CLOUD COMPUTING-(18MCA43C)

UNIT – V

'Monitorzing And Management'

FACULTY:

Dr. R. A. Roseline, M.Sc., M.Phil., Ph.D.,
Associate Professor and Head,
Post Graduate and Research Department of Computer Applications,
Government Arts College (Autonomous), Coimbatore – 641 018.

UNIT-IV

- SLA concepts review
- Contrail SLA syntax
- SLA management
- SLA terms
- Multilevel SLA management
- Federation Negotiation
- Provider/SLA selection
- SLA splitting

Federated cloud computing

- A federated cloud (also called cloud federation) is the **deployment and management of multiple external and internal cloud computing services to match business needs.** A federation is the union of several smaller parts that perform a common action**.**

SLA – Service Level Agreement

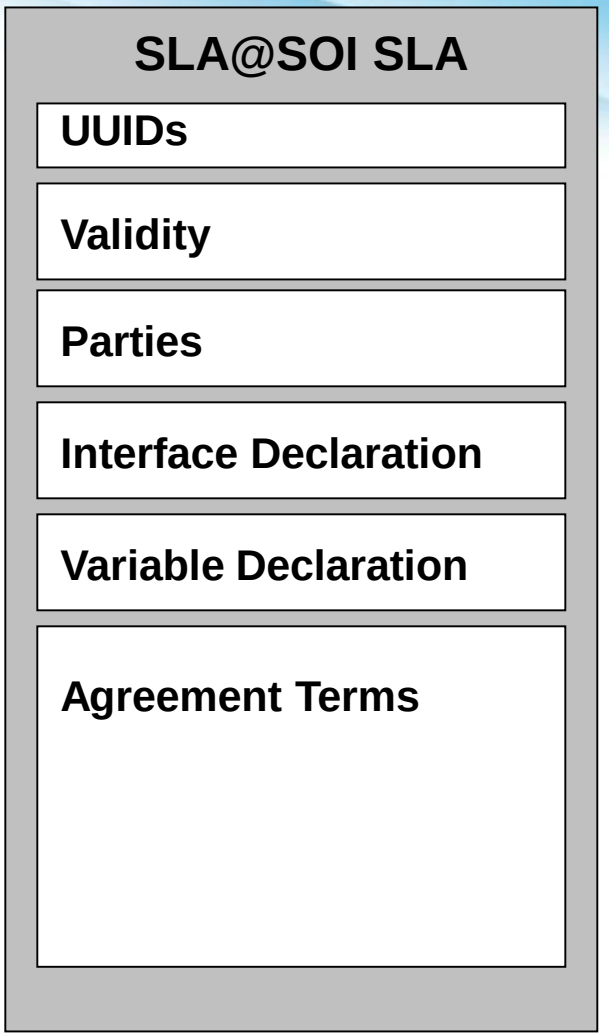
- SLA: agreement (contract) between a Provider and a Consumer of a service
- A SLA is made of one or more SLOs
- SLO (Service Level Objective): it is a condition on a Term
- Term: measure of a given QoS or QoP attribute
- QoS (Quality of Service) attribute: it describes a specific measurable aspect of the quality of a service
  - Examples: Response time, Throughput
- QoP (Quality of Protection) attribute: it describes a specific measurable aspect of the security of a service
  - Examples: Authentication strength, Reputation, Resource location

Definitions of SLA

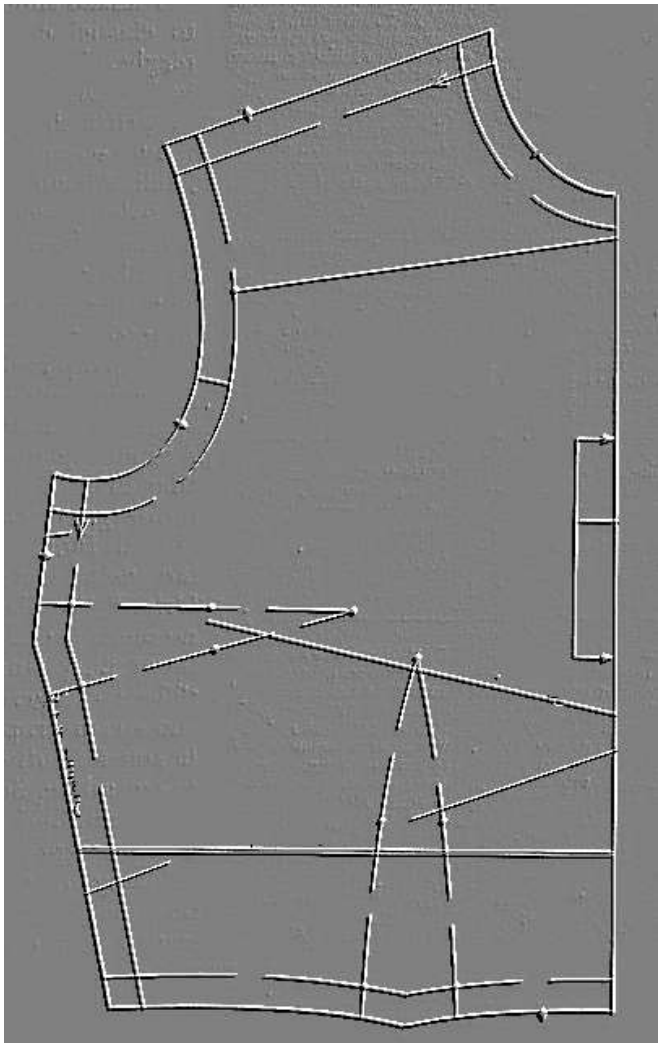
- ITIL (v3), Service Design Book – *[A SLA is] an **agreement** between an IT Service Provider and a Customer. The SLA **describes the IT service**, documents Service Level Targets, and specifies the **responsibilities** of the IT Service Provider and the Customer. A single SLA may cover multiple IT Services or multiple customers.*
- WS-Agreement specification - *An agreement defines a **dynamically-established and dynamically managed** relationship between parties. The object of this relationship is the delivery of a service by one of the parties within the context of the agreement. The management of this delivery is achieved by agreeing on the respective roles, rights and obligations of the parties. The agreement may specify not only **functional properties** for identification or creation of the service, **but also non-functional properties of the service** such as performance or availability. Entities can dynamically establish and manage agreements via Web service interfaces.*
- Contrail, D3.1 – *A SLA is the part of a **contract** between the provider and the consumer of some service where the **service** itself is defined and the **guarantees** offered, usually by the provider, are stated.*

SLA structure

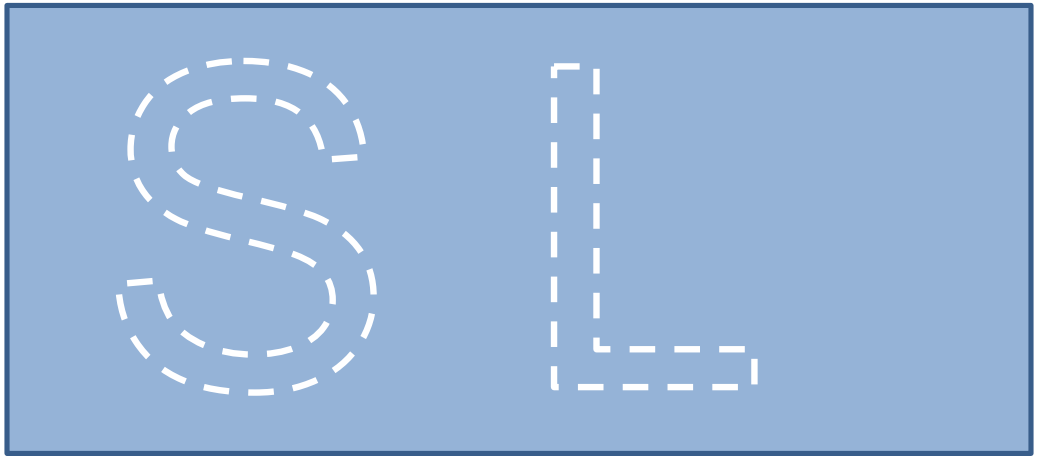
- UUIDs: SLA ID and Template ID
- Validity: when the SLA is effective
- Parties: obliged parties: Provider and Customer
- Interface Declaration: identification of the Services that are object of the agreement
- Agreement Terms: Guarantees offered on the Services, including business attributes (price and penalties)



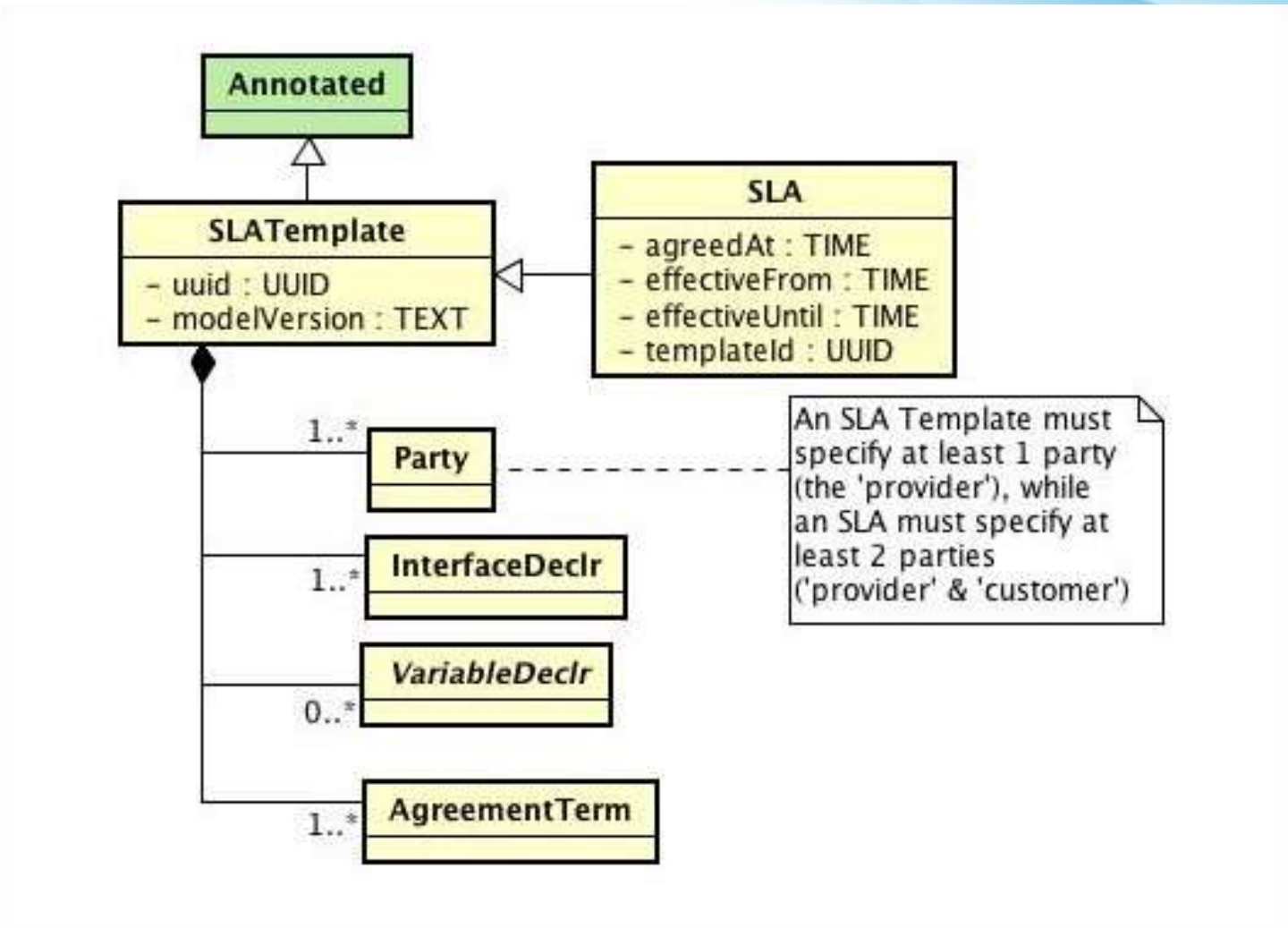
A SLA template is…



- Like a pattern…
- …for a SLA

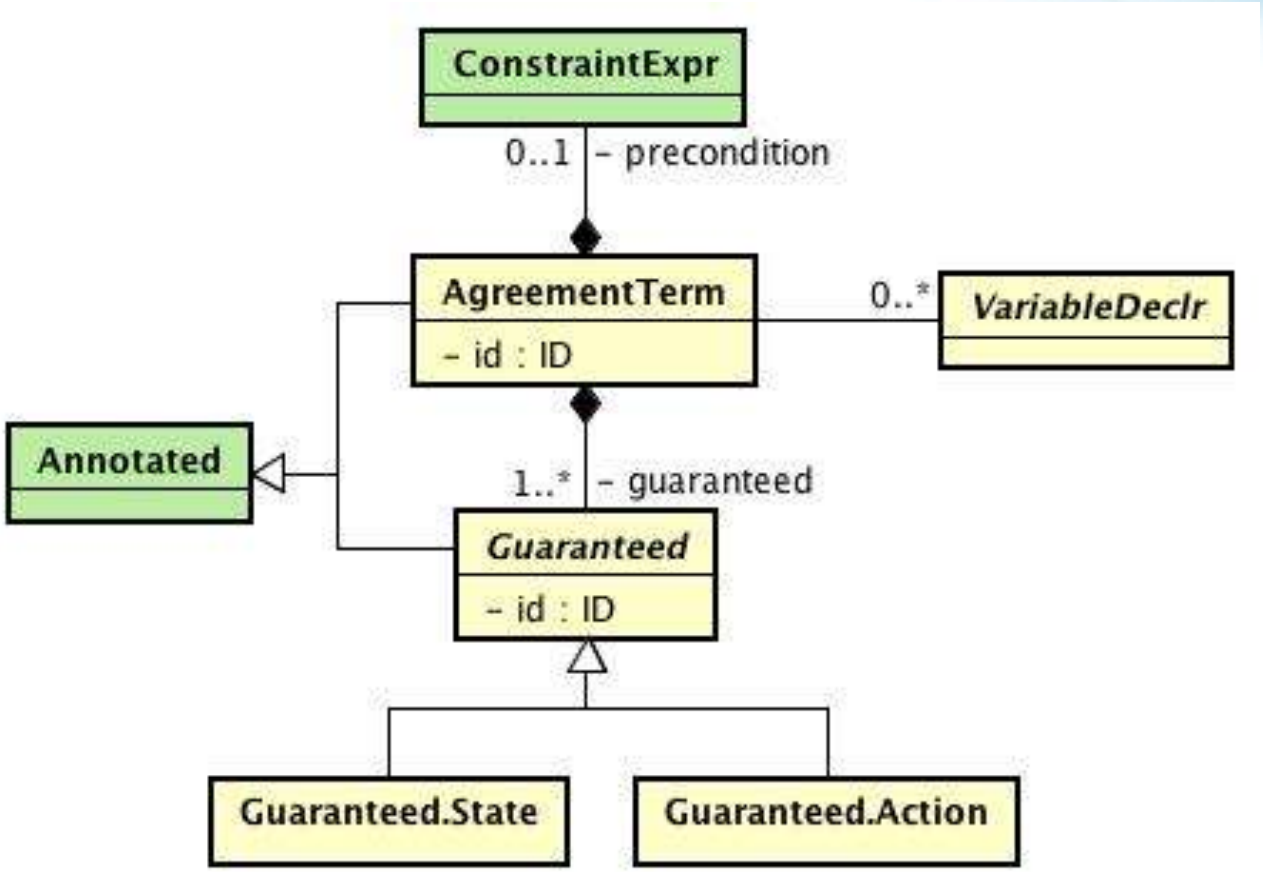


SLAs and SLA Templates (SLA@SOI)



Agreement terms (SLA@SOI)

- D
  -



SLA syntax

BNF syntax of a simple SLA Template

```
sla_template {
    uuid = example_slat_1 // universally unique identifier for this SLAT
    sla_model_version = sla_at_soi_sla_model_v1.0
    party {
        id = ACME
        role = provider
    }
    interface_declr {
        id = the_service // the ID used within the SLAT to refer to
"TheService“
        provider_ref = ACME
        interface_spec { // an "Interface.Specification“
            name = TheService
            operation{ // an "Interface.Operation“
                name = doStuff
            }
        }
    }
    agreement_term {
        id = term_1
        guaranteed_state {
            id = guaranteed_state_1
            qos:completion_time(the_service) < 10s // a
"ConstraintExpr“
        }
    }
```

UUID

Provider party

Interface declarations

Agreement terms

XML syntax of simple SLA Template 1/2

```
<?xml version="1.0" encoding="UTF-8"?>
<slasoi:SLATemplate xmlns:slasoi="http://www.slaatsoi.eu/slamodel"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
   <slasoi:Text/>
   <slasoi:Properties/>

   <slasoi:UUID>example_slat_1</slasoi:UUID>
   <slasoi:ModelVersion>1</slasoi:ModelVersion>

   <slasoi:Party>
      <slasoi:Text/>
      <slasoi:Properties/>
      <slasoi:ID>ACME</slasoi:ID>
      <slasoi:Role>http://www.slaatsoi.org/slamodel#provider</slasoi:Role>
   </slasoi:Party>

   <slasoi:InterfaceDeclr>
      <slasoi:Text/>
      <slasoi:Properties/>
      <slasoi:ID>the_service</slasoi:ID>
      <slasoi:ProviderRef>ACME</slasoi:ProviderRef>
      <slasoi:Interface>
         <slasoi:InterfaceSpec>
            <slasoi:Text/>
            <slasoi:Properties/>
            <slasoi:Name>TheService</slasoi:Name>
            <slasoi:Operation>
               <slasoi:Text/>
               <slasoi:Properties/>
               <slasoi:Name>doStuff</slasoi:Name>
            </slasoi:Operation>
         </slasoi:InterfaceSpec>
      </slasoi:Interface>
   </slasoi:InterfaceDeclr>
```

UUID

Provider party

Interface declarations

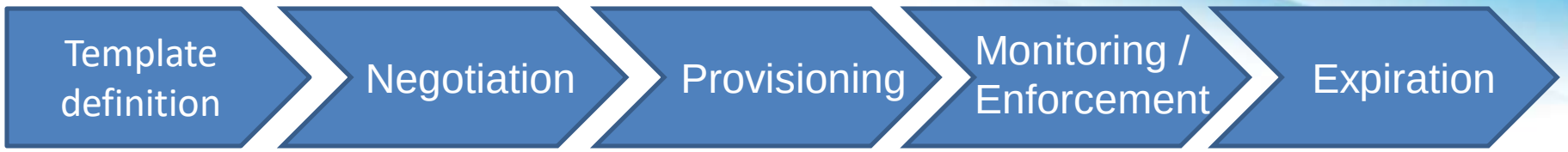
XML syntax of simple SLA Template 2/2

```
<slasoi:AgreementTerm>
  <slasoi:Text/>
  <slasoi:Properties/>
  <slasoi:ID>term_1</slasoi:ID>
  <slasoi:Guaranteed>
    <slasoi:Text/>
    <slasoi:Properties/>
    <slasoi:State>
      <slasoi:ID>guaranteed_state_1</slasoi:ID>
      <slasoi:Priority xsi:nil="true"/>
      <slasoi:Constraint>
        <slasoi:TypeConstraintExpr>
          <slasoi:Value>
            <slasoi:FuncExpr>
              <slasoi:Text/>
              <slasoi:Properties/>
              <slasoi:Operator>http://www.slaatsoi.org/commonTerms#completion_time</slasoi:Operator>
              <slasoi:Parameter>
                <slasoi:ID> the_service </slasoi:ID>
              </slasoi:Parameter>
            </slasoi:FuncExpr>
          </slasoi:Value>
          <slasoi:Domain>
            <slasoi:SimpleDomainExpr>
              <slasoi:ComparisonOp>http://www.slaatsoi.org/coremodel#less_than</slasoi:ComparisonOp>
              <slasoi:Value>
                <slasoi:CONST>
                  <slasoi:Value>10</slasoi:Value>
                  <slasoi:Datatype>http://www.slaatsoi.org/coremodel/units#seconds</slasoi:Datatype>
                </slasoi:CONST>
              </slasoi:Value>
            </slasoi:SimpleDomainExpr>
          </slasoi:Domain>
        </slasoi:TypeConstraintExpr>
      </slasoi:Constraint>
    </slasoi:State>
  </slasoi:Guaranteed>
</slasoi:AgreementTerm>
</slasoi:SLATemplate>
```

Agreement terms

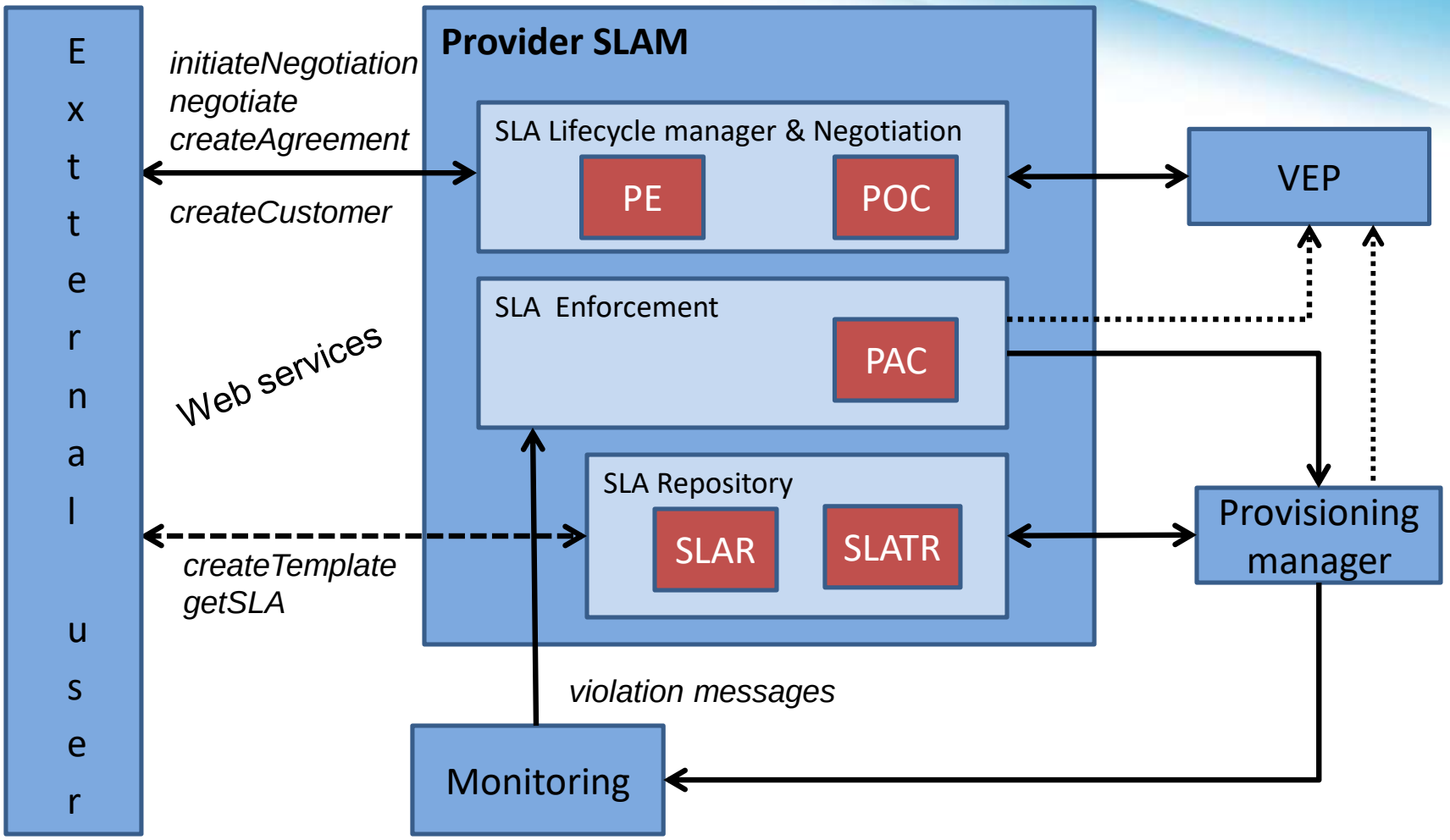
SLA management

SLA Lifecycle

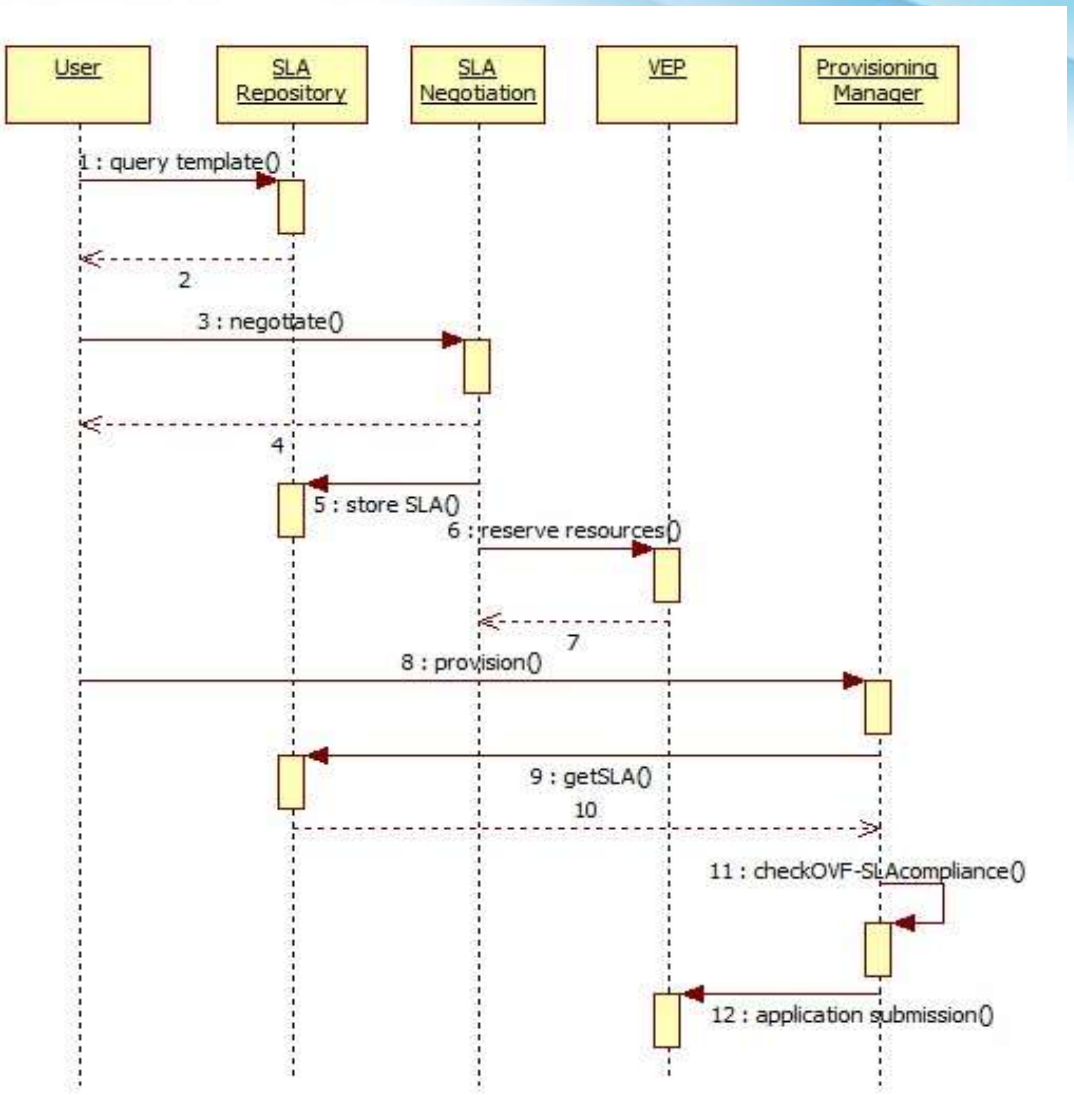


- Templates are defined according to the available services
- Customer selects a SLA template
- Customer negotiates a SLA using a SLA template
- Provisioning: service execution
- Assessment and corrective actions during execution
- Termination and decommission of the service

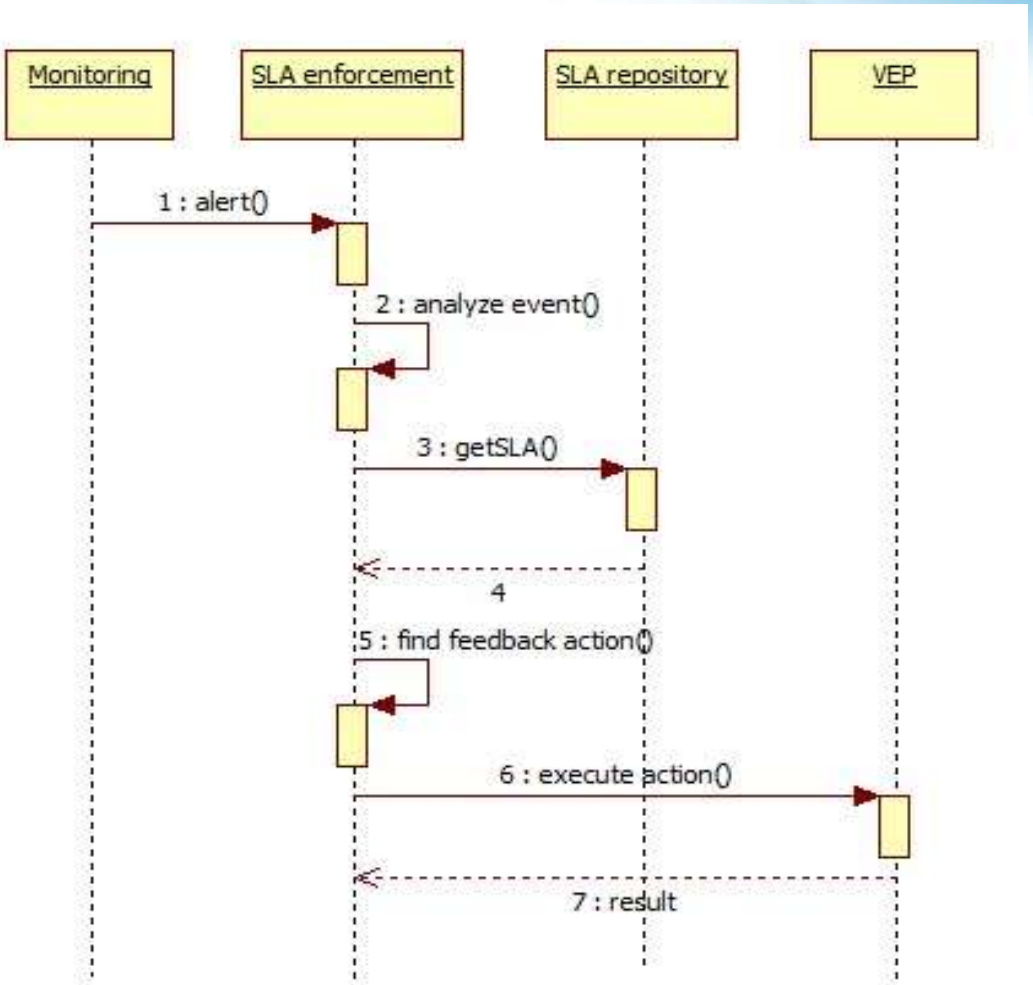
Provider SLAM Architecture



SLA negotiation and provisioning



SLA enforcement



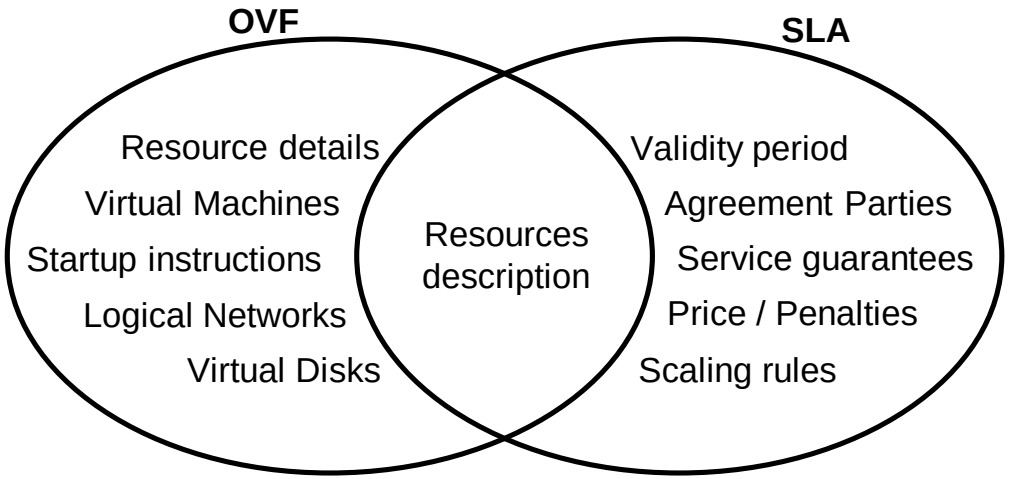
OVF (Open Virtualization Format)

```
<?xml version="1.0" encoding="UTF-8"?>
<Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:vssd="http://schemas.dmtf.org/
http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/CIM_ResourceAllocationSettingData" xmlns:ovf="http:/
  <References>

  <DiskSection>

  <NetworkSection>

  <VirtualSystemCollection ovf:id="LampService">
    <Info>Virtual appliance with a 2-tier distributed LAMP stack</Info>
    <Name>LAMP Service</Name>
    <ProductSection>
    <ProductSection ovf:class="org.linuxdist.x">
    <ProductSection ovf:class="org.apache.httpd">

    <StartupSection>

    <VirtualSystem ovf:id="AppServer">
      <Info>The configuration of the AppServer virtual machine</Info>
      <Name>Application Server</Name>
      <ProductSection ovf:class="org.linuxdist.x">
      <OperatingSystemSection ovf:id="36">
        <Info>Guest Operating System</Info>
        <Description>Linux 2.6.x</Description>
      </OperatingSystemSection>
      <VirtualHardwareSection>
        <Info>Virtual Hardware Requirements: 256 MB, 1 CPU, 1 disk, 1 NIC</Info>
        <System>
          <vssd:ElementName>Virtual Hardware Family</vssd:ElementName>
          <vssd:InstanceID>0</vssd:InstanceID>
          <vssd:VirtualSystemType>vmx-04</vssd:VirtualSystemType>
        </System>
        <Item>
          <rasd:Description>Number of virtual CPUs</rasd:Description>
          <rasd:ElementName>1 virtual CPU</rasd:ElementName>
          <rasd:InstanceID>1</rasd:InstanceID>
          <rasd:ResourceType>3</rasd:ResourceType>
          <rasd:VirtualQuantity>1</rasd:VirtualQuantity>
        </Item>
        <Item>
          <rasd:AllocationUnits>byte * 2^20</rasd:AllocationUnits>
          <rasd:Description>Memory Size</rasd:Description>
```

- OVF is a standard format used to describe multiple virtual resources, their properties and their connections
- Provisioning services in Contrail use OVF descriptors as input

Combining SLAs with OVF

- Role of OVF: to describe resources
- Role of SLA: to express guarantees on services
- SLAs contain both a service description and guarantees about the service
- There is a semantic overlap between OVFs and SLAs for IaaS resources

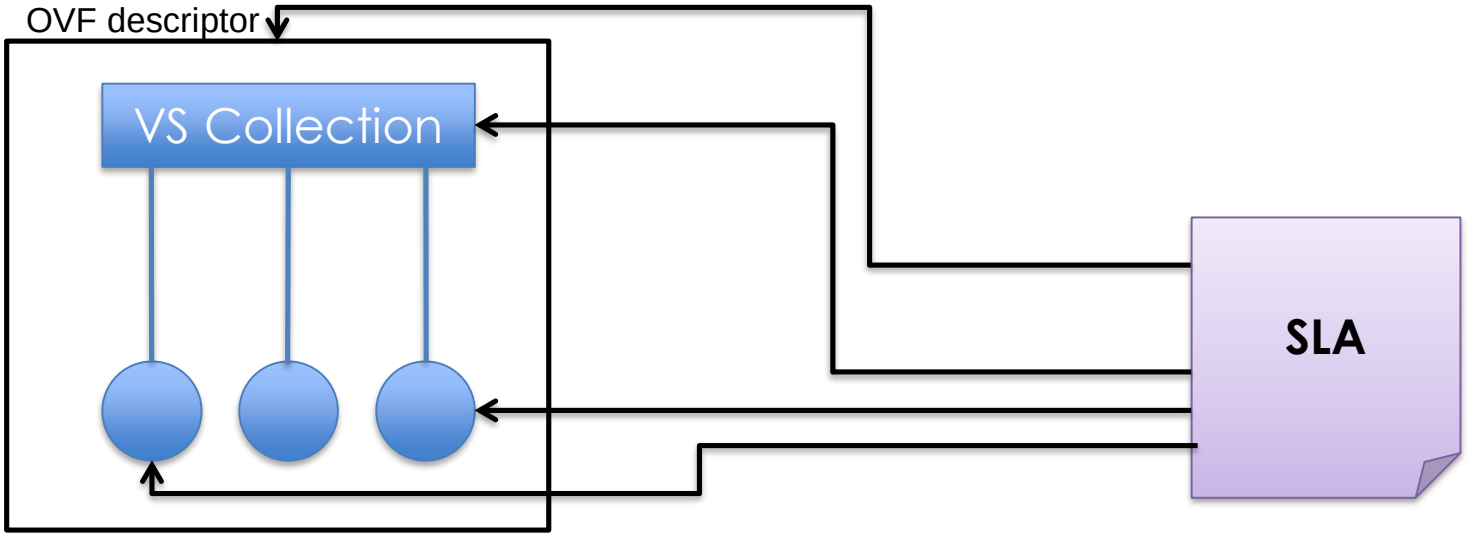


Combining OVF and SLA

- Need a link between guarantees (SLA) and resources (OVF)
  - To enforce SLA terms, resources referenced in the guarantees must be well known
- Need elasticity
  - Contrail should support negotiation of long lasting SLAs for provisioning multiple OVFs
  - A SLA should be defined independently from the OVFs to be provisioned
- Possible solutions
  1. Specific SLAs. They refer to specific appliances / OVF resources
     1. E.g. VM34 will have 8Gb of RAM
  2. Generic SLAs. They express guarantees that can be applied to all OVFs / appliances
     1. E.g. All VMs will have 4Gb of RAM

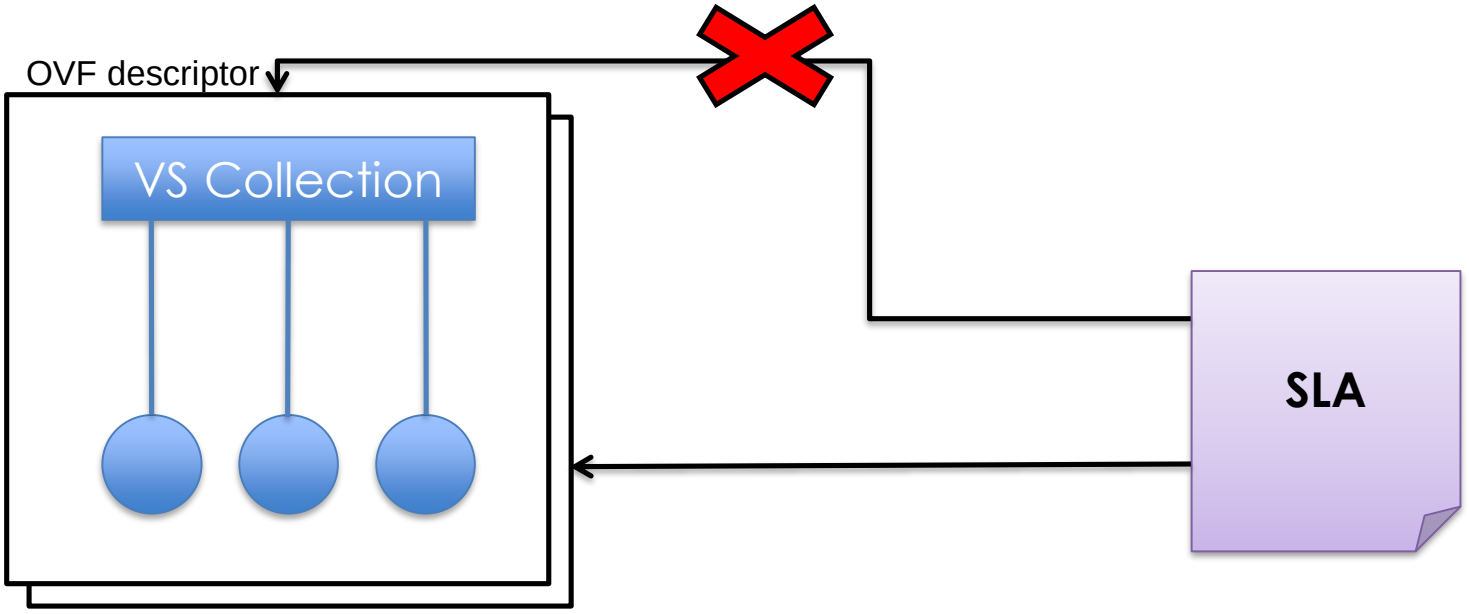
Specific SLAs

- Refer to a specific OVF file
- Express guarantees about specific OVF items (e.g. Virtual Systems or VS Collections)



Generic SLAs

- Do NOT refer to a specific OVF file
- Express guarantees valid for all OVF appliances



Specific SLA: reference to an OVF file

```
<slasoi:InterfaceDeclr>
        <slasoi:Text>Interface for overall OVF</slasoi:Text>
        <slasoi:Properties>
                <slasoi:Entry>
                        <slasoi:Key>OVF_URL</slasoi:Key>
                        <slasoi:Value>/opt/contrail/ ovf/lamp.ovf</slasoi:Value>
                </slasoi:Entry>
        </slasoi:Properties>
        <slasoi:ID>OVF-Descriptor-General</slasoi:ID>
        <slasoi:ProviderRef>ContrailProvider</slasoi:ProviderRef>
        <slasoi:Endpoint>
                <slasoi:Text/><slasoi:Properties />
                <slasoi:ID>OVF-Endpoint</slasoi:ID>
                <slasoi:Location>VEP-ID</slasoi:Location>
                <slasoi:Protocol>http://www.slaatsoi.org/slamodel#HTTP</slasoi:Protocol>
        </slasoi:Endpoint>
        <slasoi:Interface>
                <slasoi:InterfaceResourceType>
                        <slasoi:Name>OVFDescriptor</slasoi:Name>
                </slasoi:InterfaceResourceType>
        </slasoi:Interface>
</slasoi:InterfaceDeclr>
```

Reference type: OVF descriptor (file)

Specific SLA: reference to VirtualSystem

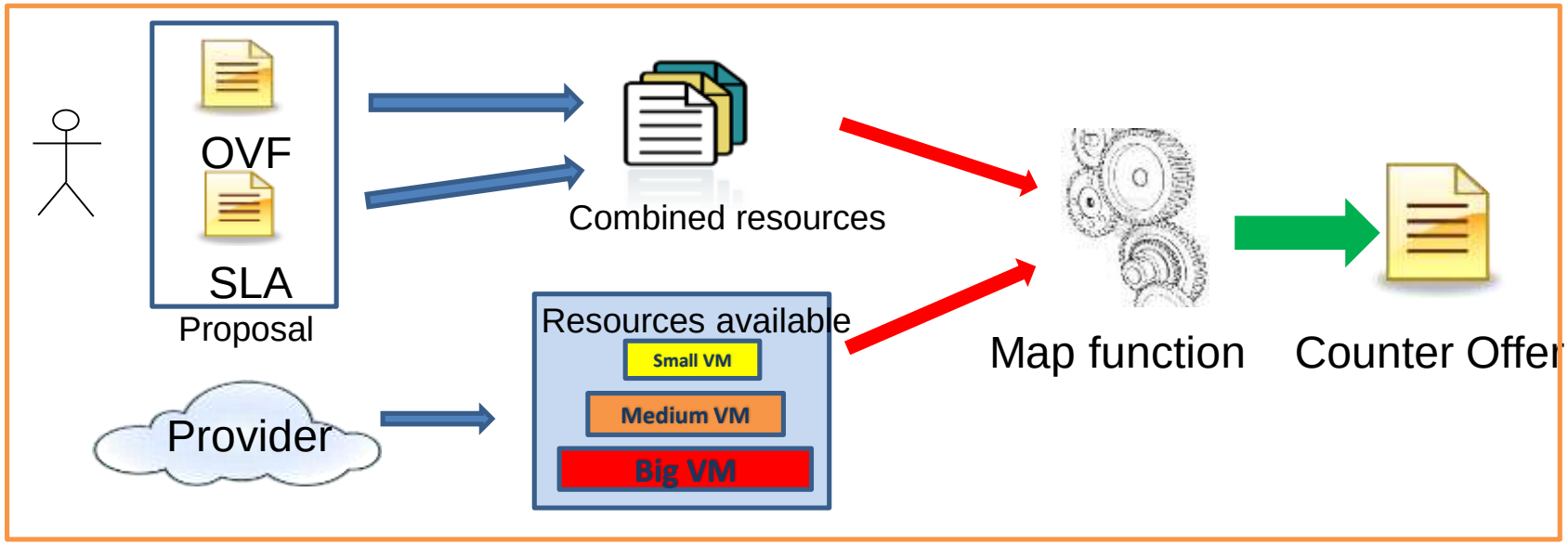
```
<slasoi:InterfaceDeclr>
        <slasoi:Text>Interface to specific OVF item</slasoi:Text>
        <slasoi:Properties><slasoi:Entry>
                                <slasoi:Key>OVF_URL</slasoi:Key>
                                <slasoi:Value>/opt/contrail/ ovf/lamp.ovf</slasoi:Value>
        </slasoi:Entry></slasoi:Properties>
        <slasoi:ID>OVF-Descriptor-LAMP</slasoi:ID>
        <slasoi:ProviderRef>ContrailProvider</slasoi:ProviderRef>
        <slasoi:Endpoint>
                <slasoi:Text/>
                <slasoi:Properties><slasoi:Entry>
                        <slasoi:Key>OVF_VirtualSystem_ID</slasoi:Key>
                        <slasoi:Value>MyLampService</slasoi:Value>
                </slasoi:Entry></slasoi:Properties>
                <slasoi:ID>VM-with-Linux-Apache-MySQL-PHP</slasoi:ID>
                <slasoi:Location>VEP-ID</slasoi:Location>
                <slasoi:Protocol>http://www.slaatsoi.org/slamodel#HTTP</slasoi:Protocol>
        </slasoi:Endpoint>
        <slasoi:Interface>
                <slasoi:InterfaceResourceType>
                        <slasoi:Name>OVFAppliance</slasoi:Name>
                </slasoi:InterfaceResourceType>
        </slasoi:Interface>
</slasoi:InterfaceDeclr>
```

VirtualSystem ID inside the OVF

Reference type: OVF appliance (VirtualSystem)

Negotiation model for specific SLA

- Optimization problem: find "nearest" offer that fulfill user's request
  - Constraints:
    - OVF (lower bound): provider can't offer less than the request
    - VM Handler (upper bound): provider can't offer more than resource available



Pricing model

- Logic determining the price of the services in the SLA Offer:

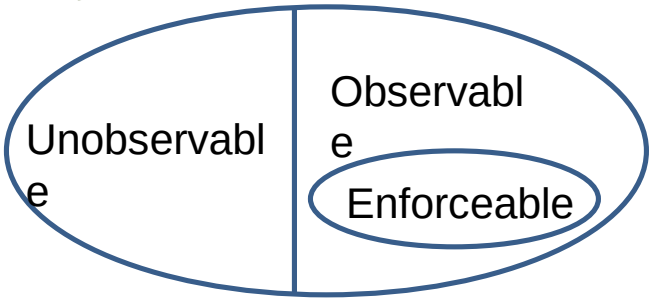
```
totalPrice = resourcesPrice + guaranteePrice

resourcesPrice= (cpu_speed *  cpu_speed_unit_price) +
                (vm_cores  *  vm_core_unit_price) +
               (memory    *  memory_unit_price)

guaranteesPrice = (resourcesPrice *
       guaranteeModificationPricePercentage)
```

- It's open and flexible to meet the particulars pricing policies of different cloud computing providers. It's possible to associate a price for each individual resource (CPU, RAM, HDD, etc.), a specific OVF, or a particular VM configuration.

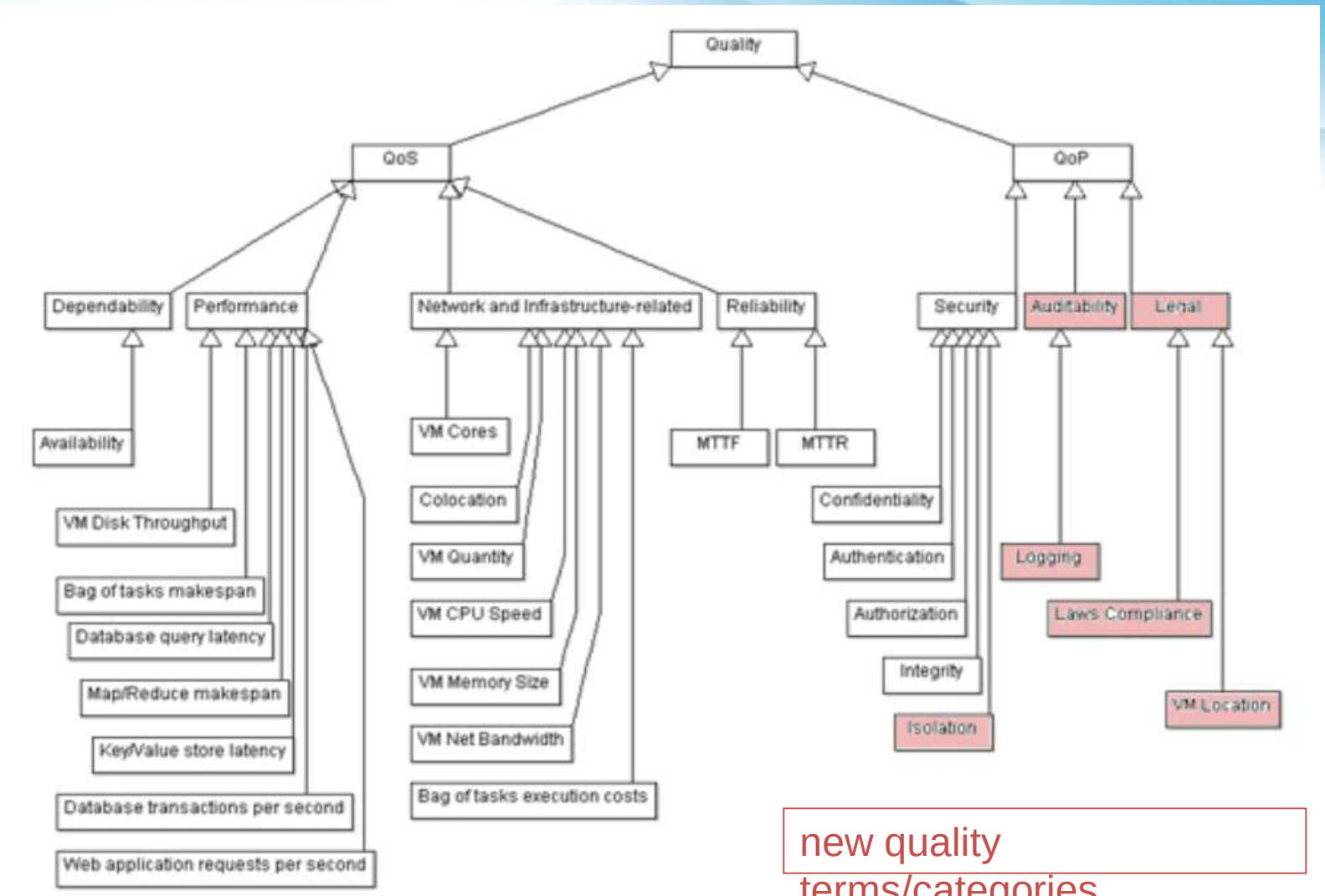
SLA terms

Classification of SLA Terms

- Term type
  - QoS (constraints on performance)
  - QoP (level of protection)
- Evolvability
  - Static (static properties, e.g. # of VM cores)
  - Dynamic (non static props., e.g. RAM)
- Monitorability & Controllability
  - Unobservable
  - Observable
    - Enforceable
- Categorization in a Quality Model (e.g. S-CUBE)
  - Performance, Dependability, Reliability, Network-related, Infrastructure-related, Security, Auditability, Legal



Quality Model



new quality terms/categories

Example QoS/QoP SLA Terms

- QoS
  - Availability
    - the probability that the cloud infrastructure is running over a predefined monitoring period (usually a month or a year)
  - Virtualization factor
    - ratio between virtual and physical CPU cores
- QoP
  - Location
    - country code where the resource (VM / storage) is located
  - Isolation level
    - depends on which measures are taken to ensure isolation between different users hosted on the same infrastructure (e.g. storage encryption, communication encryption, zero-filling released storage, …)

SLA Terms supported in Contrail (v1.2)

Term name	Description	Type	Evolv.	Enforc.
vm_cores	number of cores assigned to a VM	QoS	static	Enf.
memory	RAM size assigned to a VM	QoS	dynamic	Enf.
cpu_speed	CPU frequency assigned to a VM	QoS	static	Enf. (*)
cpu_load	average system load over a 5-minute period	QoS	dynamic	Enf.
location	country code where the resource (VM / storage) is located	QoP	dynamic / static	Obs.

(*) quantized values

Additional SLA terms in next releases

Term name	Description	Type	Evolv.	Enforc.
reservation	Resources reservation	QoS	static	Enf.
co-location (rack)	Location of VM in the same cluster/host	QoS	static	Enf.
minimum LoA	Minimum level of authentication needed to access users' resources	QoP	static	Enf.
availability	% of uptime of the whole provider infrastructure	QoS	dynamic	Obs.
vm_cpu_load	[5 min CPU avg load / # of cores] for the VM of a VirtualSystem	QoS	dynamic	Obs.
host_cpu_load	[5 min CPU avg load / # of cores] for the host of a VirtualSystem	QoS	dynamic	Enf.

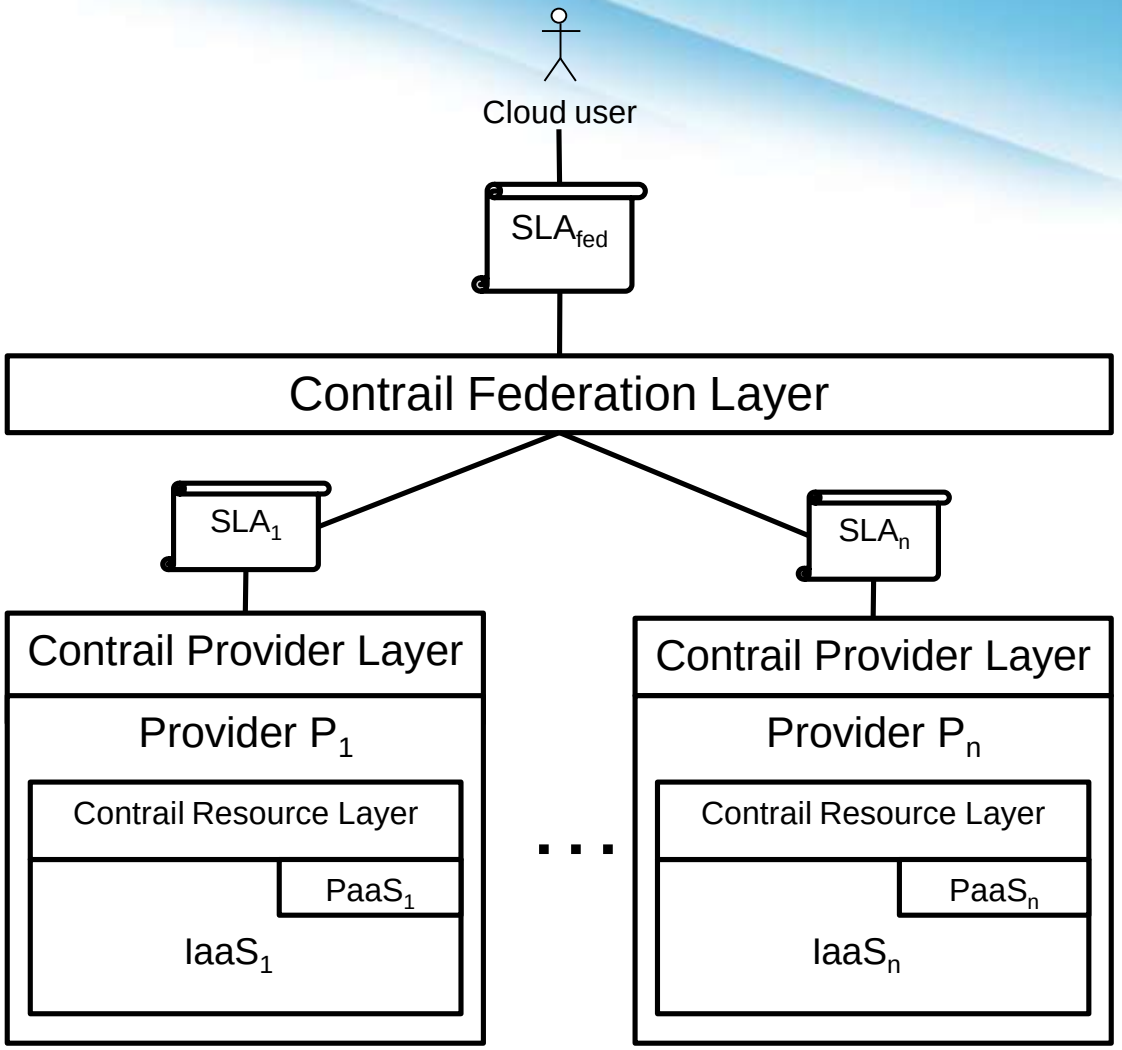
Additional SLA terms in next releases /2

Term name	Description	Type	Evolv.	Enforc.
location for storage	replicas for the given volume are placed in the given countries	QoP	dynamic	Obs.
not-co-location (host)	VMs are not allocated on the same cluster/host	QoS	static	Enf.
reliability for storage	number N of replicas for the given volume	QoS	static	Enf.
bandwidth	Applied to a single network defined in the OVF	QoS	static	Enf.
public IP address	Applied to a single network defined in the OVF	QoS	static	Enf.

Multilevel SLA management

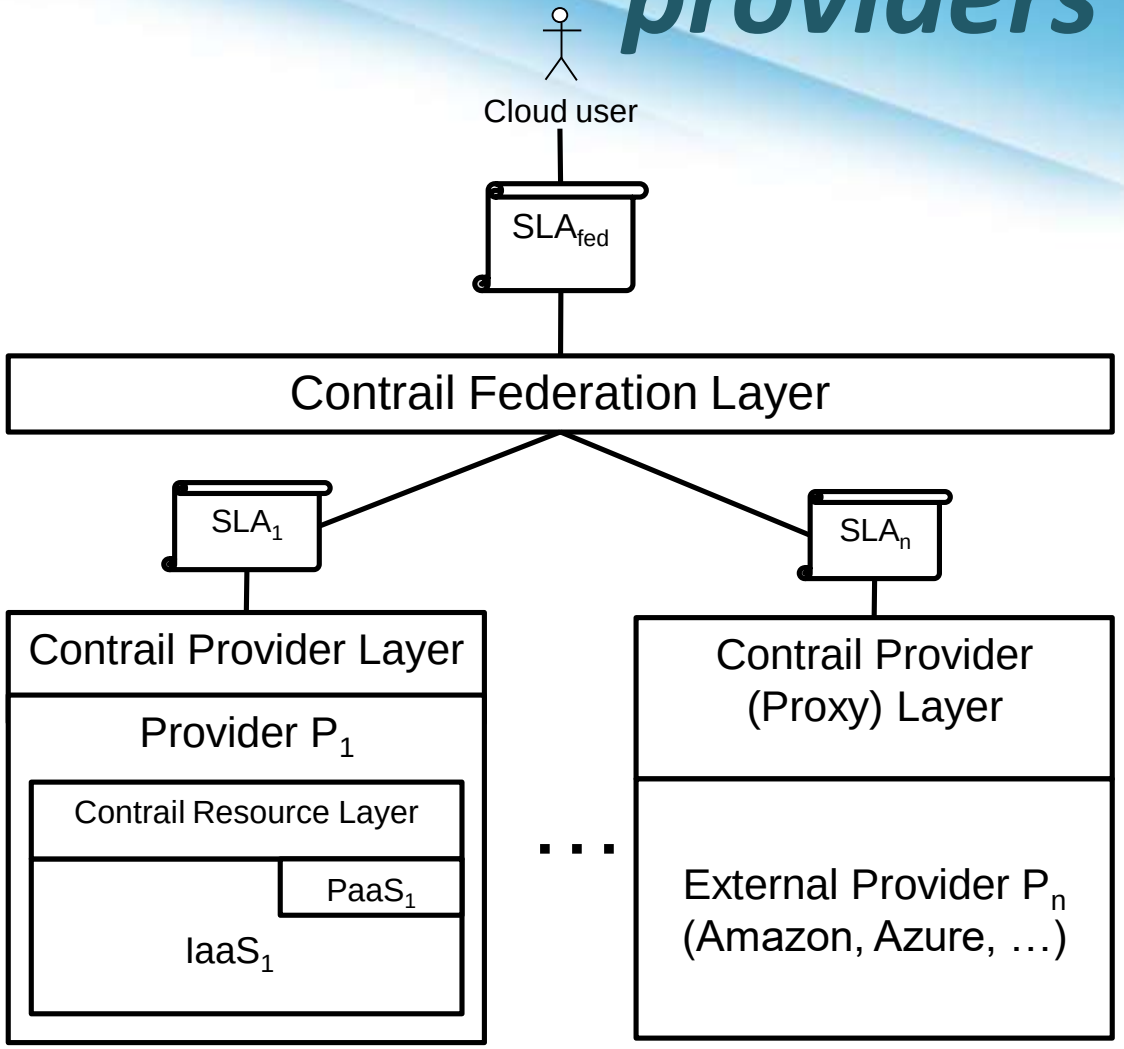
SLA Interaction Model

- User negotiates a SLA with the Federation
- Federation negotiates SLAs with one or more providers, on behalf of the user



Future: integration of external providers

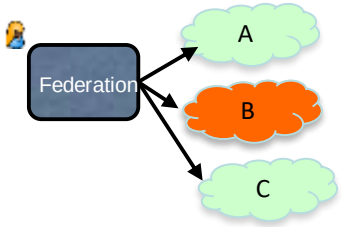
- External providers will be integrated in future versions of Contrail
- The interaction model will not change for the user



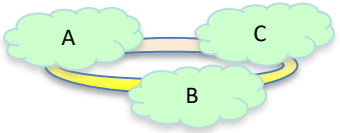
SLAs in Contrail are the main pillars for Cloud Federations

- Federation: abstraction of providers
  - SLA+OVF is a unified way for expressing user requirements
  - QoS/QoP requested by each customer can be satisfied irrespective of the provider

- Federation: broker of providers
  - Provider selection is based on SLAs
    - new market and business model for intermediate players



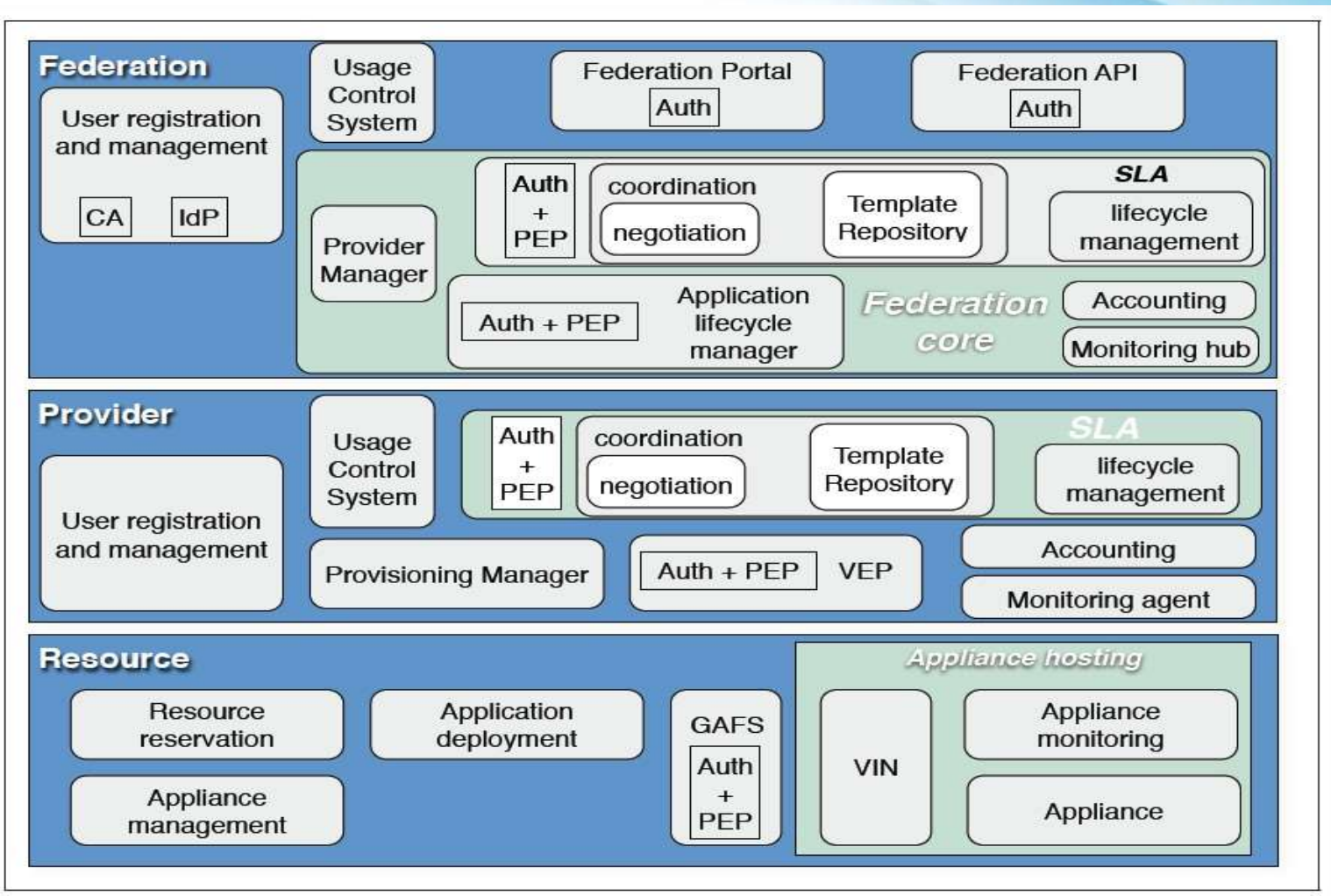
- Federation: small providers can join forces
  - SLA splitting allows distributing application over multiple providers



Added value of Contrail approach: Federation as a Virtualization of Clouds

- Contrail Federations relieve the user from managing cloud providers
  - Inner complexity is hidden to users
  - Federated Identity Management
  - Best / cheapest cloud provider is automatically selected based on user preferences
- Comparing SLAs and selecting the best provider opens new Cloud mediators market
- Automatic SLA negotiation allows Cloud providers to personalize their offer
- Worldwide Clouds made possible

Contrail high level architecture



SLA management in a Cloud Federation

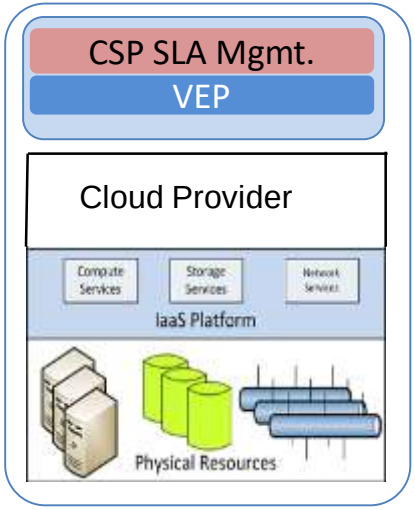
SLA

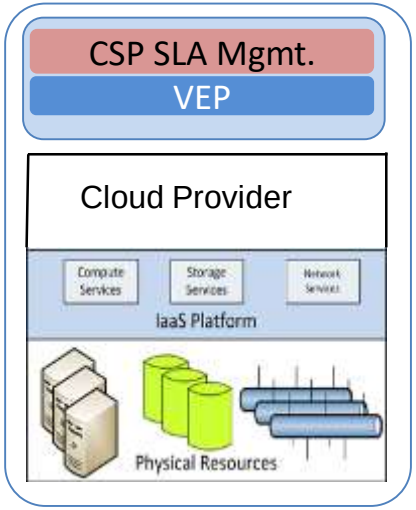
Federation

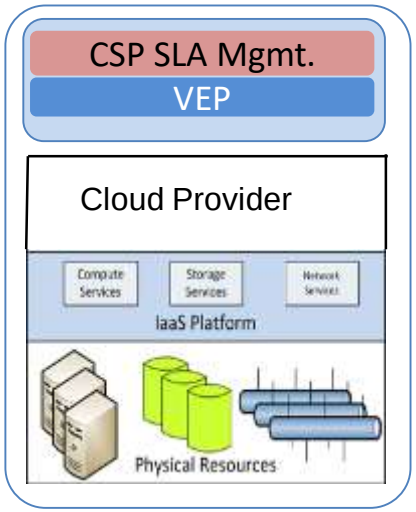
- Provider Selection
- Application Splitting
- Federation SLA Management
- Federated Identity
- Accounting
- Provisioning Manager

Security

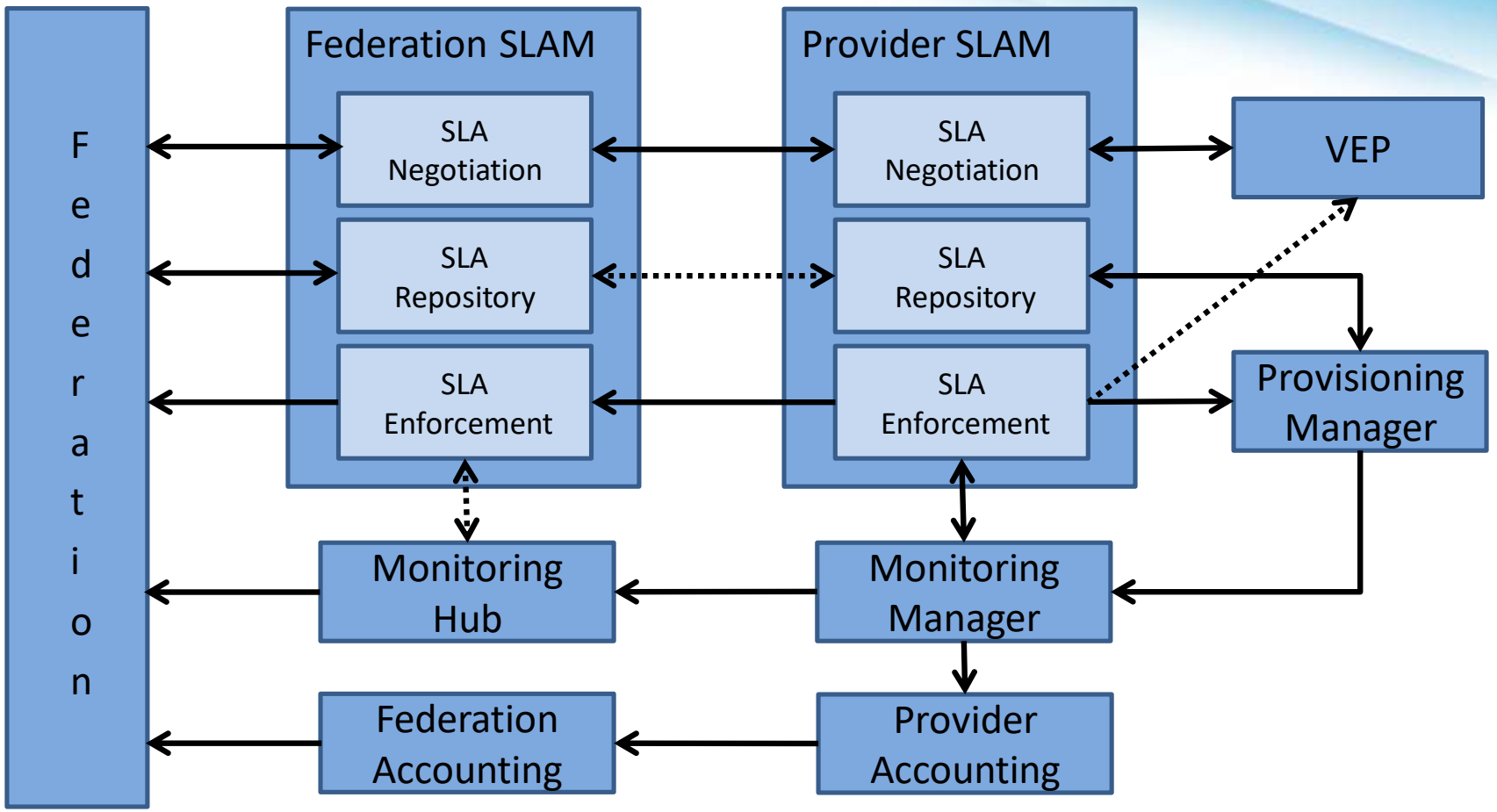
Monitoring



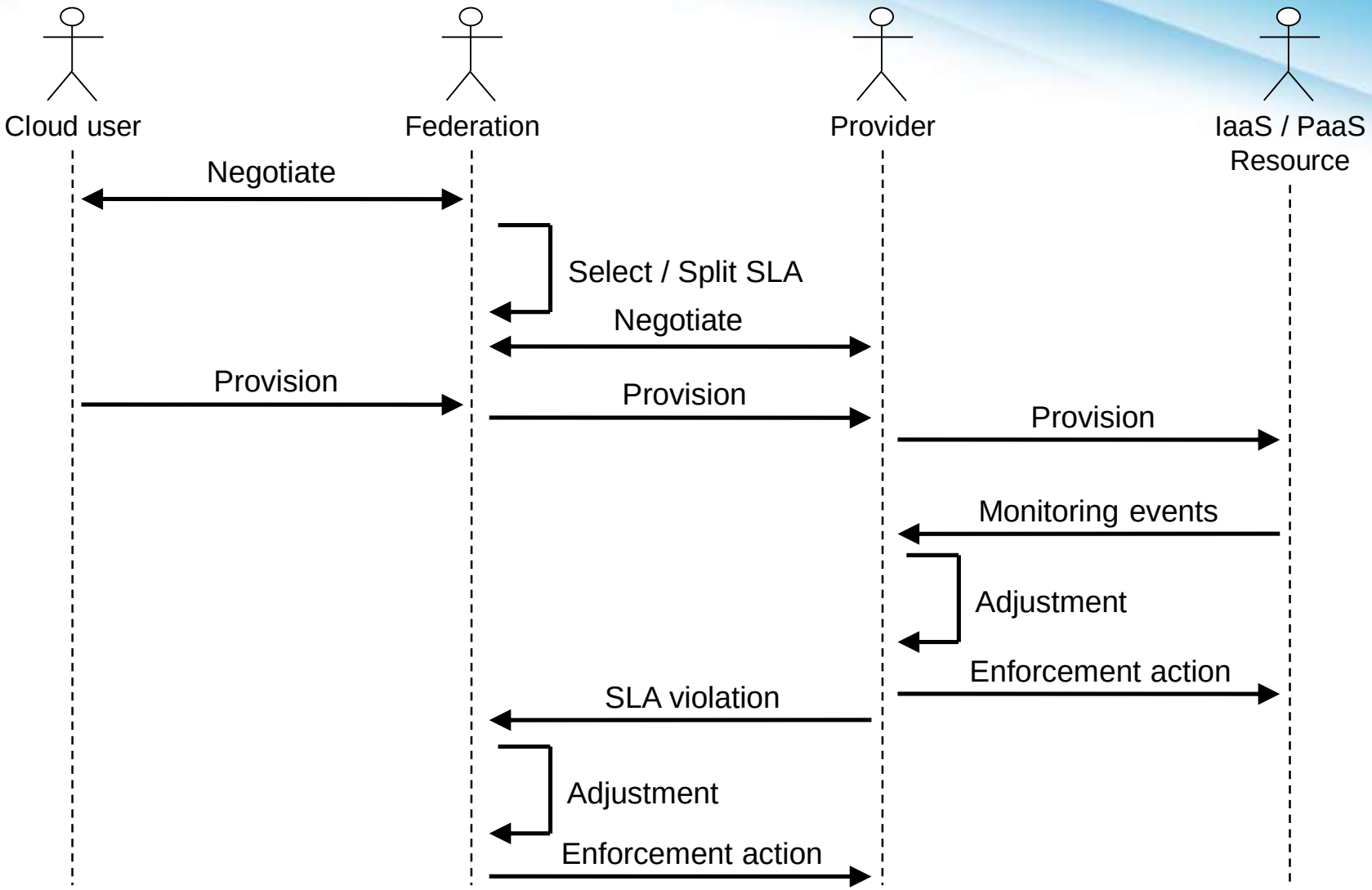




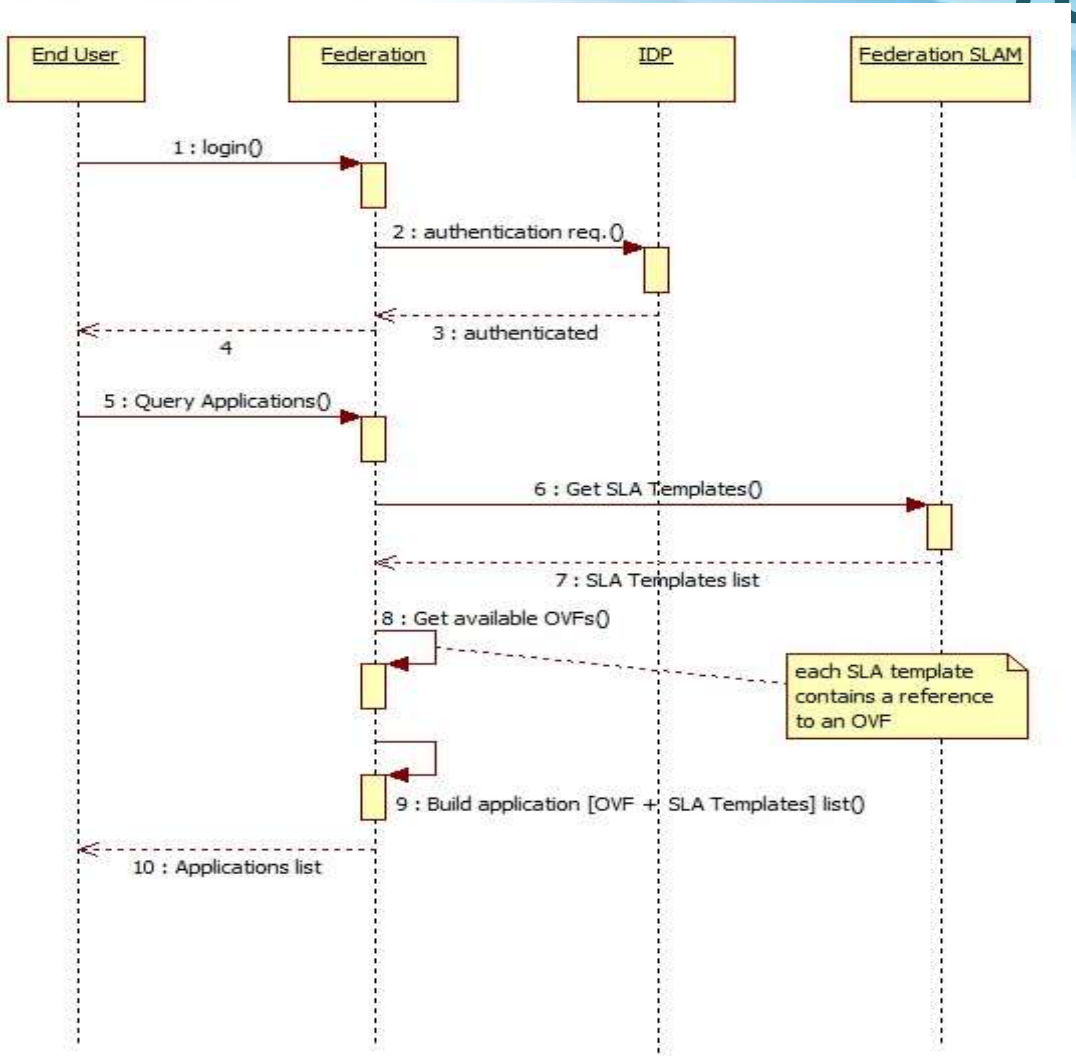
Multilevel SLA Management Architecture



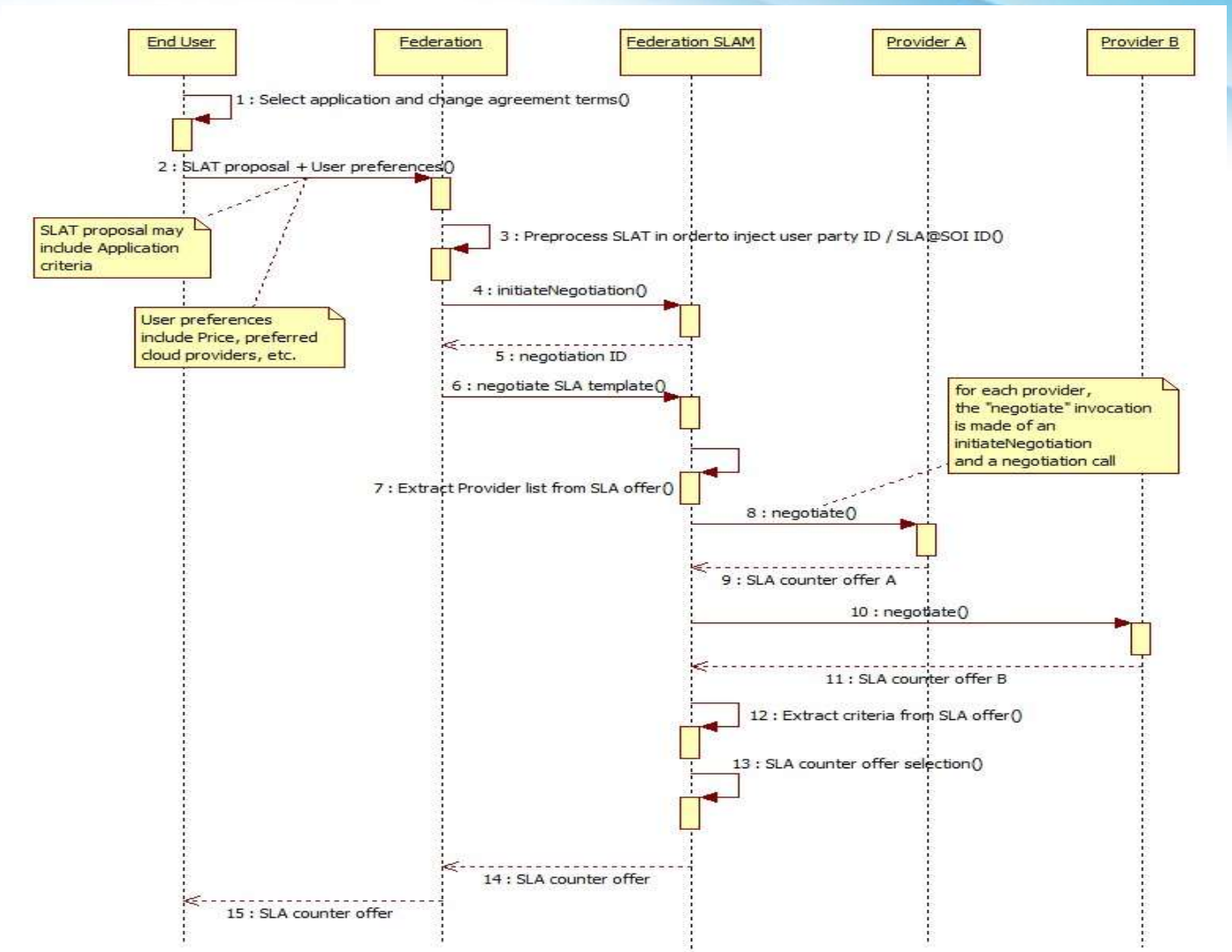
General interaction model



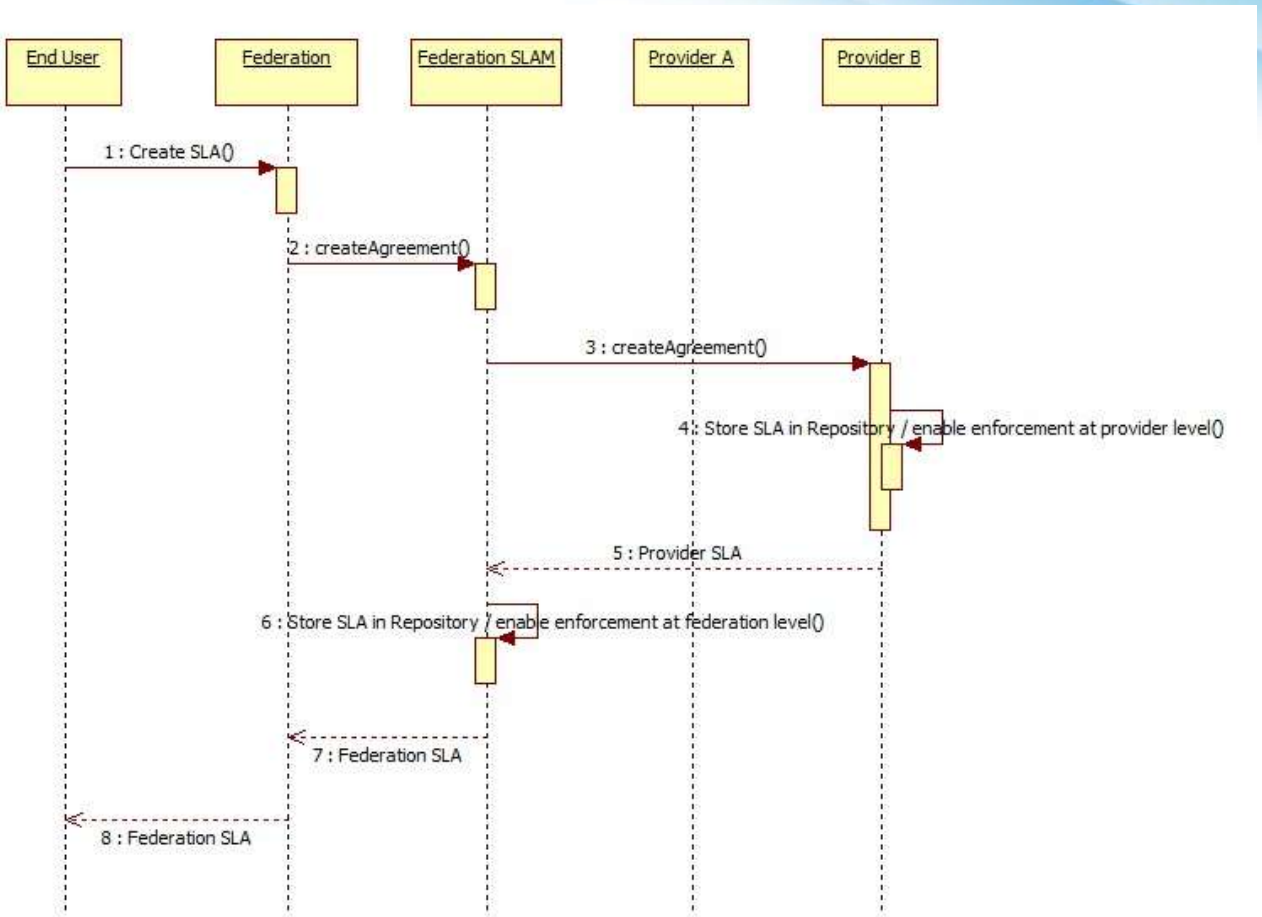
User access and applications retrieval



SLA negotiation /1



SLA negotiation /2



Selection

Provider selection

- Happens during user-federation negotiation
- Driven by user-specified preferences
- May be guided also by provider reputation, if the federation keeps track of it
- Roles:
  - The federation component gives the list of providers to be contacted by the Federation SLAM
    - It might even vary for each negotiation round
  - The Federation SLAM negotiate the SLA template (proposal) with each provider and then selects the best SLA offers

User preferences

- Users drive the provider selection process by specifying user preferences during the negotiation
- User preferences include:
  - Selection criteria (e.g. minimize price or maximize quality or even maximize penalties)
  - Positive or negative preference on individual providers (e.g. not on Azure)

User criteria

- Each criterion is a pair {Guarantee Term, Weight}, weight $\in$ {0, 1}
- Weight express the importance of the term for that criterion (max importance = 1).
  - By default, the weight is considered 0.
- Currently supported terms:
  - cpu_speed
  - core_number
  - memory
  - price

Provider list and User criteria - Syntax

- The Federation uses the SLA proposal itself to pass User criteria and Provider list to the Federation SLAM

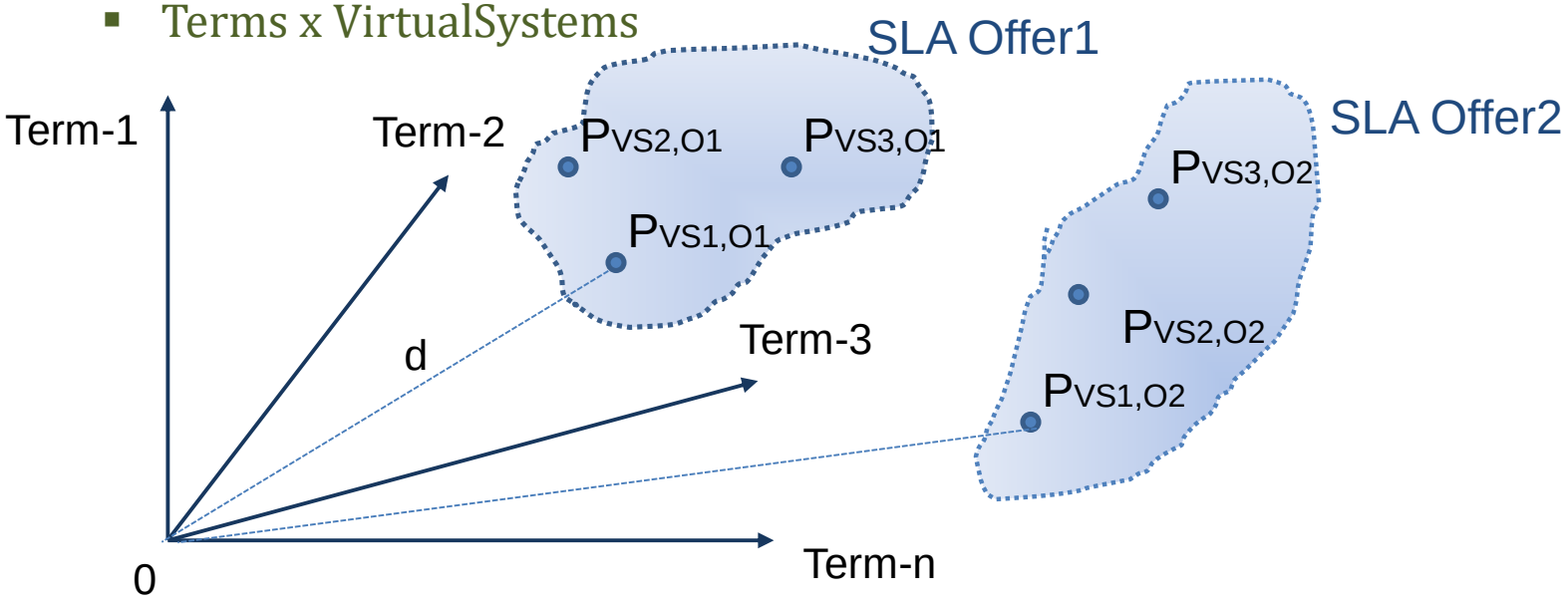
```
<slasoi:SLATemplate xmlns:slasoi="http://www.slaatsoi.eu/slamodel" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <slasoi:Text/>
  <slasoi:Properties>
    <!-- -This entry describes the list of cloud providers to be considered during the negotiation at federation level -->
    <slasoi:Entry>
      <slasoi:Key>ProvidersList</slasoi:Key>
      <slasoi:Value>
        {"ProvidersList": [
        {"provider-uuid":"42", "p-slam-url":"http://10.15.8.2:8080/services/contrailNegotiation?wsdl"},
        {"provider-uuid":"37", "p-slam-url":"http://10.15.8.23:8080/services/contrailNegotiation?wsdl"}
        ]}
      </slasoi:Value>
    </slasoi:Entry>
    <!-- -This entry describes the criteria be used during the negotiation at federation level -->
    <slasoi:Entry>
      <slasoi:Key>Criteria</slasoi:Key>
      <slasoi:Value>
        {"vm_cores":"0.8", "memory":"0.3", "price":"0.9"}
      </slasoi:Value>
    </slasoi:Entry>
  </slasoi:Properties>

  <slasoi:UUID>Contrail-SLAT-NewFeatures-02</slasoi:UUID>
  <slasoi:ModelVersion>1</slasoi:ModelVersion>
```

Selection process

1. Offer guarantee terms are normalized versus user's proposal
   - e.g. value of offer guarantee term / value of proposal guarantee term
2. User criteria are extracted from SLA Template proposal
3. Normalized values are weighted using Criteria related weight
   - e.g. normalized term value * term weight
4. An evaluation algorithm is applied to the set of weighted values
5. SLA offers are ordered according to the output values of the evaluation algorithm.
6. SLA offers are filtered

Evaluation algorithms

- Evaluation algorithms, used to order SLA offers, and so determine the best one, take into account user criteria weights for impacted terms in each offer
- The problem is transformed in maximizing the Euclidean distance of a point from the origin, in an n-dimensional space.
- Dimensions can be:
  - Terms
  - Terms x VirtualSystems

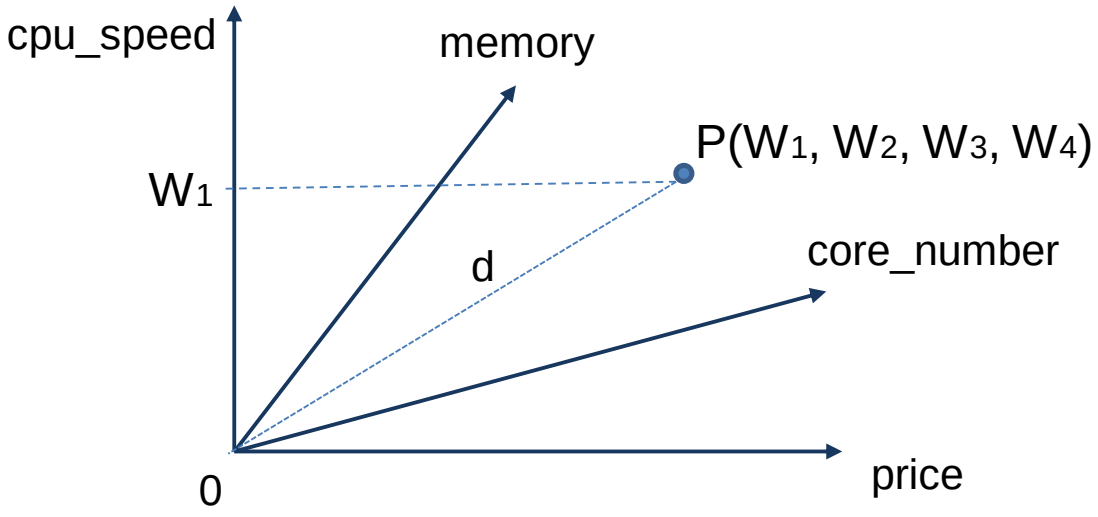


Evaluation algorithms

- Different evaluation algorithms have been identified for ranking the SLA offers
- Based upon user criteria, one can privilege the offer including the most convenient VirtualSystem, etc.
- The Federation SLA Manager is configurable: the Federation Administrator can select the preferred algorithm, or can write their own Java class for it (implementing a common interface)

Algorithm example 1: best virtual system /1

- For each VirtualSystem of each SLA offer, the algorithm computes the Euclidean distance from the origin of a t-dimensional point, t=NumTerms



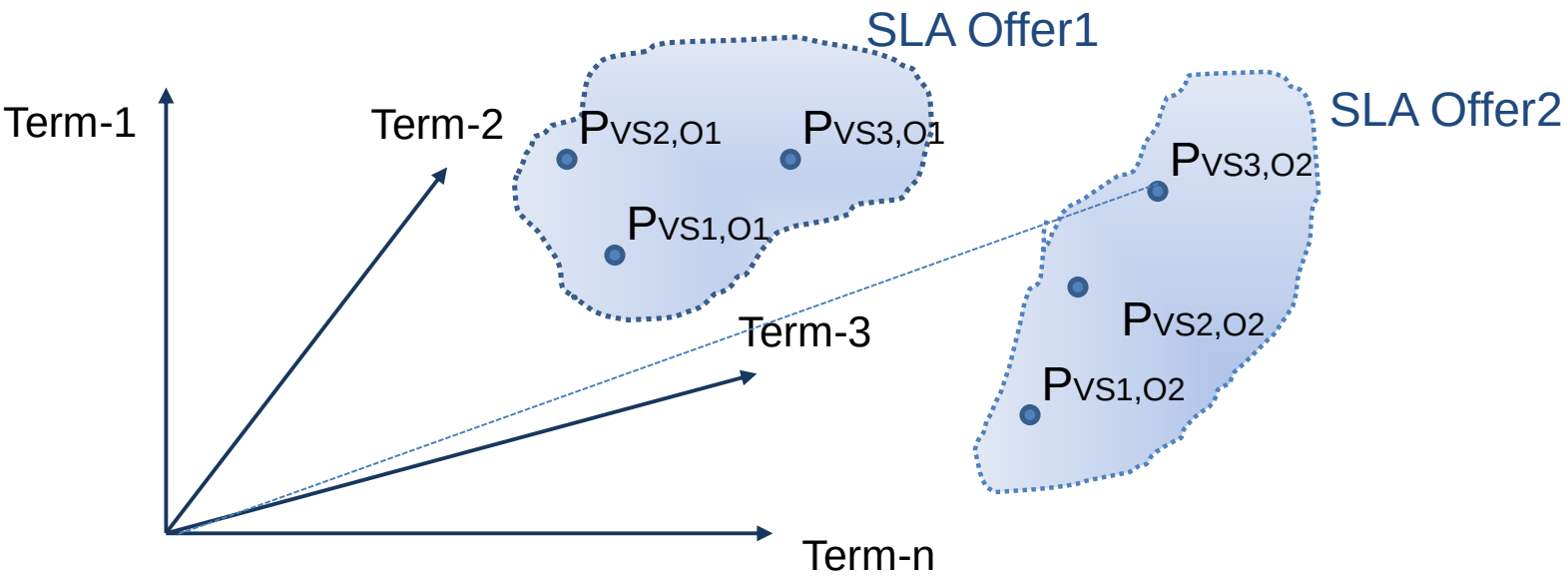
$$d_j = \sqrt{\sum_{i=1}^{t} Wi^2}$$

d_j = distance of j-th VirtualSystem

W_i = weighted value of i-th term

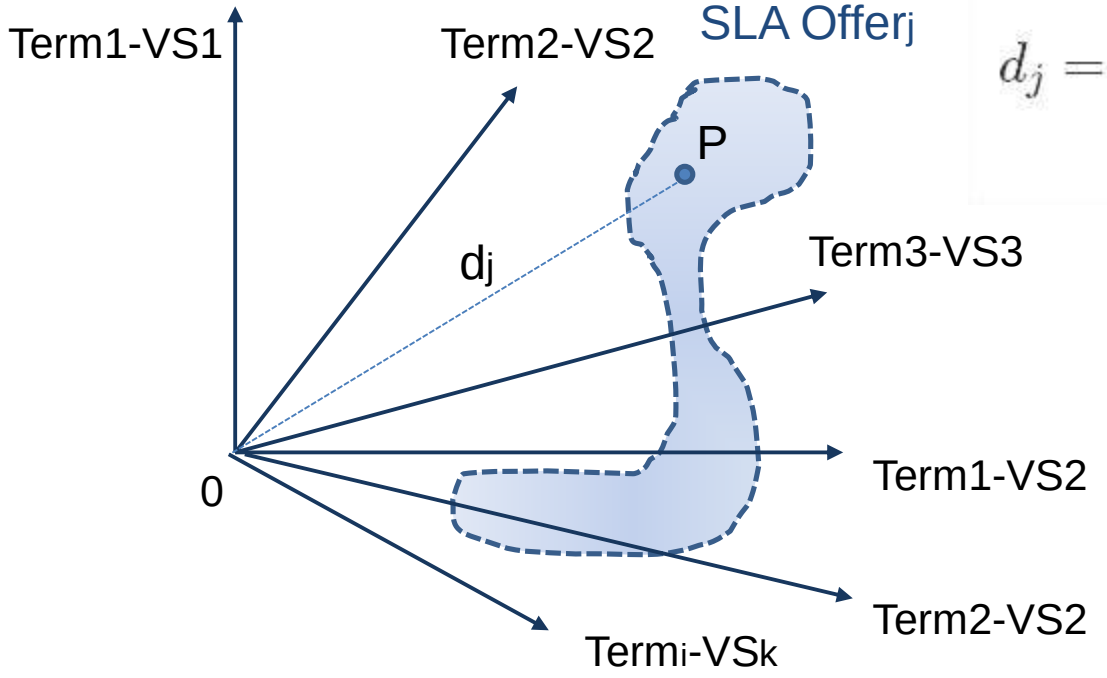
Algorithm example 1: best virtual system /2

- The SLA offers are ranked based on their VirtualSystems distances
- The best SLA Offer is the one including the farther VirtualSystem from origin



Algorithm example 2: best mean offer /1

- For each SLA offer, the algorithm computes the Euclidean distance from the origin of an {t*v}-dimensional point
  t=NumTerms, v=NumVirtualSystems



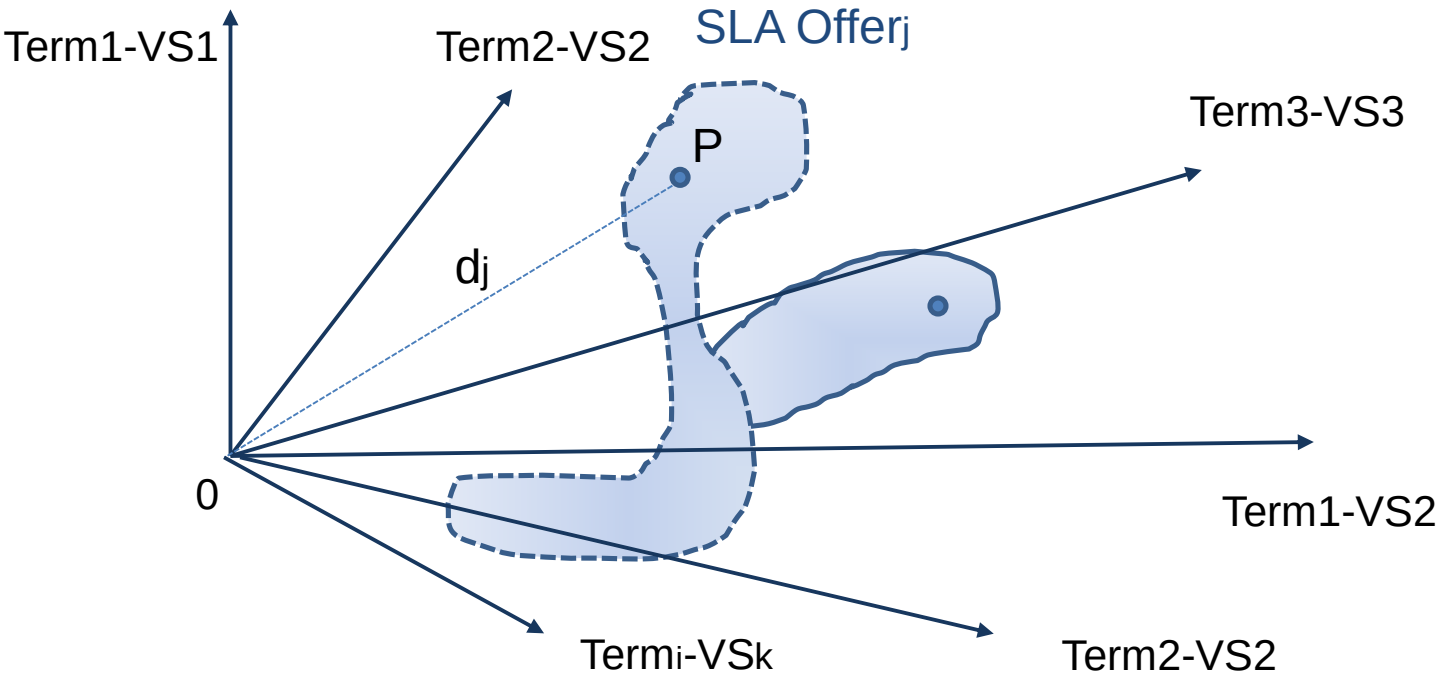
$$d_j = \sqrt{\sum_{i=1}^{t} Wi_1^2 + \ldots + \sum_{i=1}^{t} Wi_v^2}$$

j = {1 .. NumOffers}

$W_{i\,k}$ = weighted value of i-th term for the k-th virtualSystem

Algorithm example 2: best mean offer /2

- The SLA offers are ranked based on their distances
- The best SLA Offer is the farthest one



Other algorithms

- Select the offer having the best “average” VirtualSystem
- Select the offer having the best VirtualSystem in relation to the average of the same VirtualSystem for the other n-1 offers
- Select the offer having the best average VirtualSystem in relation to the other offers
- …

SLA splitting

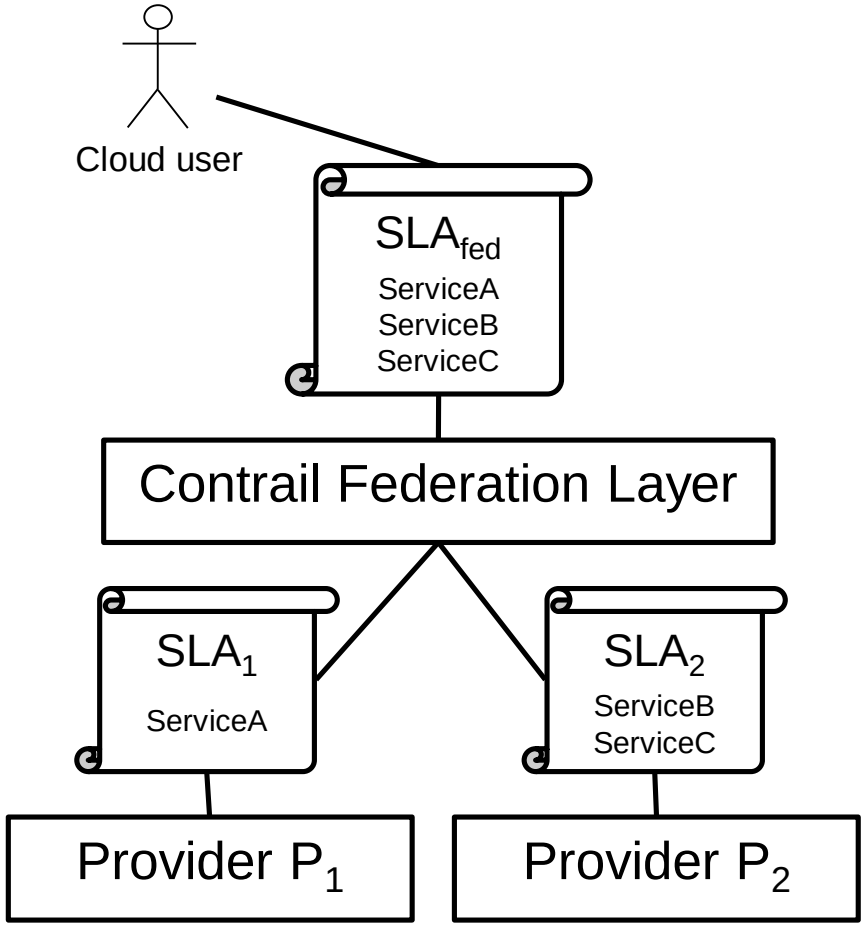
Why splitting SLAs?

- The SLA (or application) requires services that cannot be found on only one provider
  - E.g.: the customer wants to distribute its application in several countries
- The amount of resources required goes beyond what a single provider can offer (and guarantee)
  - A service provider does not have to be the size of Amazon to participate in the Federation
- The required performance cannot be guaranteed by any single provider alone
  - Scaling across multiple providers
- The Federation, according to its own risk management strategy, wants to distribute the risk of satisfying the user SLA

Splitting SLAs

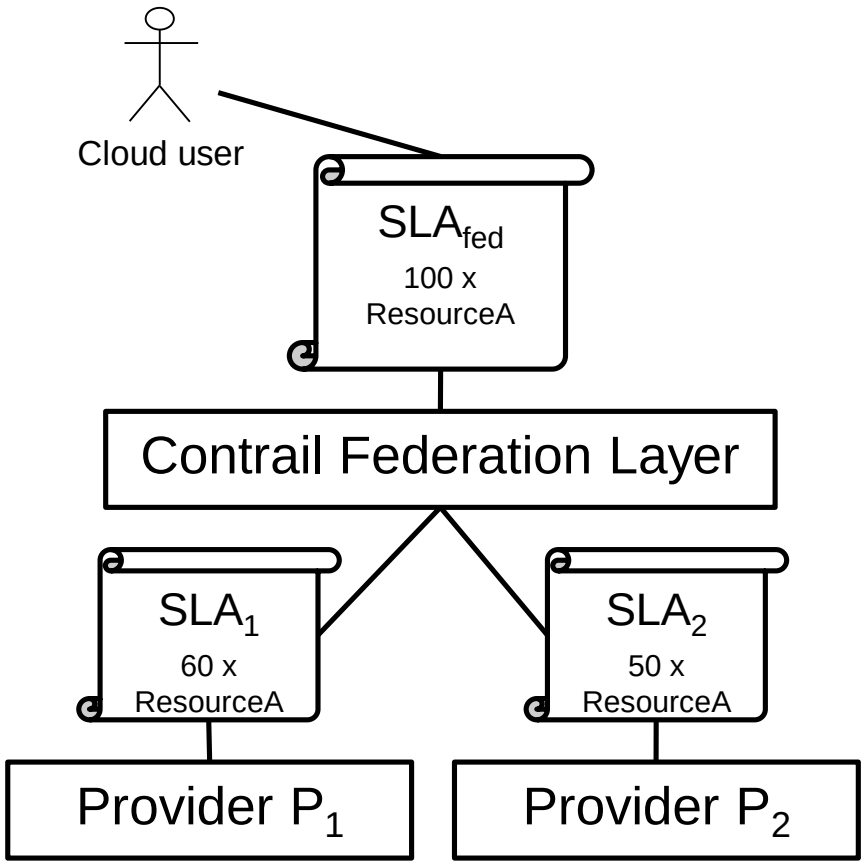
- The Federation splits the user SLA and assigns the resulting SLAs to multiple providers
- Three types of SLA splitting strategies are considered:
  - Service-based SLA splitting
  - Resource-based SLA splitting
  - Performance-based SLA splitting
- Selection of the best split for a given SLA is a multi-objective optimization problem (total price, QoS offered, reputation…)
- Some terms cannot be split (invariant terms, typically QoP)
- Not only the SLA, but also the application must be split

Service-based SLA splitting



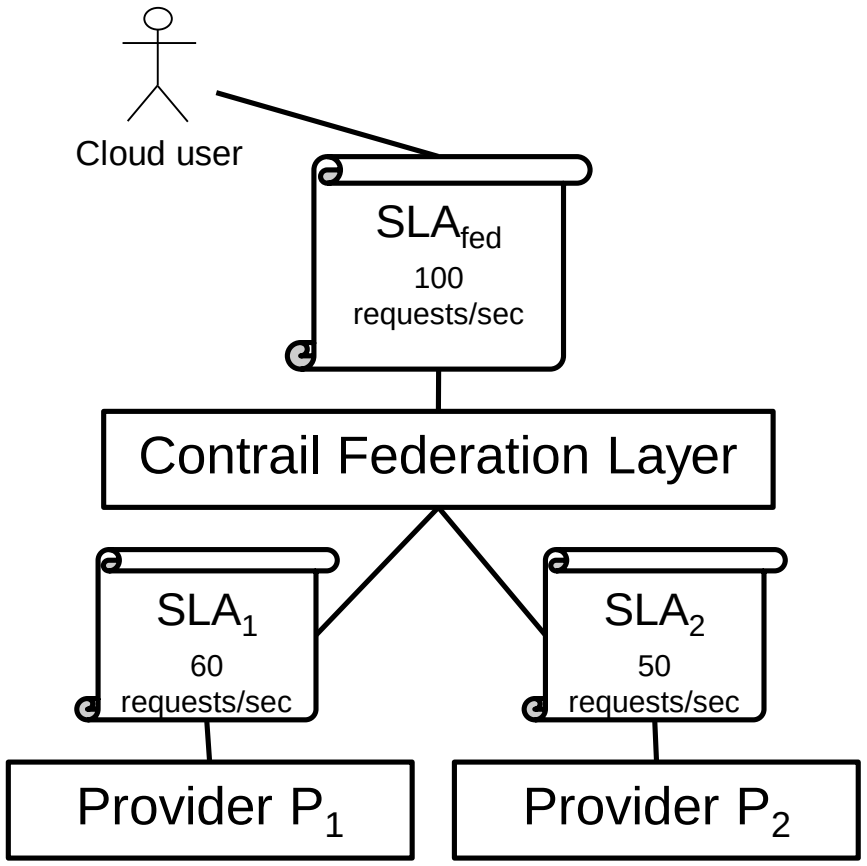
- Generation of several candidate splits, each with candidate providers
- Selection of the best split according to total price, QoS offered, reputation…

Resource-based SLA splitting



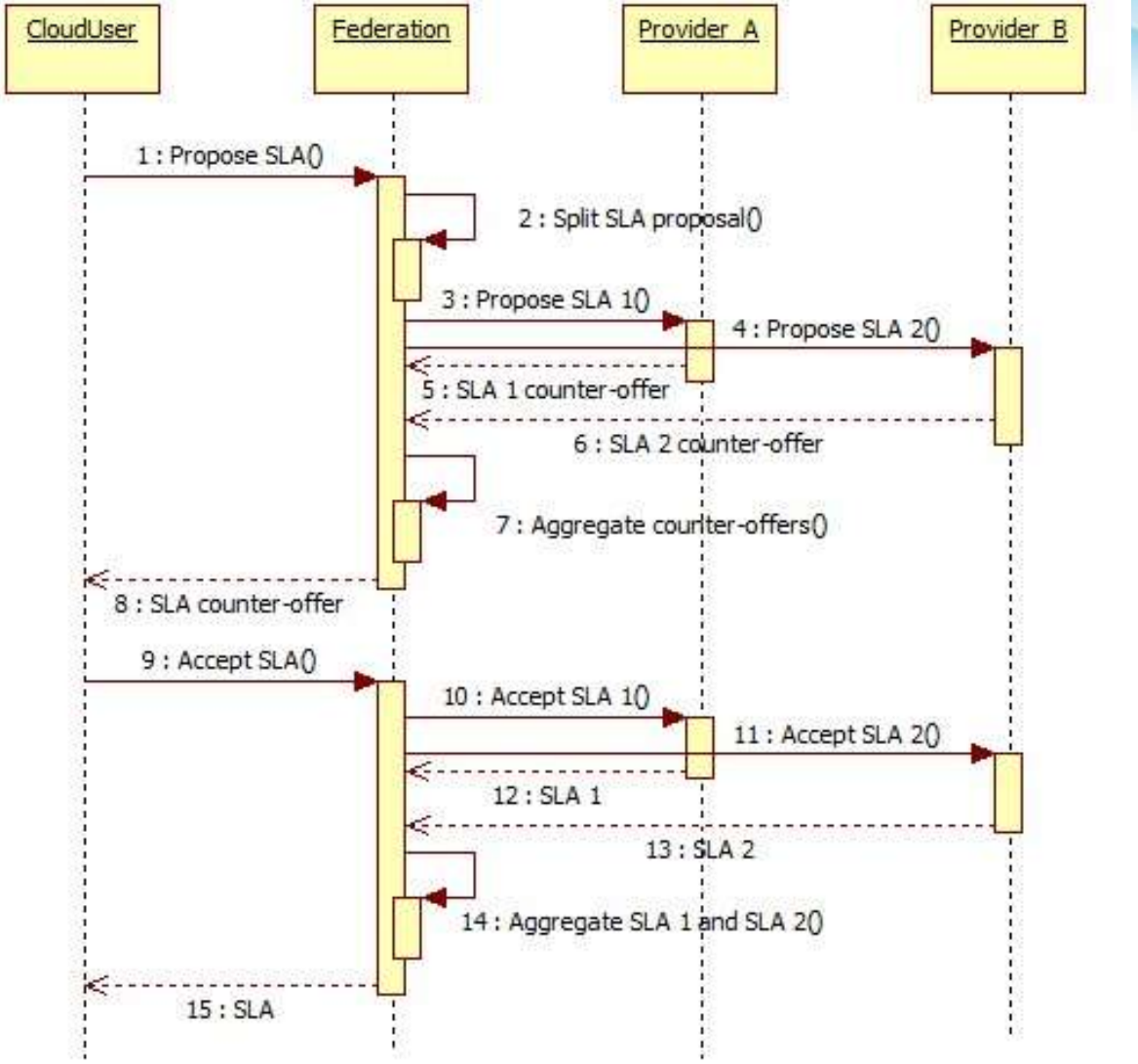
- Generation …
- Selection …
- Penalties cannot be split
- Federation, according to its risk management strategy, may ask more to providers to have some tolerance

Performance-based SLA splitting

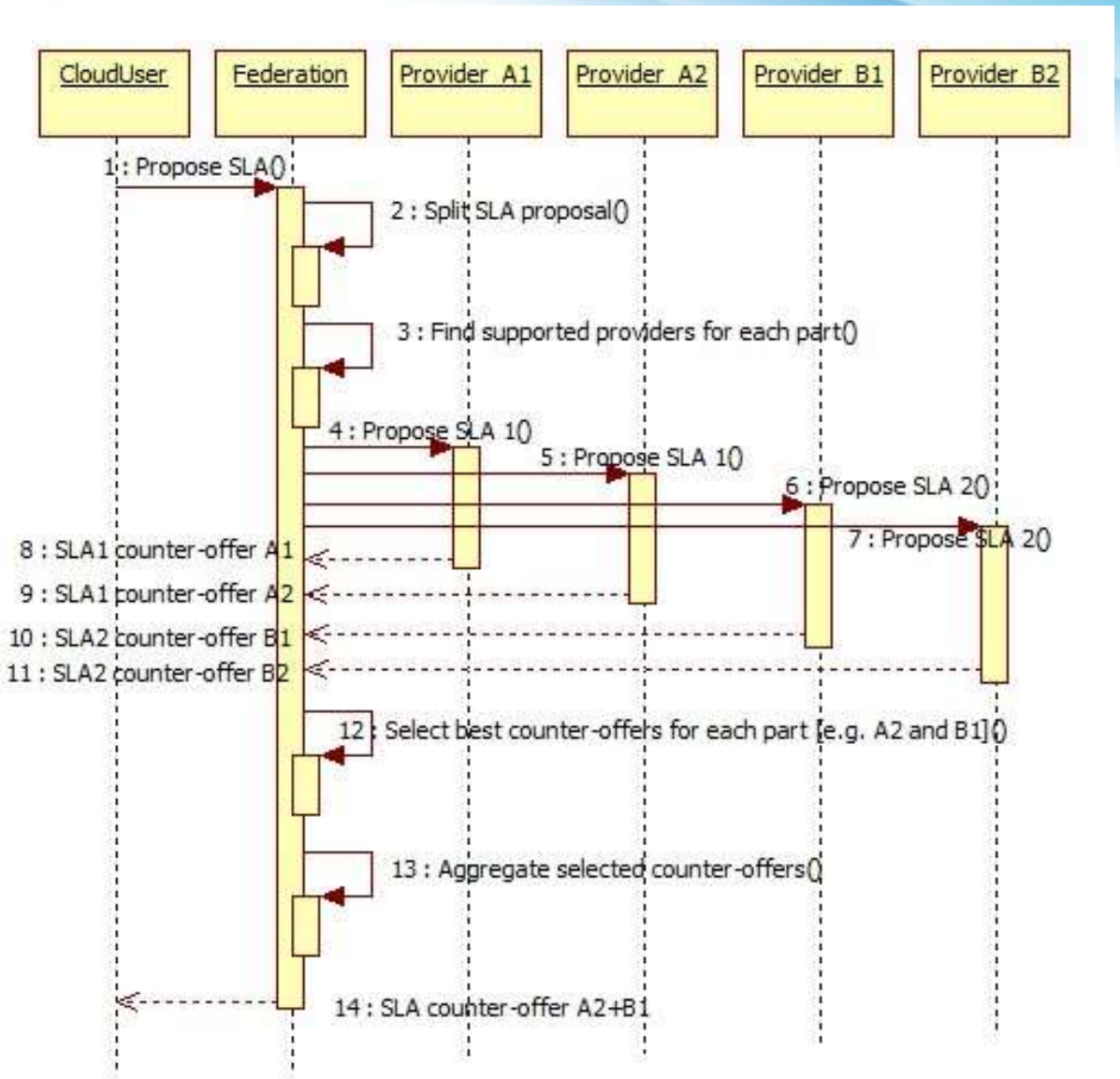


- Generation …
- Selection …
- Penalties…
- Tolerance…
- Common entry point with load balancing across providers
- More complex if Service not stateless

SLA splitting steps (without selection)



SLA splitting with selection



Cloud Platforms in Industry

Table 9.1 Some Example Cloud Computing Offerings

Vendor/Product	Service Type	Description
Amazon Web Services	IaaS, PaaS, SaaS	Amazon Web Services (AWS) is a collection of Web services that provides developers with compute, storage, and more advanced services. AWS is mostly popular for IaaS services and primarily for its elastic compute service EC2.
Google AppEngine	PaaS	Google AppEngine is a distributed and scalable runtime for developing scalable Web applications based on Java and Python runtime environments. These are enriched with access to services that simplify the development of applications in a scalable manner.
Microsoft Azure	PaaS	Microsoft Azure is a cloud operating system that provides services for developing scalable applications based on the proprietary Hyper-V virtualization technology and the .NET framework.
SalesForce.com and Force.com	SaaS, PaaS	SalesForce.com is a Software-as-a-Service solution that allows prototyping of CRM applications. It leverages the Force.com platform, which is made available for developing new components and capabilities for CRM applications.
Heroku	PaaS	Heroku is a scalable runtime environment for building applications based on Ruby.
RightScale	IaaS	Rightscale is a cloud management platform with a single dashboard to manage public and hybrid clouds.

Automated policy management in cloud

- Automated cloud policy management can be described as a **hands-free way to govern your cloud environment.** Although an automated cloud management solution does initially require some input, the description thereafter is remarkably accurate, as it's possible to automate the management of virtually any cloud governance policy.

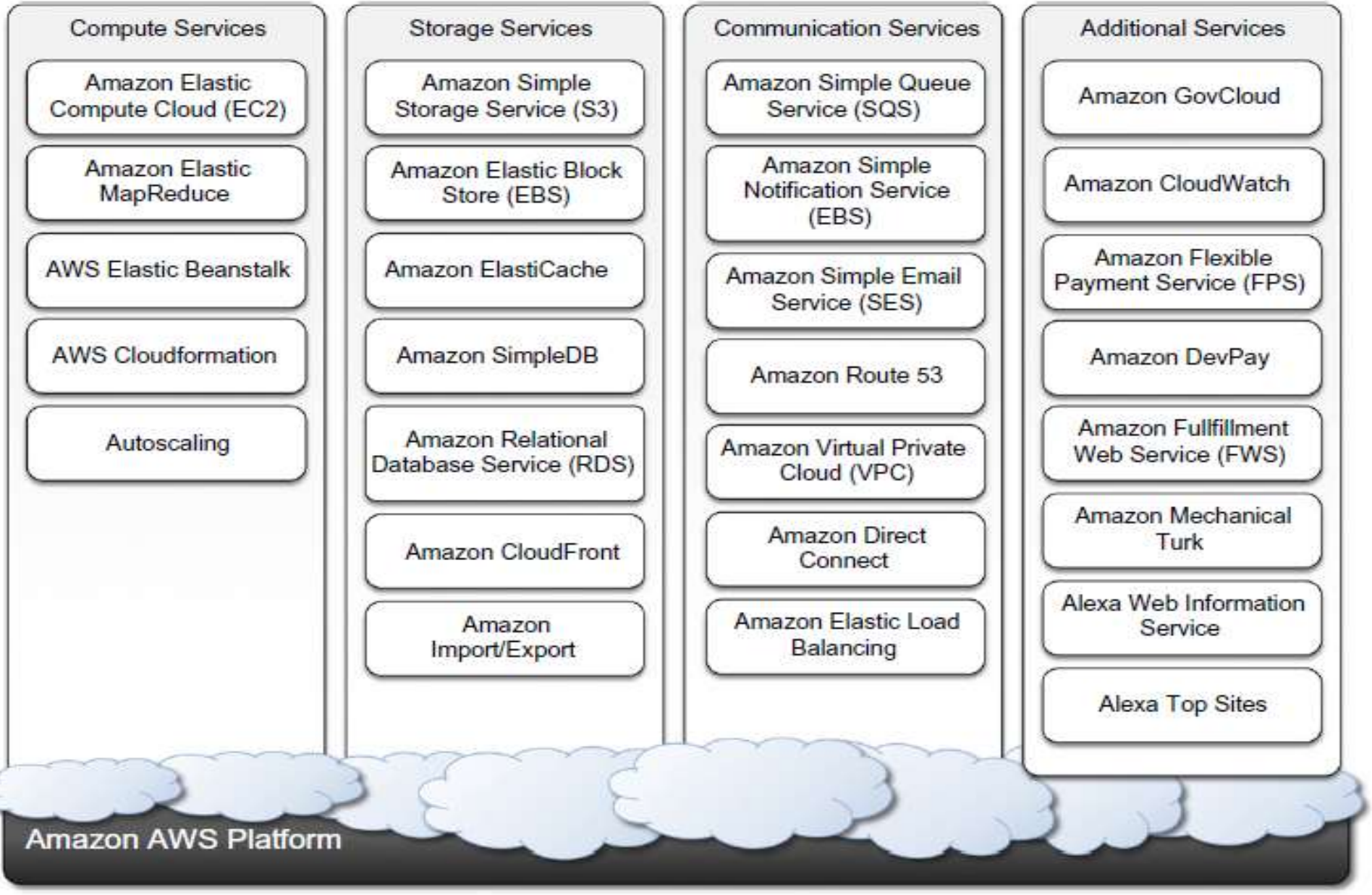
Cloud Computing Framework

- The Cloud Computing Governance Framework is a **subset of overall business governance** which includes IT and EA governance. It contains the unique characteristics from all types of governance that are essential to cloud computing governance.

Amazon web services (AWS)

- Amazon Web Services (AWS) provides elastic infrastructure scalability, messaging, and data storage.
- The platform is accessible through SOAP (Simple Object Access Protocol) or RESTful (Representational State Protocol)Web service interfaces and provides a Web-based console.
- Expenses computed on a pay-as-you-go basis
- Amazon Elastic Compute (EC2) and Amazon Simple Storage Service(S3).

Services available in the AWS ecosystem.



AWS: Compute services

- The fundamental service in this space is Amazon EC2, which delivers an IaaS solution.
- Amazon EC2 allows deploying servers in the form of virtual machines created as instances of a specific image.
- **Amazon machine images:**
  - Amazon Machine Images(AMIs) are templates from which it is possible to create a virtual machine.
  - They are stored in Amazon S3 and identified by a unique identifier in the form of **ami-xxxxxx.**
  - An AMI contains a physical file system layout with a predefined operating system installed.
  - Amazon Ram disk Image (ARI, id: ari-yyyyyy) and the Amazon Kernel Image (AKI, id: aki-zzzzzz),
  - Once an AMI is created, it is stored in an S3 bucket

AWS: EC2 instances

- EC2 instances represent **virtual machines, created using AMI** as templates, by selecting the number of cores, their computing power, and the installed memory.
- The processing power is expressed in terms of virtual cores and EC2 Compute Units (ECUs).
- One ECU is defined as giving the same performance as a 1.0-1.2 GHz 2007 Opteron or 2007 Xeon processor.

Available configurations for EC2 instances

Instance Type	ECU	Platform	Memory	Disk Storage	Price (U.S. East) (USD/hour)
Standard instances					
Small	1(1 × 1)	32 bit	1.7 GB	160 GB	$0.085 Linux $0.12 Windows
Large	4(2 × 2)	64 bit	7.5 GB	850 GB	$0.340 Linux $0.48 Windows
Extra Large	8(4 × 2)	64 bit	15 GB	1,690 GB	$0.680 Linux $0.96 Windows
Micro instances					
Micro	< = 2	32/64 bit	613 MB	EBS Only	$0.020 Linux $0.03 Windows
High-Memory instances					
Extra Large	6.5(2 × 3.25)	64 bit	17.1 GB	420 GB	$0.500 Linux $0.62 Windows
Double Extra Large	13(4 × 3.25)	64 bit	34.2 GB	850 GB	$1.000 Linux $1.24 Windows
Quadruple Extra Large	26(8 × 3.25)	64 bit	68.4 GB	1,690 GB	$2.000 Linux $2.48 Windows
High-CPU instances					
Medium	5(2 × 2.5)	32 bit	1.7 GB	350 GB	$0.170 Linux $0.29 Windows
Extra Large	20(8 × 2.5)	64 bit	7 GB	1,690 GB	$0.680 Linux $1.16 Windows
Cluster instances					
Quadruple Extra Large	33.5	64 bit	23 GB	1,690 GB	$1.600 Linux $1.98 Windows
Cluster GPU instances					
Quadruple Extra Large	33.5	64 bit	22 GB	1,690 GB	$2.100 Linux $2.60 Windows

AWS:EC2 environment

- The EC2 environment is in charge of **allocating addresses, attaching storage volumes, and configuring security** in terms of access control and network connectivity.
- instances are created with an internal IP address, which makes them capable of communicating within the EC2 network and accessing the Internet as clients.
- EC2 instances are also given a domain name that generally is in the form ***ec2-xxx-xxx-xxx.compute-x .amazonaws.com,***
- where xxx-xxx-xxx normally represents the four parts of the external IP address separated by a dash.
- compute-x gives information about the availability zone where instances are deployed.

AWS: EC2:Advanced compute services

- **AWS Cloud Formation:** Cloud Formation introduces the concepts of templates, which are formatted text files that **describe the resources needed to run an application.**
- **Services:** S3, SimpleDB, SQS, SNS, Route 53, Elastic Beanstalk
- **AWS elastic beanstalk:** AWS Elastic Beanstalk constitutes a simple and easy way to **package applications and deploy them on the AWS Cloud**.
- **Amazon elastic MapReduce :** cloud computing platform for MapReduce applications. It utilizes Hadoop as the MapReduce engine,

AWS: Storage services: Simple Storage Service(S3): S3 Key Concepts

- S3 has been designed to provide a simple storage service that's accessible through a Representational State Transfer (REST) interface.
- The storage is organized in a two-level hierarchy .(**Buckets, Objects)**
- Stored objects cannot be manipulated (renaming, modifying, or relocating) like standard files.
- Content is not immediately available to users .
- Requests will occasionally fail.

AWS:S3 Key Concepts: Resource naming

- Buckets, objects are represented by uniform resource identifiers(URIs) under the **s3.amazonaws.com** domain.
- **three different ways of addressing a bucket:**
  - Canonical form: http://s3.amazonaws.com/bukect_name/
  - Sub domain form: http://bucketname.s3.amazon.com/
  - Virtual hosting form: http://bucket-name.com/
- **Object ACL:**
  http://s3.amazonaws.com/bukect_name/object_name?acl
- **Bucket server logging:**
  http://s3.amzonaws.com/bucket_name?logging

AWS:S3 Key Concepts: Buckets

- A bucket is a container of objects.
- Buckets are top- level elements of the S3 storage architecture and do not support nesting. That is, it is not possible to create “subbuckets”.
- A bucket is located in a specific geographic location.
- Once a bucket is created, all the objects that belong to the bucket will be stored in the same availability zone of the bucket
- Users create a bucket by sending a PUT request to http://s3.amazonaws.com/ with the name of the bucket.
- The content of a bucket can be listed by sending a GET request specifying the name of the bucket.
- Once created, the bucket cannot be renamed or relocated.
- Deletion of a bucket is performed by a DELETE request .

AWS:S3 Key Concepts: Objects and metadata

- Objects constitute the content elements stored in S3.
- An object is identified by a name that needs to be unique within the bucket.
- The name cannot be longer than 1,024 bytes when encoded in UTF-8, and it allows almost any character.
- Users create an object via a PUT request that specifies the name of the object together with the bucket name, its contents.
- Maximum size of an object is 5 GB.
- it cannot be modified, renamed, or moved into another bucket.
- Deleting an object is performed via a DELETE request.
- Objects can be tagged with metadata.

AWS:S3 Key Concepts: Access control and security

- Amazon S3 allows controlling the access to buckets and objects by means of Access Control Policies(ACPs).
- An ACP is a set of grant permissions are attached to a resource expressed by means of an XML configuration file.
- A policy allows defining up to 100 access rules.
- READ allows the grantee to retrieve an object.
- WRITE allows the grantee to add an object to a bucket .
- READ_ACP allows the grantee to read the ACP of a resource.
- WRITE_ACP allows the grantee to modify the ACP of a resource.
- FULL_CONTROL grants all of the preceding permissions

AWS:S3 Key Concepts: Amazon elastic block store

- The Amazon Elastic Block Store (EBS) allows AWS users to provide EC2 instances with persistent storage.
- They accommodate up to 1 TB of space.
- Currently, Amazon charges $0.10/GB/month of allocated storage and $0.10 per 1 million requests made to the volume.

AWS:S3 Key Concepts: Amazon ElastiCache

- ElastiCache is an implementation of an elastic in-memory cache based on a cluster of EC2.
- It provides fast data access from other EC2 instances through a Memcached-compatible protocol.

AWS:S3 Key Concepts: Structured storage solutions

- RDBMS have been the common data back-end for a wide range of applications.
- Amazon provides applications with structured storage: **Amazon Relational Data Storage(RDS), and Amazon SimpleDB.**

Table 9.5 Amazon SimpleDB Data Transfer Charges, 2011–2012

Instance Type	Price (U.S. East) (USD)
Data Transfer In	
All data transfer in	$0.000
Data Transfer Out	
1st GB/month	$0.000
Up to 10 TB/month	$0.120
Next 40 TB/month	$0.090
Next 100 TB/month	$0.070
Next 350 TB/month	$0.050
Next 524 TB/month	Special arrangements
Next 4 PB/month	Special arrangements
Greater than 5 PB/month	Special arrangements

AWS:S3 Key Concepts: Amazon CloudFront

- CloudFront is an implementation of a **content delivery network.**
- AWS provides users with simple Web service APIs to manage CloudFront.
- DNS domain under the Cloudfront.net domain name(i.e.,my-distribution.Cloudfront.net).
- The content that can be delivered through CloudFront is static (HTTP and HTTPS) or streaming (Real Time Messaging Protocol, or RMTP).

AWS: Communication services

- Amazon provides the communication among existing applications and services residing within the AWS infrastructure.
- Two major categories: virtual networking and messaging.

AWS: Communication services: Virtual networking

- comprises a collection of services that allow AWS users to **control the connectivity to and between compute and storage services.**
- Amazon *Virtual Private Cloud(VPC)* and Amazon *DirectConnect* provide connectivity solutions.
- **VPC:** Prepared templates include **public subnets, isolated networks, private networks** accessing Internet through network address translation (NAT), and hybrid networks.
- Amazon Direct Connect allows AWS users to **create dedicated networks b**etween the user private network and Amazon Direct Connect locations, called ports.
- Amazon Route 53 implements **dynamic DNS** services that allow *AWS resources to be reached through domain names different from the amazon.com domain.*

AWS: Communication services: Messaging

- The three different types of messaging services offered are **Amazon Simple Queue Service (SQS), Amazon Simple Notification Service(SNS), and Amazon Simple Email Service(SES).**
- **SQS:** exchanging messages between applications by means of **message queues**, hosted within the AWS infrastructure.
- **Amazon SNS** provides a **publish-subscribe** method for connecting heterogeneous applications.
- **Amazon SES** provides AWS users with a scalable email service.

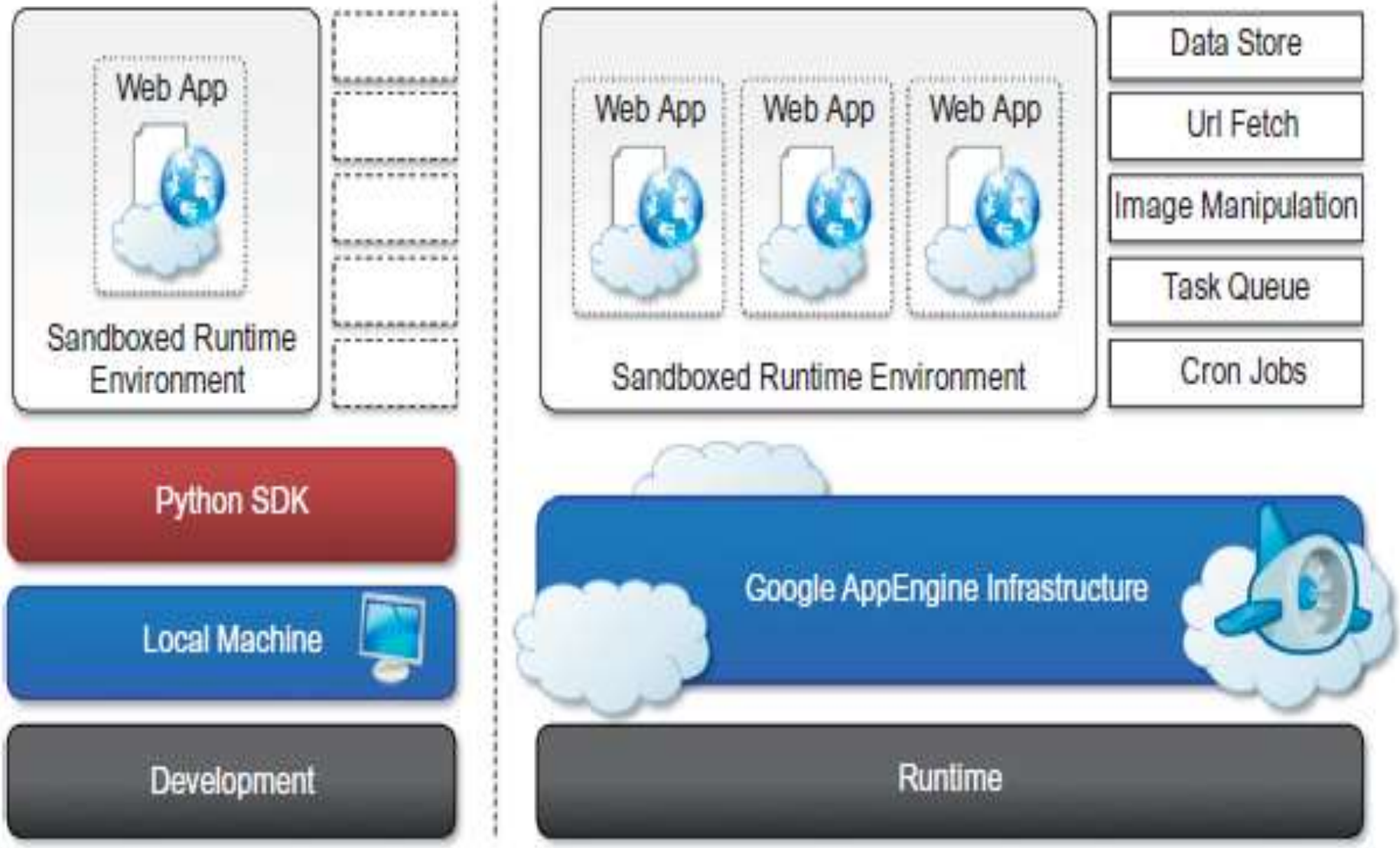
Aneka framework

- Aneka is an **Application Platform-as-a-Service (Aneka PaaS) for Cloud Computing.** It acts as a framework for building customized applications and deploying them on either public or private Clouds. One of the key features of Aneka is its support for provisioning resources on different public Cloud providers such as Amazon EC2,

Google AppEngine

- Google AppEngine is a PaaS implementation that provides services for developing and hosting scalable Web applications.

Google AppEngine: Architecture



Google AppEngine: Runtime environment

- The runtime environment represents the execution context of applications hosted on AppEngine.
- **Supported runtimes :** Java, Python, and Go.
- Java ServerPages(JSP),
- Python 2.5.2 interpreter
- Python Web application framework, called **webapp**, simplifying the development of Web applications.
- Go that is supported by AppEngine is r58.1.

Google AppEngine: Storage

- AppEngine provides in memory-cache, storage for semistructured data, and long-term storage for static data.
- **Static file servers:** Static data of web applications can be hosted on static file servers, since they are not frequently modified.
- **DataStore** : DataStore is a service that allows developers to **store semistructured data.**
- DataStore can be considered as a large object database in which to store objects that can be retrieved by a specified key.

Google AppEngine: Application services

- **UrlFetch :** provide developers with the capability of **retrieving a remote resource through HTTP/HTTPS .**
- **MemCache:** optimized for **fast access** and provides developers with a **volatile store** for the objects that are frequently accessed.
- **Mail and instant messaging:** AppEngine provides developers with the ability **to send and receive mails through Mail**. The service allows sending email on behalf of the application to specific user accounts.
- AppEngine provides also another way to communicate with the external world: the **Extensible Messaging and Presence Protocol (XMPP) ,**Google Talk **.**
- **Account management :** allowing developers to leverage Google account management by means of Google Accounts.
- **Image manipulation :** AppEngine allows applications to perform **image resizing, rotation, mirroring, and enhancement** by means of Image Manipulation.

Google AppEngine: Compute services

- Task queues:
  - Task Queues allow applications to submit a task for a later execution.
  - The service allows users to have up to 10 queues that can execute tasks at a configurable rate.
- Cron jobs
  - It is possible to *schedule the required operation at the desired time* by using the CronJobs service.

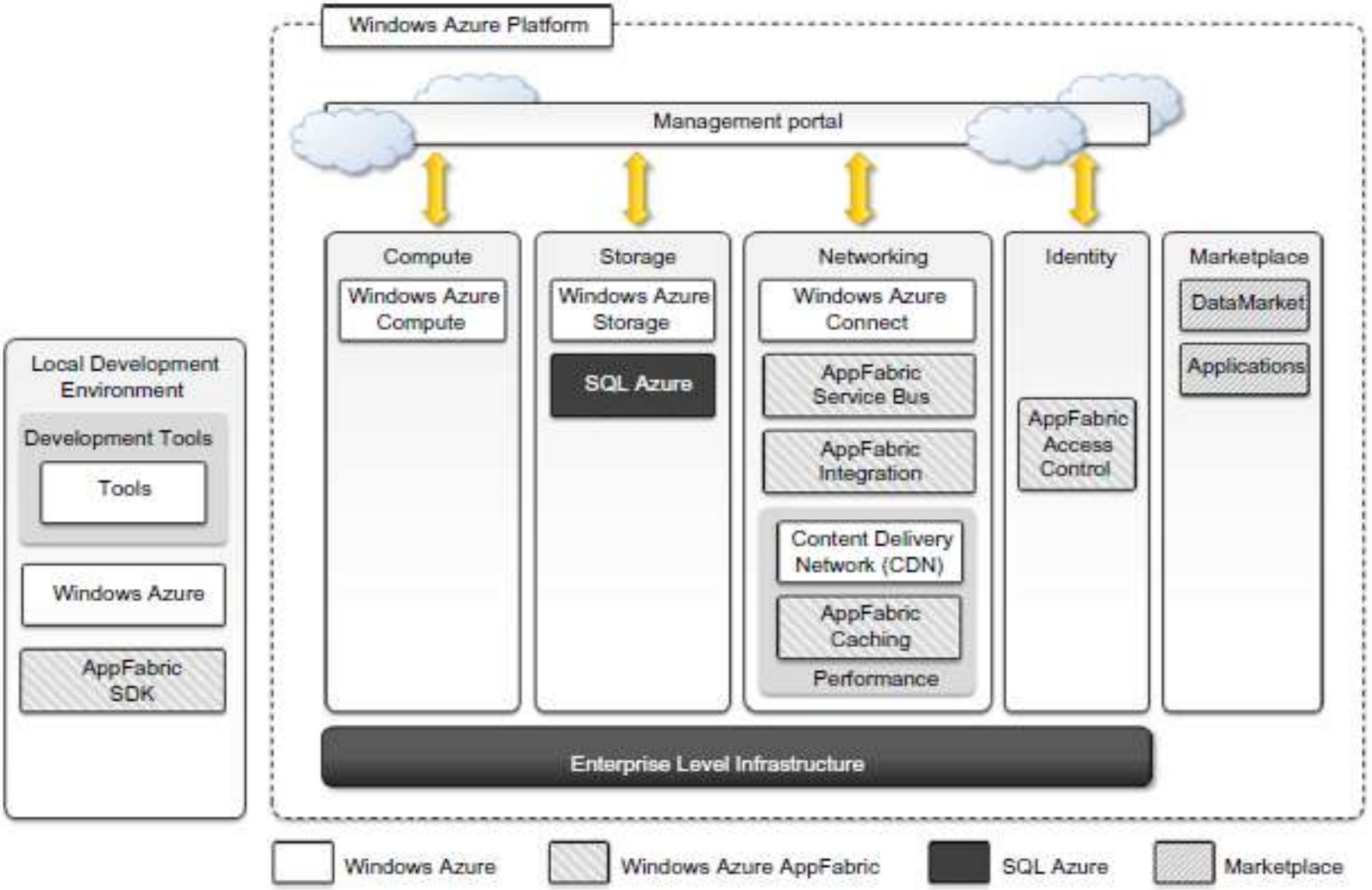
Google AppEngine: Application life cycle

- AppEngine provides support for almost all the phases characterizing the life cycle of an application: **testing and development, deployment, and monitoring.**
- Java SDK , Google Web Toolkit, and Google AppEngine plug-ins into Eclipse, servlet
- Python SDK:
  - The Python SDK allows developing Web applications for AppEngine with Python 2.5.
  - It provides a stand alone tool,called **GoogleApp EngineLauncher**, for managing Web applications locally and deploying them to AppEngine.

Microsoft Azure

- Microsoft Windows Azure is a cloud operating system built on top of Microsoft datacenters' infrastructure .
- **Services** range from **compute, storage, and networking to application connectivity, access control, and business intelligence.**
- integrates the scalability features into the common Microsoft technologies such as **Microsoft Windows Server 2008, SQL Server, and ASP.NET.**

Azure core concepts



Microsoft Azure :Compute services

- Web role, Worker role, Virtual Machine (VM) role.
- **Web role:**
  - The Web role is designed to **implement scalable Web applications.**
  - They are hosted on the IIS 7 Web Server.
  - the .NET technology natively supports Web roles; developers can directly develop their applications in Visual Studio, test them locally, and upload to Azure.
  - It is possible to develop **ASP.NET.**
  - IIS 7 also supports the PHP runtime environment by means of the **FastCGI** module.

Microsoft Azure :Compute services

- **Worker role:**
  - Worker roles are **designed to host general compute services on Azure.**
  - They can be used to **quickly provide compute power or to host services .**
  - the .NET technology provides complete support for Worker role
  - For example, Worker roles can be used to **host Tomcat and serve JSP-based applications.**

Microsoft Azure :Compute services

- **Virtual Machine role:**
  - The Virtual Machine role is based on the Windows Hyper-V virtualization technology.

Microsoft Azure :Storage services

- **Blobs:**
  - Azure allows storing large amount of data in the form of **binary large objects(BLOBs)** by means of the blobs service.
  - This service is optimal **to store large text or binary files.**
  - **Block blobs.** Block blobs are **composed of blocks** and are optimized for **sequential access;** therefore they are appropriate for **media streaming.** Currently, blocks are of 4 MB, and a single block blob can reach 200 GB in dimension.
  - **Page blobs.** Page blobs are made of pages that are identified by an offset from the beginning of the blob.
  - This type of blob is optimized for **random access**
  - Currently, the maximum dimension of a page blob can be 1 TB.

Microsoft Azure :Storage services

- **Azure drive:** Page blobs can be used to store an **entire file system in the form of a single Virtual Hard Drive (VHD) file.**
- **Tables:** Tables constitute a semi structured storage solution, allowing users to store information in the form of entities with a collection of properties.
- Currently, a table can contain up to 100 TB of data, and rows can have up to 255 properties, with a maximum of 1 MB for each row.

Microsoft Azure :Storage services

- **Queues :**Queue storage allows **applications to communicate by exchanging messages** through durable queues.
- Applications enter messages into a queue, and other applications can read them in a first-in, first-out (FIFO) style.

Microsoft Azure: AppFabric

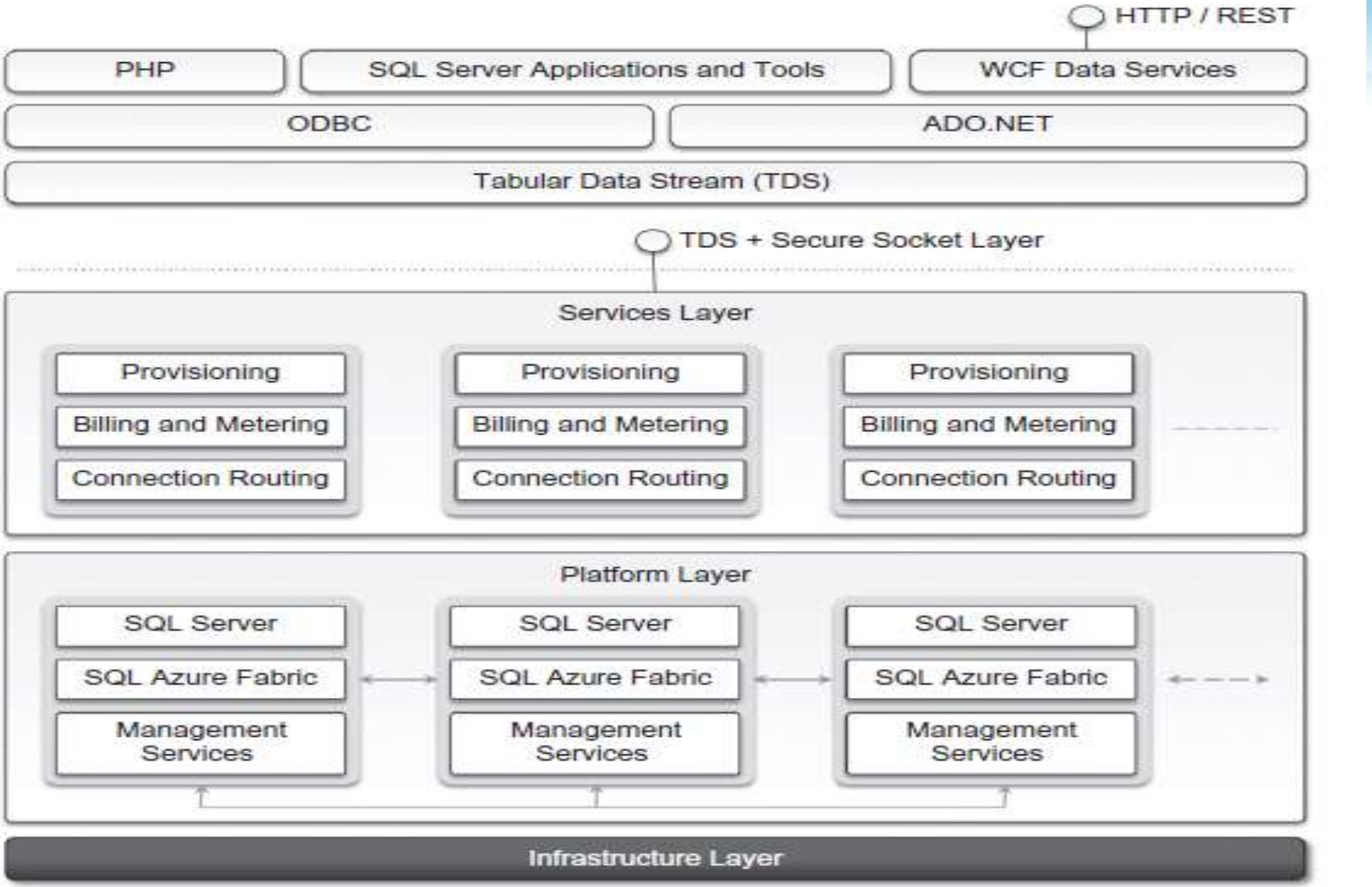
- AppFabric is a comprehensive middleware for **developing, deploying, and managing applica- tions on the cloud**.
- AppFabric implements an optimized infrastructure supporting scaling out and high availability;
- it also provides **communication, authentication and authorization, and data access**.

Windows Azure virtual network

- Networking services for applications are offered under the name Windows Azure Virtual Network, which includes **Windows Azure Connect and Windows Azure Traffic Manager.**
- Windows Azure Connect allows easy setup of IP-based network connectivity.
- Windows Azure Traffic Manager provides l**oad-balancing** features for services listening to the HTTP or HTTPS ports and hosted on multiple roles.

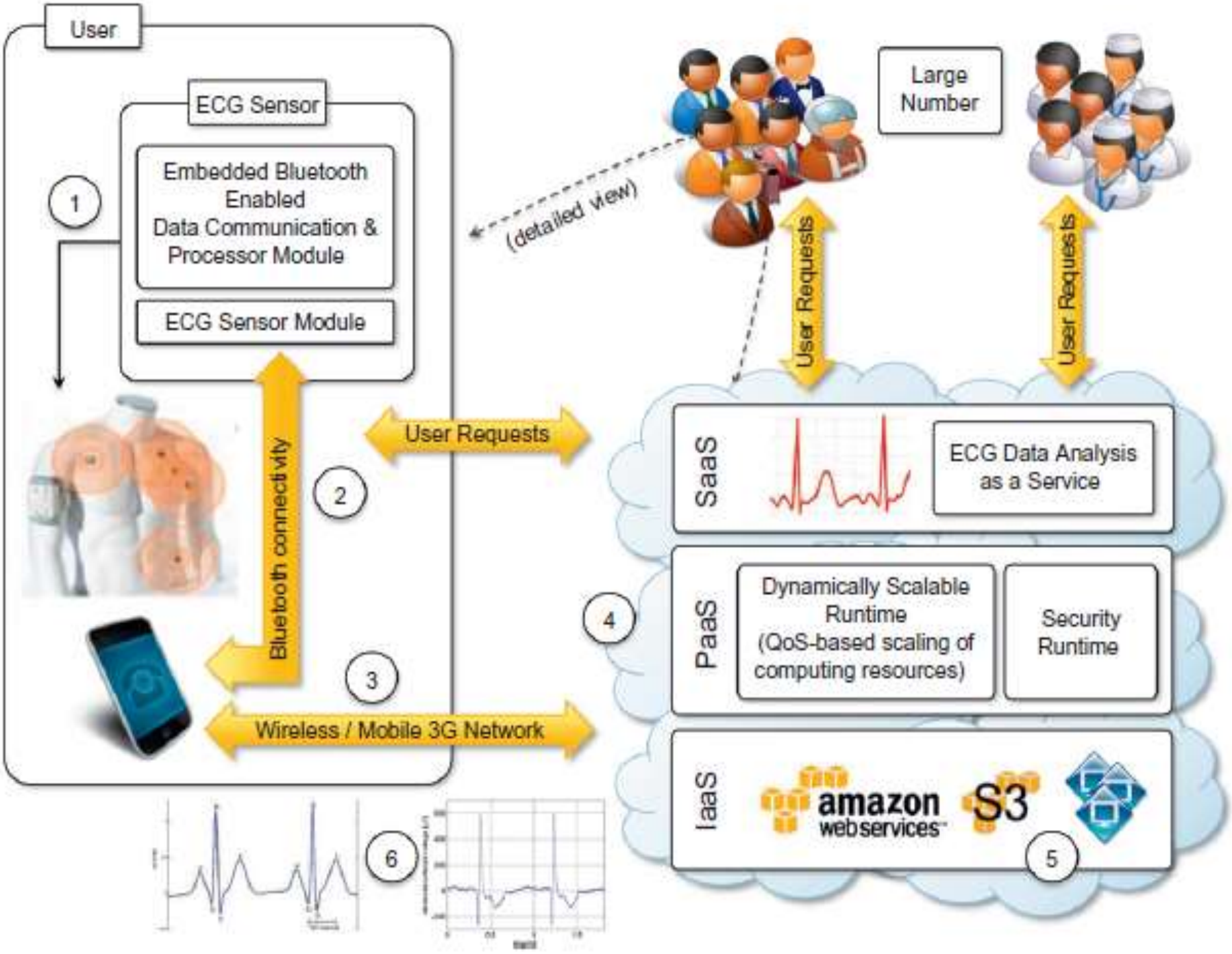
SQL Azure

- SQL Azure is a relational database service hosted on Windows Azure and built on the SQL Server technologies.



Cloud Applications: Healthcare: ECG analysis in the cloud

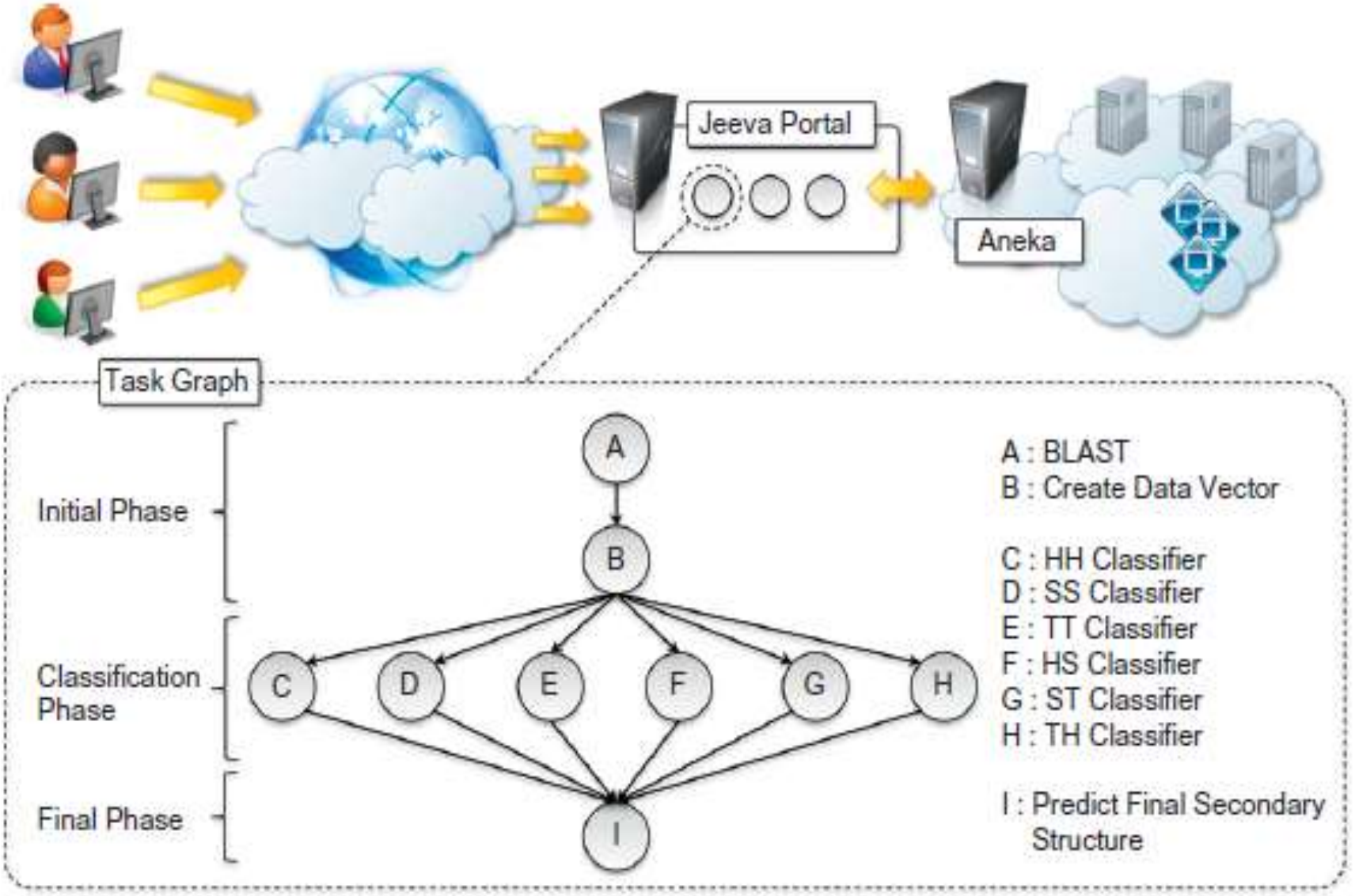
- ECG activity produces a specific waveform that is repeated over time and that represents the heartbeat.
- Cloud computing technologies allow the **remote monitoring of a patient's heartbeat data, data analysis** in minimal time.
- This way a patient at risk can be constantly monitored without going to a hospital for ECG analysis.


User
ECG Sensor
Embedded Bluetooth Enabled Data Communication & Processor Module
ECG Sensor Module
1
2
3
4
5
6
Bluetooth connectivity
User Requests
Wireless / Mobile 3G Network
(detailed view)
Large Number
User Requests
User Requests
SaaS
ECG Data Analysis as a Service
PaaS
Dynamically Scalable Runtime (QoS-based scaling of computing resources)
Security Runtime
IaaS
amazon web services
S3

Biology: protein structure prediction

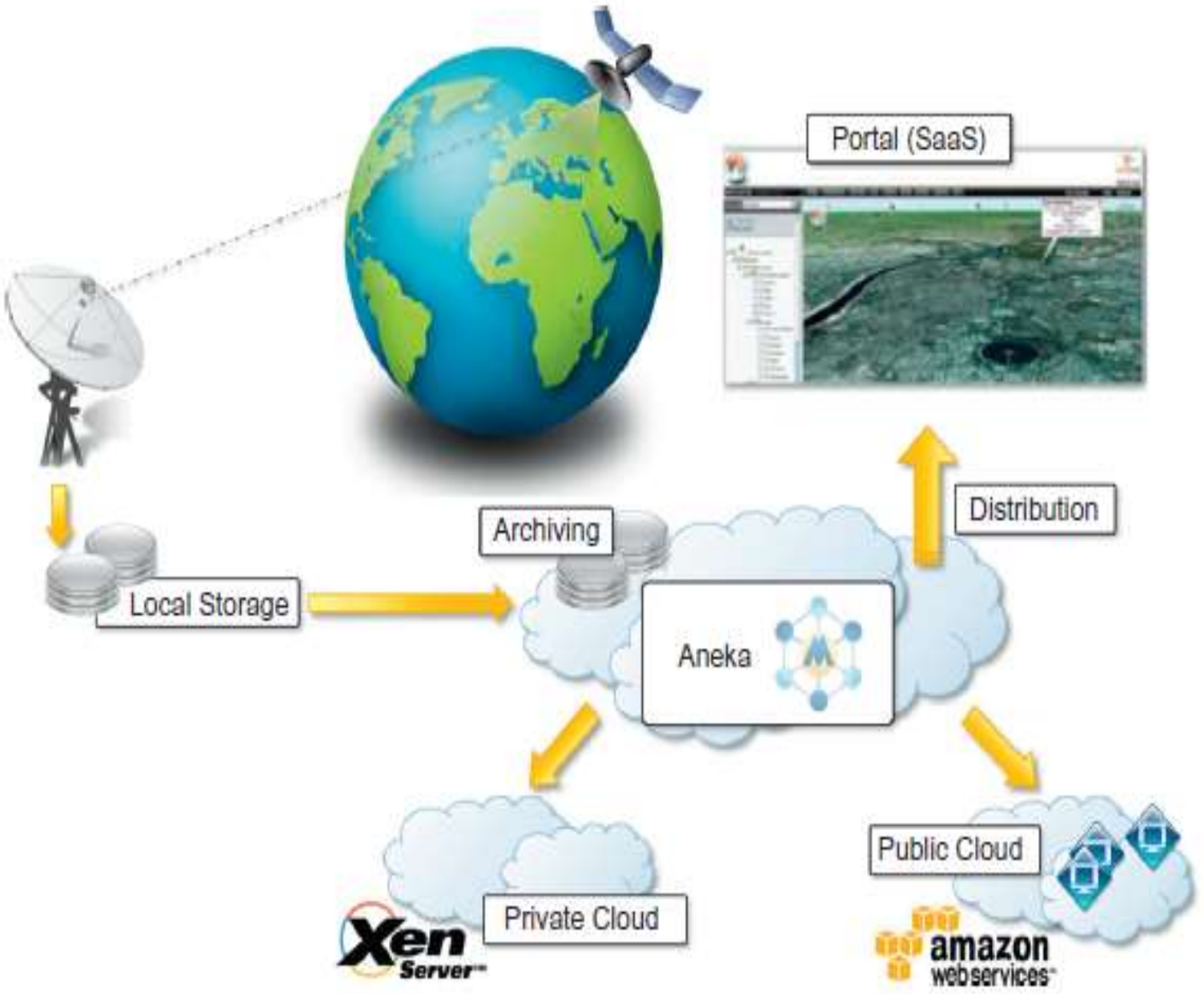
- Protein structure prediction is a computationally intensive task for the **design of new drugs for the treatment of diseases.**
- The **computational power required** for protein structure prediction can now be acquired on demand, without owning a **cluster, parallel and distributed computing facilities.**

Protein structure prediction: JEEVA



Geoscience: satellite image processing

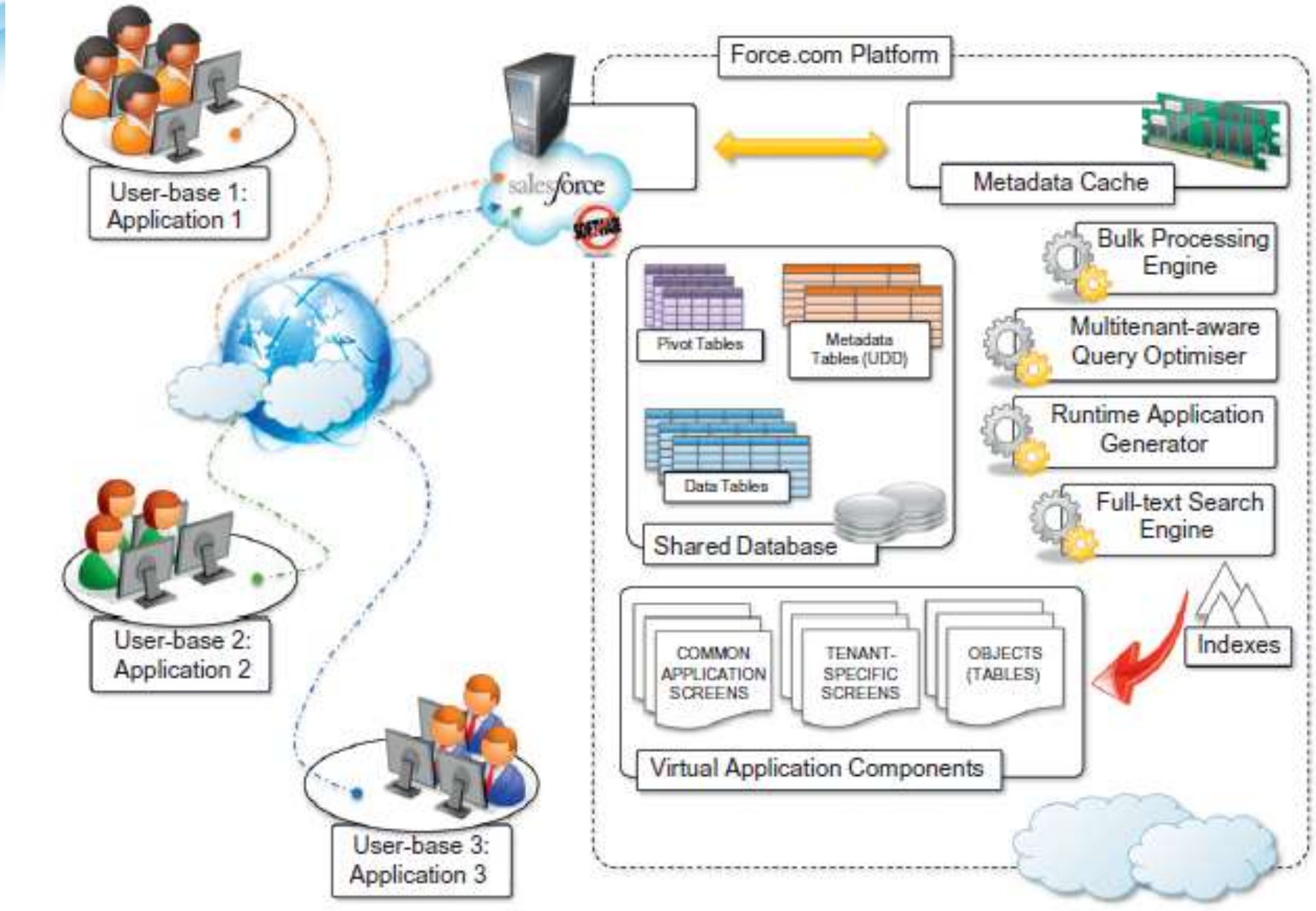
- Geoscience applications collect, produce, and analyze massive amounts of geospatial and non-spatial data.
- **Satellite remote sensing** generates hundreds of gigabytes of raw images that need to be further processed to become the basis of several different GIS products.
- Large images need to be moved from a ground station's local storage to compute facilities, where several transformations and corrections are applied.
- Cloud computing provides the appropriate infrastructure to support such application scenarios.


Portal (SaaS)
Distribution
Archiving
Local Storage
Aneka
Private Cloud
Public Cloud
Xen Server
amazon web services

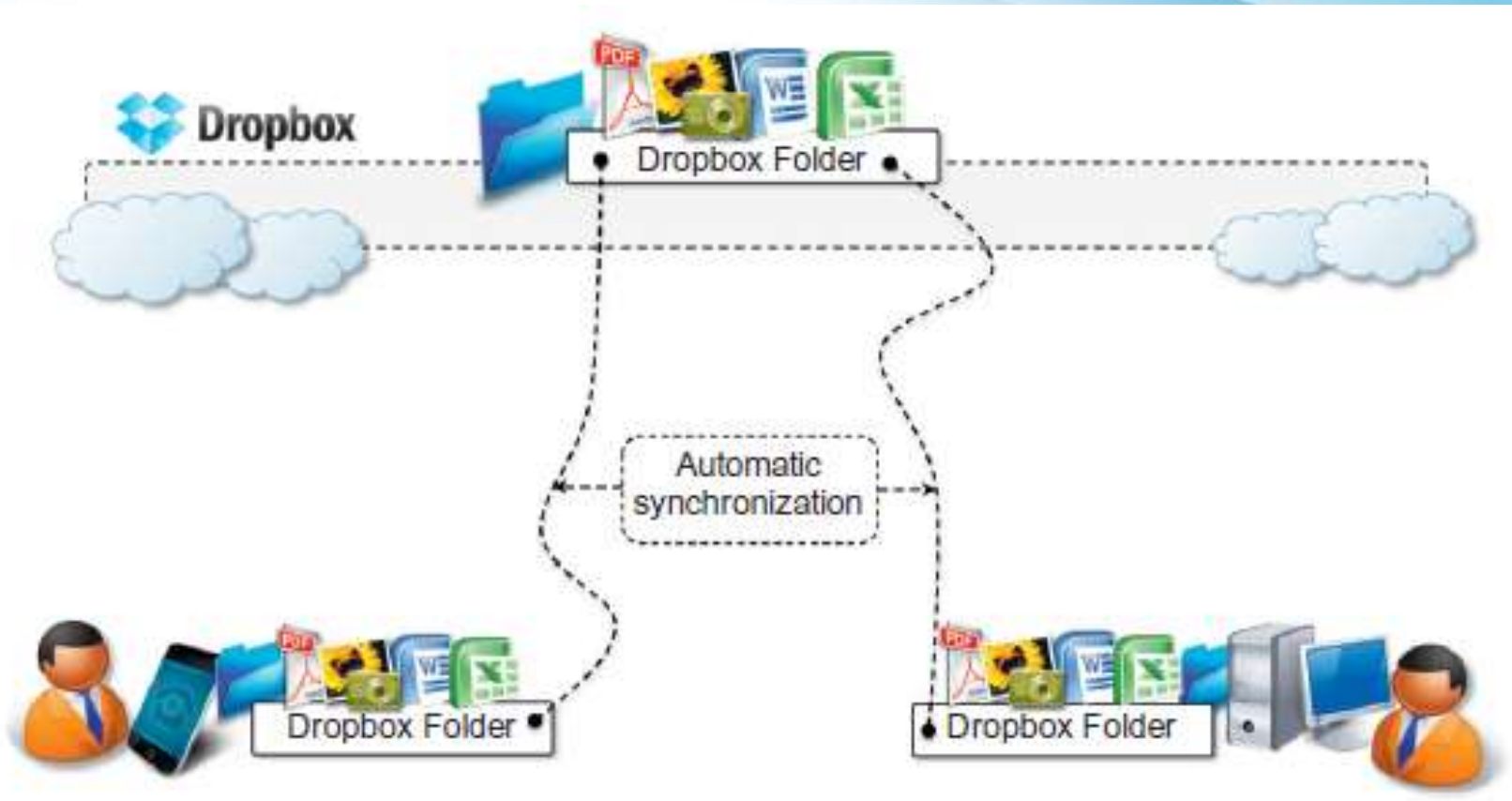
Business and consumer applications :

- Customer Relationship Management(CRM)
- Enterprise Resource Planning(ERP): finance and accounting, human resources, manufacturing, supply chain management, project management

Salesforce.com



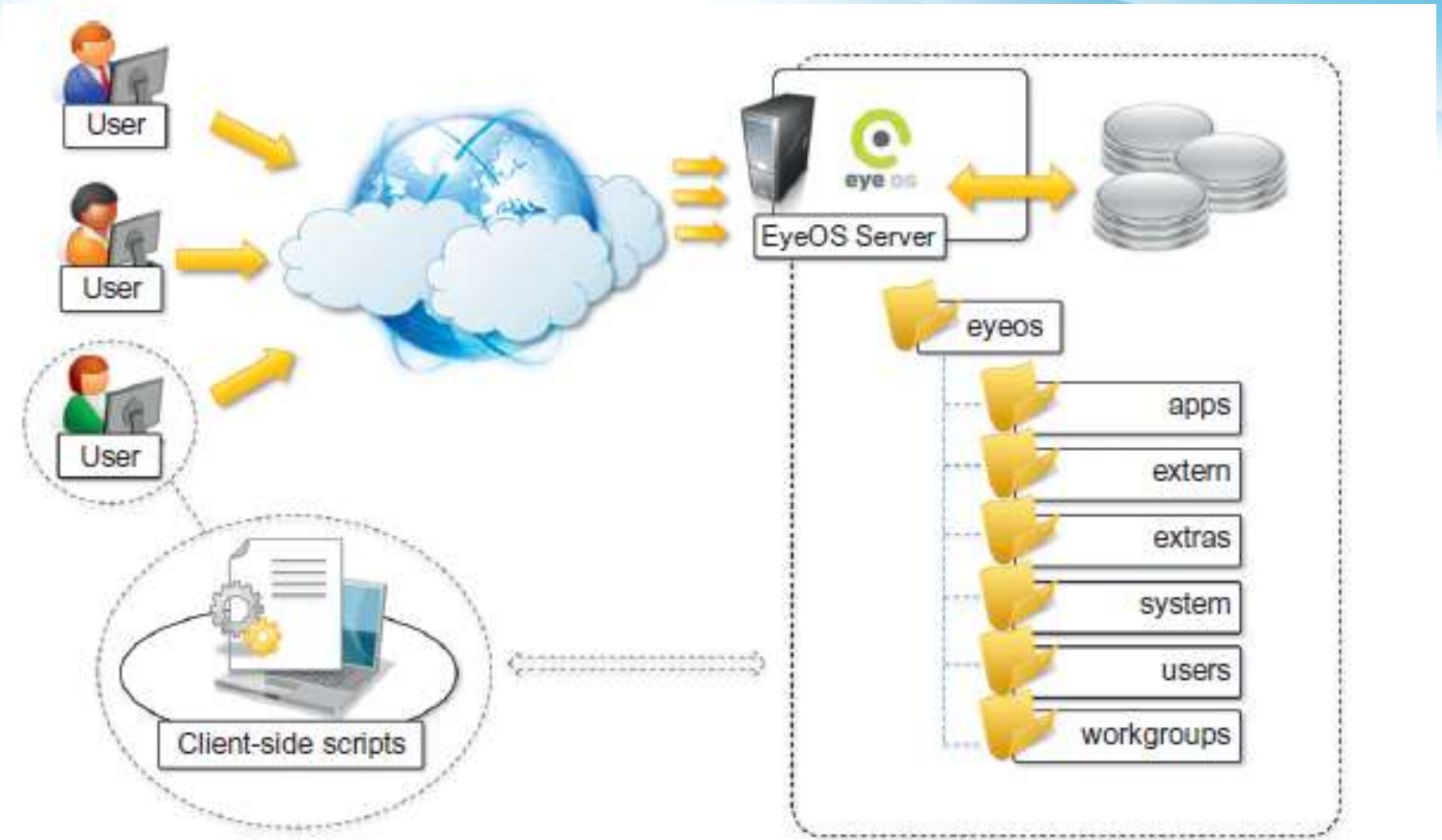
Dropbox and iCloud



Google docs

- GoogleDocs is a SaaS application that delivers the **basic office automation** capabilities with support for collaborative editing over the Web.
- Google Docs allows users to **create and edit text documents, spreadsheets, presentations, forms, and drawings.**

Cloud desktops: EyeOS and XIOS/3



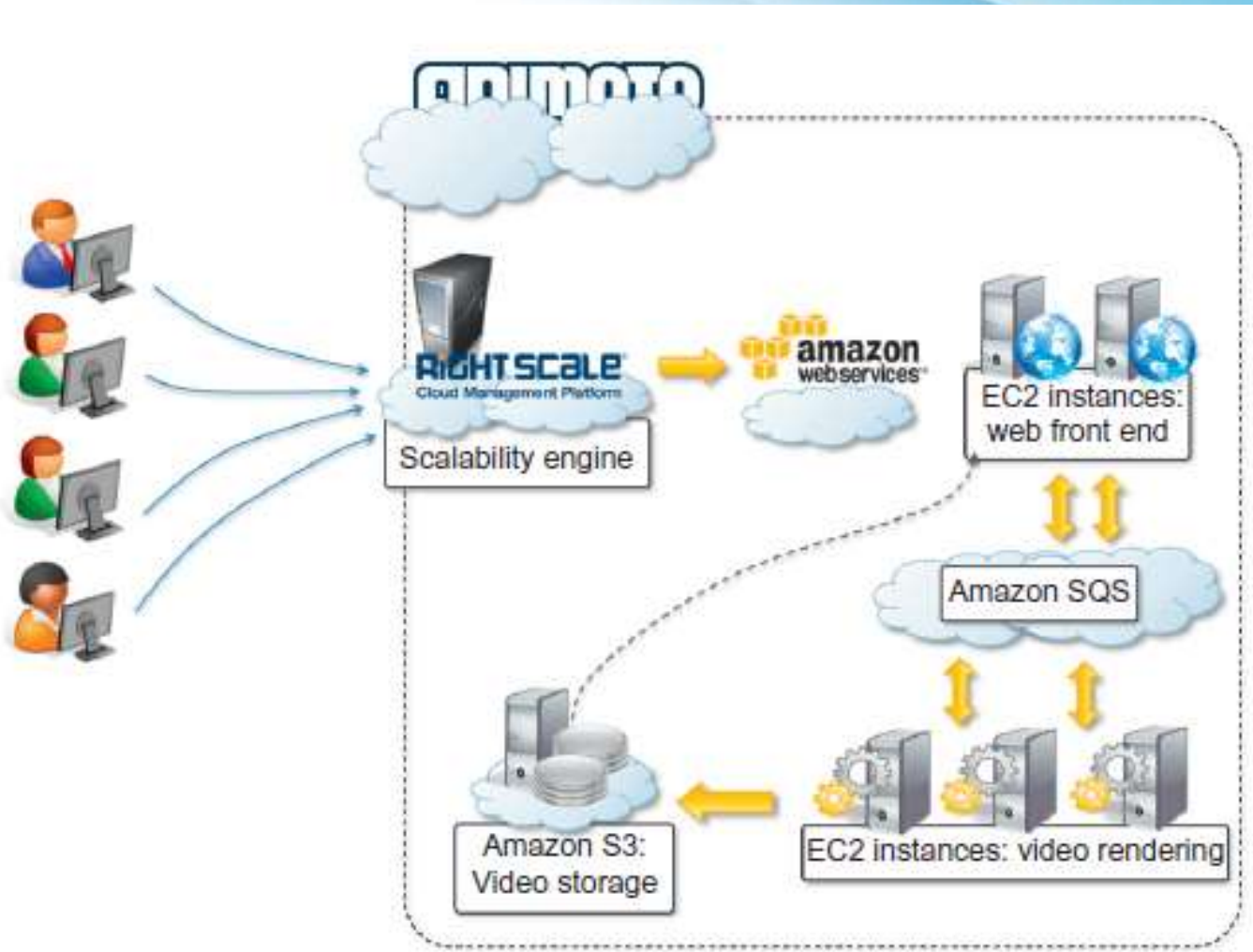
Social networking : Facebook

- Facebook is based on LAMP (Linux, Apache, MySQL, and PHP).

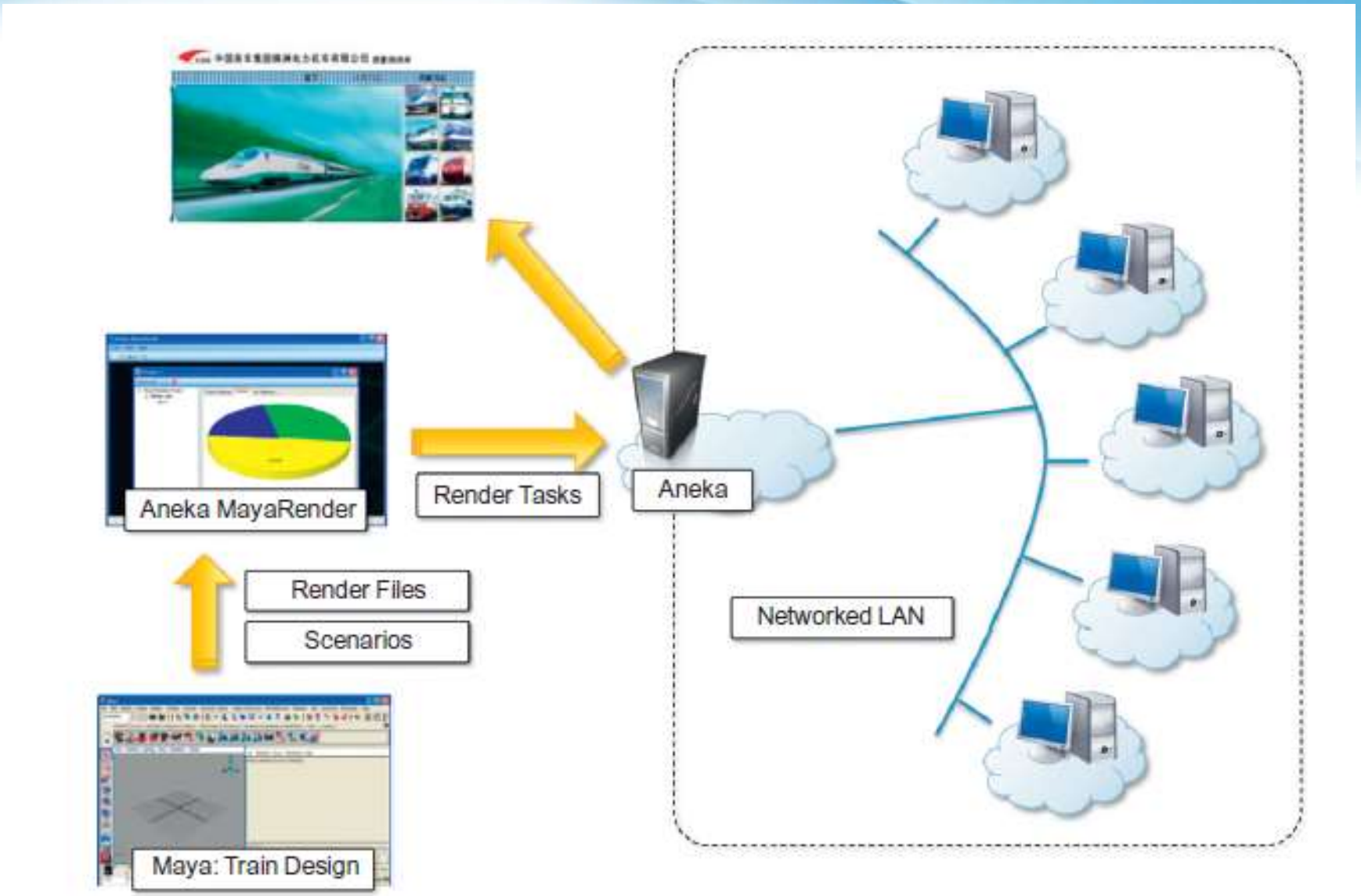
Media applications

- **Animoto:** The Website provides users with a very straightforward interface for quickly creating videos out of images, music, and video fragments submitted by users.
- A proprietary artificial intelligence (AI) engine, which **selects the animation and transition effects according to pictures and music**, drives the rendering operation.

Animoto reference architecture.



Maya rendering with Aneka



Eucalyptus cloud computing platforms

- Eucalyptus is an **open source software platform for implementing Infrastructure as a Service (IaaS)** in a private or hybrid cloud computing environment. The Eucalyptus cloud platform pools together existing virtualized infrastructure to create cloud resources for infrastructure as a service, network as a service and storage as a service.
- Eucalyptus implements infrastructure as a service (IaaS) methodology for solutions in private and hybrid clouds. Eucalyptus provides a platform for a single interface so that users can calculate the resources available in private clouds and the resources available externally in public cloud services.

IBM Bluemix

- IBM Bluemix, rebranded IBM Cloud in 2017, is a **cloud Platform as a service** (PaaS) developed by IBM. It supports several programming languages and services as well as integrated DevOps to build, run, deploy and manage applications on the cloud. Bluemix is based on Cloud Foundry open technology and runs on SoftLayer infrastructure.
- IBM Bluemix, rebranded IBM Cloud in 2017, is a cloud Platform as a service (PaaS) developed by IBM. It supports several programming languages and services as well as integrated DevOps to build, run, deploy and manage applications on the cloud. Bluemix is based on Cloud Foundry open technology and runs on SoftLayer infrastructure. Bluemix supports several programming languages including Java, Node.js, Go, PHP, Swift, Python, Ruby Sinatra, Ruby on Rails and can be extended to support other languages such as Scala through the use of buildpacks.

Thank you

The Content in this Material are from the Textbooks and Reference books given in the Syllabus