Cloud Computing: Lecture 2

Cloud Computing concept and architecture

Overview

What is Cloud Computing?

Main concept underlying the cloud idea

Main advantages and disadvantages

Cloud architecture

Cloud Computing

There are several definitions of CC

The concrete example of Utility Computing

The natural evolution of Grid Computing

NIST (National Institute of Standards and Technology) provides a “standard” definition

https://doi.org/10.6028/NIST.SP.800-145

Cloud Computing - NIST

*"Cloud computing is a model for enabling convenient, **on-demand network access** to a **shared pool** of **configurable** computing **resources** (e.g., networks, servers, storage, applications, and services) that can be rapidly **provisioned** and released with **minimal management** effort or service provider interaction. "*

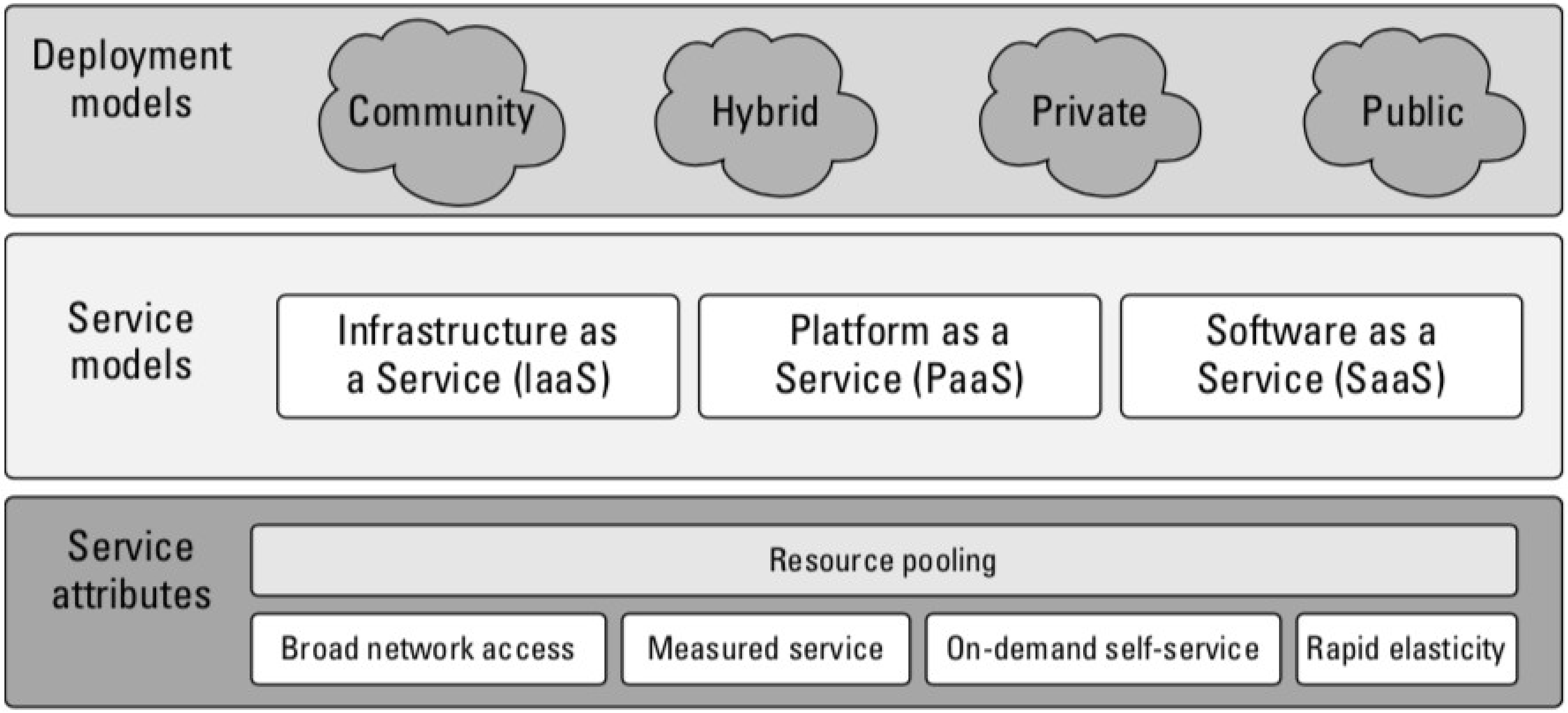
*"This cloud model promotes **availability** and is composed of **five essential characteristics**, three **service** models, and four **deployment models**."*

5 Attributes: On-demand self-service, Broad network access, Resource pooling, Rapid elasticity, Measured service

3 Service Models: Infrastructure as a Service, Platform as a Service, Software as a Service

4 Deployment Models: Public Cloud, Private Cloud, Community Cloud, Hybrid Cloud

NIST model view



Cloud Attributes

On-demand self-service

Broad network access

Resource pooling

Rapid elasticity

Measured service

On-demand self-service

“a consumer can unilaterally provision computing capabilities, as needed

without human interaction with the service provider”

computing capabilities are: server time, network storage, number of servers etc.

Broad network access

capabilities are
- available over the network
- accessed through standard mechanisms

promote use by heterogeneous thin or thick client platforms
- (e.g. laptop, Desktop, mobile devices etc.)

Resource Pooling -- Multi-tenancy

Computing resources are storage, processing, memory, network bandwidth and virtual machines

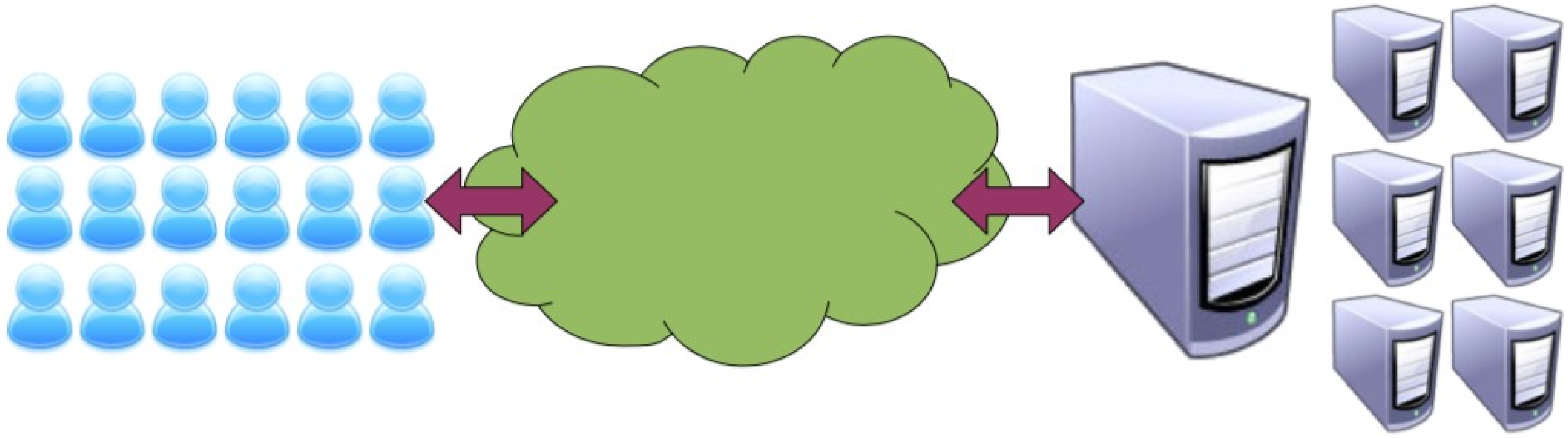
Provider’s computing resources are pooled to serve multiple consumers, allocated and deallocated as needed.

Location independence: there is no control over the exact location of the resources. This has major implications performance, scalability, security.

Rapid Elasticity

capabilities can be rapidly and elastically provisioned
we can add resources by scale up or scale out systems

virtually unlimited resources



Measured service

Metering capability of service/resource abstractions in terms of storage, processing, bandwidth, active user accounts etc.

Remember the utility computing and pay as you go model.

(more on this later when we discuss deployment models)

Additional Advantages (attributes)

- Lower costs
- Ease of utilization
- Quality of Service (QoS)
- Reliability
- Outsourced IT management
- Simplified maintenance and upgrade
- Low Barrier to Entry

Disadvantages of cloud approach

- Do I need CC if my organization is large enough to support IT solutions?
- May I customize my application as much as I want?
- Intrinsic WLAN latency is introduced (everything works on LAN, LAN fault means no access to cloud)
- Cloud Computing is a stateless system, you need additional tools as service brokers, transaction manager, and middleware
- Privacy and Security

Cloud Service Models

Infrastructure as a Service

Platform as a Service

Software as a Service

Network as a Service

Cloud Service Models

Infrastructure as a Service: IaaS provides virtual machines, virtual storage, virtual infrastructure, and other hardware assets as resources that clients can provision.

Platform as a Service: PaaS provides virtual machines, operating systems, applications, services, development frameworks, transactions, and control structures.

Software as a Service: SaaS is a complete operating environment with applications, management, and the user interface.

NaaS as a Service: NaaS is the ability to access to visìrtualized network components and compose them to build a network

Service Models: The SPI model

XaaS, or "<Some*thing*> as a Service"

Infrastructure as a Service

Platform as a Service

Software as a Service

Network as a Service

Desktop as a Service

Storage as a Service

etc …...

Practical examples of XaaS

- IaaS
  - Amazon Elastic Computing
  - Google Cloud
  - OpenStack
  - OpenNebula
- PaaS
  - Google AppEngine
  - OpenShift
  - Windows Azure
- SaaS
  - GoogleApps (Documents)
  - SQL Azure
  - Oracle on demand

Cloud Deployment Models

Public Cloud

Private Cloud

Community Cloud

Hybrid Cloud

Deployment Models

Public cloud: The public cloud infrastructure is available for public use alternatively for a large industry group and is owned by an organization selling cloud services.

Private cloud: The private cloud infrastructure is operated for the exclusive use of an organization. The cloud may be managed by that organization or a third party. Private clouds may be either on- or off-premises.

Community cloud: A community cloud is one where the cloud has been organized to serve a common function or purpose.

Hybrid cloud: A hybrid cloud combines multiple clouds (private, community of public) where those clouds retain their unique identities, but are bound together as a unit.

Cloud Computing base concepts

“**Cloud**” makes reference to two **essential concepts**:

Abstraction: Cloud computing abstracts the details of system implementation from users and developers. Applications run on physical systems that aren’t specified, data is stored in locations that are unknown, administration of systems is outsourced to others, and access by users is ubiquitous.

Virtualization: Cloud computing virtualizes systems by pooling and sharing resources. Systems and storage can be provisioned as needed from a centralized infrastructure, costs are assessed on a metered basis, multi-tenancy is enabled, and resources are scalable with agility.

How Can I build a Cloud?

Composability

Infrastructures - resources used as lego

Platforms - built composing different softwares

Virtual Appliance - example VM marketplace

Communication protocols – standards

Clients – different standards implementation

New way to build applications

Composability

A composable system uses components to assemble services that can be tailored for a specific purpose using standard part.

Modular: It is a self-contained and independent unit that is cooperative, reusable, and replaceable.

Stateless: A transaction is executed without regard to other transactions or requests.

CC Architecture: Composability

Composability diminishes going up the cloud computing stack is from the user’s point of view.

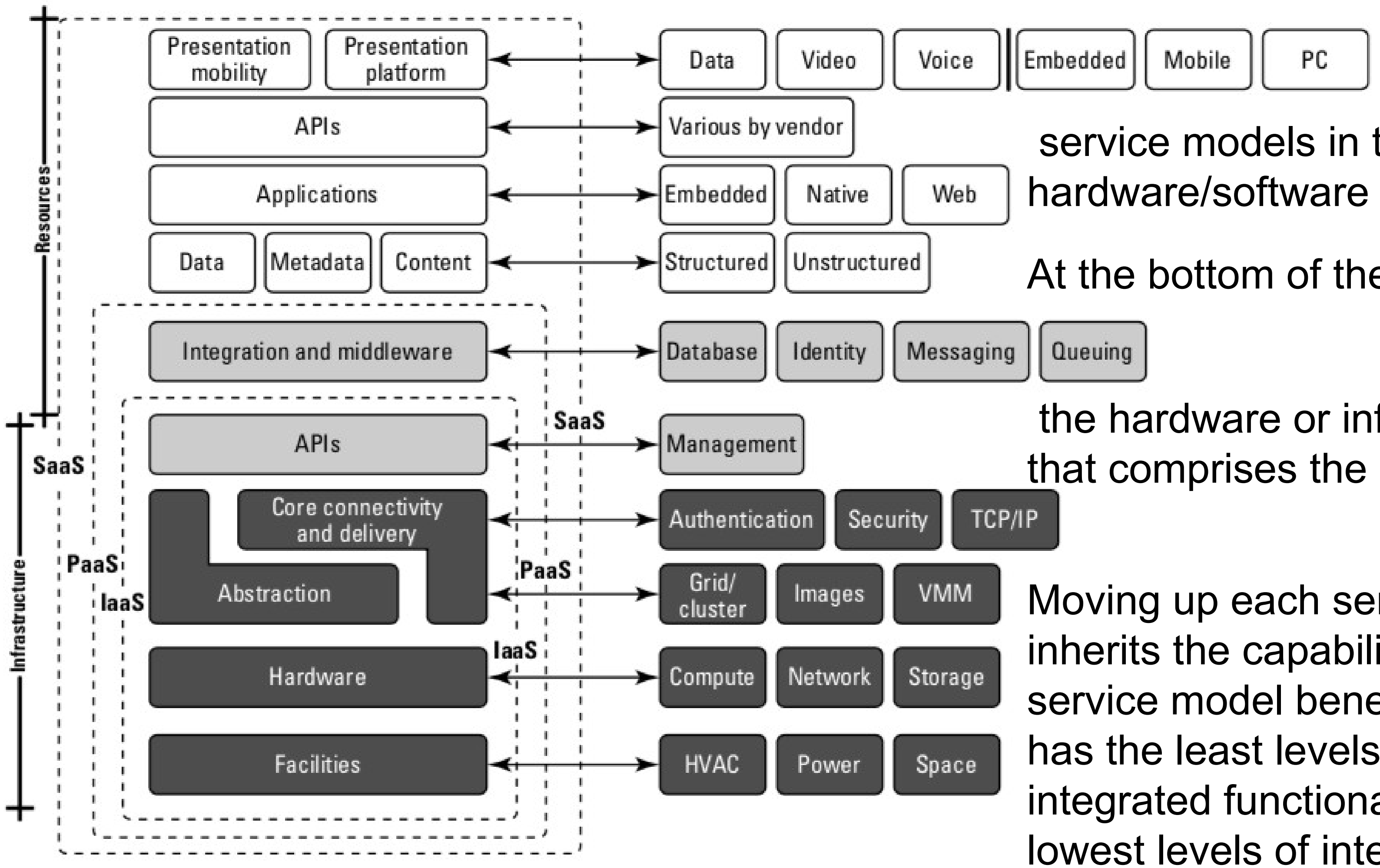
Composability still remain strong from service providers point of view.

Service Oriented Architecture (SOA)

APIs

Open Standards

Cloud Reference Model



Watching the computing's service models in terms of a hardware/software stack:

At the bottom of the stack is the hardware or infrastructure that comprises the network

Moving up each service model inherits the capabilities of the service model beneath it. IaaS has the least levels of integrated functionality and the lowest levels of integration, and SaaS has the most.

From: "Cloud Computing Bible" by Barrie Sosinsky

Virtual Appliances

Applications such as a Web server or database server that can run on a virtual machine image are referred to as virtual appliances.

Virtual appliances are software installed on virtual servers.

Virtual appliances are basis for assembling more complex services, the appliance being one of your standardized components

(see IaaS lecture for example of VA)

Communication protocols

Cloud computing arises from services available over the Internet communicating using the standard Internet protocol suite underpinned by the **HTTP and HTTPS** transfer protocols.

- **RPC → XML-RPC** protocol used to execute Remote Procedure Call (RPC) through Internet. This protocol uses the XML standard to code the request transferred through HTTP or HTTPS protocol
- **SOAP** (Simple Object Access Protocol) protocol to exchange messages between software components
- **WSDL** (Web Service Description Language) XML language used to create “documents” to describe Web Services
- **REST** (Representational State Transfer) software architecture for distributed systems
- **AtomPub** (Atom publishing protocol) is an application-level protocol for publishing and editing Web resources

Role of Open Standards

Integration, portability, interoperability and innovation.

Effort to create open standards: clients do not want to be locked into any single system, there is a strong industry push to create standards-based clouds.

Based on SoA and REST (we will discuss this later in this course)

Some de facto standards: Amazon S3 and EC2

IEEE Technical Committee on Services Computing

Open Standards & Cloud Implementations

Open Cloud Computing Interface (OCCI)

open community-lead specifications delivered through the Open Grid Forum.

Python, REST, PHP

Cloud Data Management Interface (CDMI)

create, retrieve, update and delete data elements from the Cloud

Python, REST APIs

Cloud Stack Implementation examples:

OpenStack, OpenNebula, Eucalyptus

EUROPEAN Open Science Cloud

Cloud building blocks summary

Abstraction

Virtualization

Modularity

Composability

Stateless

SOA (Service Oriented Architecture)

Communications protocols

Standards

Five minutes of REST

REST stands for Representational State Transfer, it is an architectural style designed for distributed hypermedia

REST is a certain approach to creating Web Services.

Each REST command is centered around a resource. In REST, a resource is anything that can be pointed to via HTTP protocol.

4 basic HTTP verbs we use in requests in a REST system:

- GET — retrieve a specific resource (by id) or a collection of resources
- POST — create a new resource
- PUT — update a specific resource (by id)
- DELETE — remove a specific resource by id

Five minutes of REST

CRUD operations:

- Create
- Read
- Update
- Delete

One might think that REST is no way related to CRUD but it is. There are certain operations involved in REST and it is based on HTTP protocol. It uses GET method to read data, PUT method to modify data, POST method to create data and DELETE method to remove data. The comparison arises when one has to choose between the operation sets both of them provide. One can either choose GET, PUT, POST, DELETE or CREATE, READ, UPDATE, DELETE.

Read more at:

https://www.freelancinggig.com/blog/2018/04/24/rest-vs-crud-need-know/

Five minutes of REST

"Representational State Transfer is intended to evoke an image of how a well-designed Web application behaves: a network of web pages (a virtual state-machine), where the user progresses through an application by selecting links (state transitions), resulting in the next page (representing the next state of the application) being transferred to the user and rendered for their use."

Roy Fielding.

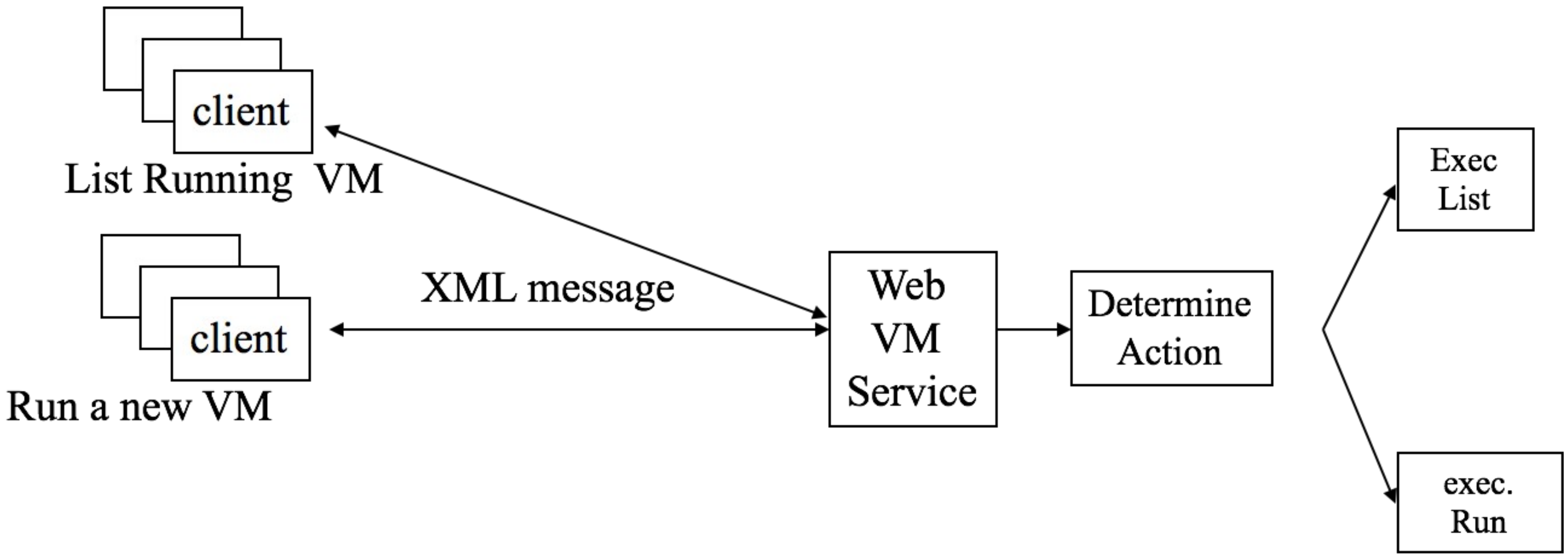
Five minutes of REST

To understand the REST design pattern/architecture, let's look at an example

(learn by example).

Five Minutes of REST

I want a service to simply execute VMs and list running VMs

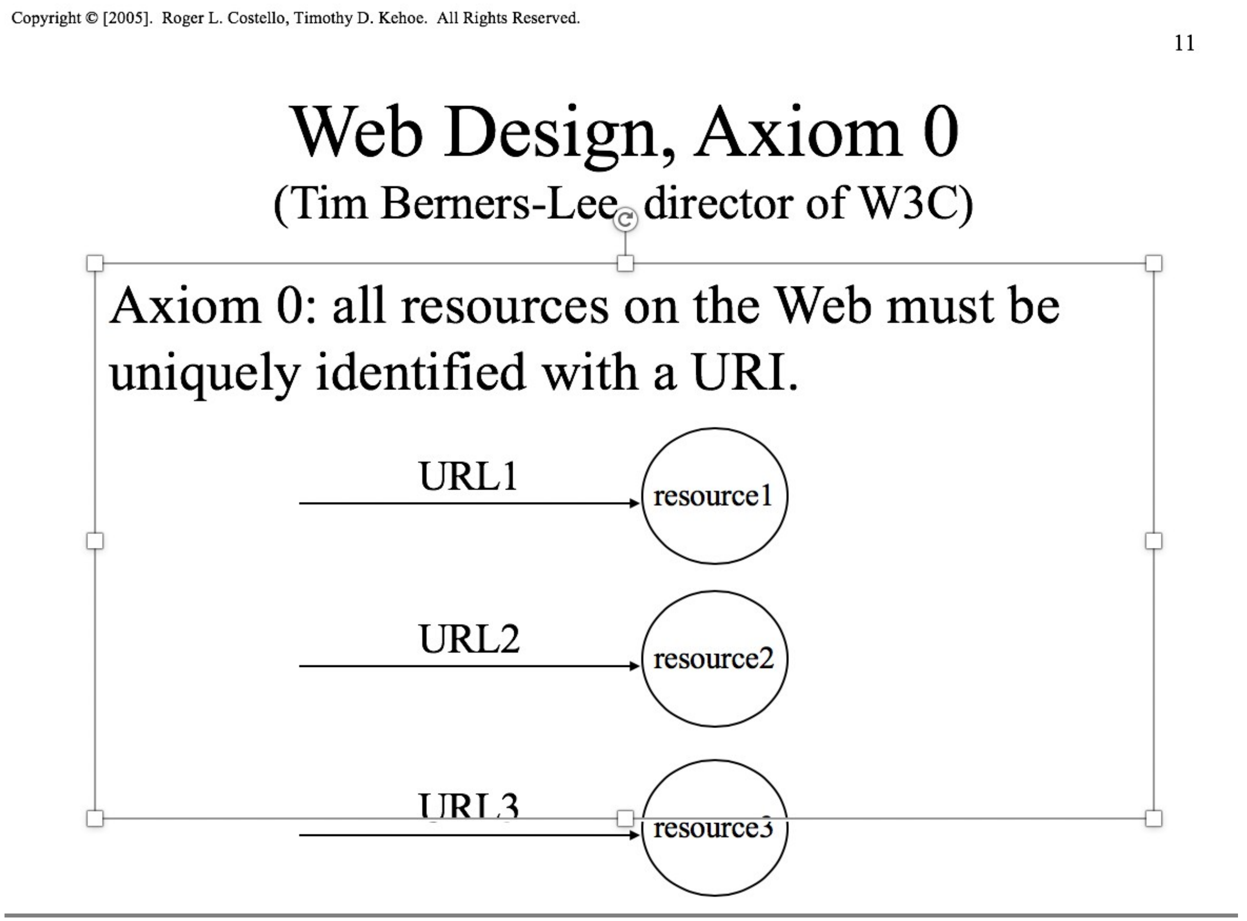


connect to the same URL

Five Minutes of REST

This approach is based upon the incorrect assumption that a URL is "expensive" and that their use must be rationed.

It violates Tim Berners-Lee Web Design, Axiom 0



Copyright © [2005]. Roger L. Costello, Timothy D. Kehoe. All Rights Reserved.

11

Web Design, Axiom 0

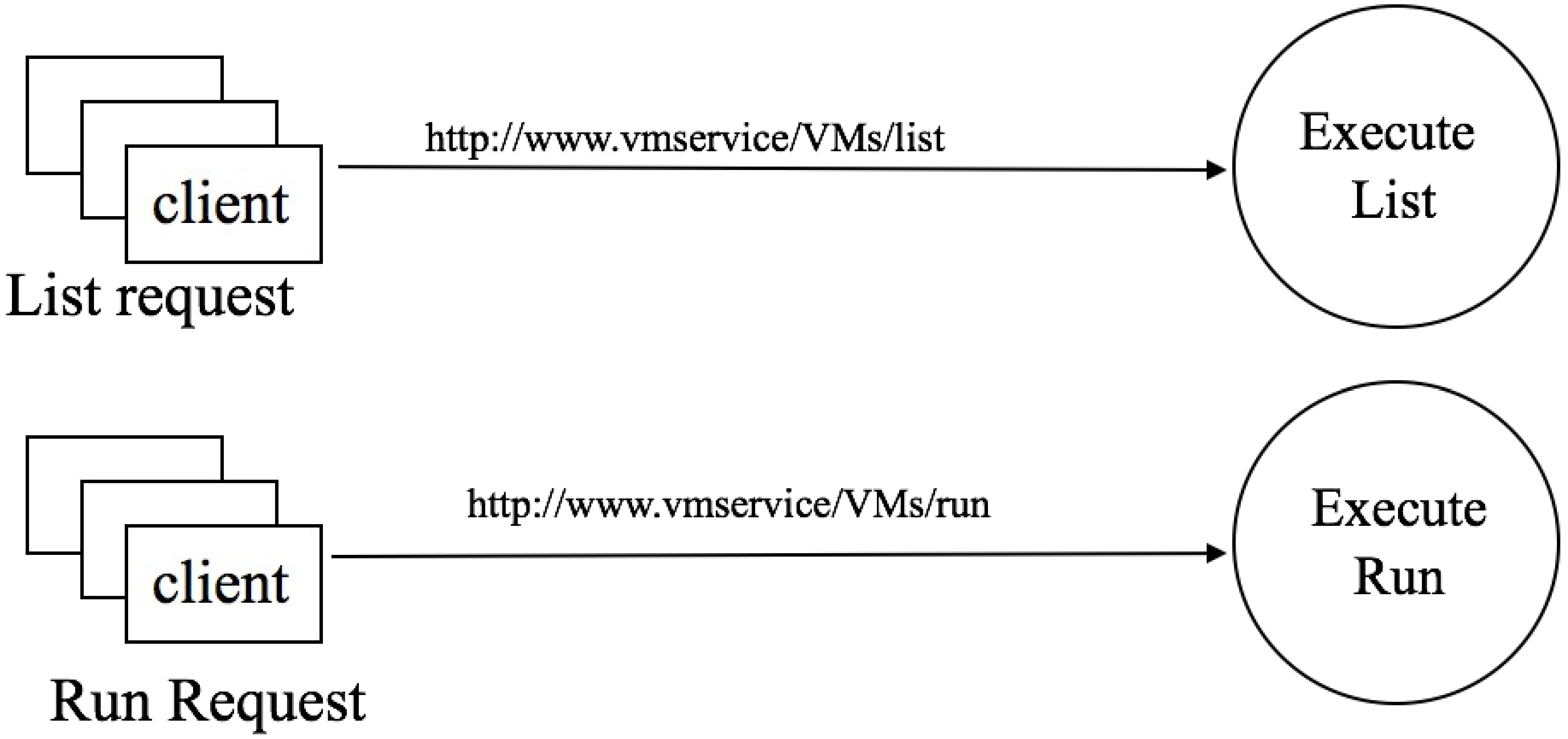
(Tim Berners-Lee, director of W3C)

Axiom 0: all resources on the Web must be uniquely identified with a URI.

URL1
resource1
URL2
resource2
URL3
resource3

Five minutes of REST

Using more than one URL (Like in REST)



connect to different URLs

Five Minutes of REST

The different URLs are discoverable by search engines and UDDI (Universal Description Discovery and Integration) registries.

It's easy to understand what each service does simply by examining the URL, i.e., it exploits the Principle of Least Surprise.

Consistent with Axiom 0.

Principle of Least Surprise or Principle of Least Astonishment (POLA): applies to user interface and software design. Typical formulation (1984): "If a necessary feature has a high astonishment factor, it may be necessary to redesign the feature."

More generally, the principle means that a component of a system should behave in a way that most users will expect it to behave; the behavior should not astonish or surprise users.

Five minutes of REST

In REST Design Pattern

Resources: Every distinguishable entity is a resource. A resource may be a Web site, an HTML page, an XML document, a Web service, a physical device, etc.

URLs Identify Resources: Every resource is uniquely identified by a URL. This is Tim Berners-Lee Web Design, Axiom 0.

Five Minutes of REST: In Summary

Client-Server: a pull-based interaction style (Client request data from servers as and when needed).

Stateless: each request from client to server must contain all the information necessary to understand the request, and cannot take advantage of any stored context on the server.

Cache: to improve network efficiency, responses must be capable of being labeled as cacheable or non-cacheable.

Uniform interface: all resources are accessed with a generic interface (e.g., HTTP GET, POST, PUT, DELETE).

Named resources: the system is comprised of resources which are named using a URL.

Interconnected resource representations: the representations of the resources are interconnected using URLs, thereby enabling a client to progress from one state to another.