Service mediation and negotiation bootstrapping as first achievements towards self-adaptable grid and cloud services

Prepared by Faisal Sibai for CS 895

Presentation Agenda

- Some definitions and clarification
- Objective of the paper
- Introduction
- Adaptable, Versatile, and Dynamic Services
- GRID Meta-Negotiations
- SLA Mappings
- VieSLAF Framework
- Conclusion and Future Work
Definitions

- **Grid Computing:** Combining multiple computer resources from multiple administrative domains to achieve a common goal or large task
  - Loosely coupled
  - Heterogeneous
  - Geographically dispersed
  - Usually constructed with aid of general-purpose grid software libraries (middleware)
- Examples: Utility companies, Software as a Service (SaaS)

- **Cloud Computing:** Shared resources, Software and information are provided to computers and devices in an on demand fashion.
  - No ownership for the physical infrastructure
  - Consuming resources as a services and pay for resource usage
  - Multiple tenants (public clouds)
  - Subscription basis
  - SaaS, IaaS, and PaaS
  - Examples: Amazon EC2, Google Apps Engine

- **ICT:** Information and communication technologies

Primary Objective

• Presenting results of establishing adaptable, versatile, and dynamic services considering negotiation bootstrapping and service mediation.

• Meta-negotiation and SLA mapping solutions for Grid/Cloud services

• Prerequisites for the establishment of autonomic Grid/Cloud service

• Document models for the specification of meta-negotiations and SLA mapping

• Sample architecture for management of meta-negation and SLA mapping

• Service negotiation in heterogeneous Grids
Introduction

• SOA a good approach for Implementing ICT systems

• SOA has been widely adopted for ICT systems, yet dynamism and adaptivity are not exploited.

• SOA and web services are a candidate technologies for Grid and cloud computing services

• Existing Grid/Cloud frameworks assume that communication partner knows about the negotiation protocols and that they have a matching SLA template, this is an unrealistic assumption, services are discovered dynamically and on-demand

• Meta-negotiation is required to reach agreement on elements (protocol, security, etc.)

• SLA mapping is required due to differences in terminology (price vs. service price)

• Achieving Grid/ Cloud service via autonomic principles
Adaptable, Versatile, and Dynamic Services

• SOA systems should react to environmental changes, software failures, and other events exploiting the self-* properties.

• Identified Objectives:
  o Negotiation bootstrapping and service mediation:
    • Facilitating communication between publically available services
    • Obstacles arise when service providers provide unique negotiation protocols expressed in different languages (terms of contract)
    • Proposal: automatic bootstrapping between different protocols and contract formats.
  o Service Enforcement:
    • Services may fail or contracts between services may be violated.
    • Proposal: Develop methods for service enforcement where failures malfunctions are repaired.
    • Developing knowledge bases where directives, policies and rule failure adjustment and repair may be specified and stored.
  o Service Adaptivity:
    • Service failures or violation of e-agreements must be detected.
    • Proposal: Modeling of intelligent logging capabilities for single and composite services. Measurement of service execution and QoS are needed.
  o Service Governance:
    • Proposal: Policies and rules for service enforcement should not be defined in static way (rule-evolution over time)
Adaptable, Versatile, and Dynamic Services

1. **Monitoring**: QoS managed element is monitored using software sensors
2. **Analysis**: monitored and measured metrics are analyzed using knowledge base.
3. **Planning**: delivers necessary changes on the current setup based on the evaluated rules and results of analysis.
4. **Execution**: planned changes are executed using software actuators

**Autonomic computing principles**

General Architecture of an Autonomic System Explained on a QoS Example
Adaptable, Versatile, and Dynamic Services

- Candidate services are selected
- Knowledge base is queried, potential bootstrapping strategies are found
- Semi-automatic way for new strategies
- Negotiation is started using bootstrapping strategies
- Inconsistencies in SLA templates discovered
- SLA mappings analyzed
- New SLA mappings defined
- New SLA mappings applied

Negotiation Bootstrapping and Service Mediation as Part of the Autonomic Process
Grid Meta-negotiations

Meta-Negotiation Scenario:

1. **Publishing**: Service Provider publishes description and conditions of supported negotiation protocols into registry
2. **Lookup**: Service consumer performs lookup on registry DB by submitting their own documents describing the negotiations they are looking for.
3. **Matching**: registry discovers service providers who support the negotiation process and returns the documents published by service provider.
4. **Negotiation**: negotiation starts according to the conditions specified in the providers document.

Meta-Negotiation Document (MND):

```xml
1. <meta-negotiation ...
2. ...
3. <pre-requisites>
4. <role name="consumer"/>
5. <security>
6. <authentication value="GSI" location="uri"/>
7. </security>
8. <negotiation-term name="beginTime"/>
9. <negotiation-term name="endTime"/>
10. ...
11. </negotiation-term>
12. </pre-requisites>
13. <negotiation>
14. <negotiation>
15. <document name="WSLA" value="uri" version="1.0"/>
16. <protocol name="alternateOffers" schema="uri" version="1.0" location="uri"/>
17. </negotiation>
18. <agreement>
19. <confirmation>
20. name="arbitrationService" value="uri"/>
21. </confirmation>
22. </agreement>
23. </negotiation>
24. </meta-negotiation>
```

**Pre-requisites:**
- role,
- security credentials: authentication/authorization
- negotiation terms: QoS attributes

**Negotiation**
- document
- Protocol

**Agreement:**
- Sign an agreement (maybe verified by a third party)

This has been excrementally evaluated and was presented in previous publications.
SLA Mappings

- Management of SLA Mappings:

  - Service Registry comprises different SLA templates, each representing a specific application domain (medical, telco, etc.)
  - **Step 1:** Service provider assigns service to a particular template.
  - **Step 2:** Assigns SLA mapping if necessary
  - **Step 3:** Service consumer searches for services using meta-data/search terms
  - **Step 4:** Define mappings to the appropriate template
  - **Step 5:** Negotiation starts. Templates are not static, SLAs are updated frequently trying to reflect the actual SLAs used (Based on assigned SLA mappings and predefined rules)

- SLA Mappings defined on XML level where users define XSL transformations
Scenario for SLA Mappings:

- SLA agreements defined using Web Service Level Agreement (WSLA)
- WSLA publicly available. Published in searchable registry
- Can be downloaded and compared with local template
- Can be done automatically using tools.
- Authors were developing a GUI which helps consumers to find suitable service categories
- If there are any inconsistencies discovered, service consumers may write rules from his/her local WSLA template
- Rules stored in DB and can be applied during runtime to remote WSLA.
- During negotiation transformation are performed from remote -> local and vice versa

Scenario for XSL Transformations
Negotiation may be done using none-matching WSLAs
SLA Mappings

- SLA Mappings Document (SMD):
  - Defined using XSLT (EXtensible Stylesheet Language Transformation) and XPath expressions (XML Path Language)
  - Simple example: price in Euro is transformed to an equivalent US dollar

Example XSL Transformation

```xml
1. ...
2. <xsl:template ...>
3.  <xsl:element name="Function" ...>
4.   <xsl:attribute name="type">
5.     <xsl:text>Times</xsl:text>
6.   </xsl:attribute>
7.   <xsl:attribute name="resultType">
8.     <xsl:text>double</xsl:text>
9.   </xsl:attribute>
10.  <xsl:element name="Operand" ...>
11.   <xsl:copy>
12.     <xsl:copy-of select="@*|node()"/>
13.   </xsl:copy>
14.  </xsl:element>
15.  <xsl:element name="Operand" ...>
16.   <xsl:element name="FloatScalar" ...>
17.     <xsl:text>1.27559</xsl:text>
18.   </xsl:element>
19. </xsl:element>
20. </xsl:element>
21. </xsl:template>
22. ...
```

- Times function: mapping rule returning Dollars (WSLA Specification)
  - Times function multiplies two operands:
    - The Dollar amount
    - The Dollar/Euro quote (not hard coded)

SLA mappings for simple and complex semantic mappings
**VieSLAF Framework**

- VieSLAF architecture:
  - Represents first prototype for management of self-governing ICT infrastructure.
  - Enables application developers to efficiently develop adaptable service-oriented applications.
  - Facilitates and manages QoS models (meta negotiations and SLA Mappings)
VieSLAF Framework

VieSLAF architecture Steps:

- **Step 1**: Users may access the registry using a GUI, browse through existing templates and meta-negotiation documents using the MN and SLA mapping middleware.

- **Step 2**: Service provider specify MN documents and SLA mappings using the MN and SLA mapping middleware and submit it to the registry.

- **Step 3**: Service consumer may query existing meta-negotiation documents, define own SLA mappings to remote templates and submit it to the registry. MN and SLA mapping middleware on both sides (provider's and consumer's) facilitates management of MNs and SLA mappings.

- **Step 4**: Submitted MN documents and SLA mappings are parsed and mapped to a predefined data model.

- **Step 5**: After meta-negotiation and pre-selection of services, service negotiation may start using the negotiation protocols, document languages, and security standards as specified in the MN document.

- **Step 6**: During the negotiation SLA mappings and XSLT transformations are applied.

- **Step 7**: After the negotiation, invocation of the service methods may start.

- **Step 8**: SLA parameters are monitored using the monitoring service.

- **Step 9**: Based on the submitted SLA mapping publicly available SLA templates are adapted reflecting the majority of local SLA templates.
VieSLAF Framework

- VieSLAF components:
  - Knowledge Base:
    - Responsible for storing SLA templates, SLA mappings and meta-negotiation documents
    - MS-SQL 2008 with web services front end that provides the interface for management of LA mappings and PostgreSQL for management for meta-negotiations.
    - For scalability a cloud service would be better
    - DB is manipulated base on role-model. Registry methods implemented as Windows Communication Foundation (WCF) can only be accessed with appropriate access rights:
      - Service consumer: able to search suitable services for selected service categories, May also create SLA-Mappings. May submit and query MN documents
      - Service provider: able to publish their services and bind it to specific template categories. May submit and query MN documents
      - Registry admin

- Meta-negotiation Middleware:
  - Facilitates the publishing of meta-negotiation documents into registry and the integration of meta-negotiation framework into existing clients.
  - After query registration and applying client-based strategy meta-negotiation document is parsed.
  - Meta-negotiation information is incorporated into existing client software using dependency injection framework (Spring). Follows an Inversion of control approach where software is configured at runtime to invoke services that are discovered dynamically rather than known and referenced beforehand
  - Semi-automatic integration of integration of existing clients into meta-negotiation middleware
VieSLAF Framework

• SLA Mapping Middleware:
  o This section discusses the usage of some WCF services like RegistryAdministrationService, WSLAMappingService, and WSLAQueringService. (for more info on each please visit the original paper)
  o Service consumers also use some of methods listed above in order to search for appropriate services or create SLA-Mappings.
  o .Net 3.5 and LINQ are used during transformation implementation.

• Monitoring Service:
  o The aim is to frequently check the status of the SLA parameters of an SLA agreement and deliver information to the service consumer/Provider.
  o It also monitors values of SLA parameters.
  o Its implemented as an internal registry service (similar to the other services mentioned)
  o For steps on how monitoring is setup please visit the original paper.

• Adaption:
  o Remote SLA templates should not be statically defined, they should reflect consumer/provider needs
  o For steps on how monitoring is setup please visit the original paper.
Conclusion and Future work

• Achieving goals of self-Governing ICT Infrastructure using principles of autonomic computing

• Discussed meta-negotiation and SLA mapping solutions for Grid/Cloud service bridging the gap between current QoS Models and Grid/Cloud Middleware.

• Discussed approaches for meta-negotiation and SLA mapping representing partial implementation for negotiation bootstrapping and service medication.

• Presented the VieSLAF framework used for management of meta-negotiation and SLA-mappings

• SLA template adaption based on SLA mappings

• **Future**: plan to implement bootstrapping strategies where consumer/provider understand different negotiation protocols and document languages can communicate