On Usage Control

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

- Usage Control (UCON) on
  - GRID
  - CLOUD
  - Network
  - Mobile
- UCON and trust
- UCON and risk
- Conclusion

GRID/Cloud

Grid Computing

- Large scale
- Cross-organizational
- Geographical distribution
- Distributed Management

Cloud Computing

- Computing, Software, Infrastructures as “services”
- Virtualization of resources
- Elasticity/Scalability
- No knowledge of provider

Security is a main challenge in these environment

- In addition to usual problems of distributed systems
  - Virtualization
  - Multiple administration domains
  - Lack of control
  - Different stakeholders interested to protect different assets
  - …

Access control

Access control

- Goal:
  - Evaluate access request to resources and allow or not the access (grant/deny)
- Access control models:
  - DAC, MAC, RBAC, Attribute-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
  - …
- Access 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 regulate the access to the resources
  - High level specification:
    - **Open** policy: if it is not explicitly forbidden, then it is allowed
    - **Closed** policy: if it is it not explicitly allowed, then it is forbidden
- Mechanism:
  - IT instrument to implement and enforce the access policy
    - Generic/programmable
    - Hard-wired

Policies and mechanisms (2)

- We should consider both:
  - A 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 encapsulates (logically) the resource
- Tamper-proof
  - It should be impossible to modify its behavior
- Not-bypassable
  - All the accesses to the resources must pass through the reference monitor
- Security-Kernel
  - 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

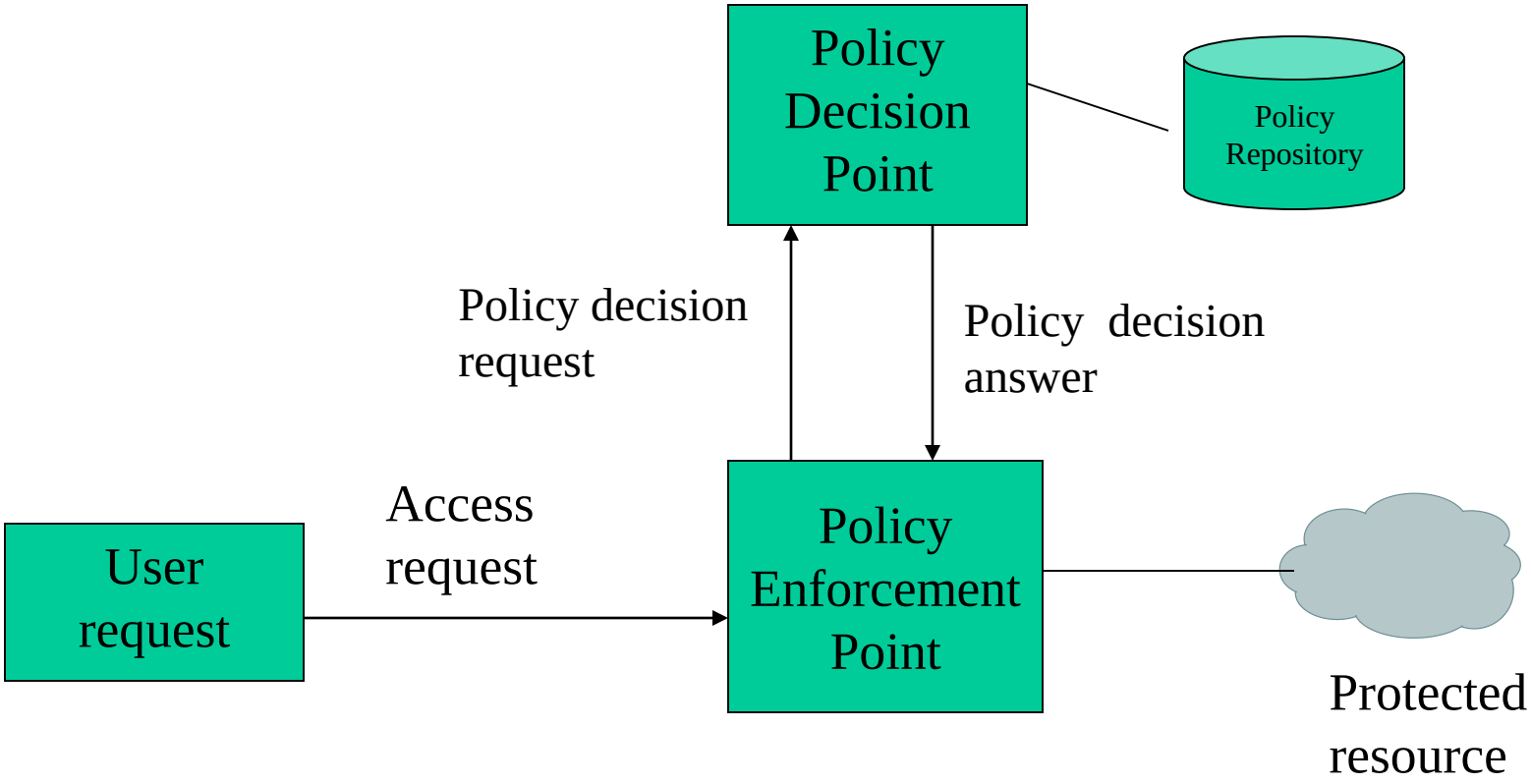
Policies (constraints)

- 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) demands formal methods analysis

Common access control policy models

- Discretionary Access Control (DAC)
  - Resource owner control access to their resources
  - Unix security/Windows
- 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

Some authorization frameworks

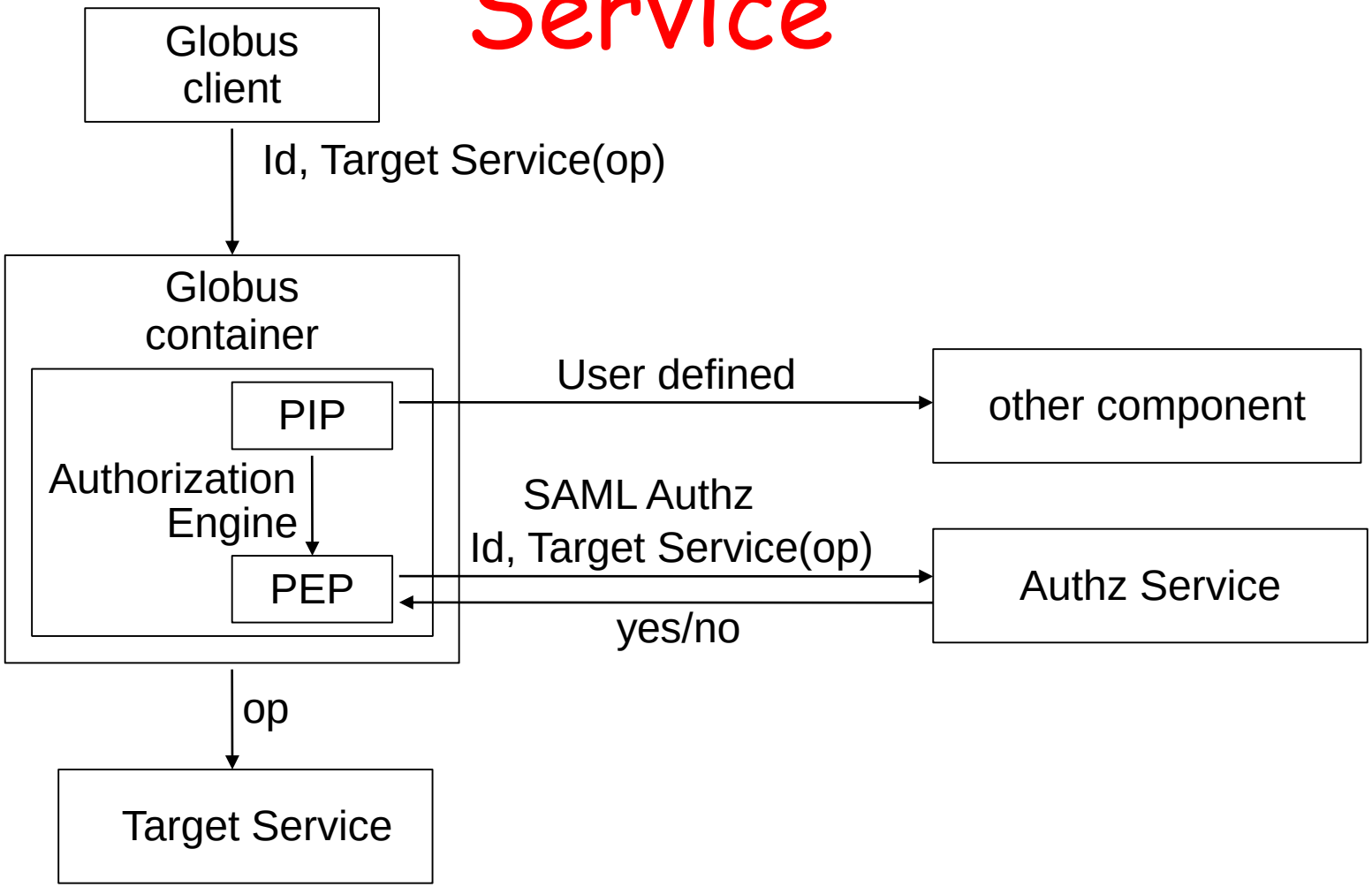
Globus toolkit

- GRID implementation
- Produced by Globus alliance
  - Compliant with Open Grid Services Architecture (OGSA)
- 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
  - …
- A common feature is the lack of further control after granting access to resources

Community Authorization Service

- CAS manages a data base of Virtual Organization (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 cert. 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

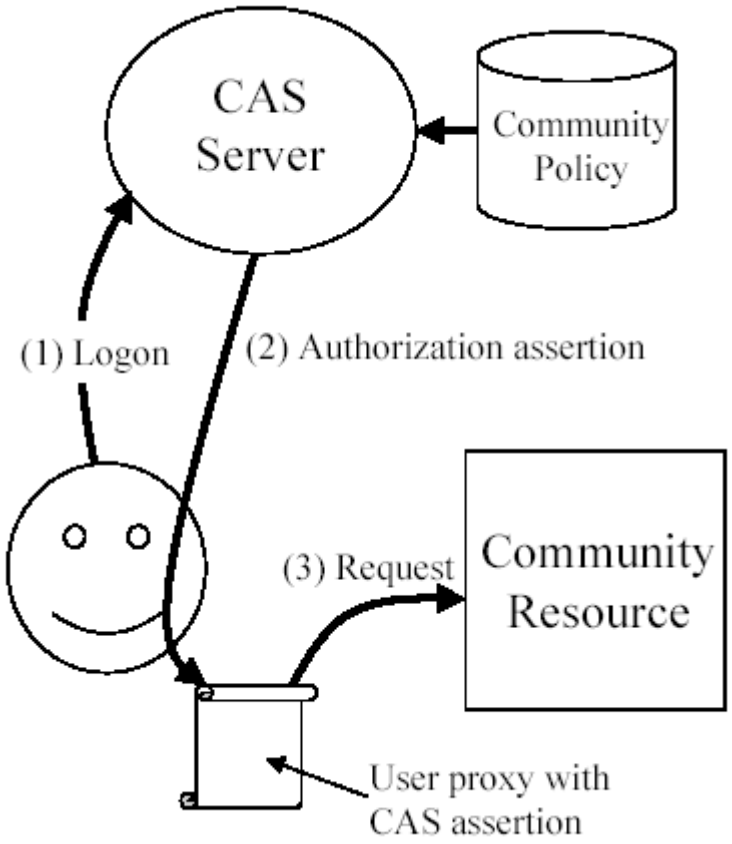
Valid: 3/25/03 13:00 –15:00

These actions are allowed:

Read gridftp://myhost/mydir/*

Signature (of assertion, by the VO CAS server)

Signature (of all above, by the user)



PERMIS

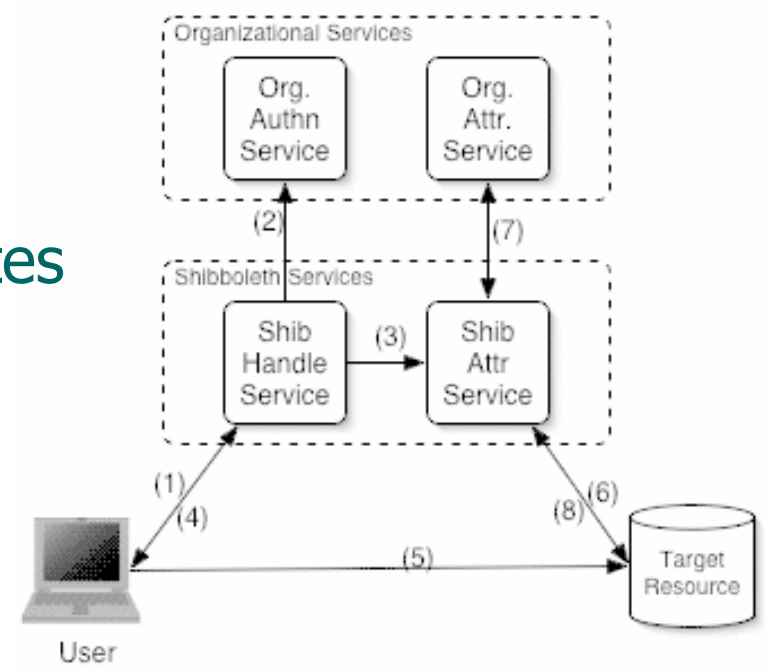
- Role Based Access Control Auth. Infrastructure
- Runs as Globus Service
  - SAML Callout Authz Service
- X.509 Attribute certificates to assign roles to users
  - Issued by an Attribute Authority
- X.509 Attribute certificates to store the policy
  - Definition of roles and permissions (XML)
  - Issued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

Akenti

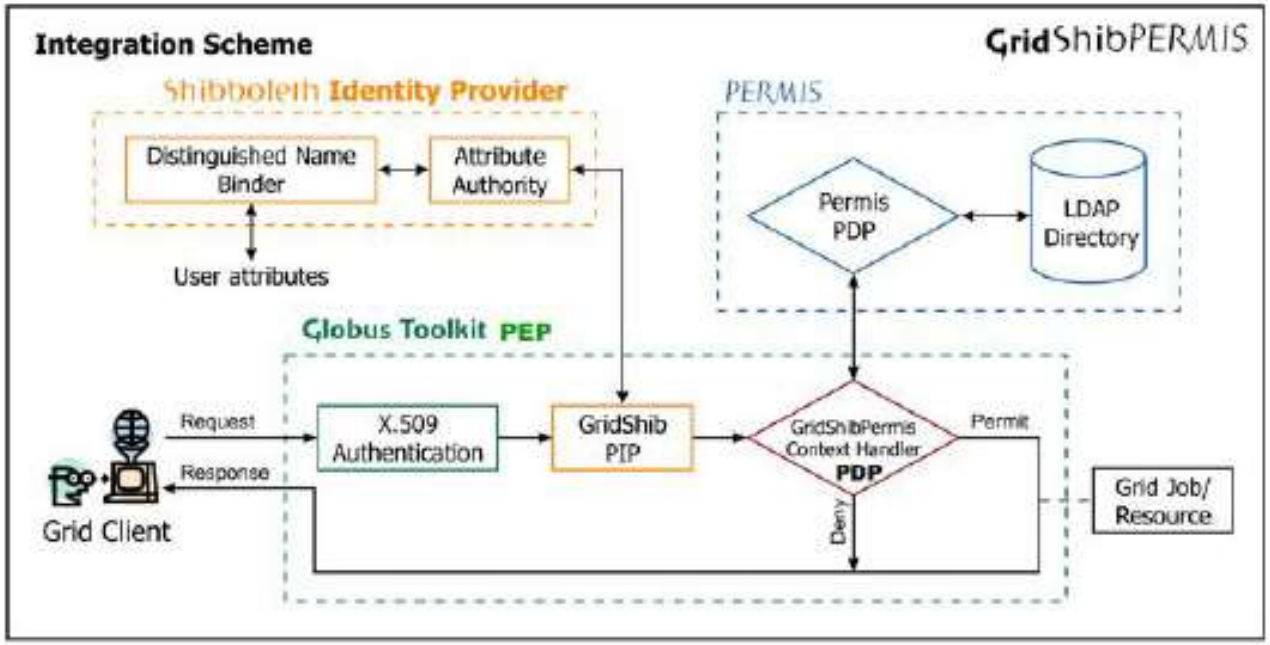
- Distributed authorization system where certificates are created independently by distinct stakeholders
- Certificates type
  - Authentication
  - Policy certificates
    - Specifies the Source of Authority for resources
  - Use condition certificates
    - Constraints to access resources
  - Attribute 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



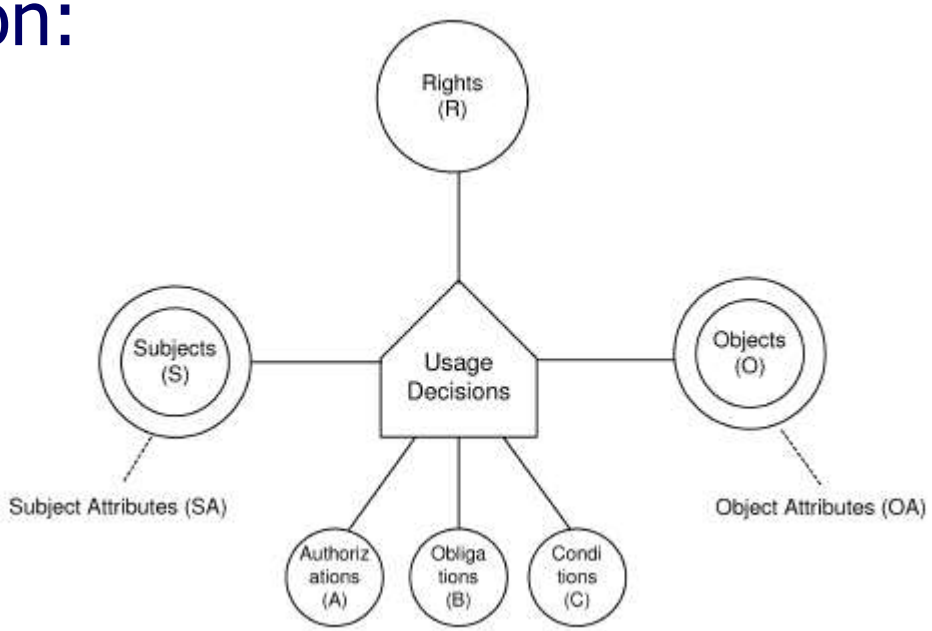
Usage Control

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”

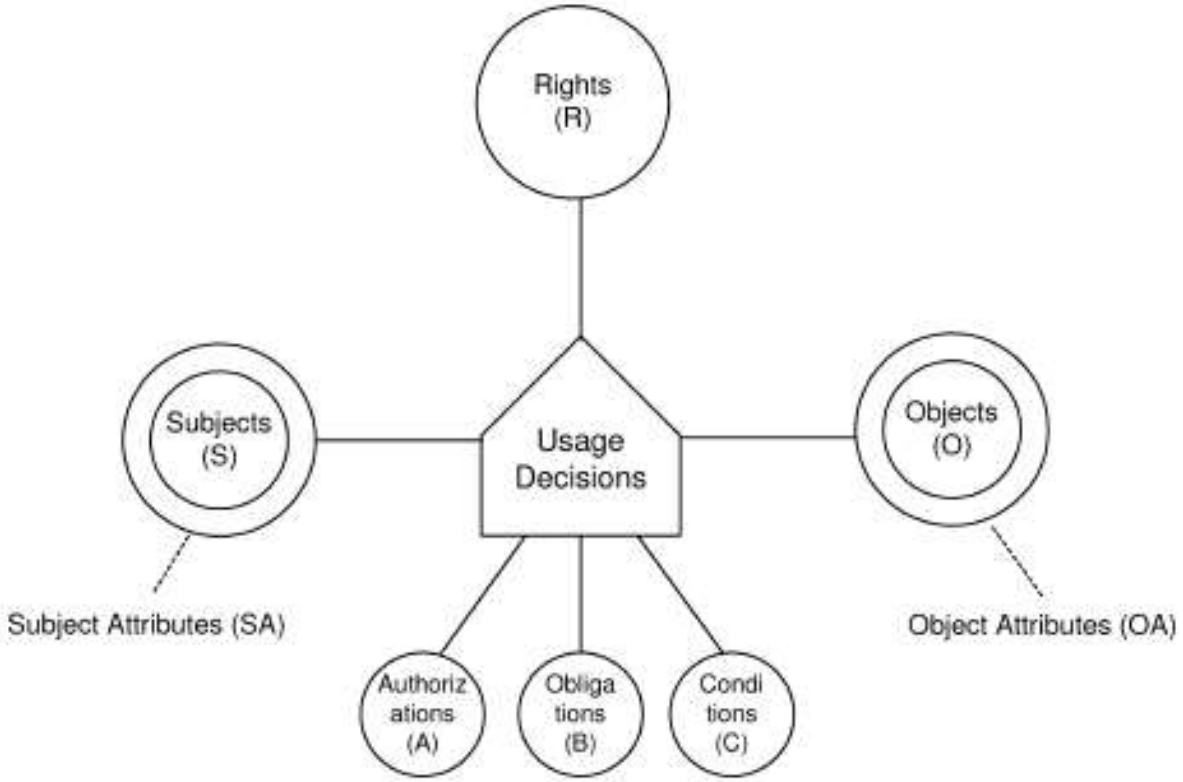
Obligations

- Functional predicates about 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.
  - During access each 15 mins the user should ping the system
- Note that obligations do not simply correspond only to actions from the subject that requested the access to the resource (Relevant feature!)

Conditions

- Environmental or system based decision factors
  - Not directly related with Subjects and Objects
  - e.g.
    - Current local time
    - Current overall 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.

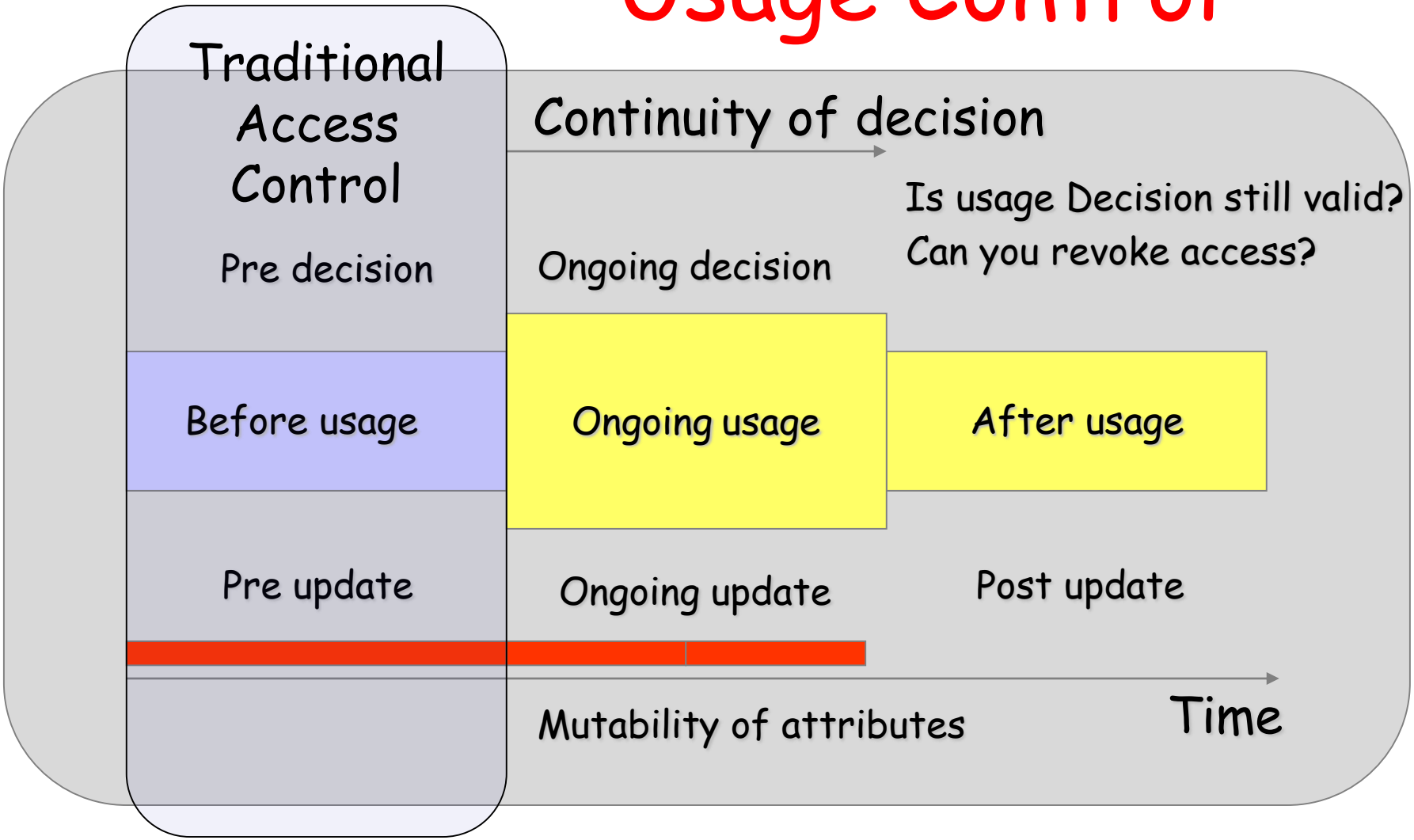
Usage Right Evaluation



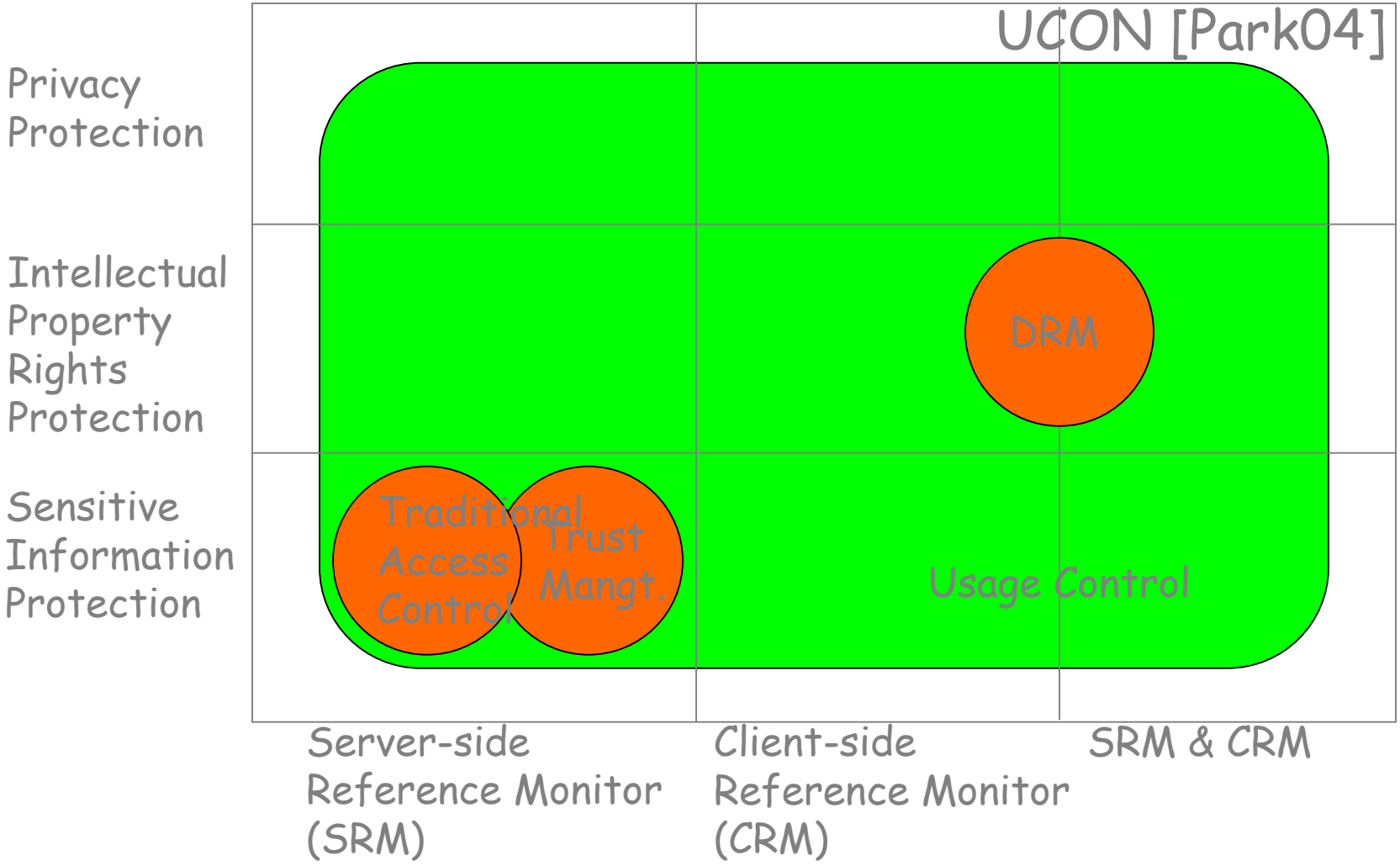
Continuity: A main UCON feature

- 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 at least 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)

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)}

Usage control

- GRID
- Cloud
- Network
- Mobile (WIP)

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/Web 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 do not 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)))
```

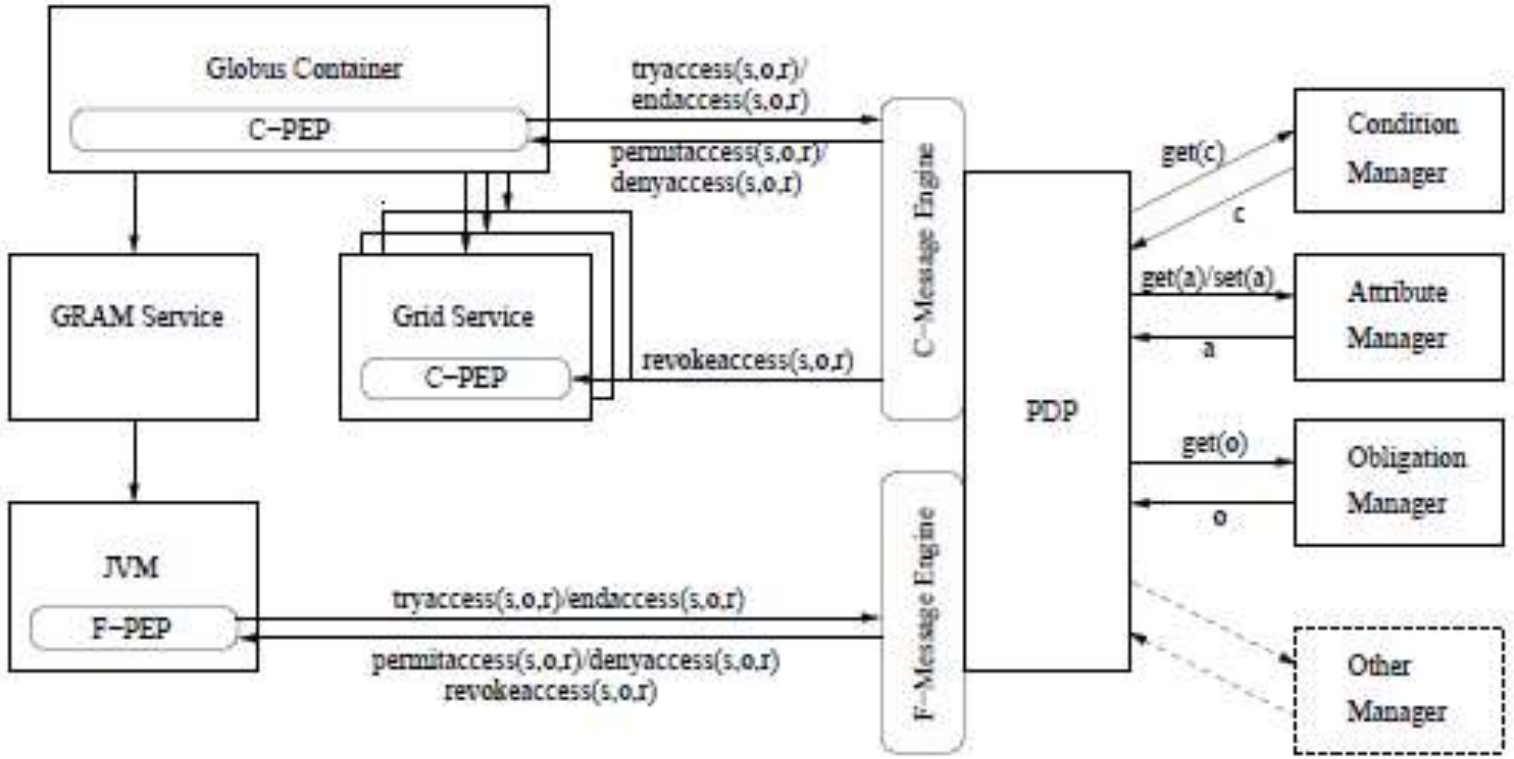
UCON in GRID

Our focus on …

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

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)

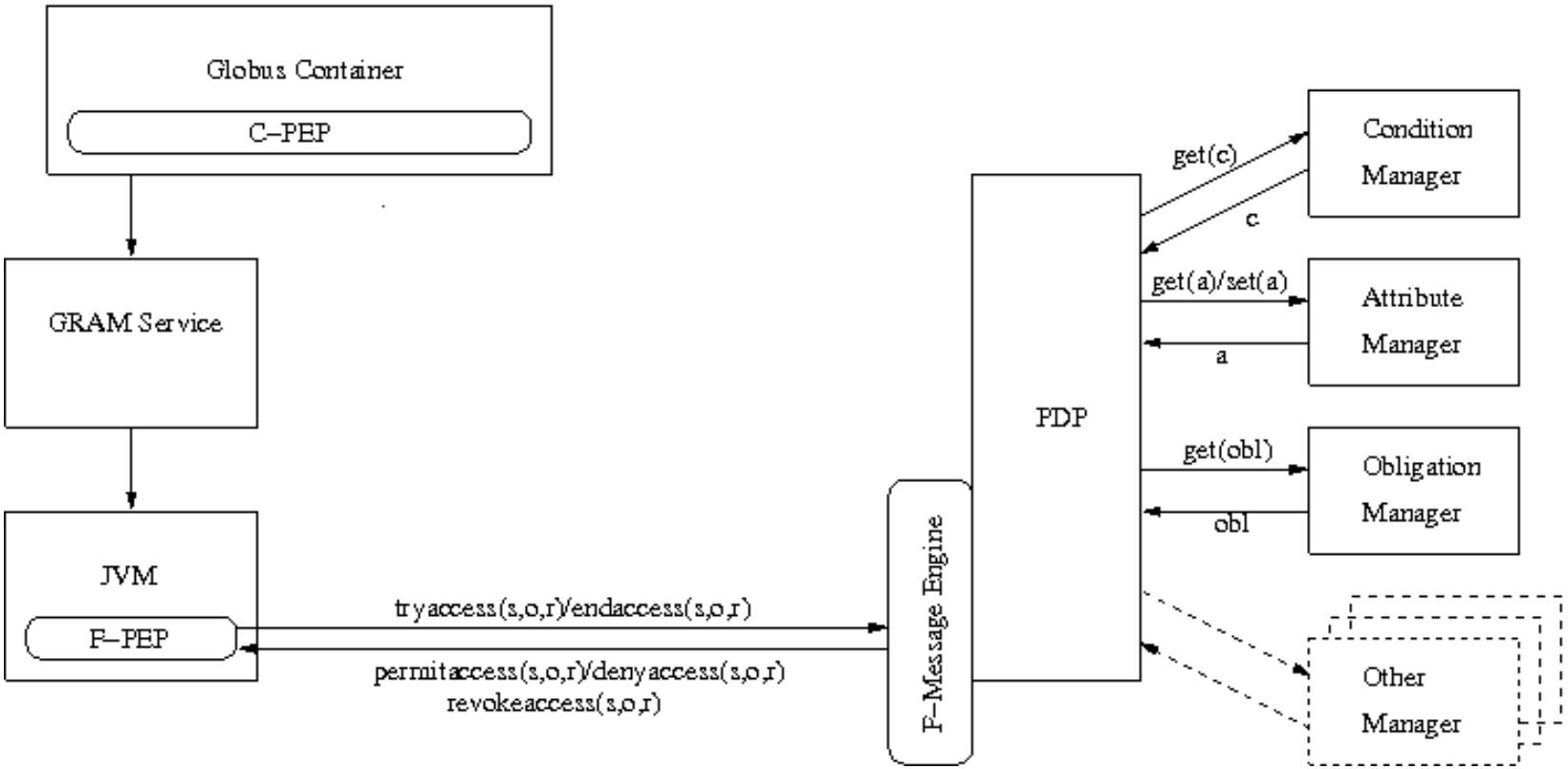
To monitor the application, we check the system calls executed by the JVM

The PEP is in the JVM

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

Fine grain history based usage control

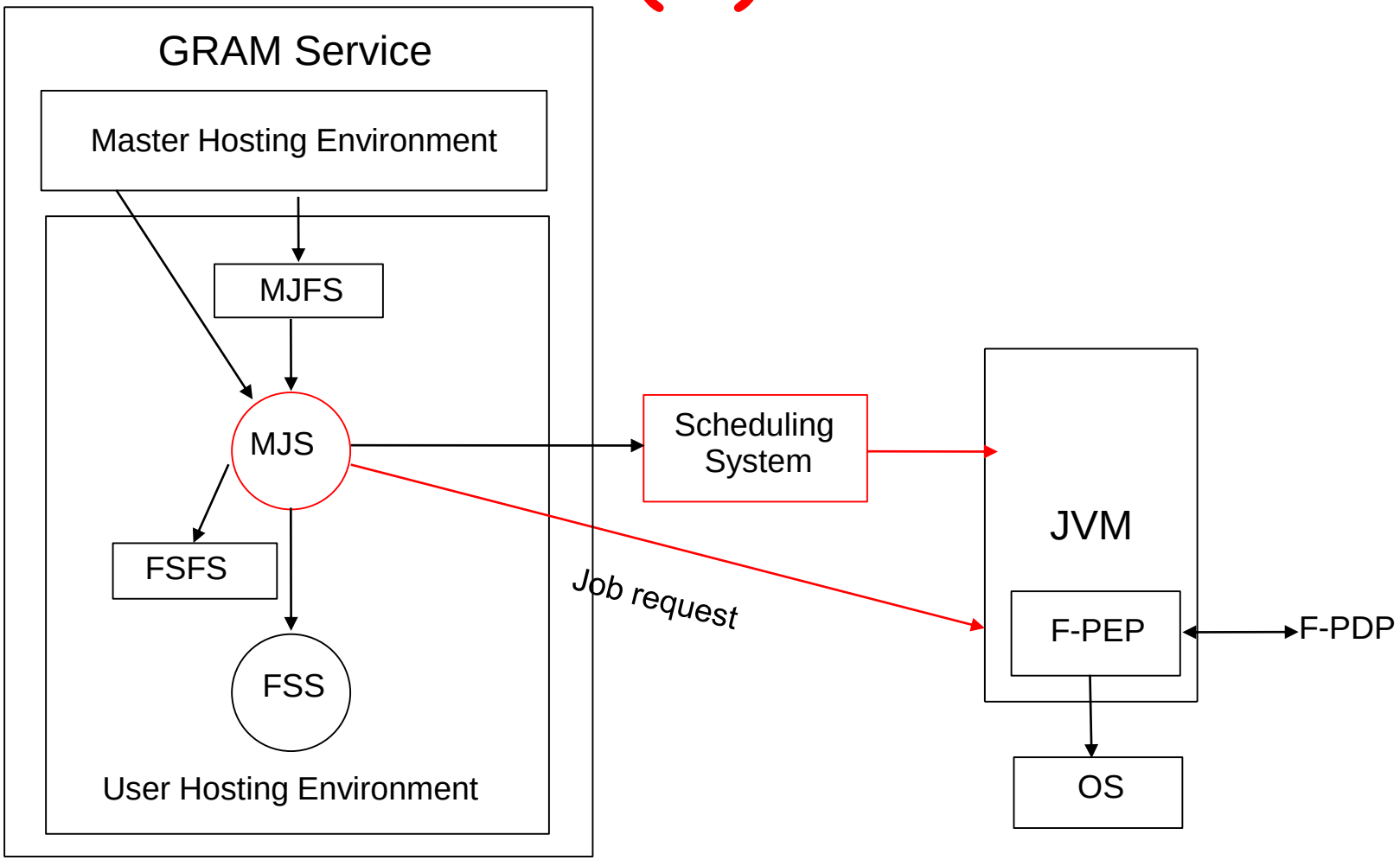
UCON for Computational Services (3)



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

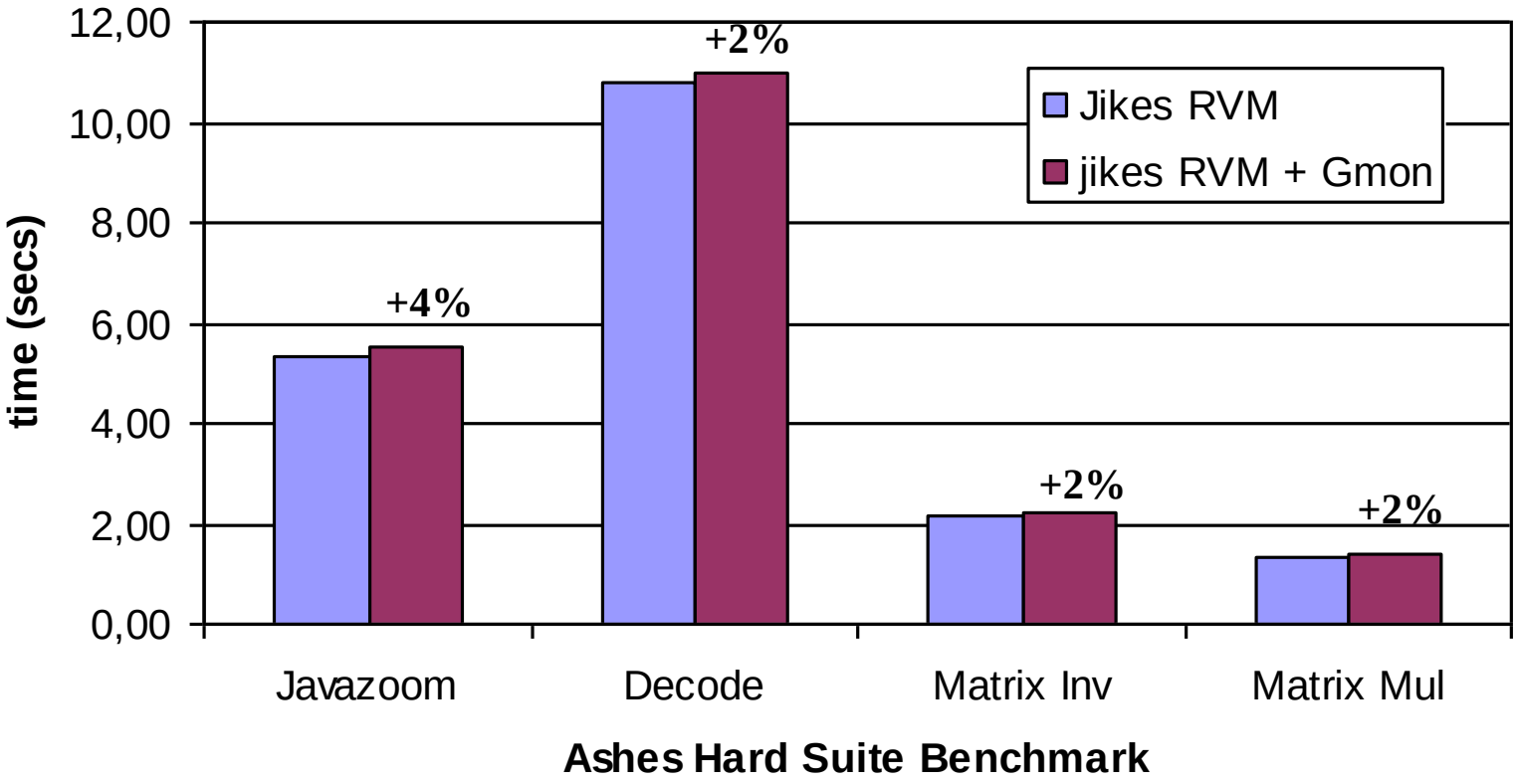
Integration with Globus GRAM (2)



Java Virtual Machine

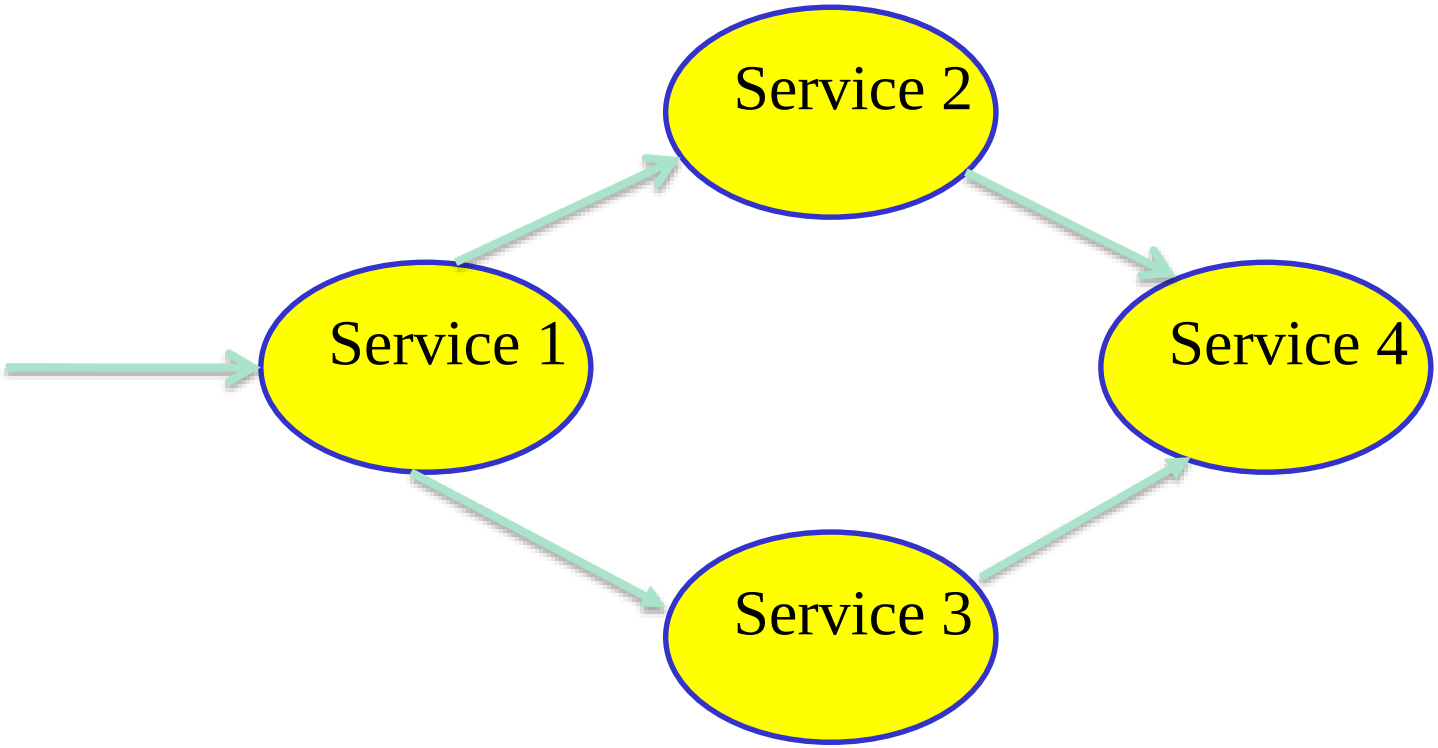
- IBM Jikes Research Virtual Machine
  - Open source Java Virtual Machine
  - Research oriented
  - Follows:
    - The Java Language Specification
    - The Java Virtual Machine Specification

Performances



UCON for GRID services

Example of GRID service workflow authorization

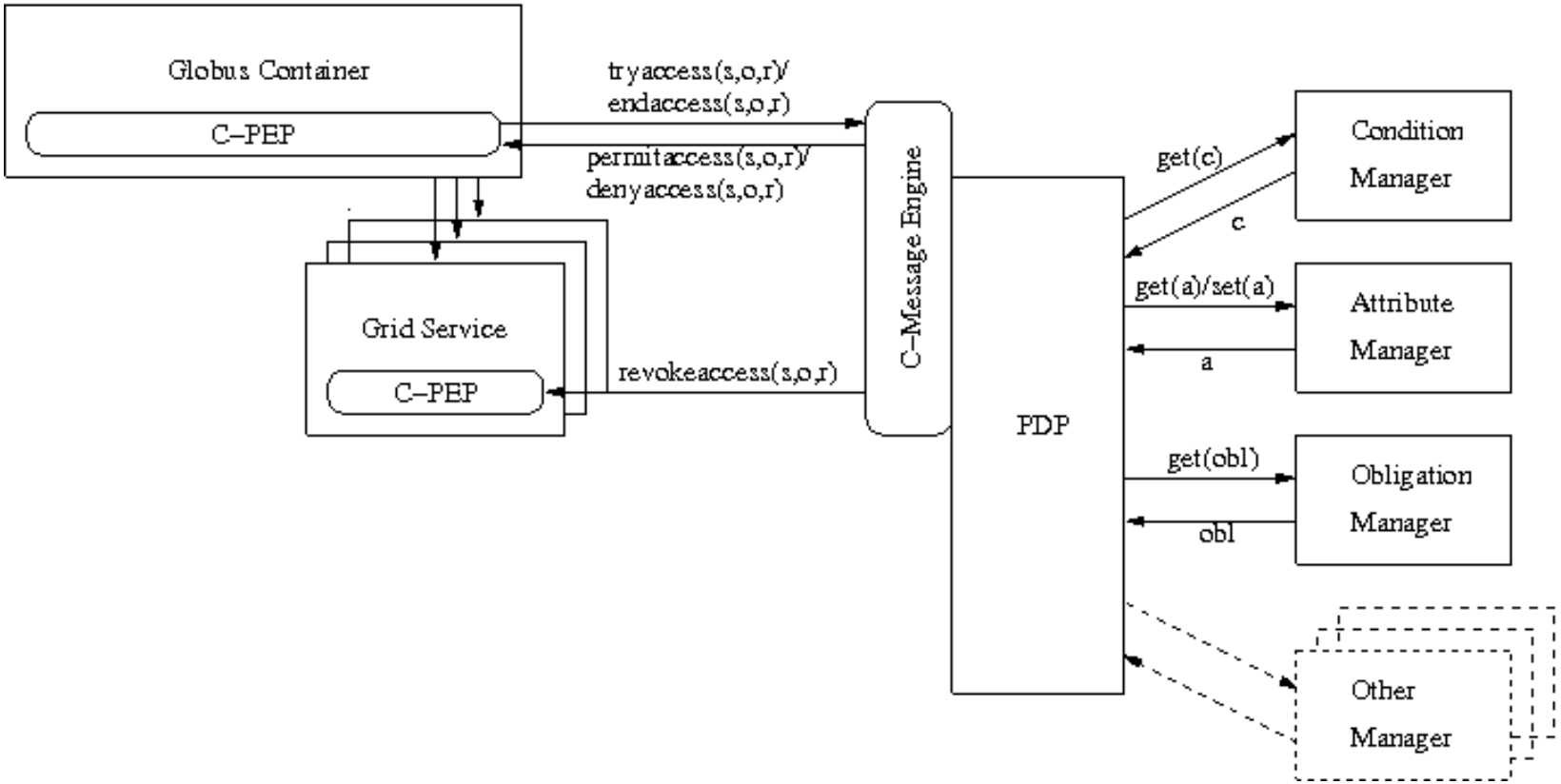


Service1.(Service2 **par** Service 3); Service 4

UCON at Service level

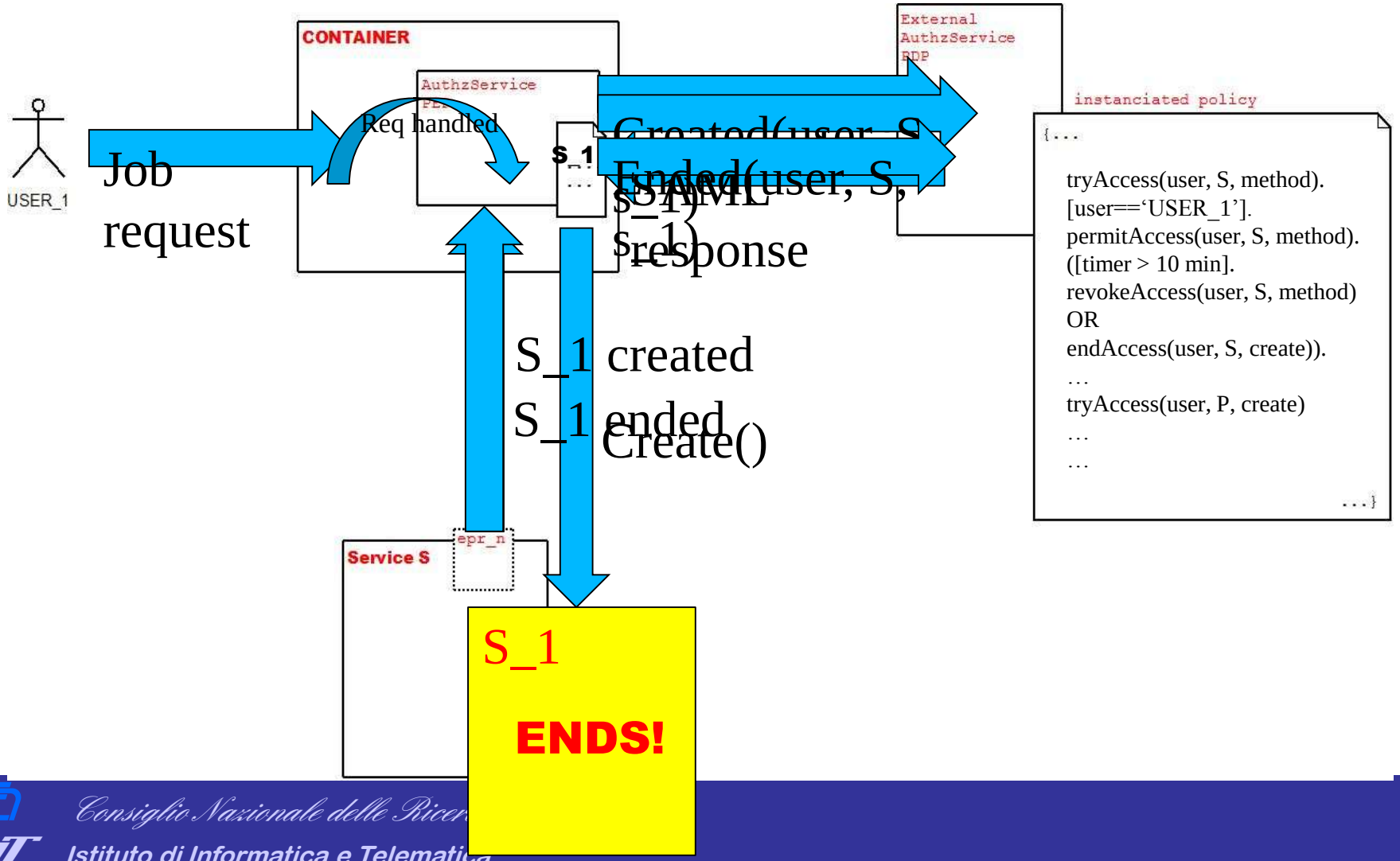
- The UCON PDP is invoked when:
  - A new service instance is created
  - A service instance terminates
- The PDP evaluates the policy
  - Every time a new service instance is created (to check the execution right)
  - Continuously to evaluate some usage conditions
- The PDP interrupts (!) a service 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



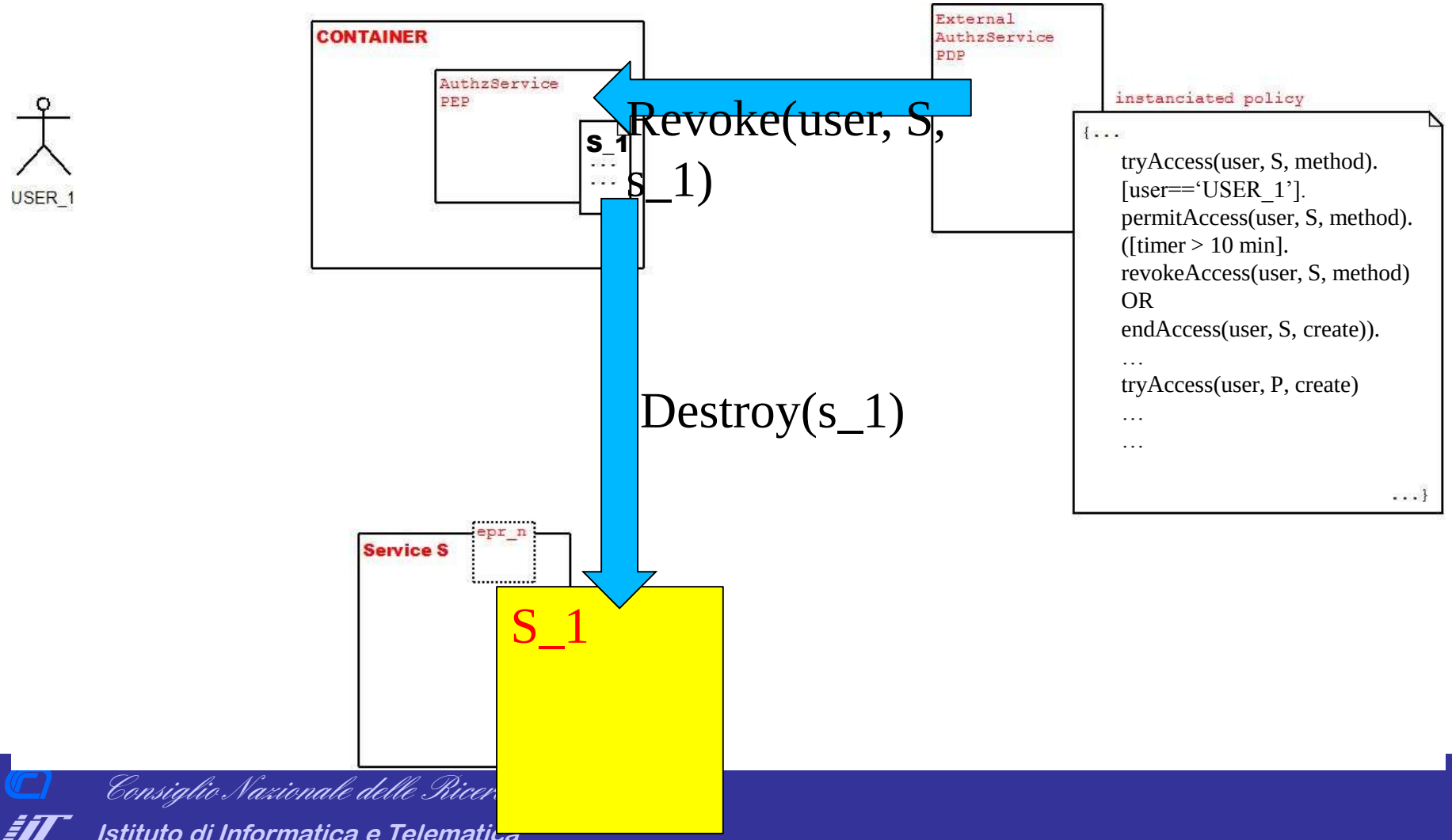
Creation of a new service

Execution Flow (Creation)



Revocation of a running service

Execution Flow (Revocation)



Usage Control in Cloud

Why Usage Control in Cloud?

- Similar arguments as for the GRID
- Accesses to some resources are long-lasting (hours, days,..)
  - e.g., Virtual Machines in IaaS model
- The factors that granted the access when it was requested could change while the access is in progress
  - User's reputation could decrease
  - Workload of resources could change
  - ...
- The policy should be re-evaluated every time factors change
  - An access that is in progress could be interrupted

A new policy language

- XACML is the language recommended for access control in cloud.

- Thus we developed a variant of it for our usage control framework denoted U-XACML.

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
    - There is a lack of generality.
- 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 (cont)

- 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>

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>
```

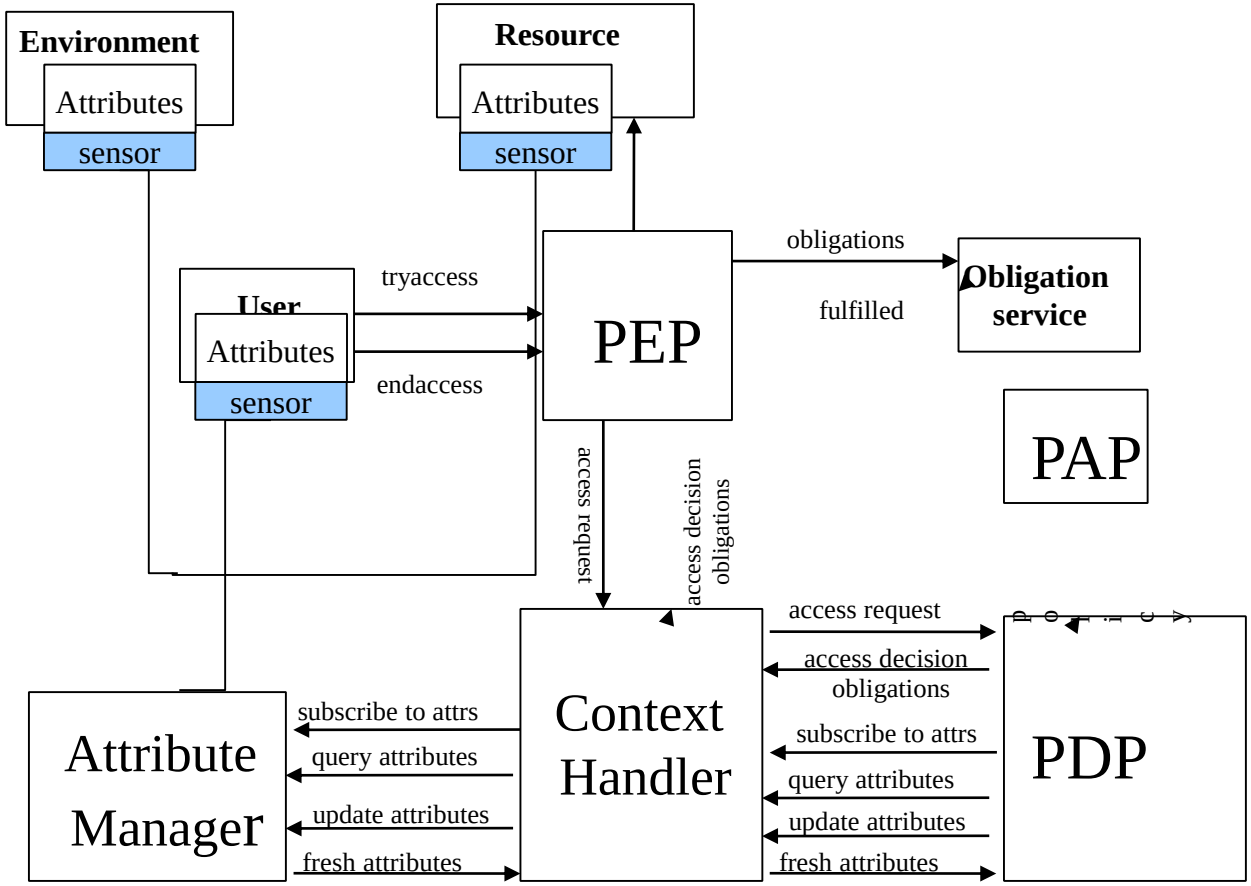
Architecture

- 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

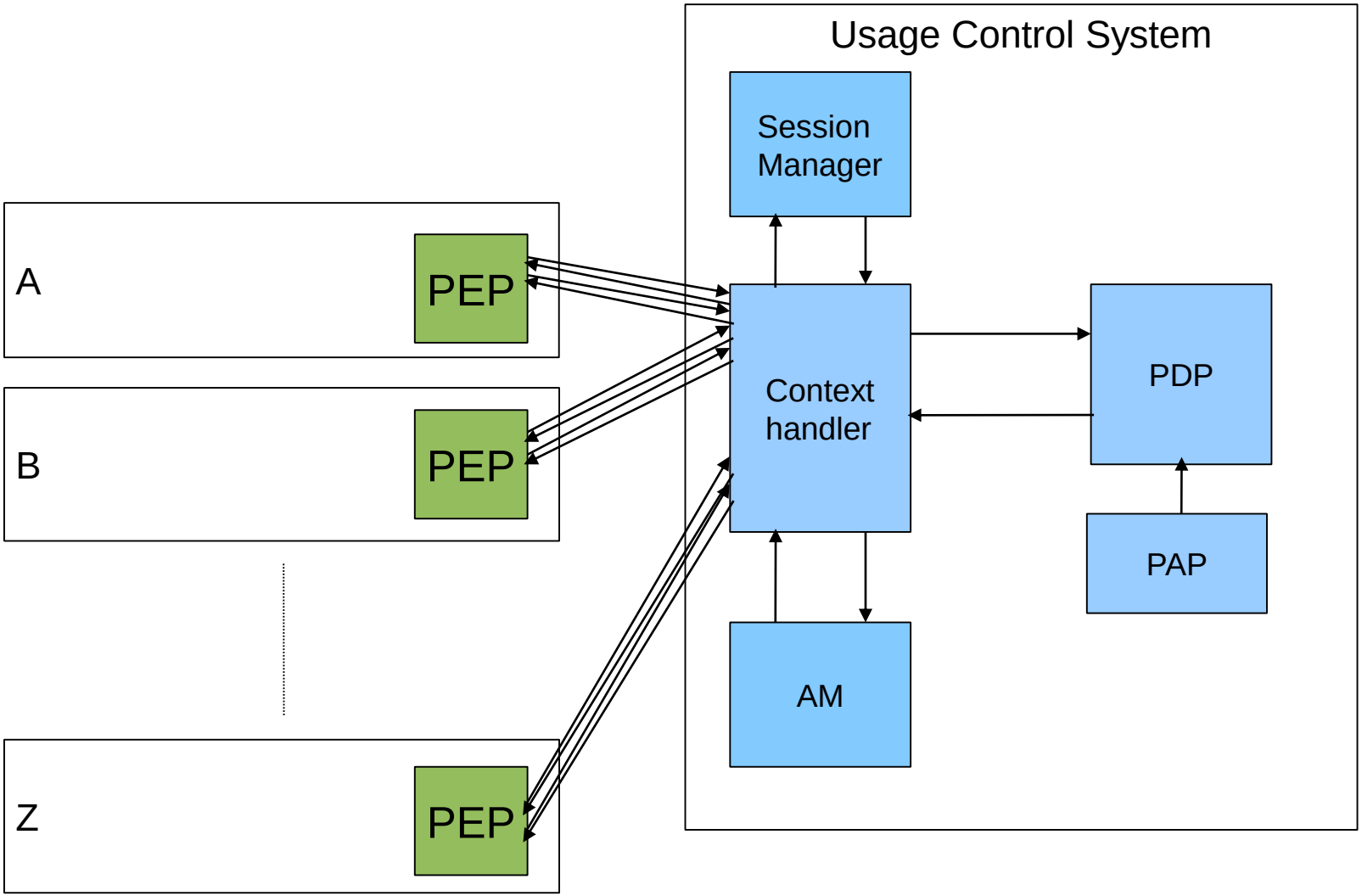
Architecture (cont)



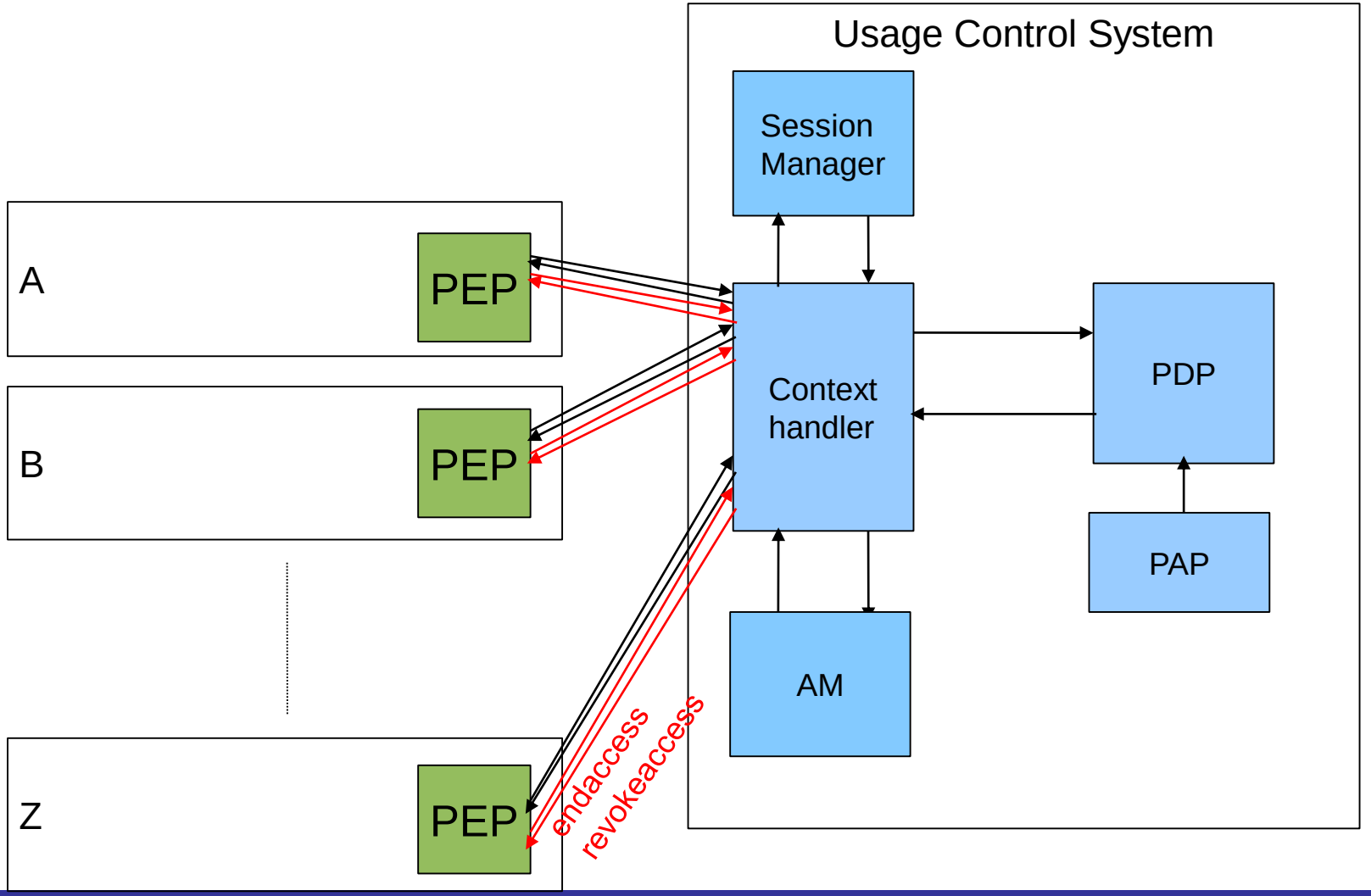
Usage Control System

- Extension of the XACML reference architecture to deal with continuous policy enforcement:
  - PEPs intercept end of accesses (besides access requests)
  - Session Manager (new component) keeps trace of accesses in progress
  - AM monitors mutable attributes
  - PDP revokes ongoing accesses
    - is triggered also for re-evaluation for policies when attribute change

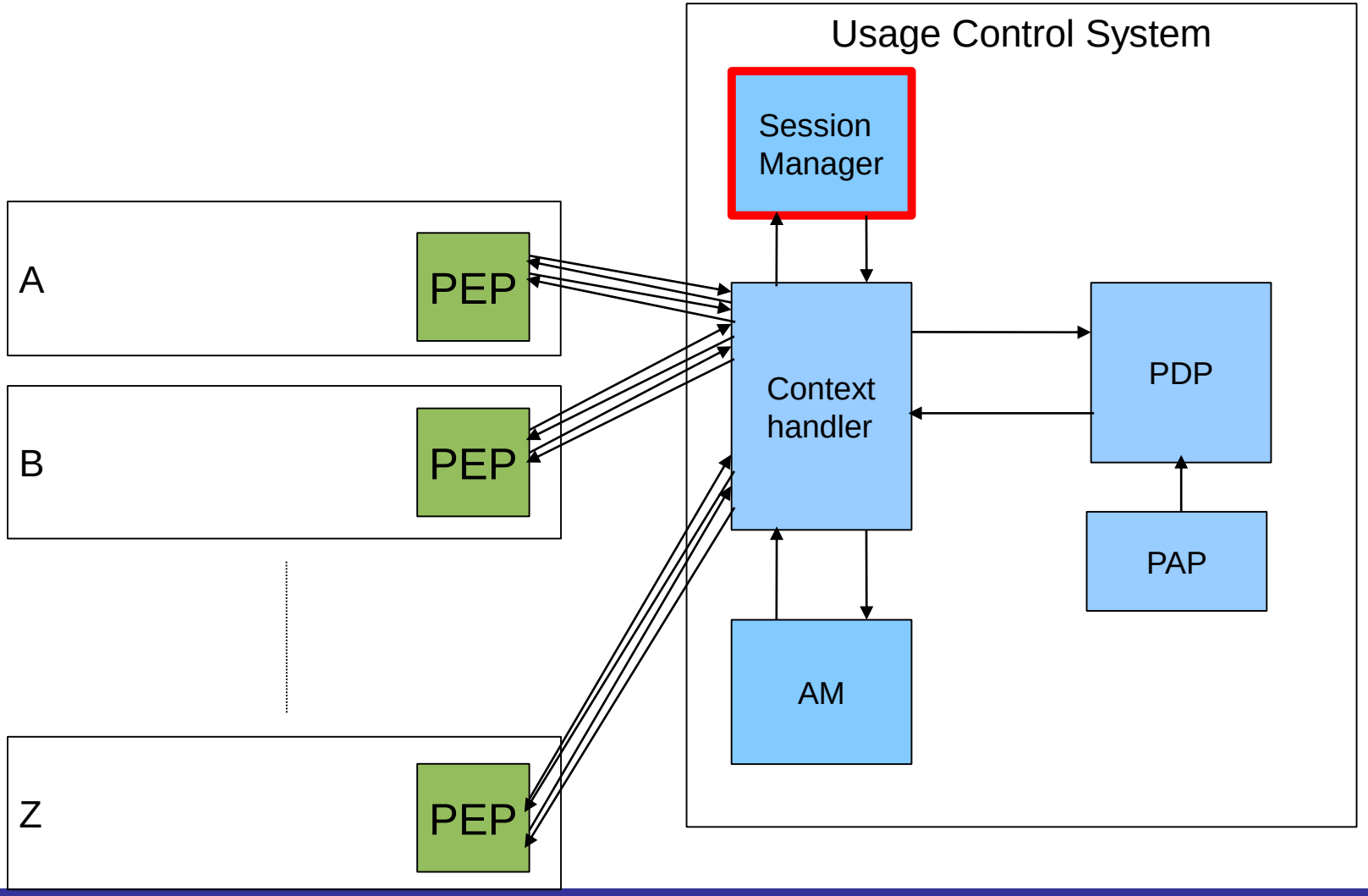
Usage Control System Architecture



Usage Control System Architecture



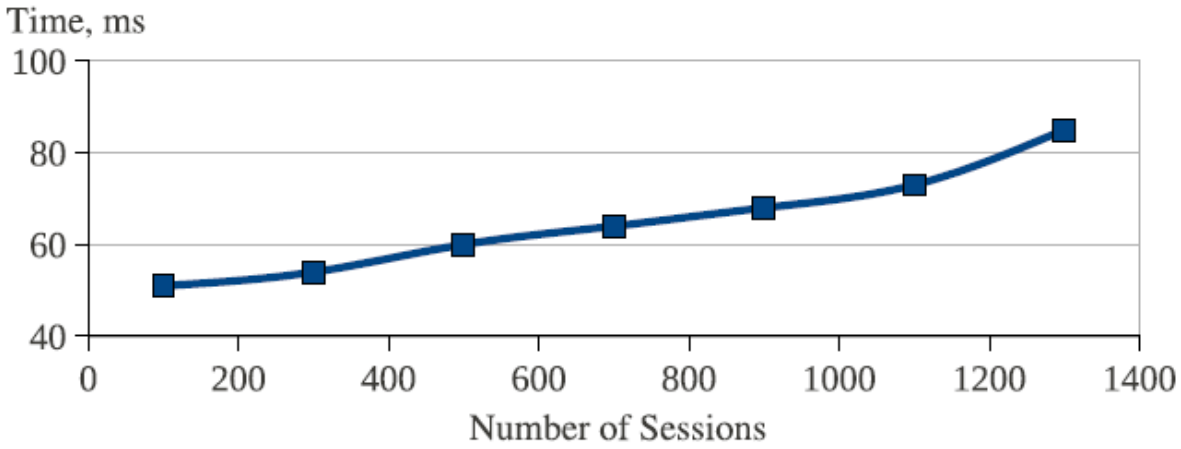
Usage Control System Architecture



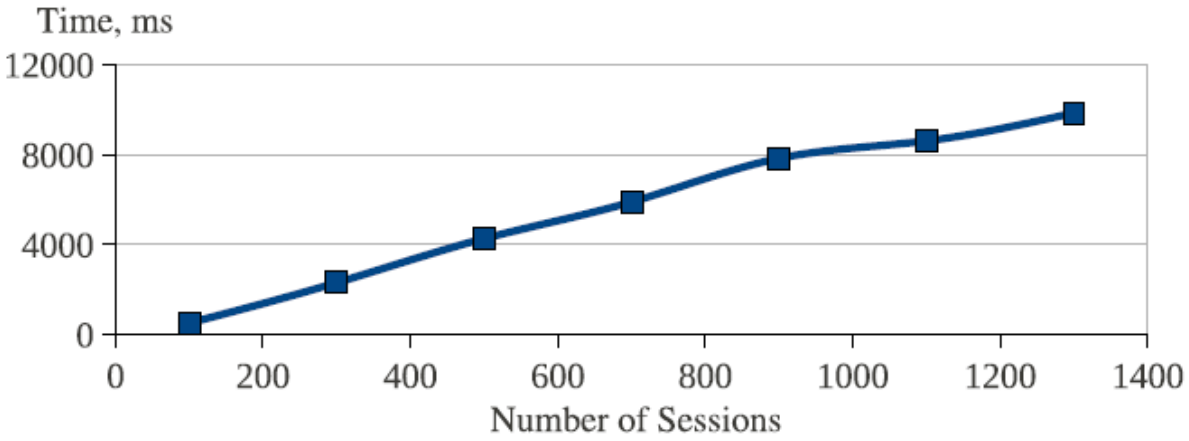
Possible management of the usage sessions

- We developed a couple of implementations
  - Using ad-hoc Table to record the active sessions
    - Efficient method
  - Using the XACML call-me-back obligation feature
    - XACML compliant
    - slower

Performance



1 revocation

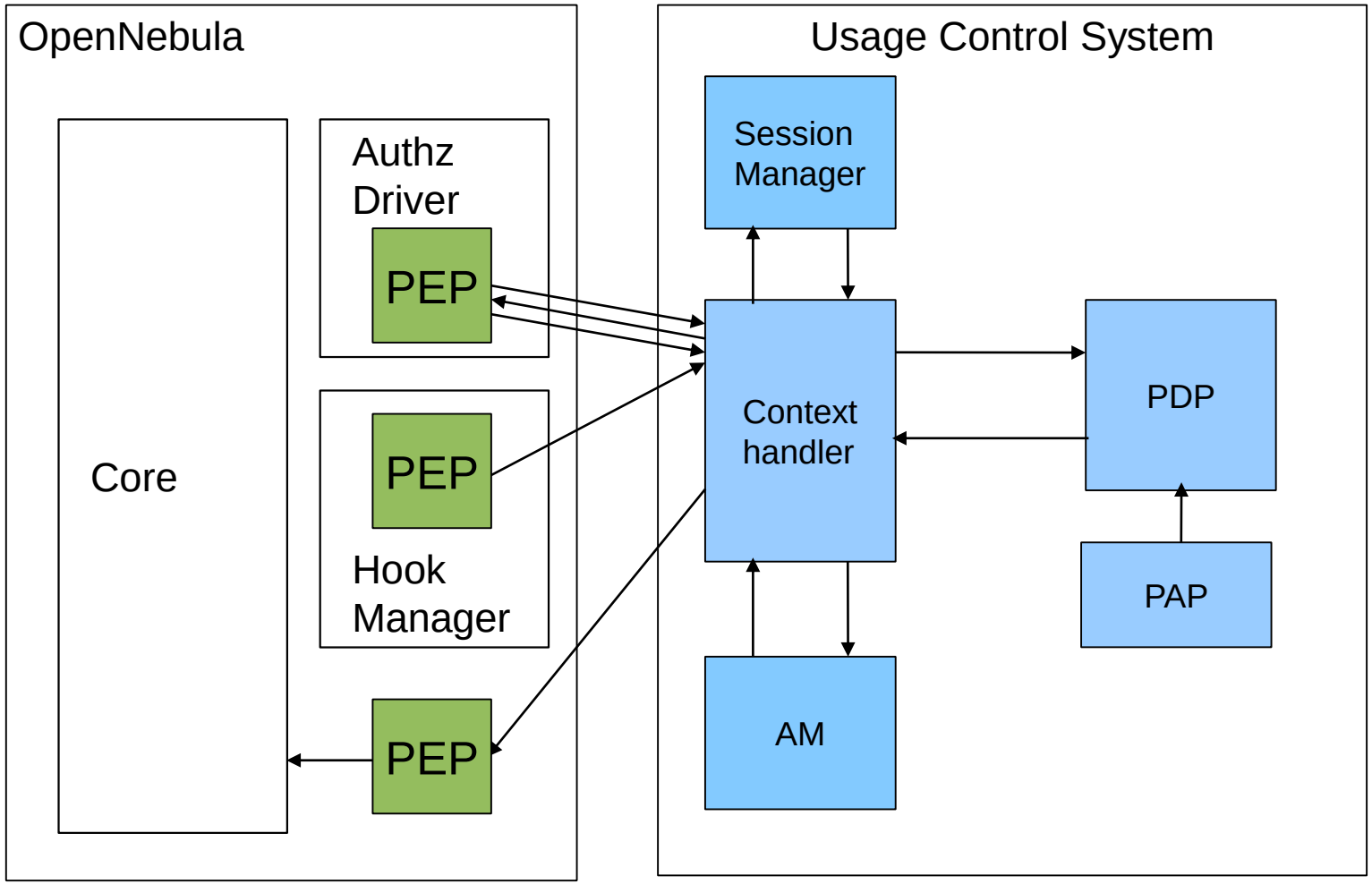


Massive revocation

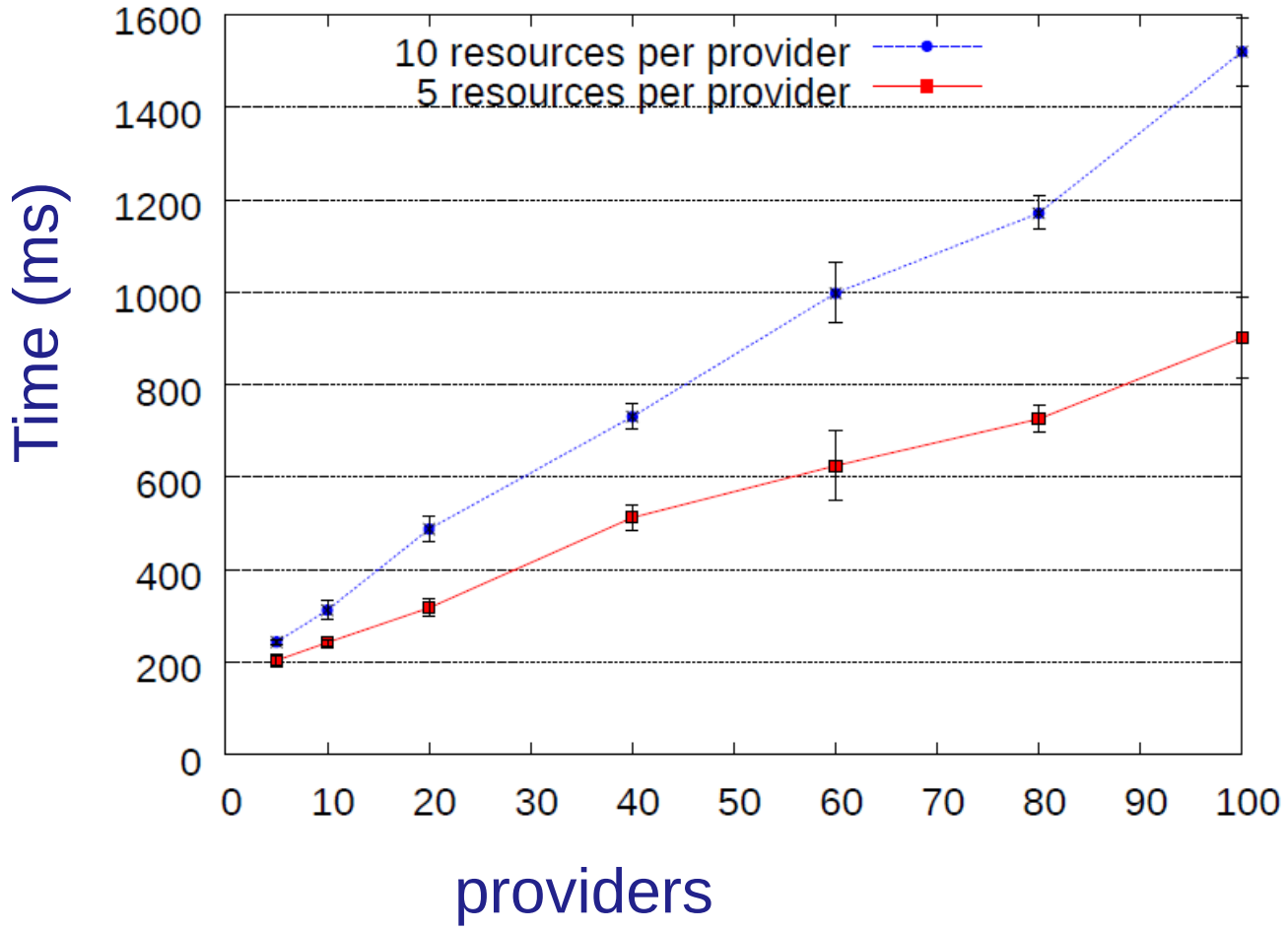
Our Prototype: Usage Control in IaaS Cloud

- IaaS framework: OpenNebula
- Resources: VMs
- Usage of VMs: create – shutdown - suspend - resume – stop - cancel
- Integration of Policy Enforcement Point in OpenNebula
  - Access Request: Authorization Driver
  - Access Start: Hook Manager
  - Access End: Authorization Driver
  - Access Revocation: ONE core

Integration with OpenNebula



Performance



Access revocations

Usage Control Advantages

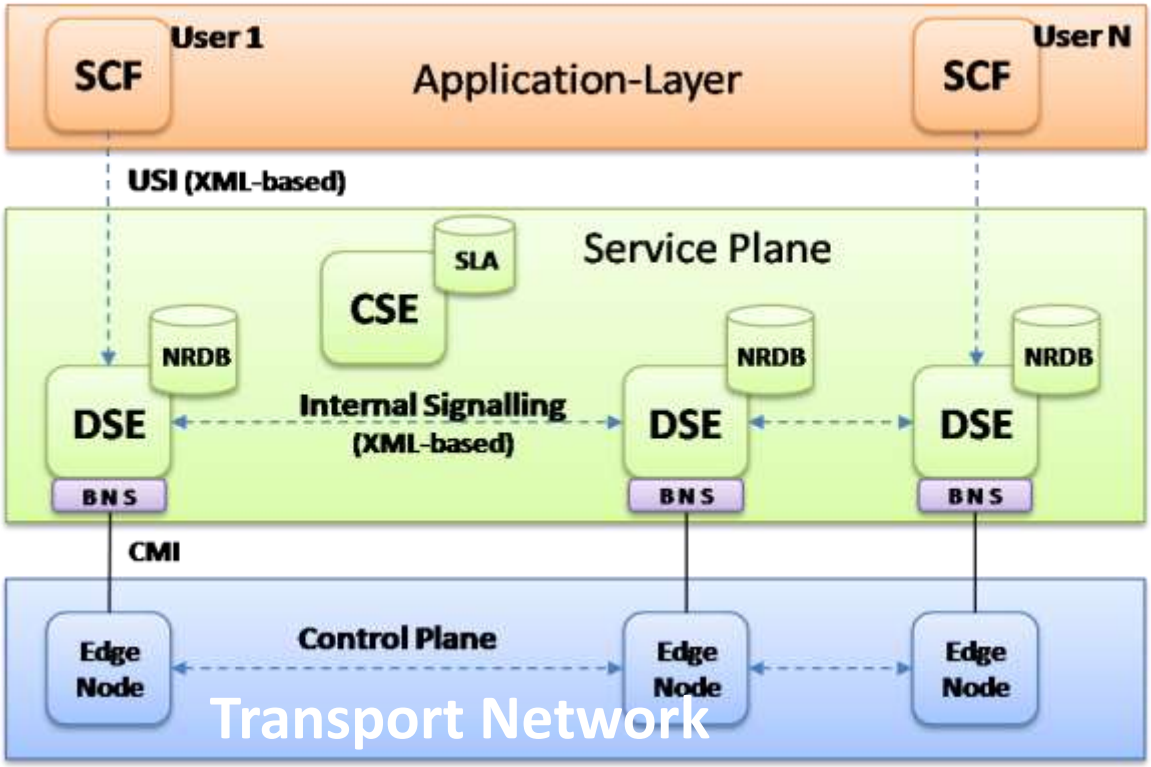
- Improves the IaaS provider security by enforcing security policies
  - At access request time (standard authorization)
  - Continuously, while the access is in progress (usage control)
  - If the policy is violated, the ongoing access is interrupted
- Main advantage: avoiding the prosecution of accesses when the corresponding rights are not valid any more
  - Interrupts potentially dangerous accesses
  - Saves resources
- Recommended when
  - Accesses are long lasting (e.g., VM execution)
  - Mutable factors are evaluated in the security policy (e.g., user reputation or system workload)

UCON for network services

Service Oriented Networks

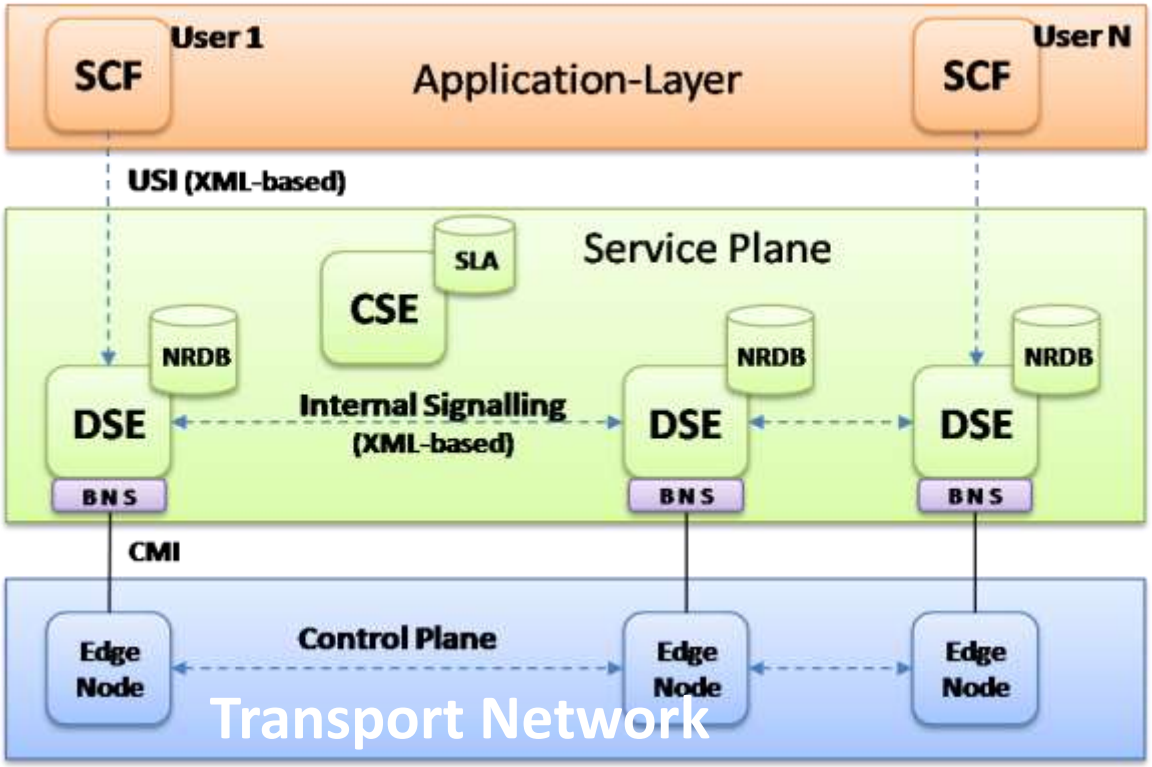
- On-demand framework for network services provisioning, e.g.
  - VOIP
  - MultiMedia on Demand
  - ….
- Level of abstraction suitable for being invoked directly by applications

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

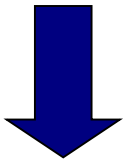
Need for an advanced security support to control the usage of network services!

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 (smartphones, 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

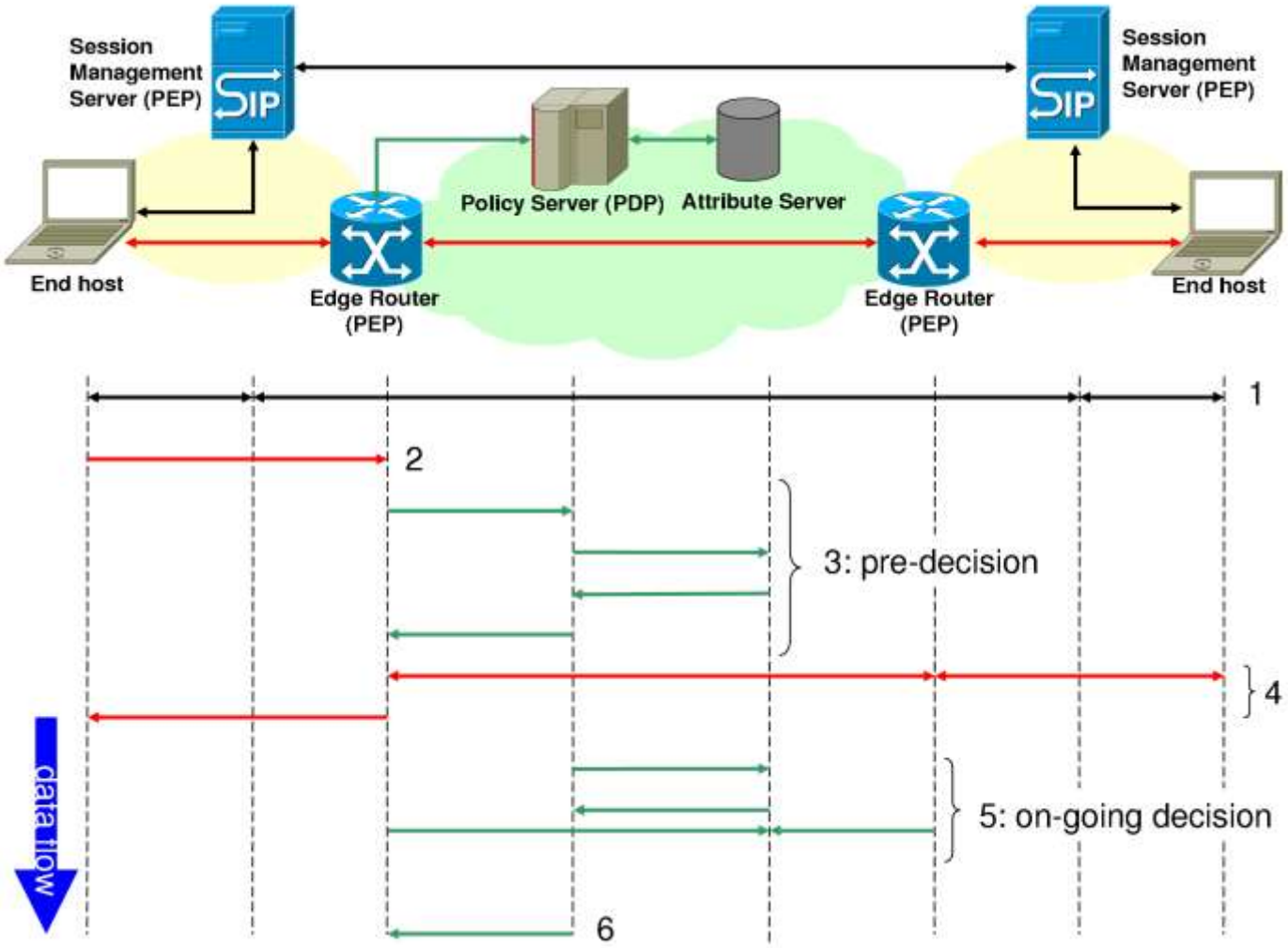
Example of Security Policy (natural lang)

- 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

Usage control for Android

- NOTE: Work in progress

UXACML on Android

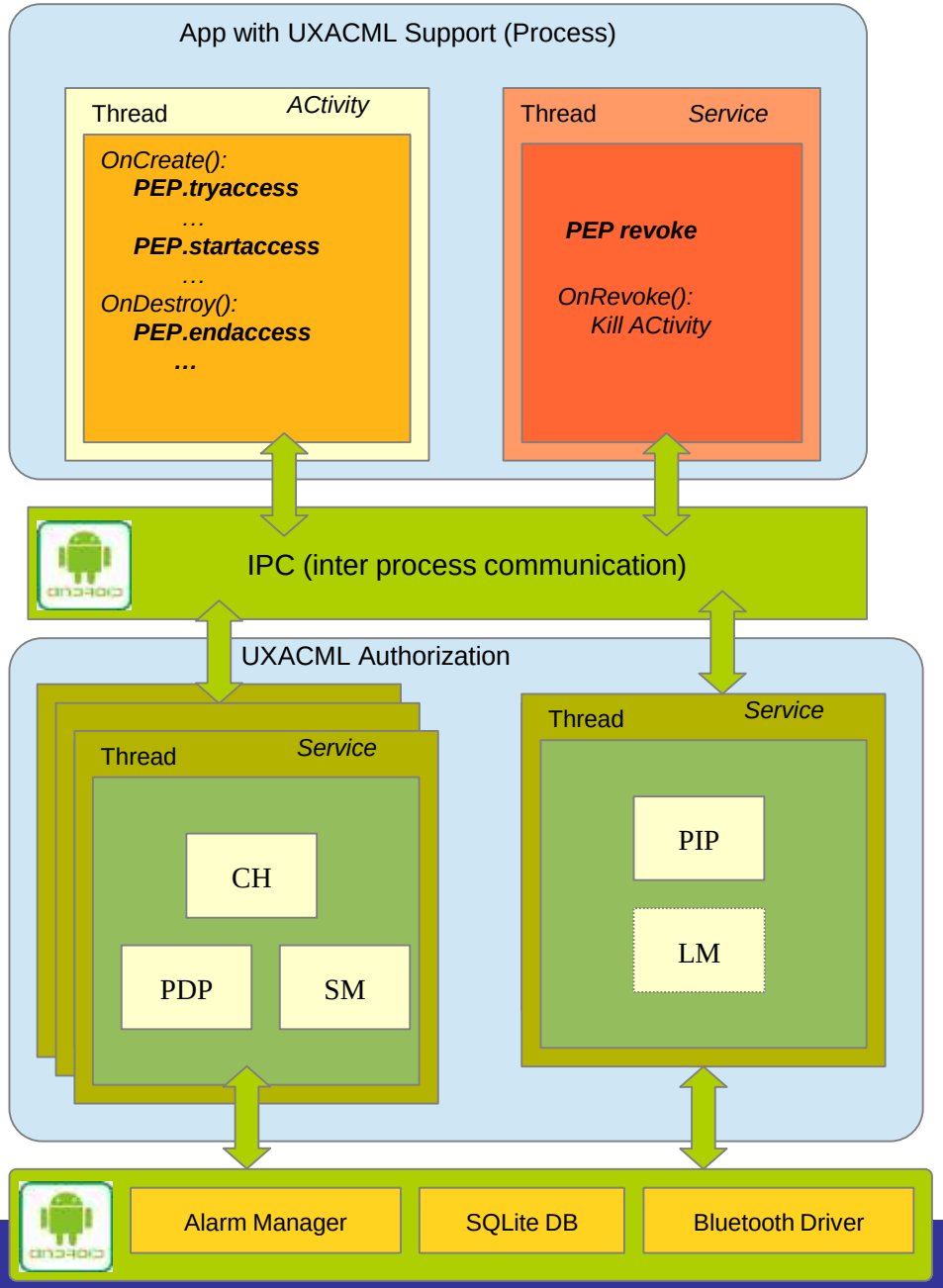
- The UXACML authorization system for Android is based on UCON
- It allows Android applications to protect their resources/data from other Android applications and users of the device
- Examples of the resources
  - Android Application Components:
    - Activity
    - Services
    - Content Providers
  - Internal data of applications:
    - Files
    - DataBase

Policy Examples

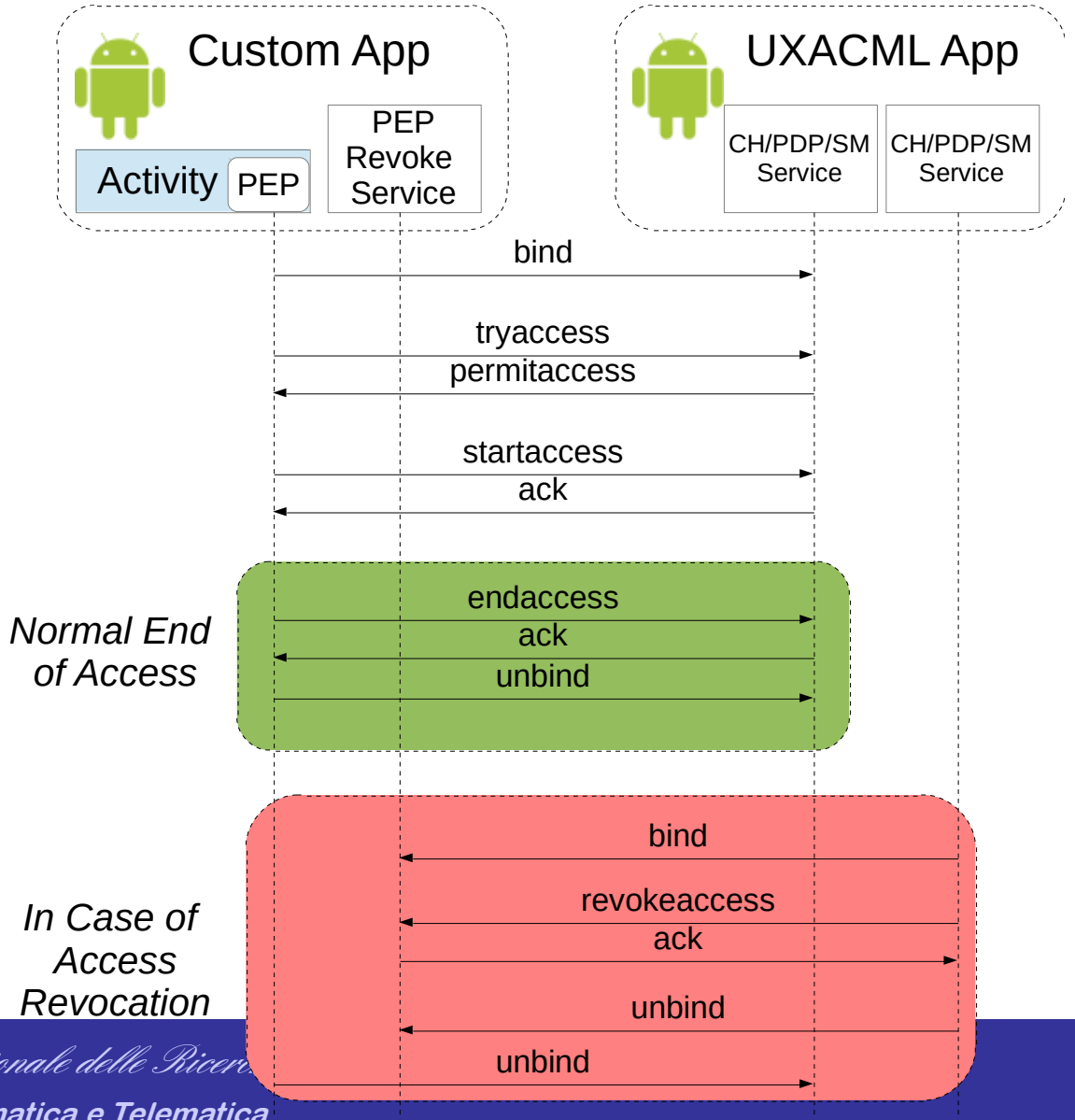
- If network connection (wireless, Bluetooth, NFC, etc) is turned on when the application activity is running, then the authorization system should revokes the access and destroys the activity
- Application allows users to download only 5 data objects (e.g., media files). Each new download is allowed but it triggers the deletion of the oldest data object

Architecture

- PEP (Policy Enforcement Point)
- CH (Context Handler)
- PDP (Policy Decision Point)
- SM (Session Manager)
- PIP (Policy Information Point)
- LM (Lock Manager)



Message Frow



Implementation Notes

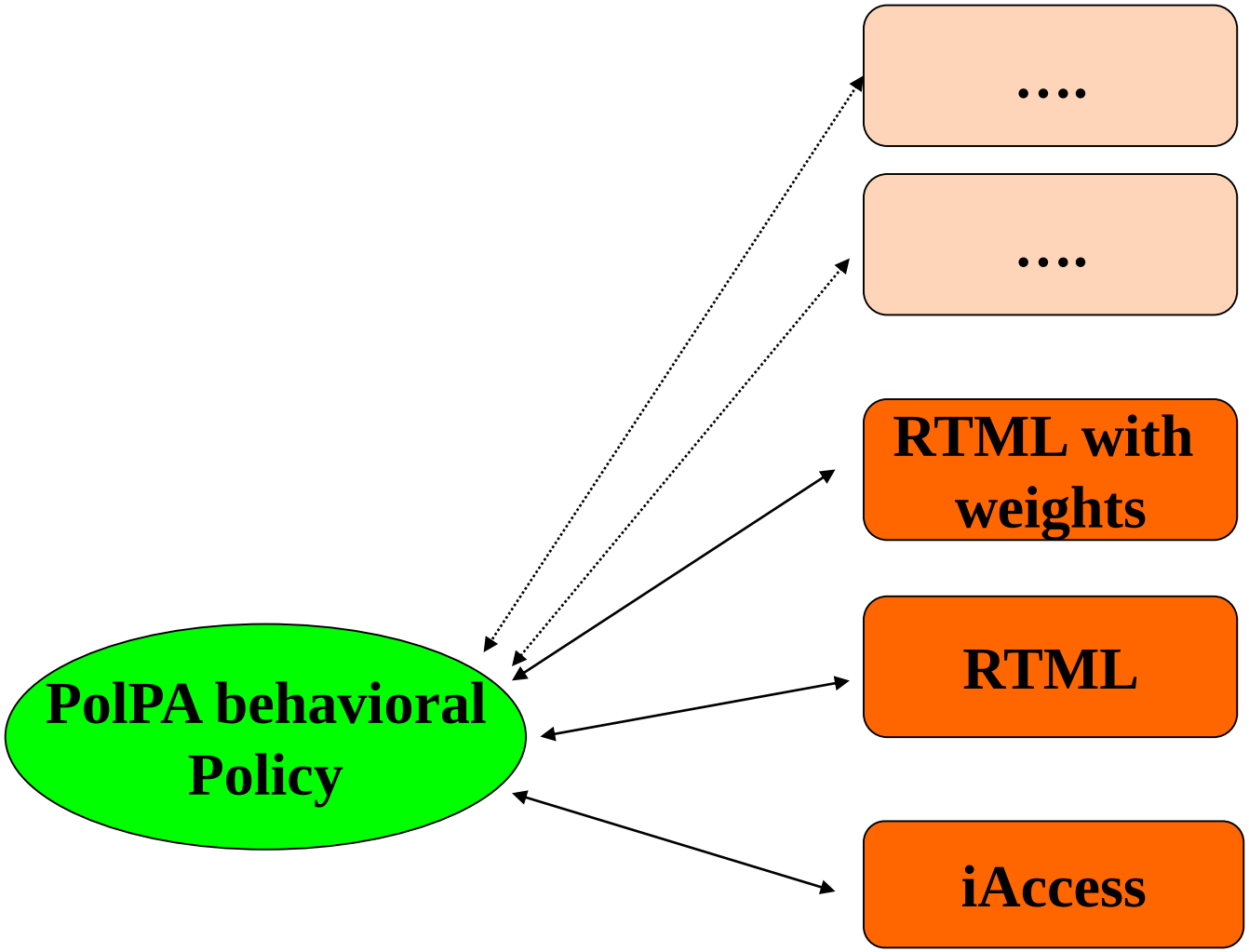
- UXACML App evaluates each request is a separate thread to enhance performance
- Communication between App to control and UXACML App are synchronous
- App PEP binds to UXACML App for try/start/endaccess invocations while UXACML App binds to App PEP service for access revocation
- Alarm Manager triggers the policy reevaluation
- All session-related metadata and security attributes are stored in SQLite DB which is private data of UXACML App

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)



Trust

Outline

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

(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/credential 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)

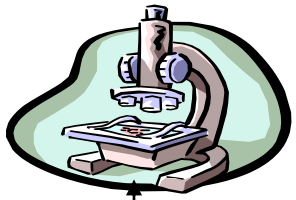
- A family of languages for reasoning about trust relationships in distributed environments
  - A, B, C, D, … entities
  - r roles/attributes
- Rules:
  - A.r <- D means D has role r for A
  - A.r <- $B.r_1$ means if C has role r_1 for B then C has role r for A
  - A.r <- $A.r_1.r_2$,means if B has role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

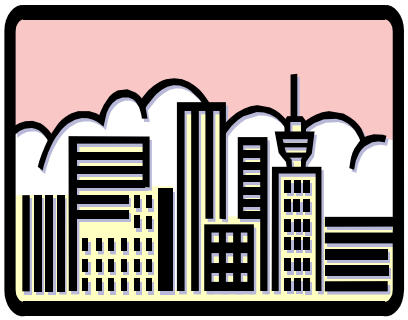


DI

2. UniPI.stuID ← DI.stuID

UniPI

3 CRUI.accredited ← UniPI



CRUI

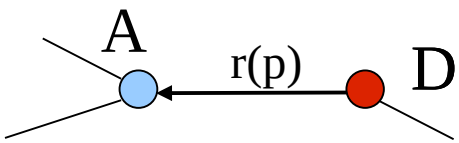
4. Karadar.university ← CRUI.accredited

5. Karadar.student ← Karadar.university.stuID

RTML (another view)

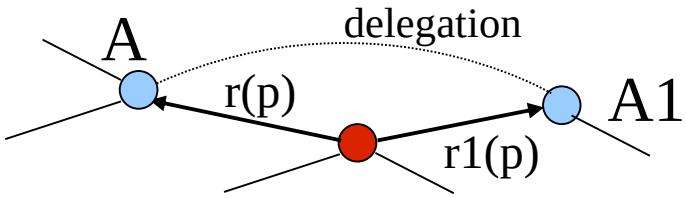
- There are four types of credentials (basic) in RT:
  - Simple Member:

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



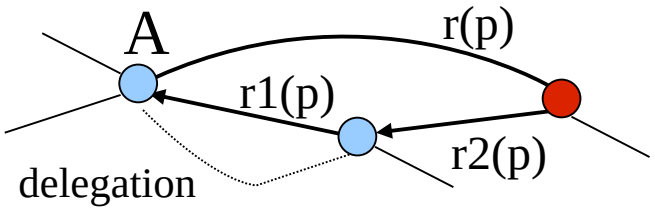
  - Simple Inclusion:

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



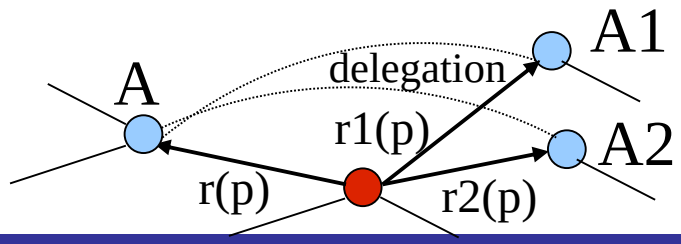
  - Linking Inclusion:

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



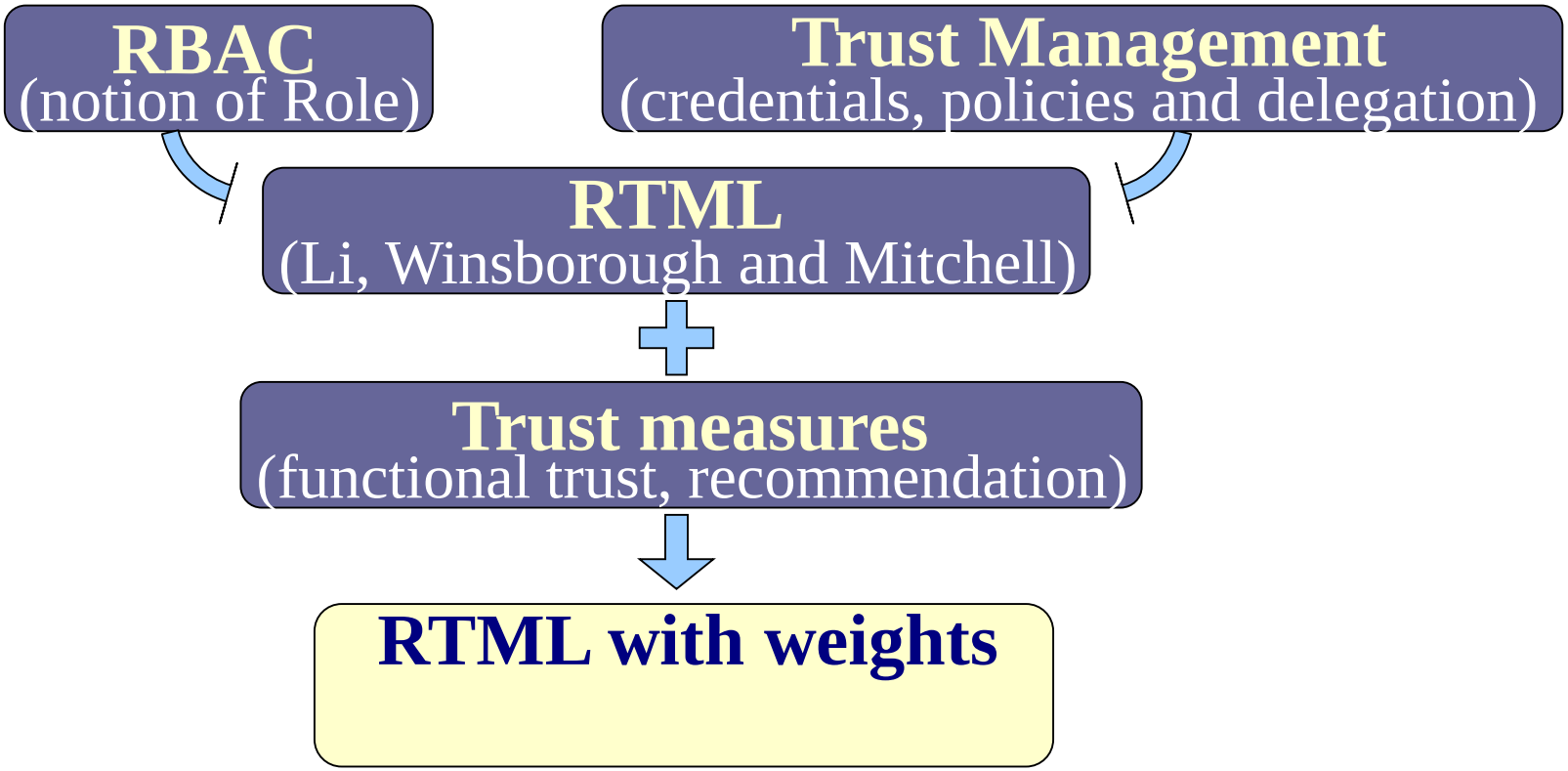
  - Intersection Inclusion:

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



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 among 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 minimal number of steps
    - $\otimes$ -> sum on natural numbers
    - $\odot$ -> min
  - 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

Another more complex c-semiring

- The operators are defined as follows:

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

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

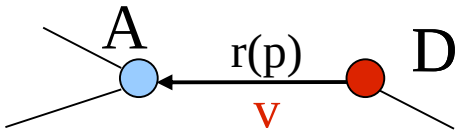
RTML with weights

Credentials enriched with trust measures (1):

- Simple Member:

  A.**r**(*p*,*v*)←D

  – A.r is covered with weight v

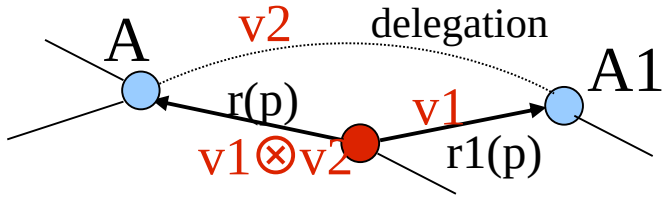


- Simple Inclusion:

  A.**r**(*p*) ←v2 A1.**r1**(*p1*)

  – All members in A1.r1 with

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



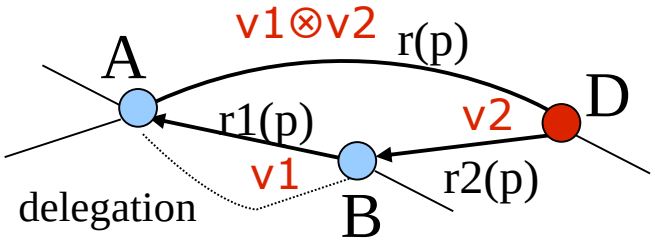
RTML with weights

Credentials enriched with trust measures (2):

- Linking Inclusion:

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

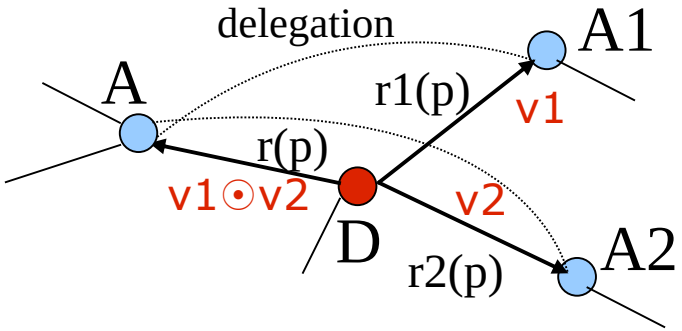
  - If B has role A.r1 with v1 and D has role B.r2 with v2 then D has role A.r with v = v1⊗v2



- 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



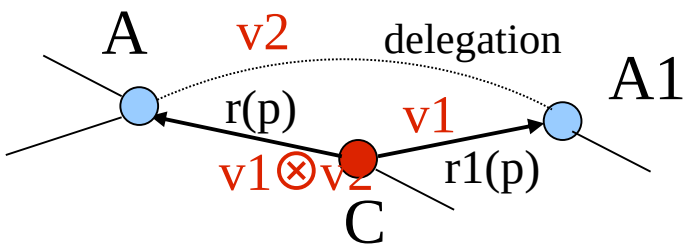
RTML with weights

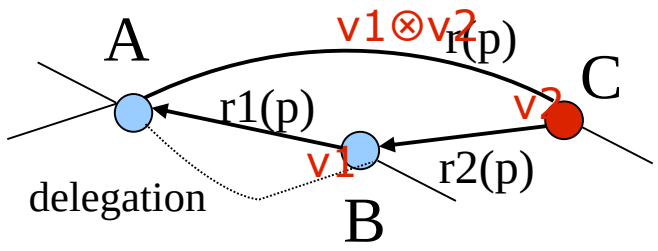
- For authorization answer a query:
  - Given an entity D, its credential and access rules, determine all the roles it is a member of with its max weight.

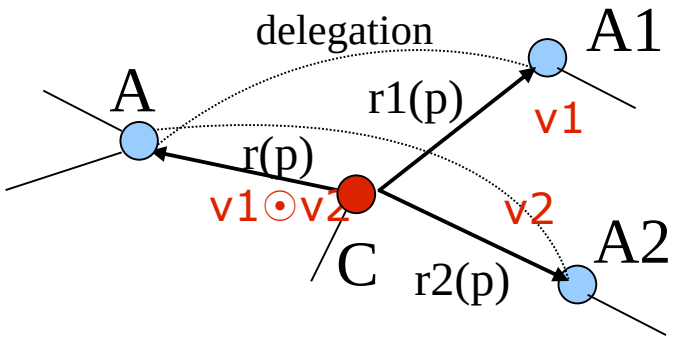
RTML with weights

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(v)** ← 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 recommendation 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

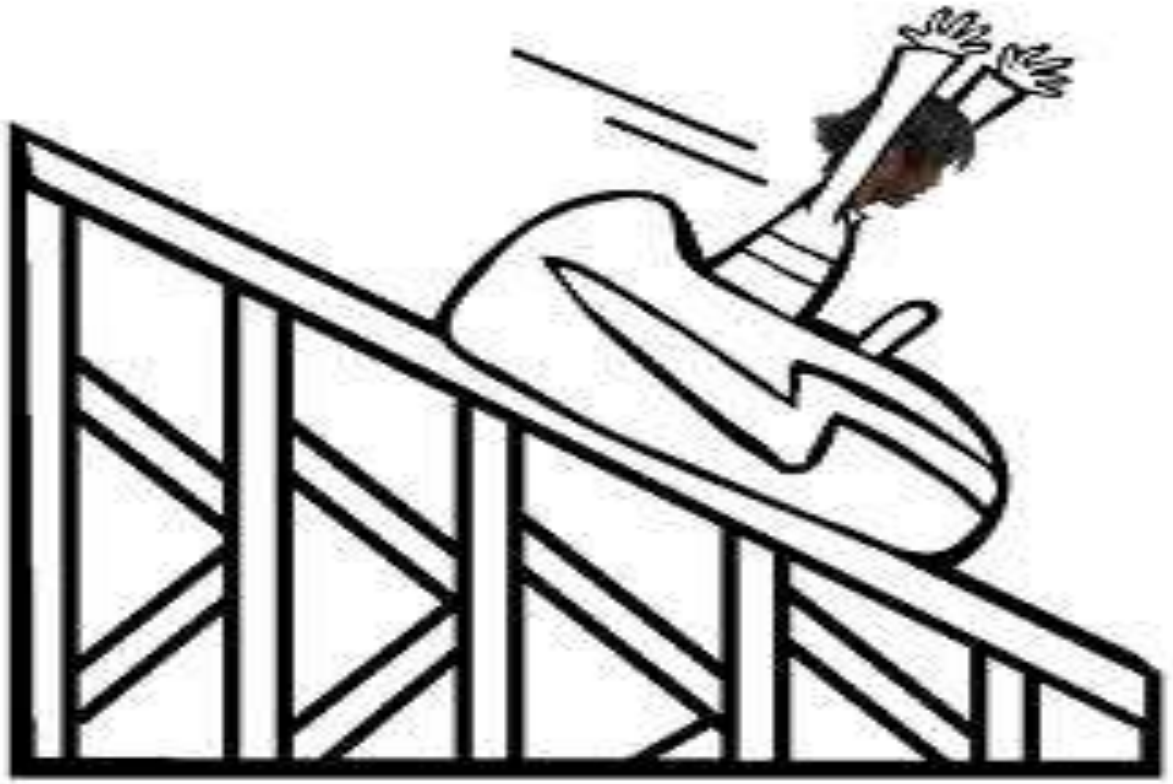
$$\frac{A \xrightarrow{rf} B \quad B \xrightarrow{f} D}{A \xrightarrow{f} D} \quad (4^*) \text{ INDIRECT FUNCTIONAL TRUST}$$

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*)	$\dfrac{A \xrightarrow{rf} B \quad B \xrightarrow{f} C}{A \xrightarrow{f} C}$	$A.f \leftarrow A.rf.f$

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.

Risk-aware Usage Decision Making

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.

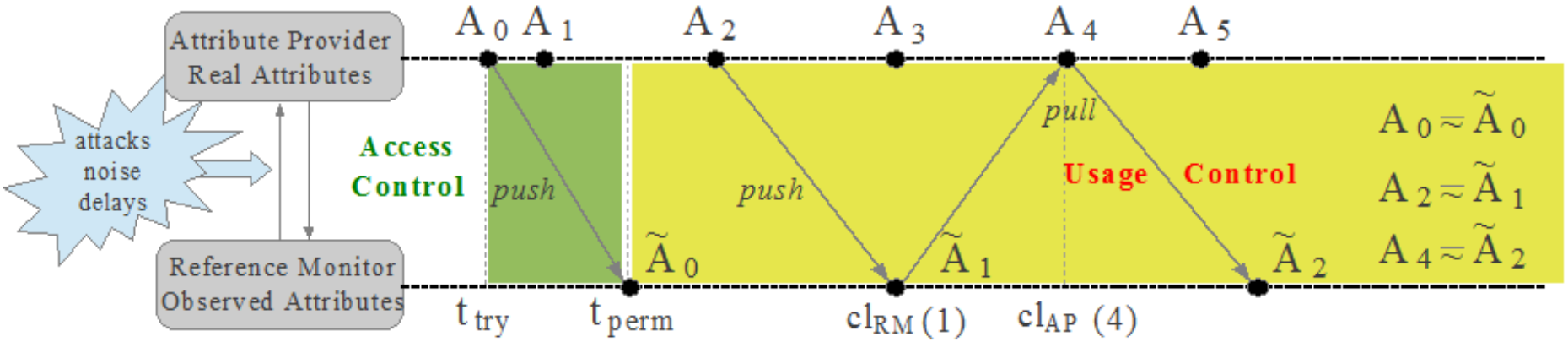
•Result: **wrong access decision** caused by not fresh attributes.

Enforcement of UCON Policy

- Reference monitor can only observe <u>some</u> attribute changes

Access decision is based on observed (<u>not real</u>) attribute values

- How to make access decision under uncertainty?
- When to retrieve fresh attribute values?



Access Decision in UCON

Attribute values are uncertain, so…

1) Evaluate the policy using observed attribute values (traditional approach)

2) If the observed values satisfy the policy, estimate the probability $\mathbf{Pr}$ that real values satisfy the policy too

3) If $\mathbf{Pr}$ is acceptable, grant the access

- Assumptions
  - We can assign costs of possible decisions outcomes
  - We know statistics of attribute changes, thus can compute $\mathbf{Pr}$

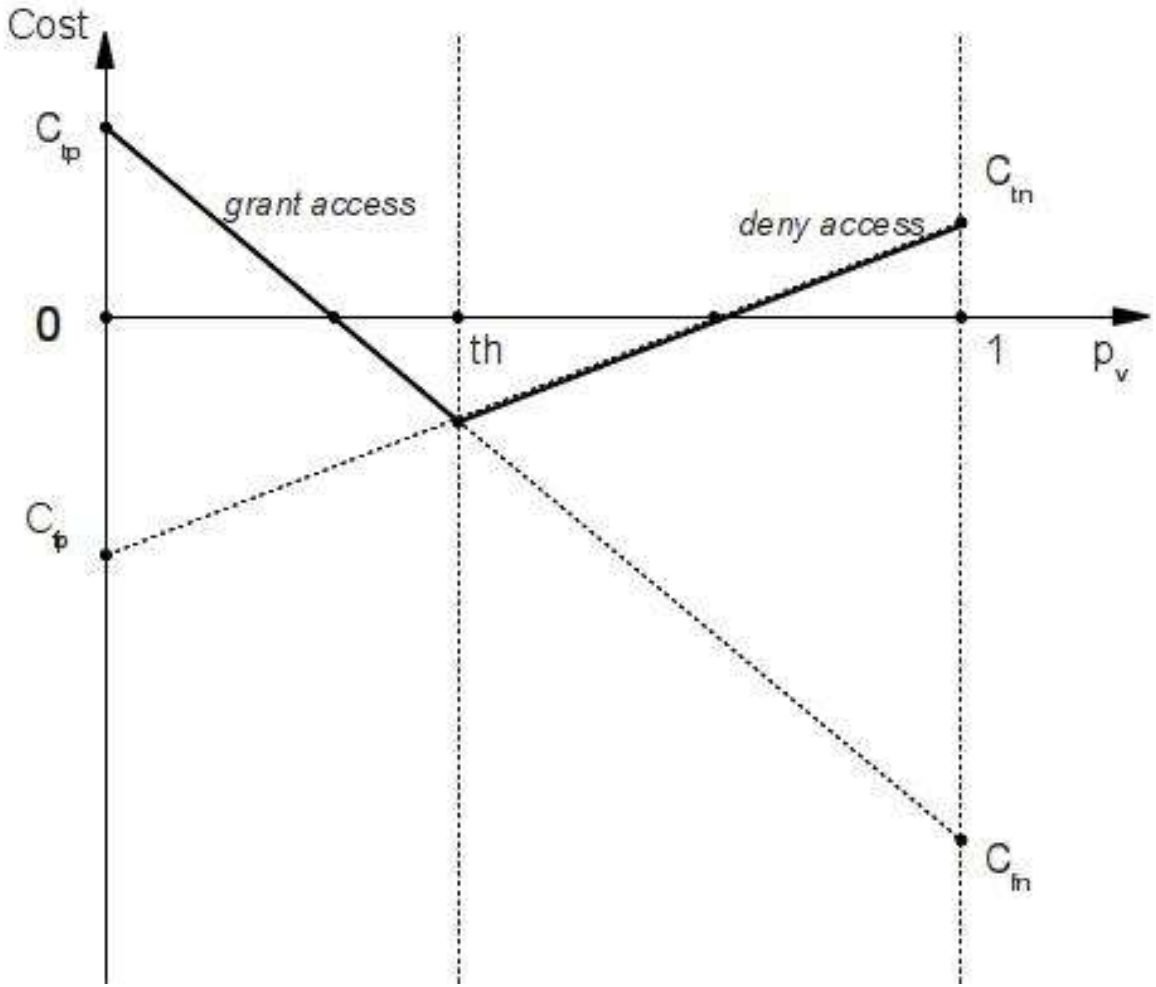
Cost-effective Access Decision

- Costs for granting and revoking access to legitimate and malicious users:
  - grant access when policy holds: C_{tp}
  - grant access when policy is violated: C_{fn}
  - deny access when policy holds: C_{fp}
  - deny access when policy is violated: C_{tn}

- Weight permit-deny options

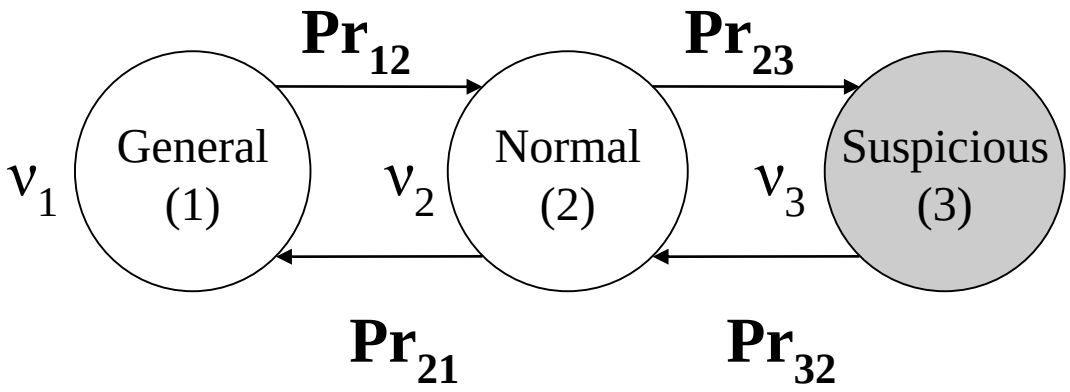
$$\mathbf{Pr} \geq \frac{C_{fn} - C_{tn}}{C_{fp} + C_{fn} - C_{tn} - C_{tp}}$$

PERMIT  DENY


Cost
C_{tp}
grant access
C_{tn}
deny access
0
th
1
p_v
C_{fp}
C_{fn}

Policy Example

- POLICY: User is permitted to execute a VM in Cloud if his/her reputation remains above *the threshold*
- ATTRIBUTE: reputation (Markov chain)
  - ATTR DOMAIN: general, normal, suspicious



Example. Reputation

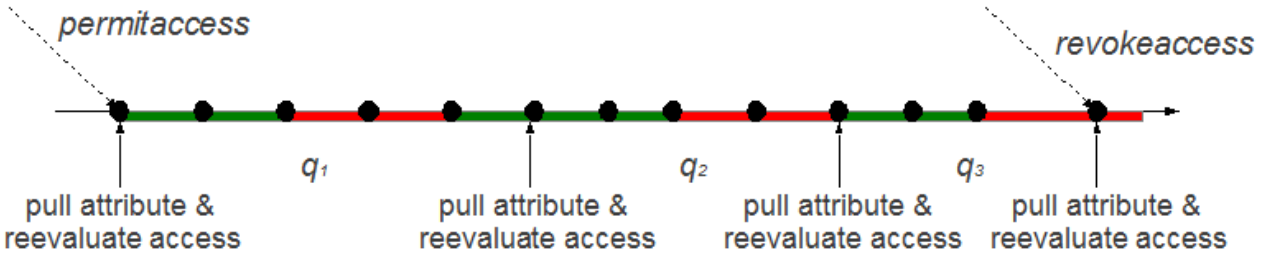
- Take the initial state (general)
- Take time passes since the last attribute query
- Compute probability to appear in State 3 (suspicious)
- Make decision

Attribute Retrieval in UCON

- Frequent attribute queries can be impossible or ineffective
- Pull fresh attribute values periodically or aperiodically
- How to choose the best interval between adjacent queries?

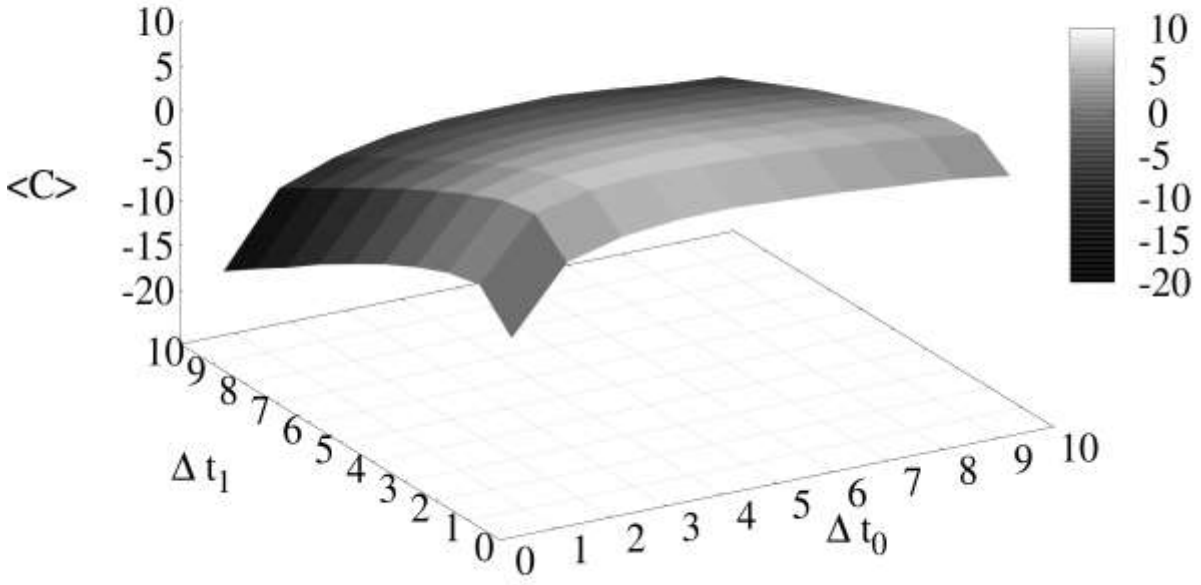
Cost-effective Attribute Retrieval

- Assumptions
  - We know statistics of attribute changes,
  - We can assign the following costs
  - C_a – cost of a check
  - C_{tp} – our gain when policy is satisfied
  - C_{fn} – our loss when policy is failed
- Compute the average cost of a usage session $C = c_{tp} \cdot \tau_g + c_{fn} \cdot \tau_b + C_a \cdot (n+1)$
- Retrieve attributes to maximize the profit $\arg\max_Q \langle C \rangle_Q$



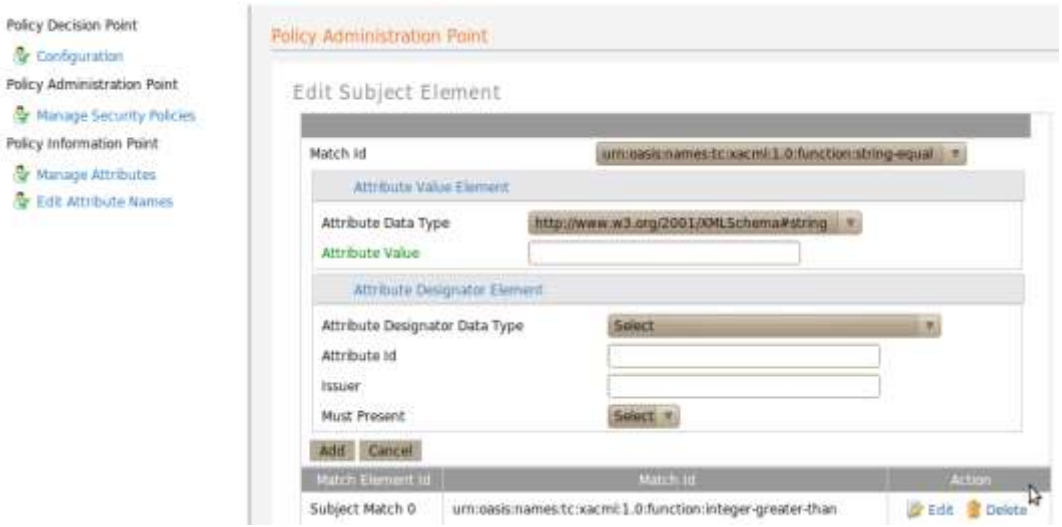
Example. Reputation

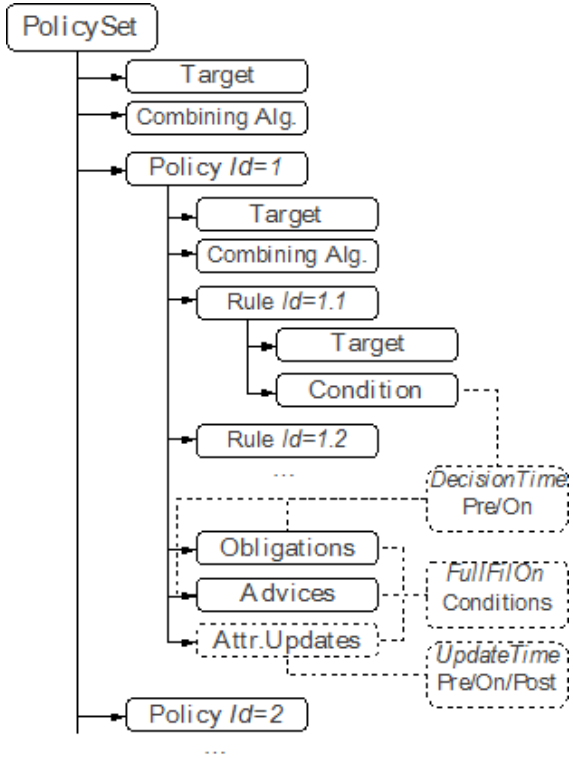
- If *reputation* is *general* – retrieve after 4.7 min
- If *reputation* is *normal* – retrieve after 1.5 min



Prototype

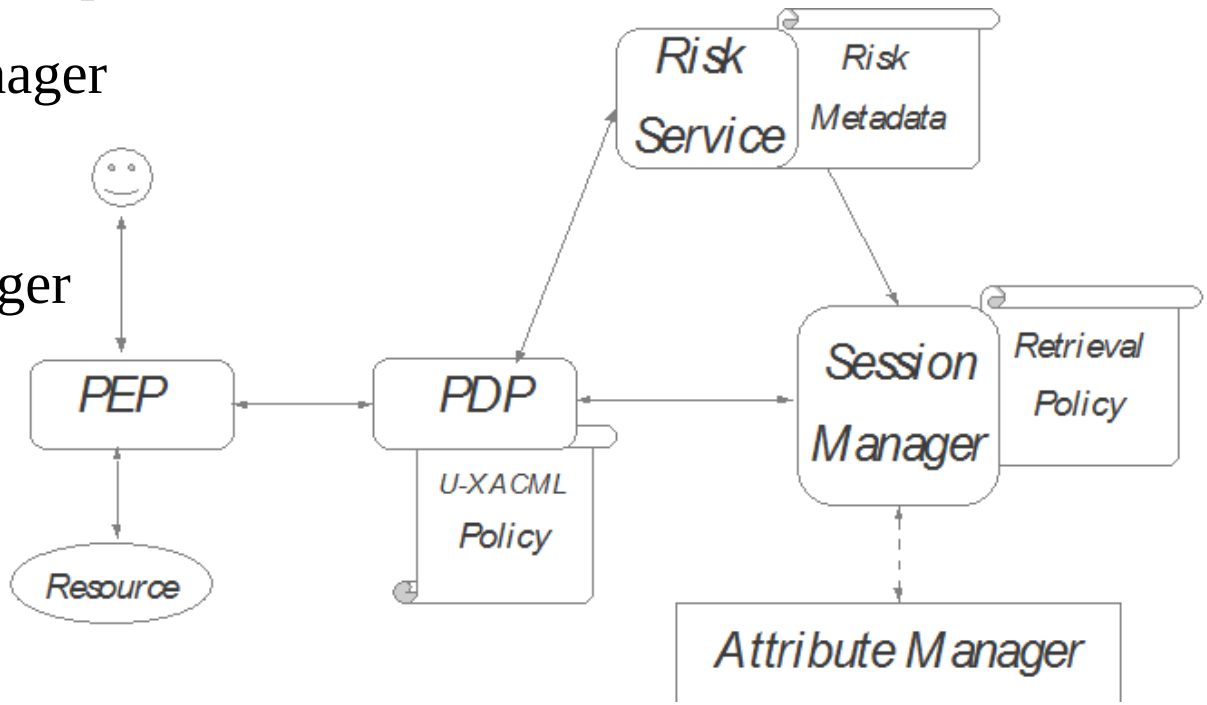
- It uses U-XACML as policy language





Reference Architecture

- PEP: policy enforcement point
- PDP: policy decision point
- AM: attribute manager
- RS: risk service
- SM: session manager



Components

- Session Manager
  - Enforces attribute retrieval
  - Keeps observed attributes
  - Repeatedly triggers PDP for the access reevaluation

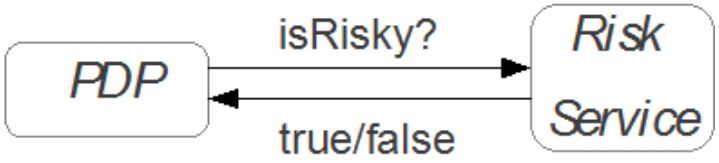
- Risk Service
  - Stores statistics of attribute changes and all costs
  - Computes functions
    - isRisky
    - getRiskTolerance

Cost-effective Access Decision

- Combine UCON and Risk policies as “deny-override”

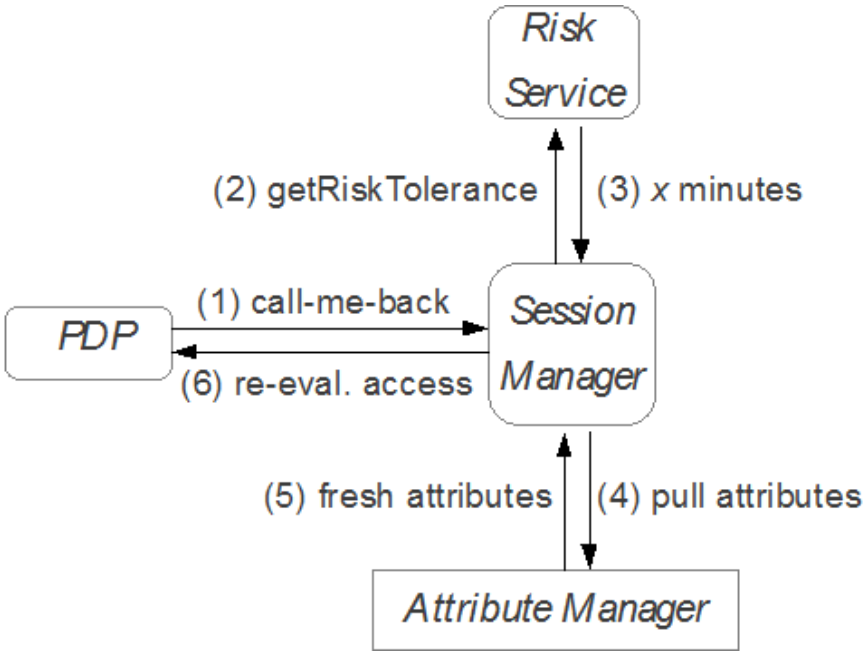
```
<PolicySet PolicyCombiningAlgId="deny-override" ... >
 <!-- U-XACML policy, e.g., a reputation is above a threshold --!>
 <Policy ... > ... </Policy>

 <!-- RISK POLICY --!>
 <Policy ... >
  <Rule Effect="Deny" ... > <Condition>
   <Apply FunctionId="isRisky">
    <AttributeDesignator Category="subject" AttributeId="reputation"/>
    <AttributeDesignator Category="subject" AttributeId="reputation:bad-values-domain"/>
   </Apply>
 </Condition></Rule></Policy>
</PolicSet>
```

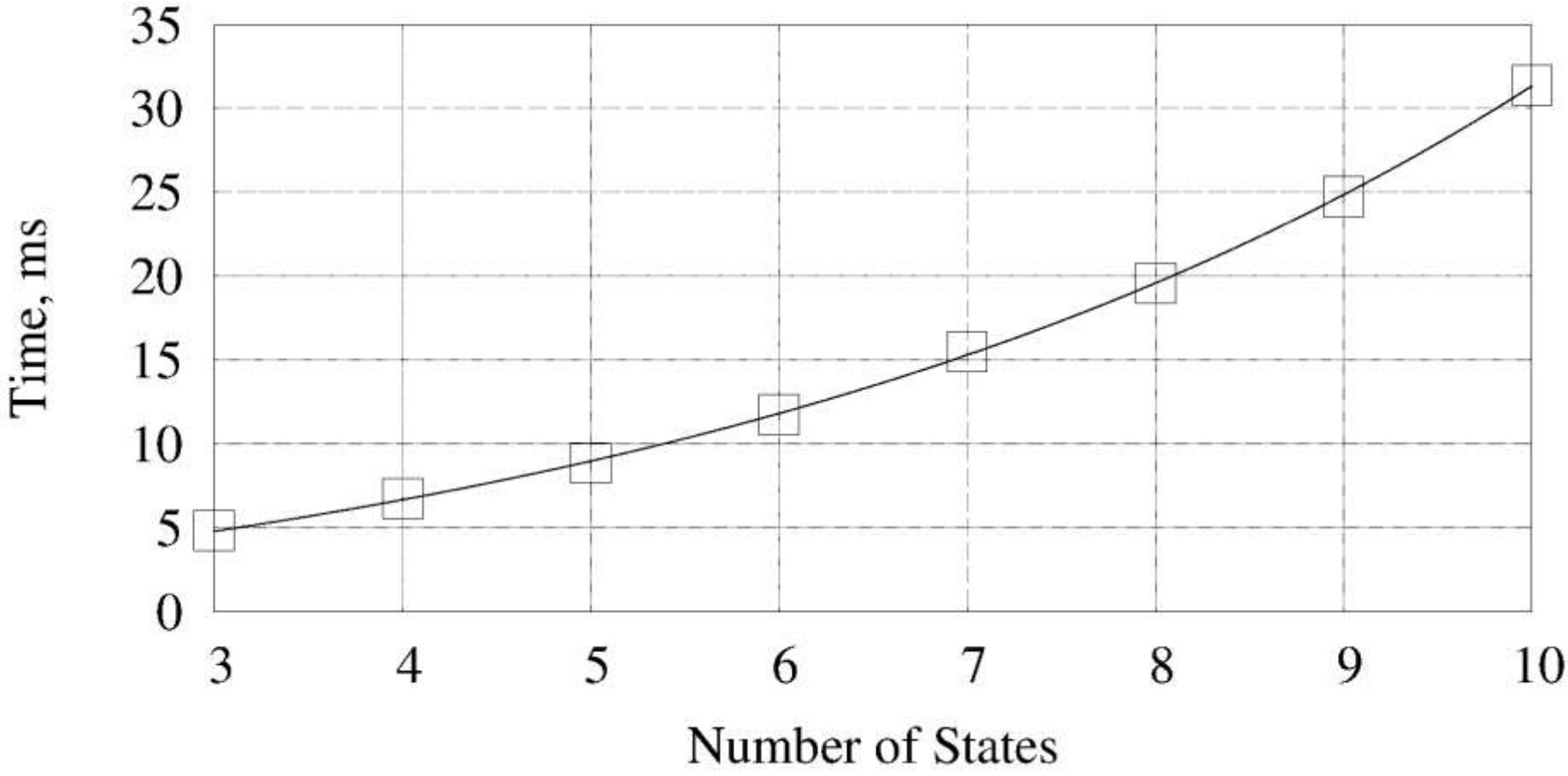


Cost-effective Attribute Retrieval

- PDP sends a “call-me-back” request to SM
- SM enforces attribute retrieval:
  - Asks RS when the next attribute query should be performed
  - Waits until this time elapses and then pulls fresh attributes
- SM triggers PDP for access reevaluation with new observed attributes
- … and continue

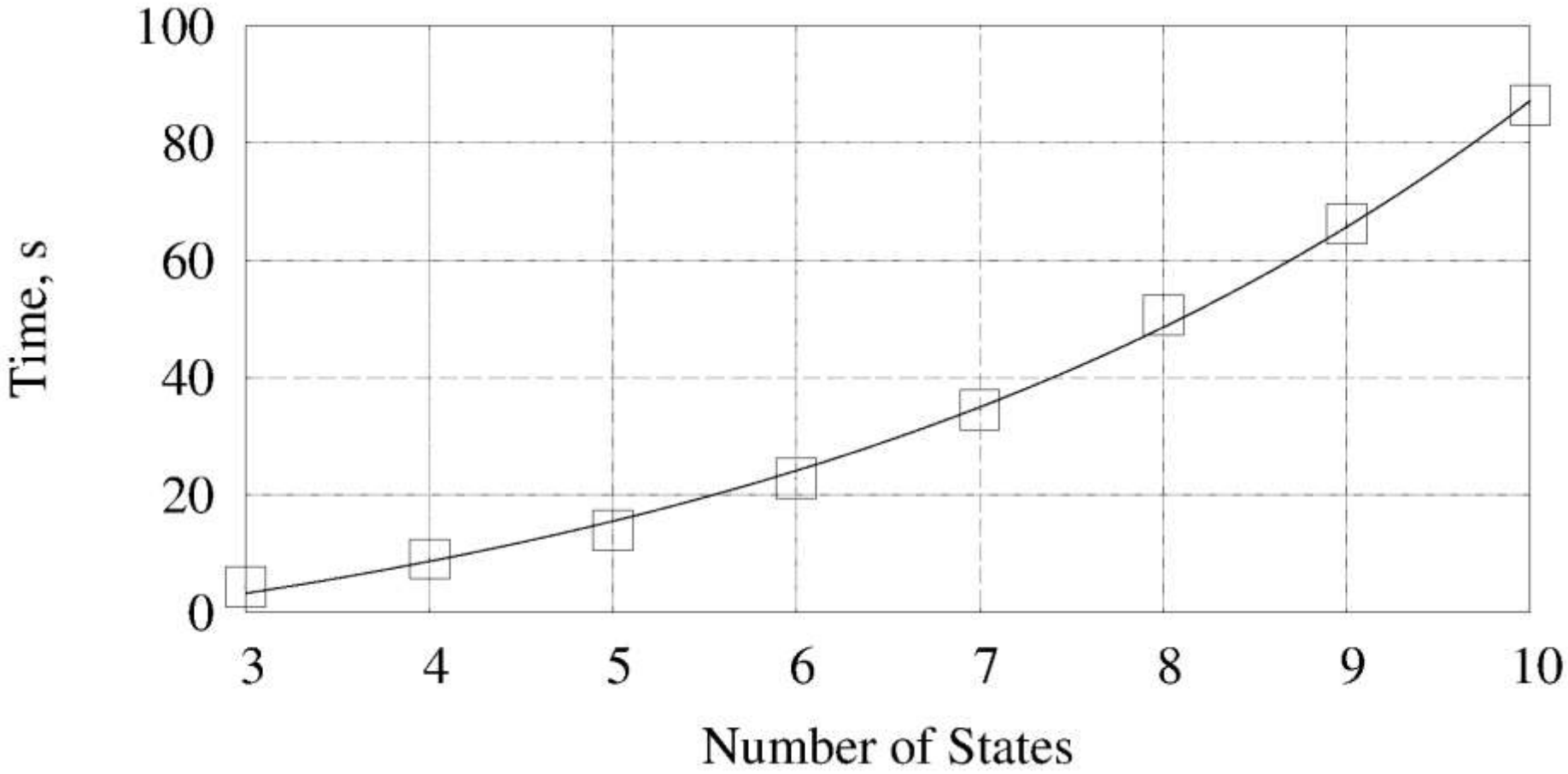


Performance: isRisky



Performance: getRiskTolerance

Calculated only once at deployment time



Conclusions

- We proposed a set of quantitative methods for a cost-effective enforcement of UCON policies:
  - Access decision making with uncertain attributes
  - Attribute retrieval strategies

To sum up

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

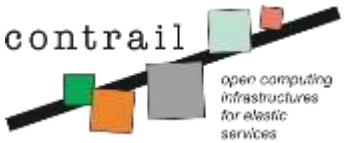
UCON for android is work in progress

Some selected references

- Georgios Karopoulos, Paolo Mori, Fabio Martinelli: Usage control in SIP-based multimedia delivery. Computers & Security 39: 406-418 (2013)
- Leanid Krautsevich, Aliaksandr Lazouski, Fabio Martinelli, Paolo Mori, Artsiom Yautsiukhin: Integration of Quantitative Methods for Risk Evaluation within Usage Control Policies. ICCCN 2013
- Aliaksandr Lazouski, Gaetano Mancini, Fabio Martinelli, Paolo Mori: Usage control in cloud systems. ICITST 2012
- On Usage Control for GRID Systems (FGCS 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)
- 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)

Supporting EU projects

- GRIDtrust
- CONSEQUENCE
- CONTRAIL
- COCO-Cloud



Thanks and contacts

- Fabio.Martinelli@iit.cnr.it
- http://security.iit.cnr.it