quality of service
- Integrates mechanisms of establishing and releasing connections
- Controls the flow of data (stream-oriented)
- Used by many application protocols: HTTP, SMTP, ...

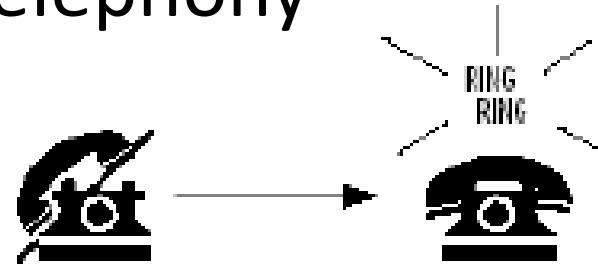


[conform IBM – TCP/IP Tutorial and Technical Overview, 2006]

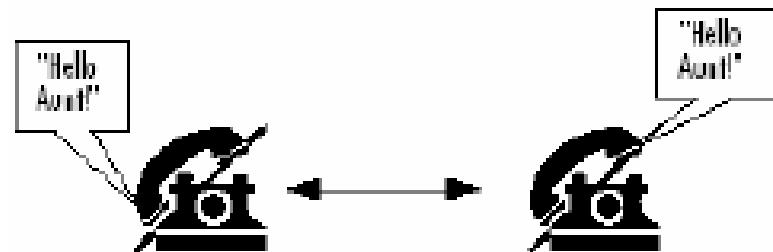
# TCP

- Analogy: TCP – similar to telephony

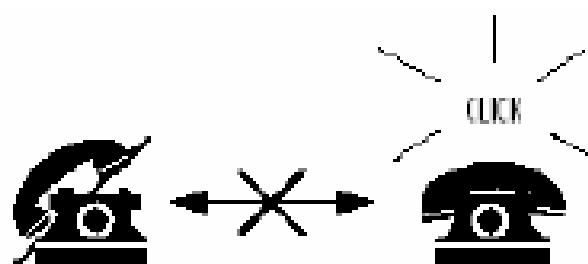
➤ Initiating a call



➤ Dialogue between parties



➤ Call ends



[conform Retele de calculatoare – curs  
2007-2008, Sabin Buraga]

# TCP

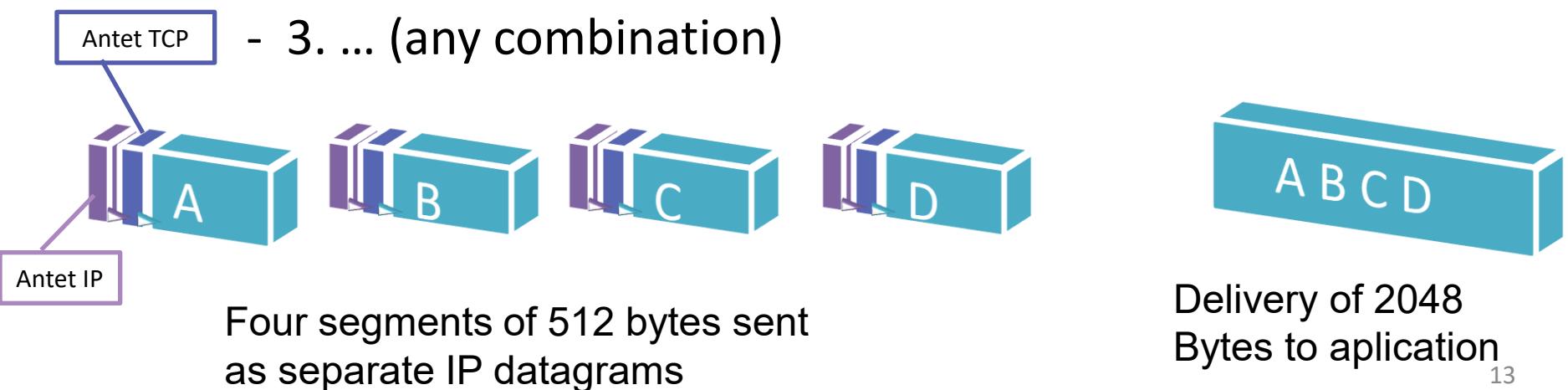
- Uses **connections**, not ports as fundamental abstractions
- Both parties (sender, recipient) must participate to the connection
- Connections are identified by pairs represented by  
**IP adress :PORT (socket)**
  - Example:  
(85.122.23.146: 12345, 85.122.23.146: 22)
- One party provides a **passive open** - expect occurrence of connection requests; the communication partner accomplishes an **active open**
- A socket can be shared by multiple connections from the same machine

# TCP

- TCP connections are full-duplex
- A TCP is a stream of bytes and not a message flow

Example:

- The transmitter sends four fragments of 512 bytes
- The receiver may receive:
  - 1. Two fragments of 1024 bytes
  - 2. A fragment of one byte and one of 2047 bytes
  - 3. ... (any combination)



# TCP

- **TCP finite state machine**
  - Shapes protocol behavior
  - The states are used for connection management

State	Description
CLOSED	No connection is active or pending
LISTEN	The server is waiting for an incoming call
SYN RCVD	A connection request has arrived; wait for ACK
SYN SENT	The application has started to open a connection
ESTABLISHED	The normal data transfer state
FIN WAIT 1	The application has said it is finished
FIN WAIT 2	The other side has agreed to release
TIMED WAIT	Wait for all packets to die off
CLOSING	Both sides have tried to close simultaneously
CLOSE WAIT	The other side has initiated a release
LAST ACK	Wait for all packets to die off

Figure. States of TCP finite state machine

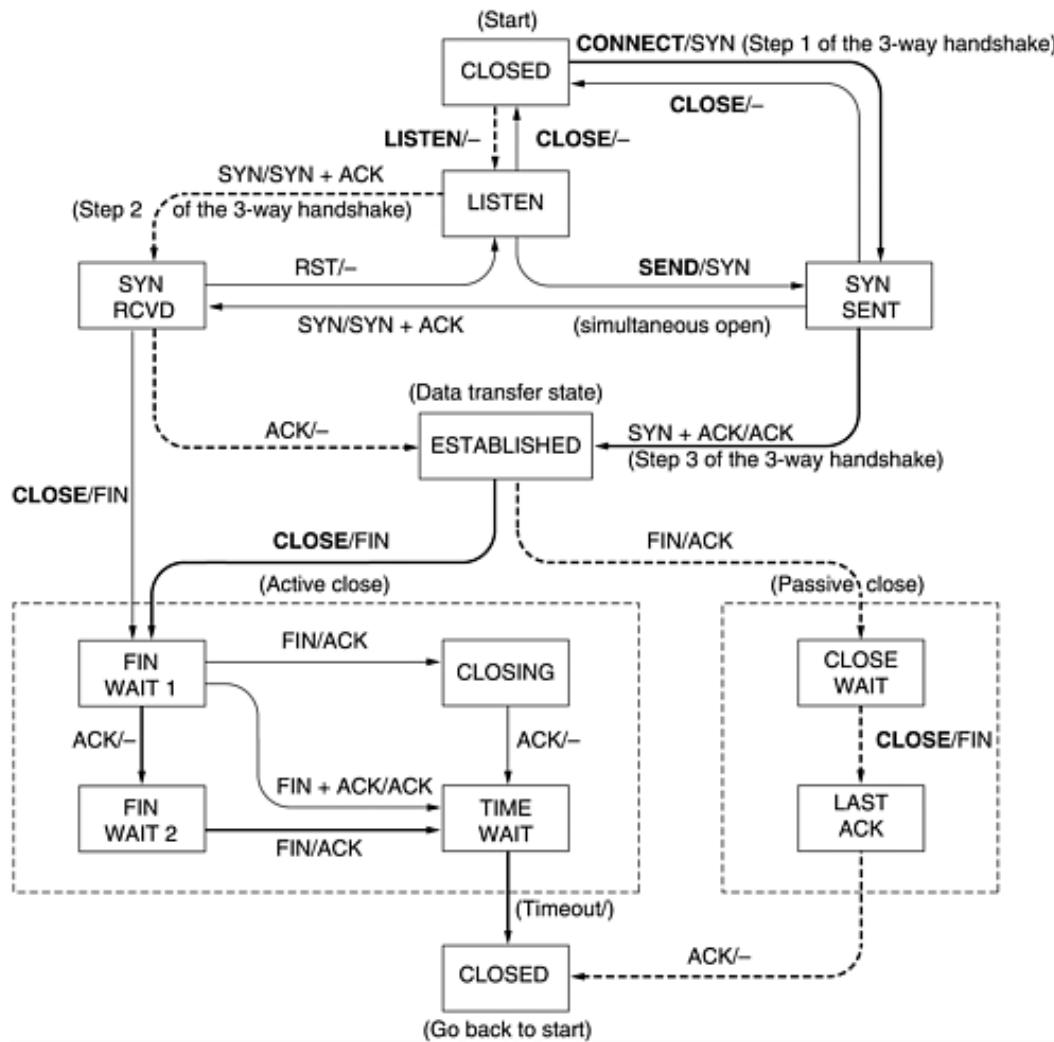
[conform Computer Networks, 2010 – Andrew S. Tanenbaum, et.al.]

# TCP

## TCP finite state machine

- Shape protocol behavior

- The states are used for connection management



- Each line is labeled with a pair event / action
- Example: ACK/-

[Computer Networks, 2010  
– Andrew S. Tanenbaum,  
et.al.]

# TCP

- TCP finite state machine
  - Connection Establishment:
    - CLOSED – from this state an active open can be requested (it passes in SYN\_SENT) or active open (SYN\_RCVD)
    - LISTEN – – from this state an active open can be requested (it passes in SYN\_SENT) or active open (SYN\_RCVD\0)
  - Established connection:
    - ESTABLISHED – The data can be sent (from this state it can be passed in CLOSE\_WAIT or FIN\_WAIT\_1)
  - Disconnecting initiated by the partner
    - CLOSE\_WAIT, LAST\_ACK, CLOSE
  - States that intervene in the process of disconnection
    - FIN\_WAIT\_1, FIN\_WAIT\_2, CLOSING, TIME\_WAIT

# TCP

Example:

netstat -t

```
adria@thor:~/html/teach/courses/net$ netstat -t
Active Internet connections (w/o servers)
Proto Recv-Q Send-Q Local Address          Foreign Address        State
tcp      0      0 thor.info.uaic.ro:smtp   186.122.246.108:33374  SYN_RECV
tcp      0      0 localhost:60000           localhost:45740       ESTABLISHED
tcp      0      0 thor.info.uaic.ro:imaps  mail.traveltech.c:55156  ESTABLISHED
tcp      0      0 localhost:45740           localhost:60000       ESTABLISHED
tcp      0      0 thor.info.uaic.ro:imaps  81.253.57.112:51462  ESTABLISHED
tcp      0      0 thor.info.uaic.ro:smtp   fenrir.info.uaic.:59806  TIME_WAIT
tcp      0      0 thor.info.uaic.ro:imaps  ia.info.uaic.ro:51058  ESTABLISHED
tcp      0      0 thor.info.uaic.ro:imaps  77-58-250-39.dcli:56424  ESTABLISHED
tcp      0      0 thor.info.uaic.ro:59605  fenrir.info.uaic.ro:ssh  ESTABLISHED
tcp      0      0 localhost:57572           localhost:spamd       TIME_WAIT
tcp      0      0 thor.info.uaic.ro:pop3s  79-112-42-154.iasi:2019 ESTABLISHED
tcp      0      0 thor.info.uaic.ro:pop3s  info-c-117.info.ua:1302  ESTABLISHED
tcp      0      0 thor.info.uaic.ro:ssh   mail.traveltech.c:52266  ESTABLISHED
tcp      0      0 thor.info.uaic.ro:pop3s  info-c-59.info.ua:50149  ESTABLISHED
tcp      0      0 thor.info.uaic.ro:imaps  mail.traveltech.c:55152  ESTABLISHED
tcp      0      0 localhost:58053           localhost:10025       TIME_WAIT
tcp      0      0 thor.info.uaic.ro:imaps  info-c-70.info.uai:1266  ESTABLISHED
tcp      0      0 thor.info.uaic.ro:imaps  info-c-91.info.uai:4285  ESTABLISHED
tcp      0      0 192.168.103.2:39107    pdc:65022       ESTABLISHED
tcp      0      0 thor.info.uaic.ro:pop3s  info-c-59.info.ua:55896  ESTABLISHED
tcp      0      0 thor.info.uaic.ro:imaps  77-58-250-39.dcli:56476  ESTABLISHED
tcp      0      0 192.168.103.2:39082    pdc:65022       ESTABLISHED
tcp      0      0 thor.info.uaic.ro:smtp   main.dntis.ro:52854   ESTABLISHED
tcp      0      0 thor.info.uaic.ro:imaps  ws70-228.unine.ch:35907  ESTABLISHED
tcp      0      0 thor.info.uaic.ro:imaps  info-c-70.info.uai:1364  ESTABLISHED
tcp      0      0 thor.info.uaic.ro:ssh   ia.info.uaic.ro:40542   ESTABLISHED
tcp      0      396 thor.info.uaic.ro:ssh  79-112-21-031.ias:54540  ESTABLISHED
tcp      0      0 thor.info.uaic.ro:imaps  ia.info.uaic.ro:39325   ESTABLISHED
tcp      0      0 thor.info.uaic.ro:pop3s  79-112-42-179.iasi:1146  ESTABLISHED
tcp      0      0 thor.info.uaic.ro:imaps  81.253.57.112:51461   ESTABLISHED
```

# TCP

- Examples of TCP uses:
  - Most application protocols :
    - HTTP
    - TELNET
    - SMTP
    - SSH
    - ...

# TCP versus UDP

- Both rely on IP uses ports
- Transmission unit
  - TCP -> TCP Segment
  - UDP -> UDP Packet

# TCP versus UDP

- UDP offers minimal transport services (minimum effort transmission)
- TCP provides connection-oriented, full-duplex, safe - transport streams of bytes (-> complex transmission mechanism)
- Using TCP or UDP depends on the application: e-mail, file transfer, operating in real-time multimedia transmissions in real time chat, ...

# Network Communication paradigms

- Client/Server model
- Remote Procedure Call (**RPC**) (Future Course)
- Peer-to-Peer (**P2P**) mechanism – point-to-point communication (Future Course)

# Client/Server model

- **Server Process**
  - Provides network services
  - Accept requests from a client process
  - Performs a specific service and returns the result
- **Client Process**
  - Initializes communication with the server
  - Requests a service and expects the server's response



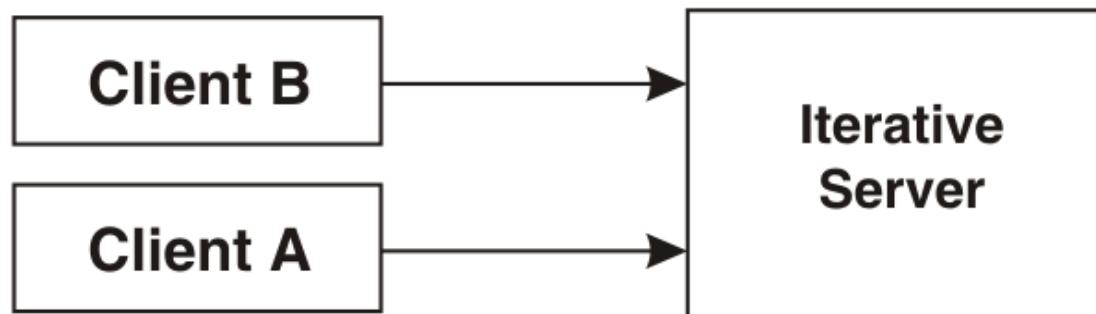
[The first Web Server]

# Client/Server model

- Interaction alternatives:
  - **Connection-oriented** – based on TCP
  - **Connectionless** – based on UDP

# Client/Server model

- Implementation:
  - **iterative** – each client is treated at a time, sequentially

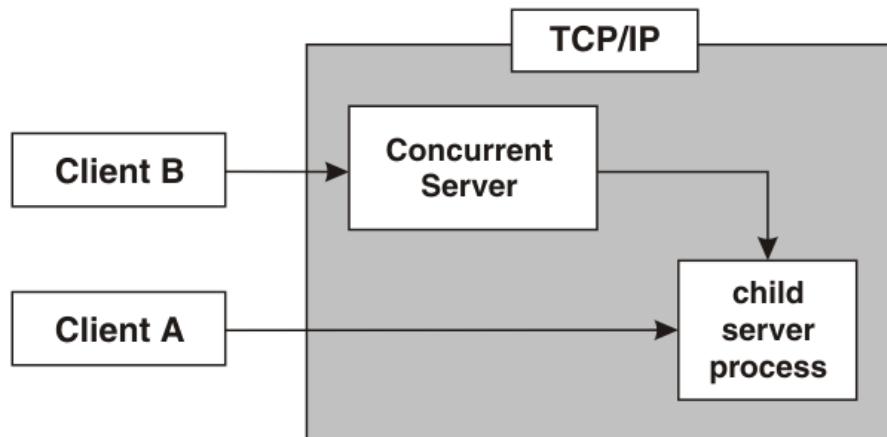


**Figure:** Example of iterative server

[<http://publib.boulder.ibm.com>]

# Client/Server model

- Implementation:
  - **Concurrency** – the requests are processed concurrently
    - A child process for each request
    - Multiplexing
    - Combination techniques



**Figure:** Concurrent server example

[<http://publib.boulder.ibm.com>]

# API for network programming

- Necessity:
  - Generic interface for programming
  - Hardware and operating system independence
  - Support for different communication protocols
  - Support for connection-oriented communications or for connectionless communications
  - Independence in address representation
  - Compatibility with the common I/O services

# API for network programming

- For programming Internet application multiple APIs can be used:
  - *BSD Sockets*(*Berkeley System Distribution*)
  - TLI (*Transport Layer Interface*) – AT&T, XTI (Solaris)
  - Winsock
  - MacTCP
- Functions offered:  
specifying local and remote endpoints, initiating and accepting connections, sending and receiving data, end connection, error treatments
- TCP/IP does not include an API definition

...

# Application programming interface based on

## BSD sockets

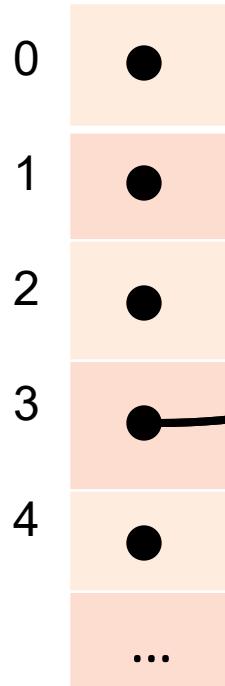
# Socket

- General facility, independent of hardware architecture, protocol and type of data transmission for communication among processes on different network machines
- Offers support to multiple family protocols:
  - UNIX domain protocol – used for local communications
  - Internet domain protocol using TCP / IP
  - Other: XNS Xerox,...
- Abstraction of an end-point at the transport level

# Socket

- Uses the existing I/O programming interface (similar to files, *pipes*, *FIFOs* etc.)
- May be associated with one/multiple processes from a communication domain
- It provides an API for network programming, having multiple implementations
- From the point of view of the programmer, a socket is similar to a file descriptor; the differences occur when sockets are created or when you set control options for a socket

# Socket



**Descriptors Table**

Family: AF\_INET  
Service: SOCK\_STREAM  
Local IP: 85.122.23.145  
Local Port: 80  
Remote IP: 79.112.89.206  
Remote Port: 2021

# Application programming interface based on BSD sockets

## Basic primitives:

- **socket()** – creates a new connection end-point
- **bind()** - attaches a local address to a *socket*
- **listen()** – allows a *socket* to accept connections
- **accept()** – blocks the caller until a connection request appears (used by TCP server)
- **connect()** attempt (active) to establish the connection (used by TCP client)
- **send()** sending data via *socket*
- **recv()** receiving data via *socket*
- **close()** releases the connection(close *socket*)
- **shutdown()** closes a *socket* in one direction

# Application programming interface based on BSD sockets

## Other primitives:

- Data read
  - `read()` / `readv()` / `recvfrom()` / `recvmsg()`
- Data sent
  - `write()` / `writev()` / `sendto()` / `sendmsg()`
- I/O Multiplexing
  - `select()`
- Managing connection
  - `fcntl()` / `ioctl()` / `setsockopt()` / `getsockopt()` /  
`getsockname()` / `getpeername()`

# Socket | Creation

## socket() call system

```
#include <sys/types.h>
```

```
#include <sys/socket.h>
```

```
int socket (int domain, int type, int protocol)
```

Communication domain:  
`AF_UNIX, AF_INET,`  
`AF_INET6, ...`

The **protocol** used for  
transmission  
(Usually: **0** for the  
transport)

**Socket type** (ways to accomplish the  
communication): `SOCK_STREAM,`  
`SOCK_DGRAM, SOCK_RAW`

# Socket | Creation

## socket() system call

Return value

- Success: the *socket* descriptor
- Error: -1
  - Error reporting is done via **errno** variable
    - EACCES
    - EAFNOSUPPORT
    - ENFILE
    - ENOBUFS sau ENOMEM
    - EPROTONOSUPPORT
    - ...



Constants defined in  
**errno.h**

# Socket-uri

Example of possible combinations for the socket()'s arguments:

```
int socket (int domain, int type, int protocol)
```

Domeniu	Tip	Protocol
AF_INET	SOCK_STREAM	TCP
	SOCK_DGRAM	UDP
	SOCK_RAW	IP
AF_INET6	SOCK_STREAM	TCP
	SOCK_DGRAM	UDP
	SOCK_RAW	IPv6
AF_LOCAL	SOCK_STREAM	Internal communication mechanism
	SOCK_DGRAM	

Note: AF\_LOCAL=AF\_UNIX (for historical reasons)

# Sockets

## Observations

- *socket()* allocates resources for a communication end-point, but it doesn't state which is the addressing mechanism
- *Sockets* provide a generic addressing mechanism; for TCP/IP , it must be specified (*IP address, port*)
- Other protocols suite may use other addressing mechanisms

## POSIX types:

`int8_t, uint8_t, int16_t, uint16_t, int32_t, int32_t,  
u_char, u_short, u_int, u_long`

# Sockets

- POSIX types used by sockets:
  - `sa_family_t` – address family
  - `socklen_t` – structure length
  - `in_addr_t` – IP address (v4)
  - `in_port_t` – port number
- Specifying generic addresses

```
struct sockaddr {  
    sa_family_t  sa_family;  
    char sa_data[14]  
}
```

Address family: AF\_INET,  
AF\_ISO,...

14 bytes – used address

# Sockets

- For IPv4 AF\_INET a special structure is used: `sockaddr_in`

```
struct sockaddr_in {
```

```
    short int sin_family;
```

Address family: AF\_INET

```
    unsigned short int sin_port;
```

Port (2 octets)

```
    struct in_addr sin_addr;
```

```
    unsigned char sin_zero[8];
```

```
}
```

```
struct in_addr{
```

Unused Bytes

```
    in_addr_t s_addr;
```

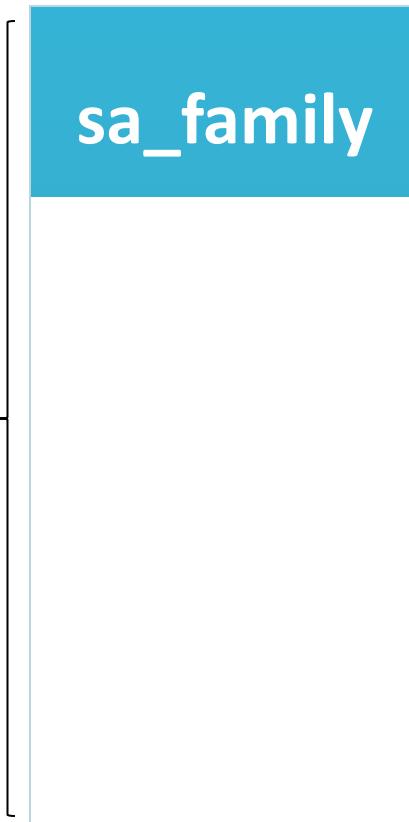
```
}
```

4 bytes for IP  
address

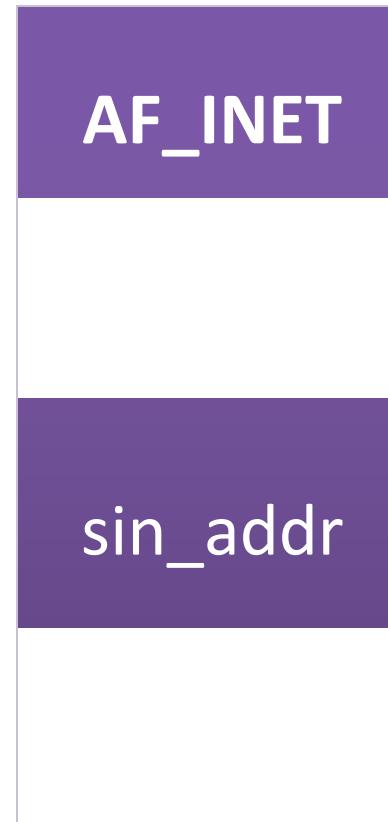
# Sockets

**sockaddr**

Allow any  
addressing  
type



**sockaddr\_in**



# Sockets

- The values from `sokaddr_in` are stored in respect to *network byte order*
- Conversion functions (`netinet/in.h`)
  - **`uint16_t htons (uint16_t)`** – converting a short integer (2 bytes) from host to network;
  - **`uint16_t ntohs (uint16_t);`**
  - **`uint32_t ntohl (uint32_t)`** – converting a long integer (4 bytes) from network to host;
  - **`uint32_t htonl (unit32_t);`**

# Discussion | Octets order

Byte order of a word (*word* - two bytes) can be achieved in two ways:

- **Big-Endian** – The most significant byte is first
- **Little-Endian** – The most significant byte is the second

**Example:**

BigEndian machine sends  
(e.g. Motorola processor)

00000000	00000010	=2
----------	----------	----

LittleEndian machine will perform:  
(e.g. Intel processor)

00000010	00000000	=512
----------	----------	------



As convention,  
*network byte order - BigEndian*

# Sockets

- For IPv6 AF\_INET6 `sockaddr_in6` structure it is used:

```
struct sockaddr_in6 {  
    u_int16_t sin6_family; /* AF_INET6 */  
    u_int16_t sin6_port;  
    u_int32_t sin6_flowinfo;  
    struct in6_addr sin6_addr;  
    u_int32_t sin6_scope_id;  
}
```

```
struct in6_addr{  
    unsigned char s6_addr[16];  
}
```

# Sockets

## Example:

// IPv4:

```
struct sockaddr_in ip4addr; int s;  
ip4addr.sin_family = AF_INET;  
ip4addr.sin_port = htons(2510);  
inet_pton(AF_INET, "10.0.0.1", &ip4addr.sin_addr);  
s = socket(PF_INET, SOCK_STREAM, 0);  
bind(s, (struct sockaddr*)&ip4addr, sizeof(ip4addr));
```

Convert IPv4 and IPv6  
addresses from string  
(x.x.x.x) in network byte  
order  
(#include <arpa/inet.h>)

// IPv6:

```
struct sockaddr_in6 ip6addr; int s;  
ip6addr.sin6_family = AF_INET6;  
ip6addr.sin6_port = htons(2610);  
inet_pton(AF_INET6, "2001:db8:8714:3a90::12", &ip6addr.sin6_addr);  
s = socket(PF_INET6, SOCK_STREAM, 0);  
bind(s, (struct sockaddr*)&ip6addr, sizeof(ip6addr));
```

? (next slide)



# Sockets (slide 19)



## Observations

- *socket()* allocates resources for a communication end-point, but it doesn't state which is the addressing mechanism
- *Sockets* provide a generic addressing mechanism; for TCP/IP it must be specified (*IP address, port*)
- Other protocols suite may use other addressing mechanism



# Sockets | Assigning an address

- Assigning an address to an existing socket is made with **bind()**

```
int bind ( int sockfd,  
const struct sockaddr *addr,  
int addrlen );
```

- It returns: 0 if successful, -1 on error  
**errno** variable will contain the corresponding error code:  
**EACCES** , **EADDRINUSE**, **EBADF**, **EINVAL**, **ENOTSOCK**,...

# Sockets | Assigning an address

## Example:

```
#define PORT 2021
struct sockaddr_in adresa;
int sd;

sd = socket (AF_INET, SOCK_STREAM, 0)) // TCP
adresa.sin_family = AF_INET; // establish socket family
adresa.sin_addr.s_addr = htonl (IPaddress); //IP address
adresa.sin_port = htons (PORT); //port

if (bind (sd, (struct sockaddr *) &adresa, size of (adresa) == -1)
{
    perror ("Eroare la bind().\n");
    ...
}
```

# Sockets | Assigning an address

- bind() uses:
  - The server wants to attach a socket to a default port (to provide services via that port)
  - The client wants to attach a socket to a specified port
  - The client asks the operating system to assign any available port
- Normally, the client does not require attachment to a specified port
- Choose any free port:

```
adresa.sin_port = htons(0);
```

# Sockets | Assigning an address

- Choosing the IP address - `bind()`
  - If the host has multiple IP addresses assigned?
  - How to solve platform independence?



To attach a socket to your local IP address,  
`INADDR_ANY` constant will be used instead

# Sockets | Assigning an address

- IP address conversion:

```
int inet_aton (const char *cp, struct in_addr *inp);
```

ASCII “x.x.x.x” -> 32 bits internal representation  
*(network byte order)*

```
char *inet_ntoa(struct in_addr in);
```

32 bits representation *(network byte order)*->  
ASCII “x.x.x.x”

Obs: [@fenrir ~]\$ man **inet\_addr**

# Sockets | Assigning an address

- Observations:
  - For Pv6 INADDR\_ANY will be replaced by IN6ADDR\_ANY(**netinet/in.h**):  
**serv.sin6\_addr = in6addr\_any;**
  - The conversion function for IPv6 (that can be used for IPv4) are:  
**inet\_ntop()**  
**inet\_ntop()**

# Sockets | listen()

- Passive interaction:
  - The system core will wait for connection requests directed to the address where the socket is attached
  - The received connections will be placed in a queue
  - It returns: 0 – success, -1 - error

**int listen(int sockfd, int backlog);**

The number of connections in the queue

TCP socket attached to an address

# Sockets | listen()

- Observations:
  - The *backlog* value depends on the application (usually 5)
  - HTTP servers should specify a bigger value for *backlog* (due the multiple requests)

# Sockets | accept()

- Accepting the connections from clients
  - When the application is ready to address a new connection, the system will interrogate for another connection

```
int accept (int sockfd,  
           struct sockaddr *cliaddr,  
           socklen_t *addrlen);
```

Socket TCP  
(passive mode)

- It must initially be equal to the length of the **cliaddr** structure
  - It will return the number of bytes used in **cliaddr**

It returns the socket descriptor corresponding to the client endpoint or -1 in an error case

# Sockets | connect()

- Trying to establish a connection to the server

*3-way handshake*

```
int connect (int sockfd,  
            const struct sockaddr *serv_addr,  
            socklen_t addrlen);
```

Socket TCP

- It does not require attaching with bind(); the operating system will assign a local address (IP, port)

Contains server address  
(IP, port)

It returns: success -> 0, error -> -1

# I/O TCP | read()

```
int read(int sockfd, void *buf, int max);
```

- The call is usually a blocking one, `read()` returns only when data are available
- Reading from a TCP socket may return fewer bytes than the maximum number desired
  - We must be prepared to read byte-by-byte at a time (see previous course)
- If the communication partner closes the connection and there are no data to receive, `0 (EOF)` is returned
- Errors: `EINTR` – a signal interrupted the reading, `EIO` – I/O error, `EWOULDBLOCK` – the *socket* set in a non-blocking mode, doesn't have data

# I/O TCP | write()

```
int write(int sockfd, const void *buf, int count);
```

- The call is usually a blocking one
- Errors:
  - EPIPE – write to a offline socket
  - EWOULDBLOCK – normally writes blocks until the writing operation is complete; if the socket is set in non-blocking mode and some problems occur, it returns this error immediately

# I/O TCP | Example

```
#define MAXBUF 127 /* reading buffer length*/  
...  
char *cerere= "da-mi ceva";  
char buf [MAXBUF]; /* response buffer*/  
char *pbuff= buf; /* buffer pointer */  
int n, lung = MAXBUF; /* the nr. of bytes read, the nr. of free bytes  
in buffer */  
...  
/* send the request*/  
write(sd, cerere, strlen(cerere));  
  
/* wait the response*/  
while ((n = read (sd, pbuff, lung)) > 0) {  
    pbuff+= n;  
    lung -= n;  
}  
}
```

Example of communication  
between client and sever

# Closing the connection | `close()`

`int close( int sockfd)`

- Effect:
  - closes the connection;
  - Frees the memory associated with the *socket*
    - for processes that share the same socket, it decreases the number of references to that socket; if it reaches 0, than the socket is deleted
- Problems:
  - The server cannot end the connection, it doesn't know if and when the client sends other demands
  - The client doesn't know if the data reaches the server

# Closing the connection | shutdown()

- Unidirectional closing
  - When a client finishes to send requests, it can call `shutdown()` to specify that data will be sent no longer (the socket is not deleted)
  - The server will receive EOF and after sending the last answer to the client, it will close the connection

```
#include <sys/socket.h>
int shutdown (int sockfd, int how);
```

- 0 – future reading from the socket will not be allowed (SHUT\_RD);
- 1 – future writings will not be allowed (SHUT\_WR);
- 2 - read/write operations are no longer allowed (SHUT\_RDWR)

# Metaphor for Good Relationships

Copyright Dr. Laura's Network Programming Corp.

To succeed in relationships...

- you need to establish your own identity.
- you need to be open & accepting. *accept()*
- you need to establish contacts. *connect()*
- you need to take things as they come, not as you expect them. *read might return 1 byte*
- you need to handle problems as they arise. *check for errors*

*bind()*

# Client/Server Model

- **TCP iterative server:**
  - Creates *socket* to address clients: `socket()`
  - Prepares data structures (`sockaddr_in`)
  - Attaches the socket to the local address (port): `bind()`
  - Prepares the socket to listen in order to establish connections with clients `listen()`
  - The expectation of making a connection with a particular client (passive open): `accept()`
  - Processes customer requests, using the socket returned by `accept()`: sequence of `read()/write()` calls
  - Closes (unidirectional close) of the connection: `close()`, `shutdown()`

# Model client/server

- **TCP client model:**
  - Creates a *socket* to connect to a server: `socket()`
  - Prepares data structures (`sockaddr_in`)
  - Attaches the socket: `bind()` – optional
  - Connects to the server (active open): `connect()`
  - Service request/receives the results sent by the server:  
sequence of `read()/write()` calls
  - Closes (unidirectional close) the connection: `close()`,  
`shutdown()`

# General model – TCP server/client

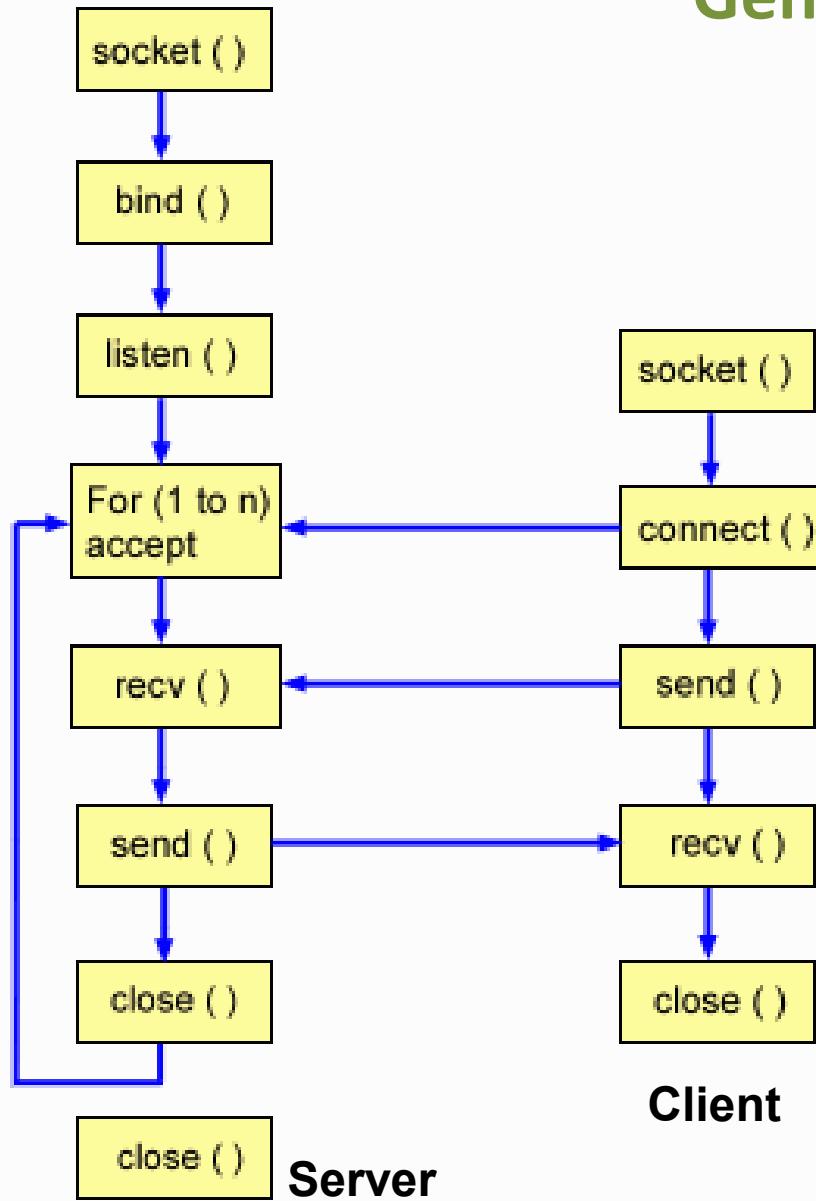
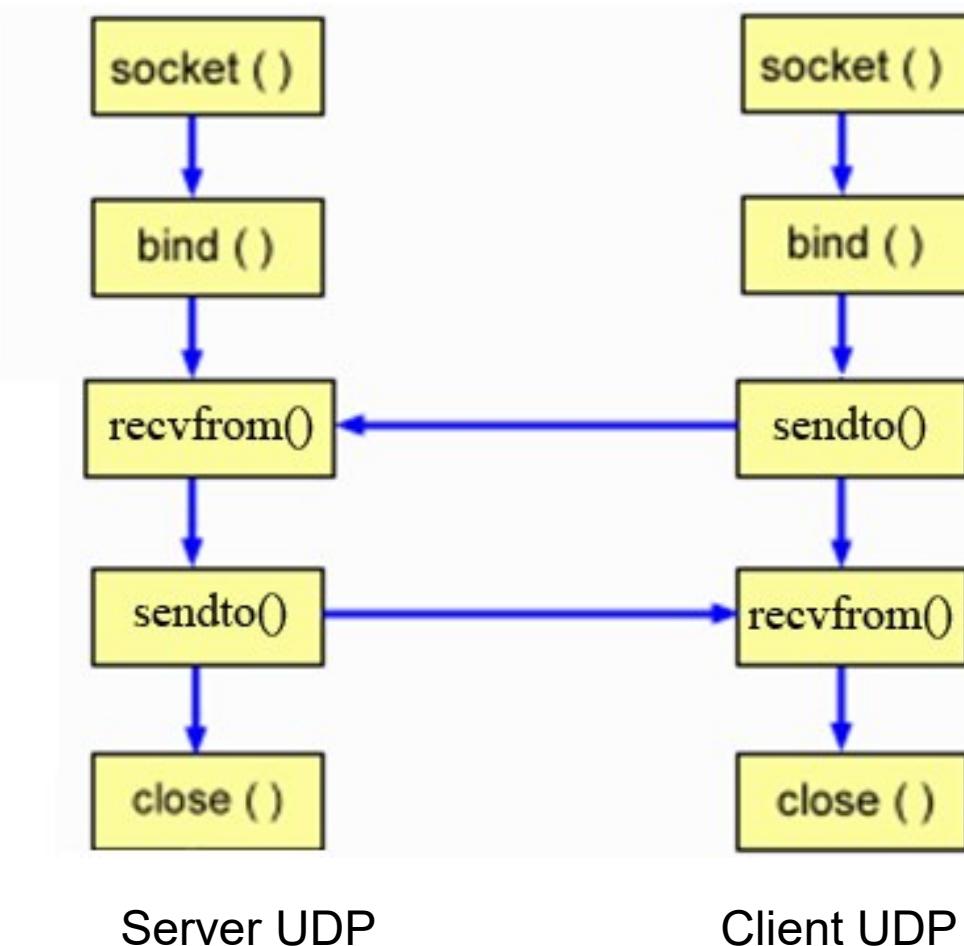


Figure: TCP Iterative Server - sequence of events

[<http://publib.boulder.ibm.com>]

# UDP Client/Server Model



Server UDP

Client UDP

# UDP Client/Server model

- For `socket()` it is used `SOCK_DGRAM`
- `listen()`, `accept()`, `connect()` are not usually used
- For datagrams sending it can be used `sendto()` or `send()`
- For datagrams reading it can be used `recvfrom()` or `recv()`
- Nobody guarantees that the sent data have reached the addressee or is not duplicate

# UDP Client/Server model

- UDP sockets can be “connected”: the client can use `connect()` to specify the server address (`IP, port`) – **pseudo-connections**:
  - Utility: sending several datagrams to the same server, without specifying server address for each datagram
  - For UDP, `connect()` will retain only the information about the endpoint without getting initiate any data exchange
  - Although `connect()` reports success does not mean that the address is a valid point or the terminal server is available

# UDP Client/Server model

- UDP Pseudo-connections
  - `shutdown()` can be used to stop transmitting data in one direction, but no message will be sent to the conversation partner
  - `close()` can be called to remove a pseudo-connection

# I/O primitives

```
#include <sys/types.h>
#include <sys/socket.h>
int send (int sockfd, char *buff, int nbytes, int flags);
int recv (int sockfd, char *buff, int nbytes, int flags);
```

- They can be used in the connection-oriented communications or pseudo-connections
- `send()` and `recv()` assume that a previous `connect()` call was performed
- The first 3 arguments argumente sunt similare cu cele de la `write()`, respectiv `read()`
- The fourth argument is usually 0, but can have other values that specify conditions for the call
- Both calls return at normal execution, the transfer length (in bytes)

# I/O primitives

```
#include <sys/types.h>
#include <sys/socket.h>

int sendto ( int sockfd, char *buff, int nbytes, int flags,
              struct sockaddr *to, int addrlen);
int recvfrom (int sockfd, char *buff, int nbytes, int flags,
              struct sockaddr *from, int *addrlen);
```

- Used for connectionless communications
- At **sendto()** and **recvfrom()** the elements to identify the remote node is specified in the last two arguments
- Both calls return, in normal execution, the transfer length in bytes

# I/O primitives

```
#include <sys/uio.h>  
ssize_t readv (int fd, const struct iovec *iov, int iovcnt);  
ssize_t writev (int fd, const struct iovec *iov, int iovcnt);
```

- Wider than read()/write(), provides the ability to work with data in non-contiguous memory areas

```
#include <sys/types.h>  
#include <sys/socket.h>  
ssize_t recvmsg (int s, struct msghdr *msg, int flags);  
ssize_t sendmsg (int s, const struct msghdr *msg, int flags);
```

- Receives / transmits messages extract them from the *msghdr* structure

# Summary

## Transport Level

- Preliminary
- UDP (User Datagram Protocol)
- TCP (Transmission Control Protocol)
- TCP versus UDP

# Bibliography

- Andrew S. Tanenbaum, David J. Wetherall, Computer Networks (5th Edition), ISBN-10: 0132126958 , Publication Date: October 7, 2010
- A. Tanenbaum, Computer Networks. 4th Edition. Prentice Hall. 2003
- James F. Kurose, Keith W. Ross; Computer Networking: A Top-Down Approach (5th Edition), ISBN-10: 0136079679
- Lydia Parziale, David T. Britt, Chuck Davis, Jason Forrester, Wei Liu, Carolyn Matthews, Nicolas Rosselot , IBM – TCP/IP Tutorial and Technical Overview, 2006
- Tamara Dean, Network +Guide to Networks (Editia 5), ISBN-10: 1-423-90245-9, 2009



# Questions?

# P2P Paradigm

**Lenuta Alboiae**  
**adria@info.uaic.ro**

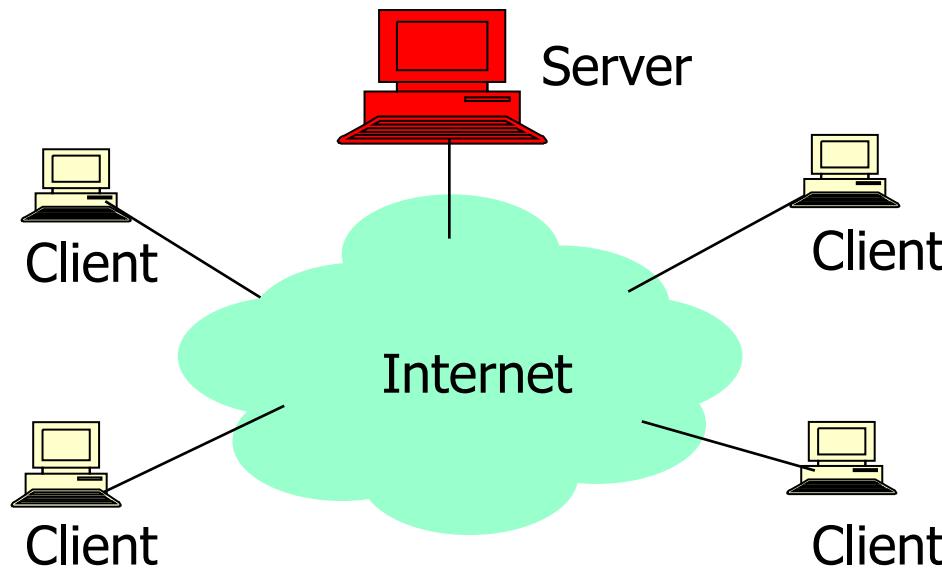


# Content

- **Peer-to-peer(P2P) paradigm**
  - Preliminaries
  - Definitions
  - Characteristics
  - Application types
  - Infrastructures
  - Instruments

# Preliminaries

...let's remember **client/server model**



# Preliminaries

...let's remember client/server model

- Usually, we look at the client as at a component having low computational capabilities
- The server is maintained and managed centrally

Problems of client / server architecture:

- The lack of robustness
- Lack of safety (Reliability)
- Lack of scalability
- Vulnerability to attack

# Definitions

**Peer** = *one that is of equal standing with another*  
(conform Webster)

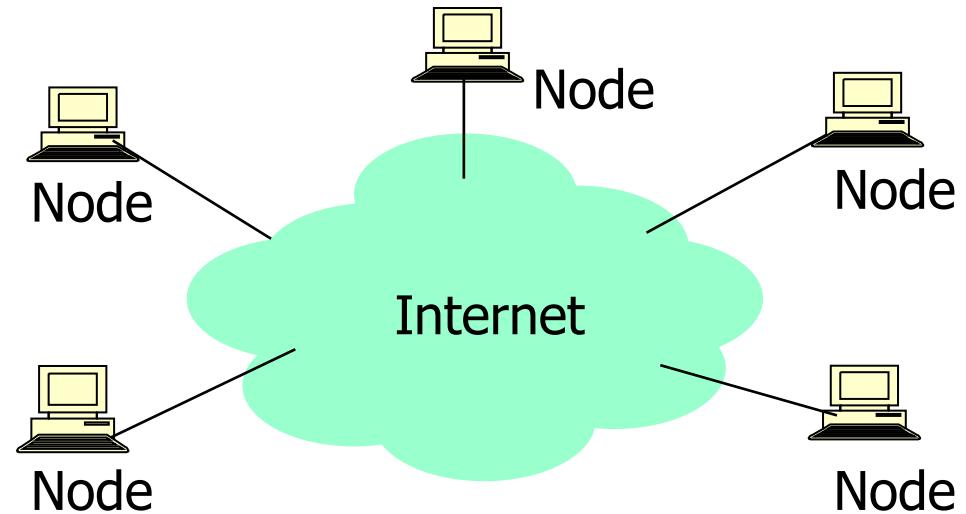
**Peer-to-peer** (P2P) = network architecture where nodes  
are relatively equal

- Meaning that each node is capable of performing specific network functions
- In practice, many of the nodes can also perform functions

# Definitions

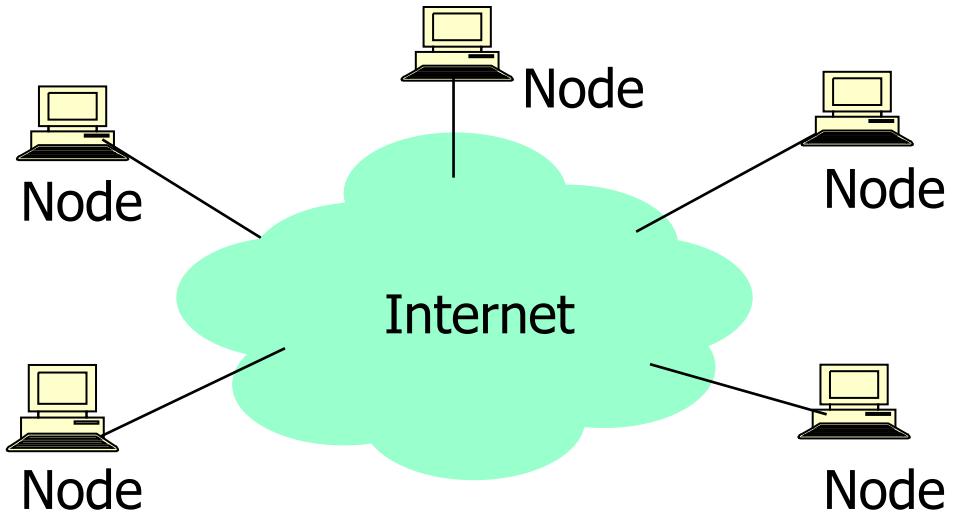
P2P Systems, in the strict sense, are fully distributed systems

- All nodes are fully equivalent in terms of functionality and activities they can perform



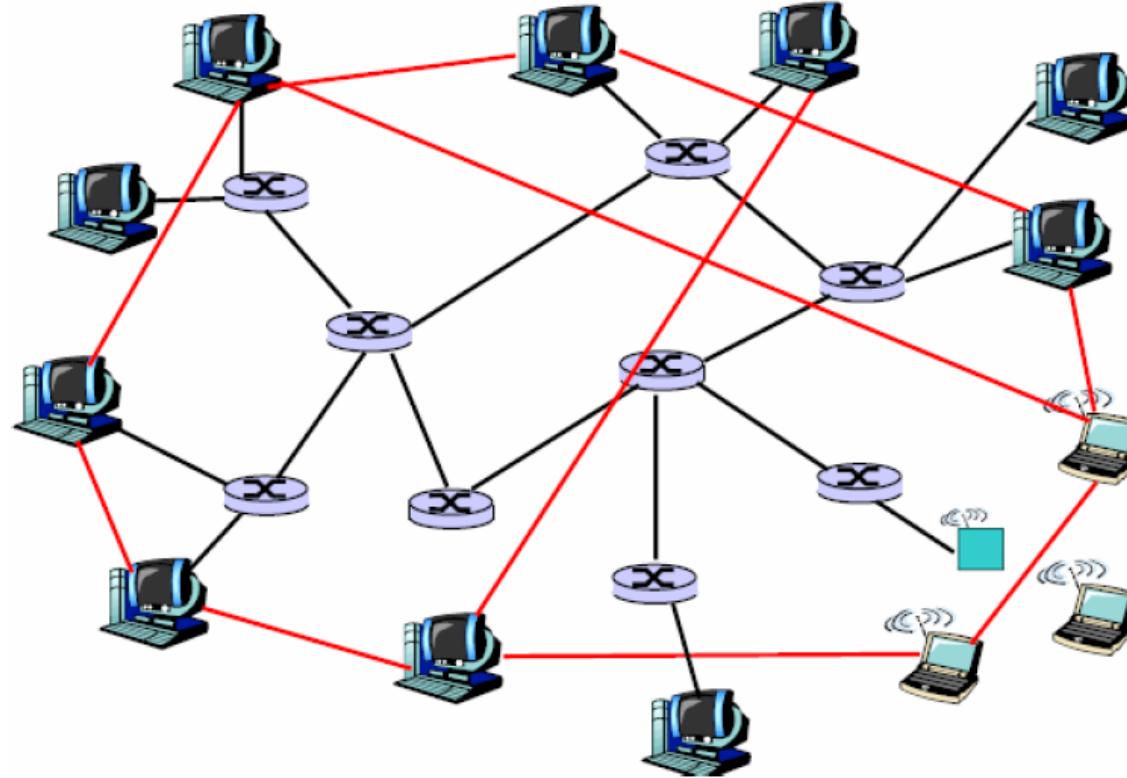
Obs. Pure P2P systems are rare (e.g. Gnutella), most are hybrids, having super-nodes or servers with different roles (data search, control)

# Definitions

- Nodes
    - Can consume and provide data
  - Any node can initiate a connection
  - There is no centralized data source =>
    - A form of democracy on the Internet
    - Copy-right protection threatened
- 

# Definitions

- “**P2P** is the class of applications that rely on the resources (storage, processing, content, human presence) available at the edges of the Internet



*Edges of the Internet  
(overlay networks)*

# Definitions

“**P2P** is a class of applications that take advantage of resources – storage, cycles, content, human presence – available at the edges of the Internet. Because accessing these decentralized resources means operating in an environment of unstable and unpredictable IP addresses, P2P nodes must operate outside the DNS system and have significant, or total autonomy from central servers”

“A distributed network architecture may be called a **P2P** network if the participants share a part of their own resources. These shared resources are necessary to provide the service offered by the network. The participants of such a network are both resource providers and resource consumers”

# Characterization

Characteristics:

- Sharing computational resources through interchange and less directly through mediation offered by a centralized authority (server)
  - Centralized servers can be used to perform specific activities (P2P network initialization, adding new nodes in the network ...)
  - Ideally, nodes participate actively in accomplishing operations such as location & caching nodes / content, routing information, transferred resource management etc.

# Characterization

Characteristics:

- The ability to address instability and variations of network connectivity, with adaptation for the occurred errors or for the node dynamics
  - P2P network topology is adaptive and fault tolerance; nodes are self-organized in order to maintain connectivity and network performance

# Characterization

P2P network is one overlay over the physical network

- It is at Application Level => flexibility
- Virtual edges are TCP connections or pointers to IP addresses
- P2P network maintenance is done by periodically verifying connectivity (ping) or existence (messages "still alive?")
- When a node fails, the P2P system could set new edges
- Nodes proximity (physical) is not important
- P2P network can be structured or not

# Aims and benefits

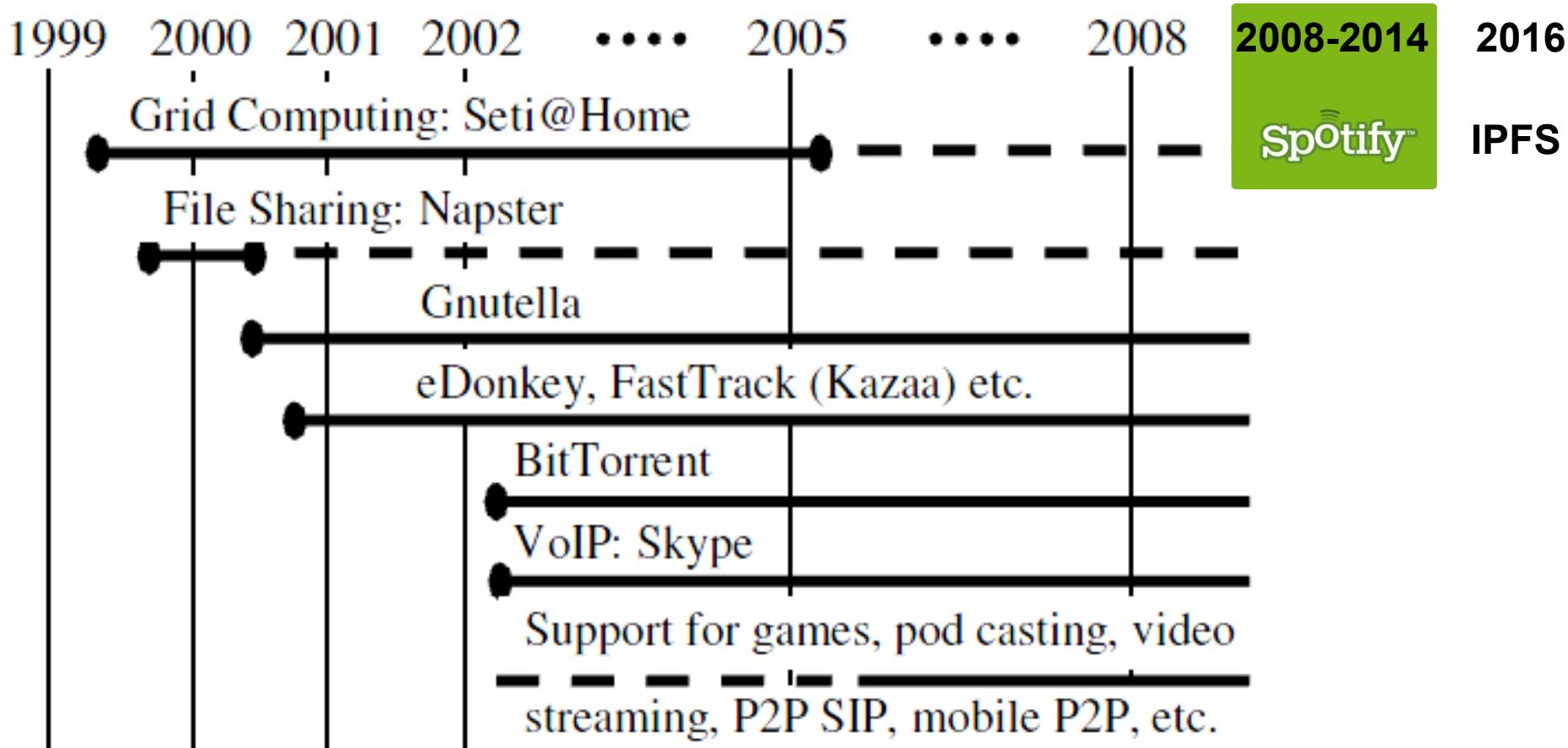
- **Efficient use of resources**
  - Unused bandwidth, storage resources, processing power available to the network edges
- **Scalability**
  - Without centralized information, without bottleneck (communication and computing)
  - Integration of resources is done naturally during the system use
- **Reliability**
  - The existence of copies of data
  - Geographical distribution
  - No more "single point of failure"
- **Easy administration**
  - The nodes are self-organizing
  - Increased fault tolerance and load balancing
  - Increased autonomy
- **Anonymity**
  - Hard to accomplish in a centralized environment
- **Dynamism**
  - Dynamic environment
  - Ad-hoc collaboration and communication

# Disadvantages / Problems

- P2P architectures are probabilistic
  - Unpredictable resource location
  - Resources are volatile
- Nonexistence of centralized control
  - Issues requiring an authority on applications, content and users
  - Difficulties in detecting and identifying users (anti-social aspects)
- Encouraging the use of P2P systems abusive and illegal purposes (e.g. copyright of digital content)
  - Lack of trust in trades, business
  - Security issues (pending future)

# Evolution...

## Timeline of Popular Peer-to-Peer Protocols



# Types of Applications

- **Communication & collaboration**

Systems that provide an infrastructure to facilitate direct communication & collaboration, often in real time between nodes

Conversational systems (chat, instant messaging):

IRC (Internet Relay Chat), ICQ (1996), YM !, MSN Messenger,

Skype, P2P multicast systems (e.g. Cirrus - Adobe Flash

<http://labs.adobe.com/technologies/cirrus/>; WebRTC) ...

- **Distributed Computing**

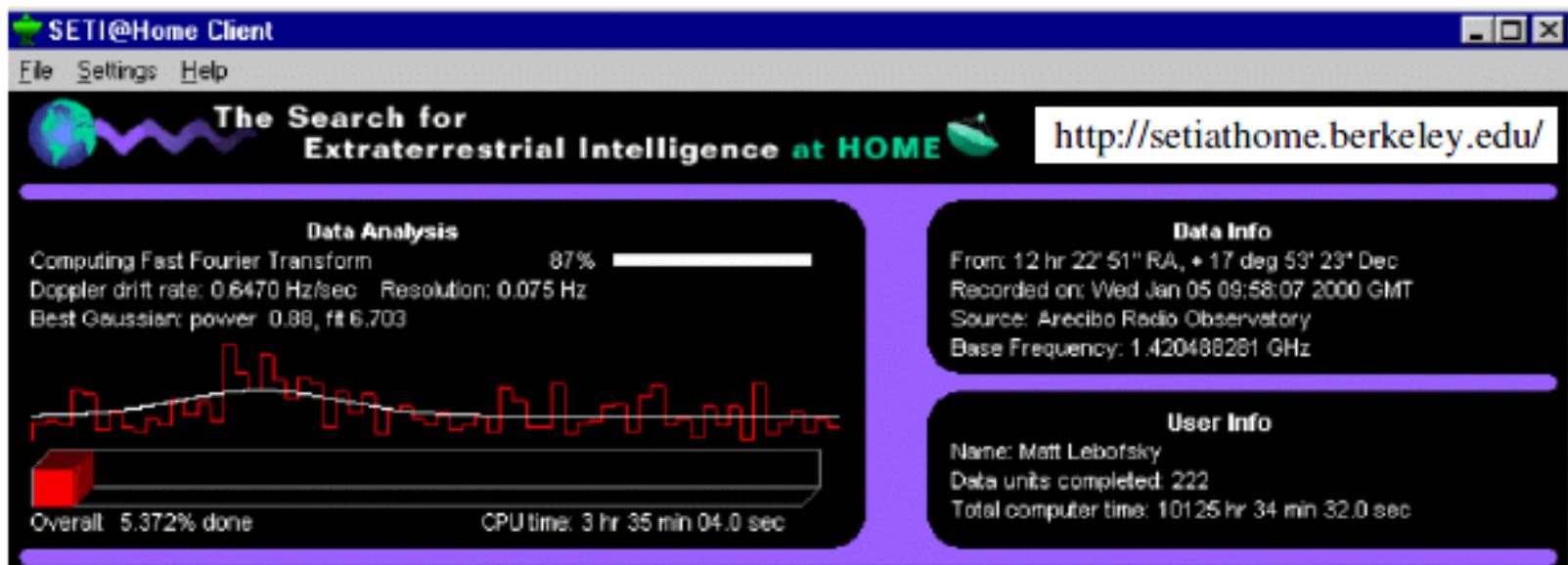
- Systems that use computer power of available nodes (processor cycles)

- Solving problems through *divide-et-impera*: SETI@home (*Search for Extra-Terrestrial Intelligence*-Berkeley), genome@home

- P2P network is a kind of a computational Grid

# Types of Applications | Distributed Computing - Example

## SETI@Home: A Public-Resource Computing Experiment



- Radio telescope signal analysis has insatiable appetite for computing power
- Usage of computers in homes and offices around the world has provided unprecedented computing power
- Grid computing application via peer-to-peer approach under central control

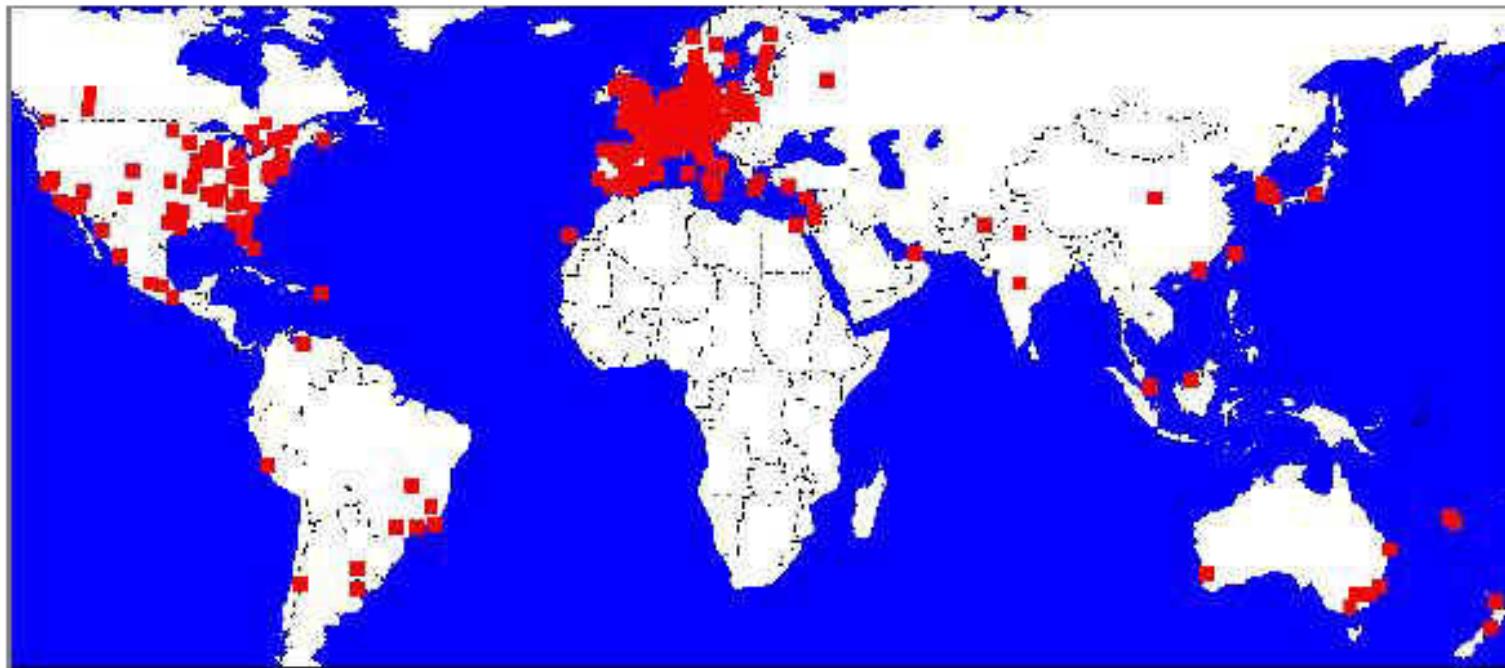
# Types of Applications

- **Storage systems (databases)**
  - Design of distributed database infrastructure based on P2P
    - PIER – a scalable engine for distributed query (<http://pier.cs.berkeley.edu/>)
    - Edutella –open-source project for queries and *meta-data* storage (P2P for Semantic Web)
- **Distribution of digital content**
  - Systems & Infrastructure to share digital resources (multimedia and other data) among users
    - File sharing applications (e.g. Napster, Gnutella, KaZaA, Freenet, BitTorrent, eDonkey etc.)
    - Distributed storage media for publishing, organizing, indexing, searching and retrieving data in secure & efficient manner (PAST, Chord, Groove, Mnemosyne, Avalanche,...)

# Types of Applications

- **Distribution of digital content | Example**

P2P File-Sharing: Fast distribution of large files



Example: Harry Potter III early propagation after 2 hours on May 28th 2004 (Source: [www.itic.ca/DIC/News/archive.html](http://www.itic.ca/DIC/News/archive.html))

# Types of Applications

- **Distribution of content through P2P**
  - P2P systems for “interchange files”
    - Nodes transfer files at a time
    - It offers facilities for the achievement of P2P networks and search & transfer files between nodes
    - Security, availability and persistence are not supported
    - Examples: Napster, KaZaA, Gnutella

# Types of Applications

- **Distribution of content through P2P**
  - P2P systems for publishing & content storage
    - Users can publish, store and distribute digital content, based on access rights (privileges)
      - Focuses on security and persistence
      - Some systems offer facilities regarding collaboration between users
      - Example: Scan, Groove, Freenet, MojoNation, Tangler

# Types of Applications

- **Distribution of content through P2P**
  - Infrastructure for:
    - Routing & Localization:  
Chord, Can, Pastry, Tapestry, Kademila
    - Anonymity:  
Onion Routing, ZeroKnowledge, Freedom, Tarzan
    - Reputation management:  
EigenTrust, PeerTrust

# Infrastruct.(Routing & Localization)

- Localization and routing mechanism that can be adopted depends on:
  - Topology
  - Structure
  - The degree of centralization of the *overlay network*

# Infrastruct.(Routing & Localization)

- Aspects regarding centralization
  - Pure decentralized architectures: all nodes perform exactly the same activities, by playing roles of servers and clients simultaneously, without the benefit of a central coordination
    - Nodes are called servents (SERVers + clieENTS)

# Infrastruct.(Routing & Localization)

- **Aspects regarding centralization**
  - **Partially centralized architectures**: some nodes have a more important role (e.g., storing local indexes for shared folders)
    - Nodes become **super-nodes** in accordance with the policies of each P2P system
    - Super-node role is determined dynamically
  - **Hybrid decentralized architecture**: there is a central server enabling the interaction between nodes, keeping catalogs with metadata of files
    - Servers can identify and verify the storage nodes
    - The systems are called ***broker mediated***

# Infrastruct.(Routing & Localization)

- Aspects regarding network structure:
  - **Unstructured**: placing content is completely independent of overlay network topology
    - The content must be located
    - Search strategies by "brute force": flooding the network - requests propagated via BFS / DFS
    - More sophisticated strategies: random path , probabilistic methods etc.
  - **Loosely structured**: although the content location is not completely specified, it is affected by routing
    - Category located between structured and unstructured networks

# Infrastruct.(Routing & Localization)

- **Aspects regarding network structure:**
  - **Structured:** topology is controlled, files are placed in precise locations
    - A mapping between the content (file identifier) and the location (node address) is performed
      - Like a distributed routing table
    - *exact-match queries* can be performed in a scalable way
    - The structure used to guide message routing is difficult to maintain for transient nodes (with high rates of attachment and disconnection from the network)

# Infrastruct.(Routing & Localization)

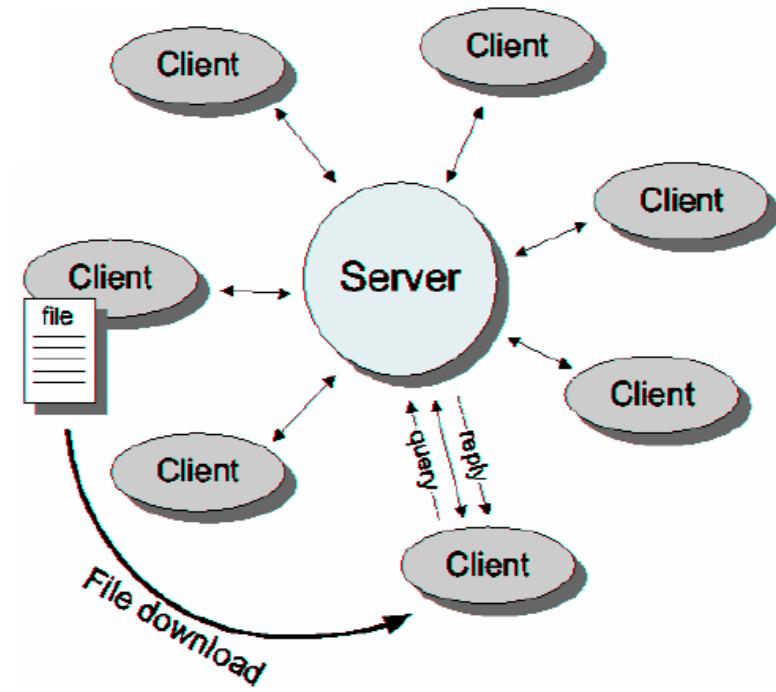
	Centralizare		
	Hibridă	Partială	Absentă
Nestruc-turată	Napster Publius	KaZaA Morpheus Gnutella Edutella	Gnutella FreeHaven
Infrastruc-structurată			Chord, CAN Tapestry, Pastry
Sisteme structurate			OceanStore Scan, PAST Kademlia Tarzan

# Unstructured Architectures

## Decentralized and hybrid

- Each client computer stores content (files) shared (s)
- The central server keeps a table with registered users (IP, bandwidth, ...) + a table with the list of files for user & meta-data

Example: **Napster, Publius**



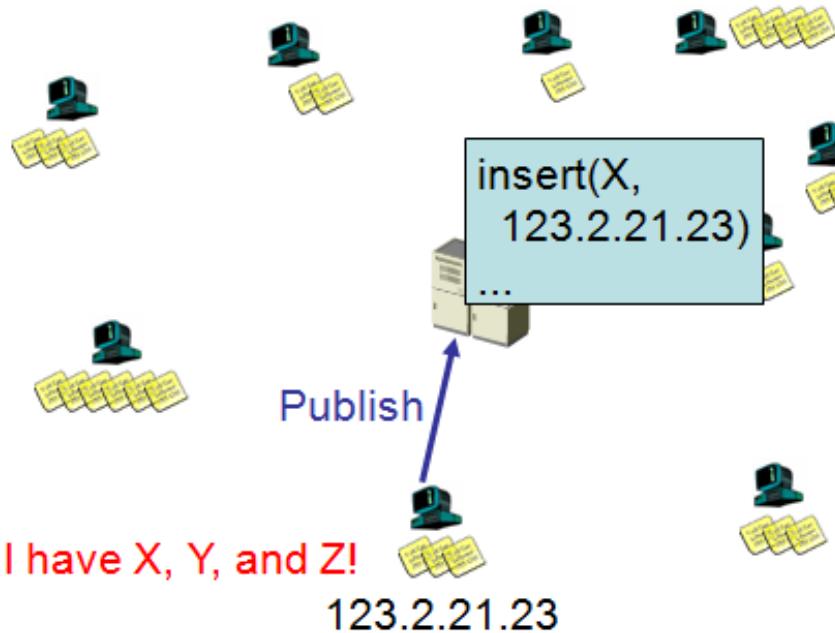
# Unstructured Architectures

## Napster

- 1999: Sean Fanning release Napster
- It has reached at 1.5 million concurrent users
- Centralized database - operations:
  - **Join**: the client contacts the central server (via TCP)
  - **Publish**: reporting a list of files to the central server
  - **Search**: query server => it returns one that stores the requested file
  - **Fetch**: take the file directly from peer (the one with the best transfer rate)
- July 2001: Napster was closed

# Unstructured Architectures

Napster: Publish



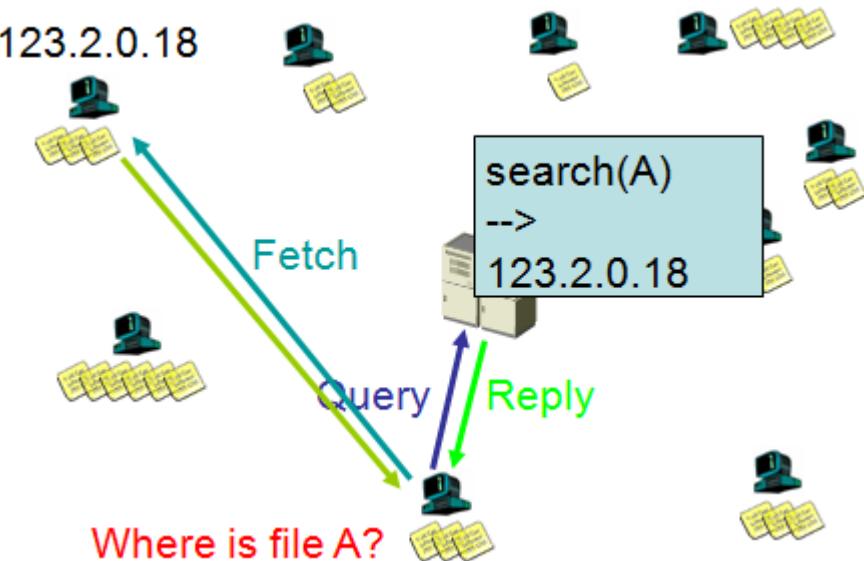
Discussions:

The server does all processing

We have "single point of failure"

Scalability problems, some systems do not allow adding other servers  
(the list of available servers is static)

Napster: Search



# Unstructured Architectures

## Pure Decentralized

- An overlay network is built using routing mechanism based on IP
- There is no central coordination
- Users are connecting via an application that has a dual role – *servent*
- Communication between *servents* is based on a protocol at the application level, with 4 types of messages:
  - Ping – requires a node to announce himself
  - Pong – reply to the *ping* message(IP, port, file size)
  - Query – search request (search string + minimum speed of transfer)
  - Query hints – response (IP, port, speed, file length, file index)

# Unstructured Architectures

## Pure Decentralized

- Searching is done by flooding (flooding)
  - If you don't have the desired file, ask the n neighbors
  - If they do not have the file, they will ask their neighbors in maximum m hops
  - On the way back answers will be returned (not the files content)
- Each message has a TTL attached
- Example: Gnutella

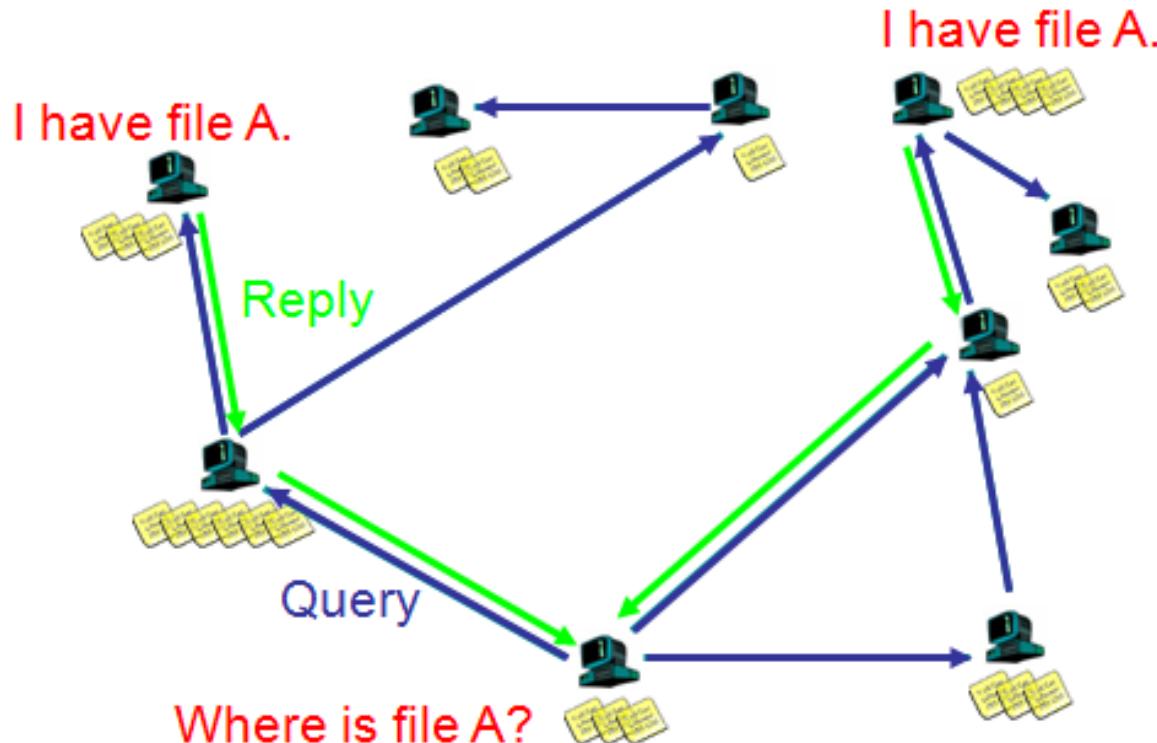
# Unstructured Architectures

## Gnutella

- 2000: L. Frankel and T. Pepper(Nullsoft) launches Gnutella
- Client application comes up: Bearshare, Morpheus, LimeWire
- *Query Flooding:*
  - **Join:** at entrance, the client contacts a few nodes that became his neighbors
  - **Publish:** Not required
  - **Search:** asks neighbors, who ask their neighbors ...
    - There is a TTL limiting the spread
  - **Fetch:** take the file directly from peer

# Unstructured Architectures

## Gnutella



Issues:

- Search times is...  $O(?)$
- The nodes leave often => unstable network

# Unstructured Architectures

## Partially centralized

- Use the concept of **super-node**: that performs various activities in a P2P network (*indexing, caching*)
- The nodes are automatically chosen as super-nodes if they have enough bandwidth and computational power
- All requests are sent initially to super-nodes
- Advantages: resource discovery time is less
- Example: **KaZaA**

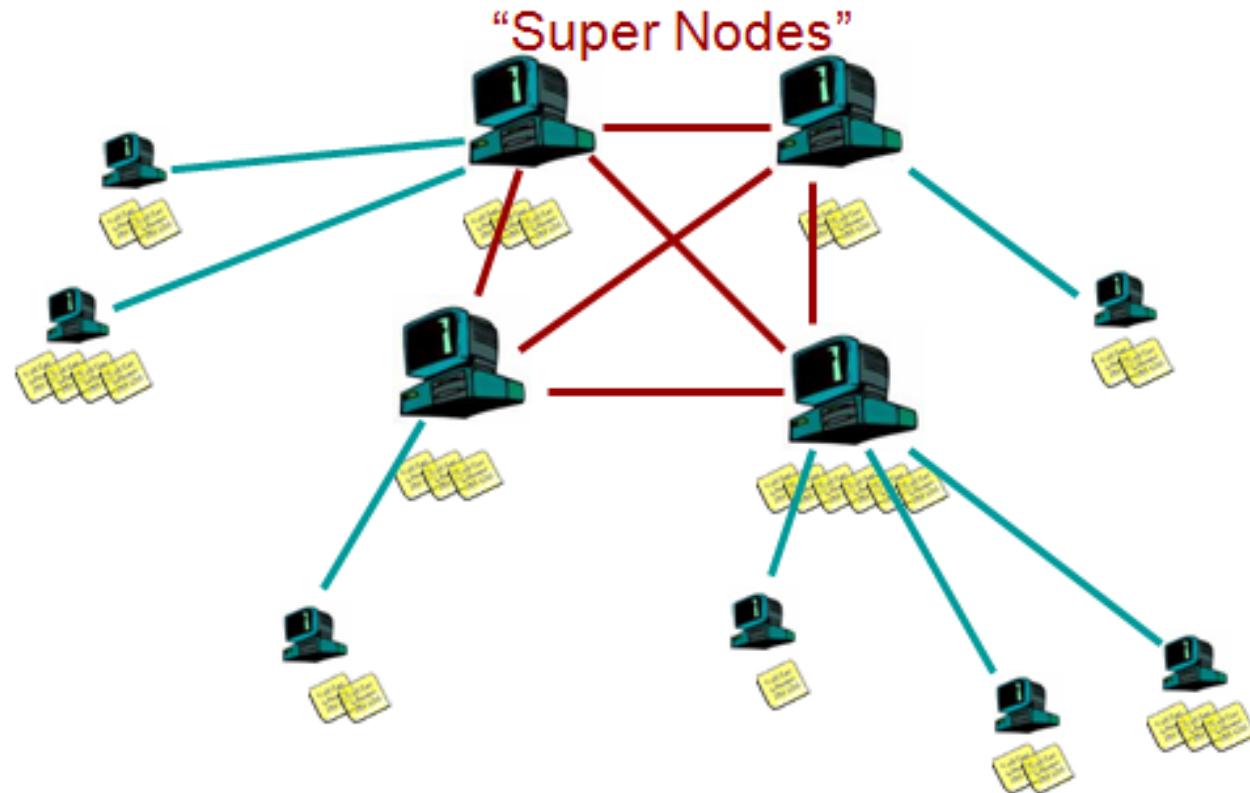
# Unstructured Architectures

## KaZaA

- 2001: KaZaA is launched
- Client application comes up: Morpheus, giFT
- It utilizes a mechanism of “*smart*” *query flooding*:
  - **Join**: at entrance, the client contacts a *super-node* (it can become super-node at a time)
  - **Publish**: sends list of files to super-node
  - **Search**: send a query to the *super-node*, and *super-nodes* interrogate each other
  - **Fetch**: take the file directly from the *peer(s)*; the file can be obtained simultaneously from multiple peers

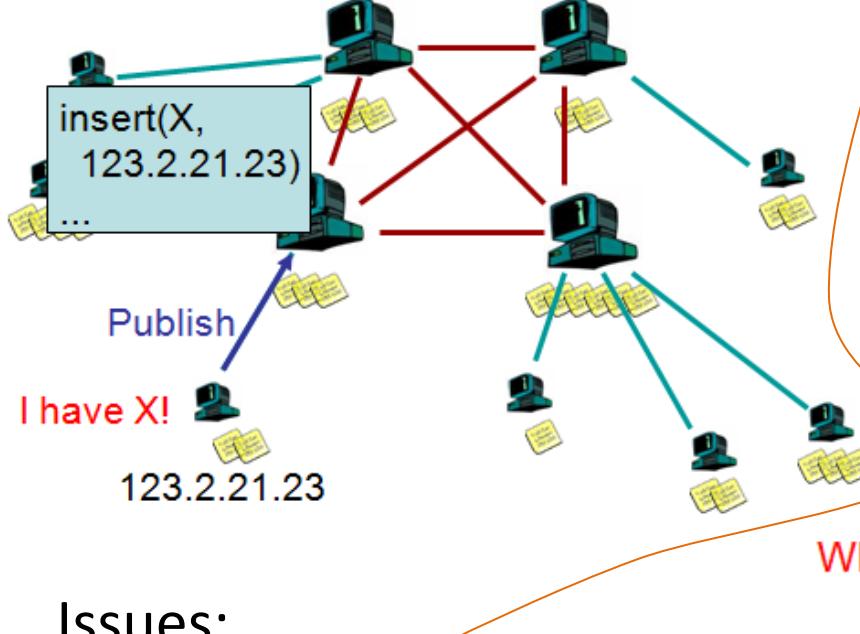
# Unstructured Architectures

## KaZaA: Network design

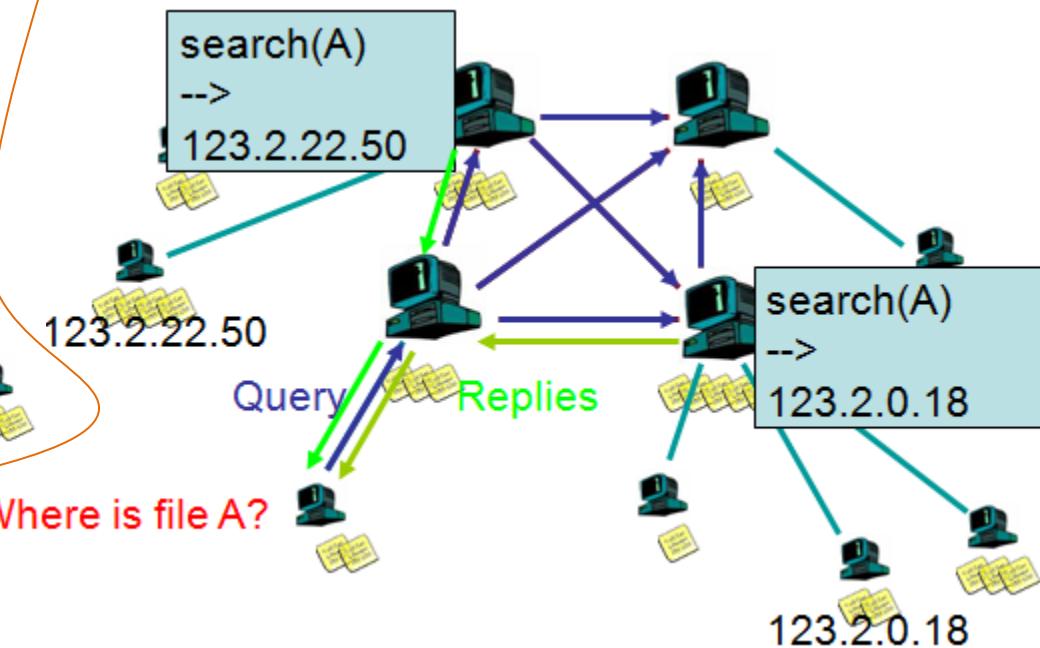


# Unstructured Architectures

## KaZaA: Inserting Files



## KaZaA: Files Search



### Issues:

- Behaviour similar to Gnutella, but more efficient
- There is no guarantee on the search time or on the search area

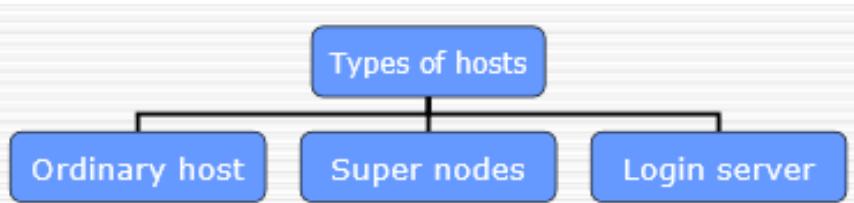
# Unstructured Architectures

## Partially centralized

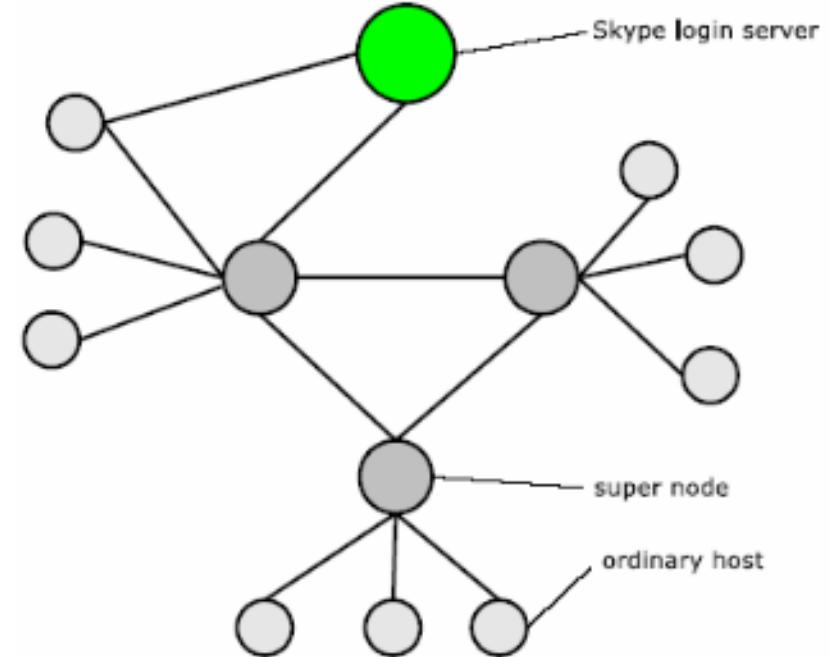
- KaZaA software is a proprietary one
- P2P control data are encrypted
- The messages use HTTP
- A node is either a super-node, or assigned to a super-node
- A super-node has 100-150 child-nodes
- A network can have ~ 30,000 super-nodes
- Each super node has TCP connections with 30-50 super-nodes
- For each file meta-data are maintained (name, size, content hash, file descriptor)
- *The content hash* is used to find another copy of a partially transferred file
- The version without *spyware* and *pop-ups*: **KaZaA-lite**

# Unstructured Architectures

## Skype



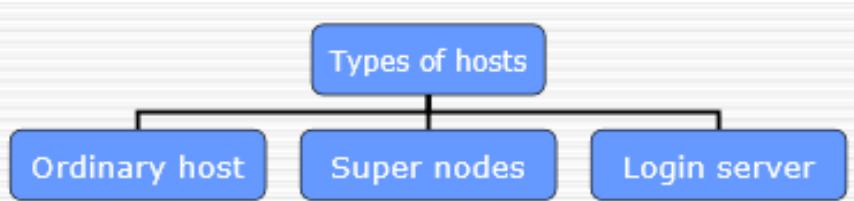
- The first P2P telephony network based on IP
- from June 2014, Microsoft announced the incompatibility with the previous Skype protocol
- Uses Microsoft Notification Protocol 24 ( first use -> MSN Messenger in 1999)
- architecture was similar to KaZaA



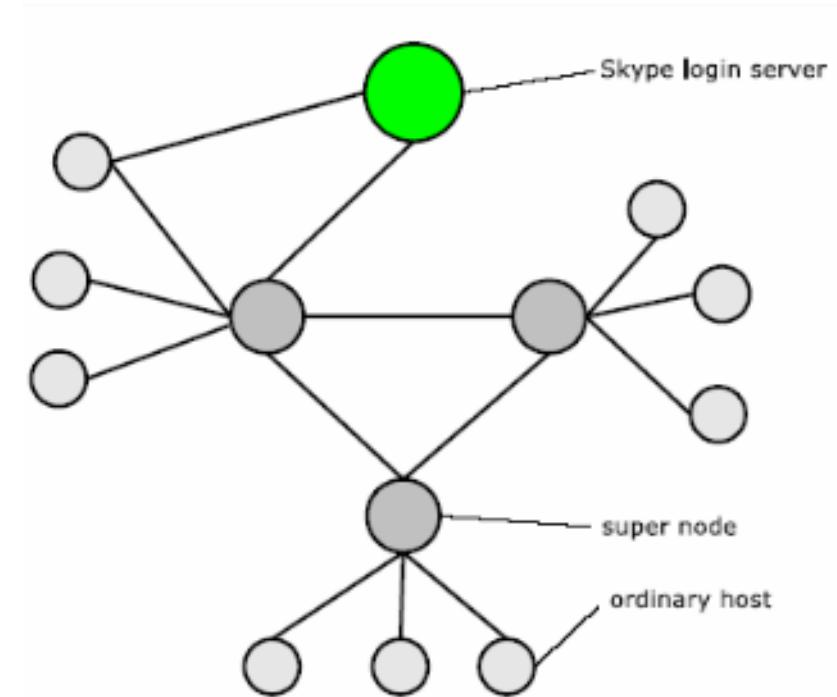
[http://www1.cs.columbia.edu/~salman/publications/skype1\\_4.pdf](http://www1.cs.columbia.edu/~salman/publications/skype1_4.pdf)

# Unstructured Architectures

## Skype



- Each client maintains a *host cache* with IP addresses and the port numbers associated with the accessible super-nodes
- Any customer with bandwidth (and without firewall or NAT restrictions) could become a super-node
- from 2012, Microsoft began super-nodes hosting servers in its data centers



[http://en.wikipedia.org/wiki/PRISM\\_%28surveillance\\_program%29](http://en.wikipedia.org/wiki/PRISM_%28surveillance_program%29)

# Unstructured Architectures

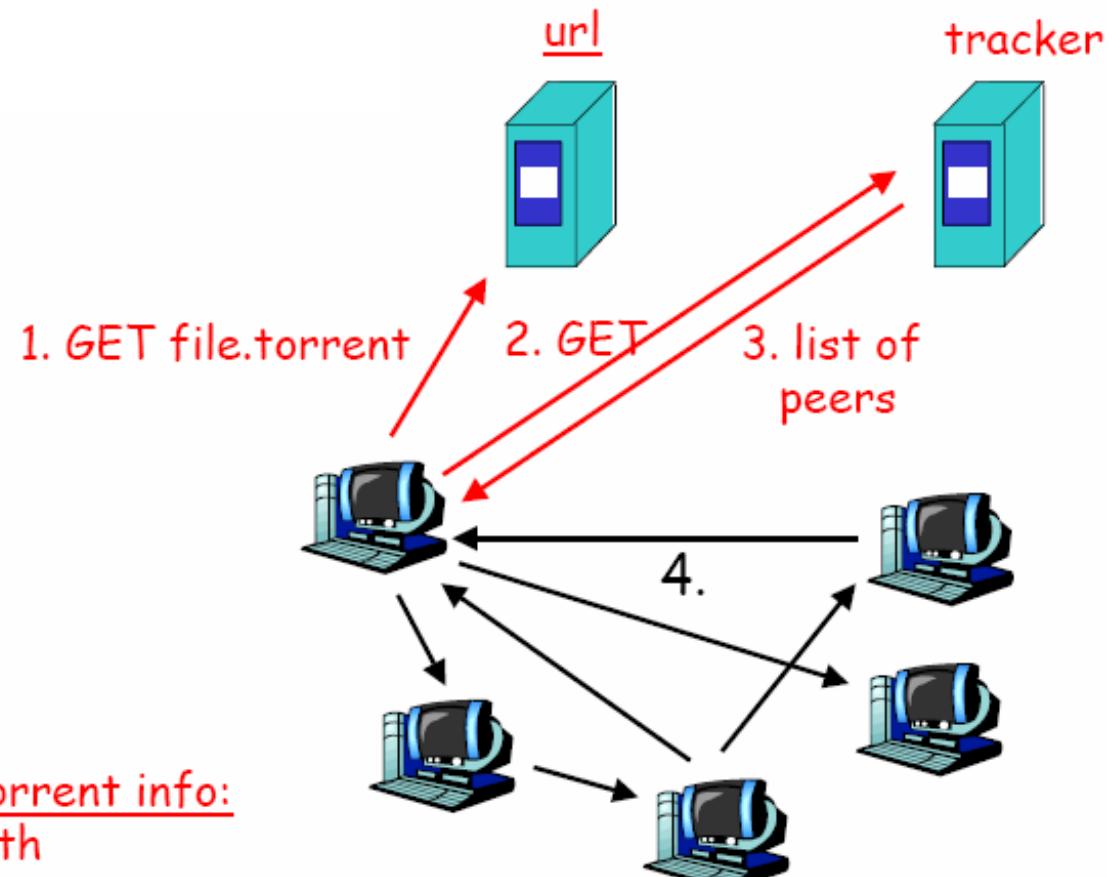
## Partially centralized

- If a file is found on multiple nodes, the transfer can be done in parallel
  - The same copies are identified via *content hash*
- Different chunks of the file is transferred from different nodes
- For interrupted transfers, an automatic recovery is performed
- Example: **BitTorrent**
  - In 2002, B. Cohen released BitTorrent
  - The focus was over *efficient fetching* and not on *searching*
  - Supporters since they have appeared
    - Blizzard Entertainment use BitTorrent to distribute beta versions of new games

# Unstructured Architectures

Partially centralized

BitTorrent - arhitecture



# Unstructured Architectures

## Partially centralized

### BitTorrent

- It is based on *swarming* mechanism:
  - **Join**: contact a centralized server (*tracker*) and get a list of *peers*
  - **Publish**: running a *tracker* server
  - **Search**: e.g. uses Google to find a *tracker* for the desired file
  - **Fetch**: Take pieces of files from peers;
- Obs. The difference from Napster
  - *File Chunk Download*
  - Using the “tit-for-tat” strategy: A may *download* from other nodes, then A should allow download from it (*free-rider problem*)

# Structured Architectures

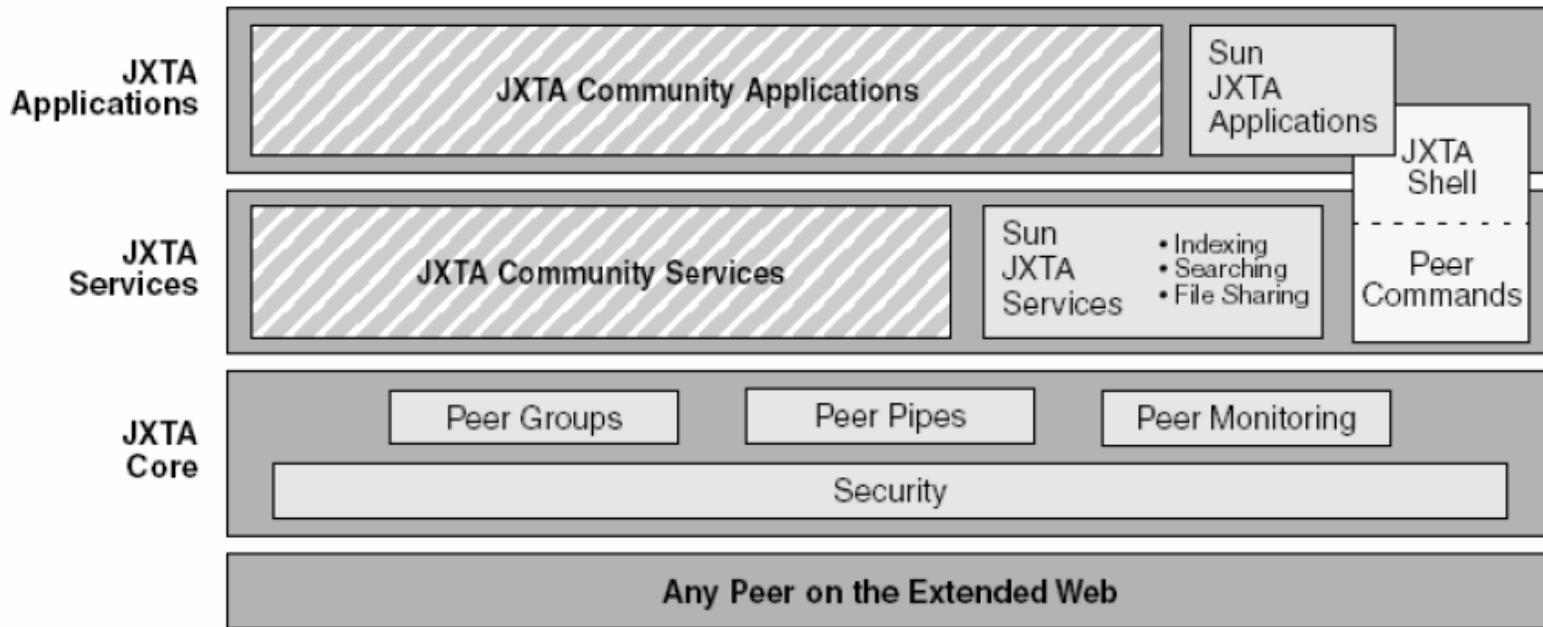
- Represents academic solution for P2P
- Scope:
  - Successful search
  - Search time is performed in known boundaries
  - proven scalability
- Approach: **DHT (Distributed Hash Table)**
  - Pairs (key, value) are stored
    - Key – file names
    - Value – file content or a pointer to a location
  - Each *peer* stores a set (key, value)
  - *Operations: find the node responsible for a Key*
    - Mapping key – node
    - Efficient routing to *insert/lookup/delete* operations associated with this node
    - A wide fluctuation of nodes is allowed

# Structured Architectures

- **Implementations**
  - Chord [MIT]
  - Pastry [Microsoft Research UK, Rice University]
  - Tapestry [UC Berkeley]
  - Content Addressable Network (CAN) [UC Berkeley]
  - SkipNet [Microsoft Research US, Univ. of Washington]
  - Kademlia [New York University]
  - Viceroy [Israel, UC Berkeley]
  - P-Grid [EPFL Switzerland]

# Instruments

- JXTA – [www.jxta.org](http://www.jxta.org)
  - Development environment for P2P systems & applications
  - Based on Java - available under open source



# Instruments

- P2P – framework for Android  
<https://code.google.com/p/p2p-communication-framework-for-android/>
- p2psim – simulator for p2p protocol  
<http://pdos.csail.mit.edu/p2psim/>
- Instruments and protocols for P2P:  
[http://en.wikibooks.org/wiki/The\\_World\\_of\\_Peer-to-Peer\\_%28P2P%29/Networks\\_and\\_Proocols/Other\\_Software\\_Implementations](http://en.wikibooks.org/wiki/The_World_of_Peer-to-Peer_%28P2P%29/Networks_and_Proocols/Other_Software_Implementations)
- “ IPFS is the Distributed Web” - <https://ipfs.io/>

# IPFS

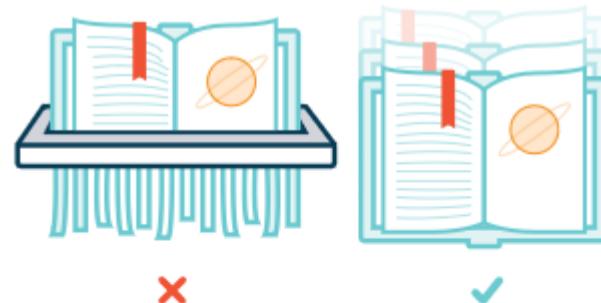
- “ IPFS is the Distributed Web” - <https://ipfs.io/>
  - A peer-to-peer hypermedia protocol to make the web faster, safer, and more open



## HTTP is inefficient and expensive

HTTP downloads a file from a single computer at a time, instead of getting pieces from multiple computers simultaneously. With video delivery, a P2P approach could save 60% in bandwidth costs.

IPFS makes it possible to distribute high volumes of data with high efficiency. And zero duplication means savings in storage.



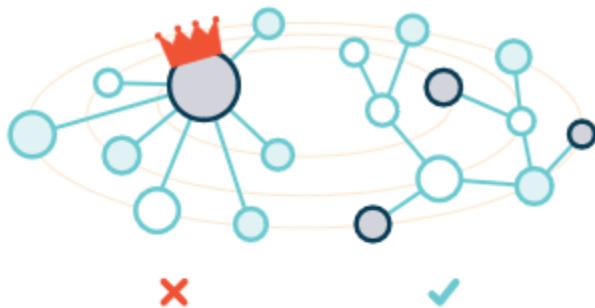
## Humanity's history is deleted daily

The average lifespan of a web page is 100 days. Remember GeoCities? The web doesn't anymore. It's not good enough for the primary medium of our era to be so fragile.

IPFS provides historic versioning (like git) and makes it simple to set up resilient networks for mirroring of data.

# IPFS

- “ IPFS is the Distributed Web” - <https://ipfs.io/>
  - A peer-to-peer hypermedia protocol to make the web faster, safer, and more open



## The web's centralization limits opportunity

The Internet has been one of the great equalizers in human history and a real accelerator of innovation. But the increasing consolidation of control is a threat to that.

IPFS remains true to the original vision of the open and flat web, but delivers the technology which makes that vision a reality.



## Our apps are addicted to the backbone

Developing world. Offline. Natural disasters. Intermittent connections. All trivial compared to interplanetary networking. The networks we're using are so 20th Century. We can do better.

IPFS powers the creation of diversely resilient networks which enable persistent availability with or without Internet backbone connectivity.

# IPFS

- “ IPFS is the Distributed Web” - <https://ipfs.io/>

Let's take a look at what happens when you add files to IPFS:



Each file and all of the **blocks within it** are given a **unique fingerprint** called a **cryptographic hash**.



Each **network node** stores only content it is interested in, and some indexing information that helps figure out who is storing what.



IPFS **removes duplications** across the network and tracks **version history** for every file.



When **looking up files**, you're asking the network to find nodes storing the content behind a unique hash.



Every file can be found by **human-readable names** using a decentralized naming system called **IPNS**.

- <https://github.com/ipfs/papers/raw/master/ipfs-cap2pfs/ipfs-p2p-file-system.pdf>

# Statistics

## Global Consumer Internet Traffic 2005–2011

	2005	2006	2007	2008	2009	2010	2011
<b>By Sub-Segment (terabytes per month)</b>							
Web, e-mail, file transfer	362,084	505,996	692,812	948,425	1,233,172	1,603,615	2,756,415
P2P	1,060,226	1,329,770	1,772,403	2,379,025	3,111,891	4,040,403	5,269,360
Gaming	66,844	91,943	133,367	188,680	250,574	318,212	386,832
Video Communications	11,629	15,575	24,932	36,638	47,173	66,101	92,453
VoIP	10,965	23,035	39,339	57,653	75,575	92,815	110,456
Internet Video to PC	53,074	174,427	484,027	838,154	1,232,461	1,726,114	2,331,908
Internet Video to TV	0	12,727	110,692	353,095	620,197	936,580	1,342,482
<b>By Geography (TB per month)</b>							
North America	534,236	618,765	917,365	1,287,026	1,698,700	2,242,841	2,861,772
Western Europe	334,600	505,329	814,015	1,281,041	1,856,310	2,515,070	3,458,721
Asia Pacific	565,782	819,072	1,201,277	1,742,834	2,315,755	3,049,294	4,663,774
Japan	60,080	98,747	147,733	223,120	319,788	436,057	556,631
Latin America	19,917	33,755	57,083	90,765	130,466	189,992	268,559
Central Eastern Europe	40,773	59,097	86,196	122,272	165,387	222,895	294,901
Middle East and Africa	9,435	18,708	33,904	54,613	84,637	127,689	185,549

# Statistics

## Legend:

Web, E-mail, and File Transfer – includes Web, e-mail, instant messaging, newsgroups, and file transfer (excluding P2P and commercial file transfer such as iTunes)

P2P – includes peer-to-peer traffic from all recognized P2P systems such as BitTorrent, eDonkey, etc.

Gaming – includes casual online gaming, networked console gaming, and multiplayer virtual world gaming

Video Communications – includes PC-based video calling, Webcam viewing, and Web-based video monitoring

VoIP – includes traffic from retail VoIP services and PC-based VoIP, but excludes wholesale VoIP transport

Internet Video to PC – free or pay TV or VoD viewed on a PC, excludes P2P video file downloads

Internet Video to TV – free or pay TV or VoD delivered via Internet but viewed on a TV screen using a STB or media gateway

Global Consumer Peer-to-Peer Traffic 2005–2011

Consumer Peer-to-Peer Traffic 2005–2011							
	2005	2006	2007	2008	2009	2010	2011
By Geography (TB per month)							
North America	381,746	378,538	462,356	560,817	673,083	852,483	1,080,979
Western Europe	223,519	304,988	411,057	540,032	757,818	991,817	1,330,885
Asia Pacific	391,235	550,664	762,276	1,074,759	1,401,028	1,811,094	2,327,648
Japan	28,621	42,883	58,463	87,446	117,967	154,868	206,803
Latin America	8,732	14,358	23,247	37,284	53,587	80,043	117,731
Central Eastern Europe	22,075	31,009	43,117	59,928	79,589	106,543	141,282
Middle East and Africa	4,297	7,329	11,886	18,759	28,819	43,553	64,033
Total (TB per month)							
Peer-to-Peer Traffic	1,060,226	1,329,770	1,772,403	2,379,025	3,111,891	4,040,403	5,269,360

Table 10. Global Consumer File-Sharing Traffic, 2015–2020

Consumer File Sharing, 2015–2020							
	2015	2016	2017	2018	2019	2020	CAGR 2015–2020
<b>By Network (PB per Month)</b>							
Fixed	5,942	5,909	5,829	5,713	5,616	5,939	0%
Mobile	22	28	29	29	29	35	9%
<b>By Subsegment (PB per Month)</b>							
P2P file transfer	4,798	4,550	4,224	3,840	3,438	3,633	-5%
Other file transfer	1,166	1,388	1,634	1,902	2,207	2,340	15%
<b>By Geography (PB per Month)</b>							
Asia Pacific	2,335	2,269	2,186	2,098	2,004	2,098	-2%
North America	1,015	1,137	1,260	1,371	1,478	1,576	9%
Western Europe	1,124	1,105	1,096	1,075	1,053	1,131	0%
Central and Eastern Europe	829	763	691	646	621	666	-4%
Latin America	554	573	558	514	454	463	-4%
Middle East and Africa	107	91	68	39	34	39	-18%
<b>Total (PB per Month)</b>	• <a href="http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/complete-white-paper-c11-481360.html">http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/complete-white-paper-c11-481360.html</a>						

**File Sharing**

This category includes traffic from P2P applications such as BitTorrent and eDonkey, as well as web-based file sharing. Note that a large portion of P2P traffic is due to the exchange of video files, so a total view of the impact of video on the network should count P2P video traffic in addition to the traffic counted in the Internet video-to-PC and Internet video-to-TV categories. Table 10 shows the forecast for consumer P2P traffic from 2015 to 2020.

Note that the P2P category is limited to traditional file exchange and does not include commercial video-streaming applications that are delivered through P2P, such as PPStream or PPLive.

# Bibliography

- P2P Networking and Applications, John F. Buford, Heather Yu, Eng Keong Lua ,2009, Elsevier
- <http://mtc.sri.com/Conficker/P2P/>
- [http://www.cisco.com/en/US/docs/cable/serv\\_exch/serv\\_control/broadband\\_app/protocol\\_ref\\_guide/01\\_p2p.pdf](http://www.cisco.com/en/US/docs/cable/serv_exch/serv_control/broadband_app/protocol_ref_guide/01_p2p.pdf)
- <http://pdos.csail.mit.edu/p2psim/>
- Statistici: [http://www.sandvine.com/news/pr\\_detail.asp?ID=312](http://www.sandvine.com/news/pr_detail.asp?ID=312)
- Statistici: [http://www.hbtf.org/files/cisco\\_IPforecast.pdf](http://www.hbtf.org/files/cisco_IPforecast.pdf)
- [http://en.wikibooks.org/wiki/The\\_World\\_of\\_Peer-to-Peer\\_%28P2P%29/Networks\\_and\\_Proocols/Other\\_Software\\_Implementations](http://en.wikibooks.org/wiki/The_World_of_Peer-to-Peer_%28P2P%29/Networks_and_Proocols/Other_Software_Implementations)
- <https://www.kirsle.net/blog/entry/skype-switched-to-the-msn-messenger-protocol>

# Summary

- **Peer-to-peer(P2P) paradigm**
  - Preliminaries
  - Definitions
  - Characteristics
  - Application types
  - Infrastructures
  - Instruments



# Questions?

# Paradigma RPC

**Lenuta Alboiae**  
**adria@info.uaic.ro**



# Cuprins

- **Remote Procedure Call (RPC)**
  - Preliminarii
  - Caracterizare
  - XDR
  - Functionare
  - Implementari
  - Utilizari

# Preliminarii

- **Proiectarea aplicatiilor distribuite**
  - Orientata pe protocol – *socket*-uri
    - Se dezvolta protocolul, apoi aplicatiile care il implementeaza efectiv
  - Orientata pe functionalitate – **RPC**
    - Se creaza aplicatiile, dupa care se divid in componente si se adauga protocolul de comunicatie intre componente

# RPC | Caracterizare

- **Idee:** In loc de accesarea serviciilor la distanta prin trimiterea si primire de mesaje, clientul **apeleaza o procedura care va fi executata pe alta masina**
- **Efect:** RPC “ascunde” existenta retelei de program
  - Mecanismul de *message-passing* folosit in comunicarea in retea este ascuns fata de programator
  - Programatorul nu trebuie sa mai deschida o conexiune, sa citeasca sau sa scrie date, sa inchida conexiunea etc.
- Este un instrument de programare mai simplu decat interfata *socket* BSD

# RPC | Caracterizare

- O aplicatie RPC va consta dintr-un **client** si un **server**, serverul fiind localizat pe masina pe care se executa procedura
- La realizarea unui apel la distanta, parametrii procedurii sunt transferati prin retea catre aplicatia care executa procedura; dupa terminarea executiei procedurii rezultatele sunt transferate prin retea aplicatiei client
- Clientul si serverul → procese pe masini diferite

# RPC | Caracterizare

- RPC realizeaza comunicarea dintre client si server prin socket-uri TCP/IP (uzual, UDP), via doua interfete **stub (ciot)**
    - Obs. Pachetul RPC (*client stub si server stub | skeleton*) ascunde toate detaliile legate de programarea in retea
  - RPC implica urmatorii pasi:
    1. Clientul invoca procedura *remote*
      - Se apeleaza o procedura locala, numita *client stub* care impacheteaza argumentele intr-un mesaj si il trimitе nivelului transport, de unde este transferat la masina server *remote*
- 
- Marshalling (serializare)* = mecanism ce include codificarea argumentelor intr-un format standard si impachetarea lor intr-un mesaj

# RPC | Caracterizare

- RPC implica urmatorii pasi:

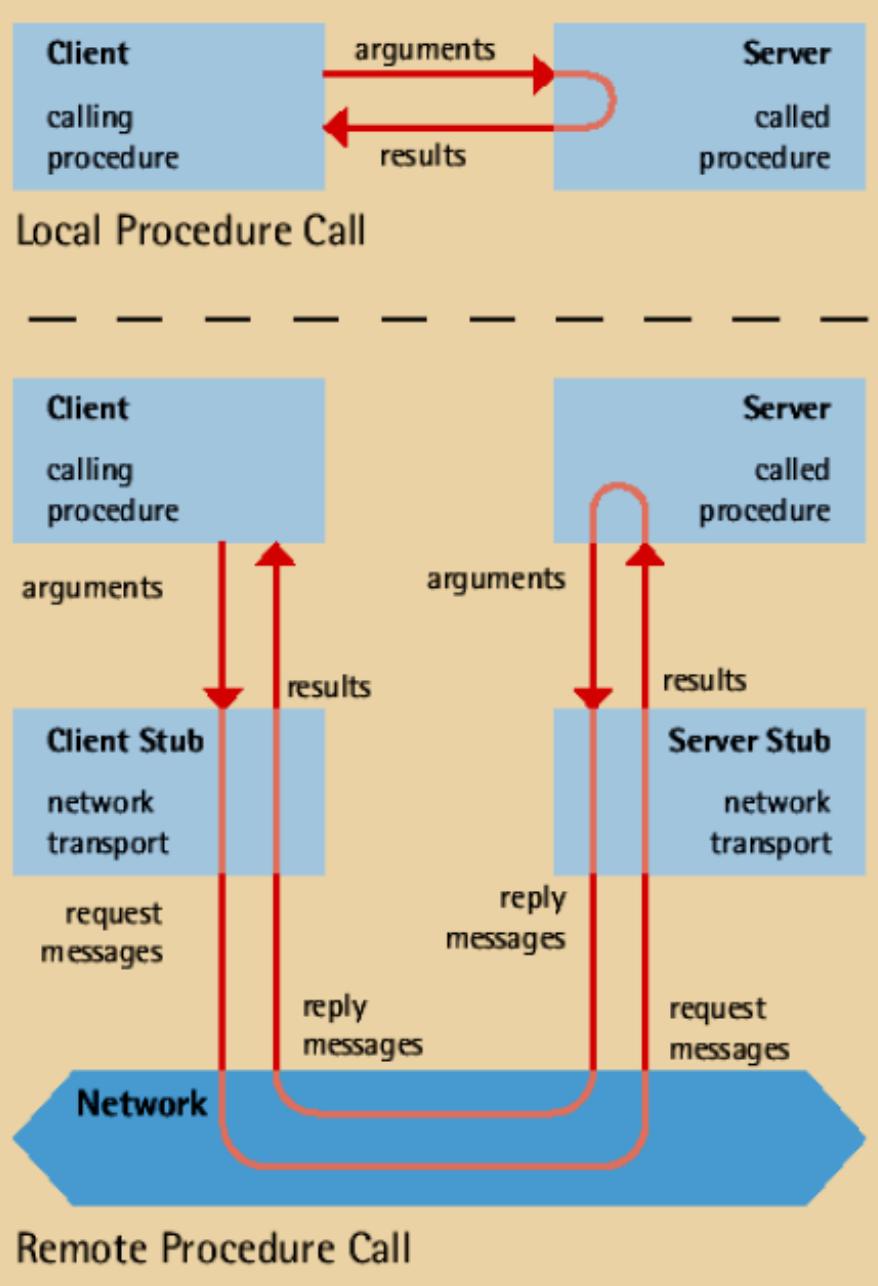
2. Server-ul:

- nivelul transport trimite mesajul catre *server stub*, care despacheteaza parametri si apeleaza functia dorita;
  - Dupa ce functia returneaza, *server stub* preia valorile intoarse le impacheteaza (*marshalling*) intr-un mesaj si le trimite la *client stub*
-  *un-marshalling (deserializare)* = decodificare

3. *Client stub* preia valorile primite si le returneaza aplicatiei client

# RPC | Caracterizare

- Interfetele care implementeaza protocolul RPC
- Diferente fata de apele locale:
  - Performanta poate fi afectata de timpul de transmisie
  - Tratarea erorilor este mai complexa
  - Locatia server-ului trebuie sa fie cunoscuta (Identificarea si accesarea procedurii la distanta)
  - Poate fi necesara autentificarea utilizatorilor



[Retele de calculatoare –  
curs 2007-2008, Sabin Buraga]

# RPC | Caracterizare

- Procedurile ciot se pot genera automat, dupa care se “leaga” de programele client si server
- Ciotul serverului asculta la un port si realizeaza invocarea rutinelor printr-o interfata de apel de proceduri locale
- Clientul si serverul vor comunica prin mesaje, printr-o reprezentare independenta de retea si de sistemul de operare:

**External Data Representation (XDR)**

# RPC | Caracterizare

- **External Data Representation (XDR)**

XDR defineste numeroase tipuri de date si modul lor de transmisie in mesajele RPC (RFC 1014)

- Tipuri uzuale:

- Preluate din C: int, unsigned int, float, double, void,...
- Suplimentare: string, fixed-length array, variable-length array, ...

- Functii de conversie (**rpc/xdr.h**)

- **xdrmem\_create()** – asociaza unei zone de memorie un flux de date RPC
- **xdr\_numetip()** – realizeaza conversia datelor

# RPC | Caracterizare

- External Data Representation (**XDR**)

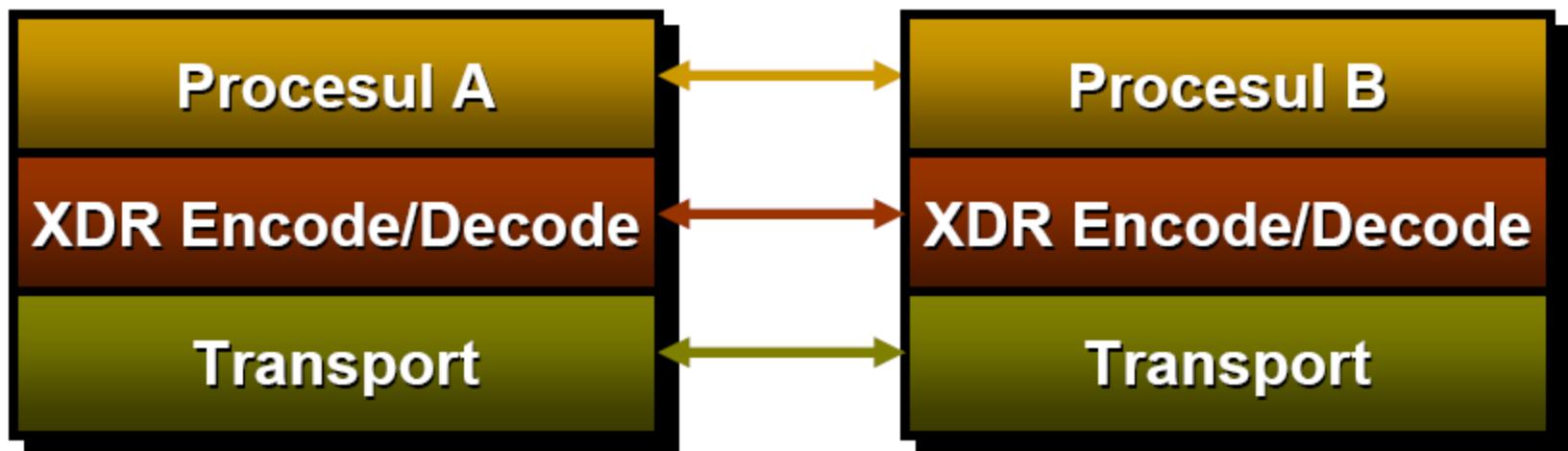
## Exemplu

```
#include <rpc/xdr.h>
#define BUFSIZE 400 /* lungimea zonei de memorie */
/* conversia unui intreg in format XDR */
...
XDR *xdrm; /* zona de memorie XDR */
char buf[BUFSIZE];
int intreg;
...
xdrmem_create (xdrm, buf, BUFSIZE, XDR_ENCODE);
...
intreg = 33;
xdr_int (xdrm, &intreg);
...
```

Inlocuit la celalalt capat al  
comunicatiei cu **XDR\_DECODE**

# RPC | Caracterizare

- **External Data Representation (XDR)**
  - Poate fi vazut ca nivel suplimentar intre nivelul transport si nivelul aplicatie
  - Asigura conversia simetrica a datelor client si server

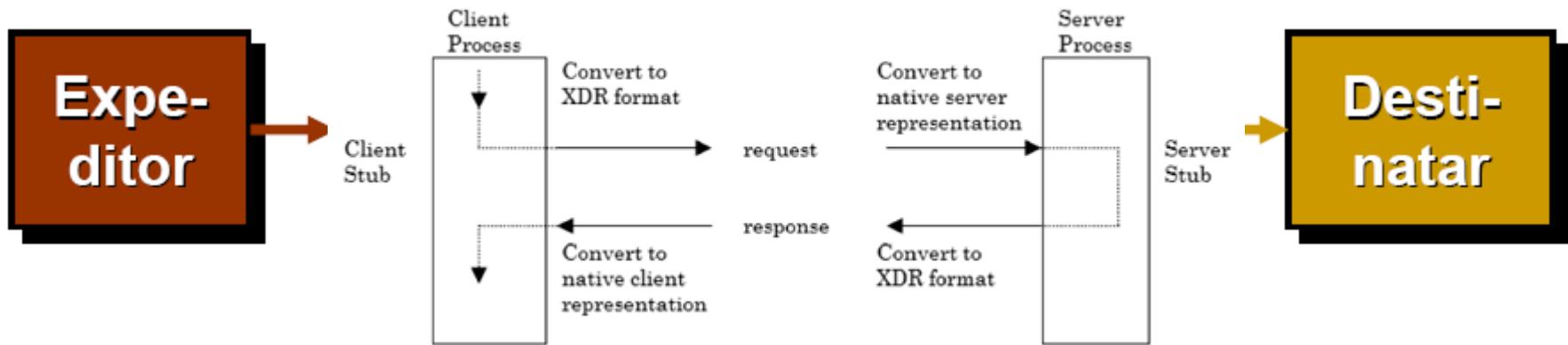


[Retele de calculatoare –  
curs 2007-2008, Sabin Buraga]

# RPC | Caracterizare

## External Data Representation (XDR)

- Activitatea de codificare/decodificare



- În prezent, poate fi înlocuit de reprezentări XML-RPC sau SOAP sau JSON-RPC (în contextul serviciilor Web)



vezi cursul de Tehnologii Web!

# RPC | Functionare

Context:

- Un serviciu de retea este identificat de portul la care exista un *daemon* asteptind cereri
- Programele server RPC folosesc porturi efemere



De unde stie clientul unde sa trimita cererea?

**Portmapper** = serviciu de retea responsabil cu asocierea de servicii la diferite porturi

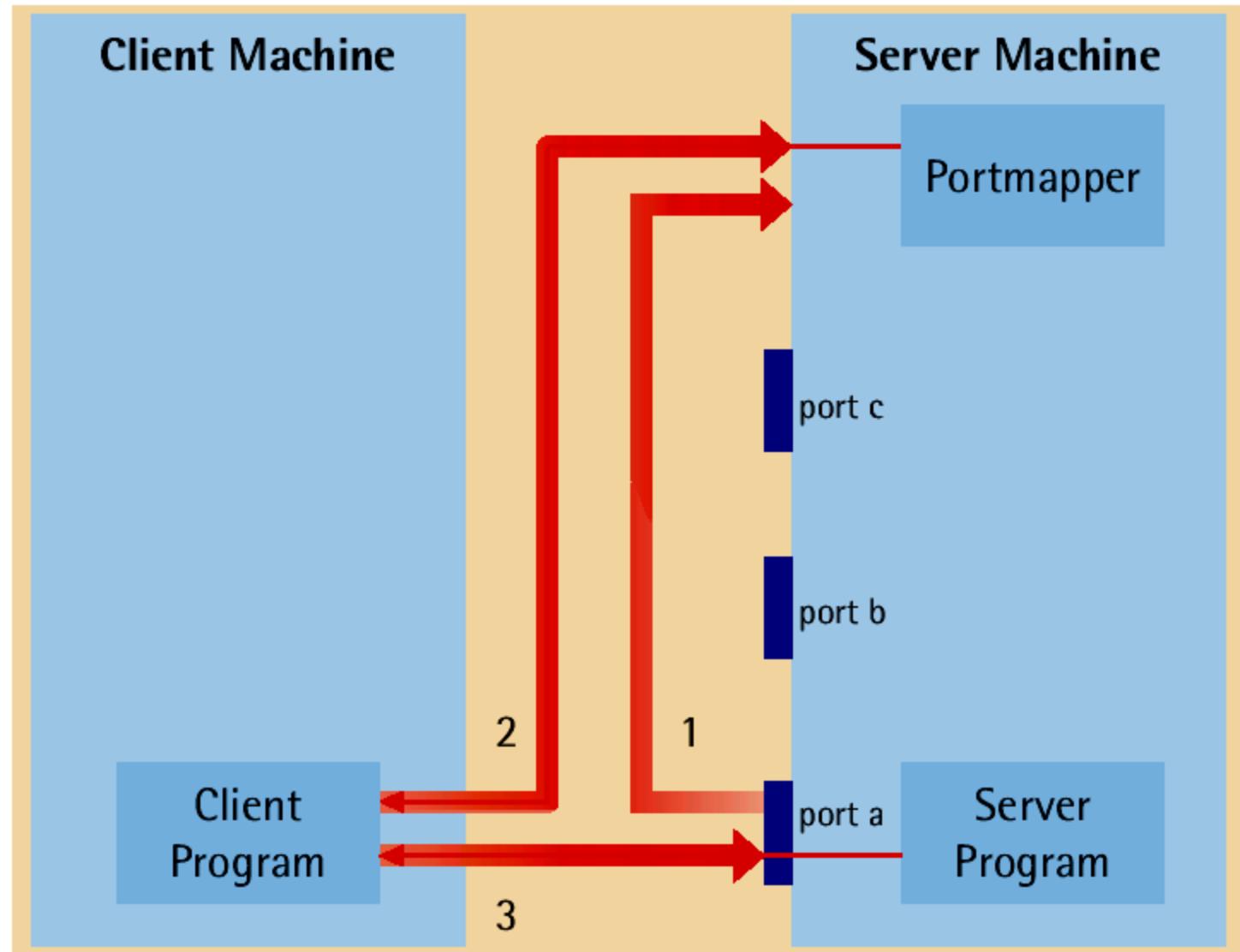
=> Numerele de port pentru un anumit serviciu nu sunt fixe

- Este disponibil la portul 111 (*well-known port*)

```
[adria@thor ~] $ rpcinfo -p
      program  vers  proto   port
      100000    2    tcp     111  portmapper
      100000    2    udp     111  portmapper
      100024    1    udp    56660  status
      100024    1    tcp    48918  status
```

# RPC | Functionare

Mecanism  
general



[Retele de calculatoare –  
curs 2007-2008, Sabin Buraga]

# RPC | Functionare

## Mecanism general:

**Pas 1:** Se determina adresa la care serverul va oferi serviciul

- La initializare, serverul stabileste si inregistreaza via *portmapper* portul la care va oferi serviciul (portul **a**)

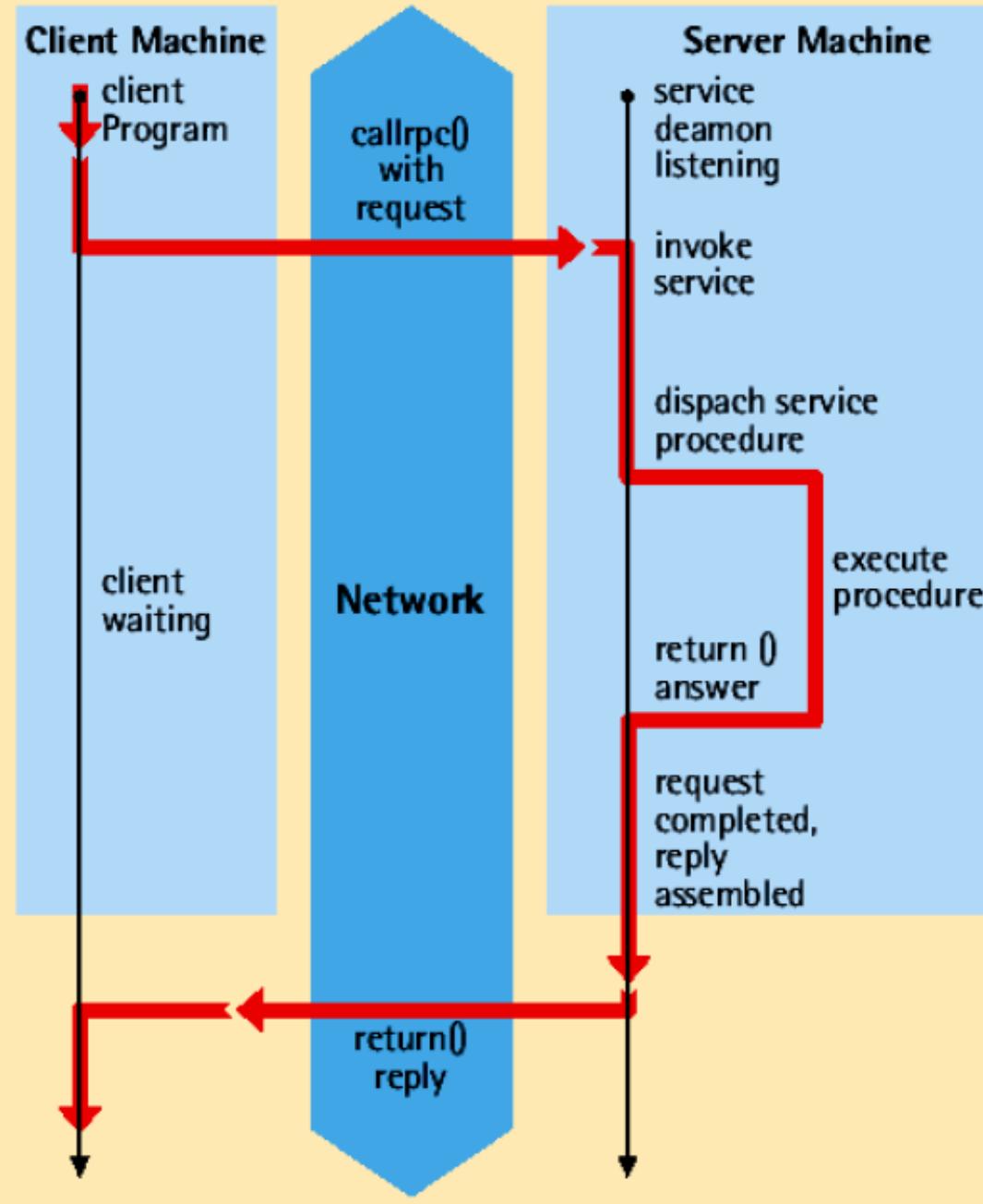
**Pas 2:** Clientul consulta *portmapper*-ul de pe masina serverului pentru a identifica portul la care trebuie sa trimita cererea RPC

**Pas 3:** Clientul si serverul pot comunica pentru a realiza executia procedurii la distanta

- Cererile si raspunsurile sunt (de)codificate prin XDR

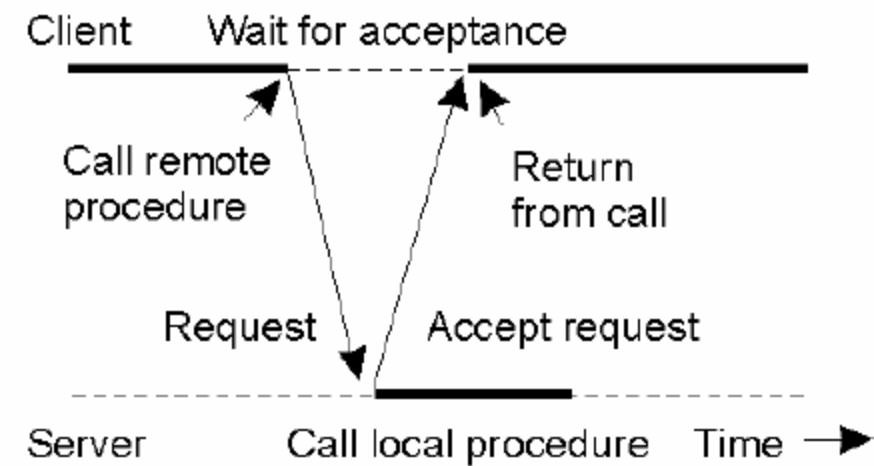
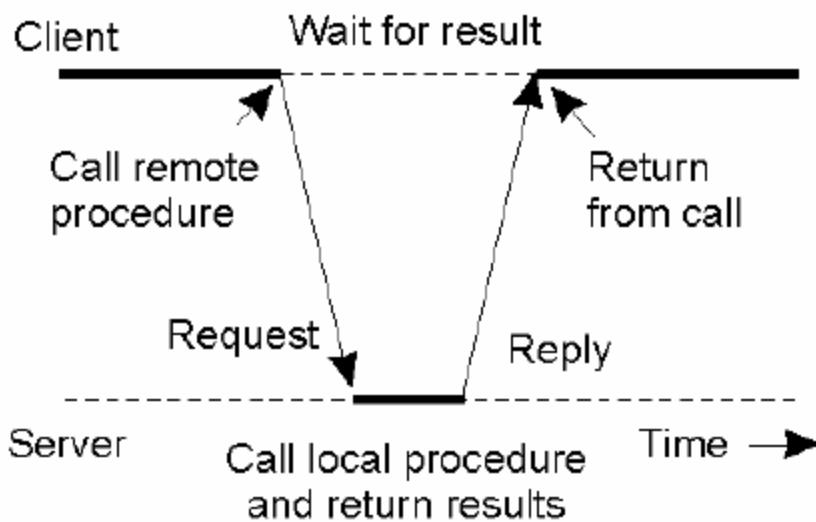
# RPC | Functionare

- Atunci cind un server furnizeaza mai multe servicii, este de obicei folosita o rutina *dispatcher*
- *Dispatcher-ul* identifica cererile specifice si apeleaza procedura corespunzatoare, dupa care rezultatul este trimis inapoi clientului pentru a-si continua executia



# RPC | Functionare

- Transferurile de date RPC pot fi:
  - Sincrone
  - Asincrone



[Retele de calculatoare –  
curs 2007-2008, Sabin Buraga]

# RPC | Implementare

- Open Network Computing RPC (ONC RPC) - cea mai raspandita implementare in mediile Unix (Sun Microsystems)
  - RFC 1057
  - Interfata RPC este structurata pe 3 niveluri:
    - **Superior:** independent de sistem, hardware sau retea
      - Exemplu: man **rcmd** -> *routines for returning a stream to a remote command ....*
    - **Intermediar:** face apel la functiile definite de biblioteca RPC:
      - **registerpc()** – inregistreaza o procedura spre a putea fi executata la distanta
      - **callrpc()** – apeleaza o procedura la distanta
      - **svc\_run()** – ruleaza un serviciu RPC
    - **Inferior:** da posibilitatea de a controla in detaliu mecanismele RPC (e.g. alegerea modului de transport al datelor etc)

# RPC | Utilizari

- **Accesul la fisiere la distanta NFS ( Network File System)**
  - Protocol proiectat a fi independent de masina, sistem de operare si de protocol – implementat peste RPC ( ... conventia XDR)
  - Protocol ce permite partajare de fisiere in retea => NFS ofera acces transparent clientilor la fisiere
    - Obs: Diferit fata de FTP
    - Ierarhia de directoare NFS foloseste terminologia UNIX (arbore, director, cale, fisier etc.)
  - NFS este un protocol => client - **nfs** , server –**nfsd** comunicind prin RPC
  - Modelul NFS
    - Operatii asupra unui fisier la distanta: operatii I/O, creare/redenumire/stergere, stat, listarea intrarilor
    - Comanda **mount** - specifica gazda *remote*, sistemul de fisiere ce trebuie accesat si unde trebuie sa fie localizat in ierarhia locala de fisiere
  - RFC 1094

# RPC|Utilizari

- Accesul la fisiere la distanta **NFS (Network File System)**
  - Este transparent pentru utilizator
  - Clientul NFS trimite o cerere RPC, serverului RPC folosind TCP/IP
    - Obs. NFS a fost folosit predominant cu UDP
  - Serverul NFS primeste cererile la portul 2049, si le trimite la modulul de accesare a fisierelor locale

Obs. Pentru deservirea rapida a clientilor, serverele NFS sunt in general *multi-threading* sau pentru sisteme UNIX care nu sunt *multi-threading*, se creaza si raman in kernel instante multiple a procesului (*nfsd*-uri)

# RPC | Utilizari

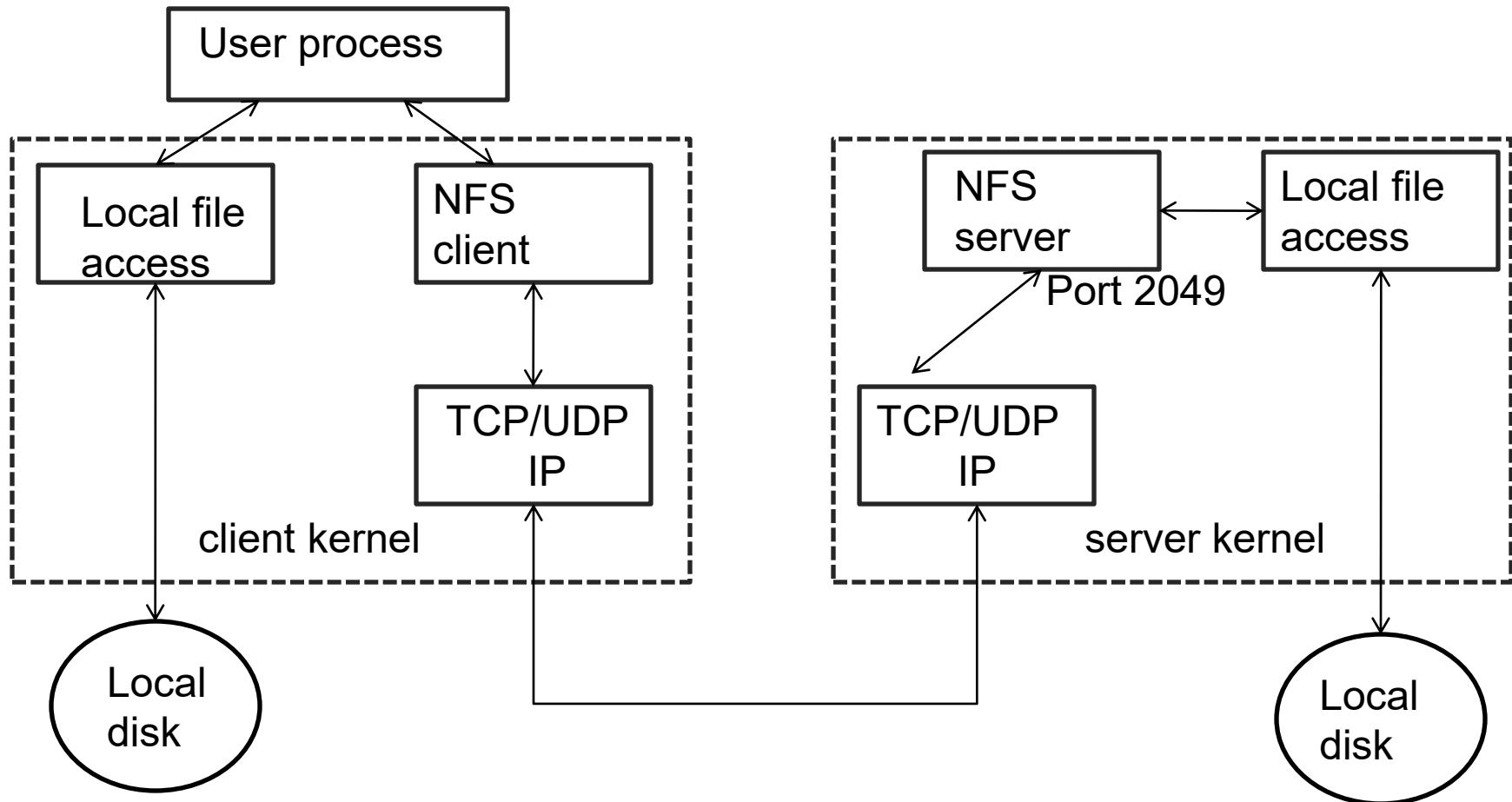


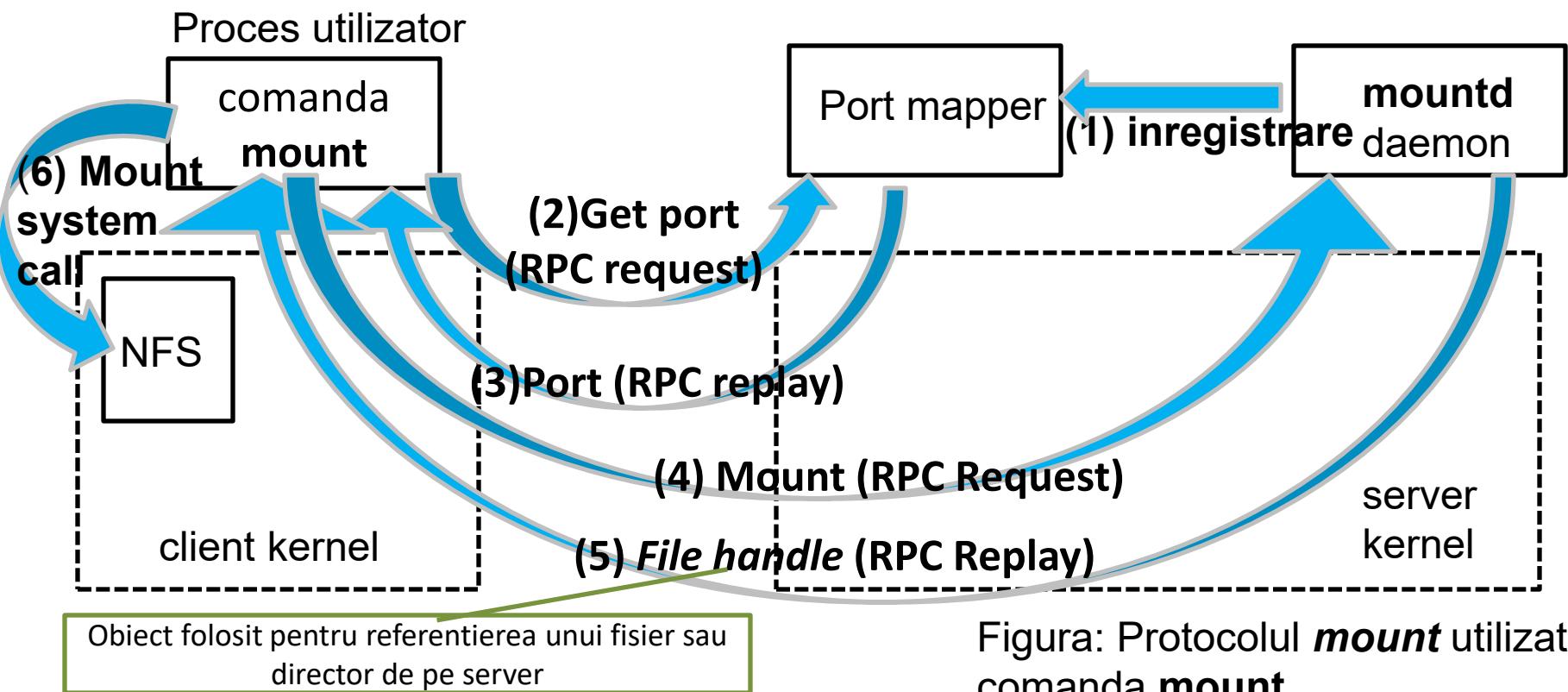
Figura: Arhitectura NFS

# RPC | Utilizari

- Accesul la fisiere la distanta **NFS (Network File System)**
  - (0) este pornit *portmapper*-ul la *boot-area* sistemului
  - (1) daemonul *mountd* este pornit pe server; creaza *end-point*-uri TCP si UDP, le asigneaza porturi efemere si apeleaza la *portmapper* pentru inregistrarea lor
  - (2) se executa comanda *mount* si se face o cerere la *portmapper* pentru a obtine portul serverului *demon* de *mount*
  - (3) *portmapper*-ul intoarce raspunsul
  - (4) se creaza o cerere RPC pentru montarea unui sistem de fisiere
  - (5) serverul returneaza un *file handle* pentru sistemul de fisiere cerut
  - (6) Se asociaza acestui *file handle* un punct de montare local pe client (*filehandle* este stocat in codul clientului NFS, si orice cerere pentru sistemul de fisiere respectiv va utiliza acest *file handle* )

# RPC | Utilizari

- Accesul la fisiere la distanta **NFS (Network File System)**
  - Procesul de montare (protocolul **mount**)
    - Pentru ca un client sa poata accesa fisiere dintr-un sistem de fisiere, clientul trebuie sa foloseasca protocolul **mount**



# Rezumat

- **Remote Procedure Call (RPC)**
  - Preliminarii
  - Caracterizare
  - XDR
  - Functionare
  - Implementari
  - Utilizari



# Intrebari?

“Alexandru Ioan Cuza” University of Iasi  
Faculty of Computer Science



Course  
Master  
I+II

## Prerequisites before Cloud Computing Cluster/Grid Computing

Lenuța Alboiae  
[adria@info.uaic.ro](mailto:adria@info.uaic.ro)

# Summary

## Paradigms

- **Cluster Computing**
- **Grid Computing**
  - Definition
  - Architecture
  - Initiatives and Applications
  - Classification
  - Evolution
  - Present & Future -> **Cloud Computing**

# What means *computing*?

- Computing



- The way one thinks



In computer science?

- “*we can define computing to mean any goal-oriented activity requiring, benefiting from, or creating computers.*”



# *... Computing?*

*“... computing may someday be organized as a public utility just as the telephone system is a **public utility**... The computer utility could become the basis of a new and important industry.”*—John McCarthy (a professor of MIT) 1961.

*“As of now, computer networks are still in their infancy, but as they grow up and become sophisticated, we will probably see the spread of **computer utilities** which, like present electric and telephone utilities, will service individual homes and offices across the country.*—L. Kleinrock (one of the chief scientists of the original ARPANET project) 1969” —John McCarthy (a professor of MIT) 1961.

# *... Computing?*

*“it was transformed in a model consisting of consumer services (commodity computing) and can be provided in a manner similar to traditional utilities”*

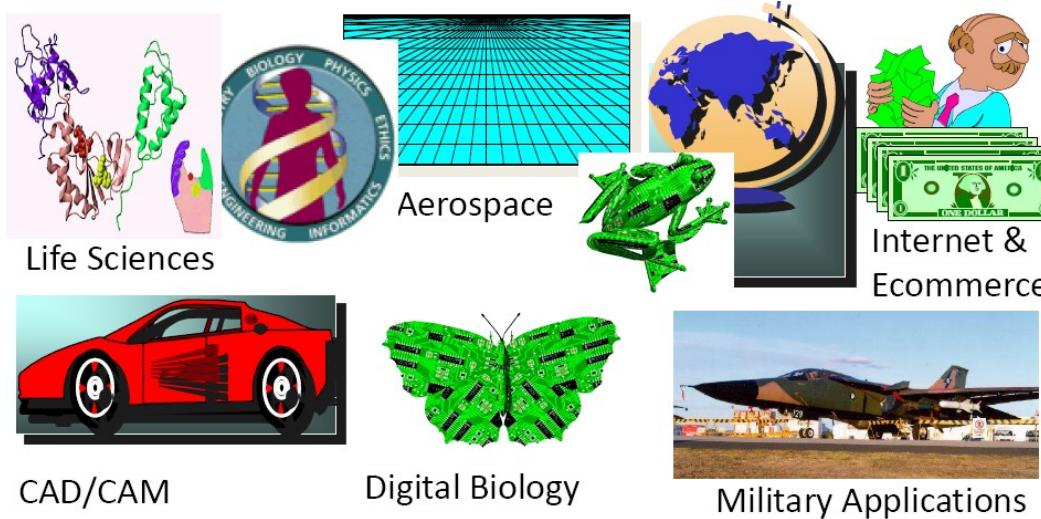


**Fifth utility -> Utility Computing or “Computing as a Utility”**

# Computing Power ?

Required:

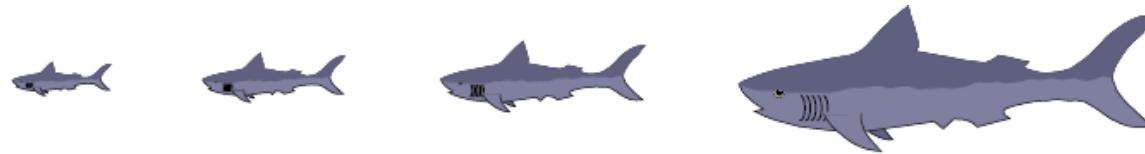
- solving problems involving modeling, simulation and analyzes



- Using unoccupied resources:
  - in the 90s almost 90% of a processor power was not used
  - the possibility to solve a wide variety of problems at affordable prices
  - cost/performance report in relation with a super-computer (HPC - high performance computer) => ....

# Trends

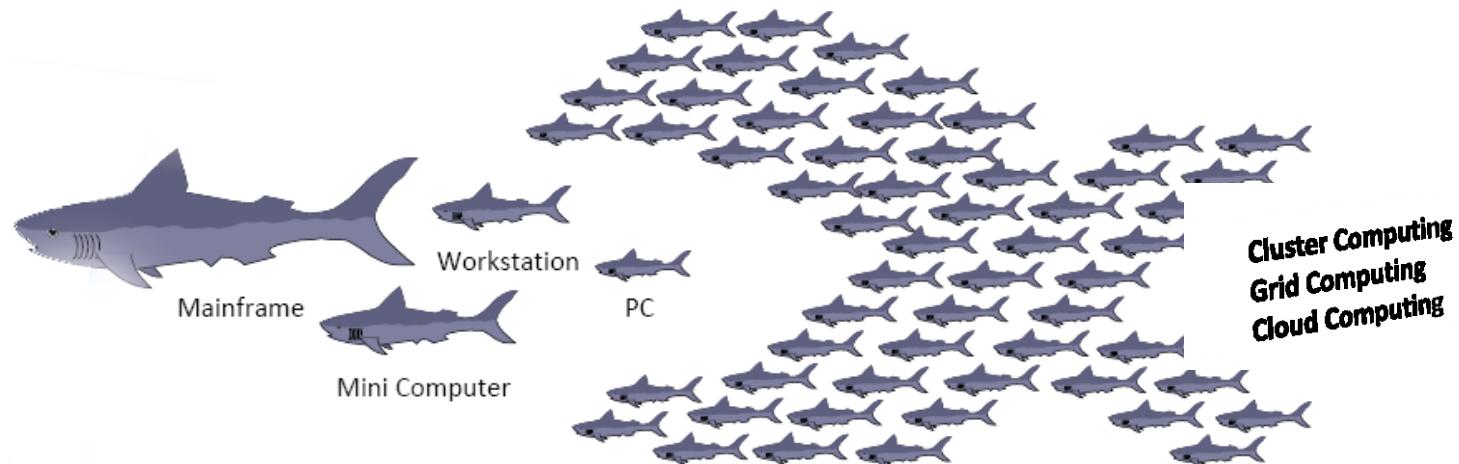
## *Traditional Food Chain*



## *Food Chain of a Computer*

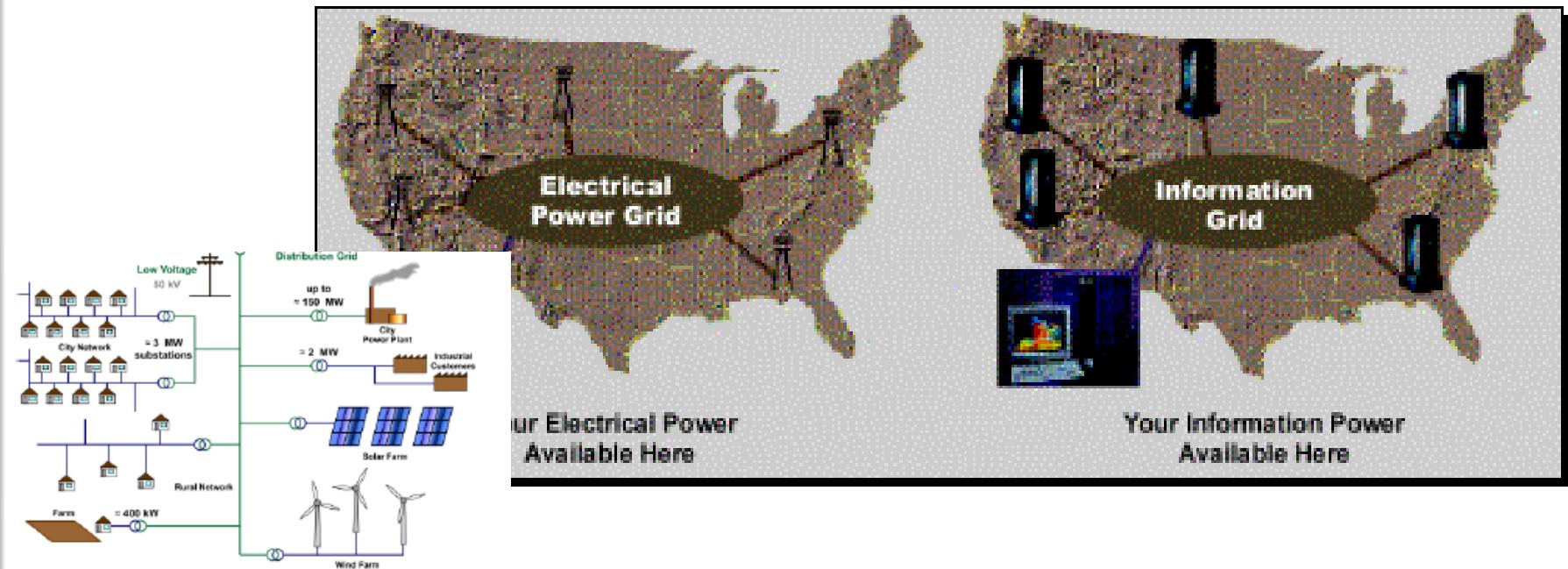


## *Food Chain of Distributed Computing*



# Grid Computing

- The Grid concept appeared in the 90s
  - In analogy with *electric power grids* ~ 1910



# Grid Computing

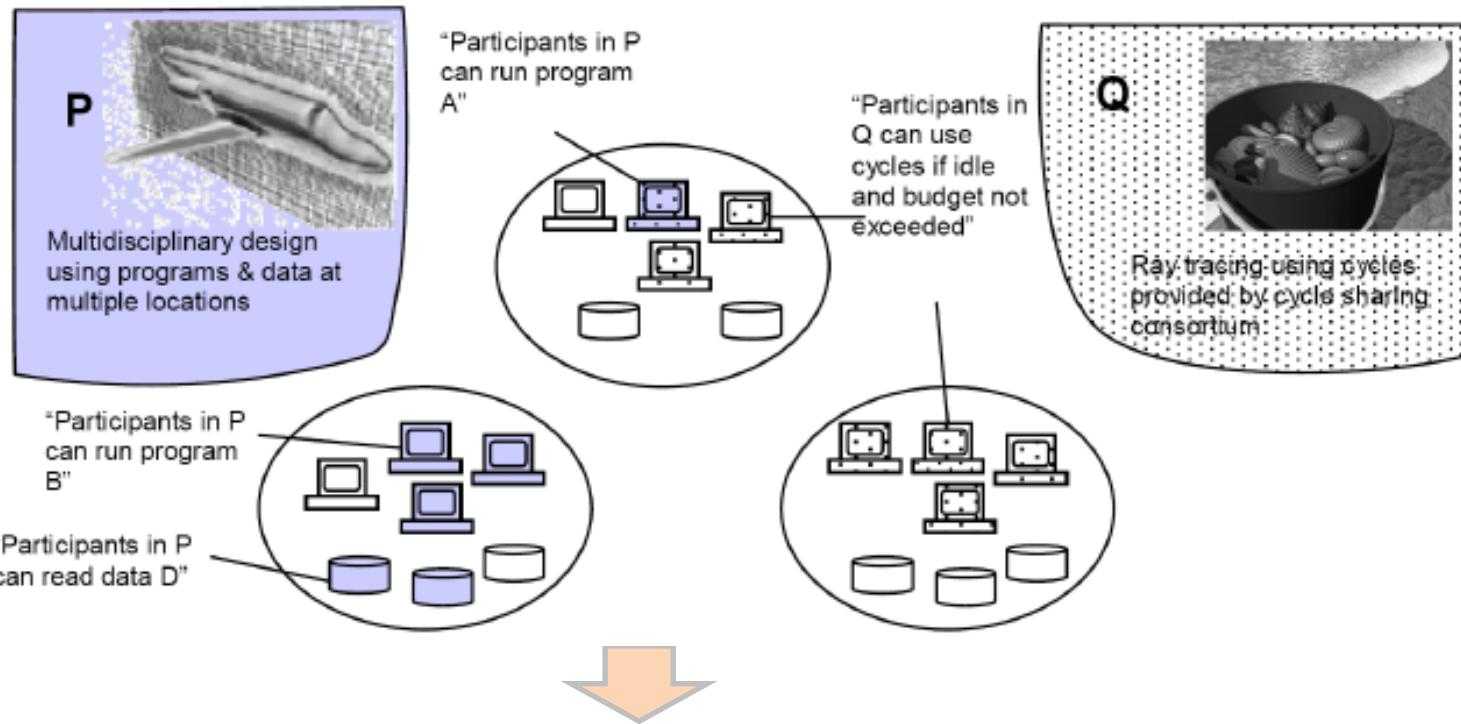
- Foster and Kesselman (1998): “*A computational grid is a hardware and software infrastructure that provides dependable, consistent, pervasive, and inexpensive access to high-end computational capabilities.*”
- “*The Grid is an emerging infrastructure that will fundamentally change the way we think – and use – computing. The word Grid is used by analogy with the electric power grid, which provides pervasive access to electricity and, like the computer and a small number of other advances has had a dramatic impact on human capabilities and society. Many believe that by allowing all components of our information technology infrastructure – computational capabilities, databases, sensors, and people – to be shared flexibly as true collaborative tools, the Grid will have a similar transforming effect, allowing new classes of application to emerge.*” (Foster and Kesselman 2004)

# Grid Computing

- Distributed computing architecture originally designed scientific projects and then the industrial ones
- Offers the existance of a software and hardware infrastructure which allows: permanent and affordable access in a consistent manner to computing resources
- Offers various mechanism to process data in a distributed manner
- Allows the execution of tasks on multiple machines that can be viewed as a single computer
- Offers support for searching and retrieving information, regardless of their physical location
- Offers the context to create VO - *virtual organizations* – which shares application, data in an open and heterogeneous environment in order to solve various complex problems
  - Rules for sharing:
    - What is shared?
    - Who can share?
    - Sharing conditions

# Grid Computing

- An organization can be involved in one or more VOs
  - Example: Three organizations and two VOs (P and Q)



What is shared: *Computing/processing power, Data storage/networked file systems, Communications and bandwidth, Application software, Scientific instruments*

# Grid Computing

- Terminology:
  - *Grid middleware* – software level providing the required functionalities needed for heterogeneous resources sharing and creating a virtual organization
  - *Grid infrastructure* – refers to the combination of hardware and Grid middleware which transforms disparate and heterogeneous computing resources in a virtual infrastructure that offers to the end user the view of a single machine
  - *Utility computing* – Grid Computing and applications are providing as services (e.g. *hosting* solutions for VO, et. al.)
    - Utility computing is based on business *pay-per-use* model

# Grid Computing | Architecture

- Grid Architectures use simultaneously a large number of resources (hardware, software, logical)
- Resource – a sharing entity that can be present in a Grid infrastructure:
  - Computation: PDA, PC, workstation, server, cluster
  - Storage: hard disk, RAID, SAN, ...
  - I/O type: sensors, networks, printersetc.
  - Logical: timers, ...
- Obs. Systems as: scientific instruments or HPC can be part of a Grid
- A Grid architecture focuses on interoperability issues , communication protocols between suppliers and the resource used in order to establish sharing relationships

# Grid Computing | Architecture

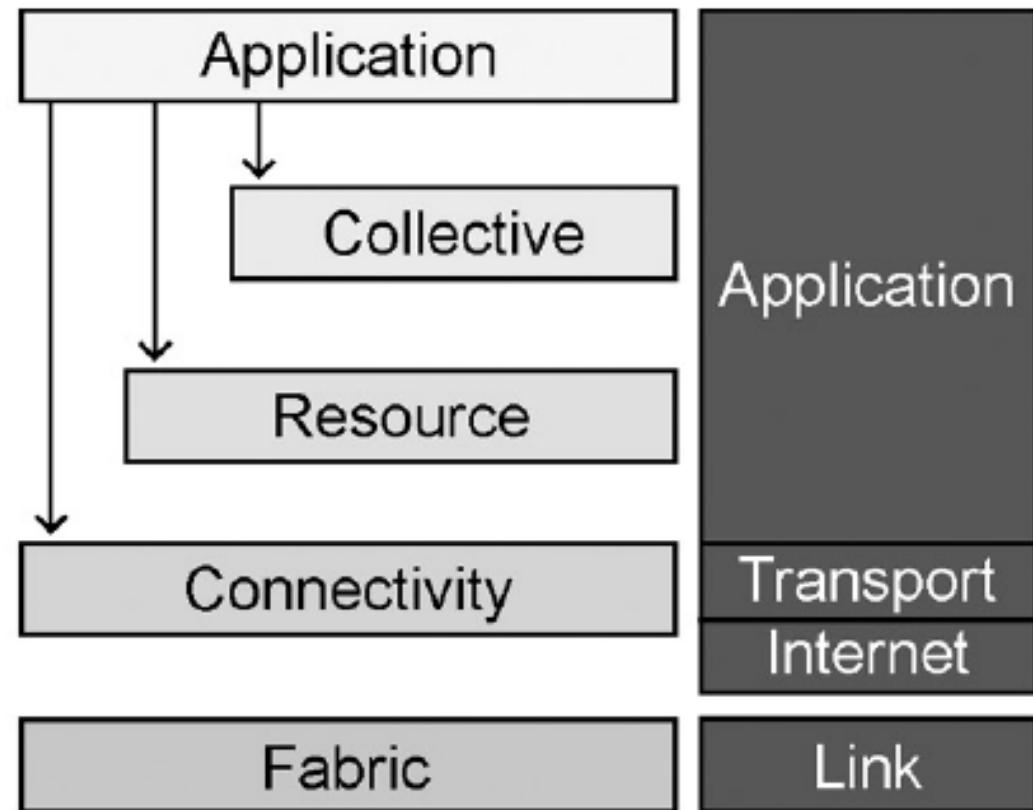
- Generic Grid architecture

"Coordinating multiple resources":  
ubiquitous infrastructure services,  
app-specific distributed services

"Sharing single resources":  
negotiating access, controlling use

"Talking to things": communication  
(Internet protocols) & security

"Controlling things locally": Access  
to, & control of, resources



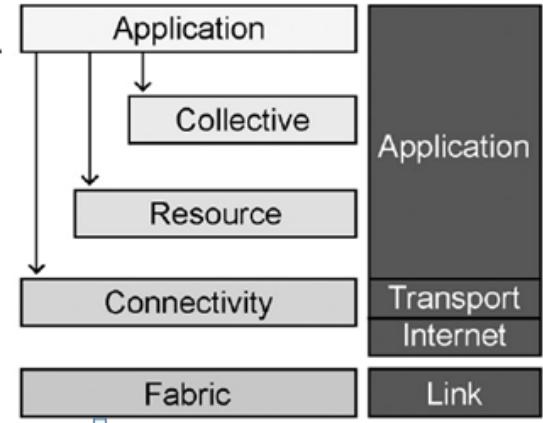
# Grid Computing | Architecture

## ■ Fabric

- Provides interfaces to physical resources (computing, storage, network, ...) for which the access is mediated by Grid protocols
- Offers components that implement local operations which are particular to each resource type

## Protocols & APIs

- Include protocols & APIs providing access to shared resources
- Offer a logical vision and not a physical one over resources



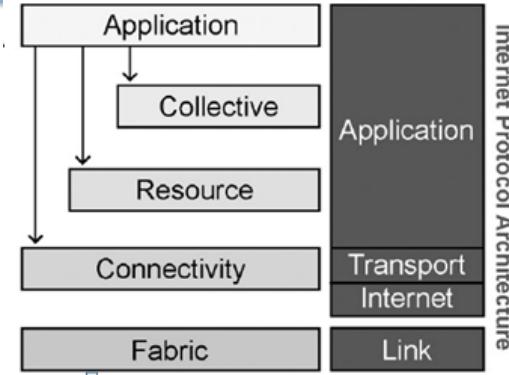
# Grid Computing | Architecture

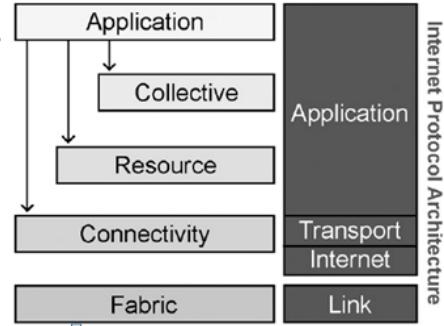
## ■ Connectivity

- Core of communication protocols and authentication protocols for network transaction within Grid
- Services for communication: transport (e.g. protocols to transfer data between resources, remote access to a resource), routing and naming
- Authentication services: *single sign on*, delegation, integration with local security solutions, relationships based on trust

## Protocols & APIs

- Standard Internet protocols
- Security protocols
  - Grid Security Infrastructure (GSI)
    - Authentication, authorization et.al.





# Grid Computing | Architecture

## ■ Resource

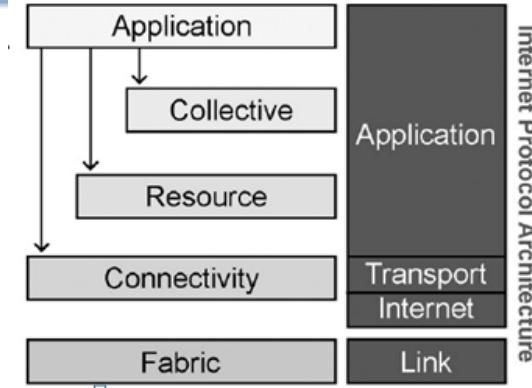
- Goal: communication and security protocols (defined in *Connectivity*) are used for secure negotiations, monitoring, control, accounting and payment of operation per each resource
- resource layer is responsible for managing a single resource
- Protocols:
  - Information protocols: used to obtain information about the structure and the state of a resource (e.g. configuration, load, use policies)
  - Management protocols: used to negotiate access to shared resources and to check the use of resources in accordance with the rules that were shared

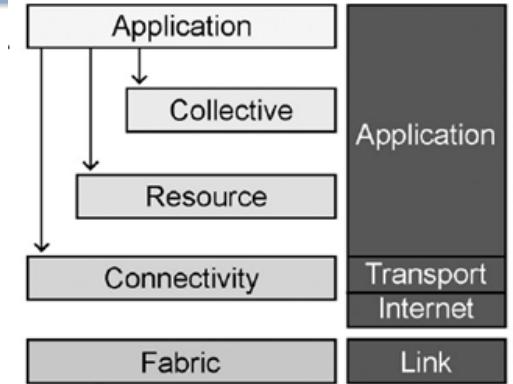
# Grid Computing | Architecture

## ■ Resource

### Protocols & APIs

- Protocols for the initiation and control of local resource sharing
- Management resource allocation: Grid Resource Allocation Management (GRAM)
  - Allocation, reservation, monitoring and resources remote control
- GridFTP – efficient data access & transport
- Information service for Grid resources:
  - Grid Resource Information Service (GRIS)
    - Access to the structure and the status of a node's Grid





# Grid Computing | Architecture

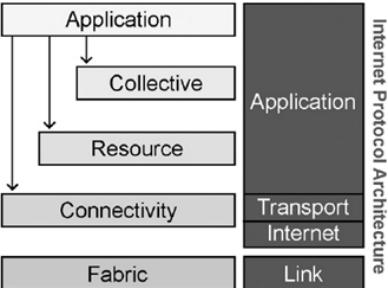
## ■ Collective

- Provides global protocols and services relating to grid resources
  - E.g. facilitates interactions between sets of resources
- Implement various sharing services:
  - *directory*
  - co-allocation, planning and intermediation (brokering services)
  - Monitoring and diagnostic (e.g. overload)
  - Replication and discovery
  - Quantification and payment

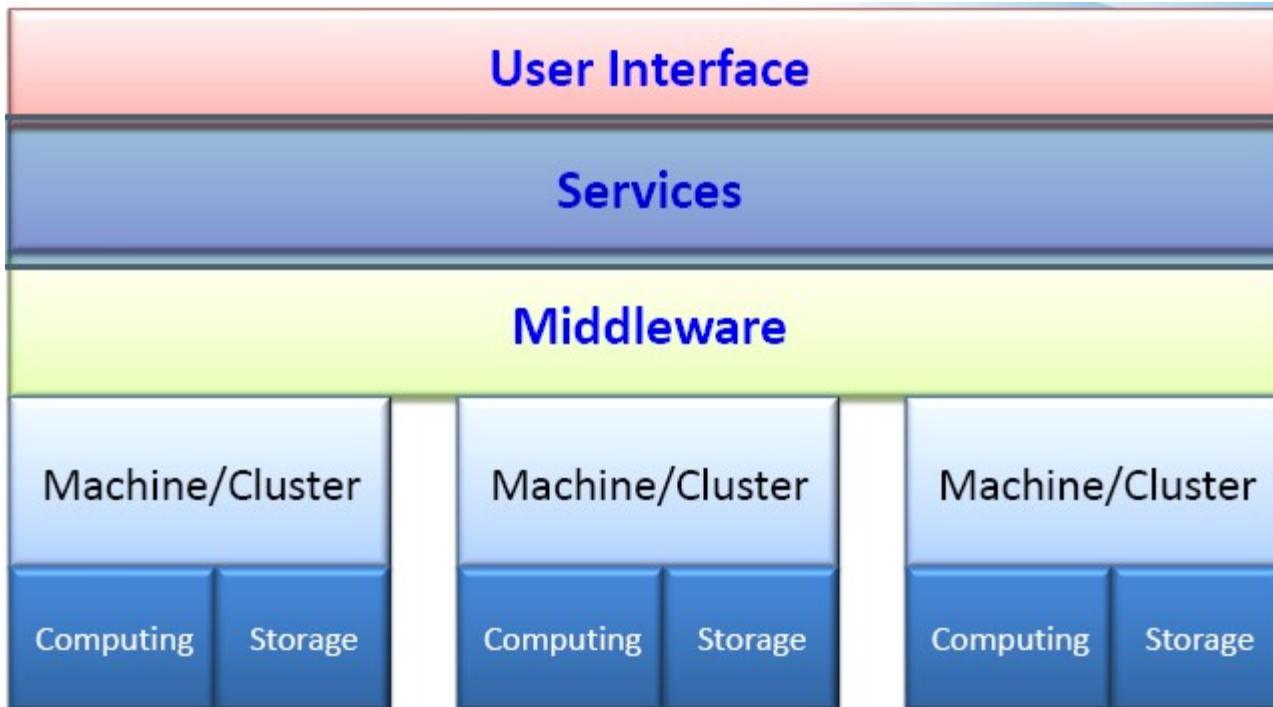
## ■ Application

- Includes user applications operating in Grid:
  - Programming environment + high-level libraries
  - Obs. *Gridified applications* –> applications designed to run in parallel and use multiple processors in Grid

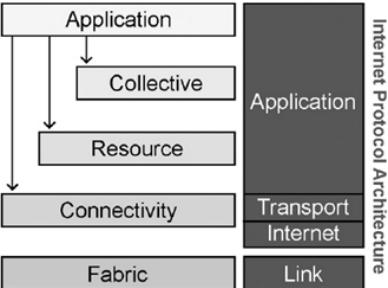
# Grid Computing | Initiative



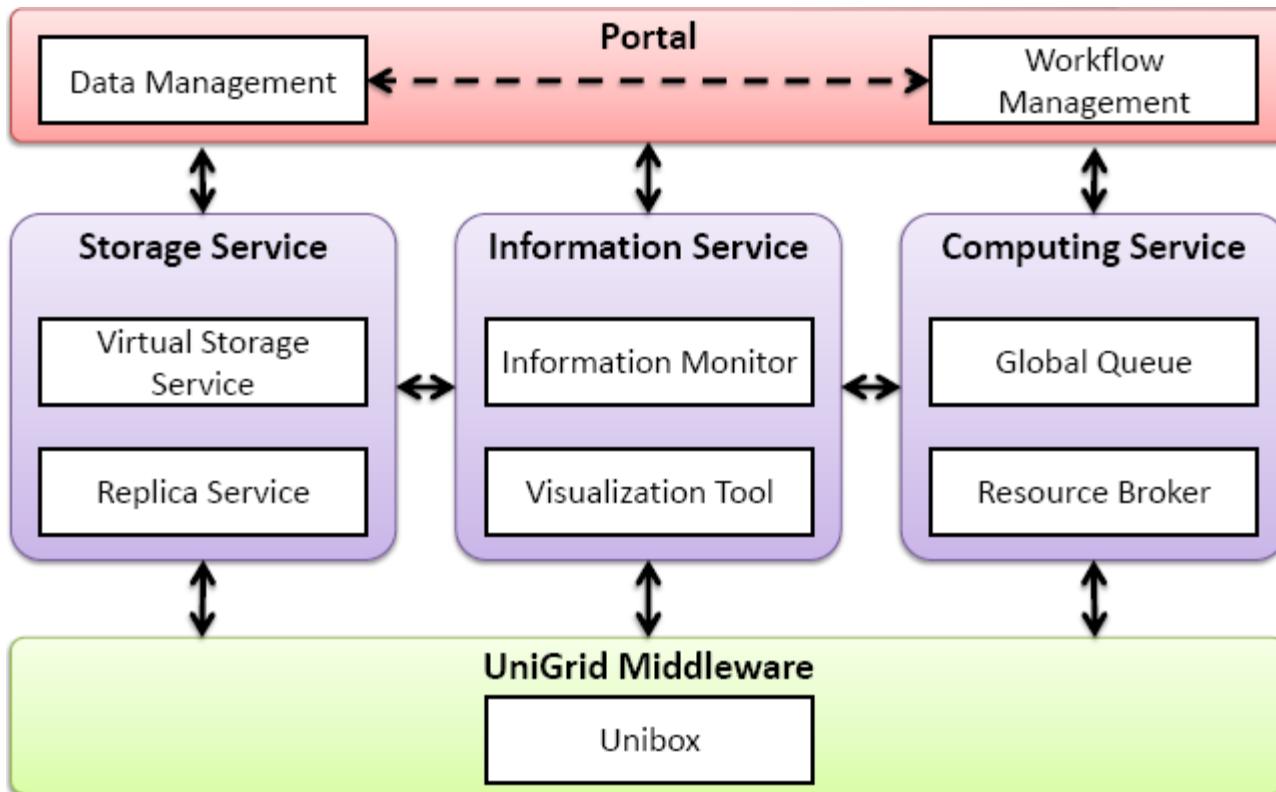
- UniGrid – a grid system that integrates computers in several universities and research institutes in Taiwan



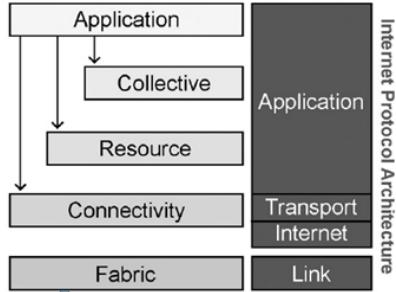
# Grid Computing | Initiative



- UniGrid – a grid system that integrates computers in several universities and research institutes in Taiwan



# Grid Computing



- Main features offered by *Grid middleware* (consisting of *Collective*, *Resource*, *Connectivity*)
  - Virtualization and integration of heterogeneous autonomous resources
  - Providing information on resources and their availability
  - A flexible and dynamic management regarding resource allocation
  - Security ( authentication and authorization ) and trust
  - Management of licenses
  - Billing and payment
  - Providing QoS



- Grid computing provides advantages at two levels
  - IT management
  - *business*

# Grid Computing

- Advantages of IT management level:
  - Grid integrates heterogeneous resources => higher availability of computing power and efficient use of resources
  - Lowering procurement costs
  - Reducing border between departments => more scalability
  - Efficiency in computing and access to resources due to : the ability of parallel computing , load balancing => increase robustness and reliability
  - In combination with Utility Computing , Grid Computing enables the transformation of capital expenditures or IT infrastructure in operating expenses and enables an increased scalability and flexibility
- Advantages at business level
  - Lower costs and higher income
  - Easier collaboration
  - Ability to create VO with business partners

# Grid Computing

- Risks and Challenges :
  - A suitable administration will avoid any ' Sever hugging " (e.g. sharing of resources that should not be shared )
  - Adjusting existing applications to function in a Grid environment
  - Lack of standards in Grid Computing leads to tough decisions on technologies used
  - Although Grid is designed to run on heterogeneous resources , involving high costs in terms of integration, it is worth considering from the perspective of keeping a standard for physical resources => full affecting the IT infrastructure

# Grid Computing | Initiative

- GridPP (UK Computing Grid for Particle Physics) - <http://www.gridpp.ac.uk/>
  - Contributes with over 40.000 PCs as part of the largest Grid in the world- LCG (LHG Computing Grid)  
LHG = Large Hadron Collider (CERN, din 2007)
  - It is part of EuroGrid project

The LHC was built in collaboration with over 10,000 scientists and engineers from over 100 countries, as well as hundreds of universities and laboratories.<sup>[3]</sup> It lies in a tunnel 27 kilometres (17 mi) in circumference, as deep as 175 metres (574 ft) beneath the Franco-Swiss border near Geneva, Switzerland.

- Fraunhofer Grid Alliance
  - Goal: providing a computational grid for easy access to grid resources via a Web portal
  - Based on Globus Toolkit
  - It is used in academic and industrial environment
  - [www.fhrg.fhg.de](http://www.fhrg.fhg.de)

# Grid Computing | Initiative

- Jgrid
  - Framework for Grids consisting of hardware/software components seen as services
  - It is based on Jini technology – infrastructure & programming model for creating dynamic distributed systems
  - jGrid applications can be developed via P-Grade (graphical development environment)
  - <http://jgrid.jini.org>
- Alchemi
  - Grid based on .NET Framework
  - Assure interoperability with other Grid systems via Gridbus Grid Service Broker

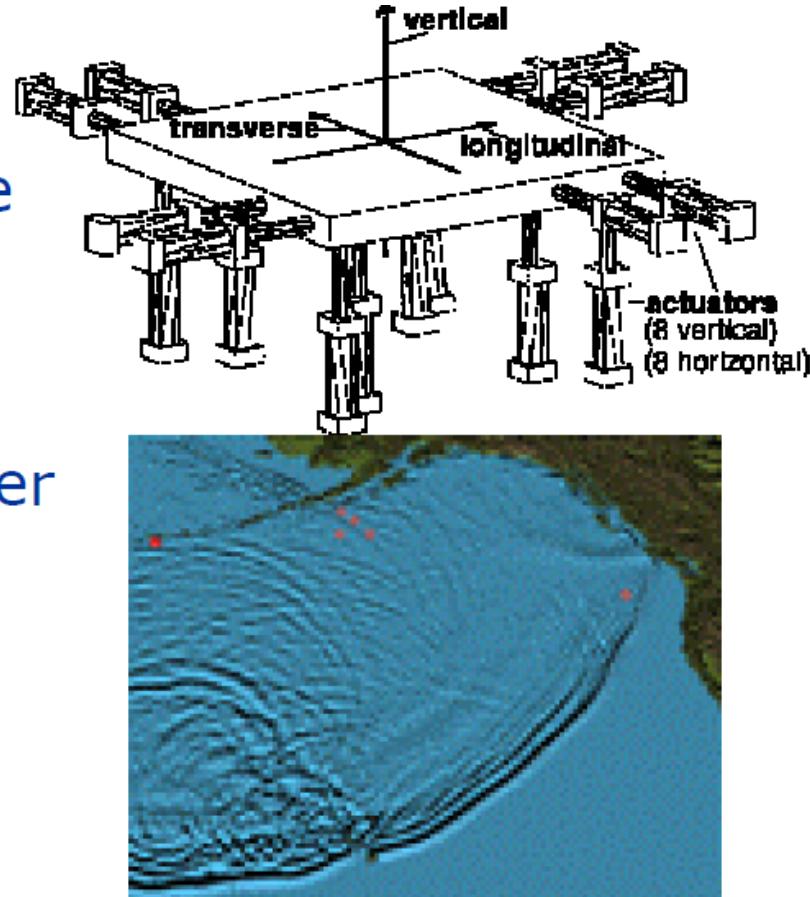
# Grid Computing | Example

- Examples of applications
  - Photorealistic 3D view
    - POV-Ray rendering (Persistence of Vision Raytracer)
  - Virtual Vascular Surgery
    - CrossGrid
    - <http://www.crossgrid.org>
  - Solving optimization problems
    - TRACER project ( use Globus, Condor, Legion, Sun Grid Engine)
    - <http://neo.lcc.uma.es/>

# Grid Computing | Example

- Example: Earthquake Engineering Simulation

- NEESgrid: national infrastructure to couple earthquake engineers with experimental facilities, databases, computers, & each other
- On-demand access to experiments, data streams, computing, archives, collaboration



NEESgrid: Argonne, Michigan, NCSA, UIUC, USC

[[http://www.nesc.ac.uk/talks/talks/Grids\\_and\\_Globus.pdf](http://www.nesc.ac.uk/talks/talks/Grids_and_Globus.pdf)]

# Grid Computing | Example

- Example: Home Computers evaluate AIDS Drugs

- Community =
  - 1000s of home computer users
  - Philanthropic computing vendor (Entropia)
  - Research group (Scripps)
- Common goal= advance AIDS research

The screenshot shows the homepage of the FightAIDS@Home website. The header features the text "fight AIDS @home" in large letters, with "the Olson laboratory at The Scripps Research Institute" and "computing toward a cure" below it. The "powered by entropia" logo is also present. The main content area includes a sidebar with links like "Free Software for Your PC", "Download", and "Get Project News via E-mail". The central text explains the project's purpose: using volunteers' computers to accelerate anti-HIV drug design research. It highlights the use of Entropia's global Internet computing grid and mentions the Olson laboratory at The Scripps Research Institute. A footer at the bottom left indicates the page was last updated on September 22, 2000.

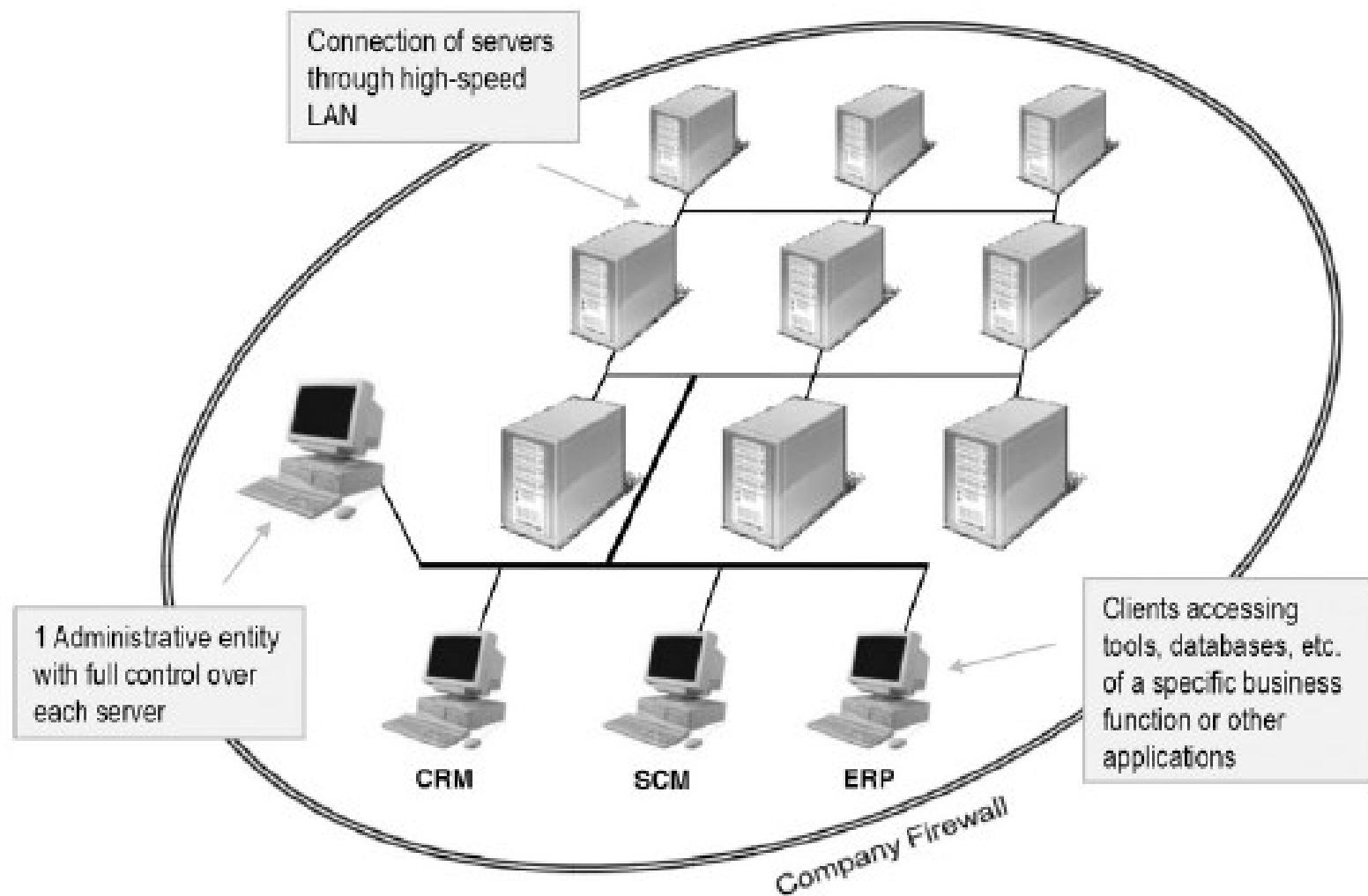
[[http://www.nesc.ac.uk/talks/talks/Grids\\_and\\_Globus.pdf](http://www.nesc.ac.uk/talks/talks/Grids_and_Globus.pdf)]

# Grid Computing

- Classifications:
  - In relation to the type of managed resources
    - Compute Grid – used to share computing resources (e.g. CPU) - Examples: intensive graphic processing
    - Data Grid – focused on storage, management and sharing of distributed and heterogeneous resources
    - Application Grid – focused on application management and transparently providing remote access to software and libraries; Example: grids in the bioinformatics field or earth science
    - Service Grid – resulting from Grid and SOA convergence, offers support for sharing services in a efficient manner
  - In relation to the resource sharing domain:
    - Cluster Grid
    - Enterprise Grid
    - Utility Grid Services
    - Partner/Community Grids

# Classification

## ▪ Cluster Grid



[Grid and Cloud Computing - A Business Perspective on Technology and Applications, 2010]

# Tipuri

- **Cluster Grid**
    - It is a type of parallel and distributed system and consists of a collection of autonomous computers interconnected used (and seen) as a unique resource at department / group
- Departmental grid (Sun)/ infra grid (IBM)*

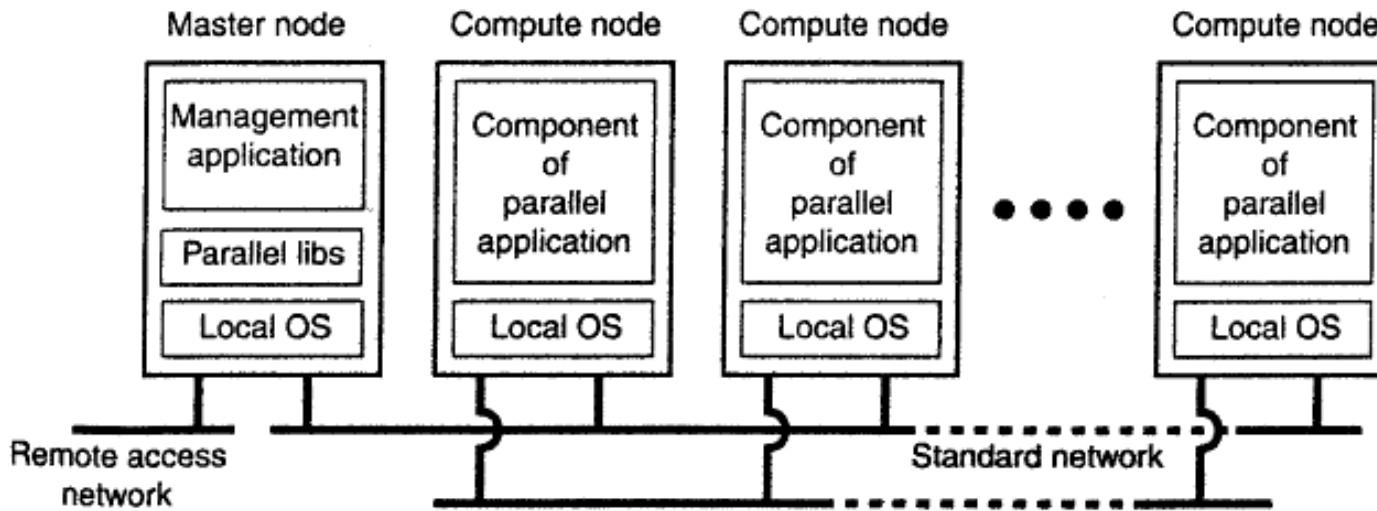


# Tipuri

- **Cluster Grid**
  - Enables full use of computer resources (*mainframes*, PCs, laptops, *smartphones*, ...)
    - Cluster = set of computers – from a LAN – which form a unique computing resource
  - Obs. Clusters offer no implicit sharing of resources (improves computing capacity and storage level), and may be considered the first step towards Grid Computing

# Implementation

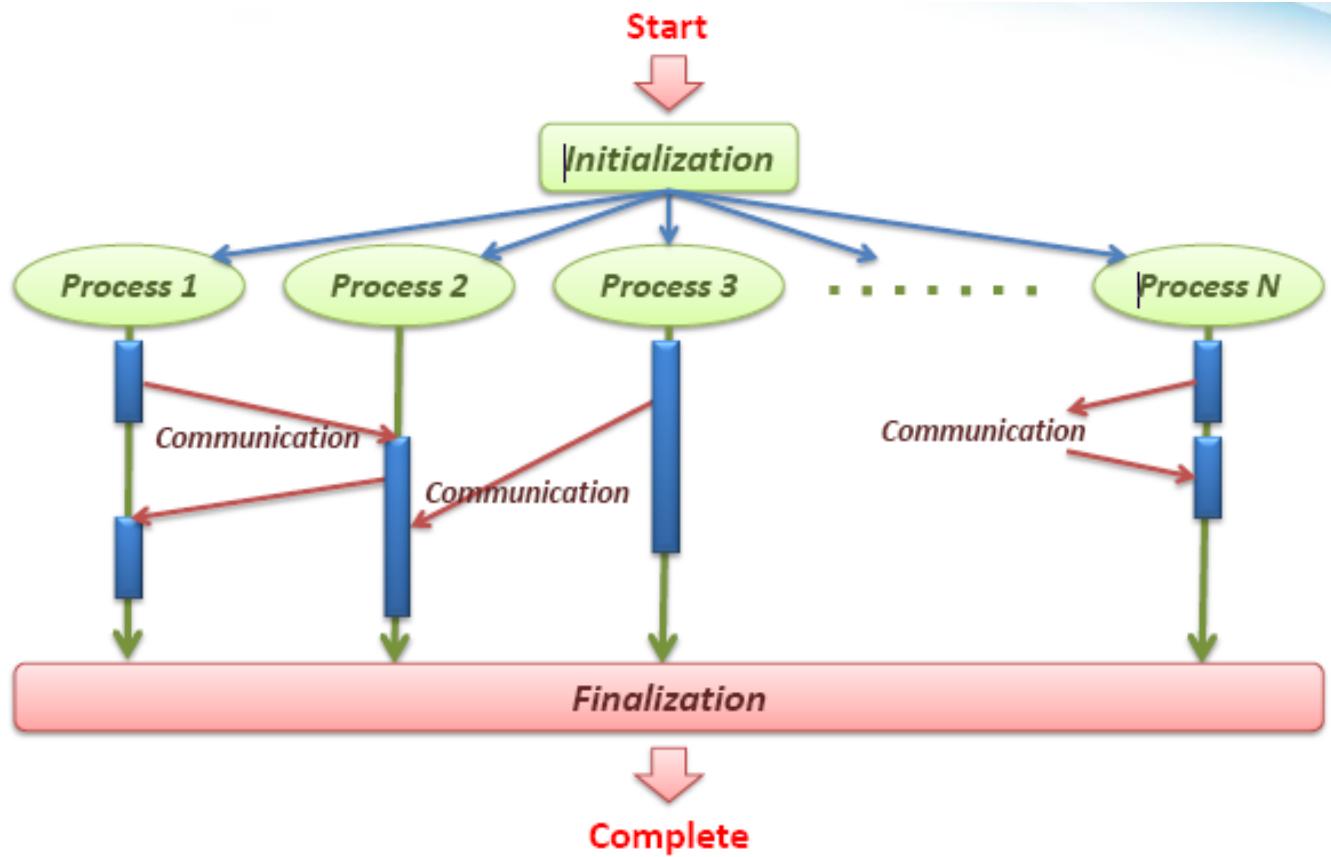
- Beowulf (published in 2003)
  - Offer support for establishment of clusters
  - Computers can be added dynamically
  - Communications via MPI (*Message Passing Interface*)
  - Offer a programming model which is independent by structure, network technologies or components used
  - It contains: *master nodes* (coordinator) and *slave/worker* (processors)



[A. S. Tanenbaum,  
M. Steen,  
DISTRIBUTED  
SYSTEMS]

# Implementation

- Typical flow for parallel executions:



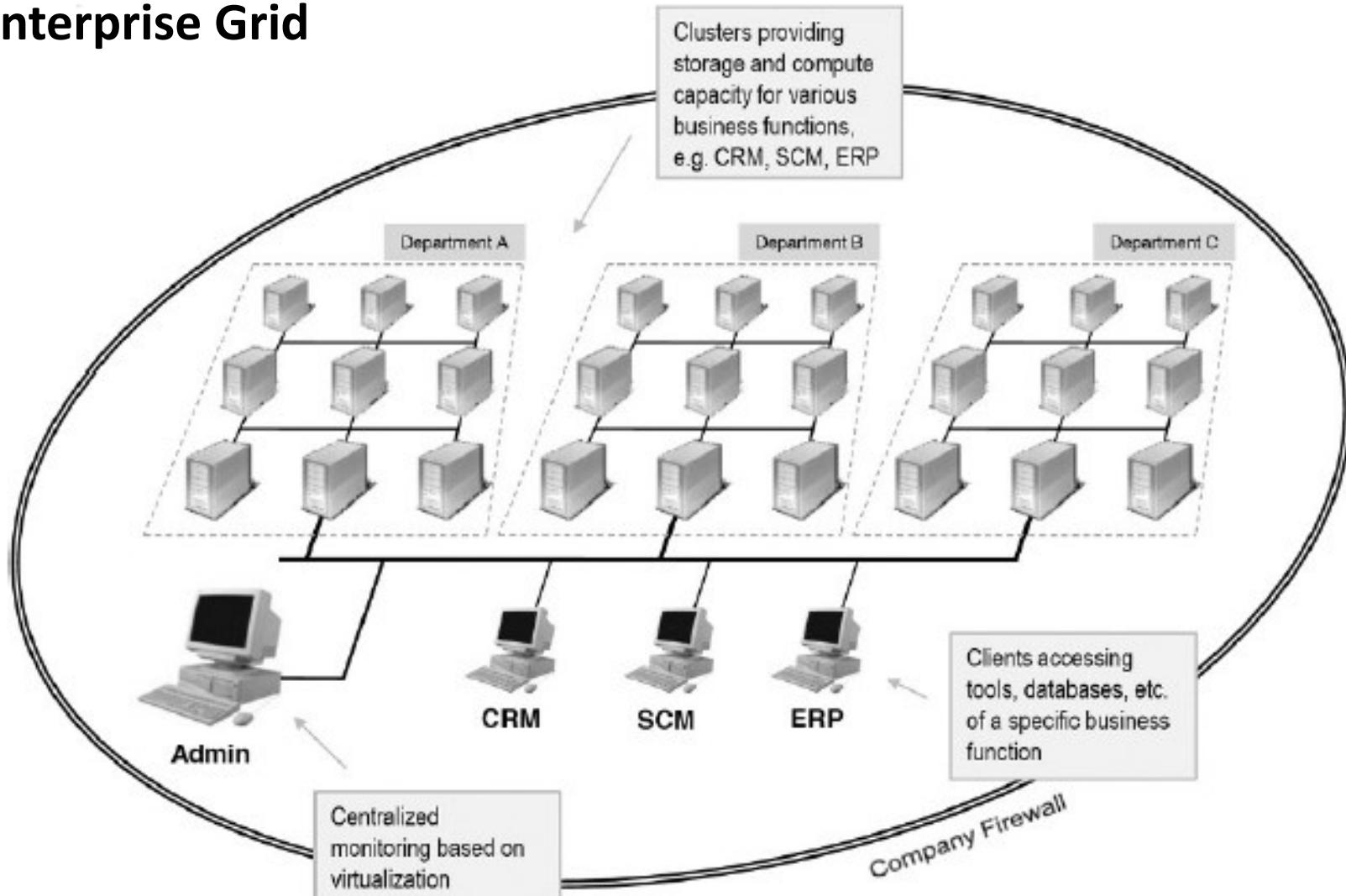
[A. S. Tanenbaum,  
M. Steen,  
DISTRIBUTED  
SYSTEMS]

# Classification

- **Cluster Grid**
  - HPC – High Performance Computing:
    - Numerical Calculus
    - Computational Graphics 2D/3D (rendering – e.g., ray tracing, shading, ...)
    - Simulations (Biocomputing, military, ...)
    - Distributed resource search
    - Real-time critical applications
    - Distributed storage of large amount of data (*warehouses*)
    - Entertainment – e.g.: online games

# Classification

## ■ Enterprise Grid



[Grid and Cloud Computing - A Business Perspective on Technology and Applications, 2010]

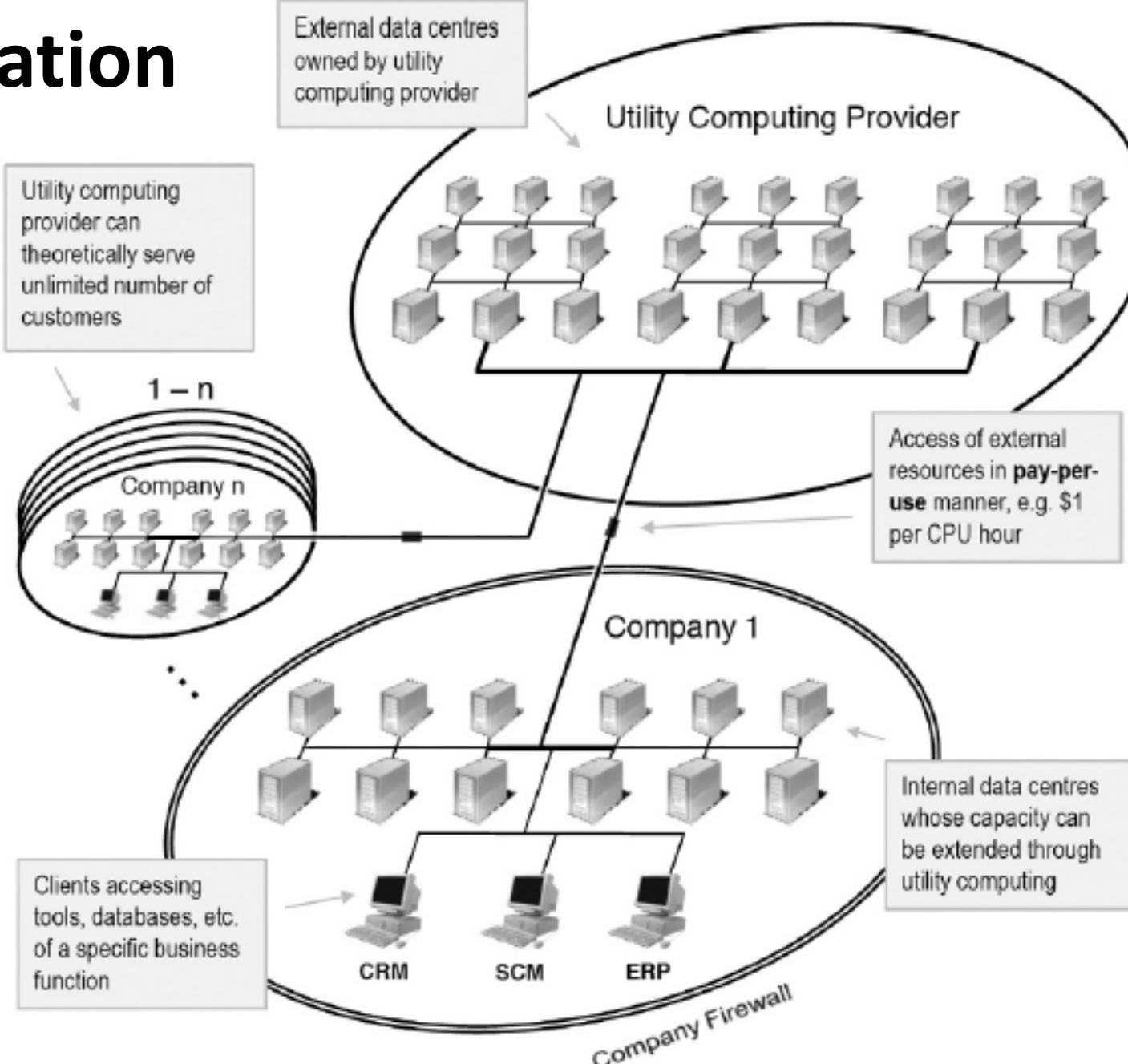
# Classification

## ■ Enterprise Grid

- Facilitates resource sharing between multiple departments within an organization ( even a virtual
  - Politics for resource management
- It is called *intra grid* or *campus grid*
- Example: Novartis - Pharmaceutical company
  - Held in 2003 an infrastructure consisting of thousands of desktop
  - Pilot Grid Project : 2003, Basel (Elvetia), 50 PCs “*Grid enabled*” connected to the existing nodes (Goal: determining the protein structure)
    - In each node there was an agent that checks system load
    - => Result: a week of running in Enterprise Grid led to results that could be obtain in 3.18 years
  - 2700 PCs (Basel, Viena, Cambridge)

# Classification

## ■ Utility Grid



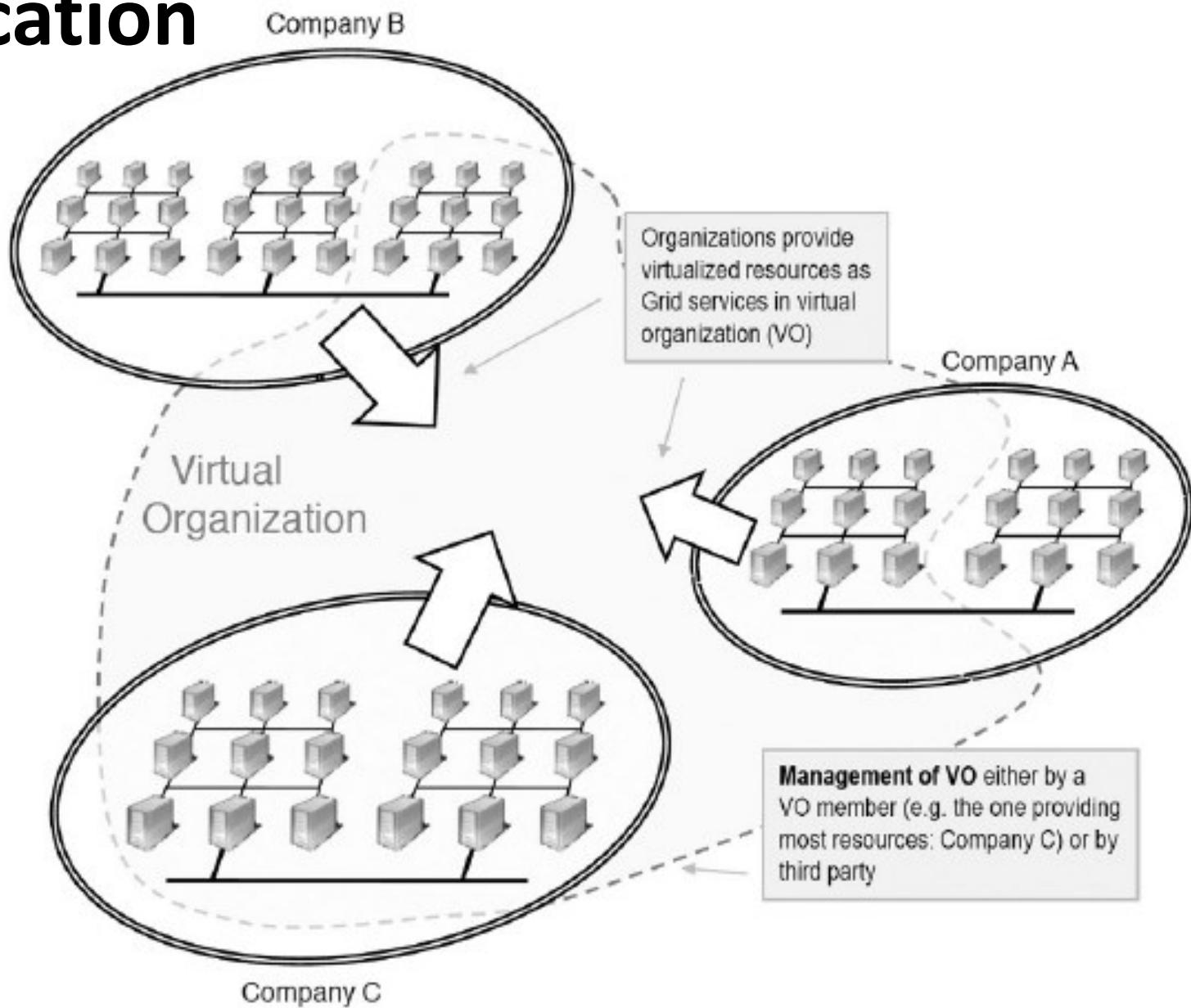
[Grid and Cloud Computing - A Business Perspective on Technology and Applications, 2010]

# Classification

- **Utility Grid**
  - The Grid environment is developed and managed by a service provider
  - The usage of computing power or storage services is in *pay-per-user manner*
  - Functionality: the user does not have the Grid, he has no control over operations
    - Data and various computing operations are transmitted and then the result is expected
    - => security and *privacy problems*
    - => *reliability problems*
    - => unnecessary IT infrastructure investments
    - => Utility Computing offers scalability and flexibility on request
- Examples:
  - Sun Grid Compute Utility from 2006
    - *Pay-per-use*: 1\$/CPU per hour
    - Latter it offered support for applications
  - HP Labs offers Utility Computing for DreamWorks

# Classification

- Partener/  
Community  
Grid



# Classification

- **Partner/Community Grid**
  - Provides support for building VO layering on shared IT infrastructure
  - The architecture can be viewed as a collection of independent resources (e.g. Cluster Grids) that are interconnected in a global Grid middleware
  - *Partner grids* – are established between companies and universities that have a common goal
    - It defines sharing politics for resources
  - Community Grids – rely on the donation of resources (often from private individuals)
    - Example: SETI@HOME
- **Vision: Open Global Grid**
  - Represents a collection of heterogeneous Grids geographically distributed over a wide area – continent or planet
    - Global Use Policy
    - General protocols for resource sharing
    - => no additional configuration is required for access

# Grid Computing | Evolution

- Generation 1 – Globus project (Goble & Foster)
  - Applications requiring high computing power
  - Includes protocols (LDAP, FTP) and heterogeneous development tools
  - Support for access and files transfer
  - Use Internet technologies, but ignore the Web
  - Employed mainly in academic environment
  - Sharing resources is achieved via GridFTP
  - Implementations: ...Legion, Condor, Unicore, ....

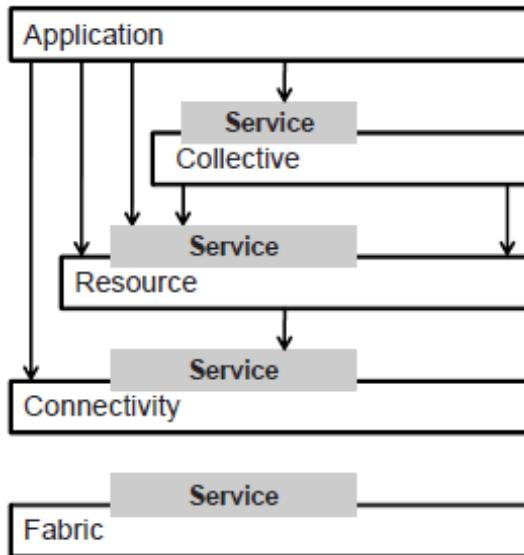
# Grid Computing | Evolution

- Generation 2 – OGSA (*Open Grid Services Architecture*)
  - There is convergence of Service-oriented computing (SOC) and Grid Computing

“Service-oriented Computing (SOC) is a new computing paradigm that utilizes services as the basic construct to support the development of rapid, low-cost and easy composition of distributed applications even in heterogeneous environments. The visionary promise of Service-Oriented Computing is a world of cooperating services where application components are assembled with a little effort into a network of services that can be loosely coupled to create flexible dynamic business processes and agile applications that may span organisations and computing platforms.” (Papazoglou et al. 2006)
- We notice the interoperability and sharing vision of SOC at application lever versus Grid computing vision mainly at hardware level
  - Generation 1: Grid Computing architecture consists of protocols and services used to describe and share available physical resources
  - By using Web Services Standard ( such as: WSDL, SOAP, BPL4WS,...) Grid protocols and Services can be described in a standardized manner

# Grid Computing | Evolution

- Generation 2 – OGSA (*Open Grid Services Architecture*)



- Using the same standards => it was possible the convergence between Grid Computing and SOC => besides hardware and system resources, applications become shareable
- OGSA:

“Building on concepts and technologies from both the Grid and Web Services communities, OGSA defines a uniform exposed service semantics (the *Grid Service*); defines standard mechanisms for creating, naming, and discovering transient Grid service instances; provides location transparency and multiple protocol bindings for service instances; and supports integration with underlying native platform facilities.” (Foster et al. 2002)

# Implementation

- Generation 2 – OGSA (*Open Grid Services Architecture*)

Grid services must be:

- **Dynamic and volatile** – set of composed services that can be invoked or removed “on the fly”
- **Ad-hoc** – there is no central location or central control
- **Widespread** – orchestrating a large number of services (> 100) should be performed anytime
- **Available** – potentially long-term (e.g. a simulation can take weeks)
- OGSI (Open Grid Service Infrastructure)
  - OGSA Infrastructure - “accommodates” interactions between Grid resources and Web Services
  - Model implemented by Globus Toolkit 3.0
    - » OGSI was replaced by WSRF (Web Service Resource Framework): WS- Security, WS- Management and other standards for Web Services => Globus 4.0

# Grid Computing | Evolution

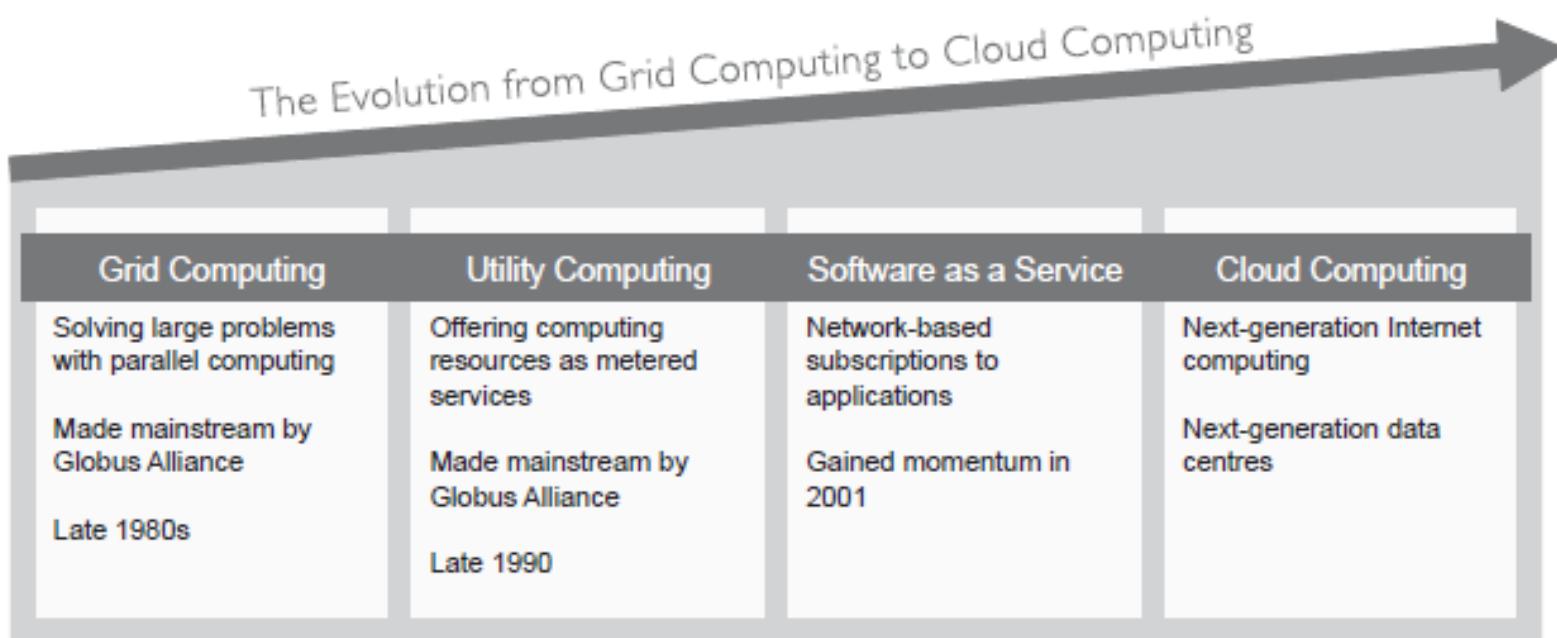
## ■ Generation 3 – present and future

- Convergence of Grid Computing and SaaS (Software-as-a-Service) paradigm
- SaaS
  - Designates software that is owned, delivered and managed by a provider
  - It is used in the *pay-per-use* principle via a Web browser or APIs
  - Versus traditional software
    - The user pays for the time of use
    - The user does not have the software, he does not invest in the infrastructure or licenses
  - History: Application Service Provisioning (ASP) – appeared in 1988
    - It was a step for IT outsourcing and it comes with the idea of Web applications that can be provided by a central supplier (one-to-many delivery model)
      - The main problem: the inability to provide personalized services
      - Issues regarding scalability, robustness,....



# Grid Computing | Evolution

- Generation 3 – present and future
  - ASP problems can be solved by using Grid Computing + Web Services
    - Web Services allows services personalization
    - Grid Environment offer flexibility and scalability
- => *many-to-many delivery model*

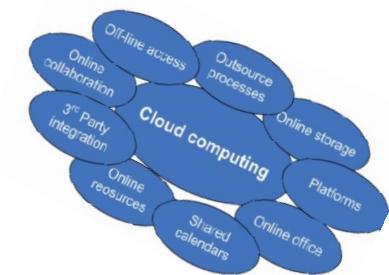


The Evolution to Cloud Computing (adapted from IBM 2009)

# Present and future

## Overview

- Two directions of evolution:
  - Grid Computing
    - Mature technology
    - It provides computational power in *pay-per-use manner* => new business models for *utility computing*
    - There were many initiatives at hardware level: Sun, IBM, etc.
  - There were many initiatives at software level -> SaaS
    - Microsoft, SAP et. al.
- **Next step...**
  - A scalable , robust and reliable physical infrastructure,
  - Services that provide developers access to infrastructure by manipulating abstracted interfaces
  - SaaS running on a flexible and scalable infrastructure



# Bibliography

- Katarina Stanoevska Slabeva, Thomas Wozniak, Grid and Cloud Computing - A Business Perspective on Technology and Applications, 2010, Editors Santi Ristol, Springer-Verlag Berlin Heidelberg
- Massimo Cafaro, Givani Aloisio, Grids, Clouds and Virtualization, 2011
- Cloud Computing, Wu Chung – online resource
- Foster I, Kesselman, C, Tuecke S (2001) The Anatomy of the Grid: Enabling Scalable Virtual Organization. International Journal of High Performance Computing Applications 15(3):200- 222

# Abstract

## Paradigms

- **Cluster Computing**
- **Grid Computing**
  - Definition
  - Architecture
  - Initiatives and Applications
  - Classification
  - Evolution
  - Present & Future -> **Cloud Computing**



# Questions?

“Alexandru Ioan Cuza” University of Iasi  
Faculty of Computer Science

Master  
Course  
I+II

Conf. Dr. Lenuța Alboie  
adria@info.uaic.ro



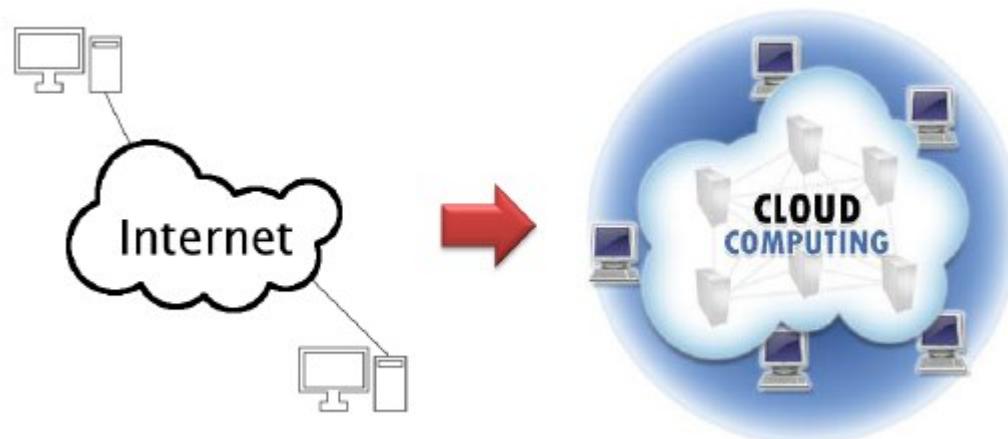
# Content

- Cloud Computing?
  - Definitions
  - Characteristics
- Services in cloud
- Deployment models

# Cloud Computing | Definitions

*Whatis.com*

- *“The name cloud computing was inspired by the cloud symbol that's often used to represent the Internet in flowcharts and diagrams. Cloud computing is a general term for anything that involves delivering hosted services over the Internet.”*



# Cloud Computing | Definitions

## Wikipedia

- “*Cloud computing is Internet-based computing, whereby shared resources, software, and information are provided to computers and other devices on demand, like the electricity grid.*”



- “*Cloud computing is a style of computing in which dynamically scalable and often virtualized resources are provided as a service over the Internet.*”

# Cloud Computing | Definitions

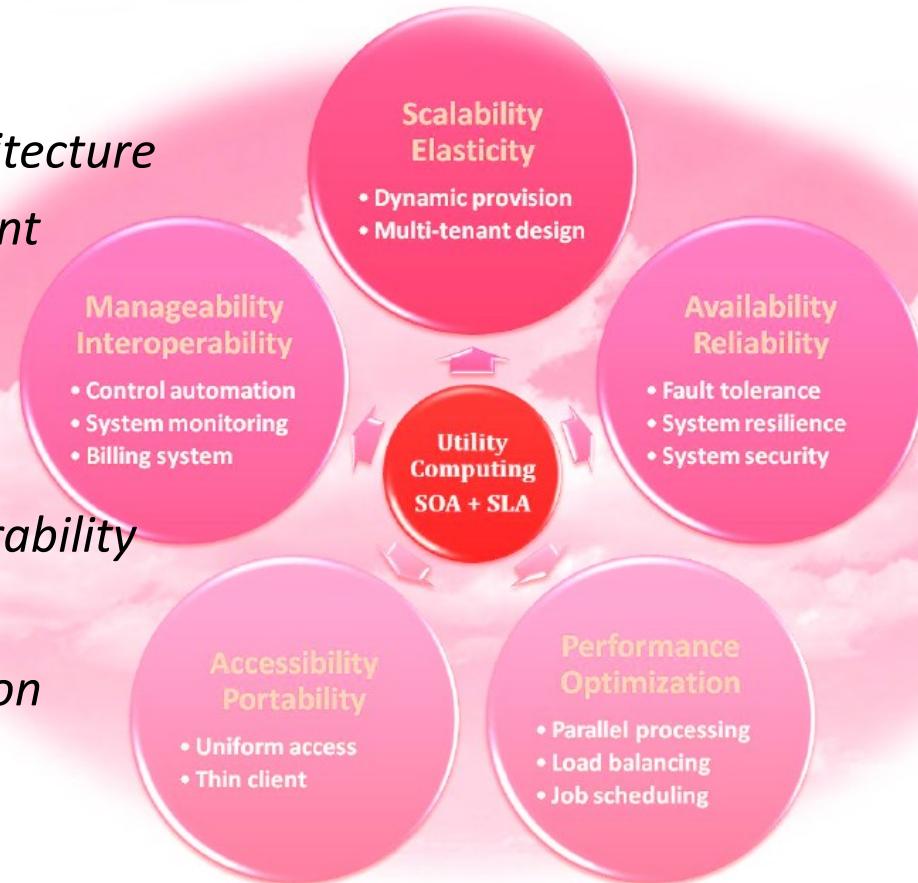
Buyya



- “A Cloud is a type of parallel and distributed system consisting of a collection of interconnected and virtualized computers that are dynamically provisioned and presented as one or more unified computing resources based on service-level agreements established through negotiation between the service provider and consumers.”<sup>5</sup>

# Cloud Computing

- Key aspects
  - *Utility Computing*
  - *SOA – Service Oriented Architecture*
  - *SLA – Service Level Agreement*
- Proprieties and characteristics:
  - *scalability and elasticity*
  - *availability and reliability*
  - *manageability and interoperability*
  - *accessibility and portability*
  - *performance and optimization*
- Used technics:
  - Virtualization
  - Parallel and distributed computing
  - Web Services



# Cloud Computing

- **Utility Computing** (Course 2)
  - ....
  - Example:
    - 2006, Sun Grid Compute Utility
      - Offer computing power in pay-per-use manner (1\$/CPU/hour)
    - 2009 , Sun: Open Cloud Platform
      - Sun Cloud Storage Service and Sun Cloud Compute Service
  - *utility computing* can be successful if ... => there is an accessible interface that can be easily understood and explored by developers

# Cloud Computing

- What is a Web Service?

## Definitions

- “Web service is self-describing and stateless modules that perform discrete units of work and are available over the network”
  - “Web service providers offer APIs that enable developers to exploit functionality over the Internet, rather than delivering full-blown applications”
- **SOA - Service Oriented Architecture**
    - A service collection that communicates
    - Contains design principles that are used in development and integration

# Cloud Computing

***“A cloud needs an access API”***

- An API enables the use of services
- Example:
  - Amazon's EC2 API is an *SOAP API- and HTTP Query-based*, with operations such us: creation, storage, management for AMI (Amazon Machine Images)
  - Kenai Cloud API (Sun) is an REST API used to create and manage various resources in cloud (computing, storage, network components)
- Security mechanism are used for authorized access (e.g. Amazon – X.509)
- Obs. There are no standards for cloud APIs

# Cloud Computing

- **QoS (Quality of Service)** “*is a set of technologies for managing network traffic in a cost effective manner to enhance user experiences for home and enterprise environments*” (definitions for a computer network)
- QoS includes:
  - End customer evaluations
  - Evaluations related to technical aspects (e.g. error rates, bandwidth, transmission delay, availability)
- **SLA** – “*Service-level agreement is a contract between a network service provider and a customer that specifies, usually in measurable terms (QoS), what services the network service provider will furnish.*”
  - There are metrics that assure performance (upload/download time, RTT, transfer rate)
  - Management details
  - Penalties if there were problems
  - Security aspects

# Cloud Computing

Scalability  
Elasticity

- Dynamic provision
- Multi-tenant design

## Properties:

- *Scalability*
  - Represents a characteristic of a system, network or process which indicates the ability to easily handle the increase of data management
- *Elasticity*
  - Represents an infrastructure ability to adapt to the requirements based on real-time analysis methodologies of the entire system
  - *Elasticity* is an ability to increase the system performance when the demands are high and to decrease the performances when demands are low
- *How to get it?*
  - **Dynamic Provisioning**
  - **multi-tenant design**

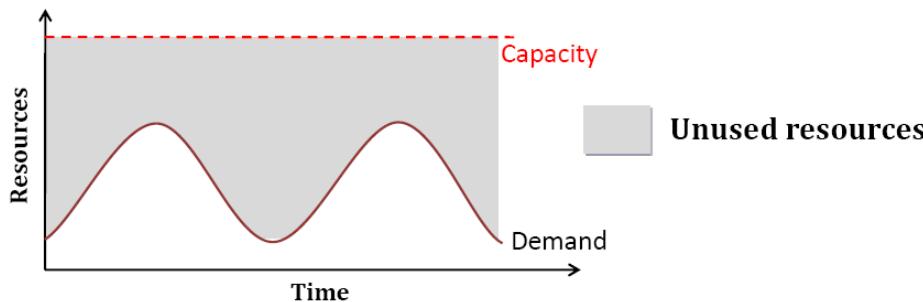
# Cloud Computing

Scalability  
Elasticity

- Dynamic provision
- Multi-tenant design

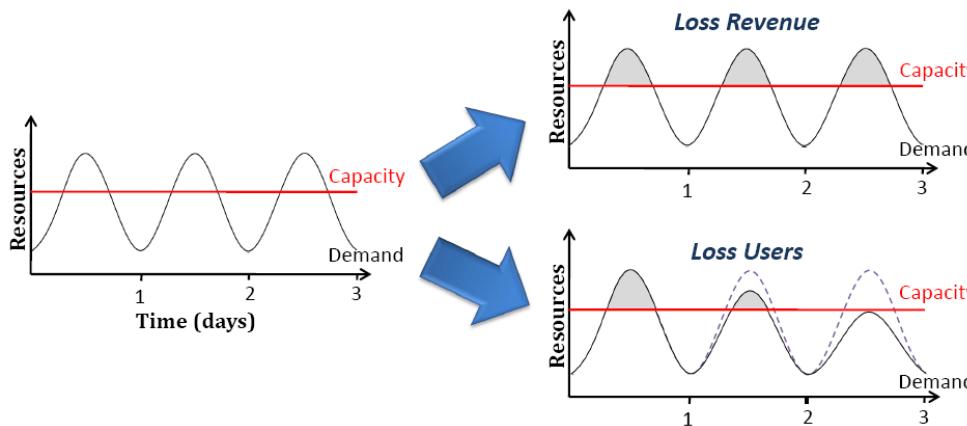
## Dynamic Provisioning:

- Traditional model
  - Problem 1: Overestimation of the used resource



- Problem 2: Underestimation of the used resources

Example: Twitter, LA Times etc...



## Scalability Elasticity

- Dynamic provision
- Multi-tenant design

## Dynamic Provisioning

**"A cloud needs elasticity: scaling your application as demand rises and falls."**

Examples from Amazon EC2: we want to assure scalability from 2 to 20 instances =>elasticity). If CPU usage exceeds 80 % from its capacity, a new instance is add; if it is lower than 40% for 10 minutes , an instance is removed

### CreateLoadBalancer:

```
AvailabilityZones = us-east-1a
LoadBalancerName = MyLoadBalancer
Listeners = lb-port=80,instance-port=8080,protocol=HTTP
```

### CreateLaunchConfiguration:

```
ImageId = myAMI
LaunchConfigurationName = MyLaunchConfiguration
InstanceType = m1.small
```

### CreateAutoScalingGroup:

```
AutoScalingGroupName = MyAutoScalingGroup
AvailabilityZones = us-east-1a
LaunchConfigurationName = MyLaunchConfiguration
LoadBalancerNames = MyLoadBalancer
MaxSize = 20
MinSize = 2
```

### CreateOrUpdateScalingTrigger:

```
AutoScalingGroupName = MyAutoScalingGroup
MeasureName = CPUUtilization
Statistic = Average
TriggerName = MyTrigger1a
Namespace = AWS/EC2
...
LowerThreshold = 40
LowerBreachScaleIncrement = -1
UpperThreshold = 80
UpperBreachScaleIncrement = 1
BreachDuration = 600
```

# Cloud Computing

Scalability  
Elasticity

- Dynamic provision
- Multi-tenant design

## Multi-tenant design

- Principle in software architecture where a single software instance runs on a server and multiple clients are served (tenants)
- In a multi-tenant architecture applications share in a virtual manner data and configurations => each client (e.g. company, end-user,...) use a personalized instance of a virtual application
- Multi-tenant applications
  - are provided with a certain level of customization that can be adapted to customer requirements
  - can provide a certain level of security and resistance
- > It facilitates the *release management* process
- ... Docker....Virtualization (next course)

# Cloud Computing

Availability  
Reliability

- Fault tolerance
- System resilience
- System security

## Properties:

- *Availability*
  - The degree to which a system / subsystem equipment is in a functional state and is ready to work in any time
  - Cloud systems require high availability (~ 99.999%)
  - If the supplier is subject to a DOS attack, will client's critical systems collapse?

Service and Outage	Duration	Date
S3 outage: authentication service overload leading to unavailability	2 hours	2/15/08
S3 outage: Single bit error leading to gossip protocol blowup.	6-8 hours	7/20/08
AppEngine partial outage: programming error	5 hours	6/17/08
Gmail: site unavailable due to outage in contacts system	1.5 hours	8/11/08

Examples: AWS, AppEngine and Gmail indisponibility

[<http://www.eecs.berkeley.edu/Pubs/TechRpts/2009/EECS-2009-28.htm>]

15

# Cloud Computing

Availability  
Reliability

- Fault tolerance
- System resilience
- System security

## Properties:

- *Reliability*
  - The ability of a system or component to perform the duties required under specified conditions for a specified period of time.
- *How to obtain Availability and Reliability?*
  - *Fault tolerance*
  - *Resilience*
  - Security

# Cloud Computing

Availability  
Reliability

- Fault tolerance
- System resilience
- System security

## Fault tolerance

- Fault tolerance is the property that enables a system to continue operating properly in the event of failure of some of its components.
- Measure the functionality level: the decrease or increase in a proportional degree with the failure severity (in naïve systems design even a small failure can collapse the entire application)
- Characteristics:

- There is no SPOF



- Preventing: The system should function during the repairing processes
- Error detection and isolation of the failed component
- Fault isolation to prevent the spread(*fault containment*)
- The existence of check points allowing restoration

# Cloud Computing

Availability  
Reliability

- Fault tolerance
- System resilience
- System security

## Resilience

- The ability to provide and maintain an acceptable level of service in case of error in order to offer usual services
- Indicates the system's ability to return to the original state as quickly as possible after errors have occurred
- Events
  - Loss of power supply
  - Corrupted Database
  - Natural Disasters
- Strategies
  - Backup
  - Additional sources of energy, Uninterruptible Power Supply (UPS)
  - ...



# Cloud Computing

***“A cloud needs servers on a network, and they need a home”*** -> Data center

Aspects:

- Structure: Layout, maintenance, physical and logical security
- Scalability provided at optimal costs
- Why?
  - 1.2% of US electricity consumption
  - Acquisition of hardware in huge quantities

=>The possibility of negotiation

Ex. Amazon: 90 million on 50,000 servers from Rackable / SGI in 2008 versus \$ 215 million
- Research for reduction of consumption(e.g. Google - *dynamic voltage/frequency scaling*)

# Cloud Computing

- “*A cloud needs servers on a network, and they need a home*”



Photograph of Google's top-secret Dalles, OR data center, built near the Dalles Dam for access to cheap power. Note the large cooling towers on the end of each football-sized building on the left. These towers cool through evaporation rather than using more power-hungry chillers.  
Source: Melanie Conner, *New York Times*.

[The Cloud at Your Service, Jothy Rosenberg, Arthur Mateos, ]

# Cloud Computing

Availability  
Reliability

- Fault tolerance
- System resilience
- System security

## Cloud Computing Security

- It is a subfield of computer security and computer networks
- Reflects a broad set of policies, technologies and controls developed to protect data, applications and cloud infrastructure
- Aspects
  - Privacy:
    - Does Sensitive data remain confidential? (Is the Cloud provider honest?)
    - Integrity
      - How do I know cloud provider performs the correct calculation? Is my data stored without being altered?
  - Privacy
    - Cloud stores data from multiple clients and "anyone" can run data mining algorithms ....

# Cloud Computing

Availability  
Reliability

- Fault tolerance
- System resilience
- System security

## Cloud Computing security

### Challenges

- Increasing the area vulnerable to attack
  - Data is stored and handled outside the organization
  - Attackers can target communication link between the client and cloud provider
  - Cloud provider employees may be subject to phishing attacks
- Auditing
  - Estimation of risk, prevention, detection, response to attacks - is doable because the data is outside the organization
- Legal Issues and reliable transfer
  - Who is responsible for regulatory compliance?
  - What happens in the case of cloud services subcontracting ...?

# Cloud Computing

Availability  
Reliability

- Fault tolerance
- System resilience
- System security

## Cloud Computing Security

- Cloud Computing is a **security nightmare** and it can't be handled in traditional ways  
(Course 13)



John Chambers  
CISCO CEO

# Cloud Computing

## Properties:

- *Manageability*
  - Issues closely related to enterprise management features tailored to cloud computing systems
- *Interoperability*
  - A system property to own interfaces that are fully understood and that enable present and future interaction with other systems, without access restrictions or implementation limits

## How to get it?

- *Control automation*
- *System monitoring*



### Manageability Interoperability

- Control automation
- System monitoring
- Billing system

# Cloud Computing

Manageability  
Interoperability

- Control automation
- System monitoring
- Billing system

## *Control automation and Monitoring*

- *Autonomic Computing* – development of systems capable of self-management, to overcome management's increasing complexity of computing systems, now and in the future
- *Autonomic Components* – can be seen as some control entities: sensor (self-monitoring), self-regulating mechanisms, with some knowledge and planning abilities based on rules/policies; the action is performed according to its own condition and environment
- What can be monitored:
  - Status hardware level (physical and virtualized)
  - Measuring performance parameters resources
  - Network access patterns
  - System Logs

# Cloud Computing

## Billing

- Users pay what they use
- Cloud Service Provider
  - Monitors the system (previous slide )
    - Automatically computes the costs for used services and forwards the payment request

Manageability  
Interoperability

- Control automation
- System monitoring
- Billing system



# Cloud Computing

Performance Optimization

**Properties:** Performance and Optimization

- Parallel processing
  - Hardware approaches (*multi-core* systems, *vector processor* systems, distributed systems(e.g. Cluster Computing, Grid Computing)
  - Software approaches (programming languages for parallel computing, platforms that provide automatic parallelization)
- Load balancing
  - The technique implies a uniform distribution load on two or more computers, network links, CPUs, hard drives or other resources, in order to best use resources to maximize traffic to decrease response time and to avoid overloading
- Tasks planning
  - A job scheduler is a software system responsible for executions in the background (~*batch processing*)
    - Intensive computing tasks, tasks used in complex processing dependent on each other
    - Approaches: pre-defined workflows, automatic configuration system

# Cloud Computing

Accessibility  
Portability

- Uniform access
- Thin client

## Properties

- Accessibility
  - Describes the level to which a product, device, service or environment is accessible to more customers
- Portability
  - It is the ability to access the service using any device, anywhere, continuously and in an adaptive manner to the resource availability
- *How can you get it?*
  - Uniform access
  - *Thin client*

# Cloud Computing

Accessibility  
Portability

- Uniform access
- Thin client

## Uniform access

- Using different OS, different platforms should provide access to cloud services



## Thin client

- A device or program that depends on another computer system for performing various computational roles
- Versus *fat client*
- Features
  - Devices at affordable prices
  - A greater variety of end devices
  - Simplification at the client level

# Cloud Computing | Services



# Cloud Computing | Services

**Question: Would you move to a town and look for solutions to live?**

- Would you build a new house?



- Would you buy a cold dark shell house?



- Would you live in a hotel?



# Cloud Computing | Servicii

**Question: How to build an IT department?**

- IaaS (Infrastructure as a Service)
  - You rent a virtualized infrastructure and you can build an IT system that you can fully control
- PaaS (Platform as a Service)
  - You develop an IT system on an existing cloud platform, without care of low level resources
- SaaS (Software as a Service)
  - You use the existing IT systems offered by a cloud provider and you are not aware of technical aspects

# Cloud Computing

## General architecture

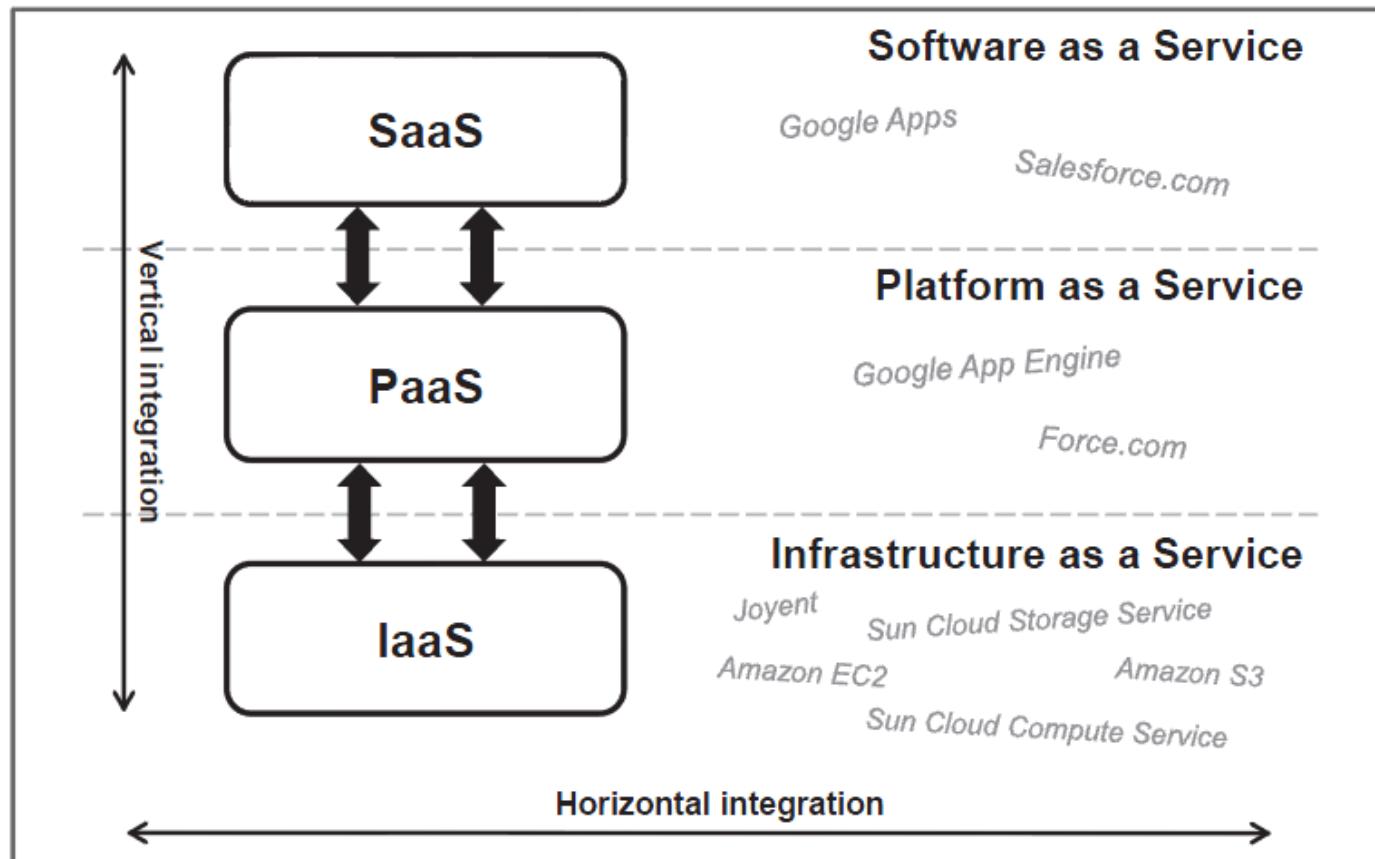
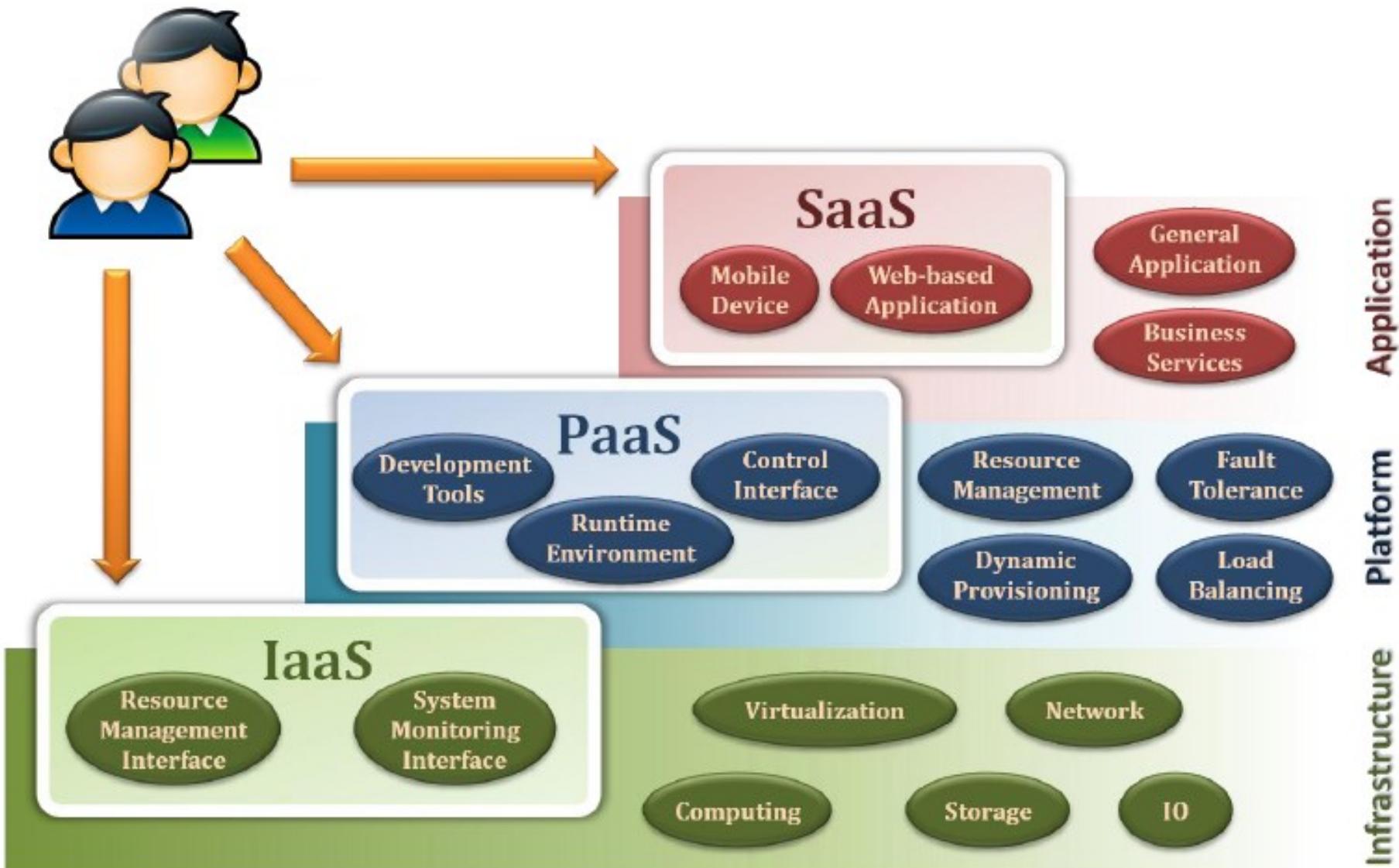


Figure: First IaaS, PaaS, SaaS providers

# Cloud Computing | Services



# Cloud Computing

## Cloud Computing - Typical Architecture

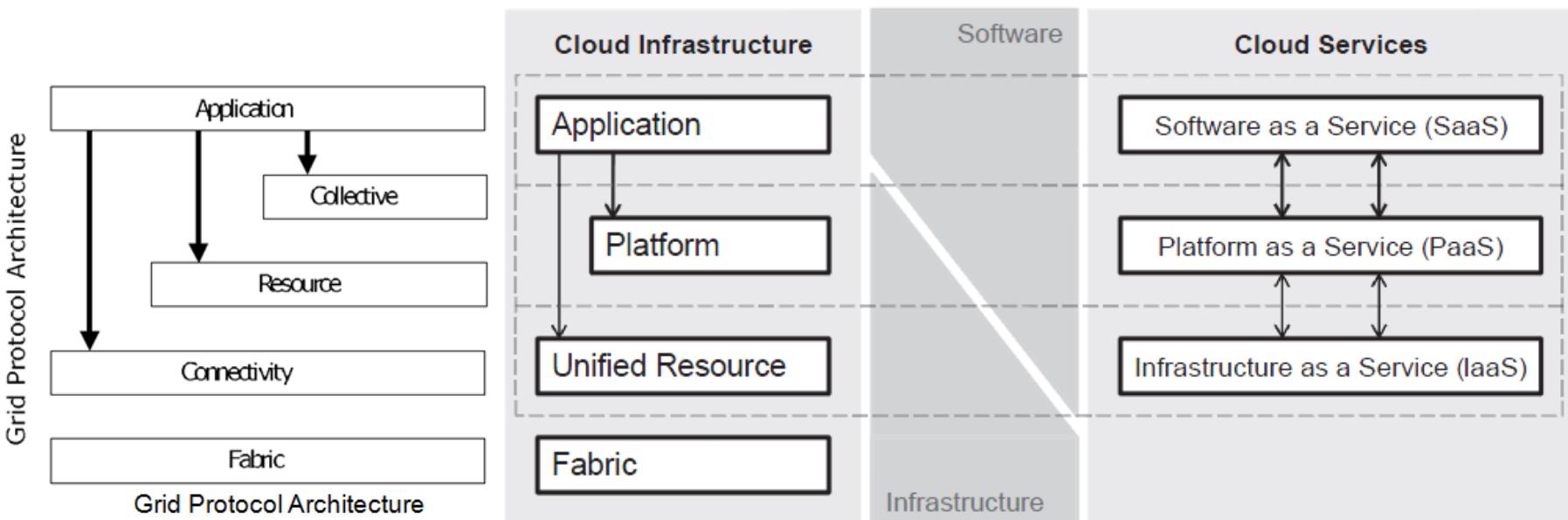


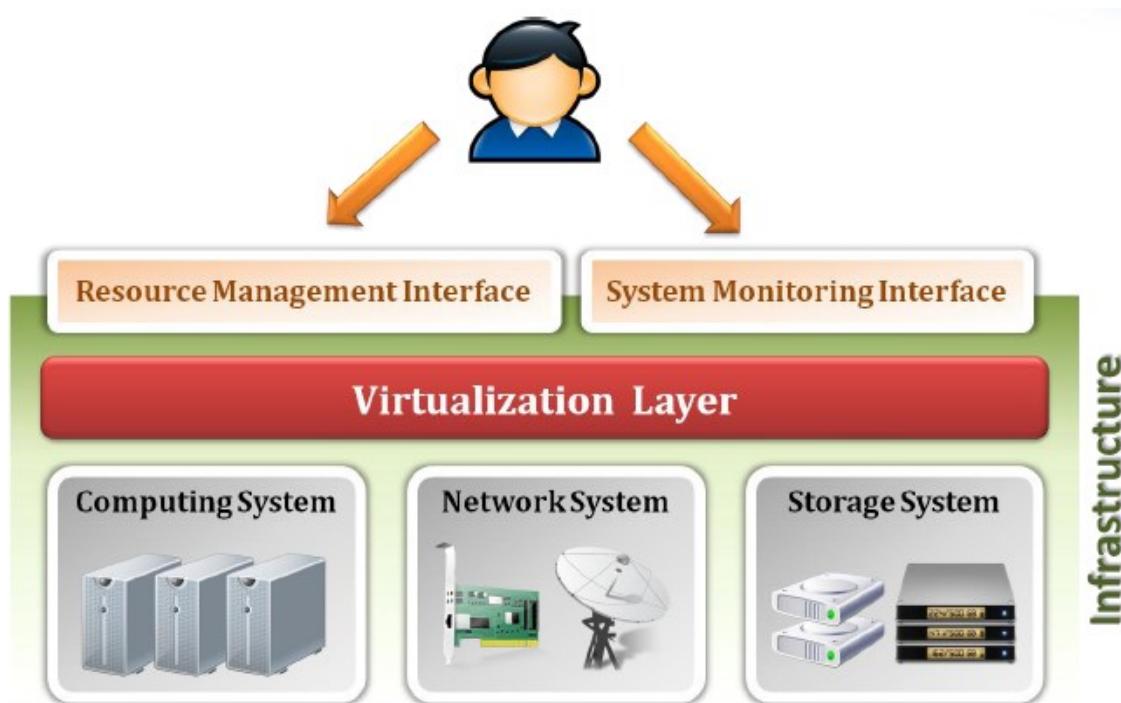
Figure: Cloud Architecture and Cloud Services (Foster, 2008)

# Cloud Computing | Services

## IaaS - Infrastructure as a Service

**Note:** *Fabric or Hosting Platform* - Provides the physical machine, operating system, the communication network, storage systems and virtualization software

- IaaS abstracts Fabric level and offer virtualized infrastructure (versus *raw hardware*) as a service , for processing, storage and communication





# Cloud Computing | Services

## IaaS - Infrastructure as a Service

- The consumer has no access to the fabric, but has control over operating systems, storage, application development and configurations related to network
- Example:
  - Amazon
    - Elastic Compute Cloud (EC2) for processing
    - Simple Storage Service (S3) for storage
  - Eucalyptus: Cloud open source implementation compatible with EC2 (used for on-premise (private) IaaS infrastructure)
  - OpenStack
  - ...



# Cloud Computing | Services

## IaaS - Infrastructure as a Service

- Key technology: virtualization .... recently -> Dockers
- Virtualization - is the emulation of one or more workstations within a single physical computer
- The hardware resources are transformed and virtualized (CPU, RAM, hard disk, network controller) => functional virtual machines that can run its own OSs

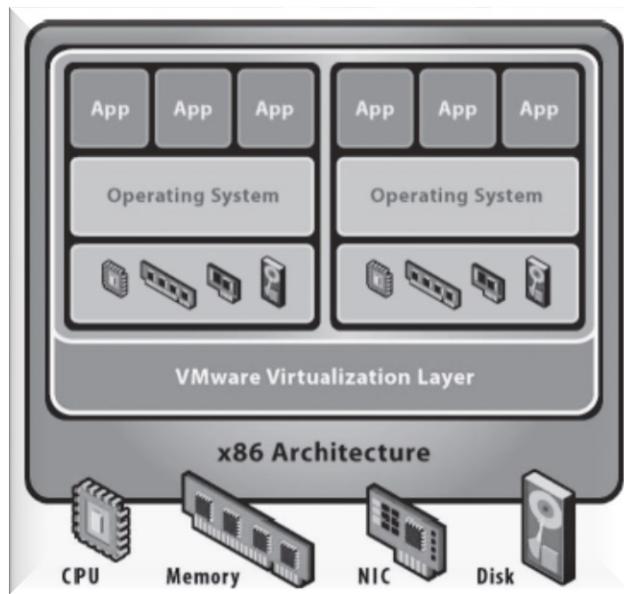


Figure. Architecture of virtual machines using VMware on x86

- 1999 VMware introduced the first virtualization application for x86 systems

[Cloud Computing  
Virtualization  
Specialist Complete]

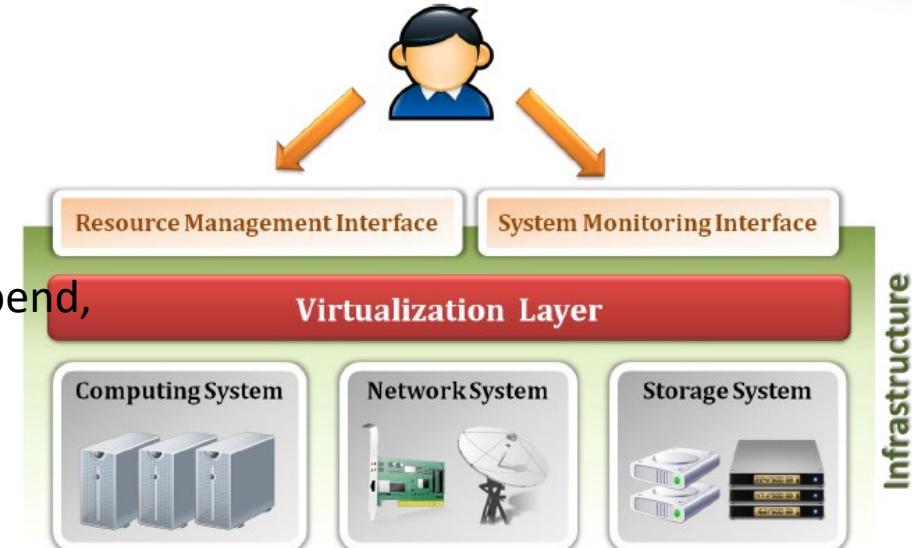
# Cloud Computing | Services

## IaaS - Infrastructure as a Service

IaaS provider can offer:

### RMI (Resource Management Interface)

- *Virtual Machine* – operations: create, suspend, reboot
- *Virtual Storage* – operations: allocation of space, free space , writing or reading
- *Virtual Network* – operations: allocation of IP addresses, domain registration, establish connections etc.



### SMI (System Monitoring Interface) – examples of metrics for monitoring::

- *Virtual Machine*: CPU load, memory usage, IO load, internal network load.
- *Virtual Storage*: using virtual space, duplication, bandwidth to access the storage device
- *Virtual Network*: virtual network bandwidth, level of network load

# Cloud Computing | Services

## IaaS - Infrastructure as a Service

Virtualization supports properties as: *scalability and elasticity*

- *availability and reliability*
- *manageability and interoperability*



# Cloud Computing | Services

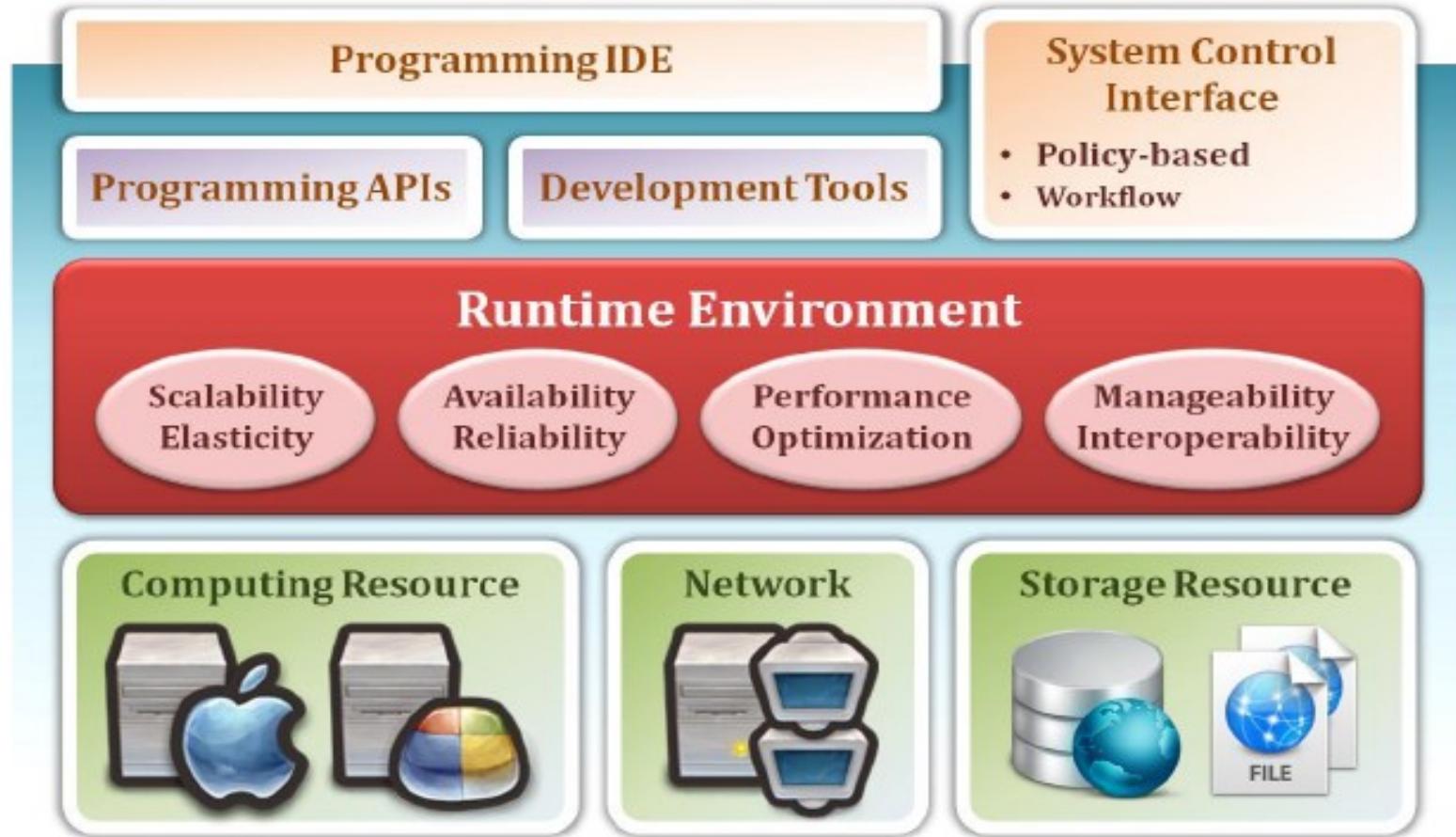
## PaaS - Platform as a Service

- Designed for software developers who develop applications according to the specifications of a platform, without involving factors related to hardware infrastructure
  - Example: The platform is one that dynamically allocates resources if the application is widely used
- The consumer does not have access to the Management of cloud infrastructure (network, server, operating system or storage), but has control over the applications developed and on some configurations related to application hosting
- PaaS provides a standardized interface and a platform for SaaS level
- Example:
  - Google AppEngine – the applications run using Google infrastructure
  - Microsoft Windows Azure – PaaS level
  - ....

# Cloud Computing | Services

## PaaS - Platform as a Service

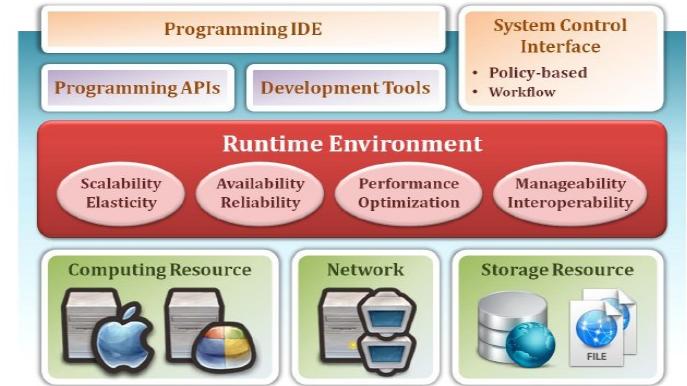
- Runtime Environment: collection of software services



# Cloud Computing | Services

## PaaS - Platform as a Service

- **Programming IDE**
  - IDE includes all the functionality that allows access to running environment and development tools, test environment and so on
  - API runtime environment varies between cloud providers, but there are common operations: computing, storage and communication.
- **System Control Interface**
  - Police-Based Control: described as a principle or rule that help in making decisions for final output
  - Workflow-Control:
    - Describes the steps involved in resources installation and configuration
    - The demon that process the workflow offer in an efficient way cloud resources



# Cloud Computing | Services

## PaaS - Platform as a Service

*Runtime Environment* supports properties as:

*performance and optimization*

- *scalability and elasticity*
- *availability and reliability*
- *manageability and interoperability*



# Cloud Computing | Services

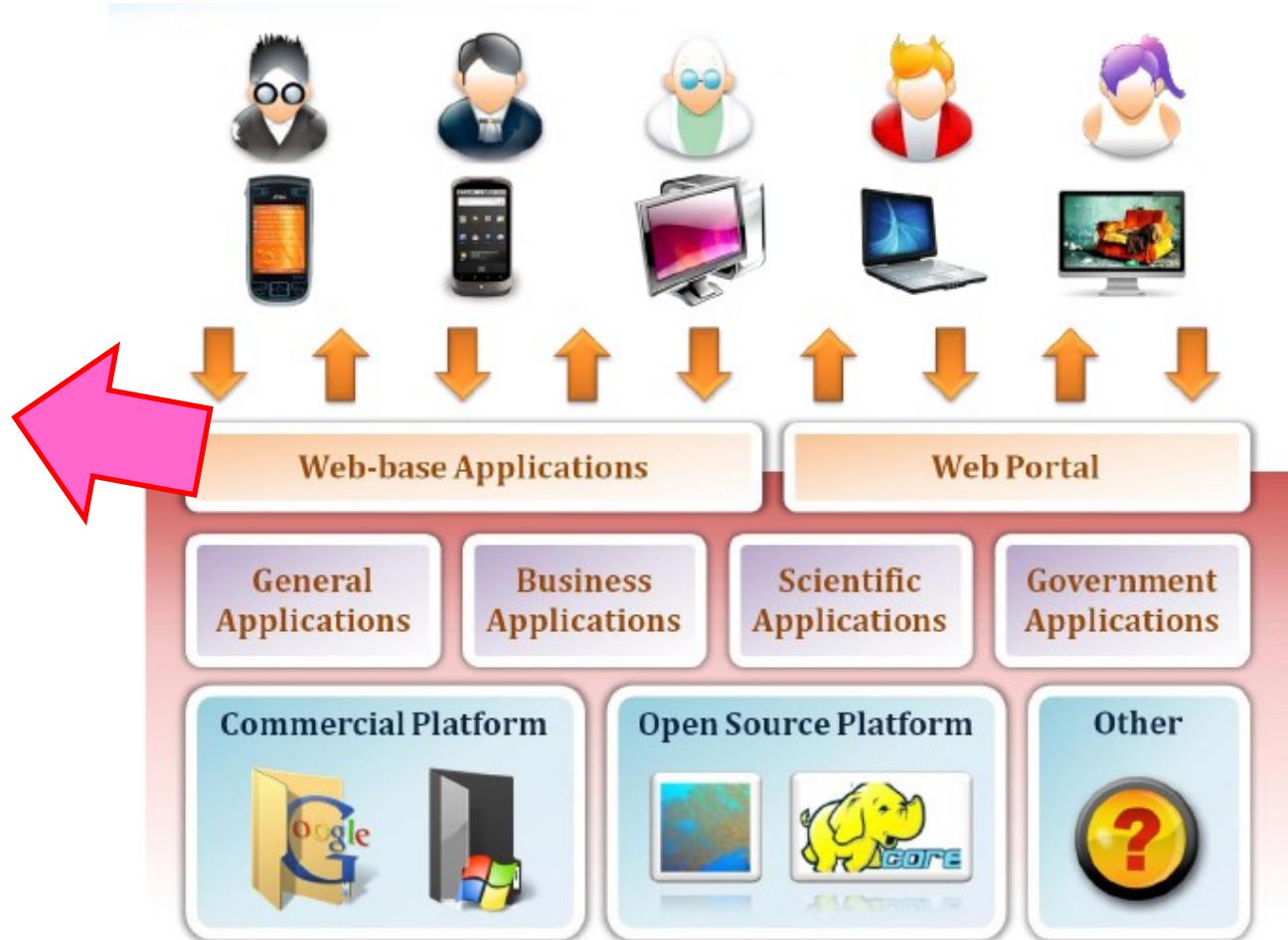
## SaaS - Software as a Service

- The highest visible level from the Cloud for end-users; it provides software applications exposed as Web interfaces or Web services;
- Applications are available on a wide variety of thin clients
- “SaaS is software that is owned, delivered and managed remotely by one or more providers and that is offered in a pay-per-use manner” (Mertz 2007)
- Usually SaaS users do not know the details of the infrastructure
- Examples:
  - Google Apps (Google Mail, Google Drive, Google Spreadsheets, ....),
  - Live Mesh (Microsoft),....
  - SalesForce.com
  - EyeOS (<http://www.eyeos.com/>)
  - ....

# Cloud Computing | Services

## SaaS - Software as a Service

Assured proprieties:  
*accessibility* and  
*portability*



# Cloud Computing

## *Deployment models*

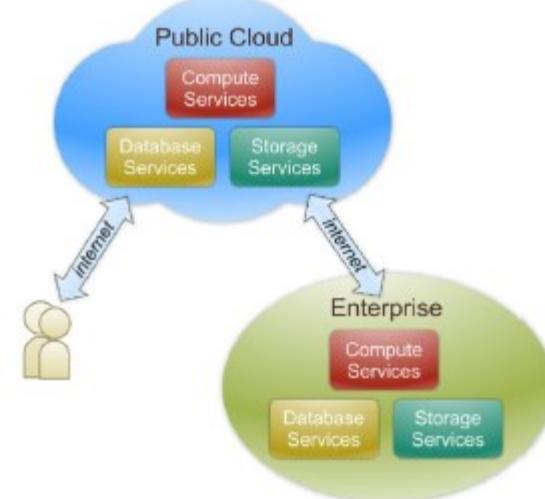
- In terms of data center owner
  - Public Cloud
  - Private Cloud
- In terms of how many cloud environments are integrated (*multiple-Cloud environments*)
  - *Community Cloud*
  - *Hybrid Cloud*

# Cloud Computing

## *Deployment models*

### Public Cloud

- “A Public Cloud is data center hardware and software run by third parties, e.g. Google and Amazon, which expose their services to companies and consumers via the Internet” (IBM, 2009)
- It is available in pay-per-use manner
- Known as the external cloud and multi-tenant cloud
- Features
  - Homogeneous infrastructure
  - General rules
  - Shared resources and multi-tenancy
  - Rented infrastructure
  - It involves operating expenses

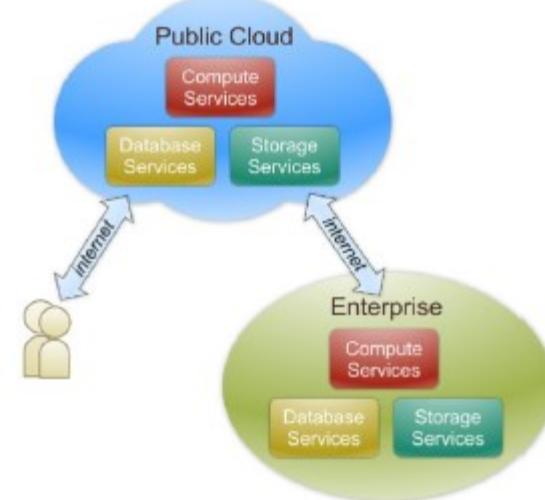


# Cloud Computing

## *Deployment models*

### Private Cloud

- Motivation: limiting the risks associated with a Public Cloud
- Cloud infrastructure is used for a single organization; the cloud management can be done by another organization
- Also called internal cloud or on-premise cloud, is based on virtualization of the existing infrastructure from the organization => more efficient use of resources
- Features
  - Heterogeneous infrastructure
  - Personalized and tailored policy
  - Dedicated resources
  - Infrastructure in-house
  - Involve capital expenditure



# Cloud Computing

## *Deployment models*

### Community Cloud

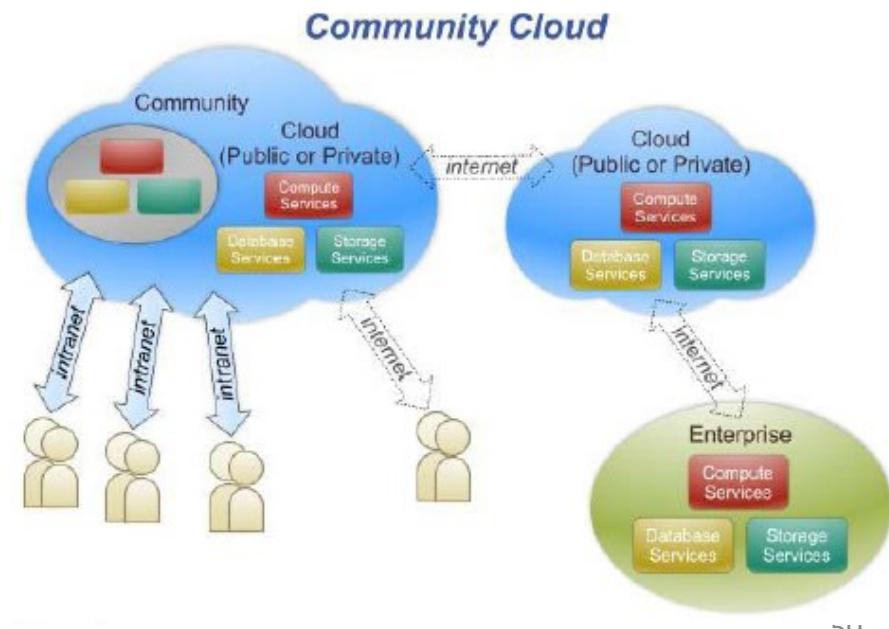
- Multiple organizations sharing their cloud infrastructure to achieve a common goal
- Also called Federation of Clouds, each cloud is independent, but can interoperate (exchange data and computing resources) with other clouds through standard interfaces
- Example: RESERVOIR
- Standardization- > Open Cloud Computing Interface Working Group

Computing Interface Working Group

(<http://occi-wg.org/about/specification/>)

- Open Grid Forum (OGF);

DMTF - <http://dmtf.org/standards/cloud>

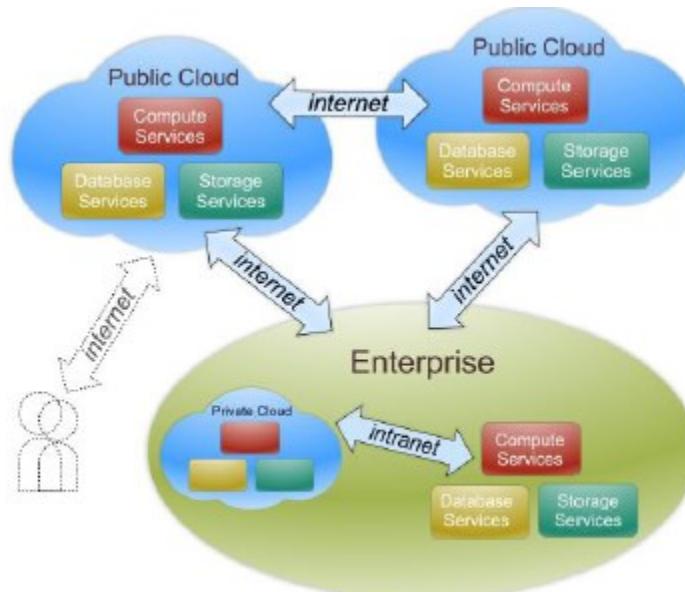


# Cloud Computing

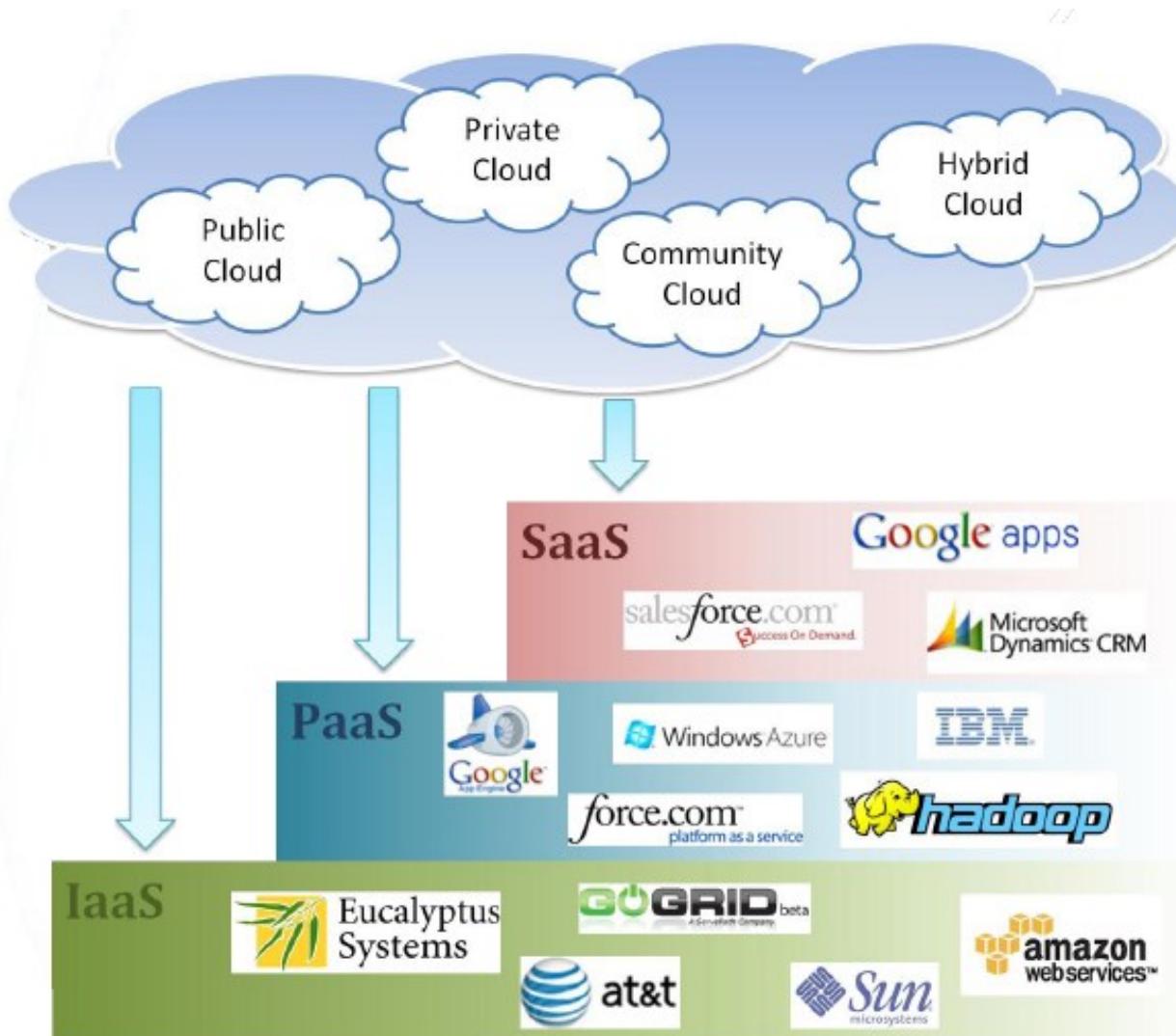
## *Deployment models*

### Hybrid Cloud

- The infrastructure consists of multiple clouds (private, community, public) that remain unique entities , but they are linked together with standardized or proprietary technologies that ensure data and application portability (e.g. in situations where load balancing is required)



# Cloud Ecosystem



# Bibliography

- Massimo Cafaro, Givani Aloisio, Grids, Clouds and Virtualization, 2011
- Katarina Stanoevska Slabeva, Thomas Wozniak, Grid and Cloud Computing - A Business Perspective on Technology and Applications, 2010, Editors Santi Ristol, Springer-Verlag Berlin Heidelberg
- Open Cloud Computing Interface - <http://occi-wg.org/>
- RESERVOIR - <http://ercim-news.ercim.eu/en83/special/reservoir-a-european-cloud-computing-project>
- DMTF - <http://dmtf.org/standards/cloud>
- LIBVIRT - <http://libvirt.org/apps.html>
- Chow et al., Cloud Computing: Outsourcing Computation without Outsourcing Control, 1<sup>st</sup> ACM Cloud Computing Security Workshop, November 2009
- Foster, Zhao, Raicu and Lu, Cloud Computing and Grid Computing 360-Degree Compared, 2008
- Above the Clouds: A Berkeley View of Cloud Computing, Technical Report No. UCB/EECS-2009-28, <http://www.eecs.berkeley.edu/Pubs/TechRpts/2009/EECS-2009-28.htm>
- <http://my.ss.sysu.edu.cn/courses/cloud/>
- <http://blogs.idc.com/ie/?p=730>
- <http://www.slideshare.net/woorung/trend-and-future-of-cloud-computing>
- <http://ganglia.sourceforge.net/>
- <http://www.focus.com/briefs/top-10-cloud-computing-trends/>

# Bibliography

- NIST (National Institute of Standards and Technology).  
<http://csrc.nist.gov/groups/SNS/cloud-computing/>
- <http://aws.amazon.com/free/>
- <https://www.windowsazure.com/en-us/community/education/program/overview/>
- <http://www.ibm.com/developerworks/java/library/j-gaestorage/index.html?ca=drs->
- M. Armbrust et. al., “Above the Clouds: A Berkeley View of Cloud Computing,” Technical Report No. UCB/EECS-2009-28, University of California at Berkeley, 2009.
- R. Buyya et. al., “Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility,” Future Generation Computer Systems, 2009.
- Cloud Computing Use Cases. <http://groups.google.com/group/cloud-computing-use-cases>
- Cloud Computing Explained. <http://www.andyharjanto.com/2009/11/wanted-cloud-computing-explained-in.html>
- Multiple materiale si imagini au fost preluate de pe Internet

# Summary

- Cloud Computing?
  - Definitions
  - Characteristics
- Services in cloud
- Deployment models

“Alexandru Ioan Cuza” University of Iasi  
Faculty of Computer Science

**Questions?**



Universitatea “Alexandru Ioan Cuza”  
Facultatea de Informatică

Prof. Dr. Lenuța Alboiae  
adria@info.uaic.ro

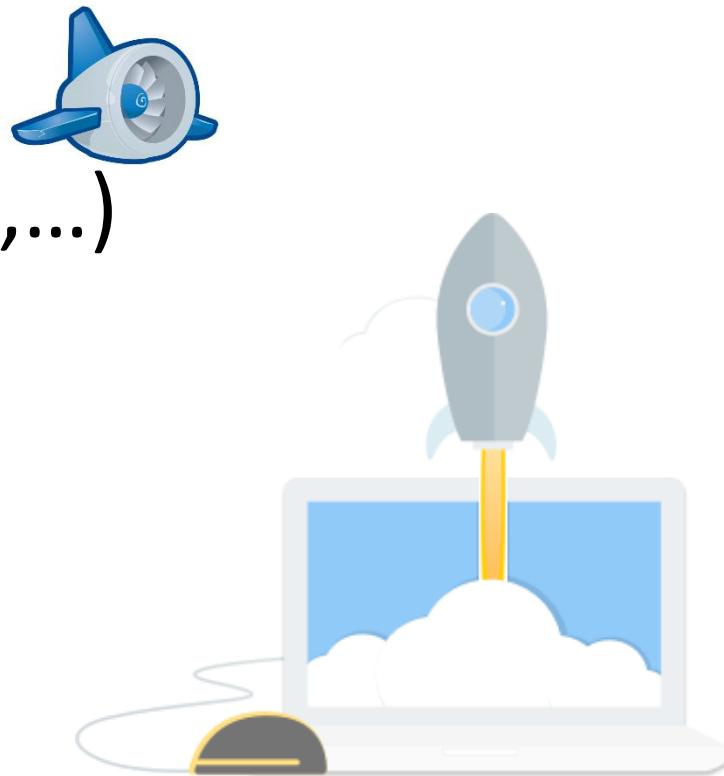


# Google in Cloud



# Cuprins

- Google in Cloud
  - ...servicii
  - Instrumente (GWT, GAS,...)
  - Caracteristici
  - Aspecte arhitecturale
  - Concluzii



# Google in Cloud | ...pasi

- ? Google face managementul celor mai mari ferme de servere din lume  
Initial
  - Intreg continutul oferit era disponibil; se baza doar pe suportul oferit de publicitate  
(Servicii: Google Maps, Google Finance, Google Voice)
- Mantra: “*it's free to the consumer*” ☺
  - 15GB pentru stocare per cont
  - Google Apps – free hosting pentru serverul de e-mail (cu propriul nume de domeniu), Google Talk, Google Calendar, Google Drive, Google Sites <- Rajen Sheth
  - Google Apps pentru Educatie
  - Urmind modelul Apple: Apps Market-  
[www.google.com/enterprise/marketplace](http://www.google.com/enterprise/marketplace)
  - Pentru dezvoltatori: versiuni free ale serviciilor, e.g. Google App Engine™  
<http://code.google.com/appengine>



# Google

- Google App pentru Business

Google Cloud Platform Customers

Discover why the world's most innovative organizations are choosing Google Cloud Platform

The grid contains the following logos:

- Row 1: HTC, FIS, PocketGems, Spotify, Coca-Cola, Best Buy, KHANACADEMY
- Row 2: ATOMIC FICTION, ocado, Heathrow, Philips, Niantic
- Row 3: EVERNOTE, HANACADEMY

Get your business  
on Google for free

## WHO'S USING CLOUD PLATFORM?

Over 4 million applications are built on Cloud Platform

## GCP FOR STARTUPS PACKAGES

# Google

## – Google App pentru Business/ *G Suite*

=>  
multi utilizatori  
*business*  
(preturi in 2016)



Google Apps

\$5

per user per month

or \$60 per user per year plus tax

Get started

Google Apps  
with unlimited storage and Vault

\$10

per user per month

or \$120 per user per year plus tax

Get started

### Google Apps for Work includes:

- Business email addresses (name@yourcompany.com)
- Video and voice calls
- Integrated online calendars
- 30GB of online storage for file syncing and sharing
- Online text documents, spreadsheets and slides
- Easy to create project sites
- Security and admin controls
- 24/7 phone and email support

### Everything in Google Apps for Work plus:

- Unlimited Storage (or 1TB per user if fewer than 5 users)
- Advanced admin controls for Drive
- Audit and reporting insights for Drive content and sharing
- Google Vault for eDiscovery covering emails, chats, docs and files
- Easily search and export to different formats
- Archive all emails sent by your company
- Set message retention policies
- Place and enforce litigation holds on inboxes

# Google



## Cloud Price Leader

Google Cloud Platform gives you the best price to performance.  
Your cloud shouldn't break the bank and compromised  
performance should never be the only tradeoff.



Boot up in

35 seconds



Archive Restore

milliseconds

## Milliseconds matter

Google Compute Engine instances boot up in 35 seconds on average. Coldline delivers millisecond data availability for archive restore – other public clouds can take up to 5 hours<sup>1</sup>. Our Local SSDs offer 680,000 IOPS of sustained read performance – some other systems don't reach half of that IOPS. BigQuery can scan up to 35 billion rows, 20 TB of data, in seconds. Price and performance: we have both.

[<https://cloud.google.com/pricing/price-leader>]

# Google



## Do more for less

Pricing innovations like rightsizing and sustained-use discounts help deliver an average savings of 35% for many compute workloads. Our storage prices average 21% less than AWS for online storage workloads.<sup>1</sup>

[Get started for free](#)

[Contact sales](#)



\*21% sustained-use discounts and 18% rightsizing recommendations

[<https://cloud.google.com/pricing/price-leader>]

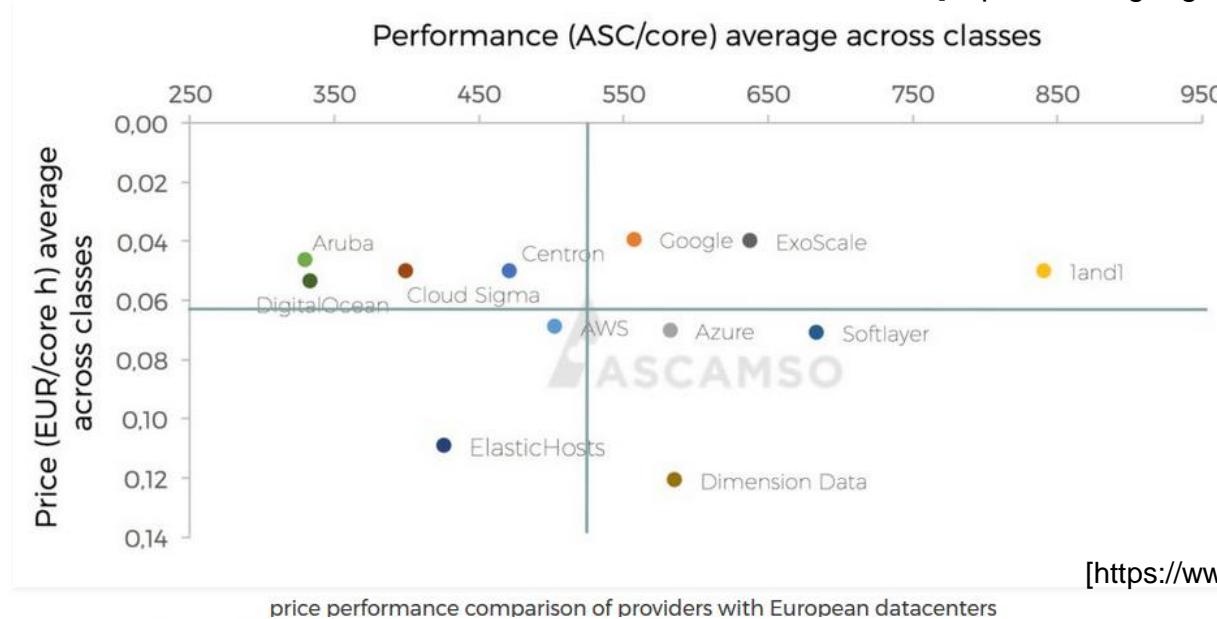
# Google

## Data center efficiency through DeepMind machine learning

Google has applied artificial intelligence to optimize power usage in Google data centers. The up to **40% reduction in electricity per month** needed for cooling is considered a phenomenal step forward in the industry. On average, a Google data center **uses 50% less energy** than a typical data center.



[<https://cloud.google.com/pricing/price-leader>]



[<https://www.ascamso.com/905-2/>]

# Google

## Google Web Toolkit – GWT

- <http://code.google.com/webtoolkit/gettingstarted.html>
- Set de instrumente open-source ce permite dezvoltarea de aplicatii Web complexe
- *"The GWT SDK contains the Java API libraries, compiler, and development server. It lets you write client-side applications in Java and deploy them as JavaScript."*
- ....
- *"GWT is a development toolkit for building and optimizing complex browser-based applications. Its goal is to enable productive development of high-performance web applications without the developer having to be an expert in browser quirks, XMLHttpRequest, and JavaScript. GWT is used by many products at Google, including AdWords, AdSense, Flights, Hotel Finder, Offers, Wallet, Blogger. It's open source, completely free, and used by thousands of developers around the world."*
- ....

# Google

**Google Apps Script** - <https://developers.google.com/apps-script/>

- Ofere posibilitatea automatizarii procesului de business, prin intermediul unor scripturi ce aduc un plus de functionalitate fata de suita de aplicatii deja existente
  - Legatura aplicatiilor Google cu servicii *third party*: un script poate trimite email-uri si o invitatie unei liste dintr-o baza de date MySQL
  - Crearea de functii specializate pentru spreadsheet-uri: realizarea de analize complexe asupra datelor din Google spreadsheets
  - Construirea unei interfete utilizator atractiva: o aplicatie interna dintr-o companie poate beneficia de o interfata construita cu GAS

*“With Apps Script, you can create add-ons for Google Sheets, Docs, or Forms, automate your workflow, integrate with external APIs, and more.”*

11 Google apps, 1 platform in the cloud



Increase the power of your favorite Google apps – like [Calendar](#), [Docs](#), [Drive](#), [Gmail](#), and [Sheets](#).

Apps Script lets you [do more with Google](#). All on a JavaScript platform in the cloud.

# Google Cloud Platform

*Hosting + Compute*



*App Engine*



*Compute Engine*

*Storage*



*Cloud Storage*



*Cloud Datastore*



*Cloud SQL*

*Big Data*



*BigQuery*

*Services*



*Cloud Endpoints*



*Translate API*



*Prediction API*



2015 - <https://cloud.google.com/>

# Google Cloud Platform

## Compute



*App Engine*



*Compute Engine*



*Container Engine*

## Storage



*Cloud Storage*



*Cloud Datastore*



*Cloud SQL*



*Cloud Bigtable*

## Big Data



*BigQuery*



*Cloud Dataflow*



*Cloud Dataproc*



*Cloud Pub/Sub*

## Services



*Cloud Endpoints*



*Translate API*

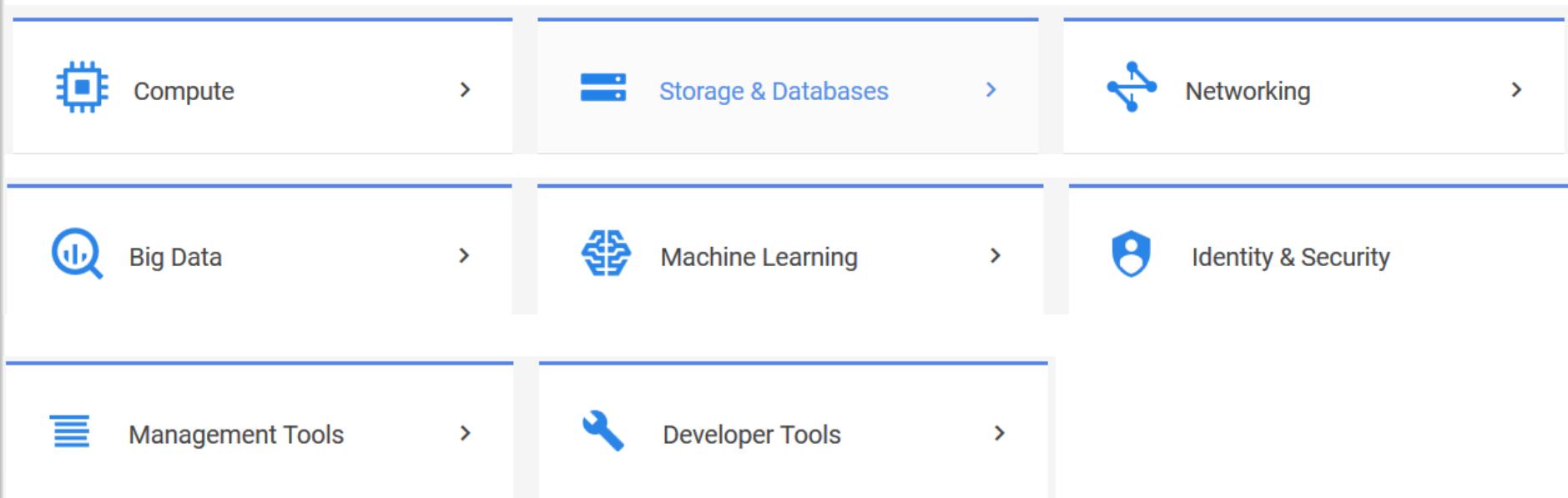


*Prediction API*



2016 - <https://cloud.google.com/>

# Google Cloud Platform



2017 - <https://cloud.google.com/>

13

# Google Cloud Platform

The screenshot shows the Google Cloud Platform dashboard with the following structure:

- Compute**:
  - Compute Engine
  - App Engine
  - Container Engine
  - Container Registry
  - Cloud Functions BETA
- Storage & Databases**:
  - Cloud Storage
  - Cloud SQL
  - Cloud Bigtable
  - Cloud Spanner BETA
  - Cloud Datastore
  - Persistent Disk
- Networking**:
  - Cloud Virtual Network
  - Cloud Load Balancing
  - Cloud CDN
  - Cloud Interconnect
  - Cloud DNS



# Google Cloud Platform

The screenshot shows the Google Cloud Platform homepage with a navigation menu on the left and a main content area on the right.

**Navigation Menu:**

- Compute** (Icon: CPU)
- Big Data** (Icon: Database)
- Manage** (Icon: List)

**Main Content Area:**

- Storage & Databases** (Icon: Database)
- Networking** (Icon: Network)
- Identity & Security** (Icon: Lock)
- Cloud AI** (Icon: Brain)

**Compute Services:**

- Compute Engine (Run VMs on Google's infrastructure)
- App Engine (PaaS for apps and backends)
- Kubernetes Engine (Run containers on GCP)
- Cloud Functions BETA (Serverless environment to build and connect cloud services)

**Big Data Services:**

- Big Data (Icon: Database)
- BigQuery (Fully managed large-scale data warehouse)
- Cloud Dataflow (Real-time batch and stream data processing)
- Cloud Dataproc (Managed Spark and Hadoop service)

**Storage & Databases Services:**

- Storage & Databases (Icon: Database)
- Cloud Storage (Object storage with global edge-caching)
- Cloud SQL (Fully-managed MySQL and PostgreSQL database service)
- Cloud Bigtable (Fully managed NoSQL database service)
- Cloud Spanner (Mission-critical, relational database service)
- Cloud Datastore (NoSQL database for non-relational data)
- Persistent Disk (Block storage for VM instances)

**Networking Services:**

- Networking (Icon: Network)
- Virtual Private Cloud (VPC) (VPC networking for GCP resources)
- Cloud Load Balancing (High performance, scalable load balancing)
- Cloud CDN (Content delivery on Google's global network)
- Cloud Interconnect (Connect directly to GCP's network edge)
- Cloud DNS (Reliable, resilient, low-latency DNS serving)
- Network Service Tiers ALPHA (Optimize your network for performance or cost)

**Data Transfer Services:**

- Data Transfer (Icon: Database)
- Google Transfer Appliance (Securely migrate large volumes of data to Google Cloud Platform)



2018 - <https://cloud.google.com/> 15

# Google Cloud Platform

## GOOGLE CLOUD PLATFORM

- [Featured products](#)
- [AI and machine learning](#)
- [API management](#)
- [Cloud Services Platform](#)
- [Compute](#)
- [Data analytics](#)
- [Databases](#)
- [Developer tools](#)
- [Internet of Things \(IoT\)](#)
- [Management tools](#)
- [Media](#)
- [Migration](#)
- [Networking](#)
- [Security](#)
- [Storage](#)

<a href="#"> Compute</a>	>	<a href="#"> Storage &amp; Databases</a>	>	<a href="#"> Networking</a>	>
<a href="#">Compute Engine</a> Run VMs on Google's infrastructure	>	<a href="#">Cloud Storage</a> Object storage with global edge-caching	>	<a href="#">Virtual Private Cloud (VPC)</a> VPC networking for GCP resources	>
<a href="#">App Engine</a> PaaS for apps and backends	>	<a href="#">Cloud SQL</a> Fully-managed MySQL and PostgreSQL database service	>	<a href="#">Cloud Load Balancing</a> High performance, scalable load balancing	>
<a href="#">Kubernetes Engine</a> Run containers on GCP	>	<a href="#">Cloud Bigtable</a> Fully managed NoSQL database service	>	<a href="#">Cloud CDN</a> Content delivery on Google's global network	>
<a href="#">Cloud Functions <small>BETA</small></a> Serverless environment to build and connect cloud services	>	<a href="#">Cloud Spanner</a> Mission-critical, relational database service	>	<a href="#">Cloud Interconnect</a> Connect directly to GCP's network edge	>
<a href="#"> Big Data</a>	>	<a href="#">Cloud Datastore</a> NoSQL database for non-relational data	>	<a href="#">Cloud DNS</a> Reliable, resilient, low-latency DNS serving	>
<a href="#">BigQuery</a> Fully managed large-scale data warehouse	>	<a href="#">Persistent Disk</a> Block storage for VM instances	>	<a href="#">Network Service Tiers <small>ALPHA</small></a> Optimize your network for performance or cost	>
<a href="#">Cloud Dataflow</a> Real-time batch and stream data processing	>	<a href="#"> Data Transfer</a>	>	<a href="#"> Cloud AI</a>	
<a href="#">Cloud Dataproc</a> Managed Spark and Hadoop service	>	<a href="#">Google Transfer Appliance</a> Securely migrate large volumes of data to Google Cloud Platform	>	 Cloud AutoML <small>Alpha</small>	 Train high quality custom machine learning models with minimum effort and machine learning expertise



2019 - <https://cloud.google.com/products/> 16

# Google Cloud Platform

## Google Cloud Platform

Featured products

AI and Machine Learning

API Management

Compute

Containers

Data Analytics

Databases

Developer Tools

Healthcare and Life Sciences

Hybrid and Multi-cloud

Internet of Things (IoT)

Management Tools

Media and Gaming

Migration

Networking

Operations

Security and Identity

Serverless Computing

Storage

## Google Cloud Platform

Grow your business with our secure storage, powerful compute, and integrated data analytics products.

[Get started for free](#)

[See pricing](#)

### Featured products

#### Compute Engine

Scalable, high-performance VMs.

#### Cloud Run

Run stateless containers on a fully managed environment or on Anthos.

#### Anthos

Modernize existing apps and build new apps rapidly in hybrid and multi-cloud environments.

#### Vision AI

Derive insights from images, text, and more with AutoML Vision and Vision API.

#### Cloud Storage

Object storage with global edge-caching.

#### Cloud SQL

MySQL, PostgreSQL, and SQL Server database services.

#### BigQuery

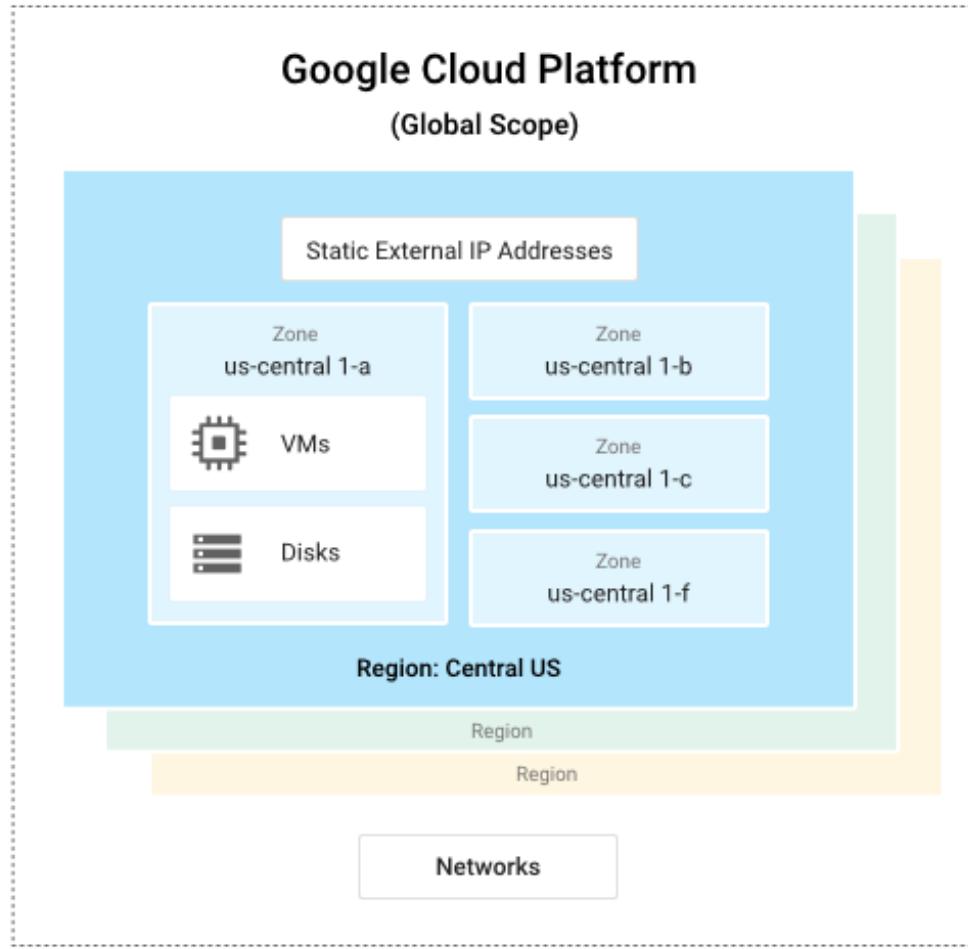
A fully managed, highly scalable data warehouse with built-in ML.

#### Security key enforcement

Enforce the use of security keys to help prevent account takeovers.

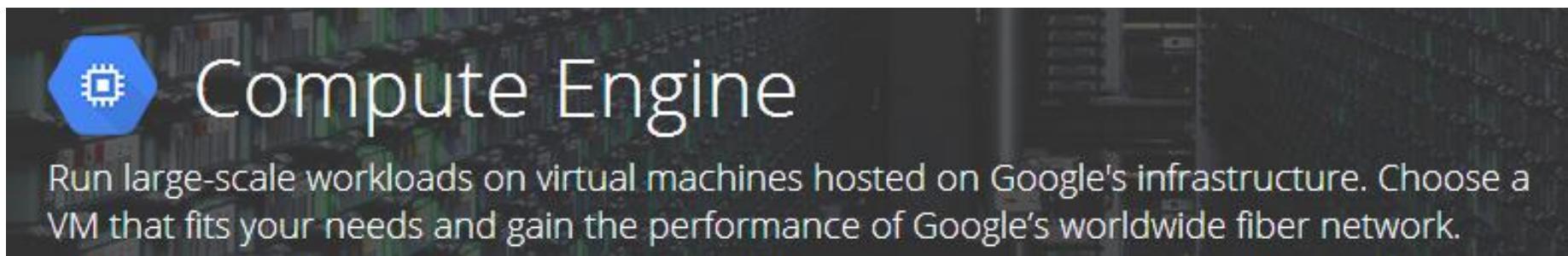


# Google Cloud Platform



2017 - <https://cloud.google.com/docs/overview/>

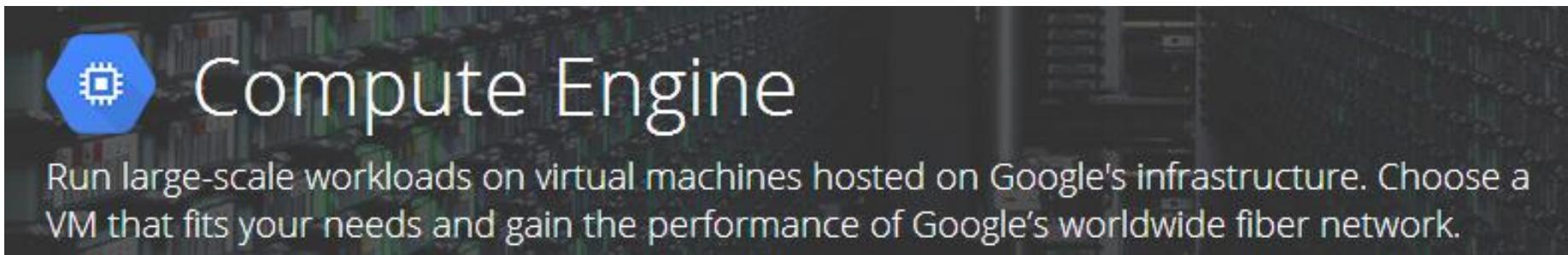
# Google Cloud Platform | Compute



The image shows the Google Cloud Platform Compute Engine landing page. It features a blue hexagonal icon with a white circuit board pattern. To its right, the word "Compute Engine" is written in large, white, sans-serif font. Below this, a descriptive text reads: "Run large-scale workloads on virtual machines hosted on Google's infrastructure. Choose a VM that fits your needs and gain the performance of Google's worldwide fiber network." The background of the page is a dark image of server racks.

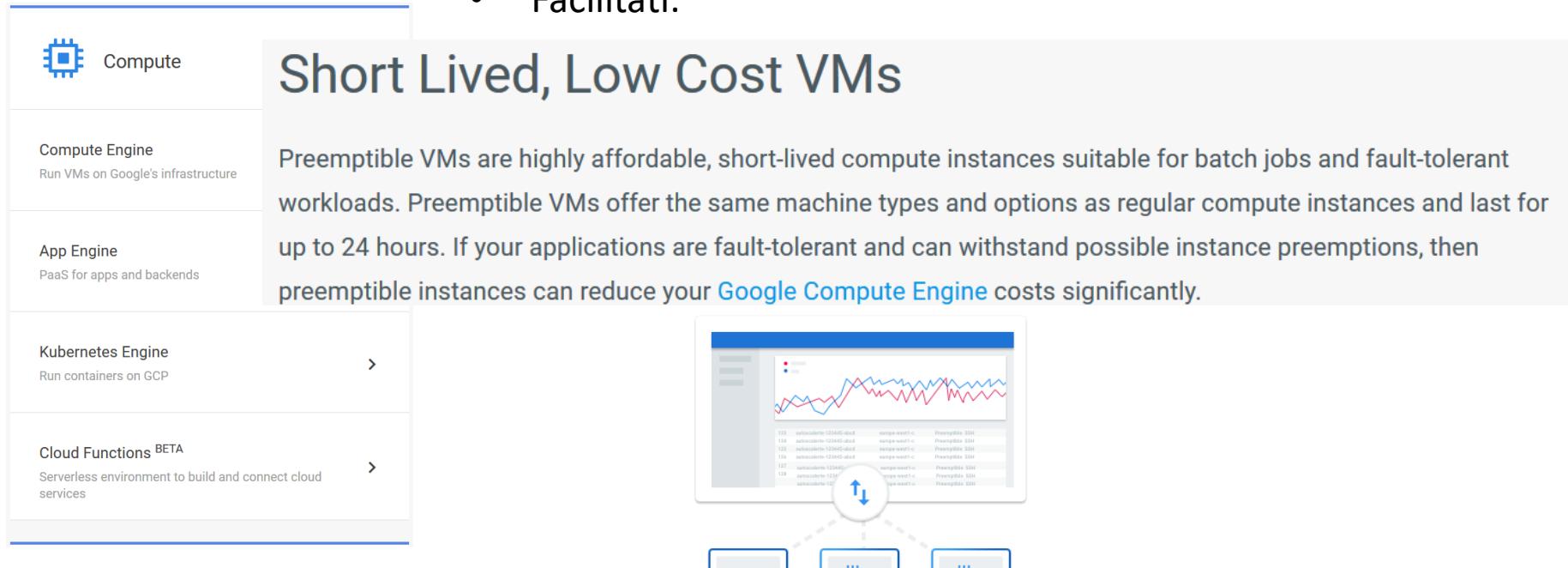
- <https://cloud.google.com/compute/>
- Facilitati:
  - *High-performance virtual machines* (Debian, CentOS, .... <-> instante: microVM -> large)
  - Comunicare: *Google's private global fiber network*
  - *pay-per-use* – Obs. “*Google bills in seconds-level increments ...*”
  - Management automat si facil (API RESTful, command-line interface, etc)
    1. // CREATE INSTANCE WITH 4 vCPUs and 5 GB MEMORY
    2. gcloud compute instances create my-vm --custom-cpu 4 --custom-memory 5
  - Securitate: Certificari pentru Google Compute Engine: ISO 27001, SSAE-16, SOC 1, SOC 2, si SOC 3

# Google Cloud Platform | Compute

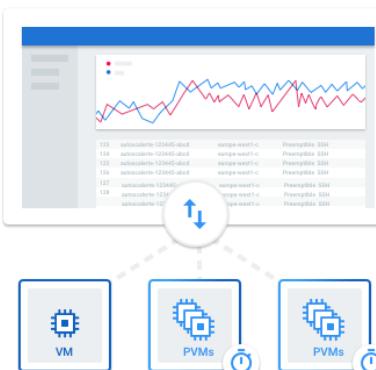


The image shows the Google Cloud Platform Compute Engine landing page. It features a blue hexagonal icon with a white gear-like symbol. To its right, the word "Compute Engine" is written in large, white, sans-serif letters. Below this, a descriptive text reads: "Run large-scale workloads on virtual machines hosted on Google's infrastructure. Choose a VM that fits your needs and gain the performance of Google's worldwide fiber network." The background of the page has a blurred image of server racks.

- Facilitati:

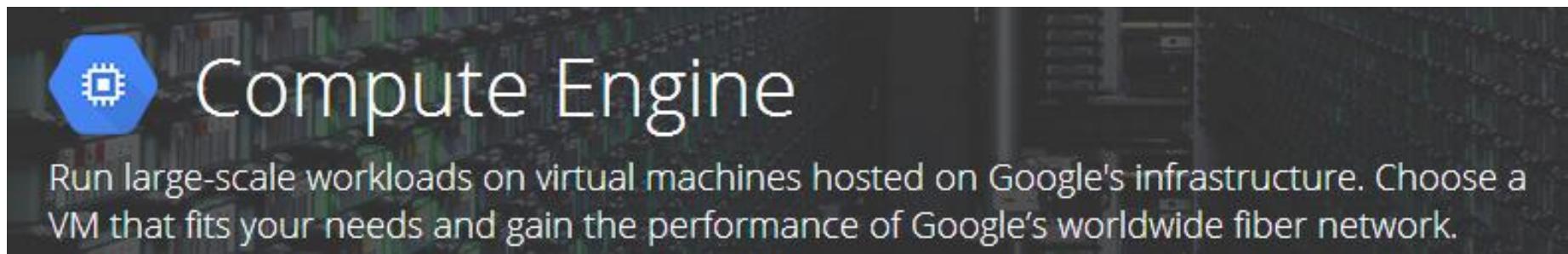


The screenshot shows the Google Cloud Platform Compute Engine interface. On the left, there is a sidebar with several options: "Compute" (selected), "Compute Engine" (Run VMs on Google's infrastructure), "App Engine" (PaaS for apps and backends), "Kubernetes Engine" (Run containers on GCP), and "Cloud Functions BETA" (Serverless environment to build and connect cloud services). The main content area is titled "Short Lived, Low Cost VMs". It contains text explaining that Preemptible VMs are highly affordable, short-lived compute instances suitable for batch jobs and fault-tolerant workloads. They offer the same machine types and options as regular compute instances and last for up to 24 hours. If applications are fault-tolerant and can withstand possible instance preemptions, then preemptible instances can reduce costs significantly. To the right of the text is a diagram illustrating the concept of preemptible VMs. It shows a central circular node with a double-headed arrow pointing to three smaller nodes below it, each labeled "VM" or "PVMs". Above the central node, a table lists 128 preemptible instances (e.g., n1-standard-128-preemptible) and 324 regular instances (e.g., n1-standard-324).



Preemptible VMs are up to 80% cheaper than regular instances.

# Google Cloud Platform | Compute



The image shows the Google Cloud Platform Compute Engine landing page. It features a blue hexagonal icon with a white gear symbol. To its right, the word "Compute Engine" is written in large, white, sans-serif letters. Below this, a dark banner contains the text: "Run large-scale workloads on virtual machines hosted on Google's infrastructure. Choose a VM that fits your needs and gain the performance of Google's worldwide fiber network." The background of the page is a blurred image of server racks.

Click to Deploy

Deploy popular open stacks on Google Compute Engine via our own gallery.

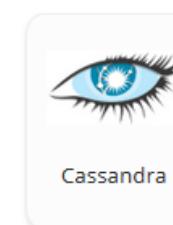
[More Click to Deploy apps](#)



Aerospike



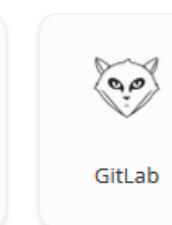
Apache Hadoop



Cassandra



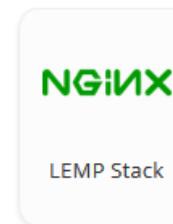
Drupal CMS



GitLab



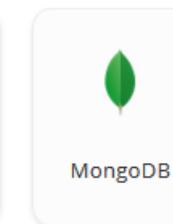
LAMP Stack



LEMP Stack



MEAN Stack



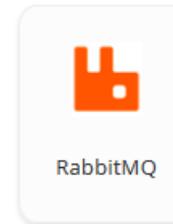
MongoDB



Percona



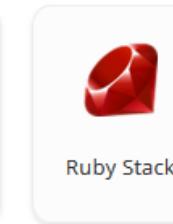
Puppet



RabbitMQ



Redis



Ruby Stack

# Google Cloud Platform | Compute

## Google Cloud Functions

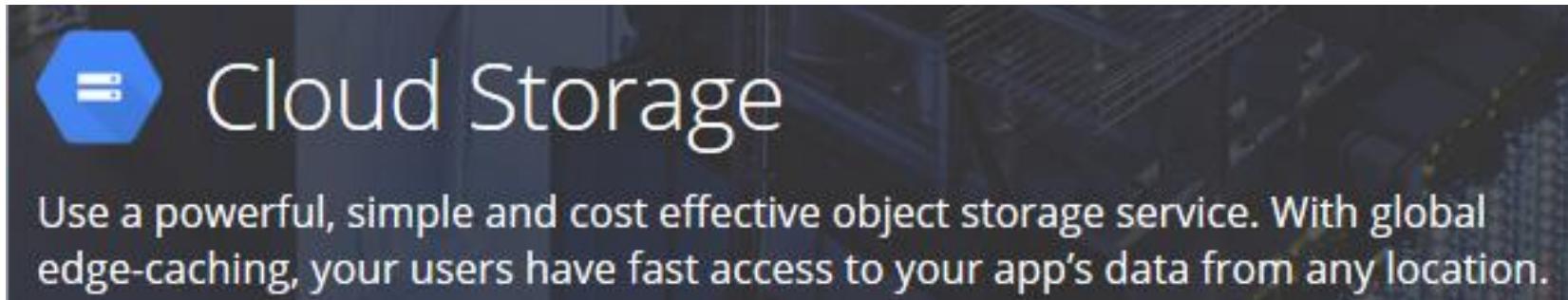
Event-driven serverless compute platform



- ✓ Simplest way to run your code in the cloud
- ✓ Automatically scales, highly available and fault tolerant
- ✓ No servers to provision, manage, patch or update
- ✓ Pay only while your code runs
- ✓ Connects and extends cloud services

Cloud Functions lets application developers spin up code on demand in response to events originating from anywhere. Treat all Google and third-party cloud services as building blocks, connect and extend them with code, and build applications that scale from zero to planet-scale—without provisioning or managing a single server.

# Google Cloud Platform | Storage

A screenshot of the Google Cloud Storage landing page. It features a blue hexagonal icon with three horizontal bars inside, followed by the text "Cloud Storage". Below this, a large, bold, italicized sentence reads: "Use a powerful, simple and cost effective object storage service. With global edge-caching, your users have fast access to your app's data from any location." In the background, there is a blurred image of server racks in a data center.

Use a powerful, simple and cost effective object storage service. With global edge-caching, your users have fast access to your app's data from any location.

- <https://cloud.google.com/storage/>
- Facilitati:
  - Standard Storage – util pentru stocarea de date care necesita un nivel ridicat de disponibilitate si performanta (accesare frecventa, aplicatii mobile, streaming video)
  - Archive Storage – util pentru stocarea de date care necesita latenta mica
  - Nearline Storage – util pentru stocare de date accesate mai putin de odata pe luna
  - Coldline Storage – pentru stocare de date *long-lived* dar care sunt accesate mai putin frecvent (cel mult odata pe an, *disaster-recovery*)

# Google Cloud Platform | Storage



The image shows the Google Cloud Storage landing page. It features a blue hexagonal icon with three horizontal bars inside, followed by the text "Cloud Storage". Below this, a large call-to-action text reads: "Use a powerful, simple and cost effective object storage service. With global edge-caching, your users have fast access to your app's data from any location." The background of the header is a dark image of server racks.

- ✓ Unlimited storage with no minimum object size
- ✓ Worldwide accessibility and worldwide [storage locations](#)
- ✓ Low latency
- ✓ High durability (99.99999999% annual durability)
- ✓ [Geo-redundancy](#) if the data is stored in a multi-region or dual-region



BLOG POST

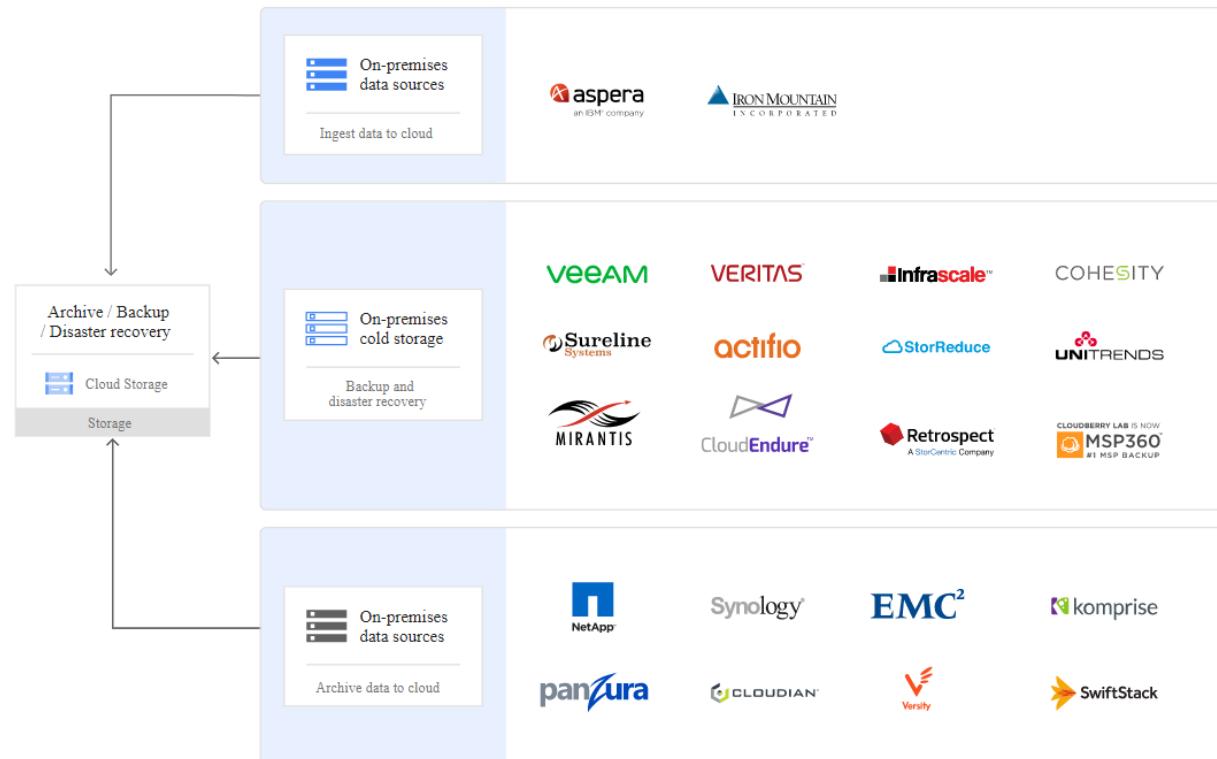
Twitter chooses Google Cloud for its flexibility in storage and compute.

# Google Cloud Platform | Storage

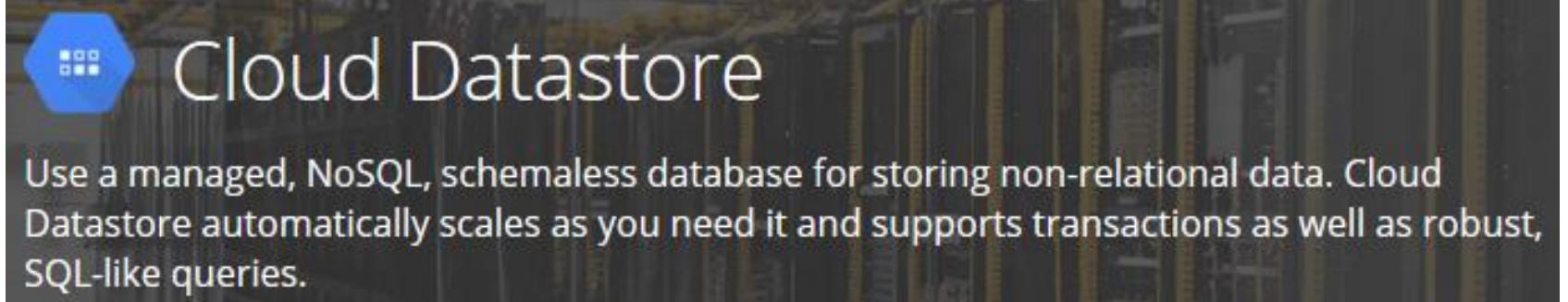


The image shows the Google Cloud Storage landing page. It features a large blue hexagonal icon with a white equals sign symbol. Below it, the word "Cloud Storage" is written in a large, white, sans-serif font. A sub-headline reads: "Use a powerful, simple and cost effective object storage service. With global edge-caching, your users have fast access to your app's data from any location."

## Backups and archives



# Google Cloud Platform | Storage

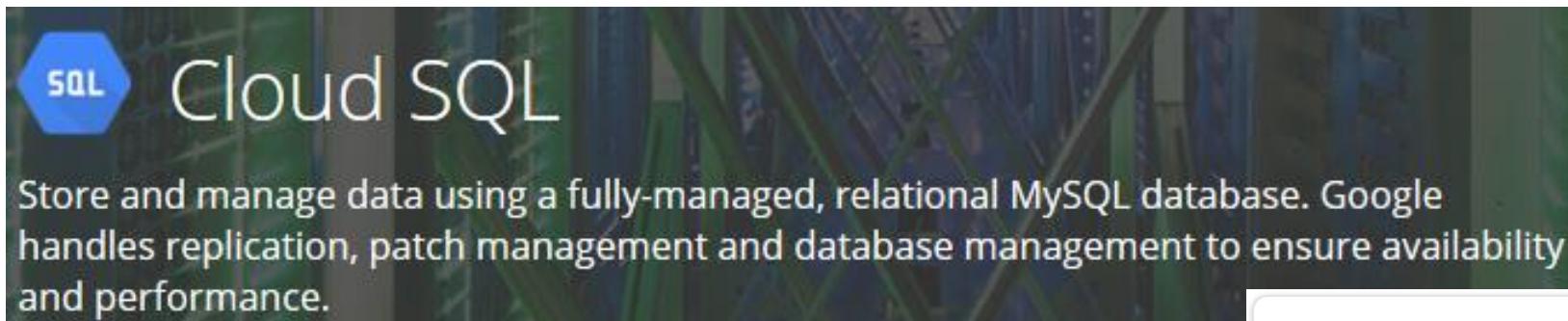


The image shows the Google Cloud Datastore landing page. It features a blue hexagonal icon with a grid of squares. The title "Cloud Datastore" is displayed in large white text. Below the title, a descriptive text reads: "Use a managed, NoSQL, schemaless database for storing non-relational data. Cloud Datastore automatically scales as you need it and supports transactions as well as robust, SQL-like queries." The background of the page is a dark image of server racks.

- <https://cloud.google.com/datastore/>
- Facilitati:
  - Serviciu de stocare *schemaless* pentru *document-oriented database*
  - Permite replicare si sharding automat => disponibilitate si consistenta
  - Cloud Datastore Development Kit – permite dezvoltarea locala de aplicatii web sau mobile

	FREE limit per day	PRICE above free limit (per unit)	Price Unit
Stored data	1 GB storage	\$0.18	GB/Month
Entity Reads	50,000	\$0.06	per 100,000 entities
Entity Writes	20,000	\$0.18	per 100,000 entities
Entity Deletes	20,000	\$0.02	per 100,000 entities
Small Operations	50,000	Free	-

# Google Cloud Platform | Storage



**Cloud SQL**

Store and manage data using a fully-managed, relational MySQL database. Google handles replication, patch management and database management to ensure availability and performance.

- <https://cloud.google.com/sql>
- Facilitati:
  - Baze de date MySQL si PostgreSQL
  - Se asigura securitatea (ISO/IEC 27001), performanta, scalabilitate si usurinta utilizarii de catre aplicatii care pot rula oriunde
  - Acces: consola Web sau interfata in linia de comanda
  - *No Lock-in*

Gartner

Gartner recognizes Google Cloud as a Leader in the 2020 Magic Quadrant for Cloud Database Management Systems

# Google Cloud Platform | Storage

- <https://cloud.google.com/bigtable/>
- Cloud Bigtable este serviciul Google's de NoSQL Big Data databases
- Este folosit de: Search, Analytics, Maps, and Gmail
- Este proiectat sa realizeze managementul in contexte de incarcare masiva, asigurand latenta mica si viteza mare de procesare (=> optiune buna pentru aplicatii de tipul *IoT, user analytics sau financial data analysis*)



# Google Cloud Platform | Storage

## PERSISTENT DISK

Reliable, high-performance block storage for virtual machine instances

- <https://cloud.google.com/persistent-disk/>

Google Persistent Disk is durable and high performance block storage for the Google Cloud Platform. Persistent Disk provides SSD and HDD storage which can be attached to instances running in either Google Compute Engine or Google Kubernetes Engine. Storage volumes can be transparently resized, quickly backed up, and offer the ability to support simultaneous readers.

### Scale Without Interruption

You no longer have to worry about undersizing your block devices. Persistent Disk gives you unlimited flexibility by allowing you to resize your storage while it's in use by one or more virtual machines with no downtime.



# Google Cloud Platform | Big Data



## BigQuery

Analyze Big Data in the cloud with BigQuery. Run fast, SQL-like queries against multi-terabyte datasets in seconds. Scalable and easy to use, BigQuery gives you real-time insights about your data.

- <https://cloud.google.com/bigquery/>
- 1TB/luna de date este free + 10Gb storage
- Interogarile se pot executa in mod asincron in background; se permite accesul la istoria interogarilor si job-urilor

Resource	Pricing
Loading Data	Free
Exporting Data	Free
Storage	\$0.020 per GB / month <sup>1,4</sup>
Interactive Queries	\$5 per TB processed <sup>2,3,4</sup>
Batch Queries	\$5 per TB processed <sup>2,3,4</sup>
Streaming Inserts	\$0.01 per 100,000 rows.

# Google Cloud Platform | Big Data



BigQuery

Analyze Big Data in the cloud with BigQuery. Run fast, SQL-like queries against multi-terabyte datasets.

language correlation

Top 10 repos by total number of forks

Token occurrence per language

hello

Send request

```
1 SELECT repository_language, COUNT(*) AS cntlang
2 FROM [githubarchive:github.timeline]
3 WHERE repository_language != ''
4 AND payload_commit_msg != ''
5 AND PARSE_UTC_USEC(created_at) > PARSE_UTC_USEC('2014-03-09T00:00:00Z')
6 AND REGEXP_MATCH(payload_commit_msg,
7   r'(?i)\b(hello)\b')
8 GROUP BY repository_language
9 ORDER BY cntlang DESC
```

Query complete: 2.5s elapsed

JavaScript	14989
Java	12000
CSS	9076
Ruby	8661

Gartner names Google a leader in the 2019 Magic Quadrant for Data Management Solutions for Analytics (DMSA). Get your complimentary report here.

real-time



CASE STUDY

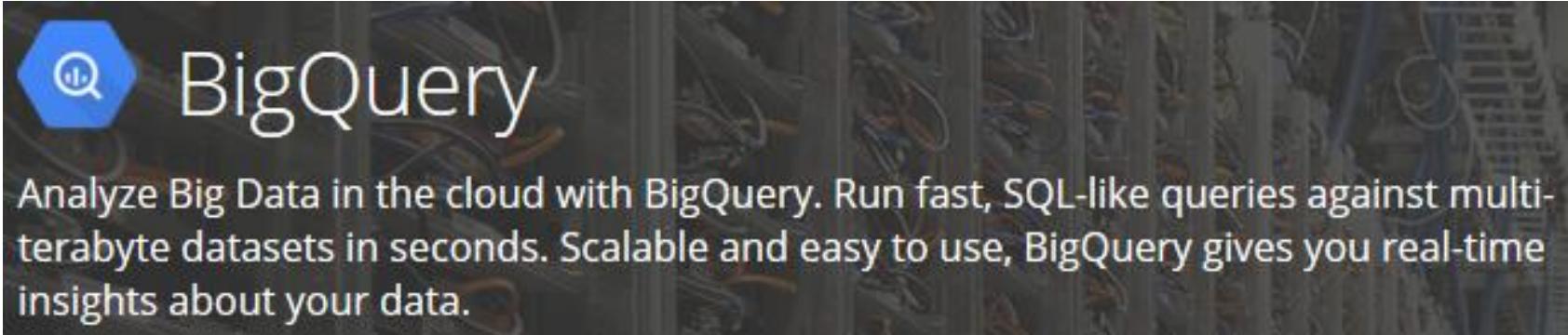
Twitter modernizes ad engagement platform; combines millions of metrics/second.

<https://cloud.google.com/bigdata-analytics/modernizing-twiters-ad-engagement-analytics-platform>

31

2021 | <http://www.info.uaic.ro/~adria>

# Google Cloud Platform | Big Data

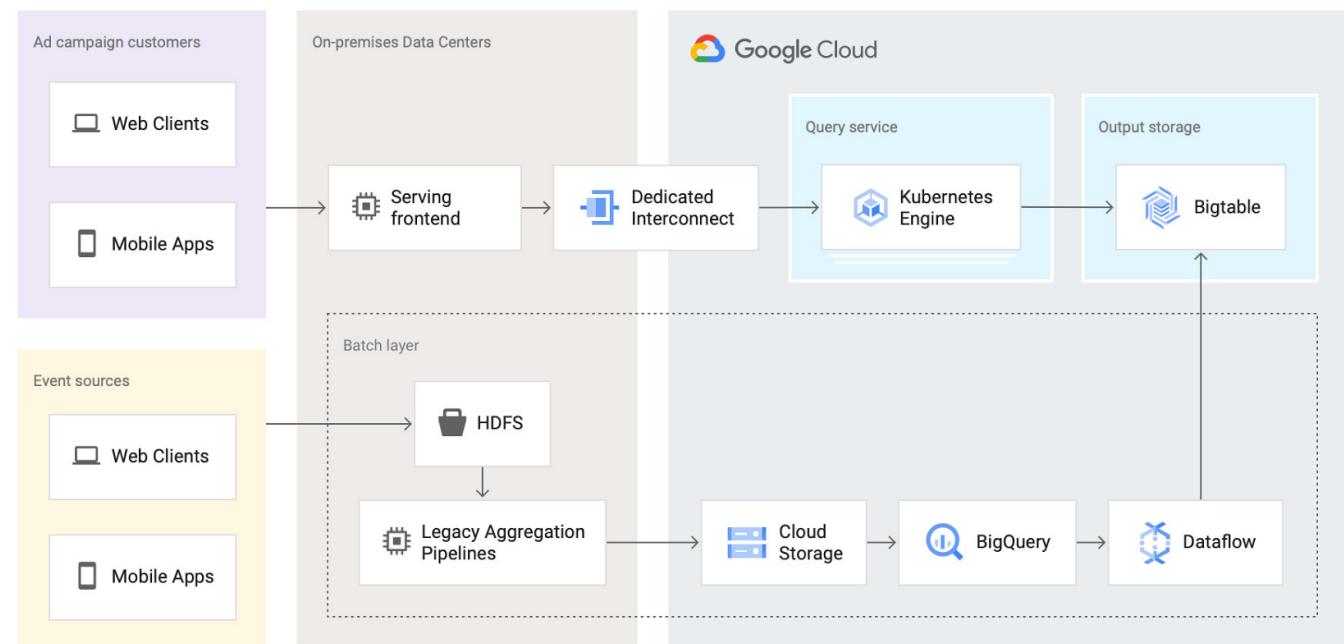


The image shows the Google Cloud BigQuery landing page. It features a large hexagonal icon with a magnifying glass and a blue 'Q' inside. The word 'BigQuery' is written in a large, white, sans-serif font. Below the title, there is a descriptive text: 'Analyze Big Data in the cloud with BigQuery. Run fast, SQL-like queries against multi-terabyte datasets in seconds. Scalable and easy to use, BigQuery gives you real-time insights about your data.'



## CASE STUDY

Twitter modernizes ad engagement platform; combines millions of metrics/second.



[https://cloud.google.com/blog/products/big-data/modernizing-twit...  
ters-ad-engagement-analytics-platform](https://cloud.google.com/blog/products/big-data/modernizing-twit...)

(prima iteratie)

# Google Cloud Platform | Networking

- <https://cloud.google.com/cdn/>

Google Cloud CDN leverages Google's globally distributed edge points of presence to accelerate content delivery for websites and applications served out of Google Compute Engine and Google Cloud Storage. Cloud CDN lowers network latency, offloads origins, and reduces serving costs. Once you've set up HTTP(S) Load Balancing, simply enable Cloud CDN with a single checkbox.



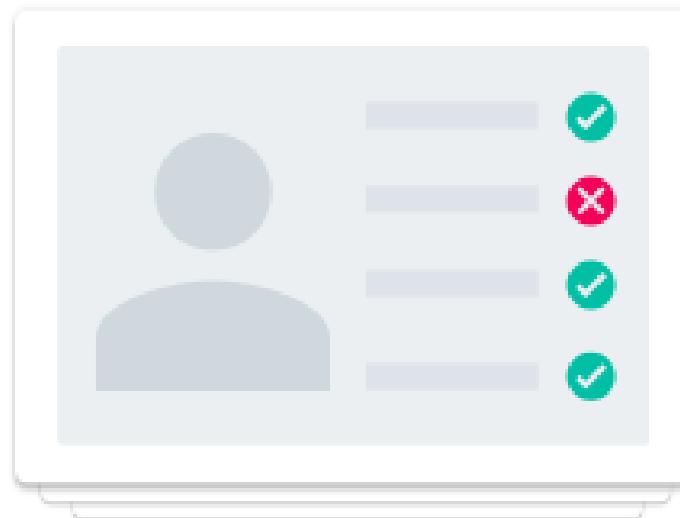
ITEM	PRICE (USD)
Cache egress	Charges vary based on location and usage: \$0.02/GB - \$0.08/GB for North America and Europe destinations \$0.04/GB - \$0.20/GB for other destinations worldwide
Cache fill	Charges vary based on location: \$0.04/GB - \$0.06/GB discounted pricing for in-region cache fills \$0.08/GB - \$0.15/GB for cross-region cache fills
HTTP(S) requests	\$0.0075 per 10,000 requests
Cache invalidation	\$0.005 per invalidation

With caches at more than 90 sites around the world,

# Google Cloud Platform | Identity & Security

- <https://cloud.google.com/iam/>

Google Cloud Identity & Access Management (IAM) lets administrators authorize who can take action on specific resources, giving you full control and visibility to manage cloud resources centrally. For established enterprises with complex organizational structures, hundreds of workgroups and potentially many more projects, Cloud IAM provides a unified view into security policy across your entire organization, with built-in auditing to ease compliance processes.

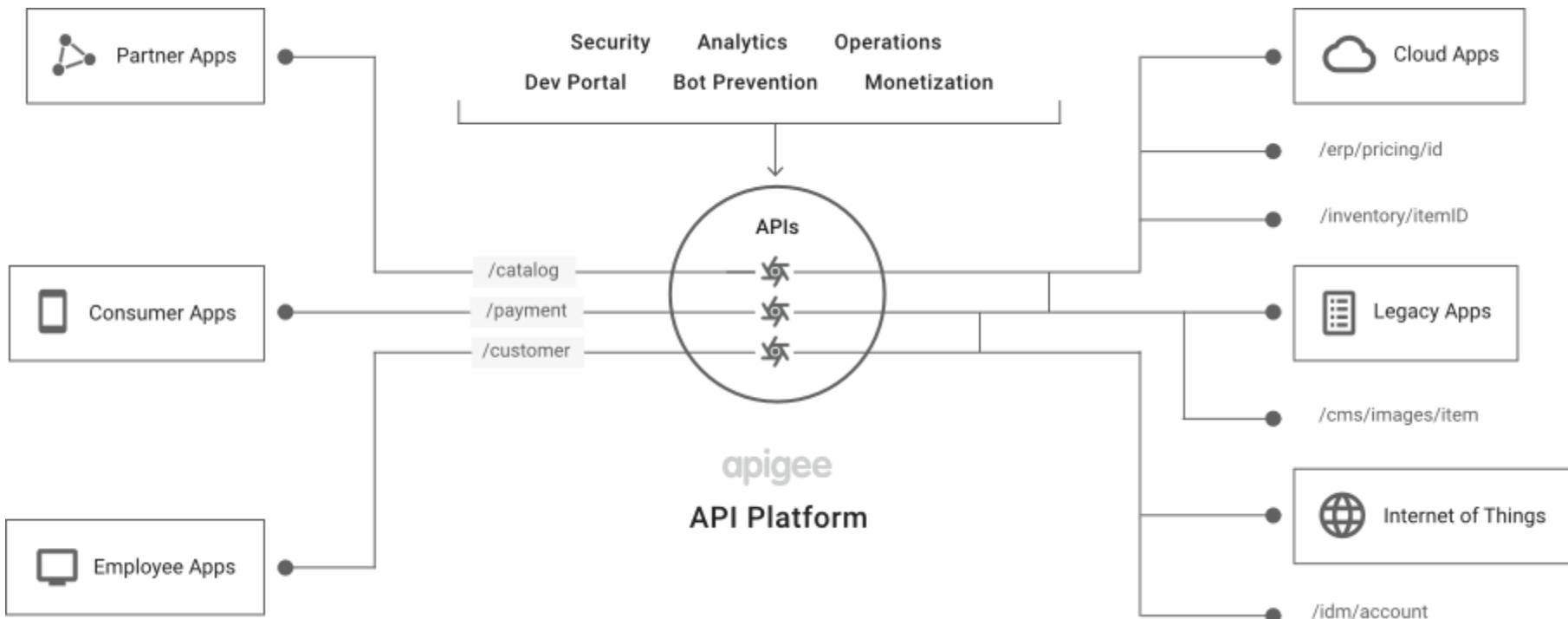


<https://cloud.google.com/iam/docs/quickstart>

# Google Cloud Platform | API Platform and Ecosystems

- <https://cloud.google.com/apigee-api-management/>

Apigee is a full lifecycle API management platform that enables API providers to design, secure, deploy, monitor, and scale APIs. Apigee sits in-line with runtime API traffic and enforces a set of out-of-the-box API policies, including key validation, quota management, transformation, authorization, and access control. API providers use the customizable developer portal to enable developers to consume APIs easily and securely, and to measure API performance and usage.



# Google Cloud Platform | Cloud AI

## AI and Machine Learning →

### AI Building Blocks

#### [AI building blocks](#)

Easily infuse AI into applications with custom or pre-trained models.

#### [AutoML](#)

Easily train high-quality, custom ML models.

#### [Vision AI](#)

Derive insights from images, text, and more with AutoML Vision and Vision API.

#### [Video AI](#)

Enable powerful content discovery and engaging video experiences.

#### [Cloud Natural Language](#)

Derive insights from unstructured text.

#### [Cloud Translation](#)

Dynamically translate between languages.

#### [Text-to-Speech](#)

Text-to-speech conversion powered by ML.

#### [Speech-to-Text](#)

Speech-to-text conversion powered by ML.

#### [Dialogflow](#)

Create conversational experiences across devices and platforms.

#### [AutoML Tables \(beta\)](#)

Build and deploy machine learning models on structured data.

#### [Cloud Inference API \(alpha\)](#)

Quickly run large-scale correlations over typed time-series datasets.

#### [Cloud Data Labeling](#)

Managed annotation for high quality model training data.

## CLOUD NATURAL LANGUAGE

Derive insights from unstructured text using Google machine learning

### Powerful Text Analysis

Google Cloud Natural Language reveals the structure and meaning of text by offering powerful machine learning models in an easy to use REST API. You can use it to **extract information** about people, places, events and much more, mentioned in text documents, news articles or blog posts. You can use it to **understand sentiment** about your product on social media or **parse intent** from customer conversations happening in a call center or a messaging app. You can **analyze text uploaded in your request** or integrate with your document storage on Google Cloud Storage.

# Google Cloud Platform | Cloud AI

## AI Building Blocks

### AI building blocks

Easily infuse AI into applications with custom or pre-trained models.

### AutoML

Custom machine learning model training and development.

### Vision AI

Custom and pre-trained models to detect emotion, text, more.

### Video AI

Video classification and recognition using machine learning.

### Cloud Natural Language

Sentiment analysis and classification of unstructured text.

### Cloud Translation

Language detection, translation, and glossary support.

### Media Translation (beta)

Add dynamic audio translation directly to your content and applications.

### Text-to-Speech

Speech synthesis in 220+ voices and 40+ languages.

### Speech-to-Text

Speech recognition and transcription supporting 125 languages.

### Dialogflow

Conversation applications and systems development suite for virtual agents.

### AutoML Tables (beta)

Service for training ML models with structured data.

### Cloud Inference API (alpha)

Quickly run large-scale correlations over typed time-series datasets.

### Recommendations AI

Deliver highly personalized product recommendations at scale.

## AI Infrastructure

Options for every business to train deep learning and ML models cost-effectively.

## Cloud GPUs

GPUs for machine learning, scientific computing, and 3D visualization.

## Cloud TPU

Tensor processing units for machine learning applications.

## AI Platform and Accelerators

### AI Platform

Platform for training, hosting, and managing ML models.

### AI Platform Deep Learning VM Image

Preconfigured VMs for deep learning applications.

### AI Platform Notebooks

An enterprise notebook service to get projects up and running in minutes.

### Deep Learning Containers (beta)

Preconfigured and optimized containers for deep learning environments.

### Cloud Data Labeling

Managed annotation for high-quality model training data.

### TensorFlow Enterprise

Reliability and performance for AI apps with enterprise-grade support and managed services.

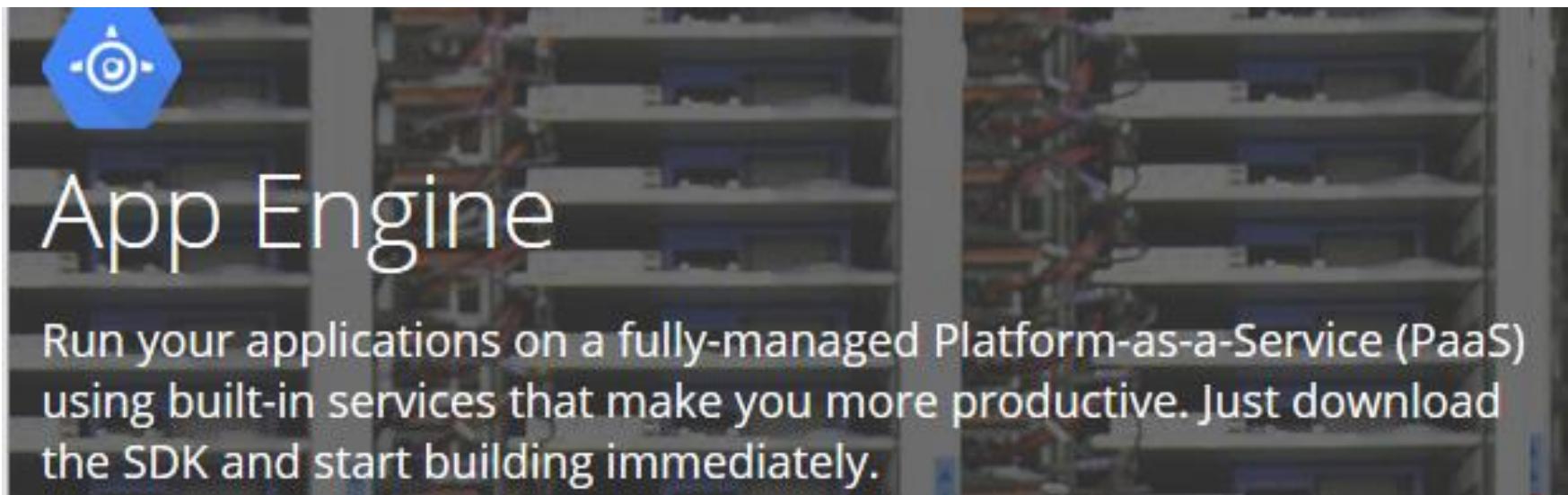
### Document AI

Document data capture at scale with AI and machine learning.



# Google App Engine

2008 – lansarea GAE - <http://www.youtube.com/watch?v=3Ztr-HhWX1c&feature=youtu.be>



The screenshot shows the Google App Engine landing page. It features a large blue hexagonal icon with a white target symbol. Below it, the text "App Engine" is displayed in a large, sans-serif font. A descriptive paragraph follows: "Run your applications on a fully-managed Platform-as-a-Service (PaaS) using built-in services that make you more productive. Just download the SDK and start building immediately." The background of the slide is a blurred image of server racks.

[<https://developers.google.com/appengine/training/intro/whatisgae>]

# Google App Engine

## Google App Engine

- 2018 | Mediul de dezvoltare App Engine ofera

### Standard Environment

[About the Standard Environment](#)

[Python 2.7](#)

[Java 7](#)

[PHP 5.5](#)

[Go 1.6](#)

### Flexible Environment

[About the Flexible Environment](#)

[Python 2.7, 3.6](#)

[Java 8](#)

[Node.js](#)

[Go 1.8, 1.9](#)

[Ruby](#)

[PHP 5.6, 7](#)

[.NET](#)

[Custom Runtimes](#)

[Known Issues](#)

[ 2018 - <https://cloud.google.com/appengine/docs/> ]

# Google App Engine

## Google App Engine

- 2019 | Mediul de dezvoltare App Engine ofera

### Standard Environment

[About the Standard Environment](#)

[The Standard Environment](#)  
[Runtimes](#)

[Python 2.7, 3.7](#)

[Java 7, 8](#)

[Node.js 8, 10](#)

[PHP 5.5, 7.2](#)

[Go 1.9, 1.11](#)

### Flexible Environment

[About the Flexible Environment](#)

[Python 2.7, 3.6](#)

[Java 8](#)

[Node.js](#)

[Go 1.9, 1.10, 1.11](#)

[Ruby](#)

[PHP 5.6, 7.0, 7.1, 7.2](#)

[.NET](#)

[Custom Runtimes](#)

[Known Issues](#)

[2019 - <https://cloud.google.com/appengine/docs/>]

# Google App Engine

## Google App Engine

- 2020 | Mediul de dezvoltare App Engine ofera

- Standard Environment

- [About the Standard Environment](#)
- [The Standard Environment Runtimes](#)
- [Long-term Support for Standard Runtimes](#)
- [Python 2.7, 3.7, 3.8](#)
- [Java 8, 11](#)
- [Node.js 10, 12](#)
- [PHP 5.5, 7.2, 7.3](#)
- [Ruby 2.5](#)
- [Go 1.11, 1.12, 1.13](#)

- Flexible Environment

- [About the Flexible Environment](#)
- [Python 2.7, 3.6](#)
- [Java 8](#)
- [Node.js](#)
- [Go 1.9, 1.10, 1.11](#)
- [Ruby](#)
- [PHP 5.6, 7.0, 7.1, 7.2](#)
- [.NET](#)
- [Custom Runtimes](#)

[2020 - <https://cloud.google.com/appengine/docs/>]

# Google App Engine

## Google App Engine

- Mediul de dezvoltare App Engine ofera:
  - **App Engine Standard Environment - Sandbox Environment** – cod + server Web+ *language runtime (modificat)* a.i. se respecta restrictiile *sandbox* => aplicatiile ruleaza intr-un mediu securizat, izolat de nivelul hardware, sistemul de operare si localizarea fizica a serverului
    - Aceasta limitare permite distribuirea cererilor web la mai multe servere web (pornirea/oprirea lor) in functie de cresterea/scaderea cererilor
  - **App Engine Flexible Environment** – permite rularea de aplicatii App Engine folosind **Google Compute Engine**
    - “*VM hosting environment offers more flexibility and provides more CPU and memory options*”.
    - Nu mai exista restrictiile din *sandbox runtimes*  
[<https://cloud.google.com/appengine/docs/>]

Feature	Standard environment	Flexible environment
Instance startup time	Seconds	Minutes
Maximum request timeout	Depends on the <a href="#">runtime and type of scaling</a> .	60 minutes
Background threads	Yes, with restrictions	Yes
Background processes	No	Yes
SSH debugging	No	Yes
Scaling	Manual, Basic, Automatic	Manual, Automatic
Scale to zero	Yes	No, minimum 1 instance
Writing to local disk	<ul style="list-style-type: none"> <li>• Java 8, Java 11, Node.js, Python 3.7, PHP 7.2, PHP 7.3, Ruby 2.5 (beta), Go 1.11, Go 1.12, and Go 1.13 have read and write access to the <code>/tmp</code> directory.</li> <li>• Python 2.7 and PHP 5.5 don't have write access to the disk.</li> </ul>	Yes, ephemeral (disk initialized on each VM startup)
Modifying the runtime	No	Yes (through Dockerfile)
Deployment time	Seconds	Minutes
Automatic in-place security patches	Yes	Yes (excludes container image runtime)
Access to App Engine APIs & Services such as <a href="#">NDB</a> , <a href="#">Users API</a> , <a href="#">Memcache</a> , <a href="#">Images API</a> and others.	<ul style="list-style-type: none"> <li>• Yes for Python 2.7, PHP 5.5, and Java 8</li> <li>• No for Java 11, Node.js, Python 3.7, PHP 7.2, PHP 7.3, Ruby 2.5 (beta), Go 1.11, Go 1.12, and Go 1.13 (beta)</li> </ul>	No
WebSockets	No Java 8, Python 2, and PHP 5 provide a proprietary Sockets API, but the API is not available in newer standard runtimes.	Yes
Supports installing third-party binaries	<ul style="list-style-type: none"> <li>• Yes for Java 8, Java 11 Node.js, Python 3.7, PHP 7.2, PHP 7.3, Ruby 2.5 (beta), Go 1.11, Go 1.12, and Go 1.13 (beta).</li> <li>• No for Python 2.7 and PHP 5.5.</li> </ul>	Yes
Location	North America, Asia Pacific, or Europe	North America, Asia Pacific, or Europe
Pricing	Based on <a href="#">instance hours</a>	Based on usage of <a href="#">vCPU</a> , <a href="#">memory</a> , and <a href="#">persistent disks</a>

[2019 -  
<https://cloud.google.com/appengine/docs/the-appengine-environments>]

# Google App Engine

## ***App Engine Flexible Environment***

- Folosind Serviciile Google Compute Engine, App Engine asigura automat:
  - scalarea si *load balancing*
  - verificarea starii instantelor si co-locarea optima alaturi de alte servicii din proiect
  - se ofera acces de root la instancele Compute Engine VM (accesul ssh este dezactivat in mod implicit)
  - suport pentru update-uri
  - se permite modificarea *environment-ului* si a sistemului de operare prin utilizarea *Dockerfiles*
- Sunt suportate nativ: *microservices, authorization, SQL and NoSQL databases, traffic splitting, logging, versioning, security scanning, si content delivery networks*

# Google App Engine

## ***App Engine Flexible Environment***

- ***Runtimes***

### ***Google-supplied Dockerfile => standard runtime***

- ofera suport nativ pentru: Java 8 / Servlet 3.1 / Jetty 9, Python 2.7 and Python 3.6, Node.js, Ruby, PHP, .NET core, and Go
- Nu exista restrictii sandbox
- Se pot face interogari privind starea VM, se ofera suport pentru servicii ca: Datastore, Memcache, Task Queues, Logging, Users

### ***Docker image sau Dockerfile modificat (open source) => custom runtime***

- Se foloseste daca se doreste scrierea de cod in alte limbaje
- Se pot configura componente ca: interpretoare sau servere de aplicatii

- ***Performanta***

- O gama larga de configuratii ale CPU si memoriei care odata configurate vor fi furnizate de AppEngine in mod automat

# Google App Engine

- **App Engine Flexible Environment**

<https://cloud.google.com/appengine/docs/flexible/>

GO

JAVA 8

PHP 5 / 7

PYTHON 2.7 / 3.6

.NET

NODE.JS

RUBY

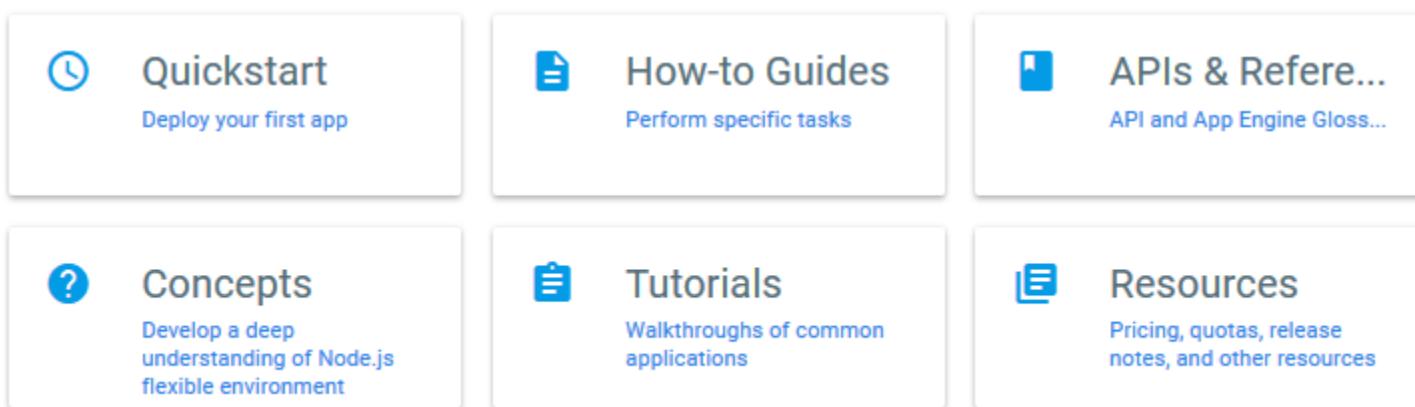
CUSTOM RUNTIMES

# Google App Engine

- **App Engine Flexible Environment - NodeJS**

**<https://cloud.google.com/appengine/docs/flexible/nodejs/>**

The App Engine flexible environment is based on Google Compute Engine and automatically scales your app up and down while balancing the load.



# Standard vs Flexible Environment

## When to choose the standard environment

Application instances run in a [sandbox](#), using the runtime environment of a supported language listed below.

Applications that need to deal with rapid scaling.

The standard environment is optimal for applications with the following characteristics:

- Source code is written in **specific versions of the supported programming languages**:
  - Python 2.7, Python 3.7
  - Java 8, Java 11
  - Node.js 8, Node.js 10
  - PHP 5.5, PHP 7.2, and PHP 7.3
  - Ruby 2.5 (beta)
  - Go 1.11, Go 1.12 and Go 1.13 (beta)
- Intended to **run for free or at very low cost**, where you pay only for what you need and when you need it. For example, your application can scale to 0 instances when there is no traffic.
- Experiences **sudden and extreme spikes of traffic** which require immediate scaling.

## When to choose the flexible environment

Application instances run within Docker containers on Compute Engine virtual machines (VM).

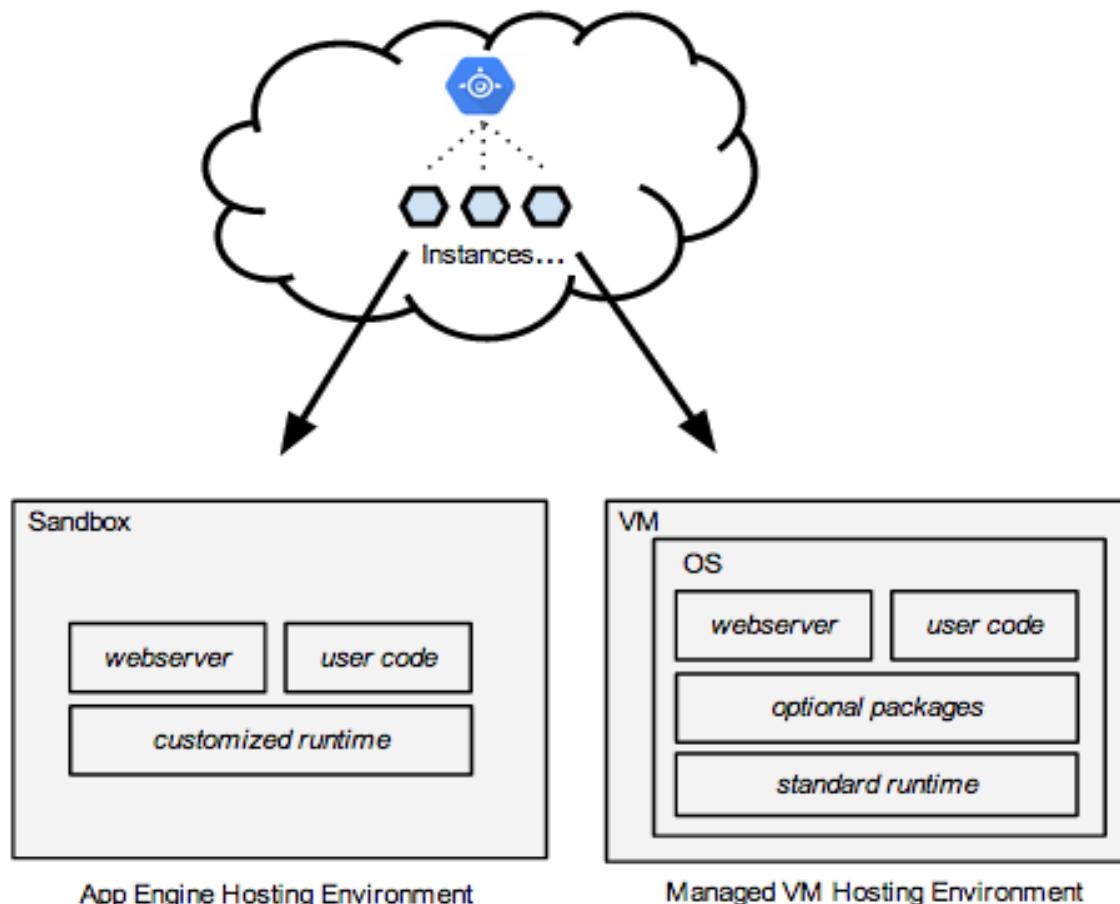
Applications that receive consistent traffic, experience regular traffic fluctuations, or meet the parameters for scaling up and down gradually.

The flexible environment is optimal for applications with the following characteristics:

- Source code that is written in a version of any of the supported programming languages:  
**Python, Java, Node.js, Go, Ruby, PHP, or .NET**
- Runs in a Docker container that includes a custom runtime or source code written in **other programming languages**.
- Uses or depends on frameworks that include **native code**.
- Accesses the resources or services of your Google Cloud project that reside in the **Compute Engine network**.

# Google App Engine

- O aplicatie poate contine module care ruleaza in medii de gazduire diferite e.g se poate folosi pentru frontend *sandbox* si pentru procesare sa se foloseasca ***App Engine Flexible Environment***



# Google App Engine | Standard Environment



*"Some users confuse Google App Engine with Amazon's EC2 service. The problem is that this is an apples to oranges comparison. Both operate at different cloud service levels, and each have their strengths and minuses. With App Engine, you only need to worry about your application and let Google take care of hosting and running it for you. With EC2, you're responsible for the app, but also its database server, web server, operating system, monitoring, load-balancing, upgrades, etc. This is the reason why typically, the costs for IaaS services run lower than that of PaaS services because with PaaS, you're "outsourcing" more work/responsibility. Cost estimates usually clouded by not considering the administration overhead when managing the infrastructure yourself. A better "apples-to-apples" comparison would be EC2 to the [Google Compute Engine](#) IaaS service."*

[<https://developers.google.com/appengine/training/intro/whatisgae>]

# Google App Engine | Standard Environment



## App Engine Standard Environment

### Caracteristici

- Permite rularea (*hosting*) de aplicatii Web folosind infrastructura Google
  - “*App Engine Does One Thing Well*” [Rossum, Google]
- A nu se intelege “*rent a piece of a server*”, deoarece aplicatia nu este gazduita pe un singur server
- Conceptual Google App Engine este la nivel PaaS
- Controlul?
  - Il are Google...
  - Fara griji legate de infrastructura, *load balancing*, managementul stocarii
- Folosind Google Apps aplicatia poate folosi propriul domeniu (<http://www.ex.com>) sau poate folosi un nume din domeniul *appspot.com*



# Google App Engine | Standard Environment

## Caracteristici

- GAE suporta aplicatii scrise in diferite limbaje (vezi slide-urile urmatoare)
- Dezvoltatorii au acces la tehnologii de stocare de tipul GFS (Google File System), Bigtable (sistem de stocare pentru date nestructurate),...
- *“With Google App Engine, developers can write Web applications based on the same building blocks that Google uses,” Kevin Gibbs, Google’s technical lead for the project, wrote in *The Official Google Blog*. “Google App Engine packages those building blocks and provides access to scalable infrastructure that we hope will make it easier for developers to scale their applications automatically as they grow.”*

# Google App Engine | Standard Environment





# Google App Engine | Standard Environment

**Cloud Applications**

**Cloud Software Environment**

**Cloud Software Infrastructure**

**Computational  
Resources**

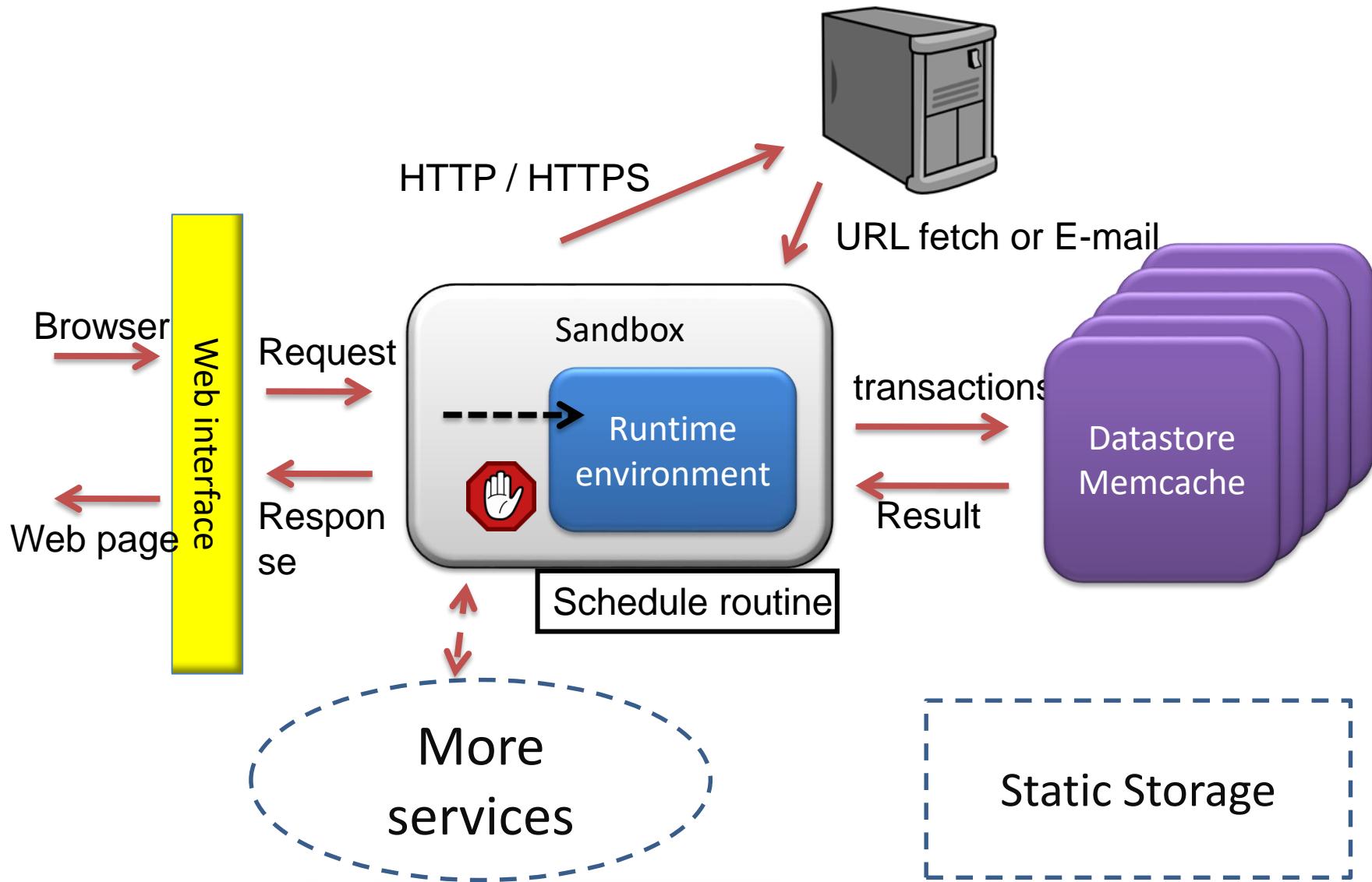
**Storage**

**Communications**

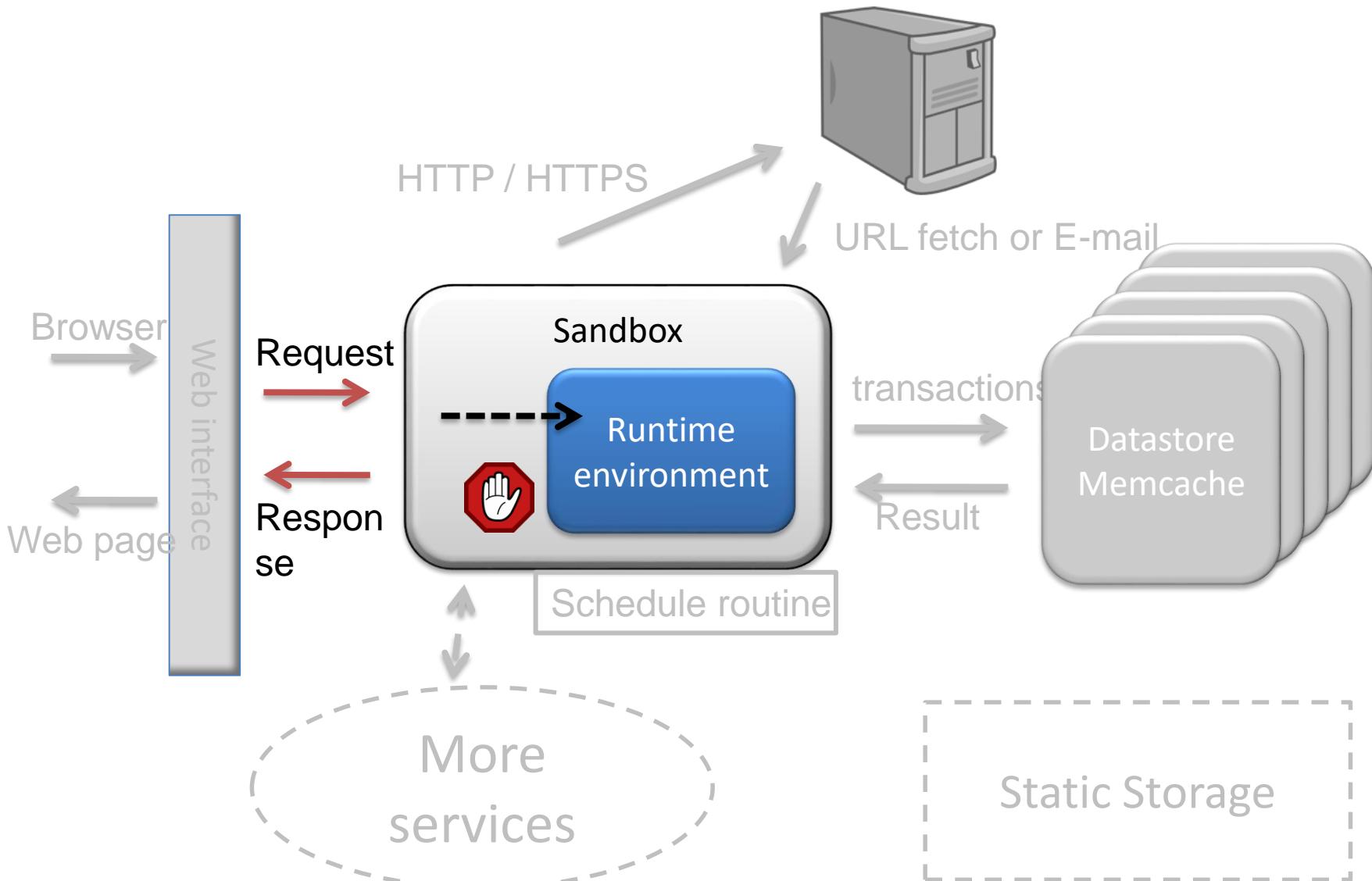
**Software Kernel**

**Firmware / Hardware**

# Google App Engine | Standard Environment



# Google App Engine | Standard Environment





# Google App Engine | Standard Environment

## **Application Standard Environment - Sandbox**

- Aplicatiile ruleaza intr-un mediu sigur, dar care asigura un acces limitat la sistemul de operare
- Este independent de nivelul hardware, sistem de operare sau localizarea fizica a serverului Web
- Aplicatiile nu pot scrie in sistemul de fisiere, pot doar citi ceea ce s-a incarcat prin codul aplicatiei





# Google App Engine | Standard Environment

## Application Standard Environment

Mecanism:

- La primirea cererii, AppEngine selecteaza serverul estimat ca fiind cel mai rapid, trimite cererea la aplicatie si returneaza clientului raspunsul
- Obs. Nu se pastreaza starea intre cereri => AppEngine asigura acelasi tratament tuturor cererilor, prin distribuirea traficului intre servere
- Aplicatiile pot accesa doar propriile fisiere din sistemul de fisiere, si nu ale altor aplicatii;
- Aplicatiile pot vedea variabilele de mediu setate de AppEngine dar manipularea acestora nu este neaparat persistenta intre cereri
  - Aplicatia trebuie sa foloseasca *Datastore* pentru a asigura persistenta intre cereri





# Google App Engine | Standard Environment

## Application Standard Environment

- Aplicatiile nu pot accesa facilitatile de retea la nivel hardware, dar se permite realizarea de operatii de retea prin utilizarea serviciilor
- O aplicatie poate accesa alte computere doar prin serviciu de email sau *URL fetch*
  - cereri HTTP(HTTPPs) folosind porturile standard
  - are limitari privind utilizarea CPU, a memoriei consumate per cerere
    - la o cerere se poate raspunde intr-un timp de pana la 60 de secunde
  - Obs. App Engine este optimizat pentru aplicatii care raspund in mai putin de o secunda
  - Daca aplicatia utilizeaza mai multi cicli de procesor, App Engine o va incetini pentru a nu fi afectate performantele celorlalte aplicatii care ruleaza pe aceeasi masina



## Application Standard Environment

- Obs. Mediul Python 2.7->... permite citirea, scrierea si modificarea *bytecode*
- **Python runtime**
  - Se foloseste o versiune a interpretorului cPython
  - Mecanism general: App Engine invoca o aplicatie Python folosind un mecanism CGI
  - Framework-uri Python cum ar fi Django, web2py, Pylons functioneaza cu App Engine
    - AppEngine include si un framework propriu Python

# Google App Engine | Standard Environment



## Application Standard Environment

### – GAE Java runtime environment

- Aplicatiile Java poate fi dezvoltate folosind Java JVM
- Se pot construi aplicatii apeland la tehnologii Java standard (servleturi Java, JavaServer Pages (JSPs),...)
- <https://cloud.google.com/appengine/docs/java/>

### – Go Runtime

- Ruleaza diverse versiuni de Go
- SDK include compilatorul Go si librariile standard
- Se furnizeaza un GoAPI pentru majoritatea serviciilor App Engine (Storage, Schedule, Communication, ....)
- Se pot folosi biblioteci *third-party*, conditia e ca acestea sa fie implementate doar in Go
- <https://cloud.google.com/appengine/docs/go/>



# Google App Engine | Standard Environment

## Application Standard Environment

- PHP runtime environment

Se creaza aplicatia folosind ***sandboxed PHP environment***

- App Engine PHP SDK include: server web pentru testarea PHP locala; serverul simuleaza serviciile AppEngine (inclusiv versiune locala a Google Accounts si abilitatea de a trimite mailuri de pe computerul local folosind App Engine API)
- Obs. Este nevoie de Python > 2.7, deoarece serverul de dezvoltare este o aplicatie Python
- Exista SDK pentru platforme ca: Linux, MacOS, Windows
- Se integreaza aplicatia App Engine cu Google Accounts pentru autentificare
- PHP runtime contine un wrapper ce permite accesul facil la serviciile oferite de Google Cloud Storage

<https://cloud.google.com/appengine/docs/php/>

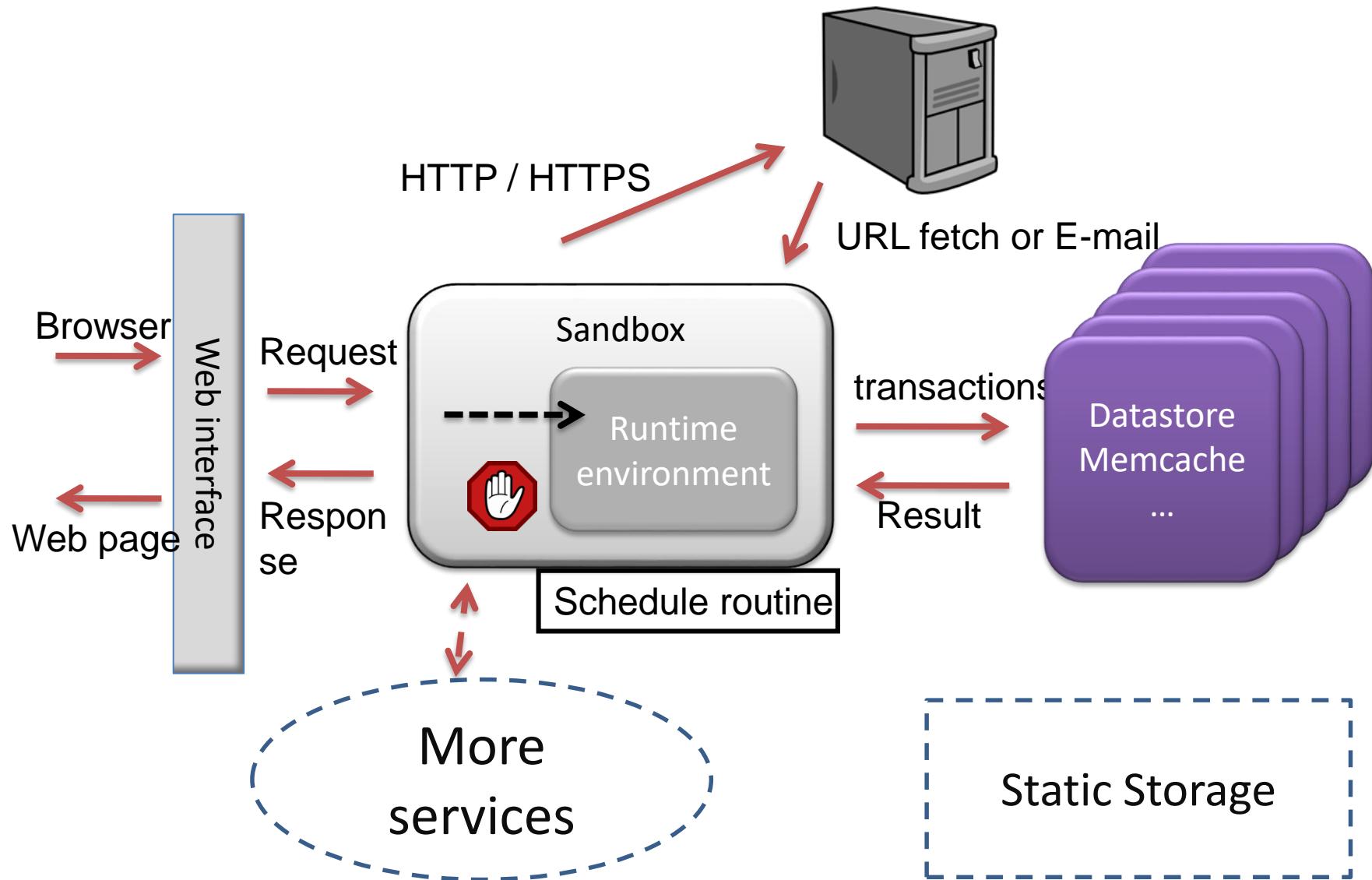
# Google App Engine | Standard Environment



## Application Standard Environment

- Aspecte privind invocarea unei aplicatii:
    - Mecanism: cererea este rutata la server, aplicatia este pornita, se creaza raspunsul, raspunsul este returnat clientului
    - Fiecare mediu ruleaza propriul interpretor (JVM sau Python) respectind restrictiile *sandbox* (e.g. incercarea de a folosi vre-o facilitate a limbajului sau a librariilor existente, care sa acceseze alte resurse decat cele permise va conduce la o exceptie)
    - Utilizarea pentru fiecare cerere a unui server diferit
      - => avantaj: scalare
      - => dezavantaj: este consumatoare de timp operatia de crea o instanta per fiecare cerere
- Solutie:
- AppEngine atenuiaza costurile de pornire prin mentinerea aplicatiei in memoria serverului, maxim posibil si reutilizarea inteligenta a serverelor
  - toate serverele au mediile *runtime* preincarcate inainte ca cererea sa ajunga la server

# Google App Engine | Standard Environment





## ***Storage Services***

- **Static**
  - Spatiu static (fisiere sursa a serviciilor Web, fisiere de configurare, imagini de background,...)
  - Blobstore
    - Contine fisiere mai mari de 1MB (imagini, video sau audio, etc)
- **Dinamic**
  - Datastore
    - Fiecare entitate nu depaseste 1MB
    - Serviciu furnizat in maniera “dynamic provisioning” ce suporta operatii dinamice de inserare/update/delete a datelor
  - Memcache
    - Folosit pentru a creste viteza interogarilor din datastore



## **Static File Servers**

- Multe site-uri web disponuند de resurse care nu suferă schimbari în timpul operațiilor obisnuite asupra site-ului (e.g. imagini, fisiere CSS, cod JavaScript, pagini HTML cu conținut static)
  - Sunt denumite *static files*
    - Furnizarea acestor resurse nu implica cod de aplicatie => nu necesita servere de aplicatii
    - AppEngine furnizeaza servere dedicate care furnizeaza acest tip de conținut - **Static File Servers**
- Pentru clientul final o astfel de resursa statică este similară cu orice altă resursă
- Dezvoltatorii pot configura ceteva aspecte privind modul de furnizare al resurselor statice (URL-uri, content types, instructiuni pentru browsere pentru a mentine copii ale fisierelor în cache, etc.)

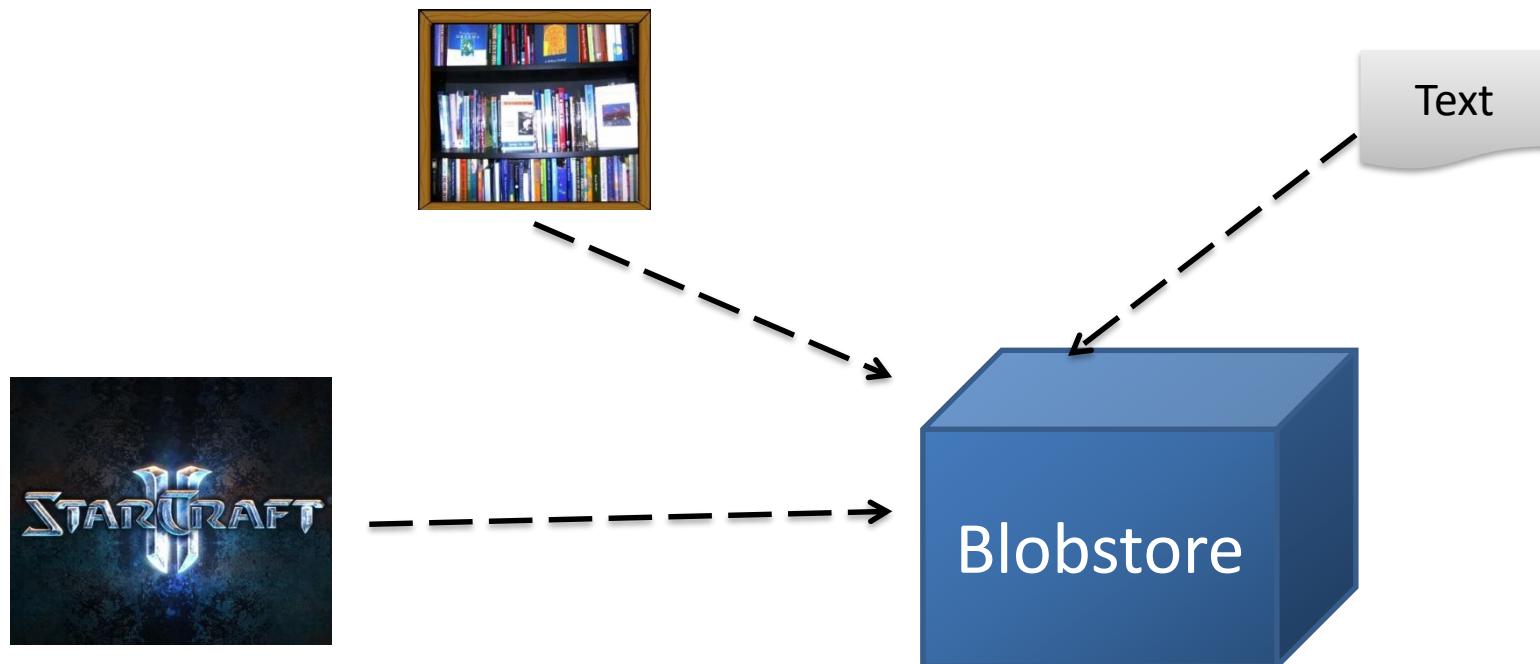
# Google App Engine | Standard Environment



## Storage Services

### – Blobstore

- *Binary large objects* – care sunt de dimensiune mai mare decat dimensiune permisa pentru obiectele din serviciul datastore (video, img,...)
- Sunt create prin incarcarea de fisiere ca rezultat al cererilor HTTP



# Google App Engine | Standard Environment



## Storage Services

### Datastore

- O aplicatie AppEngine stocheaza datele intr-una sau mai multe entitati de stocare (*datastore entities* sau *data objects*)
- *Entitatea* sunt asociate *properties* de tipul *name – value*

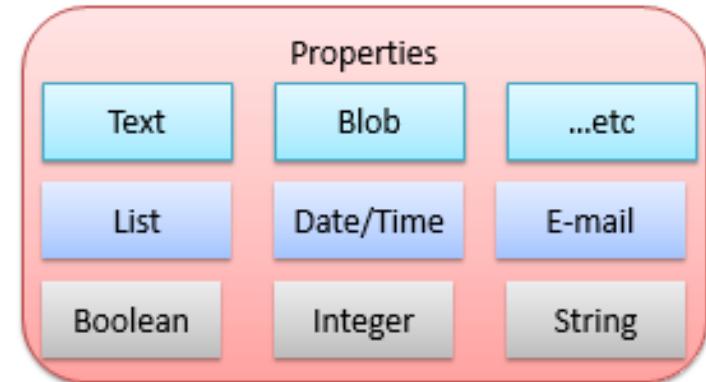




## Storage Services

### Datastore

- Proprietatile pot avea un anumit *tip*
- Constructorul proprietatii poate defini
  - Numele
  - Valoarea implicita
  - O lista de valori
  - ...



```
from google.appengine.ext import db
class Dog(db.Model):
    name    = db.StringProperty(default='dog')
    age     = db.IntegerProperty(required=True)
    weight  = db.IntegerProperty(indexed=False)
    status  = db.StringProperty( choices = ['awake', 'eat', 'play']
)
    photo   = db.BlobProperty()
```



## *Storage Services*

### **Datastore**

- O entitate are o **cheie unica (key)** care e furnizata de aplicatie sau de App Engine (dezvoltatorul decide)
  - Aceasta cheie nu este o proprietate elementara, ci este un aspect independent al entitatii
  - O cheie nu poate fi schimbată după ce entitatea a fost creata
  - Cunoscând cheia, se pot face interogări
- Cunoscând tipul entitatii și cheia asociată se poate determina unde este entitatea stocată în întreaga colecție de servere



## Storage Services

### Datastore

- Operatii:
    - *put(key)* – *upload* sau *update*
    - *delete(key)* – stergerea unei entitati
    - *get (key)* , ...
  - ? Similitudine cu baze de date relationale (... ? *entities* = *rows* din tabele; ?*properties* = *columns*)
    - Diferente:
      - O entitate de un anumit tip poate avea proprietati diferite fata de o alta entitate de acelasi tip
      - O entitate poate avea o proprietate cu acelasi nume dar de tip diferit, fata de o alta entitate de acelasi tip ca ea
- => entitatile din *datastore* sunt *schemaless* => flexibilitate marita & provocari in mentenanta



## Interogari si indexari

- Datastore foloseste *indexes* pentru fiecare interogare facuta de aplicatie
- Exemple de tipuri de interogari:
  - Dupa proprietatile entitatii si se va obtine lista entitatilor ordonate dupa valoarea proprietatii
  - Filtrare si sortare dupa chei





## Interogari si indexari

- Mecanism intern de executie a interogarilor

Baze de date relationale:

- interogarile sunt planificate si executate in timp real

App Engine Standard Environment

- Fiecare interogare are un index de corespondenta mentinut in *datastore*
- Atunci cand aplicatia realizeaza o interogare, *datastore* gaseste indexul corespunzator acelei interogari, scanarea pana la primul rand care se potriveste cu interogarea, si returneaza entitatea pentru fiecare rand consecutiv din index, procesul repetandu-se pana la primul rand care nu se potriveste cu interogarea
- Obs. App Engine trebuie sa stie dinainte ce interogari va face aplicatia; nu trebuie sa stie apriori valorile filtrelor, dar trebuie sa stie tipul entitatii dupa care se face interogarea, proprietatile dupa care se doreste filtrarea sau sortarea etc.



## Interogari si indexari

- App Engine furnizeaza un set de indecsi pentru interogari simple, bazandu-se pe ce proprietati si tipuri de entitati exista
  - Pentru interogari complexe, o aplicatie trebuie sa includa in configuratia sa specificatii pentru indecsi
    - App Engine SDK ajuta la producerea acestor fisiere de configurare (e.g. supervizarea interogarilor pe care dezvoltatorul le-a realizat in timpul dezvoltarii aplicatiei – folosind serverul de dezvoltare pe computerul local. La incarcarea aplicatiei, *datastore* va crea indecsi pentru fiecare interogare a aplicatiei realizata in faza de dezvoltare)
      - Obs. Indecsii se pot configura si manual
  - Atunci cind aplicatia creaza noi entitati, si actualizeaza pe cele existente, *datastore* actualizeaza fiecare index
- => interogarile sunt rapide



## Tranzactii

- Cand o aplicatie are mai multi clienti care realizeaza operatii de citire/scriere simultana asupra acelorasi date => necesitatea mentinerii consistentei datelor
- Cand o aplicatie actualizeaza proprietatile unei entitati, App Engine asigura update-ul complet sau operatia de update va da eroare  $\Leftrightarrow$  update-ul unei entitati se realizeaza printr-o *tranzactie* (Proprietate: orice tranzactie este atomica)
- Cand o aplicatie doreste operatii cu mai multe entitati intr-o singura tranzactie, aplicatia va crea un *entity group*, asupra caruia App Engine va supraveghea realizarea tranzactiilor
- App Engine utilizeaza *optimistic concurrency control* –> aplicatia trebuie sa incerce realizarea unei tranzactii de mai multe ori inainte de a returna eroare
- Citirea unei entitati nu poate esua din cauza concurentei, aplicatia avand acces la cea mai stabila stare a entitatii



## *Storage Services*

- ***Statefull services***
  - Blobstore
  - Datastore
  - Google Cloud SQL
  - ***Memcache (memory cache)*** – este un serviciu de stocare cheie-valoare
    - Principalul avantaj fata de datastore: rapiditate in operatiile de stocare si regasire
    - Stocheaza valori in memorie si nu pe disk
    - Este distribuit ca si *datastore*, deci fiecare cerere vede aceeasi pereche cheie-valoare
    - Nu este persistent: daca un server esueaza, memoria este stearsa
    - Este utilizat pentru cash-ul rezultatelor celor mai frecvente interogari sau calcule
    - Se furnizeaza mai multe tipuri:
      - **Shared memcache**
      - **Dedicated memcache**

# Google App Engine | Standard Environment



## ***Communication Services***

- ***Stateless service APIs***

- *URL Fetch*

- Permite aplicatiilor App Engine sa acceseze alte resurse web (cereri HTTP(S) pentru obtinerea de pagini web, interactiune cu servicii Web)
    - Deoarece serverele *remote* pot raspunde greu, URL Fetch suporta *fetching URL* in background, in timp ce aplicatiile pot efectua si alte operatii
    - Obs. O astfel de operatie trebuie sa existe doar pe timpul de viata al aplicatiei
    - Aplicatia poate seta un deadline, pentru operatia de *fetch* => apelul se va incheia daca raspunsul nu este primit in timpul corespunzator

- *Mail*

- Aplicatiile pot trimite si primi mesaje sub forma de cereri HTTP initiate de App Engine si trimise catre aplicatie
    - Exemplu: notificarea utilizatorilor, confirmarea actiunilor utilizatorilor, validarea informatiilor de contact, etc.



## ***Communication Services***

- ***Stateless service APIs***
  - *Sockets*
    - Comportamentul acestora variaza in functie de mediul de rulare
  - Twilio <sup>Third-party</sup>
    - Permite realizarea de apeluri telefonice
    - *Twillio Client* permite realizarea de apeluri VOIP de pe orice dispozitiv (suporta WebRTC)
    - <https://www.twilio.com/>
  - Google Cloud Endpoints
    - Consta din instrumente si librarii care permit generarea de API-uri pe baza unei aplicatii, a.i. sa se usureze accesul la date din alte aplicatii

# Google App Engine | Standard Environment



## *Process Management Services*

- *Task Queues*

Idea: o aplicatie web trebuie sa ofere rezultatul cat mai rapid

- Problema: uneori sunt multe operatii de facut, care necesita mai mult timp, si care pot fi organizate in task-uri
- Solutia: *task queues*
  - Permite definirea de task-uri care se pot executa in background atunci cand resursele sistemului permit acest lucru
- Tipuri:
  - *Push Queues* - asigura faptul ca aceste taskuri sunt executate automat de sistem care scaleaza si realizeaza procesarile in functie de necesitati; *automate scaling* – serviciile trebuie sa se finalizeze in maxim 10 min; *manual scaling* – serviciile pot rula pana la 24h
  - *Pull Queues*
    - » ofera un control mai bun asupra momentului in care task-urile sunt executate, intr-un anumit interval de timp, dar necesita effort crescut in managementul proceselor
    - » asigura integrarea aplicatiei cu *Task Queue REST API* - care permite managementul extern al task-urilor existente

# Google App Engine | Standard Environment



## Process Management Services

- *Migrarea Pull Queues la Pub/Sub*
- Se poate utiliza Pub/Sub ca si un *Task Queues pull queue*: *Subscriptions to a topic do not expire and can exist simultaneously for multiple workers. This means that a message can be processed by more than one worker, which is one of the primary use cases for Pub/Sub. To recreate your Task Queues pull queues as Pub/Sub pull subscriptions, create a topic for each worker and subscribe only the associated worker to the topic. This ensures that each message is processed by exactly one worker as in Task Queues.*

Task Queues workflow	Pub/Sub workflow
You create the pull queue	You create the topic and subscribe your subscriber (i.e. worker) to the topic
You create and enqueue the task	You create the message and publish it to the topic
The worker leases the task	The subscriber pulls the message from the topic
The worker processes the task	The subscriber processes the message
The worker deletes the task from the queue	The subscriber acknowledges the message
The lease expires	The topic deletes the message when all of its subscribers have acknowledged the message

# Google App Engine | Standard Environment



## ***Process Management Services***

- *Migrarea Push Queues to Cloud Tasks*
- In Cloud Tasks, all queues operate as push queues
- Cloud Tasks is a fully managed service that allows you to manage the execution, dispatch and delivery of a large number of distributed tasks.
- You can asynchronously perform work outside of a user request.
- Your tasks can be executed on App Engine or any arbitrary HTTP endpoint.

[ <https://cloud.google.com/appengine/docs/standard/python/taskqueue/push/migrating-push-queues>  
<https://cloud.google.com/tasks/docs>]

## Pricing

Cloud Tasks has its own [pricing](#). As with Task Queues, sending requests to your App Engine app with a task can cause your app to incur costs.

## Quotas

The Cloud Tasks [quotas](#) are different from the quotas for Task Queues. Like with Task Queues, sending requests to your App Engine app from Cloud Tasks might impact your App Engine [request quotas](#).

# Google App Engine | Standard Environment



## ***Process Management Services***

- *Scheduled task* (sau *cron jobs*)
  - Permite executarea de task-uri la interval fixe de timp
  - Utile pentru mentenanta periodica (update a unor date din cache la fiecare 10 minute, ...) sau trimitera de mesaje de notificare in fiecare zi
  - Aplicatiile Free pot avea pana la 20 de astfel de procese (100 in celelalte cazuri)

## ***Computation Services***

- *Images*
  - Permite realizarea de operatii asupra imaginilor (redimensionare, crop, flip, ...)
- *MapReduce*
  - model de programare pentru procesarea de cantitati mari de date
  - Foloseste Datastore si TaskQueues



## Servicii

<a href="#">App Identity</a>	A framework that provides access to the application's identity, and the ability to assert this identity using OAuth.
<a href="#">OAuth</a>	Uses the OAuth protocol to provide a way for your app to authenticate a user who is requesting access without asking for credentials (username and password)
<a href="#">Remote</a>	Access App Engine services from any application. For example, access a production datastore from an app running on your local machine.
<a href="#">Services</a>	Factor applications into logical components that can share stateful services and communicate in a secure fashion.
<a href="#">Traffic Splitting</a>	Routes incoming requests to different versions of your app, allowing you to do A/B testing and roll out new features incrementally.
<a href="#">Users</a>	Allows applications to sign in users with Google Accounts, and address these users with unique identifiers.

....

# Google App Engine | Standard Environment

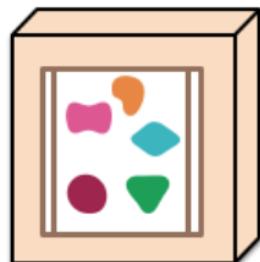


*Cum ne gandim aplicatia .... Microservices?*

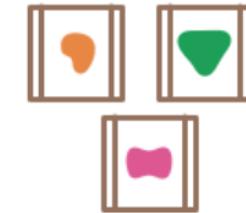
*A monolithic application puts all its functionality into a single process...*



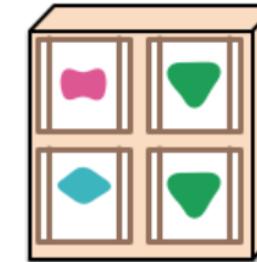
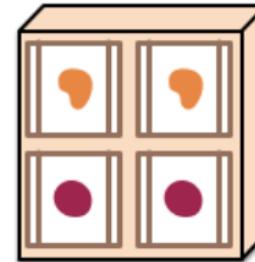
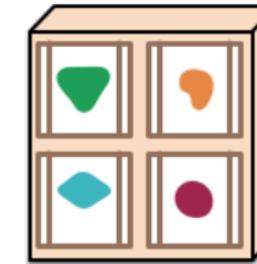
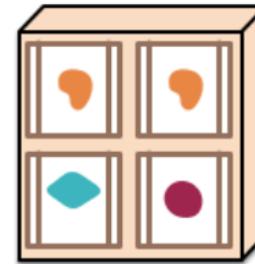
*... and scales by replicating the monolith on multiple servers*



*A microservices architecture puts each element of functionality into a separate service...*



*... and scales by distributing these services across servers, replicating as needed.*



# Google App Engine | Standard Environment

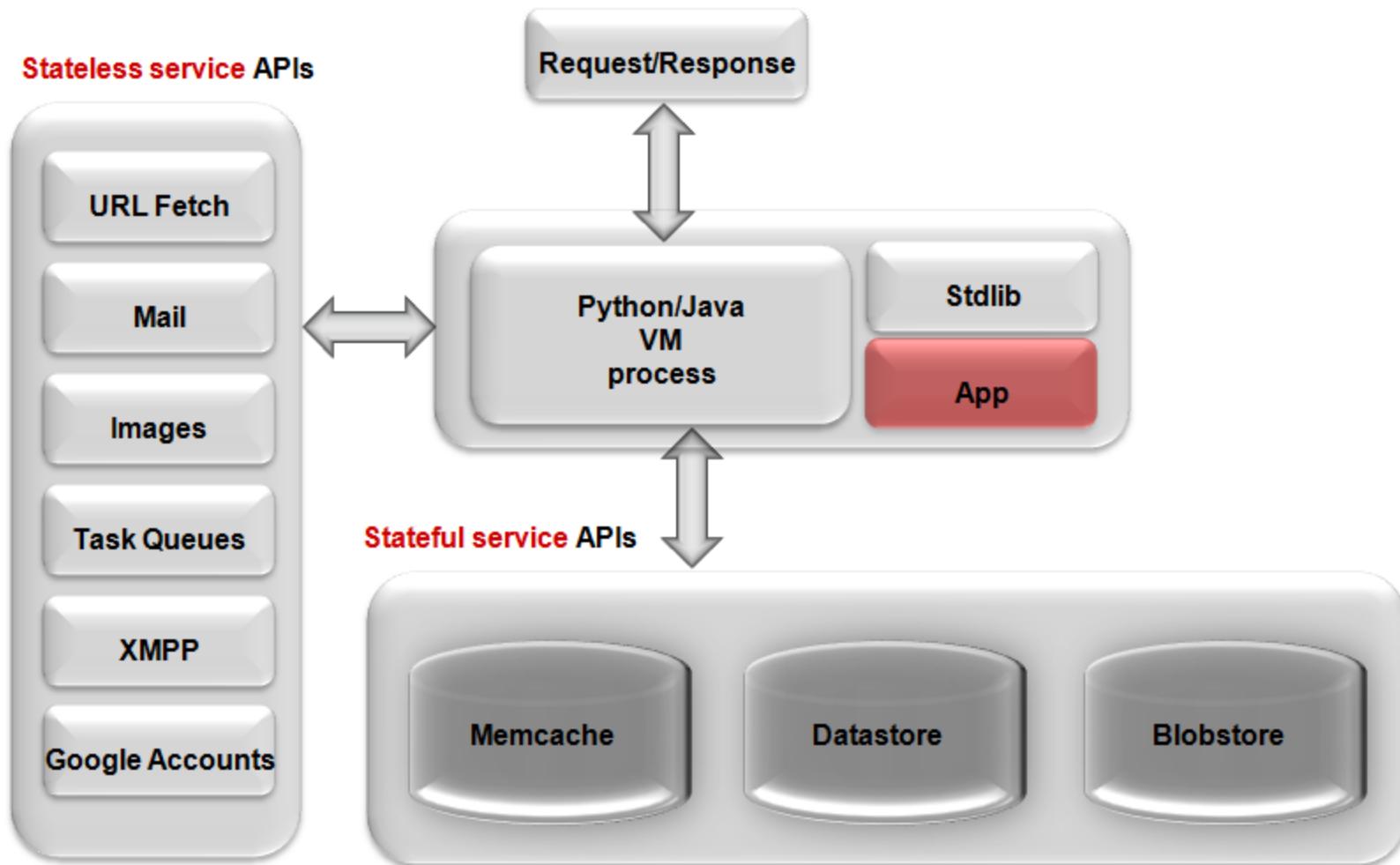


Figura. Posibil mod de a utiliza serviciile GAE

# Google App Engine



Scalability

Lower Total Cost of Ownership

Rich set of APIs

Fully featured SDK for local development

Ease of deployment

Web administration console and diagnostic utilities



[<http://www.slideshare.net/rajdeep/introduction-to-google-app-engine-presentation>]

88

# Google App Engine



## Concluzii

- GAE – permite rularea de aplicatii web, mobile, ...
- Configurare rapida
- Scalabilitate
- Securitate
- Scalabilitate
  - Totul este proiectat ca sa scaleze
  - *Low – usage apps*: multiple aplicatii per host-ul fizic
  - *High- usage apps*: multiple hosturi fizice per aplicatie
- ...



# Google App Engine

## Costuri:

- <http://code.google.com/appengine/docs/whatisgoogleappengine.html>

*“Not only is creating an App Engine application easy, it's free! You can create an account and publish an application that people can use right away at no charge, and with no obligation. An application on a free account can use up to 1 GB of storage and up to 5 million page views a month. When you are ready for more, you can enable billing, set a maximum daily budget, and allocate your budget for each resource according to your needs. You can register up to 10 applications per developer account....”*

<http://code.google.com/appengine/docs/billing.html>

Exemplu: Costuri pentru operatii in datastore

Martie 2013

Operation	Cost
Write	\$0.10 per 100k operations
Read	\$0.07 per 100k operations
Small	\$0.01 per 100k operations

Martie 2014

Operation	Cost
Write	\$0.09 per 100k operations
Read	\$0.06 per 100k operations
Small	\$0.01 per 100k operations

Martie 2015

Operation	Cost
Read / Write	\$0.06 per 100,000 operations
Small	Free



# Google App Engine

Resource	Free quota per day	Unit	Price beyond the free quota per unit
Stored data	1 GB	per GB per month	\$0.18
Entity reads	50,000	per 100K entities	\$0.06
Entity writes	20,000	per 100K entities	\$0.18
Entity deletes	20,000	per 100K entities	\$0.02
Small operations	Unlimited. Includes calls to allocate Cloud Datastore IDs, keys-only queries, and projection queries that do not use the distinct on clause. A keys-only query or a projection query that does not use the distinct on clause is counted as a single entity read for the query itself. The individual results are counted as small operations.		

[2017 -  
<https://cloud.google.com/appengine/pricing>]



# Google App Engine

Iowa (us-central1) ▾

Instance class	Cost per hour per instance
B1	\$0.05
B2	\$0.10
B4	\$0.20
B4_1G	\$0.30
B8	\$0.40
F1	\$0.05
F2	\$0.10
F4	\$0.20
F4_1G	\$0.30

Instance Class	Memory Limit	CPU Limit	Supported Scaling Types
F1 (default)	128 MB	600 MHz	automatic
F2	256 MB	1.2 GHz	automatic
F4	512 MB	2.4 GHz	automatic
F4_1G	1024 MB	2.4 GHz	automatic
B1	128 MB	600 MHz	manual, basic
B2 (default)	256 MB	1.2 GHz	manual, basic
B4	512 MB	2.4 GHz	manual, basic
B4_1G	1024 MB	2.4 GHz	manual, basic
B8	1024 MB	4.8 GHz	manual, basic

[2019 -  
<https://cloud.google.com/appengine/pricing>]



# Google - Flexible Environment

Applications running in the [App Engine flexible environment](#) are deployed to virtual machine types that you specify. These virtual machine resources are billed on a per-second basis with a 1 minute minimum usage cost.

This table summarizes the hourly billing rates of the various computing resources:

Iowa (us-central1)		
Resource	Unit	Unit cost
vCPU	per core hour	\$0.0526
Memory	per GB hour	\$0.0071
Persistent disk	per GB per month	\$0.0400

[2019 - <https://cloud.google.com/appengine/pricing>]



# Google - Flexible Environment

## Google Cloud Datastore calls

Cloud Datastore operations are billed as follows:

Iowa (us-central1)				
Resource	Free quota per day	Unit	Price beyond the free quota per unit	
Stored data	1 GB	per GB per month	\$0.18	Search
Entity reads	50,000	per 100K entities	\$0.06	Fees for use of the Search API are listed in the table below. Refer to the <a href="#">Java</a> and <a href="#">Python</a> documentation for a detailed description of each type of Search call.
Entity writes	20,000	per 100K entities	\$0.18	Iowa (us-central1)
Entity deletes	20,000	per 100K entities	\$0.02	
Small operations	Unlimited. Includes calls to allocate Cloud Datastore IDs, keys-only queries, and queries that do not use the distinct on clause. A keys-only query or a projection query with a distinct on clause is counted as a single entity read for the query itself. The insertion, update, and delete operations are counted as small operations.			
Iowa (us-central1)				
Resource	Unit	Price per unit		
Total storage (documents and indexes)	per GB per month	\$0.18		
Queries	per 10K queries	\$0.50		
Indexing searchable documents	per GB	\$2.00		

[2019 - <https://cloud.google.com/appengine/pricing>]



# Google - Flexible Environment

Iowa (us-central1)		
Resource	Unit	Unit cost (in US \$)
Outgoing network traffic - standard environment*	Gigabytes	\$0.12
Outgoing network traffic - flexible environment	Gigabytes	<a href="#">Google Compute Engine Network Rates</a>
Incoming network traffic	Gigabytes	Free
Blobstore stored data**	Gigabytes per month	\$0.026
Dedicated memcache	Gigabytes per hour	\$0.06
Logs API	Gigabytes	\$0.12
Sending email, shared memcache, cron, APIs (Task Queues, Image, Files, Users)		No Additional Charge

[2019 - <https://cloud.google.com/appengine/pricing>]



# Bibliografie

- <https://cloud.google.com/>
- Mark C. Chu-Carroll, Code in the Cloud, Programming Google App Engine, 2011
- Dan Sanderson, Programming Google App Engine, O'Reilly, 2010
- Implementing and Developing Cloud Computing Applications, DAVID E.Y. SARNA, CRC Press, Taylor&Francis Group, 2011
- Cloud Computing, Software Engineering Fundamentals, J. Heinzelreiter, W. Kurschl, [www.fh-hagenberg.at](http://www.fh-hagenberg.at)
- [Rossum, 2008] Guido van Rossum, Google App Engine, Stanford EE380 Colloquium, Nov 5, 2008
- <http://code.google.com/appengine/>
- <http://www.python.org/download/>
- <https://www.ascamso.com/905-2/>
- <https://www.martinfowler.com/articles/microservices.html>

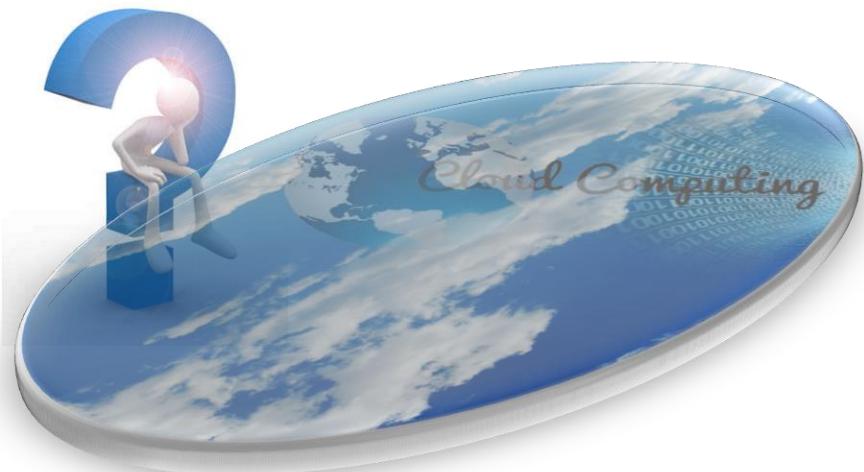
# Rezumat

- Google in Cloud
  - ...servicii
  - Instrumente (GWT, GAS,
  - Caracteristici
  - Aspecte arhitecturale
  - Concluzii

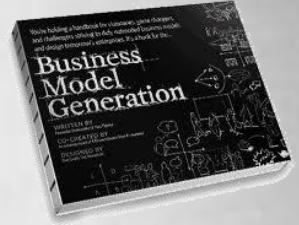


Universitatea “Alexandru Ioan Cuza”  
Facultatea de Informatică

Întrebări?



Based on



# THE Business Model Canvas

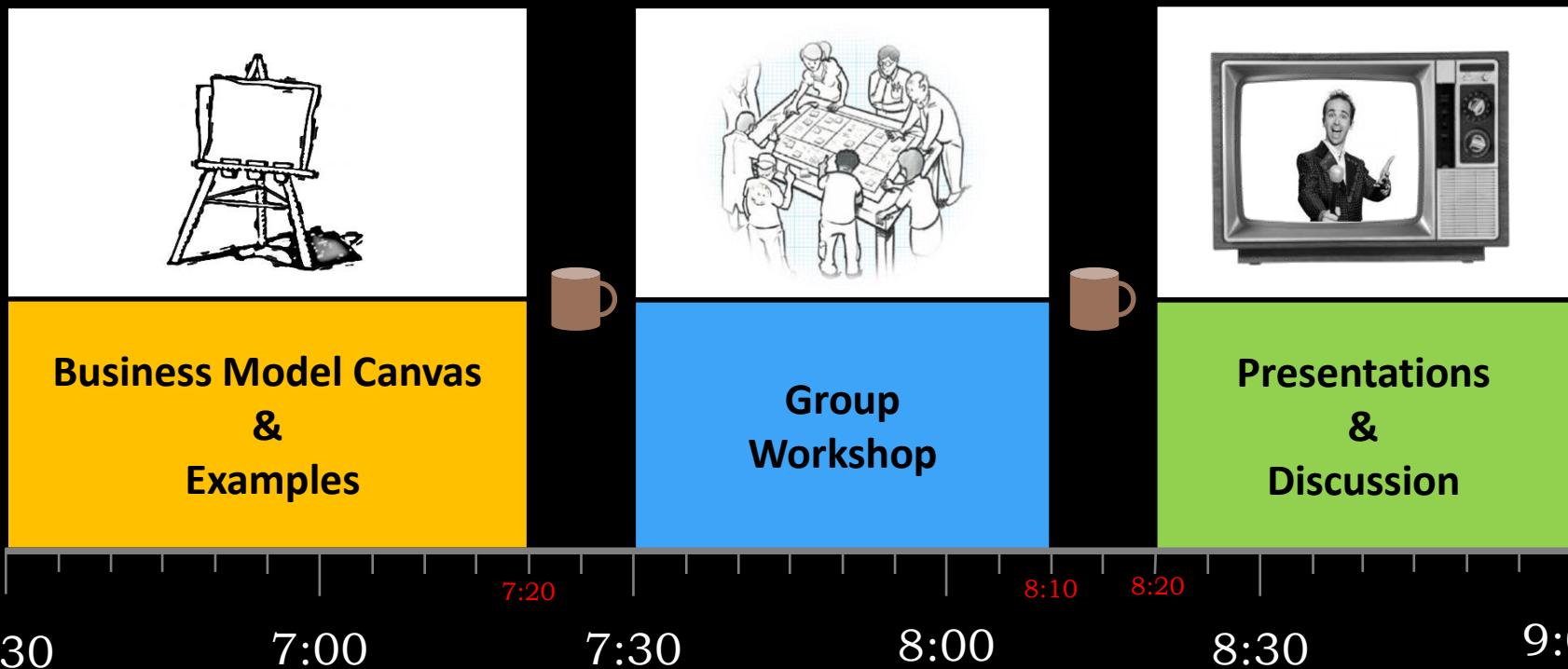


Nov 8<sup>th</sup> 2011

Created by: Emad Saif  
[www.emadsaif.com](http://www.emadsaif.com)



# Agenda



# First Plain Paper Photocopier - 1950

Fundamental new way of office copying

...makes copies on ordinary paper!

No wet chemicals ... no waste

Neither expensive sensitized paper, nor liquid chemicals are required. And with no adjustments to make or extensive settings, there's no waste of materials. Copies can be made at once, and the quality of reproduction is superb!

Copies all colors including reds & blues

With sharp black-and-white fidelity ... nothing left out. Copies everything—written, typed, printed, stamped, or drawn. Copies from any original ... even paper as thick as book covers. Just push a button—copied!

About 1¢ per copy for supplies

No heating of copy paper, or re-heating of original. Any number of copies delivered automatically—at the touch of a button—for about 1¢ per copy for supplies. For information, write XEROX CORPORATION, 800 BOSTON ROAD, WENDELL ST., WENDELL, MASS.

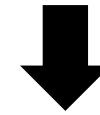
**NEW XEROX® 914  
OFFICE COPIER**

Courtesy of Xerox Historical Archives

PHOTO: XEROX HISTORICAL ARCHIVES

- ✓ Easy to use
- ✓ No risk on originals
- ✓ Low operating cost
- ✓ Use plain paper

**TOO EXPENSIVE!**  
to sell to customers!

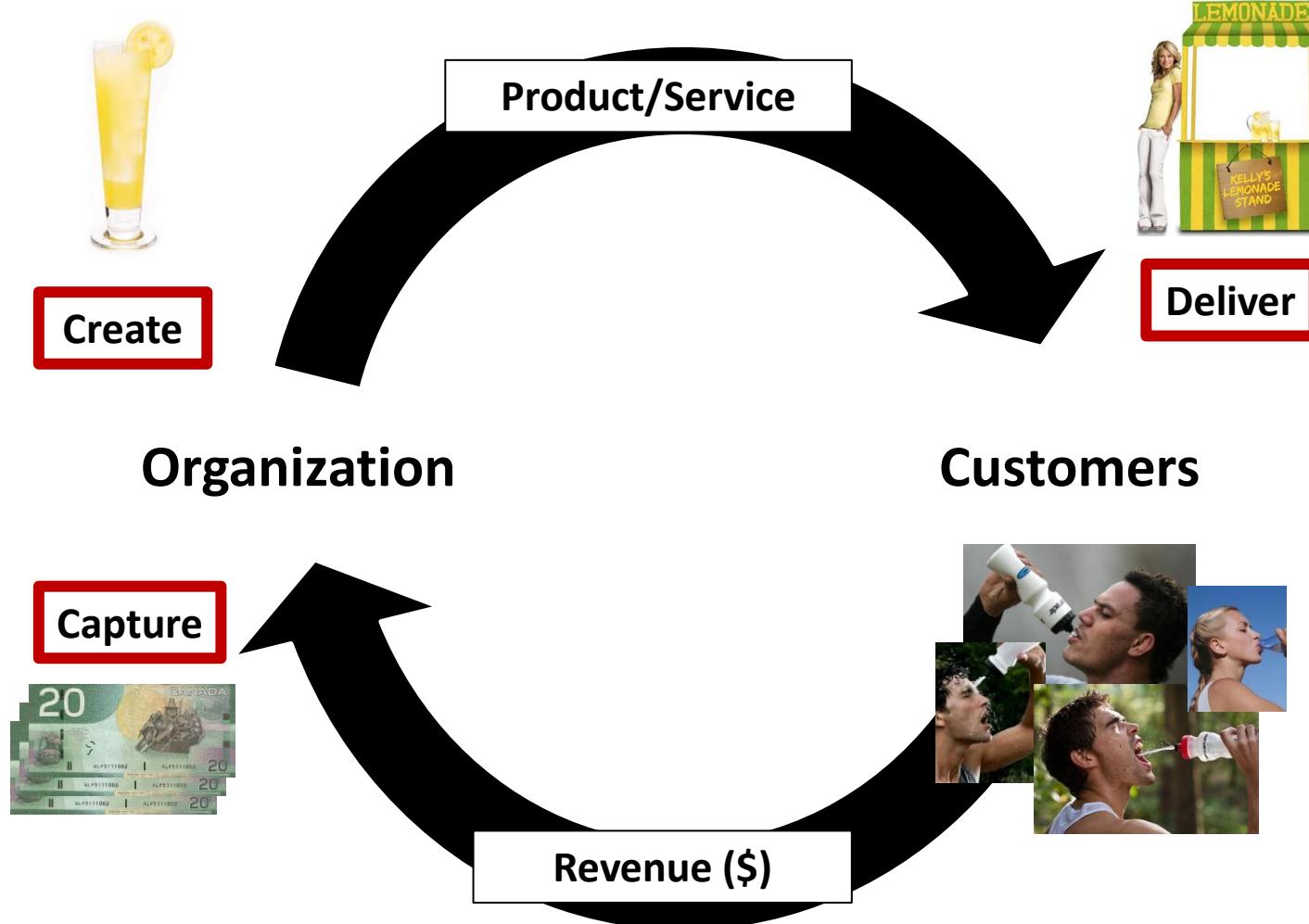


**Lease**  
\$25/month  
+  
4¢/copy (min of \$49/month)

# business model ?

“A business model describes the rationale of how an organization **creates, delivers, and captures value**”

# BUSINESS MODEL



business

model

canvas

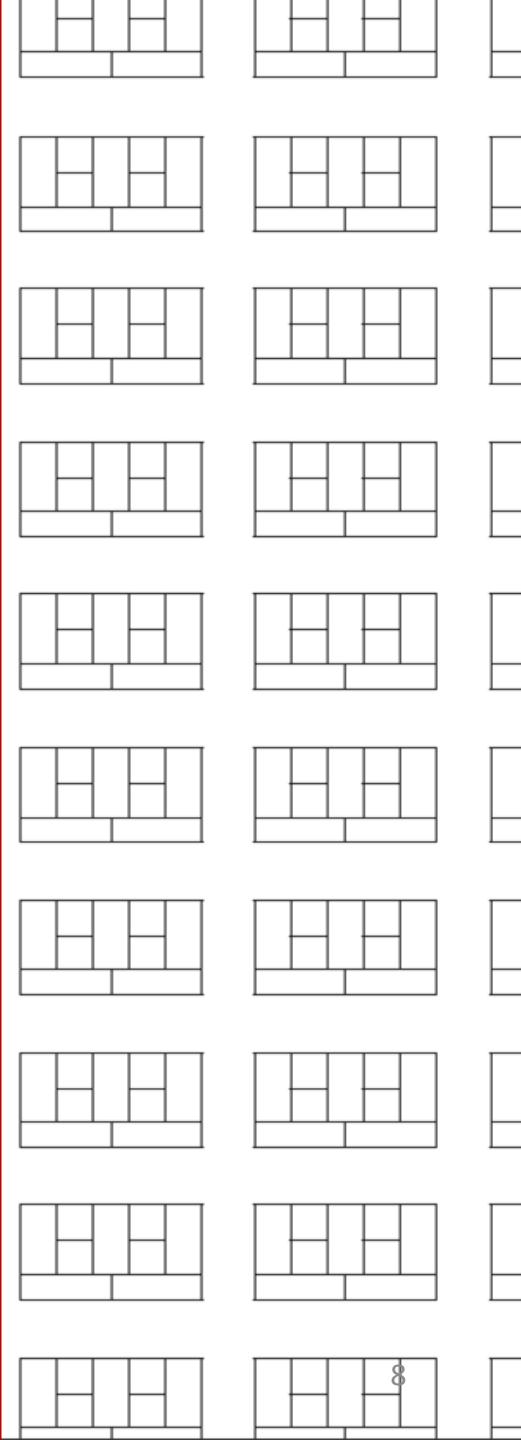


tool to create & analyze  
business models...



# YOU CAN

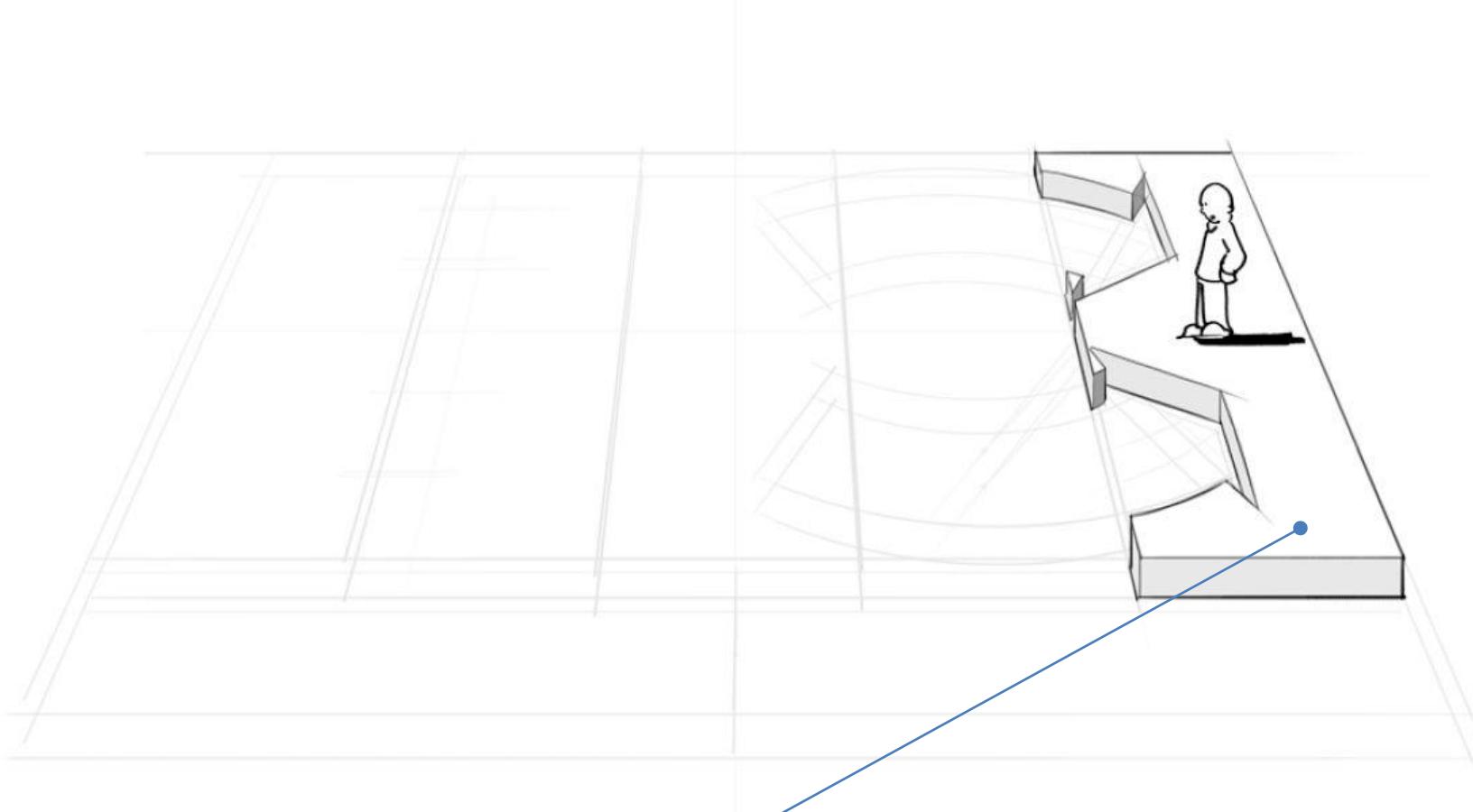
- Create new business models easily
- Analyze & update your existing business model



9

building  
blocks

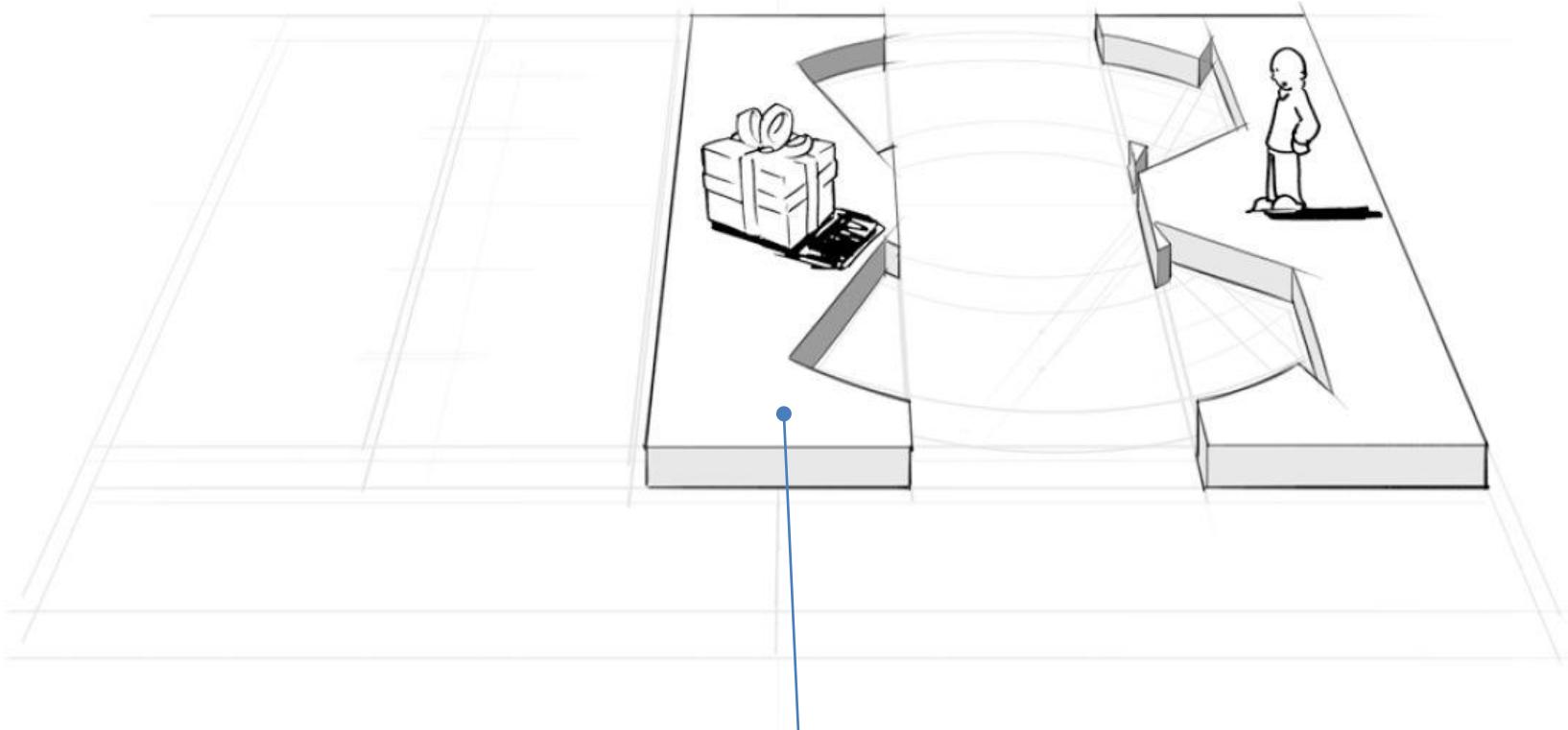
# Customer Segments



which customers and users are you serving?  
which jobs do they really want to get done?

drawings by JAM

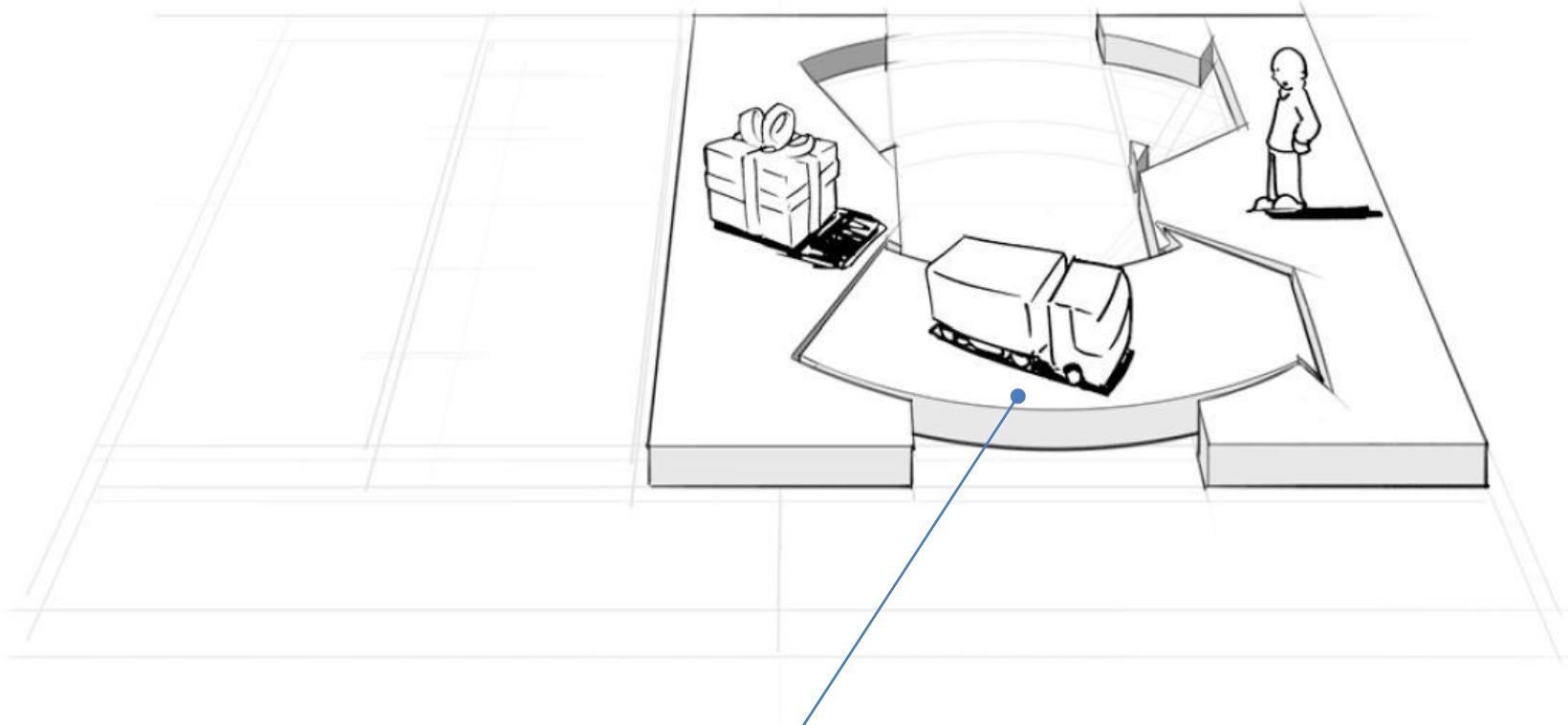
# Value Proposition



what are you offering them? what is that  
getting done for them? do they care?

drawings by JAM

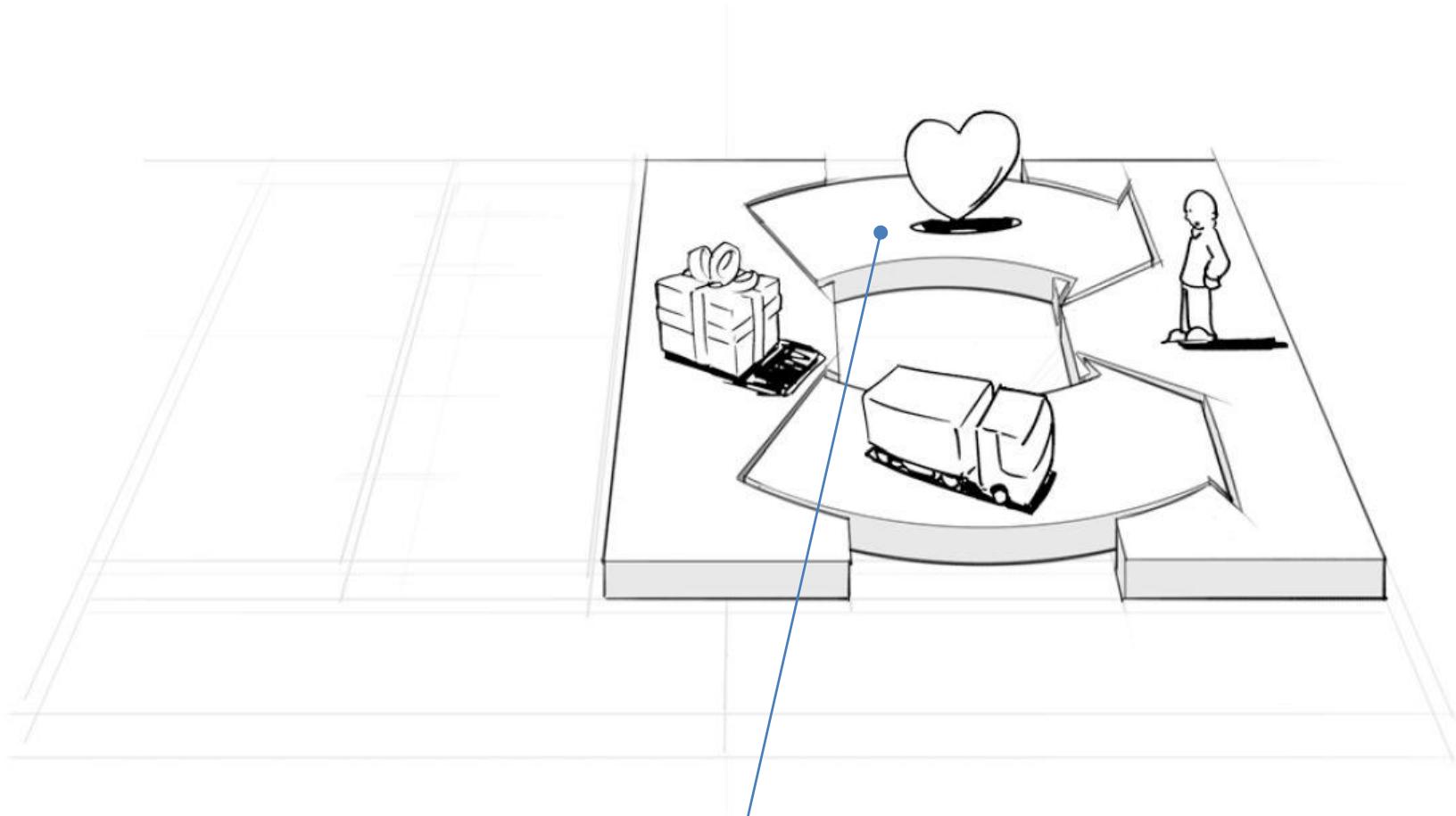
# Channels



how does each customer segment want to be reached?  
through which interaction points?

drawings by JAM

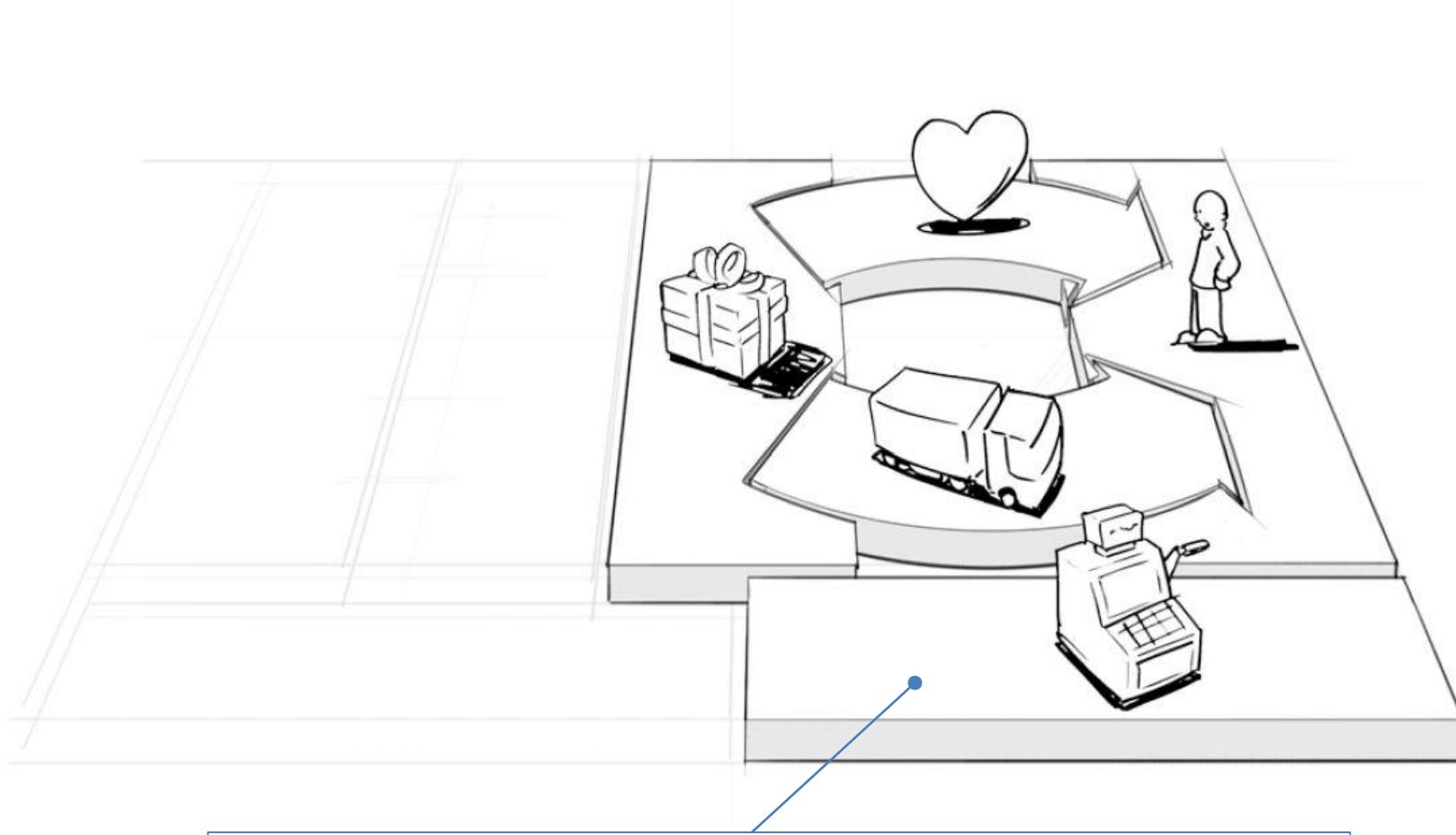
# Customer Relationships



what relationships are you establishing with each segment?  
personal? automated? acquisitive? retentive?

drawings by JAM

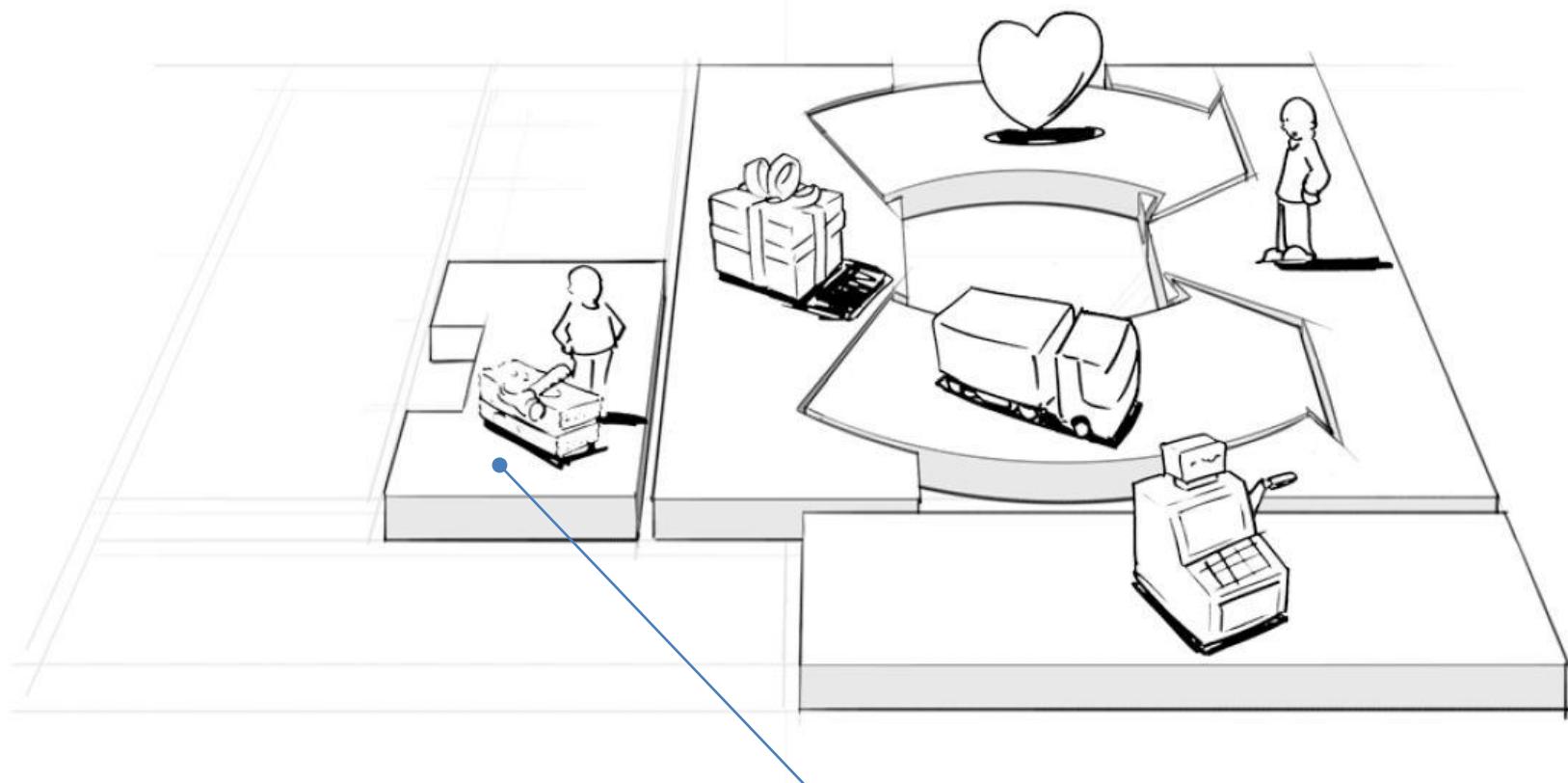
# Revenue Streams



what are customers really willing to pay for? how?  
are you generating transactional or recurring revenues?

drawings by JAM

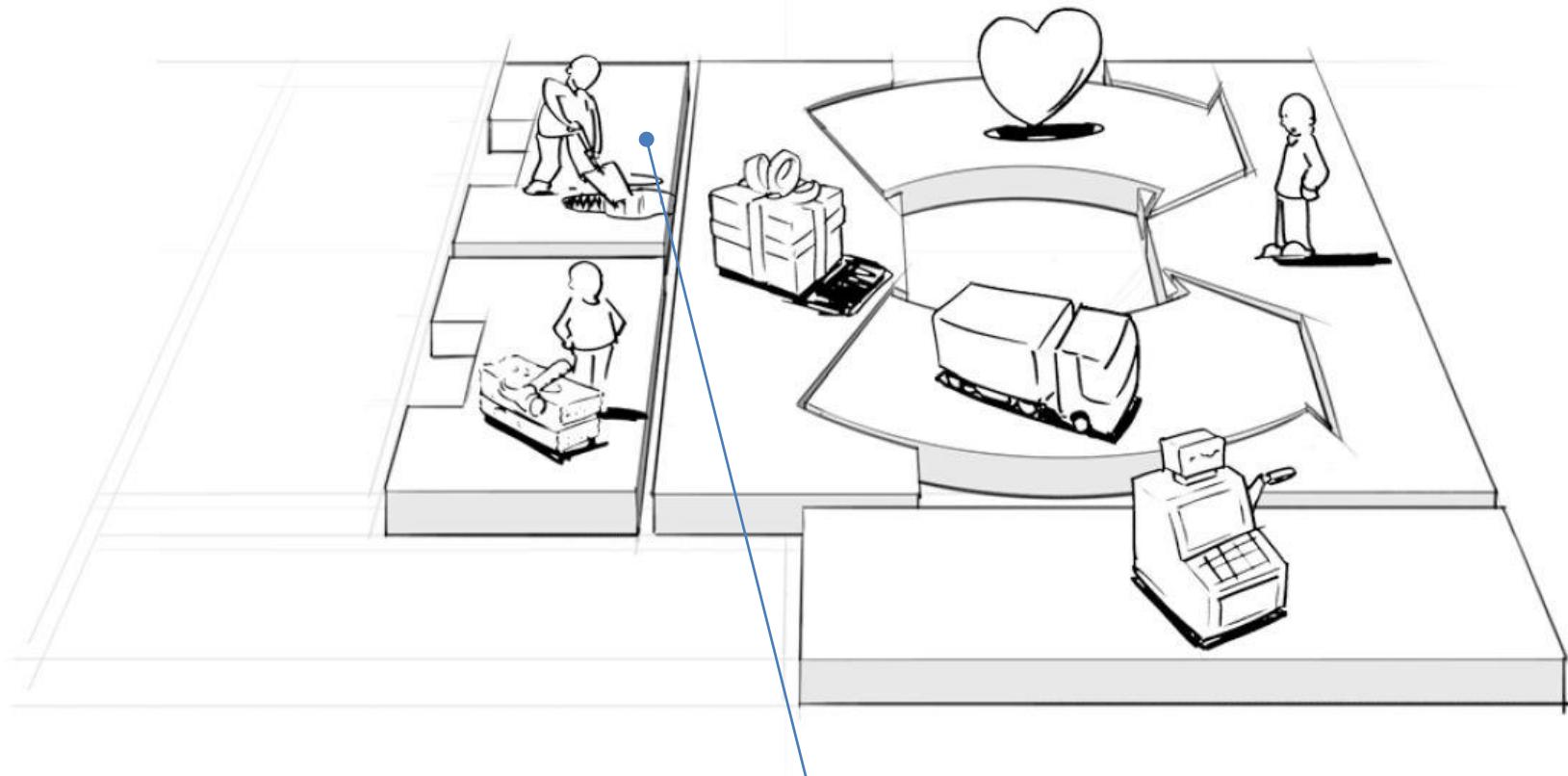
# Key Resources



which resources underpin your business model?  
which assets are essential?

drawings by JAM

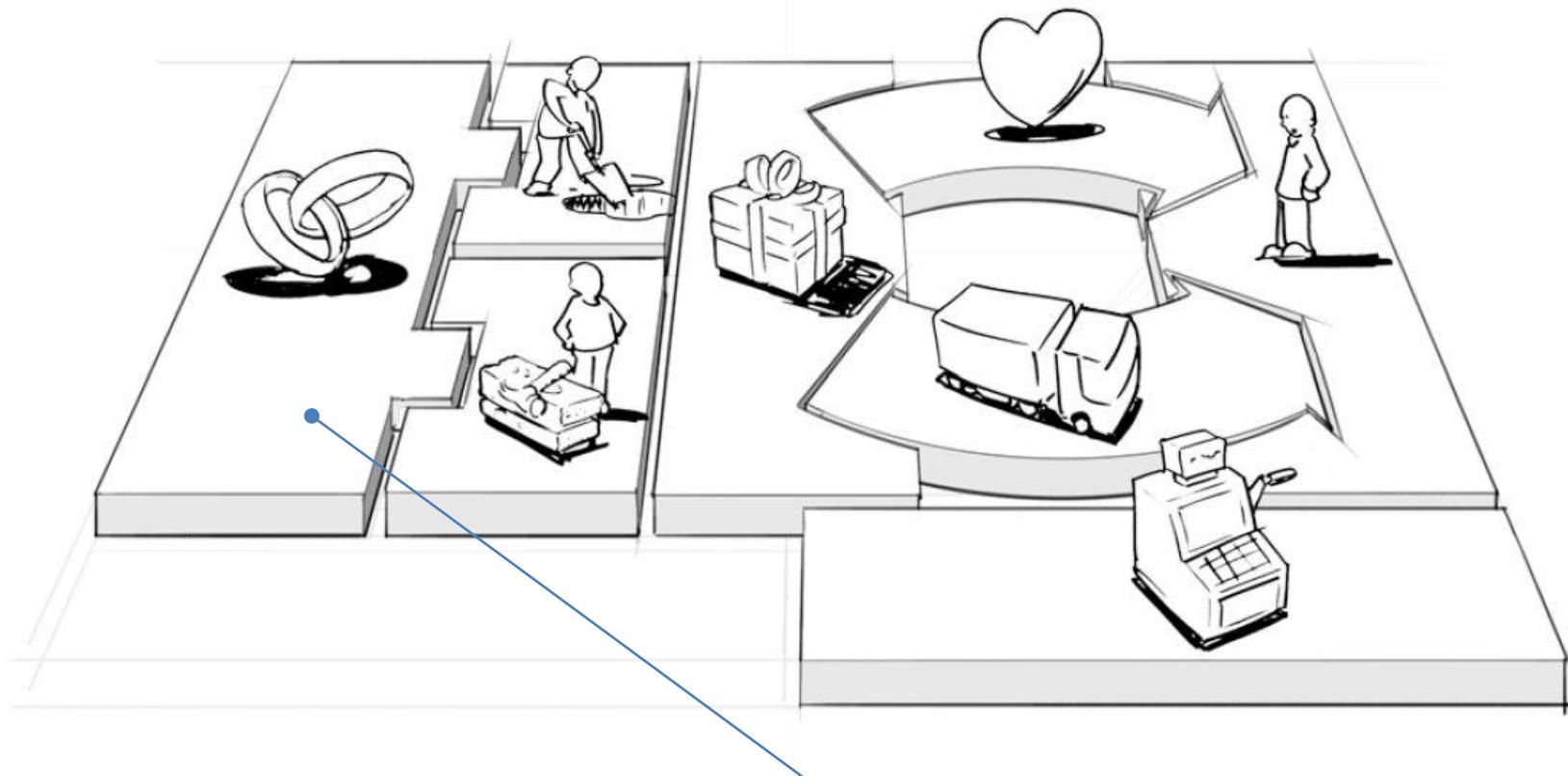
# Key Activities



which activities do you need to perform well in your business model? what is crucial?

drawings by JAM

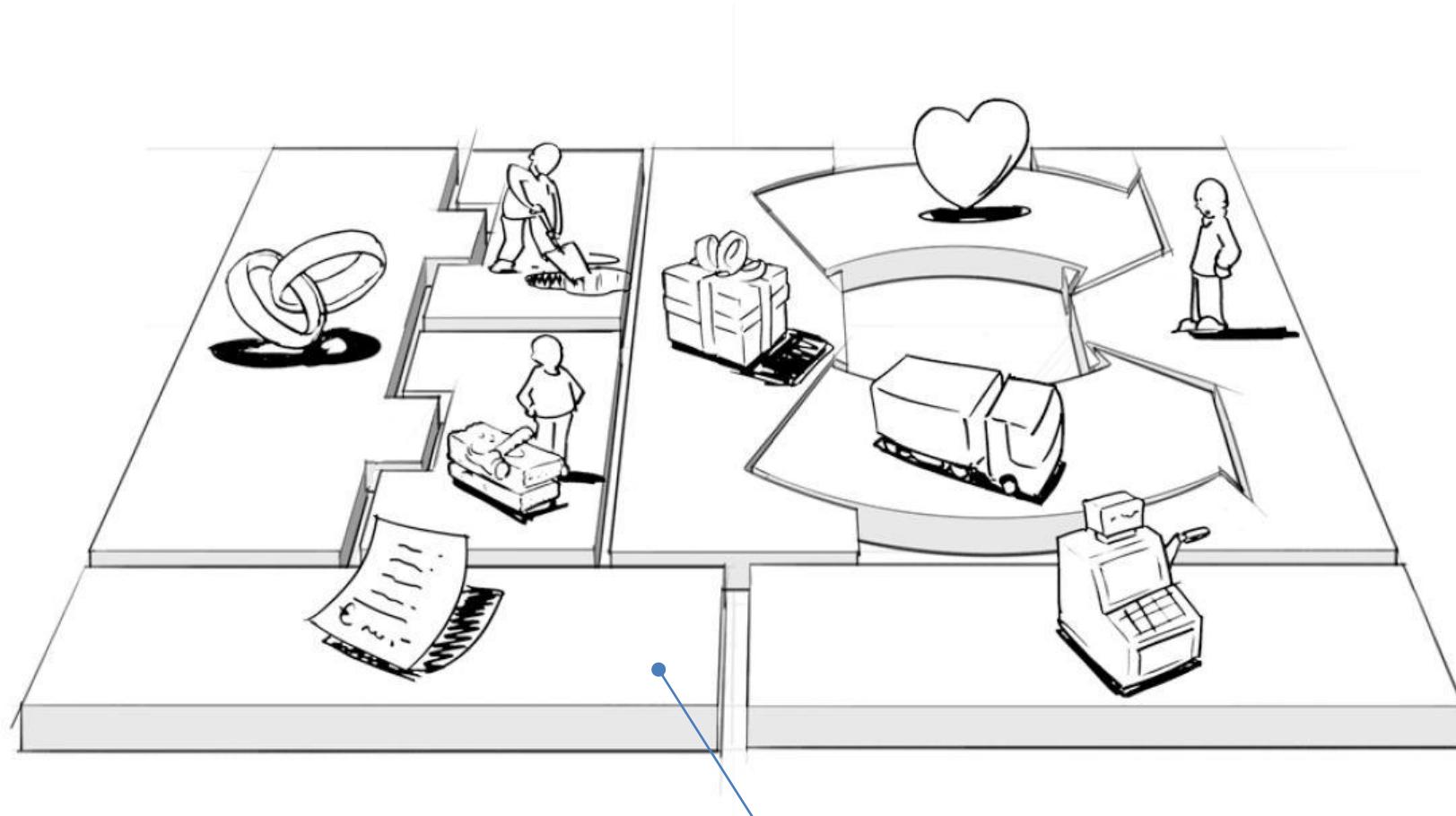
# Key Partners



which partners and suppliers leverage your model?  
who do you need to rely on?

drawings by JAM

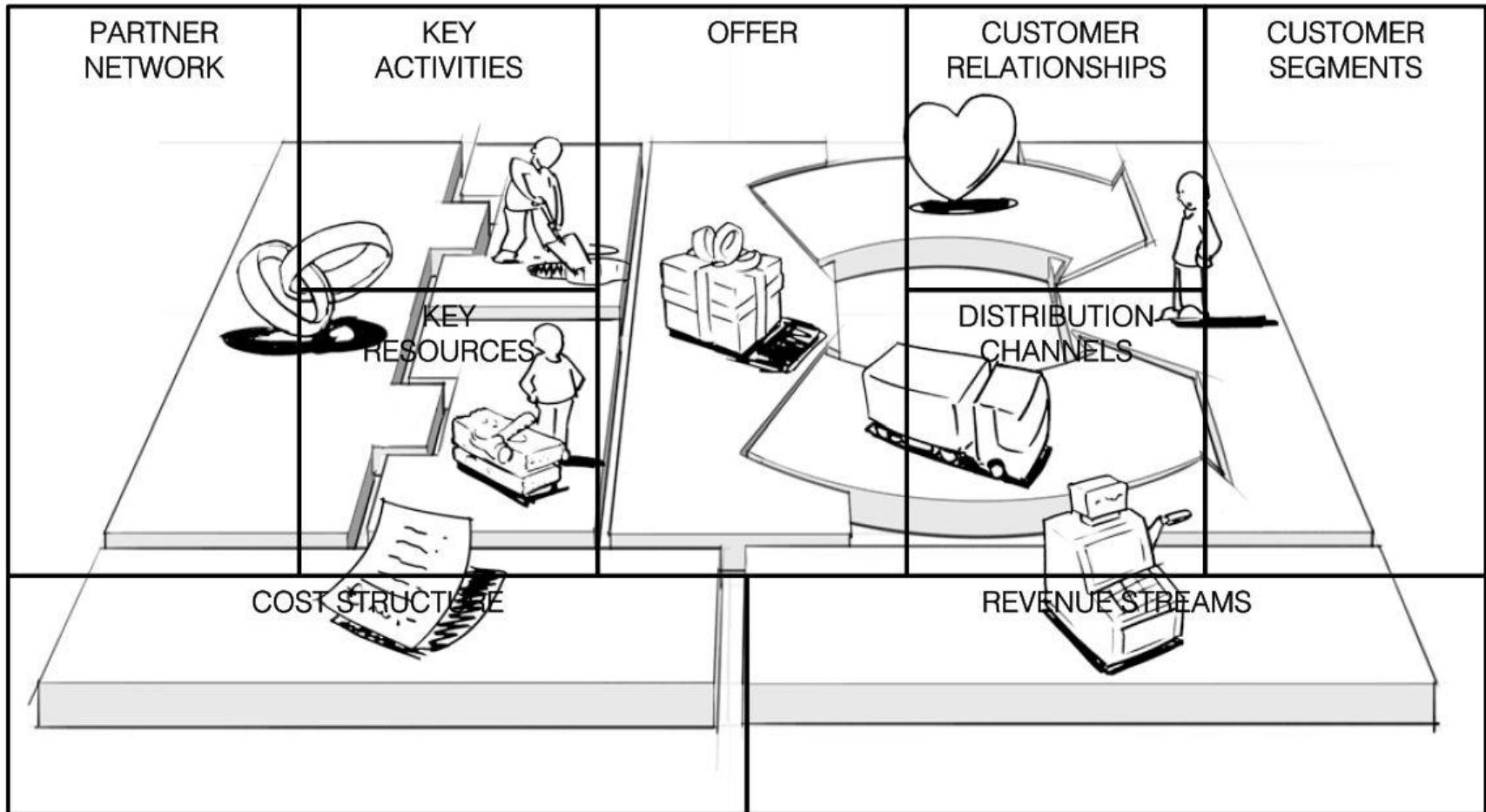
# Cost Structure



what is the resulting cost structure?  
which key elements drive your costs?

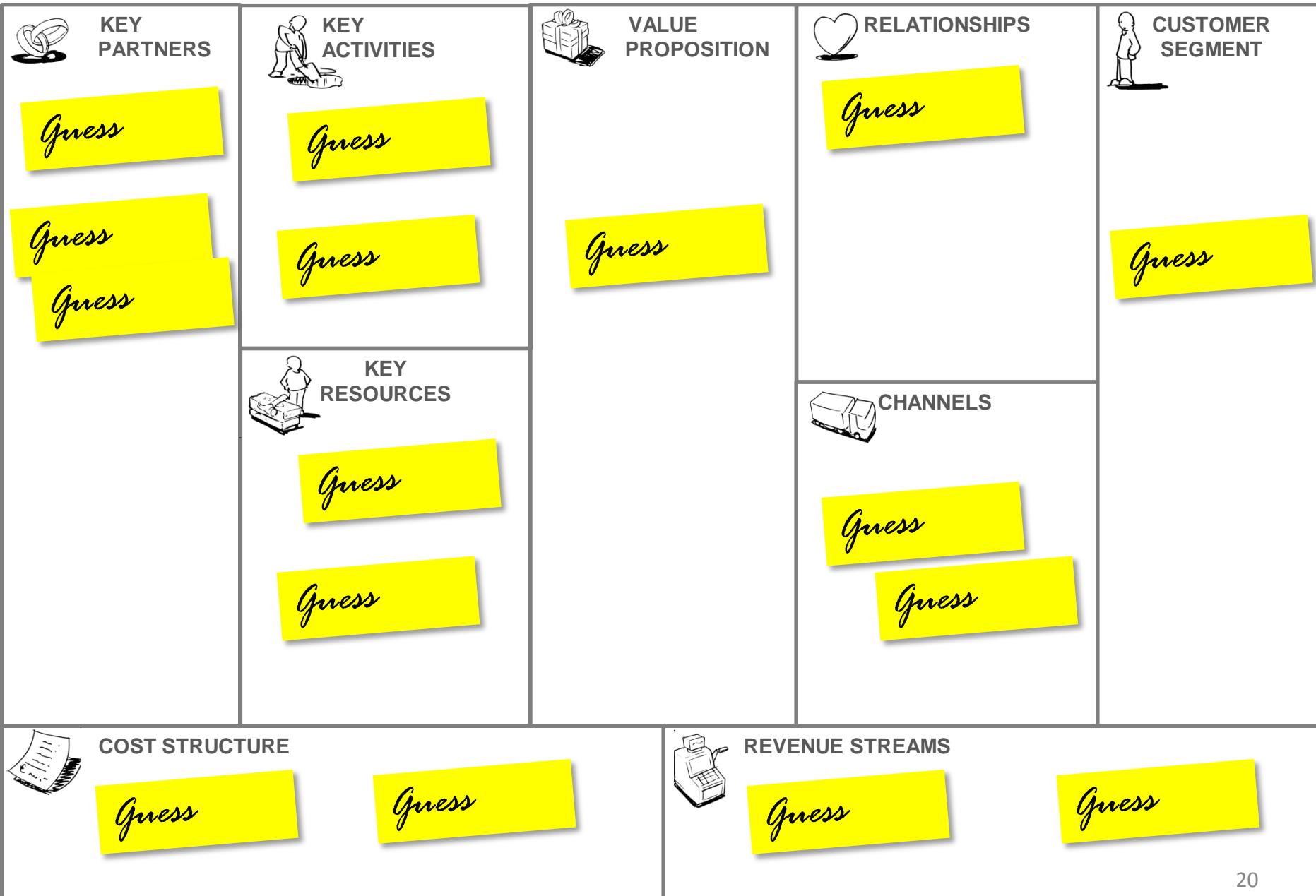
drawings by JAM

# Business Model Canvas



drawings by JAM

# Your Business Model Canvas



**EXAMPLES**  
**EXAMPLES**

# Example 1



Refreshing lemonade to joggers  
at public parks

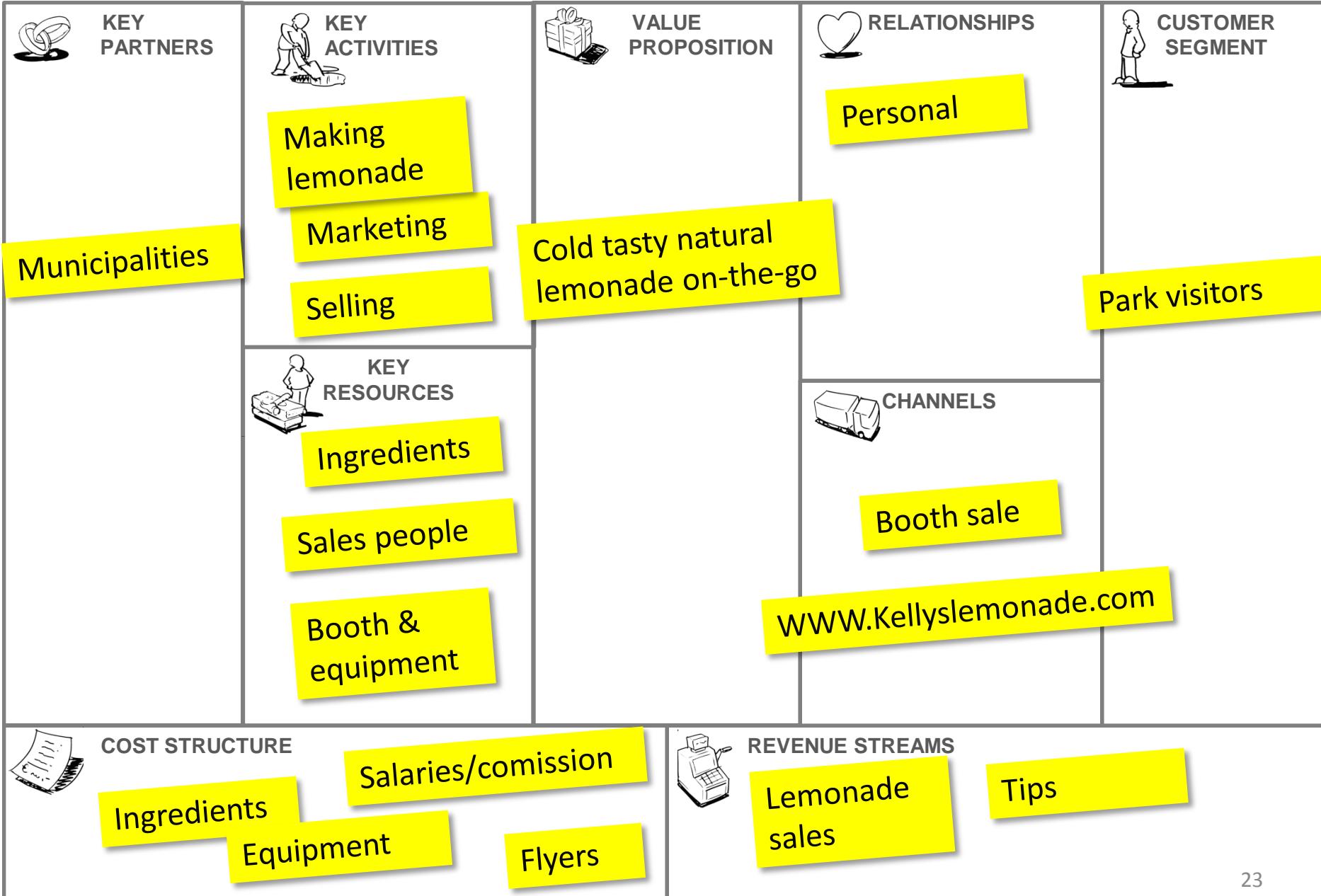
# Example 2



Affordable VOIP calls

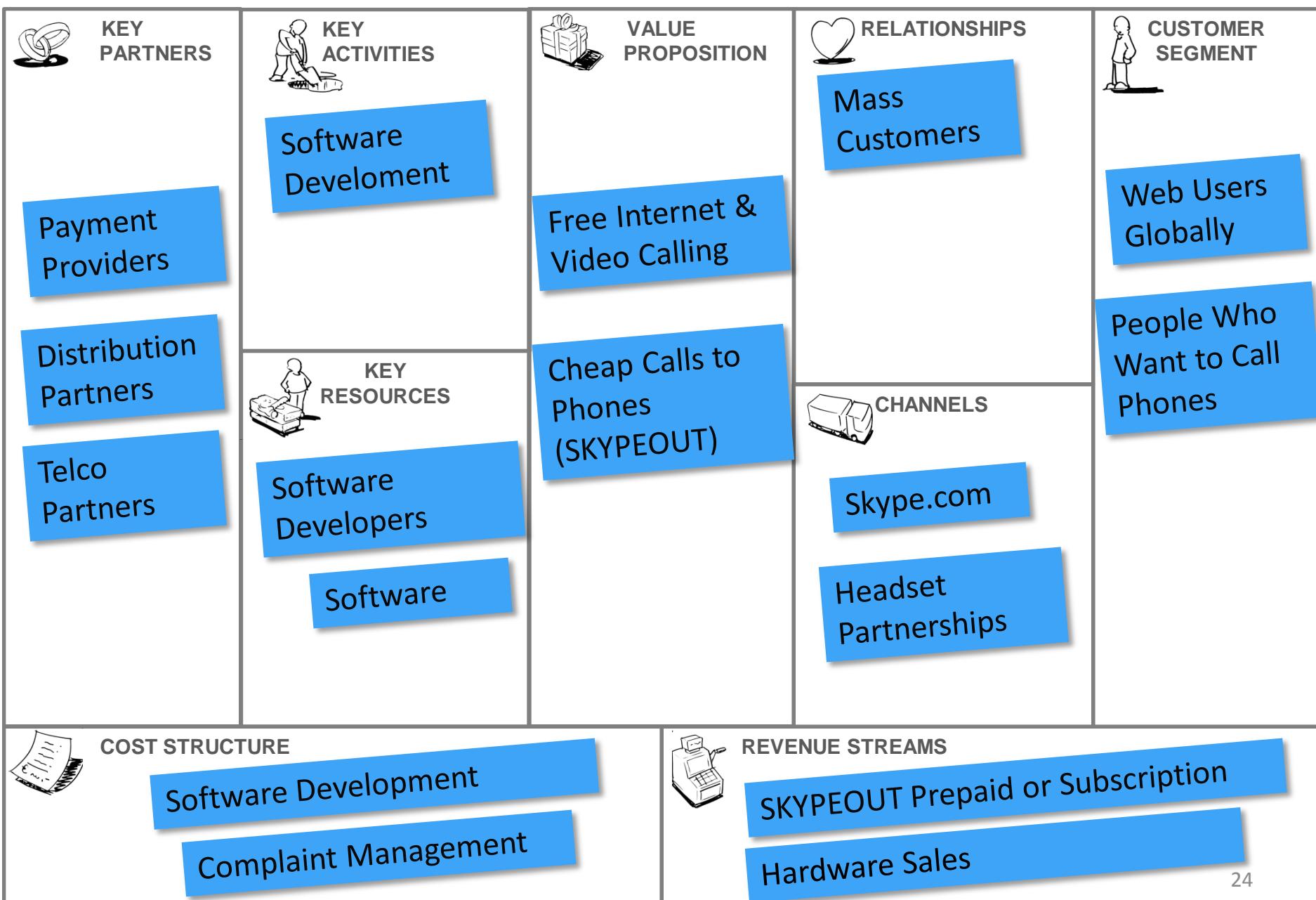


# Kelly's Lemonade Stand: Refreshing Lemonade





# Skype



## Example 3



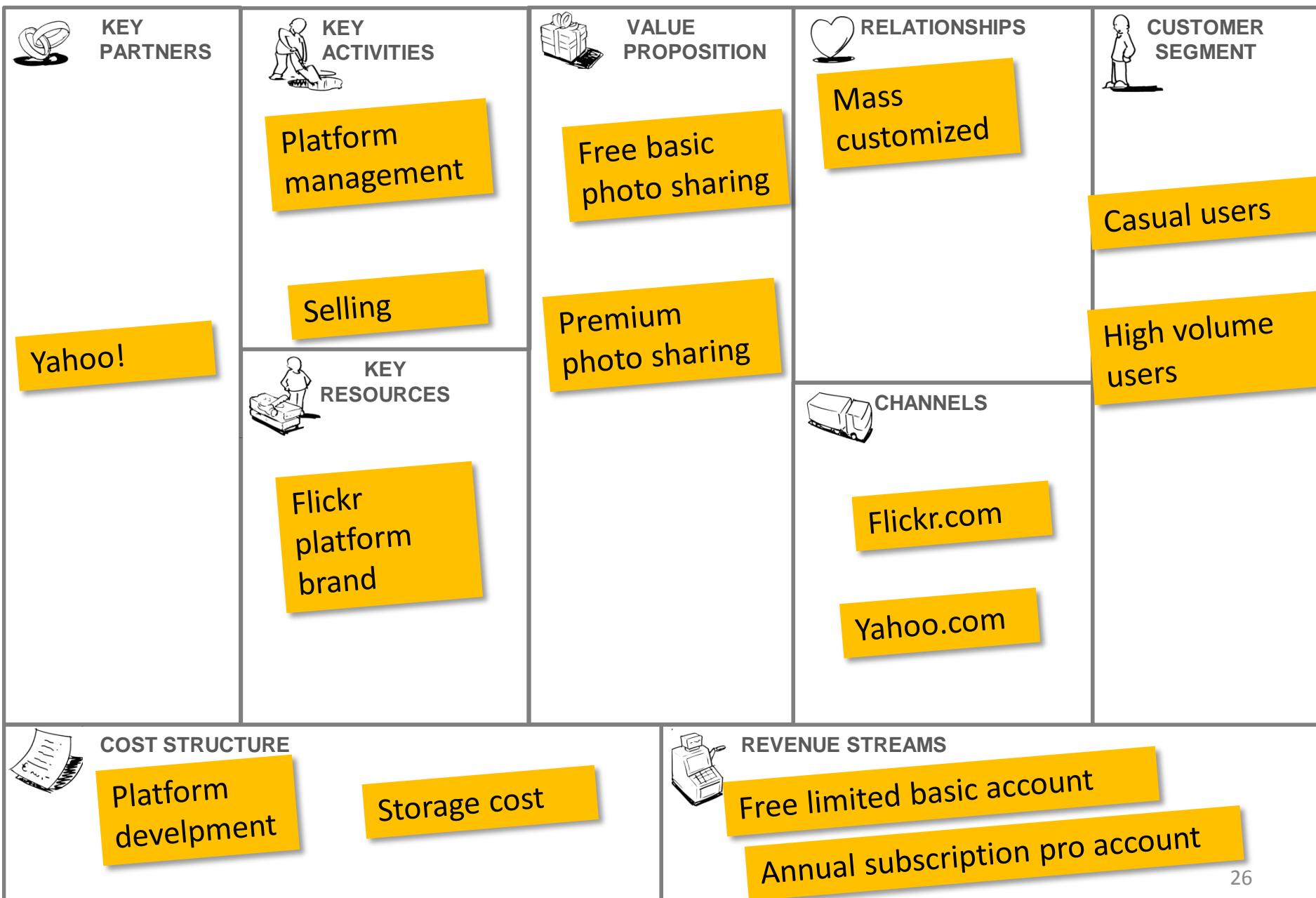
Photo sharing online

## Example 4

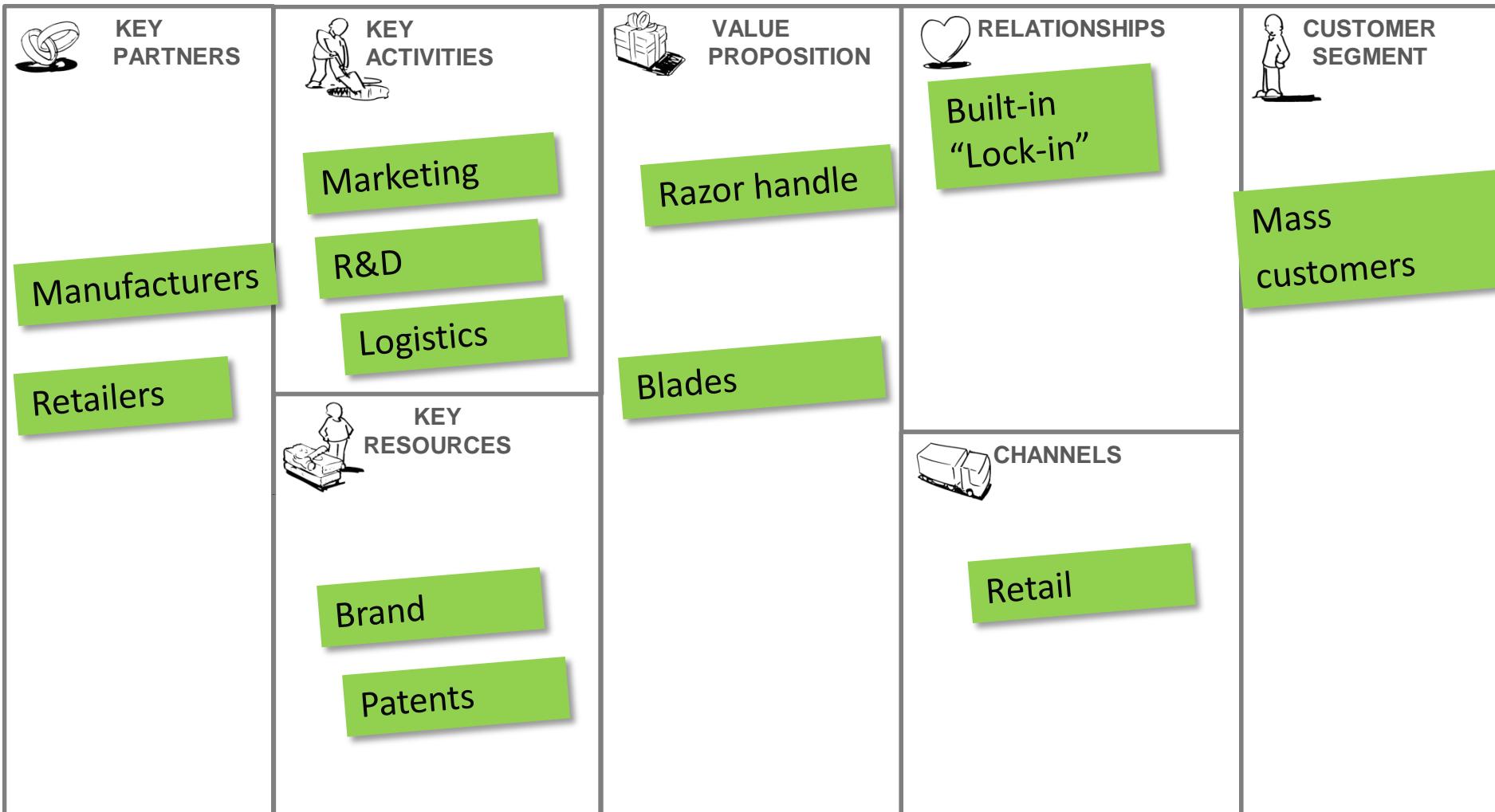


Smooth shave for men & women

# Flickr: Photo Sharing



# Gillette: Razors & Blades





Break...

# GROUP WORKSHOP

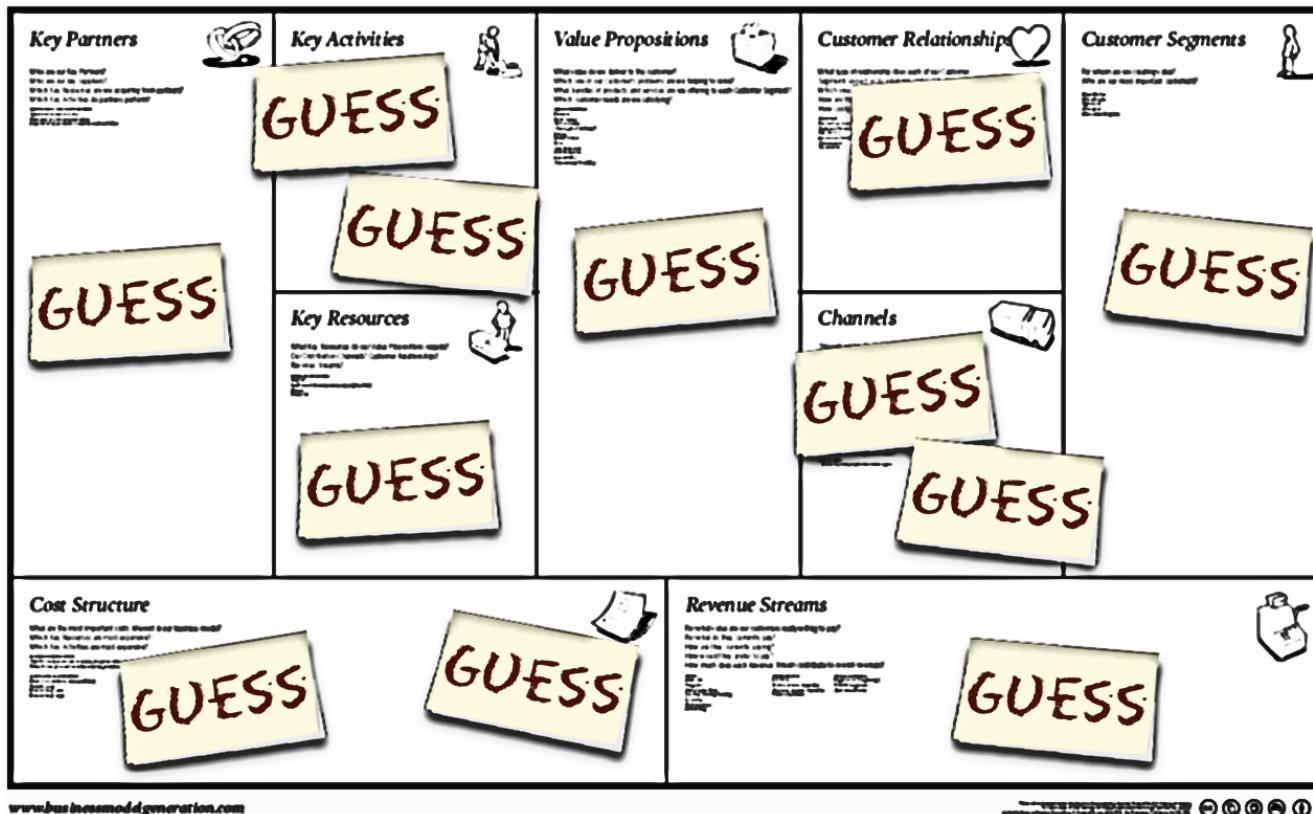


# GROUP WORKSHOP

1. Assemble in teams
2. Create canvas
3. Write key words on sticky notes
4. Place sticky notes on the canvas
5. Present your canvas

# Create a CANVAS

## of your enterprise project





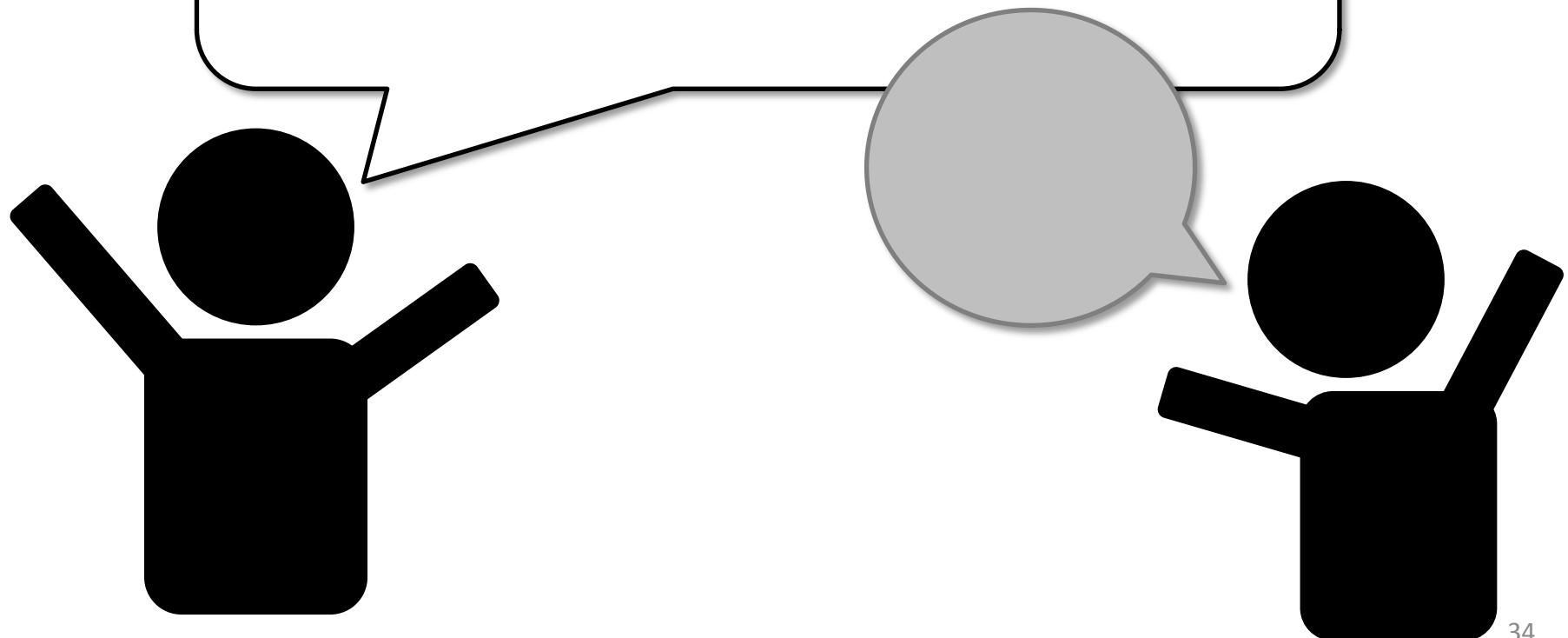
Break...

1 minute

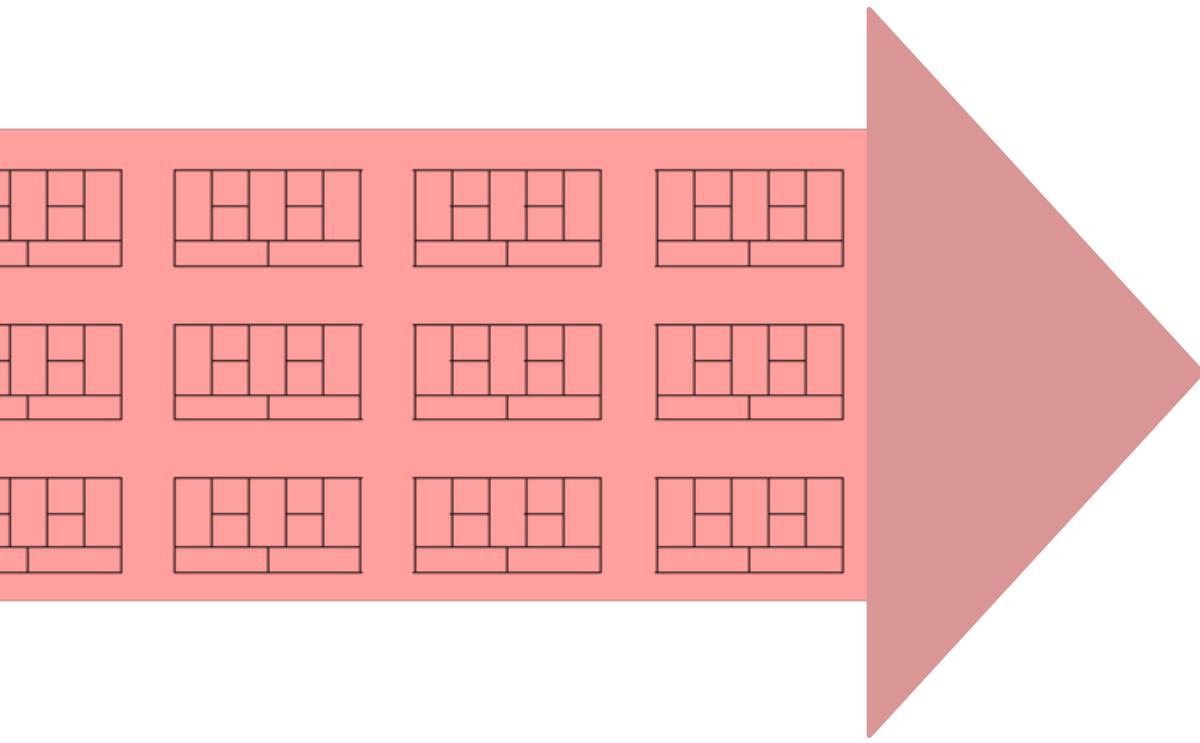
# *Presentations*



# Discussion



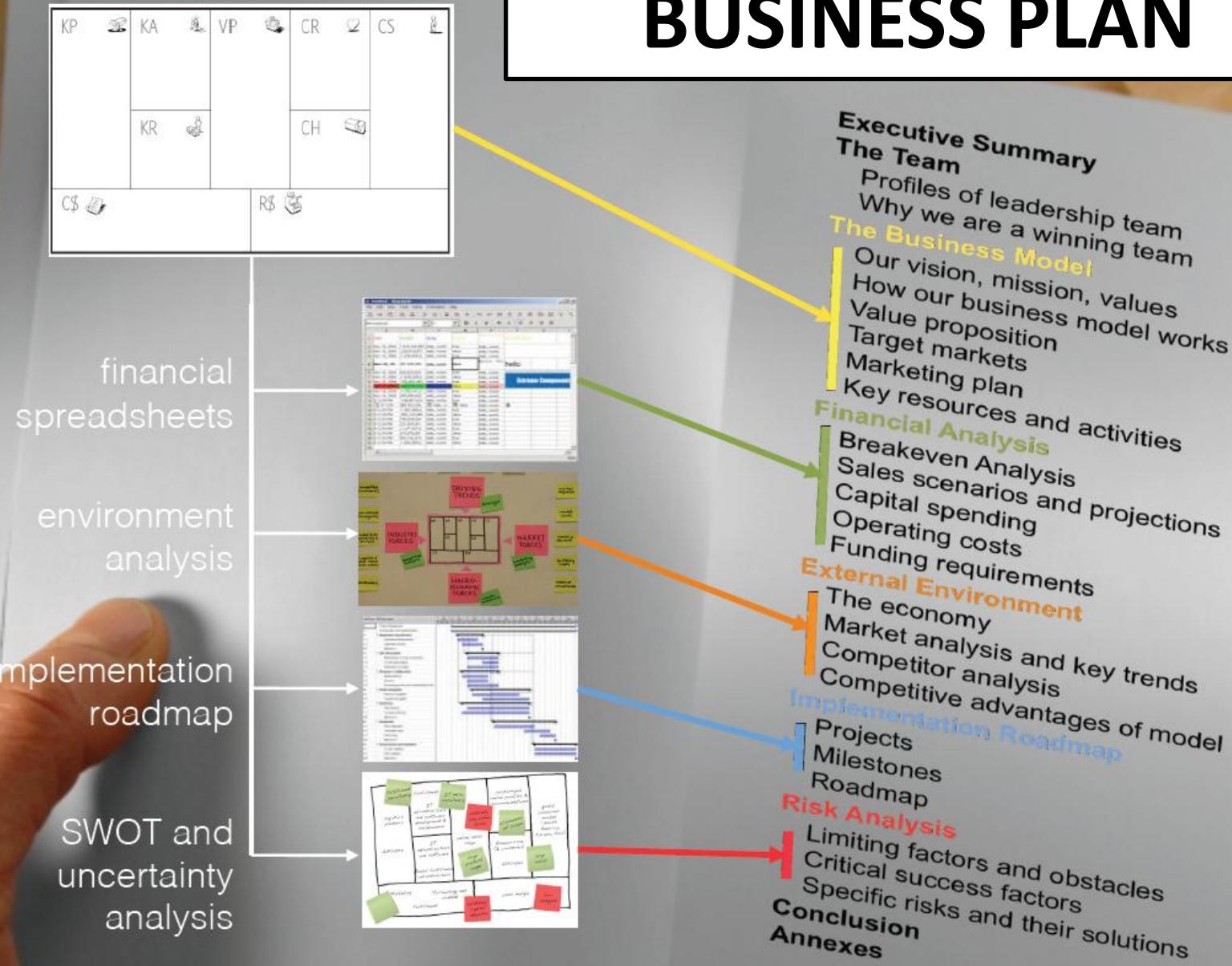
# so what's beyond the **CANVAS?**



A photograph showing two people in business attire shaking hands over a desk. On the desk, there are several papers, including one titled "Financial Plan of Company Development" with a bar chart. A pen is also visible on the desk.

you need to validate your model assumptions with the customers until you get it right!

# BUSINESS PLAN



You're holding a handbook for visionaries, game changers, and challengers striving to defy outmoded business models and design tomorrow's enterprises. It's a book for the...

# Business Model Generation

## WRITTEN BY

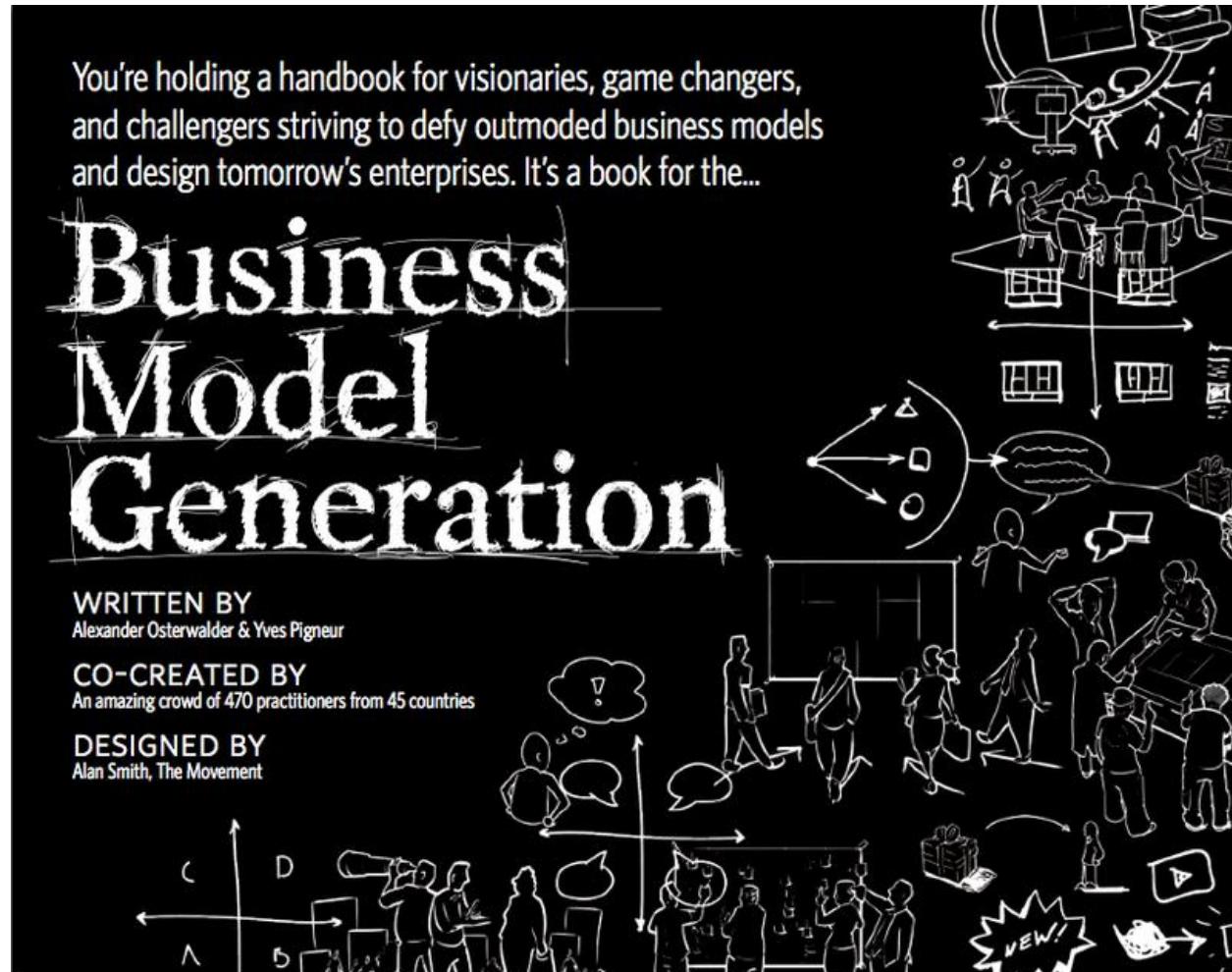
Alexander Osterwalder & Yves Pigneur

## CO-CREATED BY

An amazing crowd of 470 practitioners from 45 countries

## DESIGNED BY

Alan Smith, The Movement



[www.BusinessModelGeneration.com](http://www.BusinessModelGeneration.com)

# THANK YOU!



Emad Saif  
[www.emadsaif.com](http://www.emadsaif.com)

