Usage Control for GRID Systems

Fabio Martinelli

Joint work with L. Krautsevich, A. Lazouski, P. Mori, M. Petrocchi, A. Yautshiukin

Institute of Informatics and Telematics
National Research Council of Italy
(IIT-CNR)
Pisa - Italy

Outline

- Setting the scenario
  - GRID
  - Access control
- Usage Control (UCON)
- UCON and trust
- UCON and risk
- Conclusion

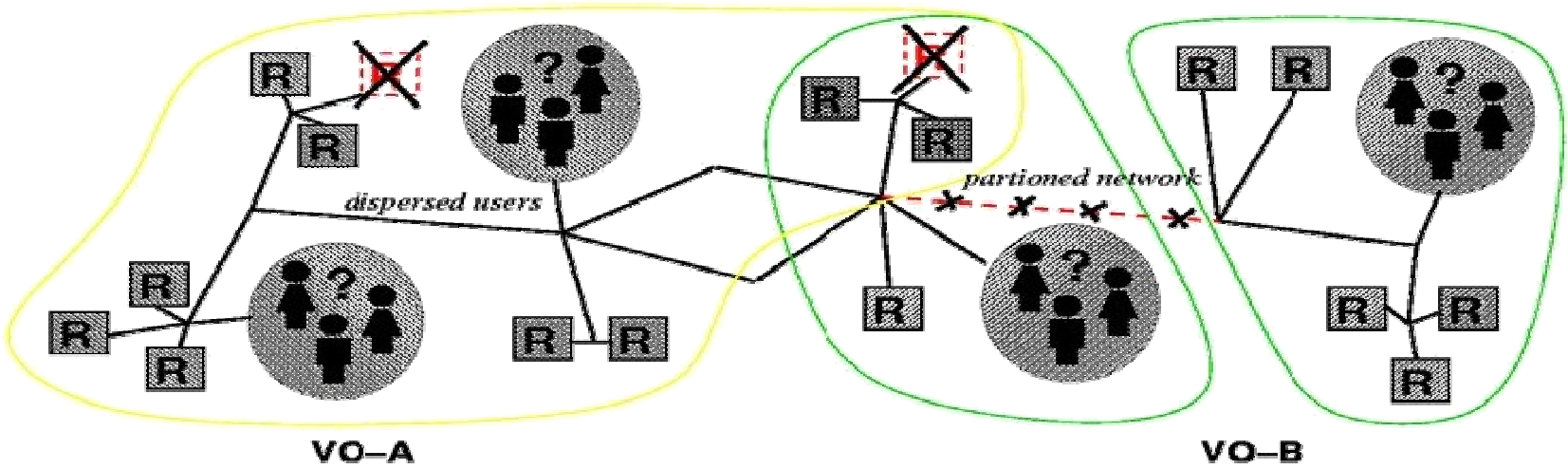
GRID

Grid Computing

- Dynamic collection of diverse and distributed individuals who share heterogeneous resources (services) in a coordinated fashion
  - Computers
  - Data Bases
  - Storage
  - ……
- User and resource/service providers might be unknown each other
  - An advanced security support is required to:
    - Protect resources from code provided by users
    - Protect the code provided by users from resources

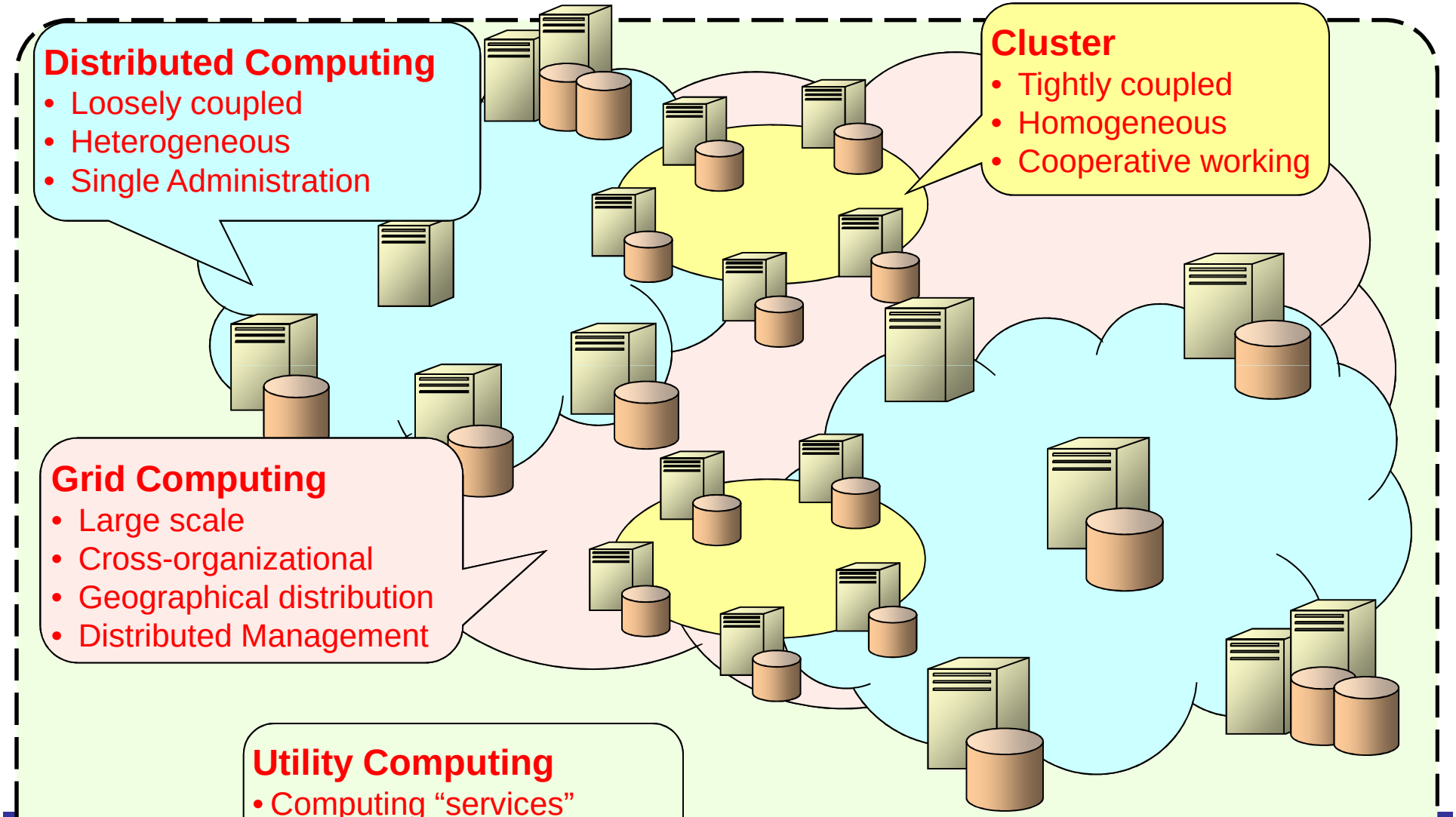
The Grid
(Foster et. al)

"Resource sharing & coordinated problem solving in dynamic, multi-institutional virtual organizations"



1. Enable integration of distributed resources
2. Using general-purpose protocols & infrastructure
3. To achieve better-than-best-effort service

Grid & Related Paradigms



Hiro Kishimoto

Security is a main challenge

- Many problems of distributed systems
  - Multiple administration domains
  - Federation issues
  - Different stakeholders interested to protect different assets
  - Need to continuously check system evolution
  - ….

Our focus on …

- How to control access and usage of resources and services in GRIDs
- Coarse grain level (Horizontal)
  - GRID services management
    - including service workflow authorization
- Fine grain level (Vertical)
  - Node resources management
    - including resources for computational services as OS SysCalls

Access control

Access control

- Goal:
  - Evaluate access request to resources and allow or not the access (grant/deny)
- Access control models:
  - DAC, MAC, RBAC, Attributed-based, etc…
- Relations with other security concepts
  - Subjects should be authenticated
  - Access control mechanisms must be secured
  - Access control policies must be preserved for integrity
  - …
- Acccess control is difficult for distributed and open systems
  - Subjects can belong to different adiminstrative domains
    - Different access control mechanisms, policies and credentials

Policies and Mechanisms (1)

- Access policy:
  - Principles and rules to regulates the access to the resources
  - High level specification:
    - **Open** policy: if it is not explicitly forbidden, then it is allowed
    - **Closed** policy: if is it not explicitly allowed, then it is forbidden
- Mechanism:
  - IT instruments to implement and enforce the access policy
    - Generic/programmable
    - Hard-wired

Policies and mechanisms (2)

- Why we should consider both:
  - An high level model allows to reason and compare different access control policies independently form implementation
    - clearer and easier analysis;
    - simpler access control policy management by system administrators
  - More implementations are possible for the same policy model
    - Unix Discretionary Access Control (DAC) may be implemented with:
      - Access Control Lists (resource-based access control)
      - Capabilities (subject-based access control)
  - Develop mechanisms implementing more policies
    - A single mechanism may implement more policies

Mechanisms (constraints)

- Reference Monitor
  - Is the component that effectively implements the access control and incapsulates (logically) the resource
- Tamper-proof
  - It should be impossible to modify its behavior
- Not-bypassable
  - All the access to the resources must pass through the reference monitor
- Security-Kernel/Trusted Computing Base
  - Complete: all the functionalities must be collected together in order to offer more control and reliability
  - Small footprint!
    - This makes it easier the debugging and formal analysis

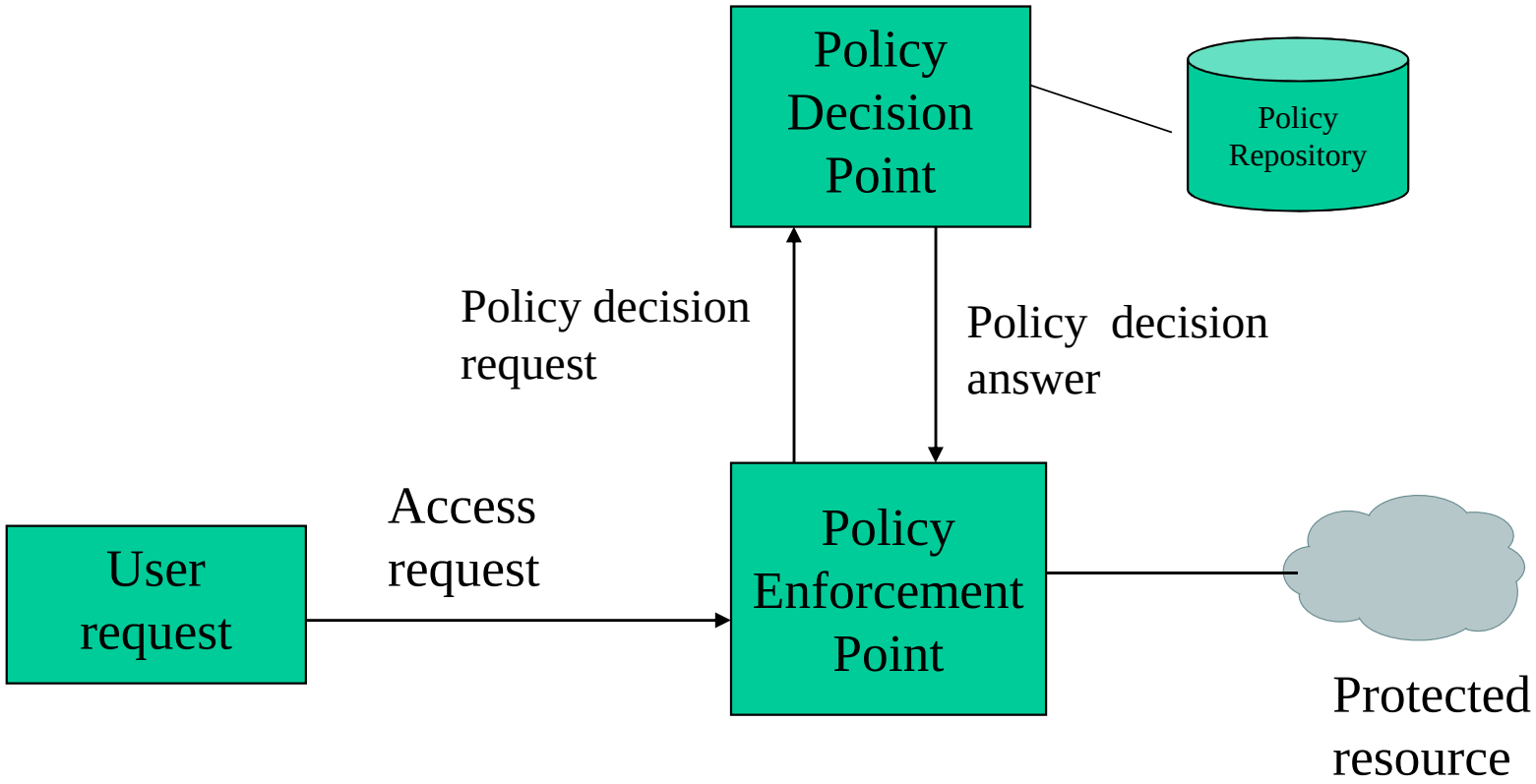
Policies (constraints)

- Security model
  - Defines the main characteristics of the policies, i.e. how to specify the access
  - Complete: for each request there is **at least** an answer
  - Consistent: for each request there is **at most** an answer
- The semantics of the policy language should be formal in order to allow (automated) verification
  - Maximal level of certification (EAL-7) ITSEC demands formal methods analysis

Common access control policy models

- Discretionary Access Control (DAC)
  - Resource owner control access to their resources
  - Unix security/Windows NT
- Mandatory Access Control (MAC)
  - System control access to its resources
  - Criteria might be either confidentiality (e.g. Bell-LaPadula) or Integrity (e.g. Biba)
- Role-Based Access Control (RBAC)
  - Access rights are assigned to subjects depending on their roles

Access control (IETF: RFC2904)



PDP = Policy Decision Point

PEP = Policy Enforcement Point

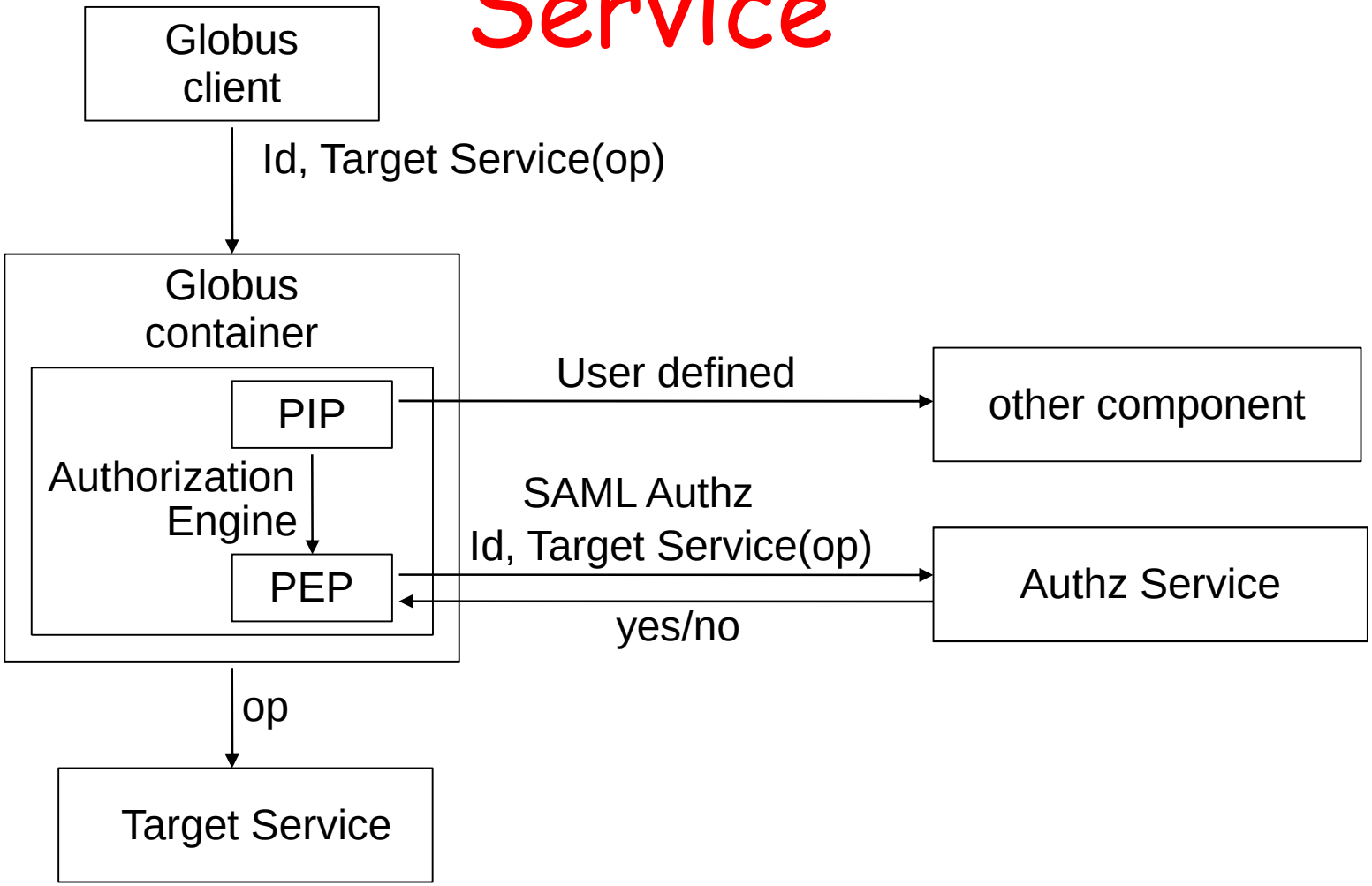
Globus toolkit

- GRID implementation
- Produced by Globus alliance
  - Compliant with Open Grid Services Architecture (OGSA)
- Currently version 5
- Used in many environments, in particular the scientific one

Globus Standard Authorization

- Authentication
  - PKI with X.509 End Entity Certificates
  - Proxy Certificates
- Authorization
  - Gridmap authorization service
    - File with the list of DNs of authorized users
    - Coarse authorization
  - SAML Callout to exploit external Authorization Services

SAML Callout Authz Service



Many existing approaches

- Existing implementations:
  - Community Authorization Service (CAS)
  - PERMIS
  - Akenti
  - Shibbolet
  - …
- A common feature is the lack of further control after granting access to resources

Community Authorization Service

- CAS manages a data base of VO policies
  - What each grid user can do as VO member
- A Grid user contacts CAS
  - Proxy cert. is exploited for authentication on CAS
  - CAS returns a signed policy assertion for the user
- Grid user creates a new proxy that embeds the CAS assertion
  - Exploits this proxy certificate to access services
- CAS-enabled services
  - Services that can enforce policies in CAS assertions

Community Authorization Service

Subject: /O=Grid/CN=Laura

Valid: 3/25/03 11:00 – 3/26/03 11:00

AuthorizationAssertion (non-critical extension):

Target Subject: /O=Grid/CN=Laura

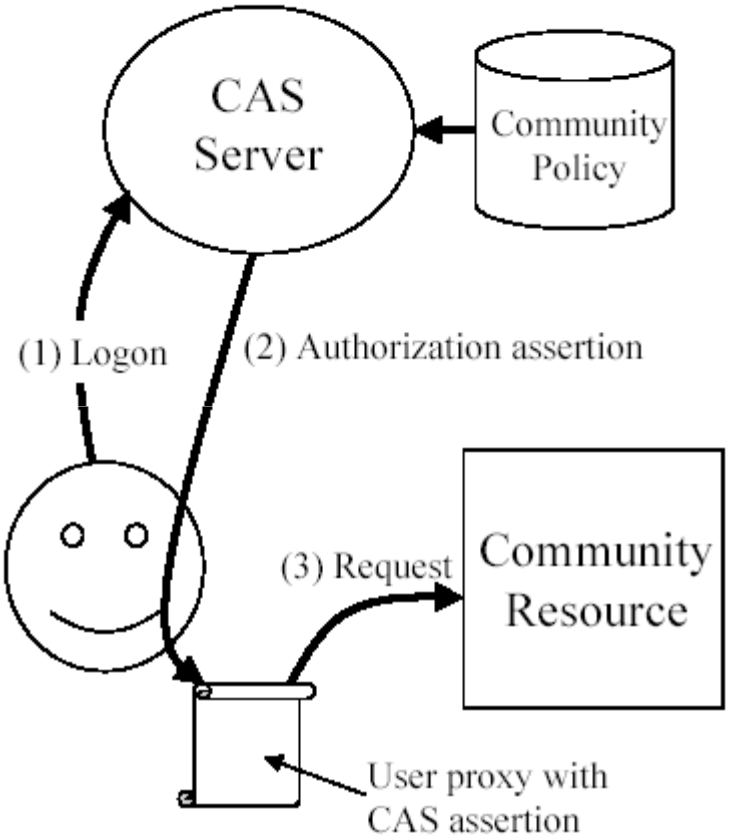
Valid: 3/25/03 13:00 –15:00

These actions are allowed:

Read gridftp://myhost/mydir/*

Signature (of assertion, by the VO CAS server)

Signature (of all above, by the user)



PERMIS

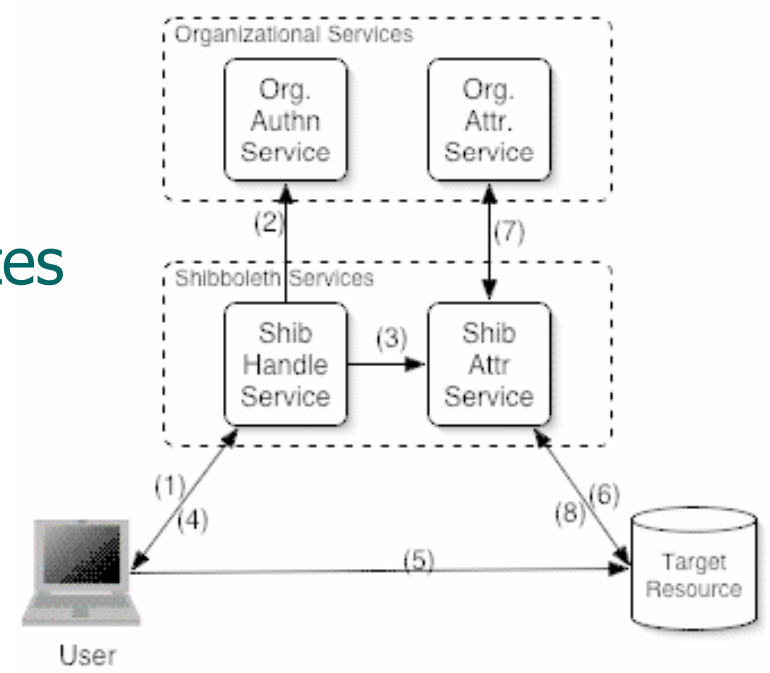
- Role Based Access Control Auth. Infrastructure
- Runs as Globus Service
  - SAML Callout Authz Service
- X.509 Attribute certificates to assign roles to users
  - Issued by an Attribute Authority
- X.509 Attribute certificates to store the policy
  - Definition of roles and permissions (XML)
  - Issued by the Source of Authority
- LDAP server(s) to store Attribute Certificates

Akenti

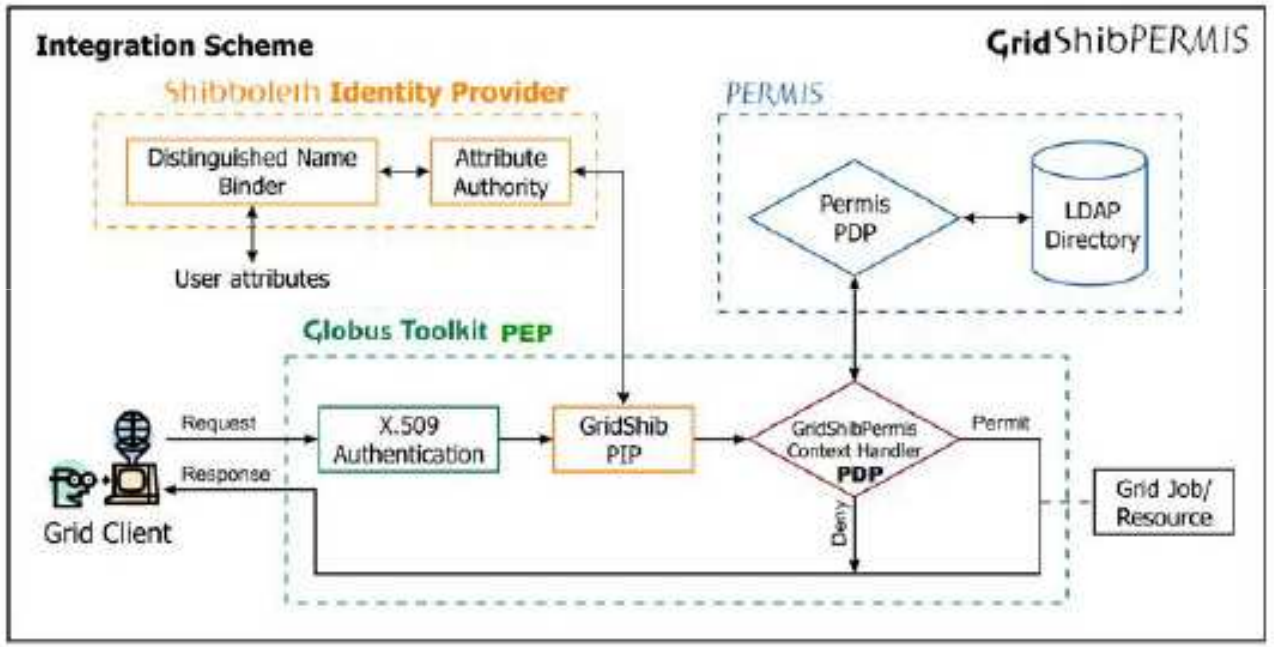
- Distributed authorization system where certificates are created independently by distinct stakeholders
- Certificates type
  - Authentication
  - Policy certificates
    - Specifies the Source of Authority for resources
  - Use condition certificates
    - Constraints to access resources
  - Attribute certificates
    - Assign attribute to users
- For each access Akenti finds (pulls) all the relevant authorization policy on LDAP/Akenti servers

Shibbolet

- Attribute Authority Service for distributed cross domain environments
  - User authentication is done on a local Shibbolet server that returns an handle to the user
  - Users use the handle to access remote services
  - Remote services use the user handle to retrieve user's attributes from a Shibbolet Attribute Server
  - Remote Service determines user access rights exploiting his attributes



Globus + Shibbolet + PERMIS



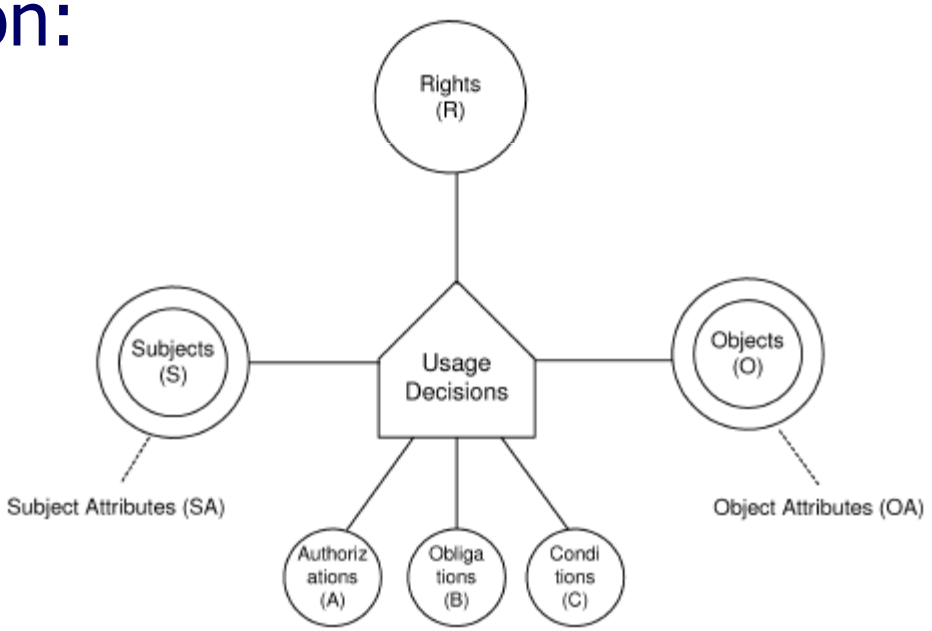
Usage Control

Usage Control Model

- Defined by J. Park and R. Sandhu

  The UCON Usage Control Model. ACM Trans. on Information and System Security, 7(1), 2004

- Usage control is based on:
  - –Authorizations (A)
  - –Obligations (B)
  - –Conditions (C)
  - –Mutability of Attributes
  - –Continuity of enforcement



Subjects and Objects

- Subjects: entities that perform actions on Objects. Are characterized by Attributes:
  - Identity
  - Role
  - Reputation
  - Credits
  - …
- Objects: entities that are used by Subjects. Are characterized by Attributes:
  - Value
  - Role permission
  - …

Mutability of Attributes: A main UCON feature

- Attributes of Subjects and Objects
  - Can be static (IMMUTABLE)
  - Can be updated (MUTABLE):
    - Before the action execution (PRE)
    - During the action execution (ONGOING)
    - After the action execution (POST)
- Example: A storage service charges its users when they read documents. The credit attribute of an user is updated before he reads a document.

Three usage decision factors

- Authorizations (A)
- Obligations (B)
- Conditions (C)

Authorizations

- Functional predicates for usage decisions that evaluate:
  - Subject Attributes
  - Object Attributes
  - Right (Action)

- Example: a computational service exploits a security policy to decide whether the user U can perform the action “read” on the file “a.txt”

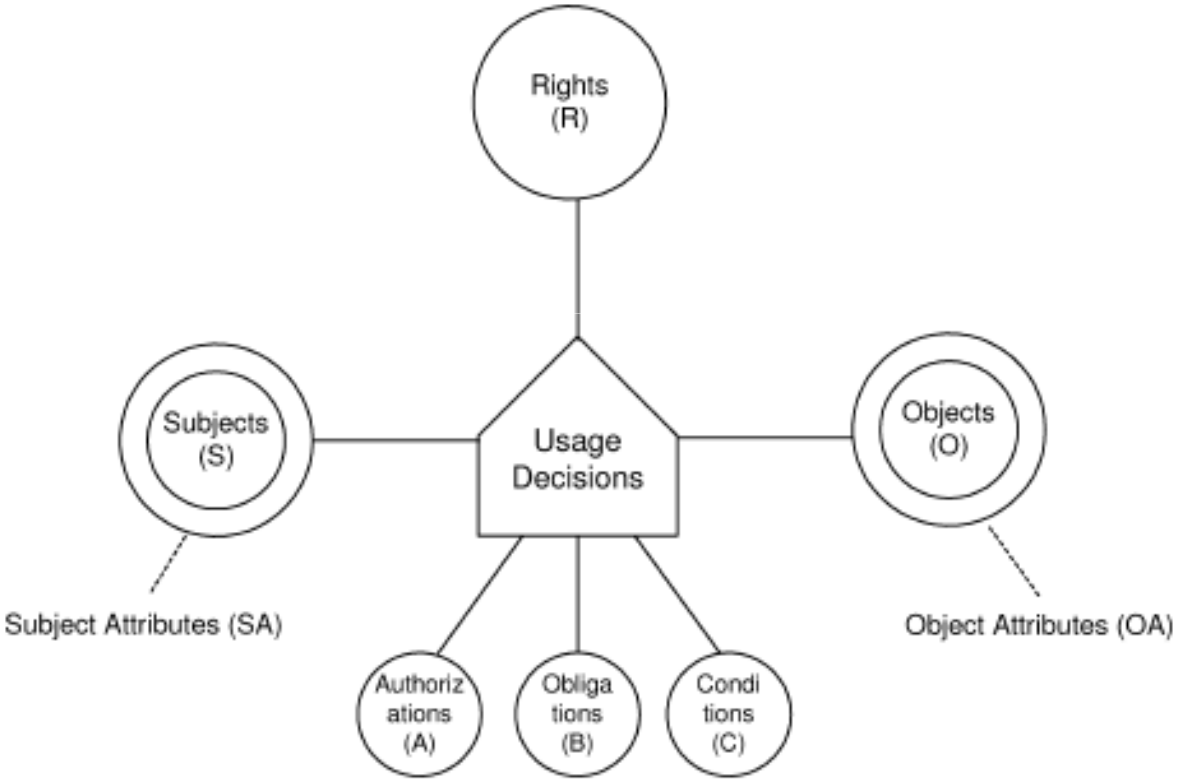
Obligations

- Functional predicates that verify mandatory requirements that must have been performed by subjects.
  - Actions
  - .....
- Example:
  - the user of a storage service must download the license agreement before downloading any other document.
- Note that obligations do not simply correspond to actions from the subject that requested the access to the resource

Conditions

- Environmental or system based decision factors
  - Not directly related with Subjects and Objects
  - e.g.
    - Current local time
    - Current system workload
    - System status
- Example: night-users can submit jobs to a computational resource only from 8pm to 8am and if the work load is low.

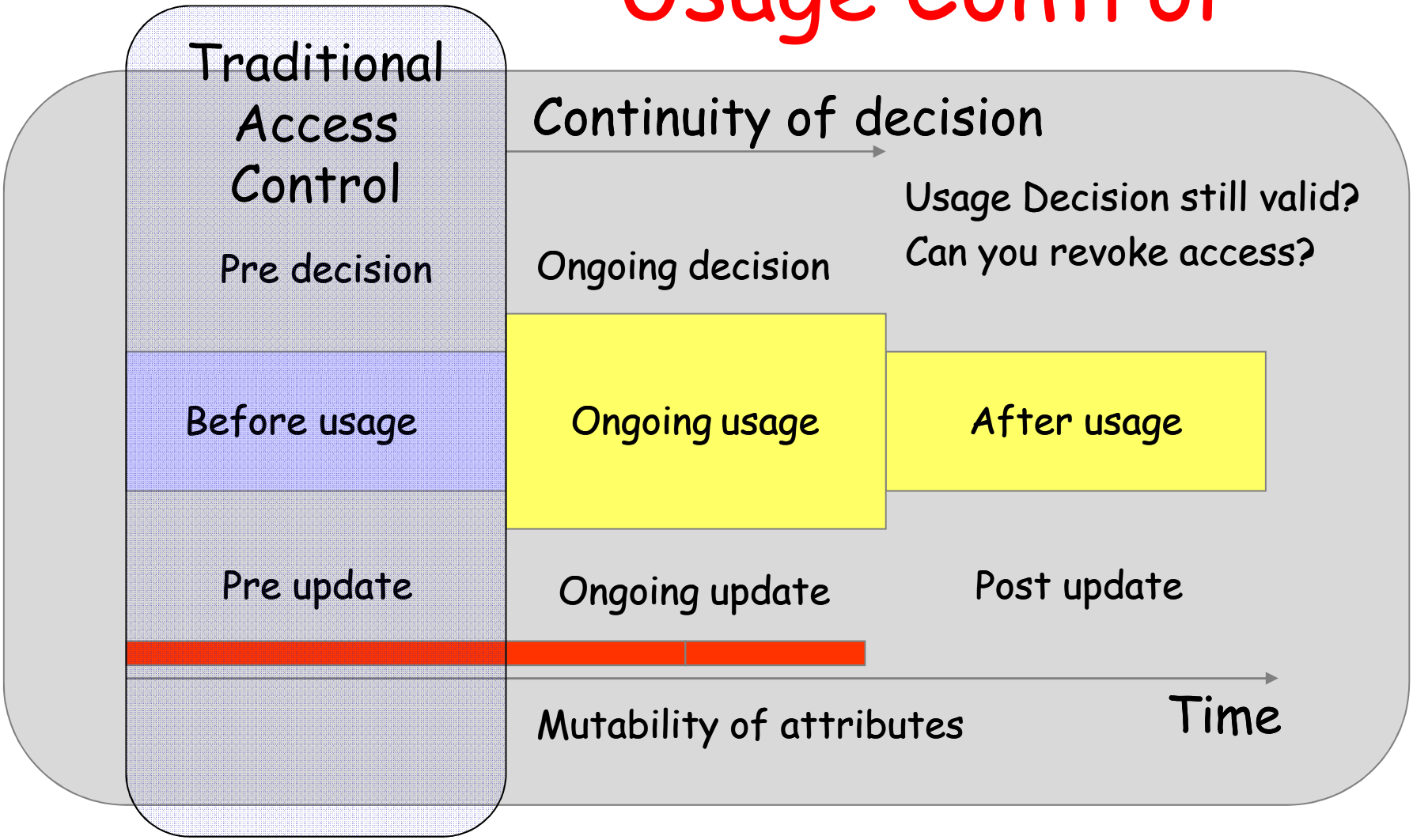
Usage Right Evaluation



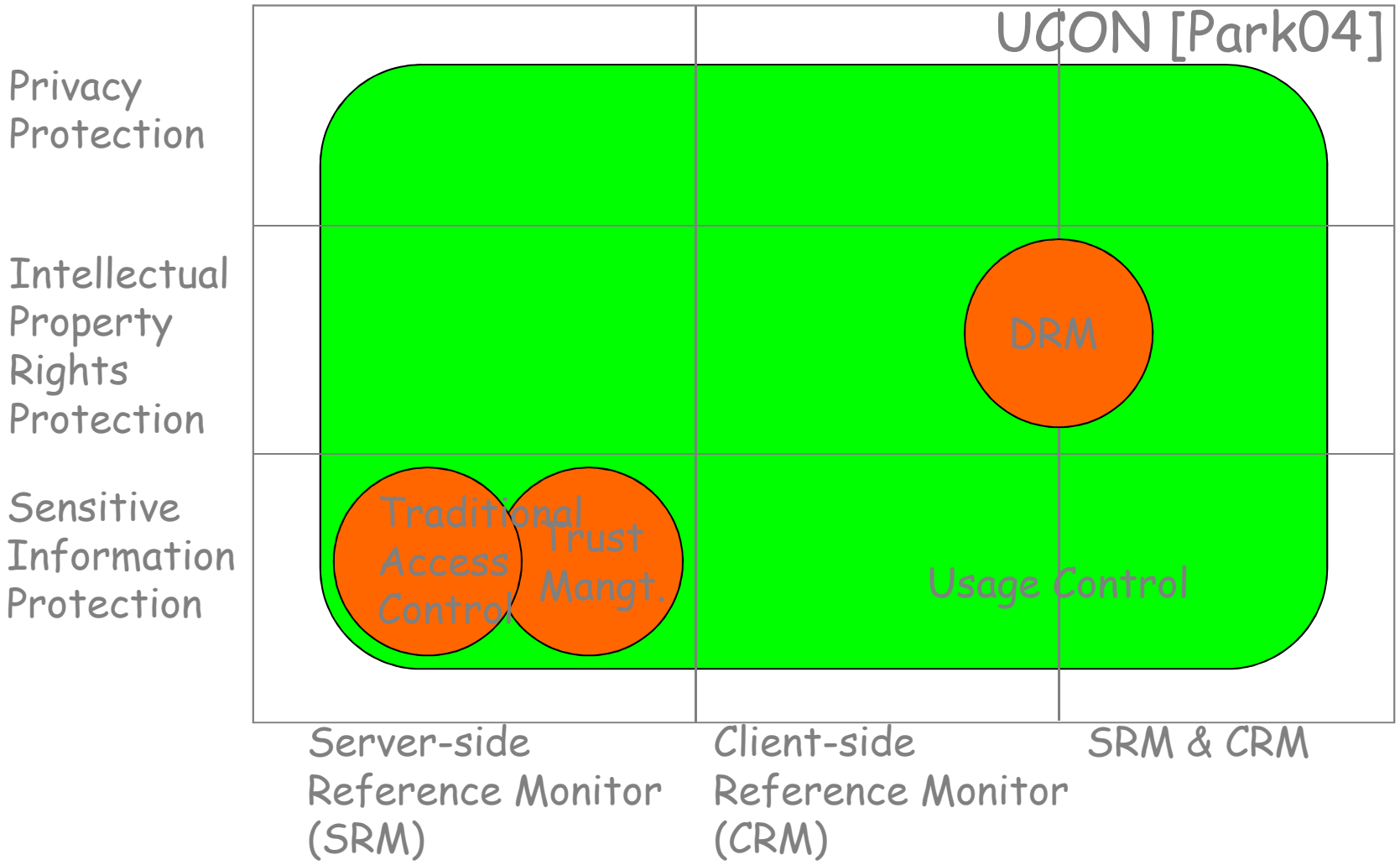
Continuity: A main UCON feature

- The evaluation of an usage right can be performed
  - Before the action (PRE)
    - Common access control models
  - Continuously during the action (ONGOING)
    - The right could be revoked and the action interrupted
    - Used for long lived actions (days, months,..)

From Access Control to Usage Control



Usage Control Model: Beyond Access Control



UCON Core Models

- We have 24 different basic models
  - used (even combined) to model real systems
- [Pre/On] Time of control
- [A/B/C] Decision Factors
- Immutability (0)/Mutability (when: 1 (pre), 2 (on), 3 (post))
- E.g. OnA_3:
  - On going authorization with post update

Examples: UCON pre A_1

- Authorization is performed before the right is exercised
  - With pre update of Attributes
    - Attributes value is updated before the usage is started

    pay per use with pre-paid credit:

    Authorizations granted when credit(s) > value(o,r)
    preUpdate(s,o): credit(s) = credit(s) - value(o,r)

Examples: UCON $preA_3$

– With post update of Attributes
  - Attributes value is updated after the usage is terminated

membership based payment

Authorizations granted when clubmember(s)
postUpdate(s): exp(s) += value(o,r) x utime(s,o)

Example: UCON Ongoing Autorization ($OnA_{1,3}$)

- The right is granted without pre decisions, but authorization decisions are made continuously (repeatedly) while the right is exercised

Authorizations initially granted then revoked when
(usageNum(o) >10) and (s,t) in startT(o) with t min

preUpdate(startT(o)): startT(o) = startT(o) U {(s,t)}
preUpdate(usageNum(o)) : UsageNum(o)++
postUpdate(usageNum(o)) : UsageNum(o)--
postUpdate(startT(o)): startT(o) = startT(o) - {(s,t)}

UCON on GRID: Some Issues

- Ongoing Controls
  - Usage revocation
  - Main novelty w.r.t. usual access control
    - **It requires active PDP!**
- Attribute Management
  - Subject - Objects
  - How to:
    - Represent
    - Store
    - Retrieve
    - Update (GRIDs are distributed environments)
- Conditions
  - Environmental conditions for Grid

A formal policy language for UCON

POLPA
Policy language for UCON (1)

- We adopted an operational language based on process description languages
  - The idea is that we should described the allowed sequential behaviour of actions
  - **Policy Language (based) on Process Algebra (PolPA)**
    - **Based on CSP-like syncrhonization**
- At this stage we wanted a model with powerful constructs, yet operational and clean
- Policies can thus be formally (under certain conditions), verified, compared, minimized, refined, etc…

Policy language for UCON (2)

- All UCONs core models can be encoded
- Suitable to express also workflow authorization statements among different GRID services
  - POLPA naturally expresses sequences of allowed actions
  - POLPA naturally expresses the conjunction of policies (both policies must be satisfied)

Usage control actions

- We assume the following usage control actions
  - tryaccess(s, o, r): performed by subject s when performing a new access request (s, o, r)
  - permitaccess(s, o, r): performed by the system when granting the access request (s, o, r)
  - revokeaccess(s, o, r): performed by the system when revoking an ongoing access (s, o, r)
  - endaccess(s, o, r): performed by a subject s when ending an access (s, o, r)
  - update(attribute): updating a subject or an object attribute.

Policy language for UCON (3)

- The main constructs are the following:
  - $\alpha(x).P$ is the sequential operator, and represents the possibility of performing an action α (x) and then be compliant with the policy P;
  - **p**(x).P behaves as P in the case the predicate **p**(x) is true;
  - x:=e.P assigns to variables x the values of the expressions e and then behaves as P;
  - P1 or P2 is the alternative operator, and represents the non deterministic choice between P1 and P2;
  - P1 par $_{\{\alpha 1, \ldots, \alpha n\}}$ P2 is the synchronous parallel operator. It expresses that both P1 and P2 policies must be simultaneously satisfied. This is used when the two policies deal with common actions (in $\{\alpha 1, \ldots, \alpha n\}$).

Operational semantics

- Behaviour is modeled through a *labeled transition system*

$$\langle \mathcal{P}, Act, \{\xrightarrow{\alpha}\}_{\alpha \in Act} \rangle$$

Often we are just interested in traces rather than full graphs.

Some rules

$$(prefix)\frac{}{\alpha.P \xrightarrow{\alpha} P}$$

$$(pred)\frac{P(\vec{e}/\vec{x}) \xrightarrow{\alpha} P'}{p(\vec{e}/\vec{x}).P(\vec{e}/\vec{x}) \xrightarrow{\alpha} P'} \quad [p(\vec{e}/\vec{x}) = true]$$

$$(par_1) \quad \frac{P \xrightarrow{\beta} P'}{P \; par_{\{\alpha\}} \; Q \xrightarrow{\beta} P' \; par_{\{\alpha\}} \; Q} \quad [\beta \notin \{\alpha\}]$$

$$(par_2) \quad \frac{P \xrightarrow{\beta} P' \quad Q \xrightarrow{\beta} Q'}{P \; par_{\{\alpha\}} \; Q \xrightarrow{\beta} P' \; par_{\{\alpha\}} \; Q'} \quad [\beta \in \{\alpha\}]$$

Examples of UCON policies (1)

- PreAuthorization without update ($PreA_0$)
  - The preAuthorization model without update is shown below where $p_A(s,o,r)$ is the predicate that grants authorization.

tryaccess(s, o, r).

p_A(s, o, r).

permittaccess(s, o, r).

endaccess(s, o, r)

Examples of UCON policies (2)

- PreAuthorization with postUpdate ($PreA_3$)
  - The preAuthorization model with post update is shown below where $p_A(s,o,r)$ is the predicate that grants authorization and update (s,o,r) is the update operation
  - Contrarily previous models, we have not to distinguish among different authorization/update operations, i.e. pre/post/on since the kind of authorization/update is implicitly defined by the relative temporal position with respect the other usage control actions in the policy.

$tryaccess(s, o, r)$.

$p_A(s, o, r)$.

$permittaccess(s, o, r)$.

$endaccess(s, o, r)$.

$update(s, o, r)$

Example of UCON policies (3)

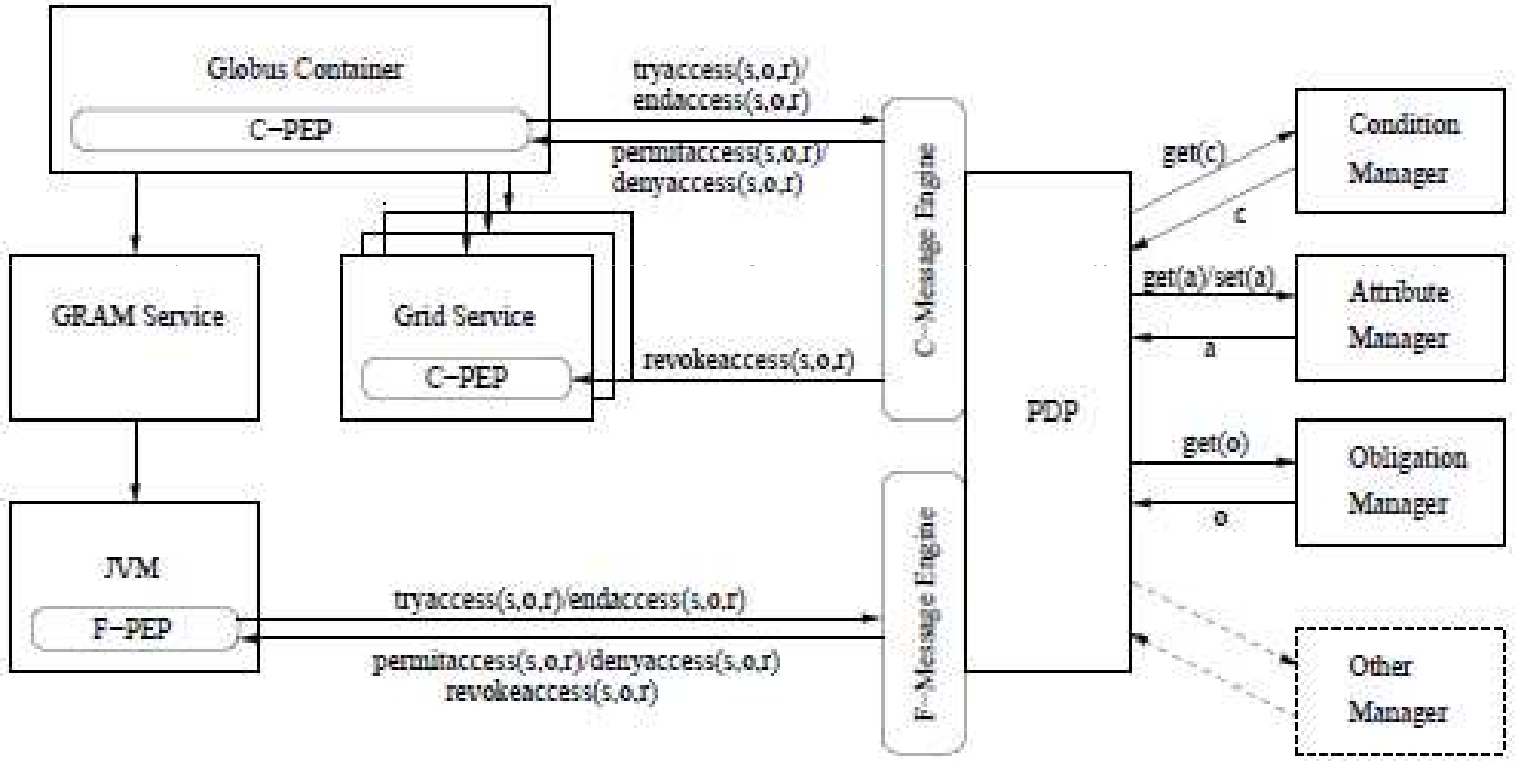
- OnAuthorization without update (OnA)

```
tryaccess(s,o,r).
permittaccess(s,o,r).
 (endaccess(s,o,r)
    or
 (not trust(s)>treshold(o)).revokeaccess(s,o,r)))
```

Our Architecture for UCON in GRID

Basic Architecture

- We have the following basic architecture:



UCON for computational services

UCON for Computational Services

- Computational service defined by the Globus Toolkit

```
Starting SOAP server at: https://146.48.99.119:8443/wsrf/services/
With the following services:

[1]: https://146.48.99.119:8443/wsrf/services/AdminService
[2]: https://146.48.99.119:8443/wsrf/services/AuthzCalloutTestService
.....................
[20]: https://146.48.99.119:8443/wsrf/services/ManagedExecutableJobService
[21]: https://146.48.99.119:8443/wsrf/services/ManagedJobFactoryService
[22]: https://146.48.99.119:8443/wsrf/services/ManagedMultiJobService
[23]: https://146.48.99.119:8443/wsrf/services/ManagementService
.....................
```

- Allows remote grid user to execute applications on remote resources
  - Transfer of executable code and data (gridFtp)
  - Execution of the code
  - Transfer of results, stdout, stderr (gridFtp)
- We focus on Java applications

UCON for Computational Services (2)

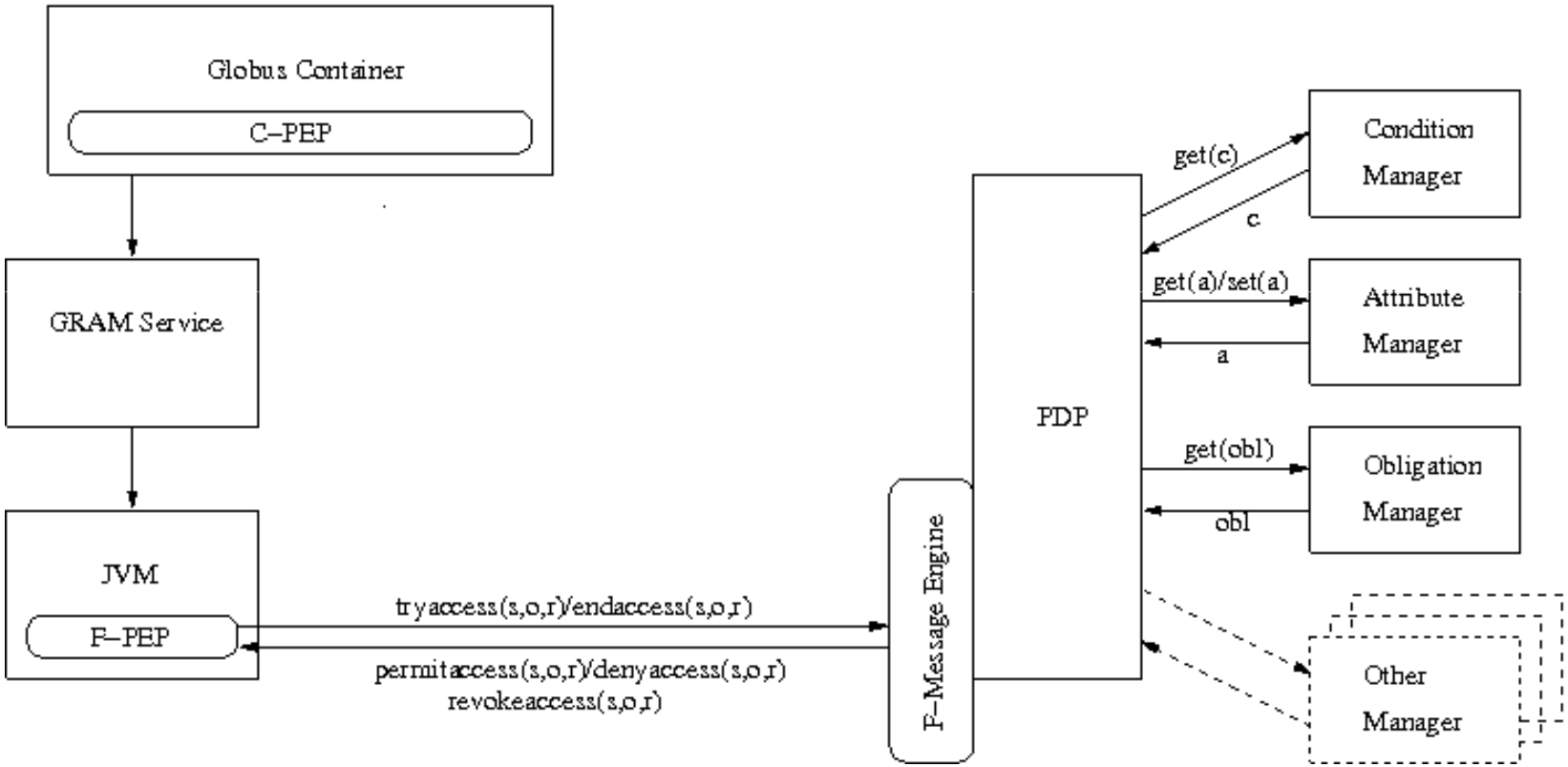
To monitor the application, we check the system calls executed by the JVM

The PEP is in the JVM

All the interactions of the application with the computational resource are controlled

Fine grain history based usage control

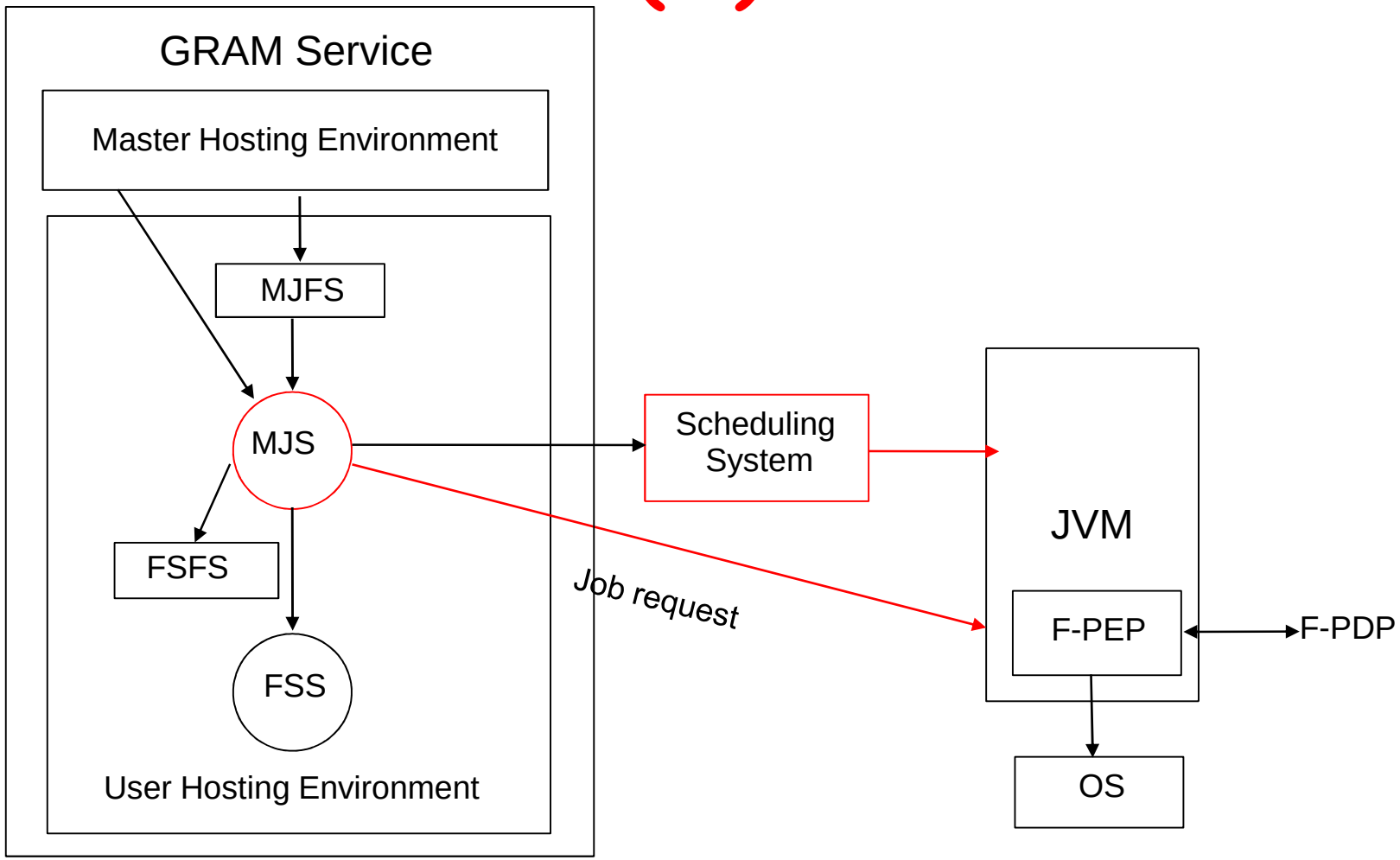
UCON for Computational Services (3)



Integration with Globus GRAM (1)

- GRAM components that have been "modified":
  - Scheduling System
    - We customized a scheduler (fork) to invoke our monitored JVM
  - Managed Job Service
    - Passes the job request attributes (e.g. MEM or CPU requirements) to our usage control monitor

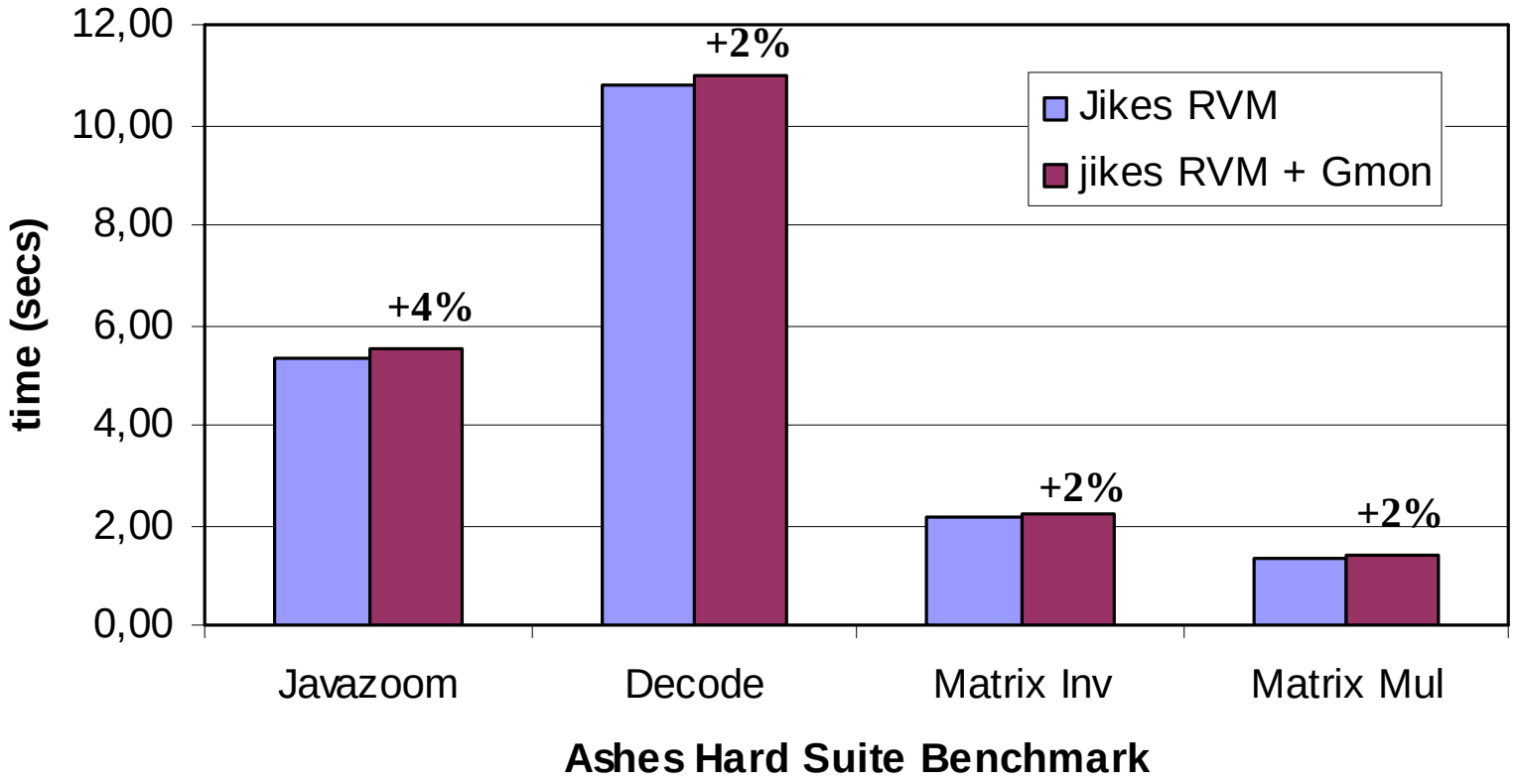
Integration with Globus GRAM (2)



Java Virtual Machine

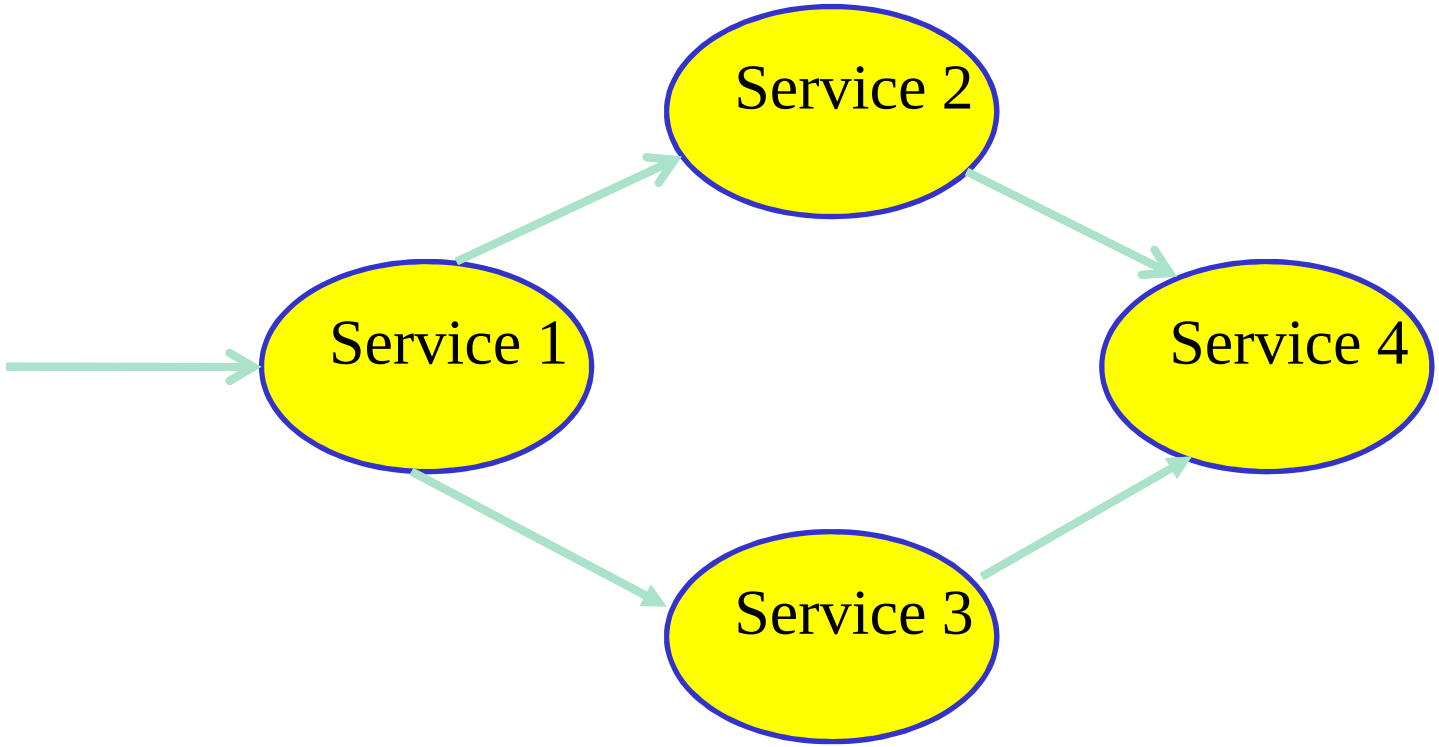
- IBM Jikes Research Virtual Machine
  - Open source Java Virtual Machine
  - Research oriented
  - Follows:
    - The Java Language Specification
    - The Java Virtual Machine Specification

Performances



UCON for GRID services

Example of GRID service workflow authorization

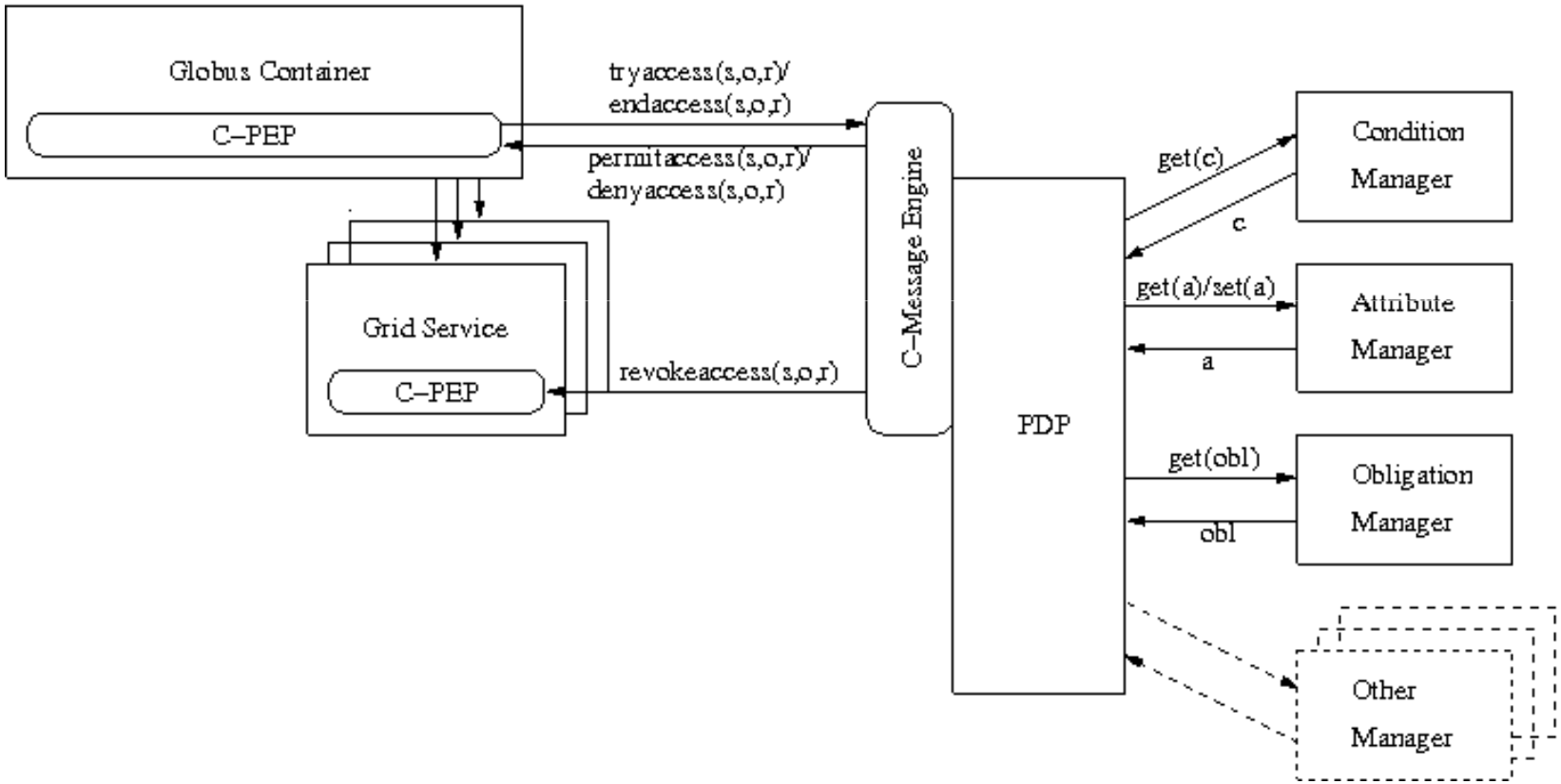


Service1.(Service2 **par** Service 3); Service 4

UCON at Service level

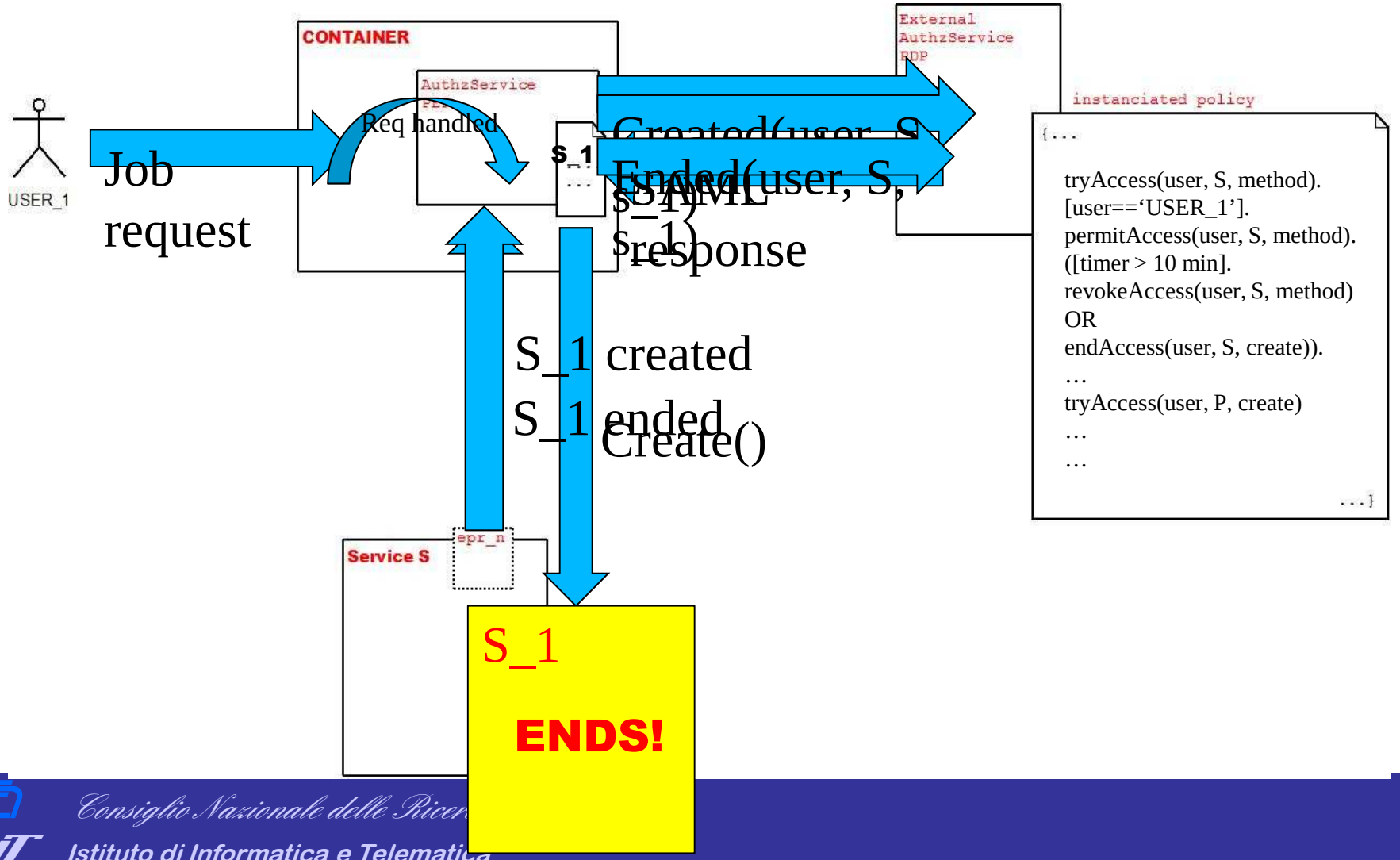
- The UCON PDP is invoked when:
  - A new service instance is created
  - A service instance terminates
- The PDP evaluates the policy
  - Every time a new service instance is created (to check the execution right)
  - Continuously to evaluate some usage conditions
- The PDP interrupts (!) a service instance (revoke the execution right)
  - If a usage condition is not valid anymore
  - While the instance is running (before its end)

UCON at Service level



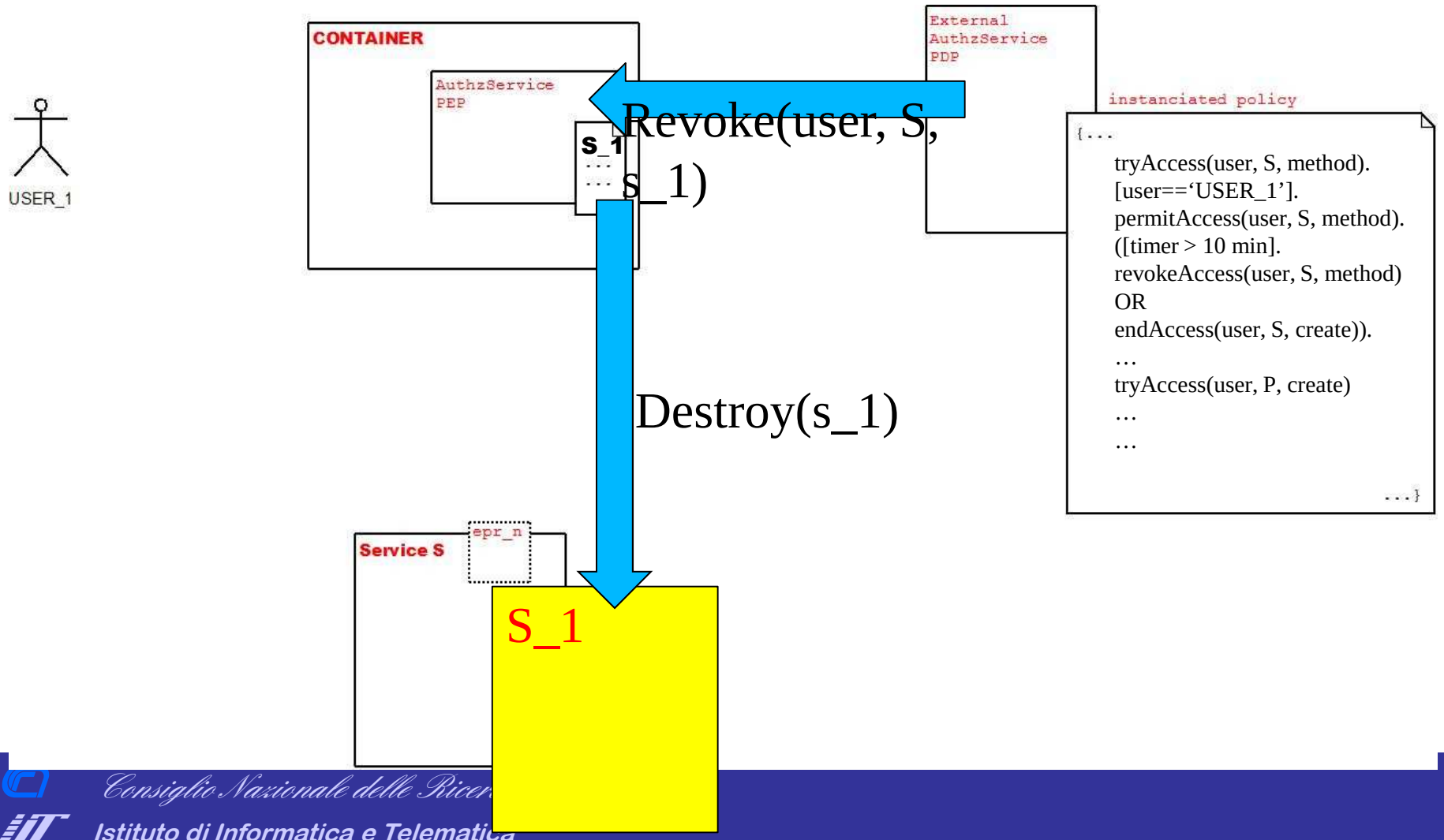
Creation of a new service

Execution Flow (Creation)



Revocation of a running service

Execution Flow (Revocation)

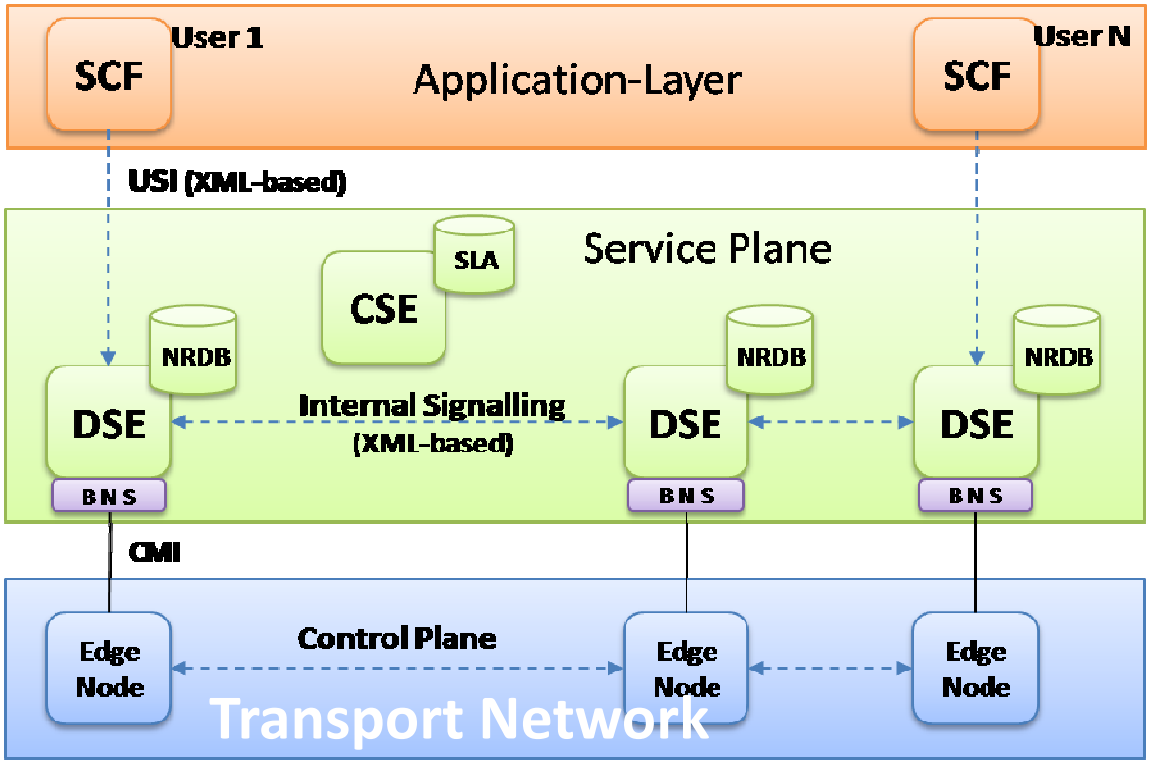


UCON for network services

Service Oriented Networks

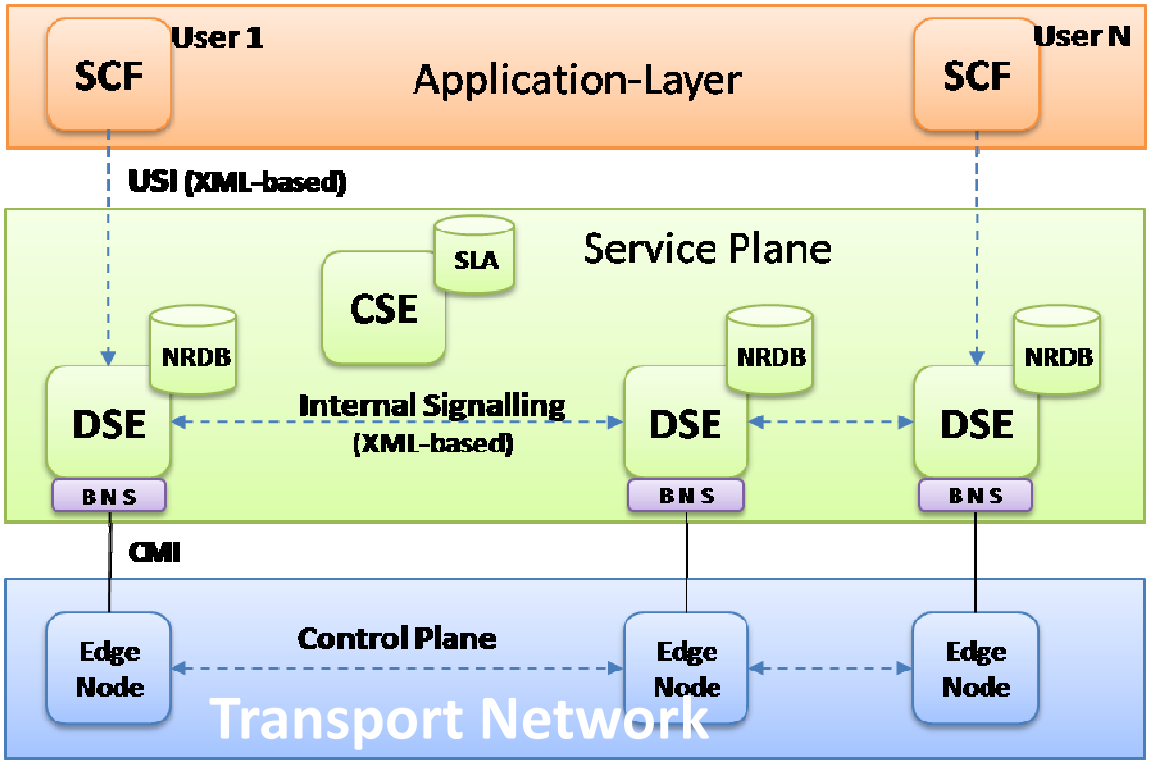
- On-demand framework for network services provisioning, e.g.
  - VOIP
  - MultiMedia on Demand
  - ….
- Level of abstraction suitable for being invoked directly by applications

Service Oriented Optical Network Architecture (SOON)



SCF: Service Control Function
(i.e., Application Entity)
DSE: Distributed Service Element
CSE: Centralized Service Element
SLA: Service Level Agreement
BNS: Basic Network Server

Service Oriented Optical Network Architecture (SOON)



SCF: Service Control Function
(i.e., Application Entity)
DSE: Distributed Service Element
CSE: Centralized Service Element
SLA: Service Level Agreement
BNS: Basic Network Server

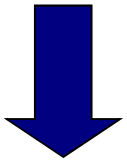
Need for an advanced security support to control the usage of network services!

Motivation and Challenges

- Next Generation Networks promote a competitive environment for service offerings
  - Providers offer a large variety of complementary services, e.g., IP-based connectivity services, multimedia services, content service
  - New and attractive services can be delivered also exploiting new advanced features of user terminals
    - Mobility, processing capabilities (PDAs, Laptops)
- Providers would benefit by combining their resources for providing value-added services (e.g., broadband wired and wireless connectivity, multimedia and e-learning), thus increasing revenue opportunities

Authorization in Service Oriented Networks

- Controls over network services are required in order to avoid malicious accesses, causing violations (e.g. overload of the network resources) that can lead to Denial of Service



- Usage Control authorization system
  - Monitors the access to network services and their usage

Usage Control in SOON

- Applications require access to network services (e.g. Multimedia on Demand) to the service provider
- The authorization system evaluates the security policy before authorizing the access
- While the application performs the access (e.g. is downloading the data stream), the usage control authorization system continuosly evaluate the policy
  - If the value of an attribute changes and the policy is not satisfied any more, the downloading is interrupted

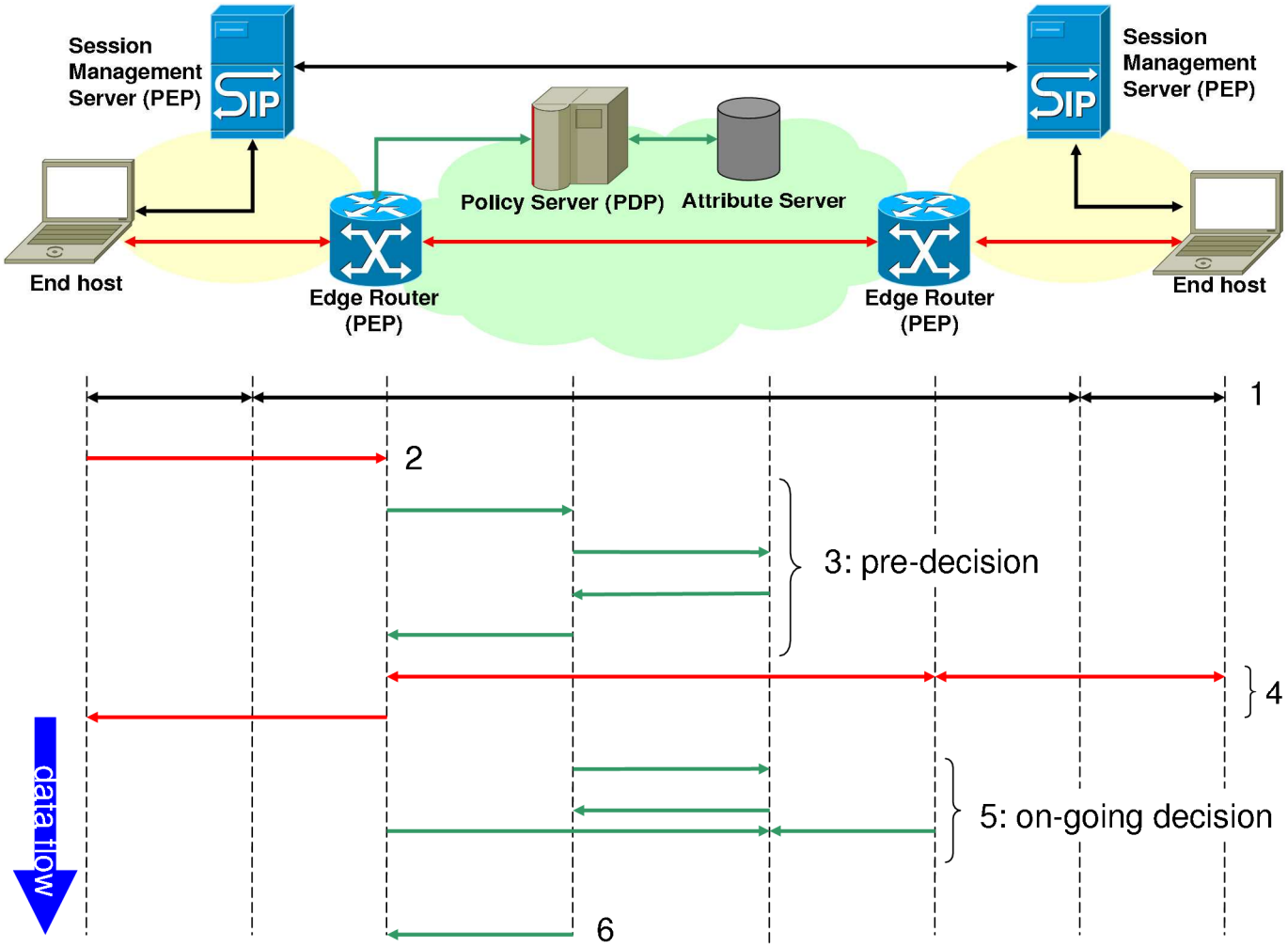
Example of Security Policy (natural lang)

- A user(application) is allowed to set up a new channel for data streaming only if:
  - The total bandwith currently allocated to the user is less than a given threshold T
  - The user's profile is "GOLD"
  - The user's reputation is more than a given value R
- During the usage of the channel, the user's reputation should be greater than R

Example of Security Policy (POLPA lang)

```
tryaccess(user, net, createChannel(dest, band, ch)).
[(user.usedBand+reqBand<=T) and (user.profile=GOLD) and
  (user.reputation>=R)]
permitaccess(user, net, createChannel(dest, band, ch)).
update(user.usedBand+=reqBand).
( [(user.reputation<R)].revokeaccess(user, net, createChannel(dest, band, ch)))
or
  endaccess(user, net, createChannel(dest, band, ch))
).
update(user.usedBand-=reqBand)
```

Authorization Workflow in SOON



Requests:

1 Set-up

2 Reserve

3 Authz

4 Reserve & response

5 Ongoing control

6 Revoke

Authorization Workflow in SOON

1 The End Host issues a set-up request to the Session Management Service

2 The End Host issues a request to reserve the required resource

3 The Edge Router intercepts the request and sends an authorization request to the Policy Server (pre)

4 The Edge Router triggers an end to end signaling to reserve all the resources on the path and receives the response

5 While the resources are in use, the Policy Server continuously evaluate the security policy (ongoing)

6 In case of violation, the Policy Server requires the Edge Router to interrupt the data stream

Ongoing and Future Works

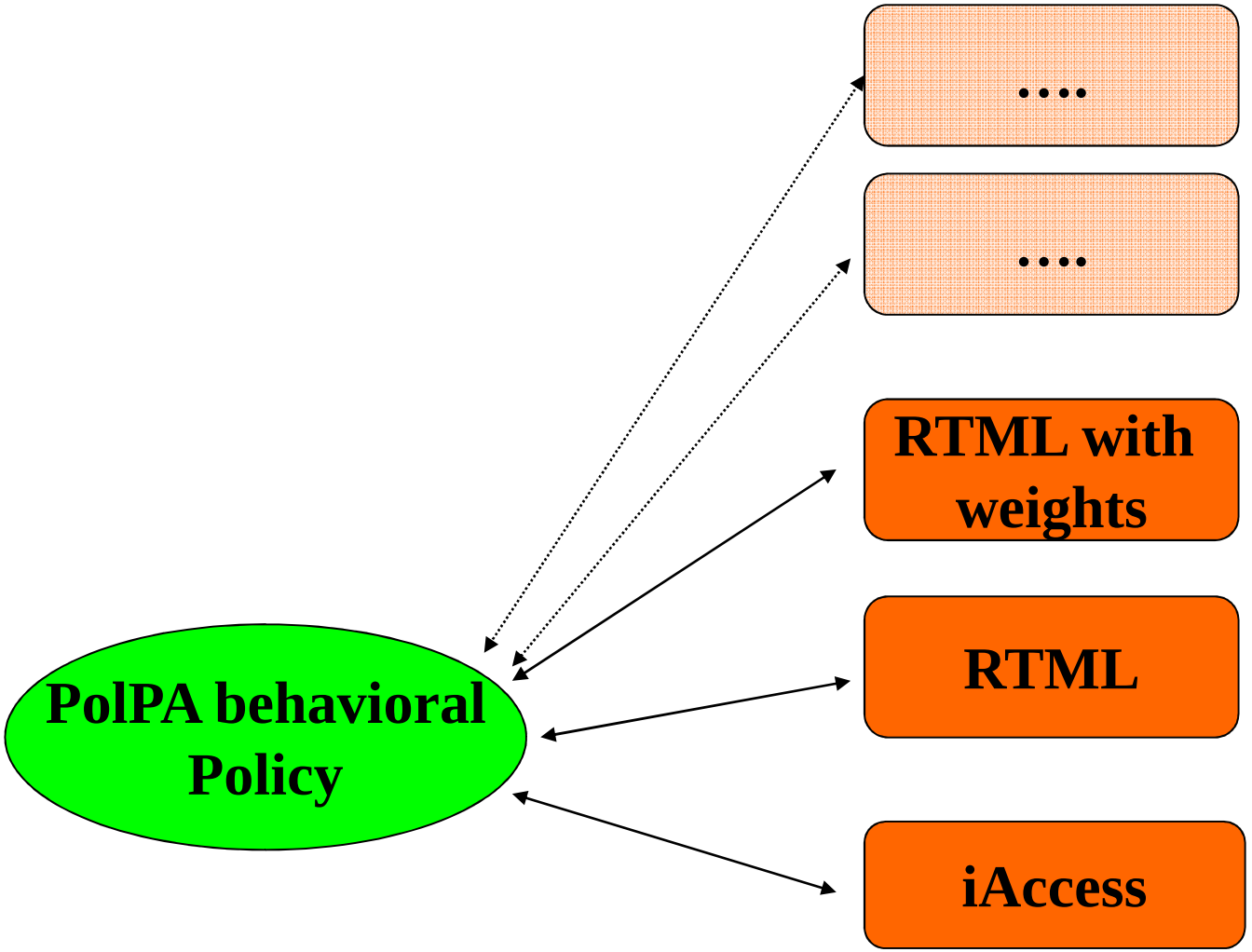
- Full implementation of the framework for Optical networks
- Integration with the Credential Based Mechanisms
  - To allow access to unknown but trustable user (see next part on trust management)

Policy orchestration

Policy orchestration (1)

- Our policy language integrates behavioural policies with **predicates**
- **Predicates** may be obtained by using several formal methods and eventually **other policy languages** (and evaluation services)
  - E.g, temporal logic, first order logic, etc.
  - ad/hoc policy mechanisms
  - Role/Based trust management
- We can decide which policy to evaluate at which time and in which order
- We already extended our framework with
  - Role-Based trust management RTML of Ninghui Li et al. (also extended with trust levels)
  - i-Access prototype of Koshutanski et al. that allow also negotiation policies

Policy orchestration (2)



U-XACML

XACML – Key Aspects

- General-purpose authorization policy model and XML-based specification language
- Input/output to the XACML policy processor is clearly defined as XACML context data structure
- Extension points: function, identifier, data type, rule-combining algorithm, policy-combining algorithm, etc.
- A policy consists of multiple rules
- A set of policies is combined by a higher level policy (PolicySet element)

XACML Schemas

Request Schema

Request

- Subject
- Resource
- Action

Policy Schema

PolicySet (Combining Alg)

- Policy* (Combining Alg)
  - Rule* (Effect)
    - Target
      - Subject*
      - Resource*
      - Action*
      - Environment
    - Effect
    - Condition
    - Obbligation*

Response Schema

Response

- Decision
  - Permit
  - Permit w/ Obligations
  - Deny
  - N/A
  - Indeterminate

Ucon A-B-C/XACML

- XACML defines obligations as task that will be executed by the PEP
  - U-XACML obligations have been modeled with XACML obligations
- XACML defines conditions as functions that involve attributes
  - U-XACML authorizations and conditions have been modeled with XACML conditions

Attributes

- Attributes in UCON could be
  - Immutable
  - Mutable
    - by the security policy
    - by the environment
    - by both
- Update could be
  - Pre (1)
  - Ongoing (2)
  - Post (3)

Attributes in U-XACML

- New tag <AttrUpdate> defines the attribute update
- Schema:

```
<xs:element name="Policy" type="xacml:PolicyType"/>
<xs:complexType name="PolicyType">
  <xs:sequence>
      ...
      <xs:element ref="xacml:Target"/>
    <xs:choice maxOccurs="unbounded">
      <xs:element ref="xacml:CombinerParameters" minOccurs="0"/>
      ...
      <xs:element ref="xacml:Rule"/>
    </xs:choice>
    <xs:element ref="xacml:Obligations" minOccurs="0"/>
    <xs:element ref="xacml:AttrUpdates" minOccurs="0"/>
  </xs:sequence>
  <xs:attribute name="PolicyId" type="xs:anyURI" use="required"/>
  <xs:attribute name="RuleCombiningAlgId" type="xs:anyURI" use="required"/>
</xs:complexType>
```

Continuous Policy Enforcement

- UCON model defines ongoing
  - Authorizations
  - Conditions
  - Obligations
- Ongoing factors should be continuously evaluate during the access
- In U-XACML this is implemented through the <DecisionTime> tag
  - Pre (1)
    - Classical access control
  - Ongoing (2)
  - Post (3)

Continuous Policy Enforcement

- Schema

- <xs:element name="Condition" type="xacml:ConditionType"/>
- <xs:complexType name="ConditionType">
- <xs:sequence>
- <xs:element ref="xacml:Expression"/>
- </xs:sequence>
- <xs:attribute name="DecisionTime" type=”xs:integer” use=”required”>
- </xs:complexType>

Continuous Policy Enforcement

- Schema

- <xs:element name="Obligation" type="xacml:ObligationType"/>
- <xs:complexType name="ObligationType">
- <xs:sequence>
- <xs:element ref="xacml:AttributeAssignment" minOccurs="0"
- maxOccurs="unbounded"/>
- </xs:sequence>
- <xs:attribute name="ObligationId" type="xs:anyURI" use="required"/>
- <xs:attribute name="FulfillOn" type="xacml:EffectType" use="required"/>
- <xs:attribute name="DecisionTime" type="xs:integerI" use="required"/>
- </xs:complexType>

Example AttrUpdates

```
<?xml version="1.0" encoding="UTF-8" ?>
- <Policy xmlns="urn:oasis:names:tc:xacml:1.0:policy" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  PolicyId="GeneratedPolicy" RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:ordered-permit-
  overrides">
 + <Target>
 - <Rule RuleId="LoginRule" Effect="Permit">
  + <Target>
  + <Condition FunctionId="urn:oasis:names:tc:xacml:1.0:function:double-greater-than" DecisionTime="2">
  </Rule>
 - <AttrUpdates>
  - <AttrUpdate UpdateExpression="http://iit.cnr.it/U-XACML/increase" UpdateTime="3" FulfillOn="Permit">
      <AttributeAssignment AttributeId="urn:iit:cnr:names:subject:reputation"
        DataType="http://www.w3.org/2001/XMLSchema#integer" />
    </AttrUpdate>
  </AttrUpdates>
  <Rule RuleId="FinalRule" Effect="Deny" />
</Policy>
```

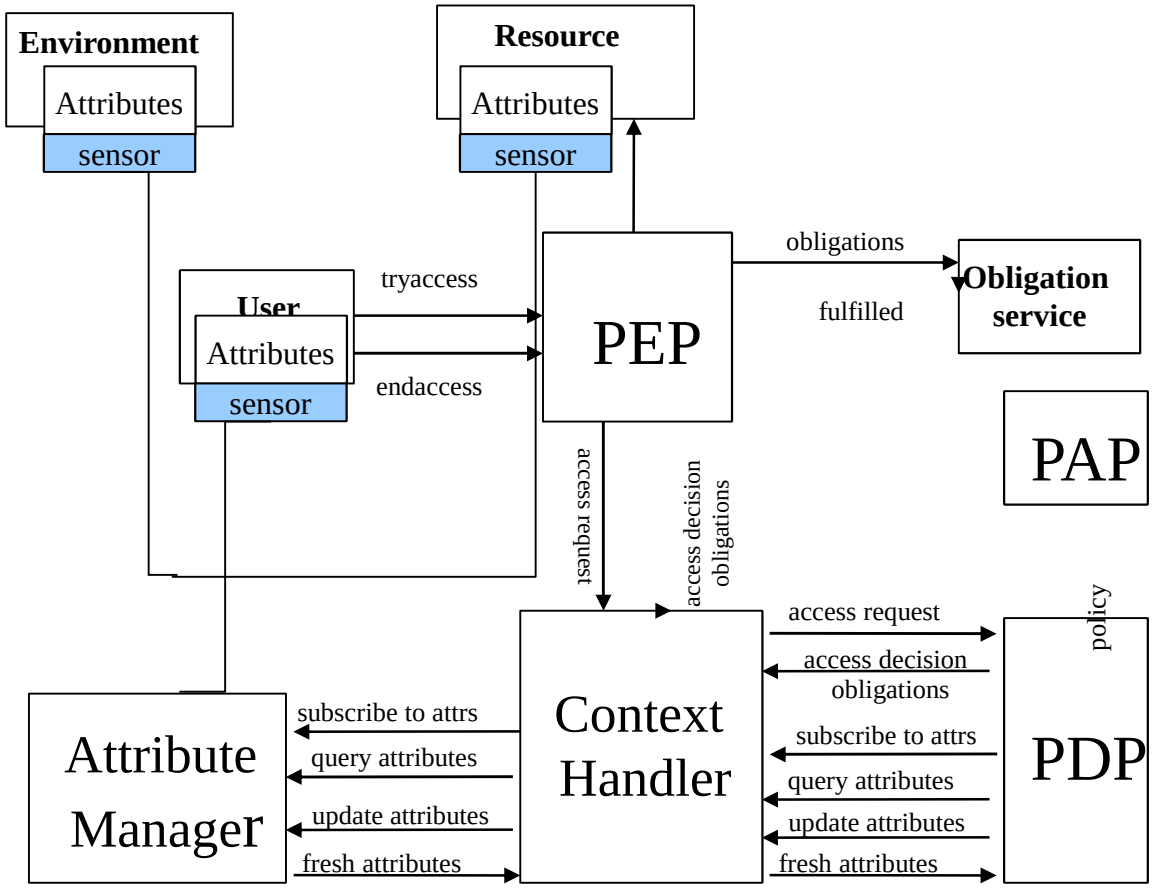
Architecture

- Main components:
  - PDP
    - Performs the decision process evaluating the security policy
  - PEP
    - Intecepts security relevant actions
    - Enforces of PDP decision
    - Obligation fulfillment
  - Attribute Manager
    - retrieves/updates atribute values

Architecture (cont)

- Attributes sensors
  - Detect changes of attributes
- Context Handler
  - Converts messages in the right format
- Obligation Service
  - Enforces the execution of the obligations

Architecture



OnGoing & Future Works

- Implementation of U-XACML framework
- Test of U-XACML framework on real scenarios

Trust

Outline

- Defs of trust
- Role-based Trust Management (RTML)
- RTML with weights
  - Josang topologies and RTML
- Implementation through soft contraints

(Too) Many meanings

- Trust is a generic keyword
  - Used in many communities with different meanings
    - E.g. Gambetta et al.: "*a particular level of the subjective probability with which an agent assesses that another agent or group of agents will perform a particular action, both before he [i.e. the trustor] can monitor such action (or independently of his capacity ever to be able to monitor it) and in a context in which it affects his [i.e. the trustor's] own action*"
    - E.g. Dimitrakos et al.: "*Trust of a party A in a party B for a service X is the measurable belief of A in B behaving dependably for a specified period within a specified context in relation to X...*"
    - E.g. Herbig et al., "*The estimation of the consistency over time of an attribute or entity*"
  - We are interested in computation aspects:
    - How trust is represented, how it is calculated, etc…
      - For us, just a weighted credential …

Trust in security

- Trust management for access control
  - Credentials, policies, access rules
    - Used for access control:
      - Your set of credentials meet my access policy
      - Crisp control: yes/no answers (or at most need more info)
        - » Trust negotiation
- Trust as quantitative notion
  - Metrics, functions, weights
    - Quantitative notions of trust
      - Recommendation/reputation models
      - Social notion of trust
        - » Soft controls: lack of trust / lack of social interaction
- Hybrid models
  - E.g. RTML with weights

UCON and Trust

- Trust may be used as a parameter to allow access to resources
  - Strictness of policy (e.g. ongoing usage) may depend on the trust level of subjects
- Trust may be updated based on authorization policy compliance
  - Users not compliant with policy (e.g. sending code not respecting specific constraints) may be revoked from usage

Trust-Management with credentials (TM)

- Access control based on delegation of rights
  - Usually distributed policies
- Elements
  - Principal attributes as permissions, roles, …
  - Credential issuers
  - Trust relationships
- Compliance Checker
  - Signed credentials to express principal statements, coded as logical rules
  - Credentials should entail access to the resource (logical deduction)

RTML Trust Management Framework

- Credential-based trust management
- Employ in open, distributed, heterogeneous environment

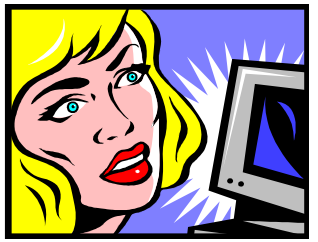
Language RT_0 (Role-based Trust Management)

- A family of languages for reasoning about trust relationships in distributed environments
  - A, B, C, D, … entities
  - r roles/attributes
- Rules:
  - A.r <- D means D has role r for A
  - A.r <- B.r_1 means if C has role r_1 for B then C has role r for A
  - A.r <- A.r_1.r_2 , means if B has role r_1 for A and if C has role r_2 for B then C has role r for A
  - A.r <- A_1.r_1 & A_2.r_2 if B has role r_1 for A_1 & R_2 for A_2 then B has role r for A

An example

Karadar

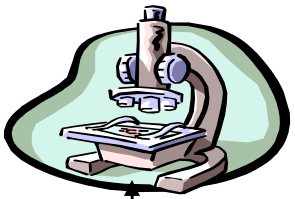
Alice



1. DI.stuID ← Alice

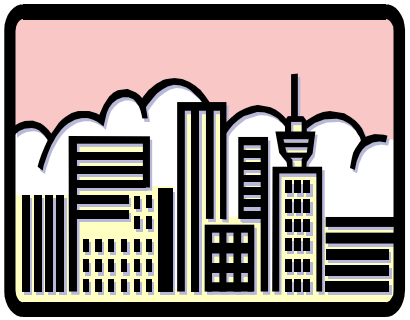
4. Karadar.university ← CRUI.accredited

5. Karadar.student ← Karadar.university.stuID



DI

2. UniPI.stuID ← DI.stuID



3 CRUI.accredited ← UniPI

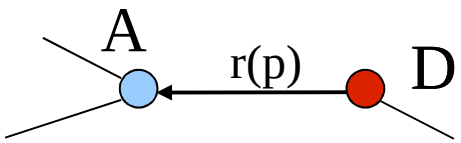
CRUI

UniPI

RTML (another view)

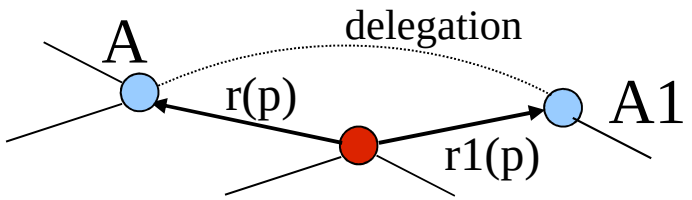
- There are four types of credentials (basic) in RT:
  - Simple Member:

    $A.\mathbf{r}(p) \leftarrow D$



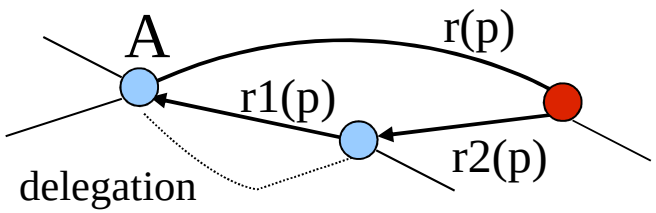
  - Simple Inclusion:

    $A.\mathbf{r}(p) \leftarrow A1.\mathbf{r1}(p)$



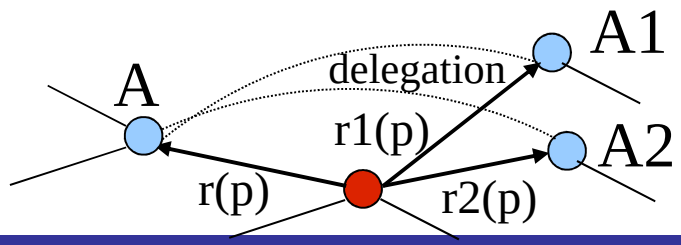
  - Linking Inclusion:

    $A.\mathbf{r}(p) \leftarrow A.\mathbf{r1}(p).\mathbf{r2}(p)$



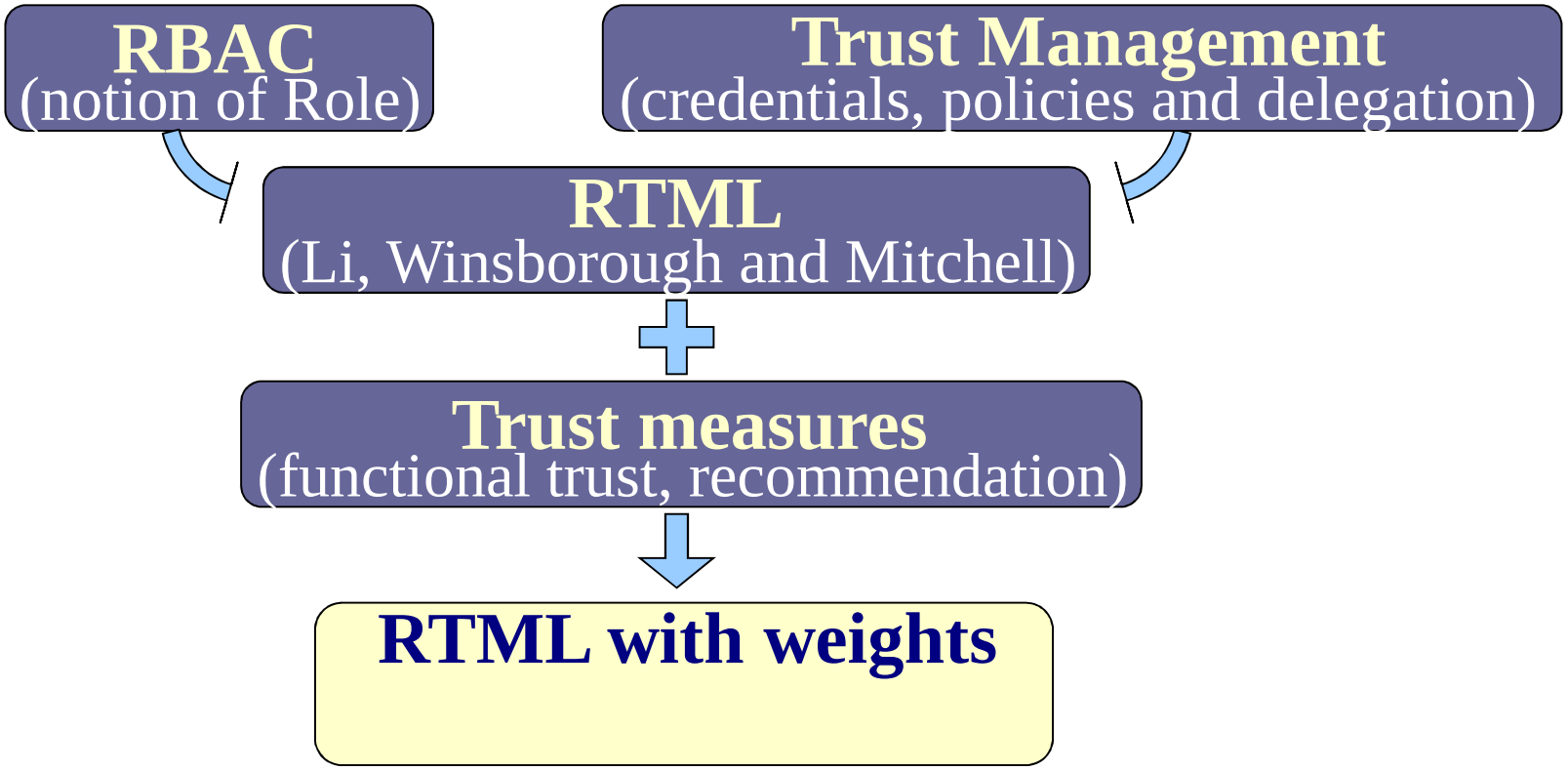
  - Intersection Inclusion:

    $A.\mathbf{r}(p) \leftarrow A1.\mathbf{r1}(p) \cap A2.\mathbf{r2}(p)$



Hybrid Trust Management Framework

- Credential-based and reputation-based trust management
- Employ in open, distributed, heterogeneous environment



RTML with weights

- The following credential:

$$A.\mathbf{r}(p,v) \leftarrow D$$

  - *A* assigns to *D* the parameterized role *r(p)*
  - *v* gives the measure of how much *A* places confidence in *D* enjoying *r(p)*

RTML with weights

- Explicit rules must be defined
  - Dealing with transitivity of trust
  - Presence of multiple paths
- We consider two operators:
  - ⊗ link
    - Combines opinions along paths
  - ⊙ aggregation
    - Combines opinions across paths

Using c-semirings

- A variant of semirings, an Algebra <A, $\odot$, $\otimes$, **0**,**1**>
  - $\odot$ is associative, commutative and **0** is the unit;
  - $\otimes$ is associative, commutative, distributes over $\odot$, and **1** is the unit, with **0** is it absorbing element;
  - $\otimes$ is inclusive and $\odot$ is idempotent;
  - $a <_W b$ iff $a \odot b = b$ (and it is total)

Several choices

- Some example of c-semiring-based trust measures:
  - We want to consider the path with maximal trust
    - $\otimes$ -> multiplication on real number
    - $\odot$ -> the maximum between two values
  - We want to consider the path with the minimal weak steps
    - $\otimes$ -> min
    - $\odot$ -> max
  - We want to consider the path with minimal number of steps
    - $\otimes$ -> sum on natural numbers
    - $\odot$ -> min

Another more complex c-semiring

- The operators are defined as follows:

$$(t_{ik}, c_{ik}) \otimes (t_{kj}, c_{kj}) = (t_{ik} t_{kj}, c_{ik} c_{kj})$$

$$(t_{ij}^{p1}, c_{ij}^{p1}) \oplus (t_{ij}^{p2}, c_{ij}^{p2}) = \begin{cases} (t_{ij}^{p1}, c_{ij}^{p1}) & \text{if } c_{ij}^{p1} > c_{ij}^{p2} \\ (t_{ij}^{p2}, c_{ij}^{p2}) & \text{if } c_{ij}^{p1} < c_{ij}^{p2} \\ (max(t_{ij}^{p1}, t_{ij}^{p2}), c_{ij}^{p1}) & \text{if } c_{ij}^{p1} = c_{ij}^{p2} \end{cases}$$

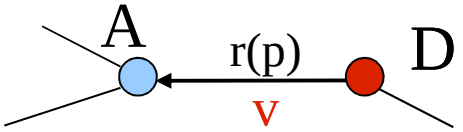
RTML with weights

Credentials enriched with trust measures (1):

- Simple Member:

  A.**r**(*p*,*v*)←D

  – A.r is covered with weight v

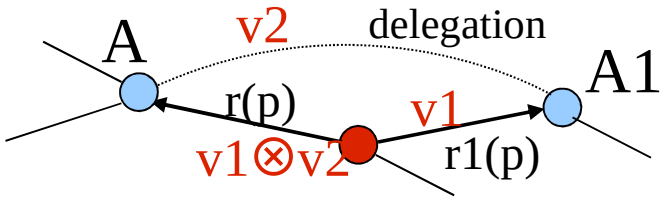


- Simple Inclusion:

  A.**r**(*p*) ←v2 A1.**r1**(*p1*)

  – All members in A1.r1 with

  v1 are members of A.r with weight v1⊗v2

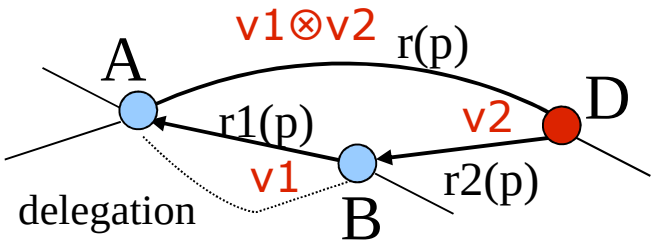


RTML with weights

Credentials enriched with trust measures (2):

- Linking Inclusion:

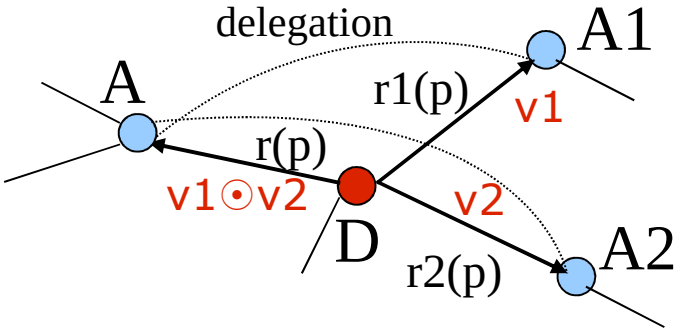
$A.\mathbf{r}(p) \leftarrow A.\mathbf{r1}(p1).\mathbf{r2}(p2)$



– If B has role A.r1 with v1 and D has role B.r2 with v2 then D has role A.r with v = v1⊗v2

- Intersection Inclusion:

$A.\mathbf{r}(p) \leftarrow A1.\mathbf{r1}(p1) \cap A2.\mathbf{r2}(p2)$



– If D has both A1.r1 with v1 and A2.r2 with v2 then D has A.r with v = v1⊙v2

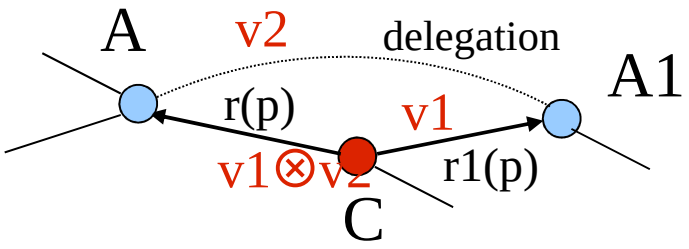
RTML with weights

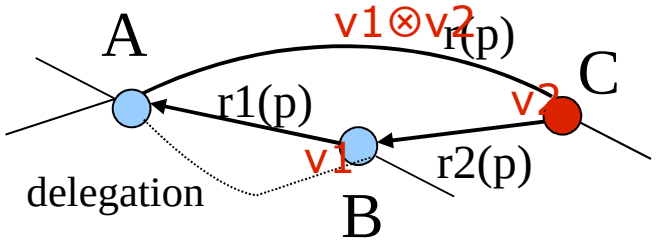
- For authorization answer a query:
  - Given an entity D, its credential and access rules, determine all the roles it is a member of with its max weight.

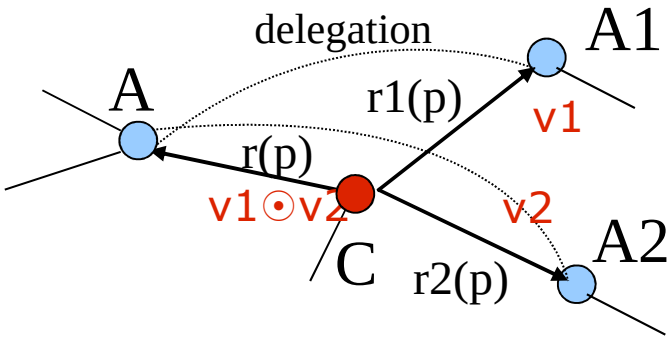
RTML with weights

Trust Calculations (simple creds, access rules)= {
Results:=simple creds; Changed := true;
While(Changed) {
Changed:=false;
For each A.**r** ←v2 A1.**r1** in rules *and*
for each A1.**r1**(v*1*)← C in simple creds
if A.**r** ← C *not* in simple creds, *or*
A.**r** ← C in simple creds with *not* v1⊗v2 ≤w v
then {
remove from simple creds all creds like A.**r** ← C;
insert A.**r**(v1⊗v2) ← C in simple creds;
Changed:=true };

For each A.r ← A.r1.r2 in rules and
for each A.r1(v1) ← B, B.r2(v2) ← C in simple creds
if A.r ← C not in simple creds, or
A.r(v) ← C in simple creds withnot v1 ⊗ v2 ≤w v
then {
remove from simple creds all the creds like A.r ← C;
insert A.r(v1 ⊗ v2) ← C in simple creds;
Changed:=true};
For each A.r ← A1.r1 ∩ A2.r2 in rules and
for each A1.r1(v1) ← C, A.r2(v2) ← C in simple creds
if A.r ← C not in simple creds, or
A.r(v) ← C in simple creds with not v1 ⊙ v2 ≤w v
then {
remove from simple creds all the creds like A.r ← C;
insert A.r(v1 ⊙ v2) ← C in simple creds;
Changed:=true};
}







RTML with weights for simplified reputation management

- RTML enriched with trust meanings and measures
  - Credentials express the fact that a principal trusts someone for:
    - Performing some functionality *f (attribute f)*
    - Giving a recommendation for performing *f (attribute rf)*

Josang's Topologies

(transitive trust model)

$$A \xrightarrow{f} D \qquad \text{(1) FUNCTIONAL TRUST}$$

$$A \xrightarrow{rf} D \qquad \text{(2) RECOMMENDATION}$$

$$\frac{A \xrightarrow{rf} B \quad B \xrightarrow{rf} D}{A \xrightarrow{rf} D} \qquad \text{(3) TRANSITIVITY}$$

$$\frac{A \xrightarrow{rf} B \quad B \xrightarrow{f} D}{A \xrightarrow{r} B \xrightarrow{f} D} \qquad \text{(4) FUNCTIONAL TRUST VIA RECOMMENDATION}$$

$$\frac{A \xrightarrow{r} B \xrightarrow{f} D \quad A \xrightarrow{r} C \xrightarrow{f} D}{A \xrightarrow{r} \{B\} \cup C \xrightarrow{f} D} \; B \notin C \qquad \text{(5) SET OF RECOMMENDERS}$$

(1) A trusts D for performing f
(2) A trusts D for recommending someone able to perform f
(3) Transitivity of recommendation
(4) The last step is a functional one. It maintains trace of the recommender
(5) C is a set.

A simplified Josang's model

- It looses information on the recommender

$$\frac{A \xrightarrow{rf} B \quad B \xrightarrow{f} D}{A \xrightarrow{f} D} \; (4^*) \text{ INDIRECT FUNCTIONAL TRUST}$$

Rules (4*) in place of (4) and (5) does not make sense anymore

Encoding the simplified Trust model into RT_0

	SIMPLIFIED TRUST MODEL	RT_0
(1)	$A \xrightarrow{f} D$	$A.f \leftarrow D$
(2)	$A \xrightarrow{rf} D$	$A.rf \leftarrow D$
(3)	$\dfrac{A \xrightarrow{rf} B \quad B \xrightarrow{rf} D}{A \xrightarrow{rf} D}$	$A.rf \leftarrow A.rf.rf$
(4*)	$\dfrac{A \xrightarrow{rf} B \quad B \xrightarrow{f} C}{A \xrightarrow{f} C}$	$A.f \leftarrow A.rf.f$

Explanation

(1) The role term f represents the capability of performing a service or a functionality that D is able to perform according to A. Thus, in A.f ← D, A defines D to have role f, i.e., that D is able to perform f.

(2) The role term rf represents the recommendation for doing a certain service or functionality. In A.rf ← D, A defines D to have role rf, i.e., that D is trusted for giving a recommendation to perform f.

(4*) B who has role A.rf is the recommender, i.e., A trusts B for choosing someone else that is trusted for performing f. C who has role B.f is trusted to perform f according to B. It follows that C is indirectly trusted to perform f according to A.

Deduction and abduction for RTML with weights with soft constraints

Joint work with S. Bistarelli and F. Santini

Weighted Datalog: Datalog^W

- Rules are the same as in Datalog

$$R_0(t_{0,1},\ldots,t_{0,k_0}) :\text{-} R_1(t_{1,1},\ldots,t_{1,k_1}),\ldots,R_n(t_{n,1},\ldots,t_{n,k_n})$$

- Facts slightly change; they are in the form:

$$R_i(x_{i,1},\ldots,x_{i,k_i}) :\text{-}\ a$$

Taken from the semiring set!

Extending RT_0: RT_0^W

- A *role* in RT_0 (and RT_0^W) takes the form of an entity (*A*) followed by a role name (*R*) separated by a dot = *A.R*
- Each entity *A* has the authority to define who are the members of each role of the form *A.R*
  - IEEE.member ← Alice
- Each statement defines one role to contain:
  - An entity
  - Another role: IEEE.member ← IEEE.studentMember
  - Other expressions

    that evaluate to a set of entities

$$\left\{\begin{array}{c} \texttt{EPub.disct} \longleftarrow \texttt{EPub.preferred} \cap \texttt{EPub.student} \\ \texttt{EPub.preferred} \longleftarrow \texttt{EOrg.preferred} \\ \texttt{EOrg.preferred} \longleftarrow \texttt{IEEE.member} \\ \texttt{EPub.student} \longleftarrow \texttt{EPub.university.stuID} \\ \texttt{EPub.university} \longleftarrow \texttt{ABU.accredited} \\ \texttt{ABU.accredited} \longleftarrow \texttt{StateU} \\ \texttt{StateU.stuID} \longleftarrow \texttt{Alice},\ \texttt{IEEE.member} \longleftarrow \texttt{Alice} \end{array}\right\}$$

4 Rules for RT_0^W

4 Rules for RT_0

- Rule 1: $A.R \longleftarrow \{B, s\}$ ≡ RT RT^W
- Rule 2: $A.R \longleftarrow B.R_1$
- Rule 3: $A.R \longleftarrow A.R_1.R_2$
- Rule 4: $A.R \longleftarrow B_1.R_1 \cap B_2.R_2 \cap \cdots \cap B_n.R_n$

IEEE.member ← <Alice, 1>

Taken from the semirng set!

- Translated in Datalog^W as:
  - Rule 1 $r(A, B)$:- s
  - Rule 2 $r(A, x)$:-$r_1(B, x)$
  - Rule 3 $r(A, x)$:-$r_1(A, y), r_2(y, x)$
  - Rule 4 $r(A, x)$:-$r_1(B_1, x), r_2(B_2, x), \ldots, r_n(B_n, x)$

member(IEEE, Alice) :- 1

Soft Constraint Systems (1)

- Given *V*, *D*, and $S = \langle A, +, \times, 0, 1 \rangle$ a constraint is

$$C = \eta \rightarrow A$$

- Constraint composition:

$$\otimes : C \times C \rightarrow C \quad (c_1 \otimes c_2)\eta = c_1\eta \times c_2\eta$$

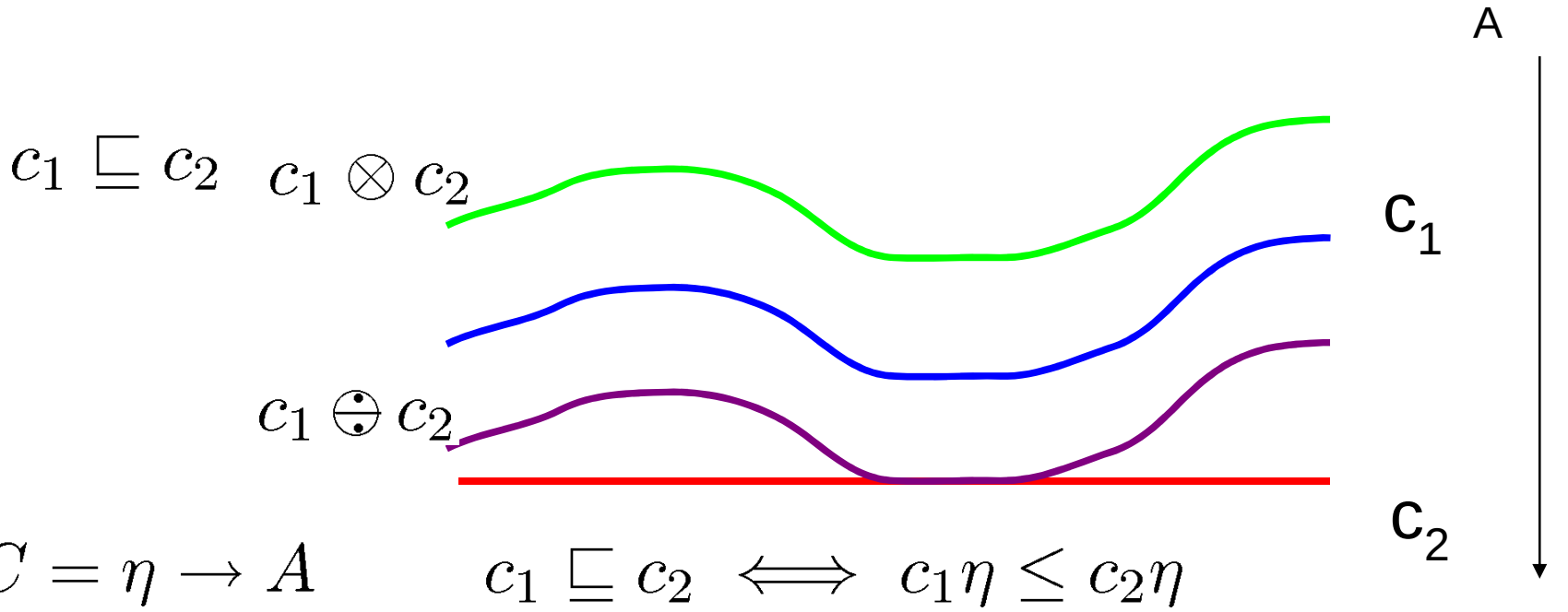
- To remove a preference value from another one we can define a $\div$ operator, weak inverse of $\times$

$$a \div b = min\{x \mid b \hat{+} x \geq a\} = \begin{cases} 0 & \text{if } b \geq a \\ a \hat{-} b & \text{if } a > b \end{cases}$$

Soft Constraint Systems (2)

- Constraint division

$$\oslash : C \times C \to C \quad (c_1 \oslash c_2)\eta = c_1\eta \div c_2\eta$$



$$C = \eta \to A \qquad c_1 \sqsubseteq c_2 \iff c_1\eta \leq c_2\eta$$

Using soft constraints to implement two basic services

Deduction and Abduction

Deduction with soft constraints

$$\sigma = policy \otimes credentials$$

Definition 1. [Deduction with soft constraints] *Given a soft constraint store σ which represents the current knowledge and a soft constraint c, if $\sigma \sqsubseteq c$ then the store entails (i.e. deduces) c.*

$\sigma = Student(Alice) \otimes HighMarks(Alice) \otimes (Student(x) \wedge HighMarks(x) \sqsubseteq Access(x))$

$$\sigma \sqsubseteq Access(Alice)$$

Abduction with soft constraints

Definition 2. [Abduction with soft constraints] *Given a soft constraint store σ and a constraint c, then the abduction process is aimed at finding a constraint d, such that $\sigma \otimes d \sqsubseteq c$, i.e. $d = c \oplus \sigma$.*

$$\sigma = Student(Alice) \otimes (Student(x) \wedge HighMarks(x) \sqsubseteq Access(x))$$

$$c = Access(Alice)$$

What you have

What you want

$$d = HighMarks(Alice)$$

What you need more to enter

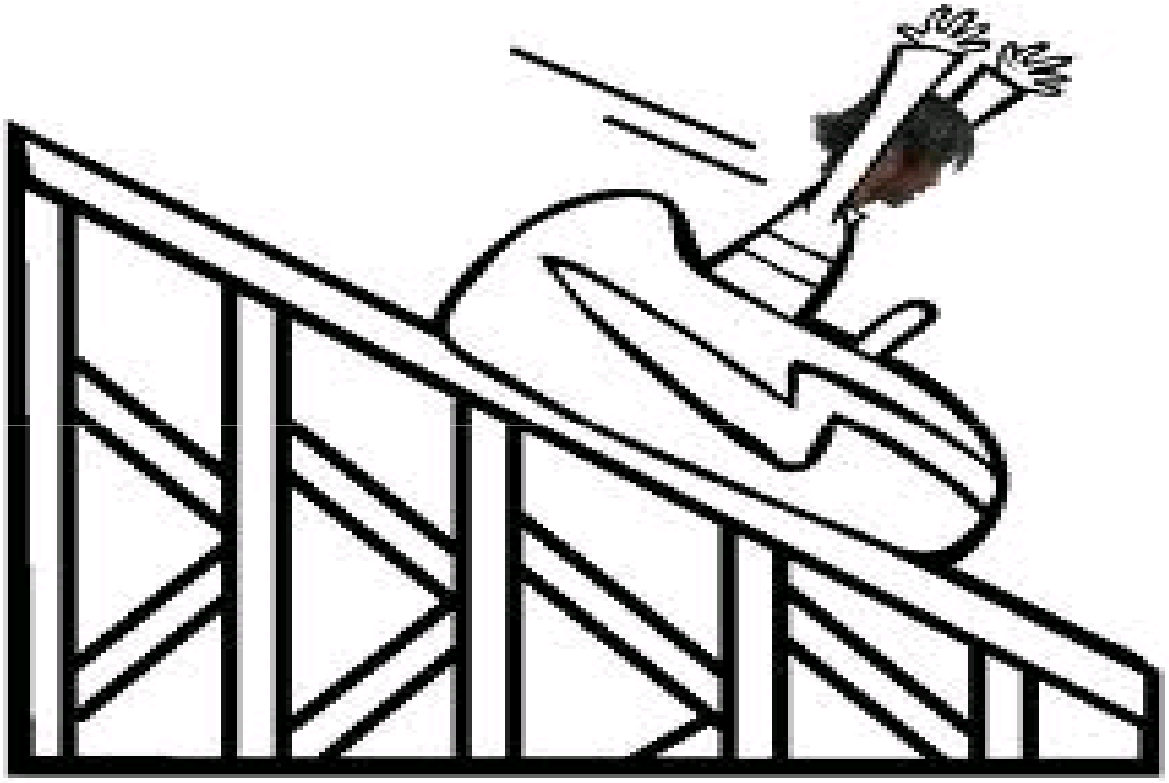
$$\sigma(x = Alice) = 7, c(x = Alice) = 10$$

$$d(x = Alice) = 3$$

Conclusions and FW

- Role-based authorization scheme based on trust scores
  - Parametric with the chosen semiring
- Two basic services
  - Deduction
  - Abduction
- Implementation of these services in CHR
- Towards a negotiation architecture (fw)

Risk



Rationale for including risk

- Many decisions are based on fuzzy (unreliable, changing, imprecise) data.
- Not necessarily all the security constraints are "continuously" fulfilled.
- **Main direction**: empower UCON model with risk assessment.

Outline

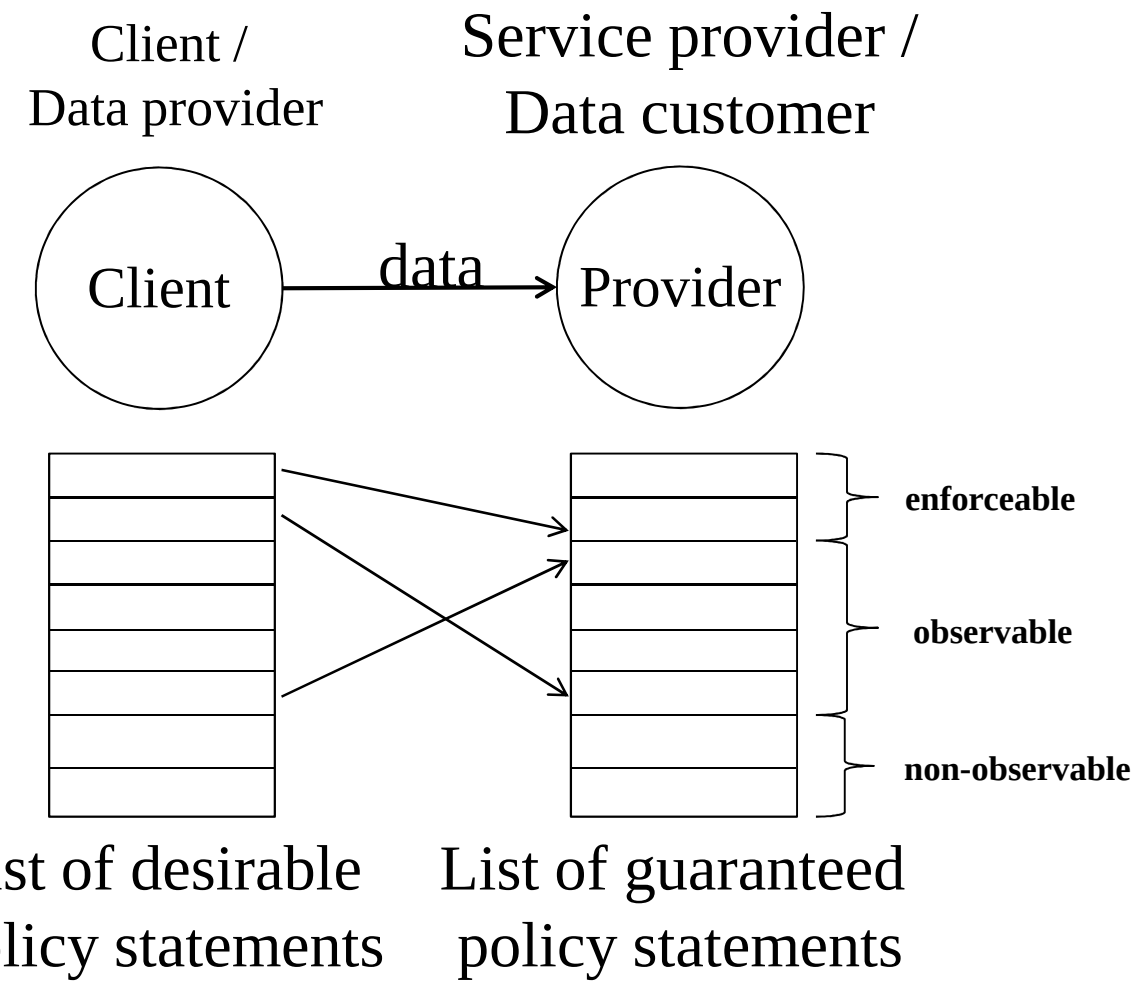
- Risk-Based Usage Control for Service Oriented Architecture
  - Foucs on appropriate security protection policy choice in distributed usage control
- Risk aware Usage Decision Making in Highly Dynamic System
  - Focus on uncertainty on the freshness of attribute values

Risk-Based Usage Control for Service Oriented Architecture

UCON in SOA

1. Client's data is processed by service provider/data consumer.
2. Client's data must be processed in a less risky way (from client perspective).
3. Client needs to select a service provider/data consumer.
4. After selecting a provider the level of risk should not become much higher.

The scenario



Goals

Main goals:

- Select the provider with the best security of data processing.
- Analyze riskiness of selected service provider during data processing.

We propose a method to empower usage control with a risk-based decision making.

Additional goal:

- Help service provider to reduce its risk level.

Example

- The client wishes that its data is deleted after 1 day
- We have:
  - One service provider/data consumer has an internal policy that cancel data after 7 days
    - Not perfect matching, although better that the other
  - Another service provider/data consumer has an internal policy that cancel data after 15 days

Plan of risk assessment

1. Analyze importance of desirable policies.
2. Map desirable policies to guarantied policies.
3. Compute strength of desirable policies.
4. Calculate the risk of policies violation using importance and strength.

Risk

Risk is the possibility of harm or loss.

Risk = Threat x Vulnerability x Impact

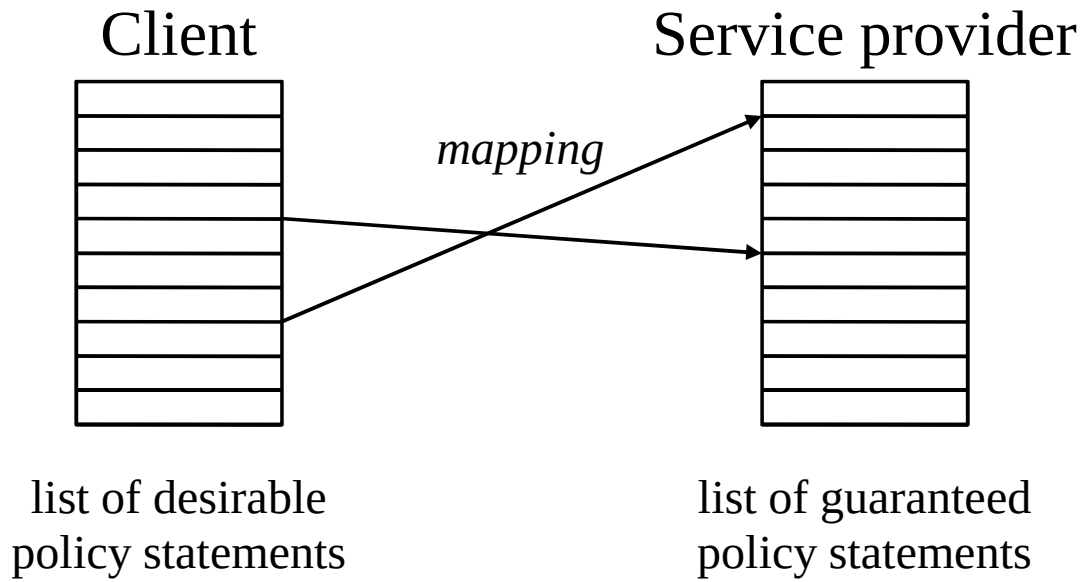
Importance of policies

Client assesses *importance* of his policy statements.

$$rank = Threat \times Impact$$

$$rank(policy) : P_d \rightarrow \{low, medium, high\}$$

Desirable and guaranteed policy statements



Strength of policies

Client assesses *level of strength* of its policy statements.

$$str(policy) : P_d \times 2^{Attributes} \rightarrow \{perfect, high, medium, low, unacceptable\}$$

Level of strength gives *vulnerabilities*.

$$str(policy) \rightarrow str^{-1}(policy) = Vulnerable$$

perfect → non vulnerable

high → low

medium → medium

low → high

unacceptable → unacceptable

Qualitative calculation of risk

$$risk(policy) = str^{-1}(policy) * rank(policy)$$

Qualitative values can be multiplied using the *matrix*.

rank\vulnerability	**perfect**	**low**	**medium**	**high**	**unacceptable**
low	*no risk*	*low*	*low*	*low*	*unacceptable*
medium	*no risk*	*low*	*medium*	*medium*	*unacceptable*
high	*no risk*	*low*	*medium*	*high*	*unacceptable*

Risk-aware UCON (client side)

- Authorization:
  - compare tuples of different providers <high, medium, low>;
  - choose the best one and allow access to the data.

- Usage control:
  - determine new values of attributes;
  - calculate current risk level (attributes → strength → risk);
  - make a decision about usage session.

Service provider side

Service provider can improve its data management practices.

Issues:

- limited budget;
- maximize reduction of risk.

This is a knapsack problem.

Service provider side

Risk mitigation plan:

- risks are reduced,
- number of medium with ,
- finally, low risks are reduced.

Conclusion

- The propose approach allows:
  - Selection of a provider with less risky policies.
  - Flexible automatic re-assessment during usage sessions.
  - Overall state of riskiness is considered.
  - Possibility for the service provider to optimize its offer

- Future work:
  - Trust
  - Certification of guaranteed policies

Risk-aware Usage Decision Making in Highly Dynamic System

Motivation

- Current systems are more and more dynamic (e.g., Web Services, Clouds, Grid, mobile networks).
- After granting access attributes can change.
- Checking attributes very frequently is impossible or inefficient.

•Result: **wrong access decision** caused by not fresh attributes.

Goals

- Develop a framework for:
- taking into account various uncertainties;
- predicting current values of attributes;
  - And consequently of policies
- making a rational access decision.

Uncertainties

- Unintentional:
  - timeliness
    - frequency of updates
  - currency
    - natural delays in delivery
- Intentional:
  - trustworthiness of data

Assumptions

1. We consider authorizations and conditions (not obligations).

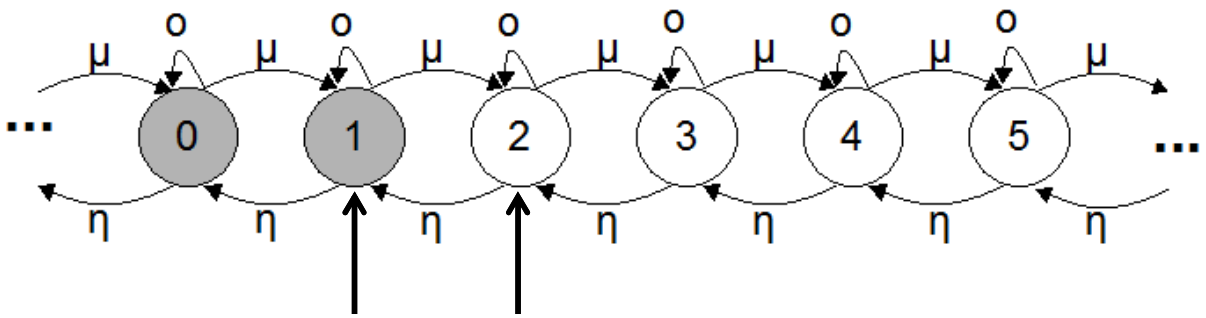
2. Attributes are independent.

3. Statistics is available and representative.

Markov chains

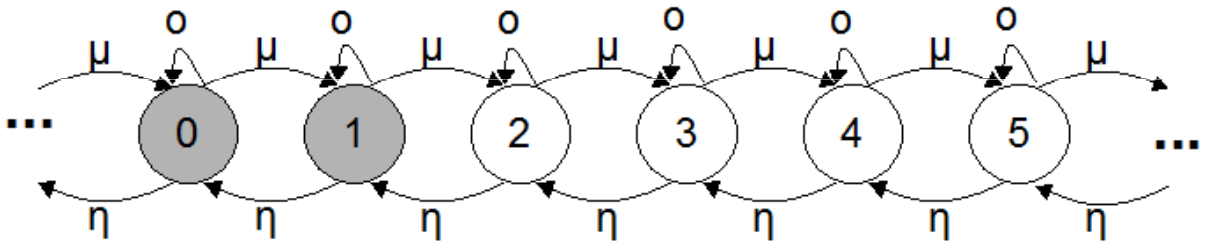
- A Markov chain is associated with every attribute (policy).

- Example: the rating of seller at an on-line auctior



Computation of probabilities using discrete-time Markov chains

- The model can be used to predict probabilities to reach an unwanted state (**state 1**) if number of transitions **N** is known.

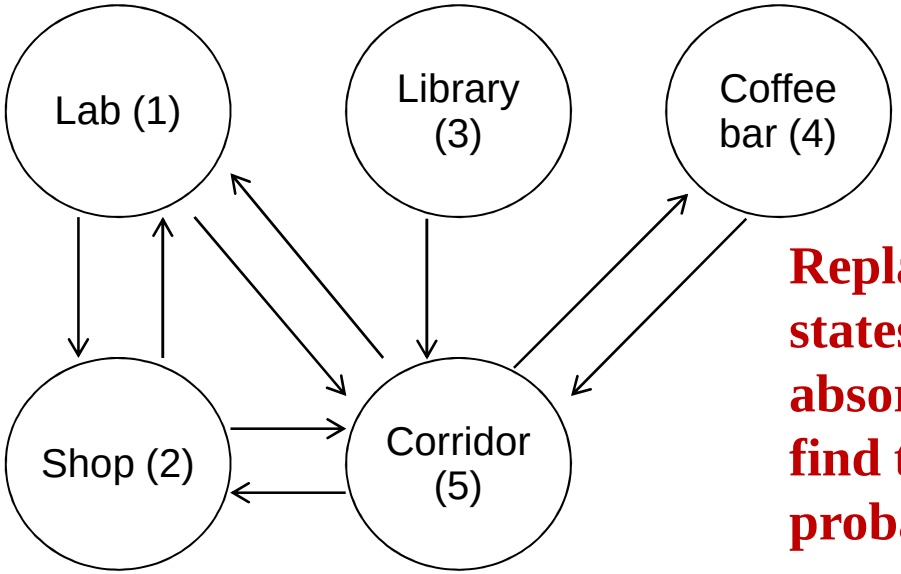


Time = 0	**0**	**0**	**1**	**0**	**0**	**0**
Time = 1	**0**	**η**	**o**	**μ**	**0**	**0**
Time = …		**……**				
Time = N	**0**	p_V	p_1	**p**	p_3	p_4

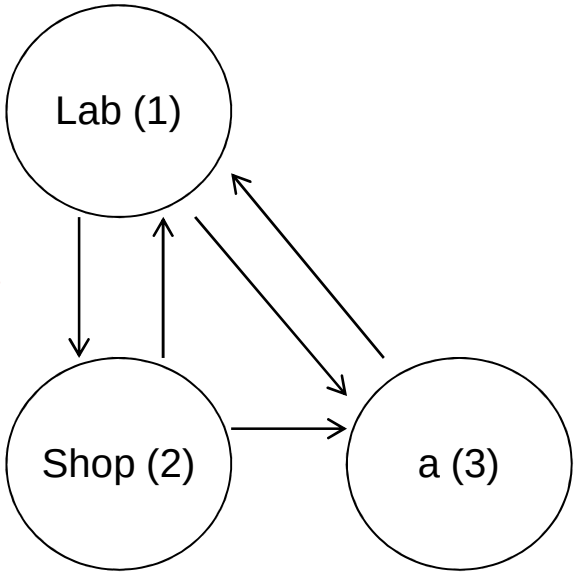
Computation of probabilities using continuous-time Markov chains

- The model can be used to predict probabilities to reach an unwanted state (**state 1**) if only time **Δt** after last check is known.

Example: the location of a person at a factory.



Replace unwanted states with absorbing state to find the probability.



Decision Making

- Decision matrix

	Policy satisfied ($1-p_V$)	**Policy failed (p_V)**
Continue access	C^{CS}	C^{CF}
Revoke access	C^{RS}	C^{RF}

- Decision: **allow**.
- If it is more profitable to allow access than revoke:
  - $(1 - p_V) * C^{CS} - p_V * C^{CF} > p_V * C^{RF} - (1 - p_V) * C^{RS}$
- Decision: **deny (or alarm).**
- Otherwise

Continue

Revoke

Risk Mitigation

- "Revoke" decisions could be different:

- Deny access (simple).
- Suspend access until fresh values are not received.
- Additional check is required.
  - E.g. impose it as on obligation
- Mark the session as potentially fraudulent (for further review).

Combination of probabilities

The complex policy is the policy of several atomic rules aggregated using operators **AND**, **OR**, **NOT**.

Atomic rule (**α**, **β**) is a rule of one attribute and one constraint.

γ = α AND β :

$$p_\gamma = p_\alpha \oplus p_\beta = p_\alpha * (1 - p_\beta) + (1 - p_\alpha) * p_\beta + p_\alpha * p_\beta = p_\alpha + p_\beta - p_\alpha * p_\beta$$

γ = α OR β :

$$p_\gamma = p_\alpha \otimes p_\beta = p_\alpha * p_\beta$$

γ = NOT p_α :

$$p_\gamma = \neg p_\alpha = 1 - p_\alpha$$

Combination of costs

- No matter which atomic rule fails

  assign new losses and benefits for whole policy;

- Fine-grained analysis is required

  assign costs for each rule and combine them.

Combination of costs C^{CF}

- Only losses caused by allowing of access C^{CF} are policy specific.

Suppose **α** and **β** are atomic rules with costs of violation C_{α}^{CF} and C_{β}^{CF}, the combined risk R_{γ}^{CF} of violation of complex rule **γ**.

γ = α
AND β :

$$R_{\gamma}^{CF} = C_{\alpha}^{CF} * p_{\alpha} + C_{\beta}^{CF} * p_{\beta}$$

γ = α OR β :

$$R_{\gamma}^{CF} = \left(C_{\alpha}^{CF} + C_{\beta}^{CF}\right) * p_{\alpha} * p_{\beta}$$

γ = NOT p_{α} :

The rule Influences only probabilities, does not influence the risk.

Status and Future work

- The basic framework for dealing with uncertainties in UCON is provided.
- The UCON model is empowered with risk analysis to become more flexible.

- Future work:
  - Implement the framework and check its complexity.
  - Focus on intentional uncertainties.

To sum up

- We developed a framework for
  - Usage Control (UCON) in GRID systems (prototype)
  - UCON for computational services (prototype)
  - Trust management engines (prototype)
  - UCON and risk (theory)

- A lot of work remains to be done!!!

Some selected references

- On Usage Control for GRID Systems (FGCS 2010)
- Risk-Based Usage Control for Service Oriented Architecture (PDP 2010)
- Usage control in computer security: A survey Computer Science Review 2010
- Controlling the Usage of Grid Services International Journal of Computational Science 2009
- A proposal on enhancing XACML with continuous usage control features CoreGRID 2009
- Enhancing grid security by fine-grained behavioral control and negotiation-based authorization IJIS 2009
- A Semantic Foundation for Trust Management Languages with Weights: An Application to the RTFamily ATC 2008
- A Secure Environment for Grid-Based Supply Chains eChallenge08 2008
- Fine Grained Access Control with Trust and Reputation Management for Globus (GADA 2007)
- On the Relationships between Two trust Management Models (VODCA2006)
- Fine Grained and History-based Access Control with Trust Management for Autonomic Grid Services (ICAS 2006)
- Towards Continuous Usage Control on Grid Computational Services. (ICAS2005)
- Improving Grid Services Security with Fine Grain Policies. (GADA 2004)

Supporting EU projects

- GRIDtrust Project
  - www.**gridtrust**.eu/
- CONSEQUENCE project
  - http://www.**consequence**-project.eu/

Contacts and advertisement

- http://security.iit.cnr.it

- 4 new positions (Post-doc/Ph.D. students) are available in the two new EU projects:
  - NESSoS: NoE on Engineering Secure Future Internet Services and systems
  - ANIKETOS: IP on Trustworthy and secure composite services
  - Areas:
    - Formal methods in security, trust management, service oriented architectures.
- 7th International Workshop on Formal Aspects in Security and Trust (Pisa, 16-17 Sept. 2010)