Introduction to Grid Architecture

What is Architecture?

- Design
- The way components fit together

Introduction to Grid Architecture

Why Discuss Architecture?

- Descriptive
  - Provide a common vocabulary for use when describing Grid systems
- Guidance
  - Identify key areas in which services are required
- Prescriptive
  - Define standard protocols and APIs to facilitate creation of interoperable Grid systems and portable applications

Introduction to Grid Architecture

The nature of grid architecture

- A grid architecture identifies fundamental system components, specifies the purpose and function of these components, and indicate how these components interact.

Introduction to Grid Architecture

The Nature of Grid Architecture

- Grid's protocols allow **VO** users and resources to negotiate, establish, manage and exploit sharing relationships.
  - Interoperability a fundamental concern
  - The protocols are critical to interoperability
  - Services are important
  - We need to consider APIs and SDKs

VO: Virtual Organization

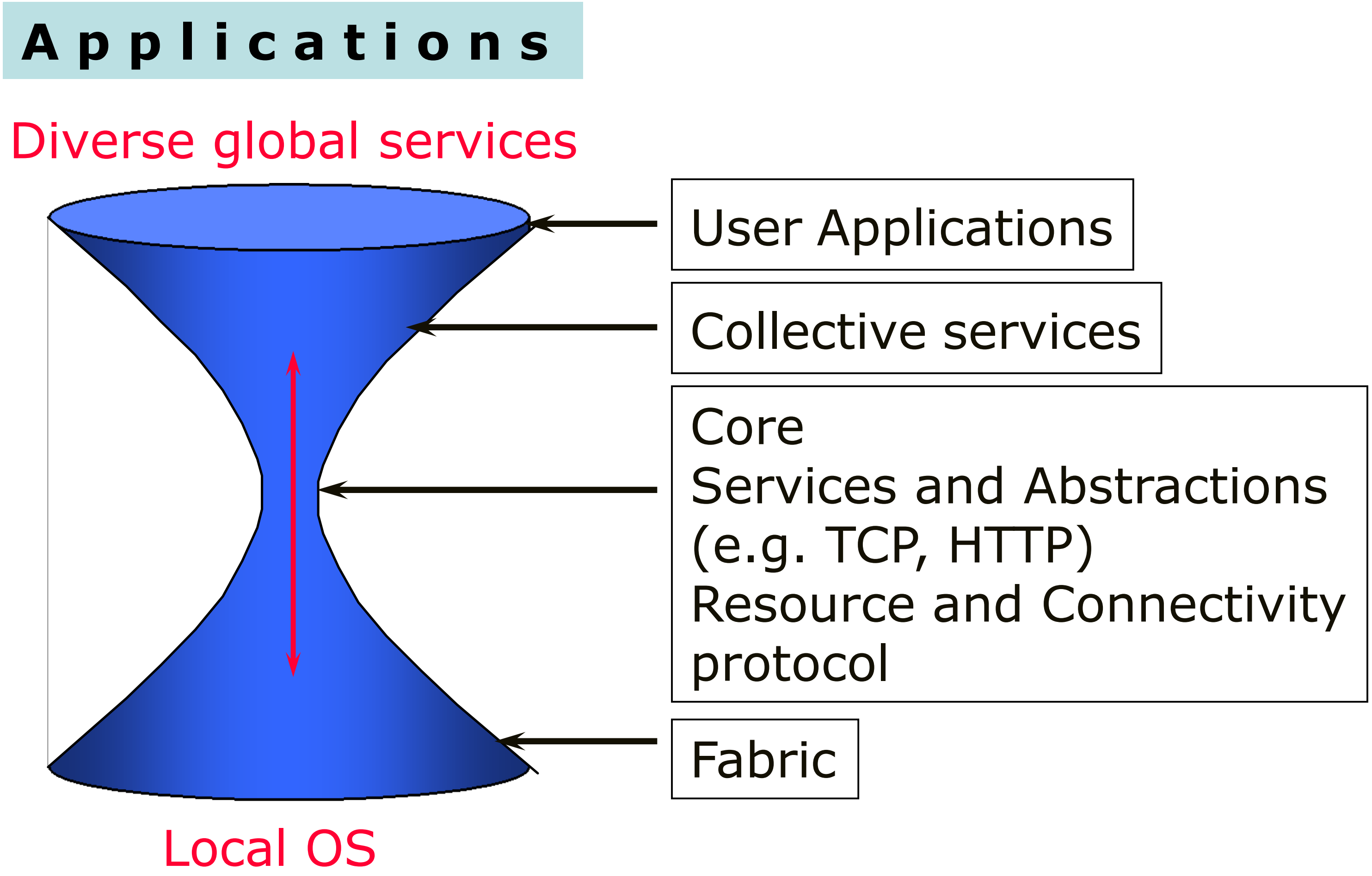
Introduction to Grid Architecture

Grid architecture requirements

- The components are
  - numerous
  - owned and managed by different, potentially mutually distrustful organisations and individuals
  - may be potentially faulty
  - have different security requirements and policies
  - heterogeneous
  - connected by heterogeneous, multilevel networks
  - have different resource management policies
  - are likely to be geographically separated

Key Components

The Hourglass Model



Applications

Diverse global services

User Applications

Collective services

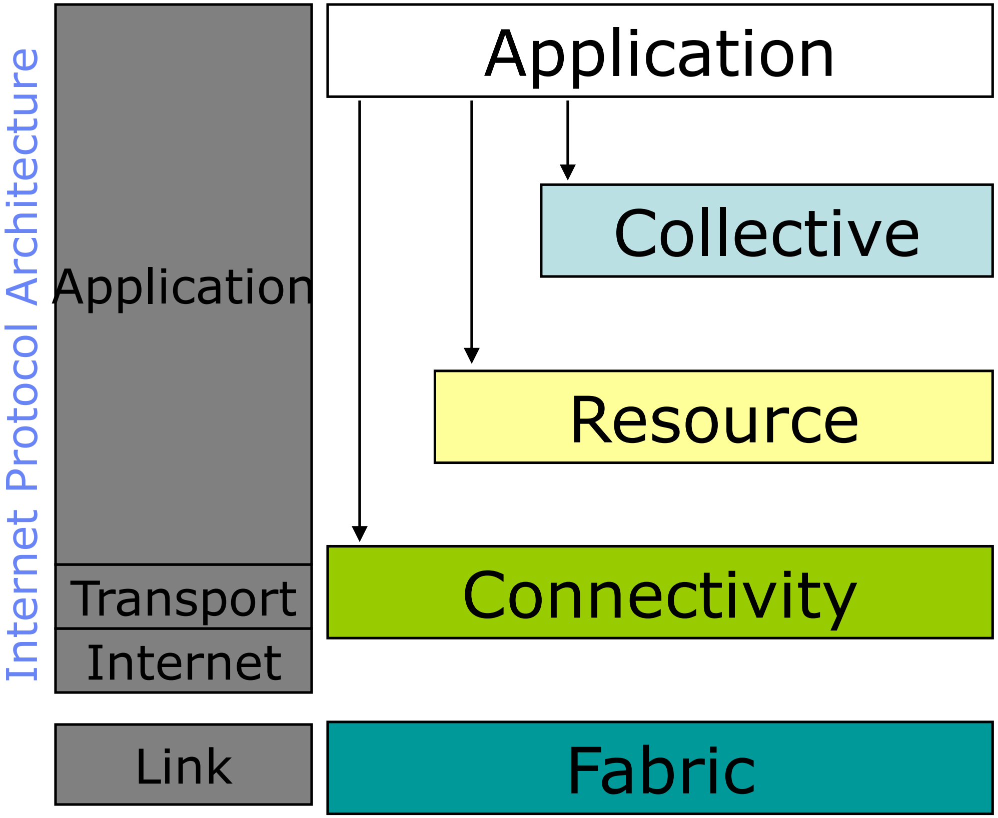
Core
Services and Abstractions
(e.g. TCP, HTTP)
Resource and Connectivity
protocol

Fabric

Local OS

Key Components

Layered Grid Architecture
(By Analogy to Internet Architecture)



"Coordinating multiple resources": ubiquitous infrastructure services, app-specific distributed services

"Sharing single resources": negotiating access, controlling use

"Talking to things": communication (Internet protocols) & security

"Controlling things locally": Access to, & control of, resources

Key Components

Layered Grid Architecture: Fabric Layer

- Just what you would expect: the diverse mix of resources that may be shared
  - Individual computers, Condor pools, file systems, archives, metadata catalogs, networks, sensors, etc., etc.
- Defined by interfaces, not physical characteristics

Key Components

Layered Grid Architecture:Connectivity Layer

- Communication
  - – Internet protocols: IP, DNS, routing, etc.
- Security: Grid Security Infrastructure (GSI)
  - – Uniform authentication, authorization, and message protection mechanisms in multi-institutional setting
  - – Single sign-on, delegation, identity mapping
  - – Public key technology, SSL, X.509, GSS-API
  - – Supporting infrastructure: Certificate Authorities, certificate & key management, …

GSI: www.gridforum.org/security

Key Components

Layered Grid Architecture:Resource Layer

- The architecture is for the secure negotiation, initiation, monitoring, control, accounting, and payment of sharing operations on individual resources.
  - Information Protocols (inform about the structure and state of the resource)
  - Management Protocols (negotiate access to a shared resource)

Key Components

Layered Grid Architecture:Resource Layer

- Grid Resource Allocation Mgmt (GRAM)
  - Remote allocation, reservation, monitoring, control of compute resources
- GridFTP protocol (FTP extensions)
  - High-performance data access & transport
- Grid Resource Information Service (GRIS)
  - Access to structure & state information
- Network reservation, monitoring, control
- All built on connectivity layer: GSI & IP

GridFTP: www.gridforum.org
GRAM, GRIS: www.globus.org

Key Components

Layered Grid Architecture:Collective layer

- Coordinating multiple resources
- Contains protocols and services that capture interactions among a collection of resources
- It supports a variety of sharing behaviours without placing new requirements on the resources being shared
- Sample services: directory services, co-allocation, brokering and scheduling services, data replication services, workload management services, collaboratory services

Key Components

Layered Grid Architecture:Collective Layer

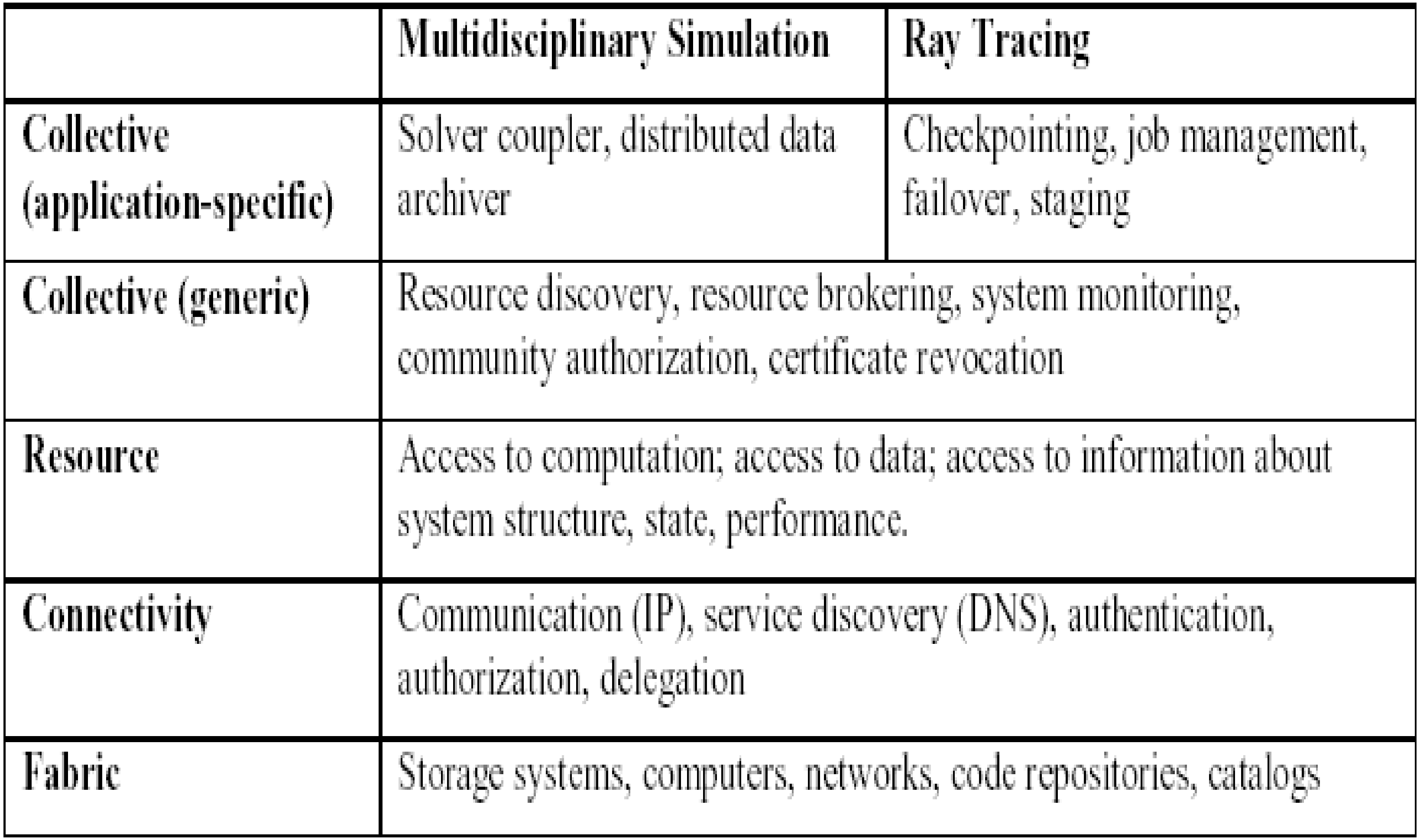
- Index servers aka metadirectory services
  - Custom views on dynamic resource collections assembled by a community
- Resource brokers (e.g., Condor Matchmaker)
  - Resource discovery and allocation
- Replica catalogs
- Replication services
- Co-reservation and co-allocation services
- Workflow management services
- Etc.

Key Components

Layered Grid Architecture:Applications layer

- There are user applications that operate within the VO environment
- Applications are constructed by calling upon services defined at any layer
- Each of the layers are well defined using protocols, provide access to services
- Well-defined APIs also exist to work with these services

Key Components



	Multidisciplinary Simulation	Ray Tracing
Collective (application-specific)	Solver coupler, distributed data archiver	Checkpointing, job management, failover, staging
Collective (generic)	Resource discovery, resource brokering, system monitoring, community authorization, certificate revocation	
Resource	Access to computation; access to data; access to information about system structure, state, performance.	
Connectivity	Communication (IP), service discovery (DNS), authentication, authorization, delegation	
Fabric	Storage systems, computers, networks, code repositories, catalogs	

Key Components

Where Are We With Architecture?

- No “official” standards exist
- But:
  - Globus Toolkit™ has emerged as the de facto standard for several important Connectivity, Resource, and Collective protocols
  - Technical specifications are being developed for architecture elements: e.g., security, data, resource management, information

Services in the Web and the Grid

Web services

- Define a technique for describing software components to be accessed, methods for accessing these components, and discovery methods that enable the identification of relevant service providers
- A distributed computing technology (like CORBA, RMI…)
- They allow us to create loosely coupled client/server applications.

Services in the Web and the Grid

Web Services:Advantages

- Platform and language independent since they use XML language.
- Most use HTTP for transmitting messages (such as the service request and response)

Services in the Web and the Grid

Web Services: Disadvantages

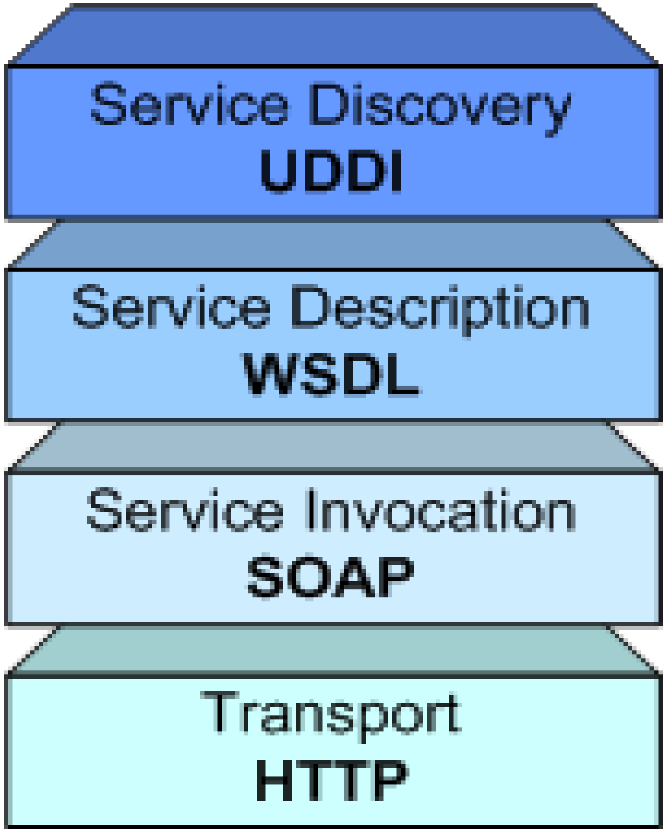
- Overhead : Transmitting data in XML is not as convenient as binary codes.
- Lack of versatility: They allow very basic forms of service invocation (Grid services make up this versatility).
  - Stateless:

    They can't remember what you have done from one invocation to another
  - Non-transient:

    They outlive all their clients.

Services in the Web and the Grid

Web Services Architecture



Find Web services which meet certain requirements
(Universal Description, Discovery and Integration)

Services describe their own properties and methods
(Web Services Description Language)

Format of requests(client) and responses (server)
(Simple Object Access Protocol)

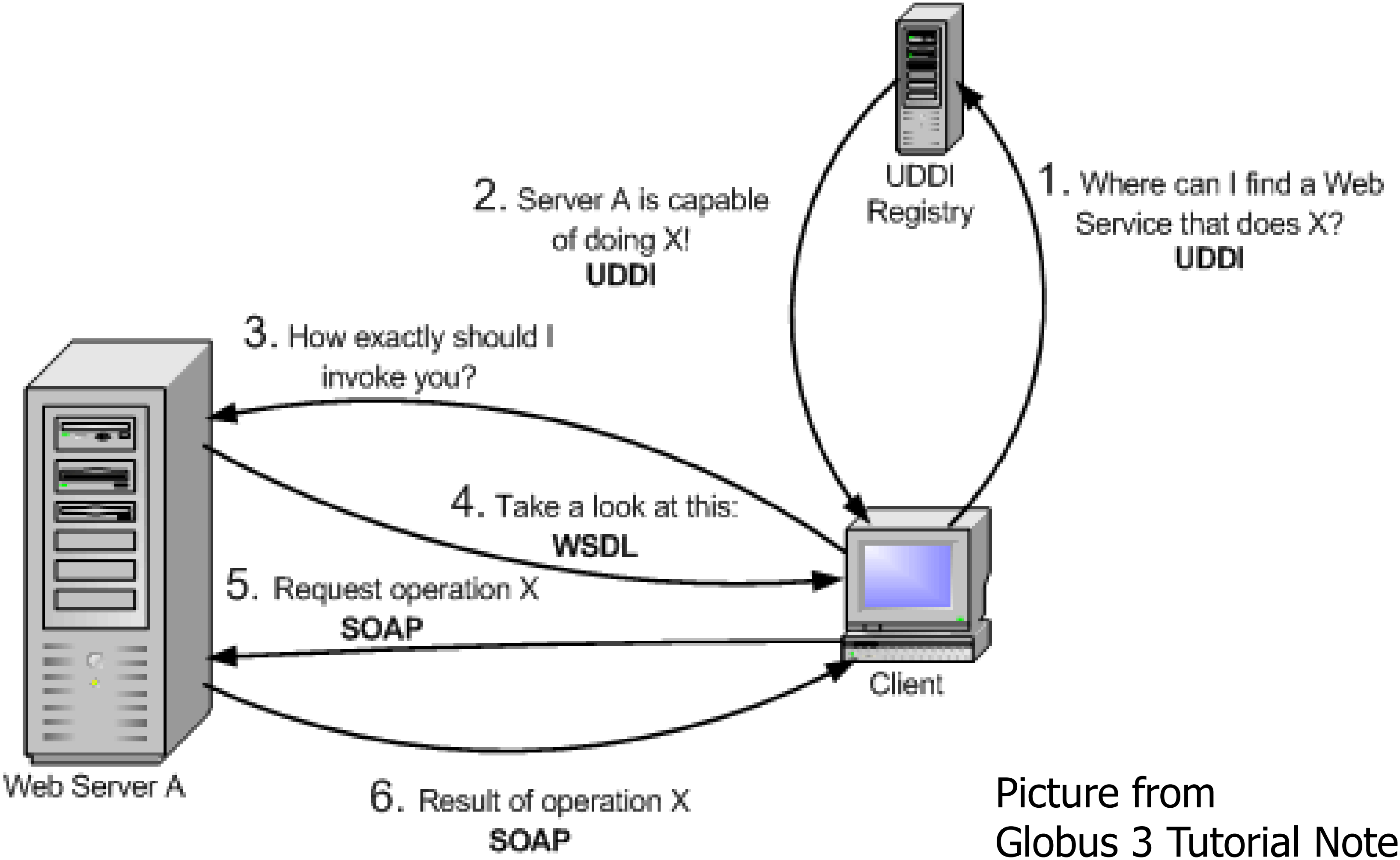
Message transfer protocol

(Hypertext Transfer Protocol)

Picture from Globus 3 Tutorial Notes www.globus.org

Services in the Web and the Grid

Invoking a Typical Web Service



Picture from
Globus 3 Tutorial Notes

Services in the Web and the Grid

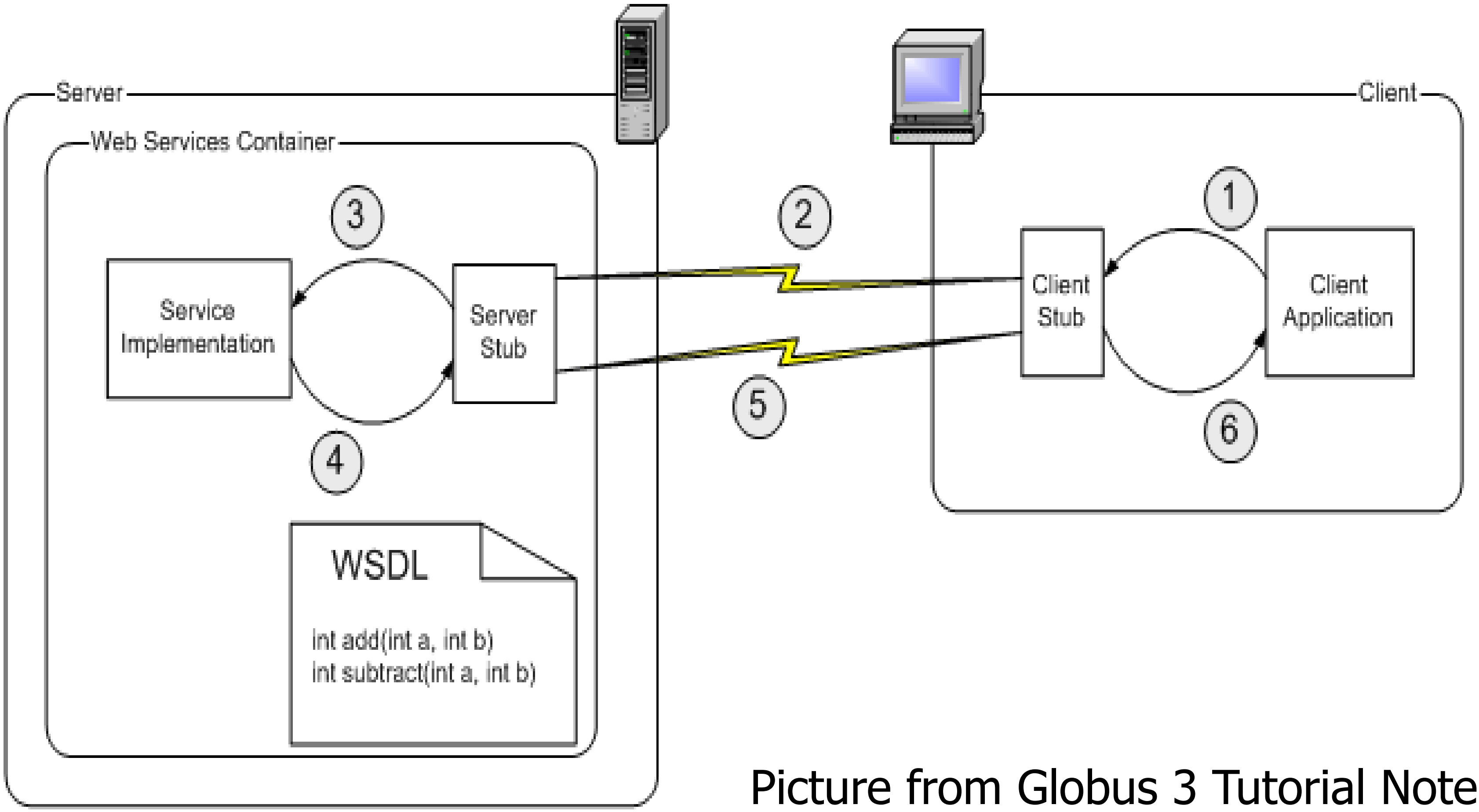
Web Service Addressing

- URI: Uniform Resource Identifiers
- URI and URL are practically the same thing.
  - Example: http://webservices.mysite.com/weather/us/WeatherService

- It can not be used with web browsers, it is meant for softwares.

Services in the Web and the Grid

Web Service Application



int add(int a, int b)
int subtract(int a, int b)

Picture from Globus 3 Tutorial Notes

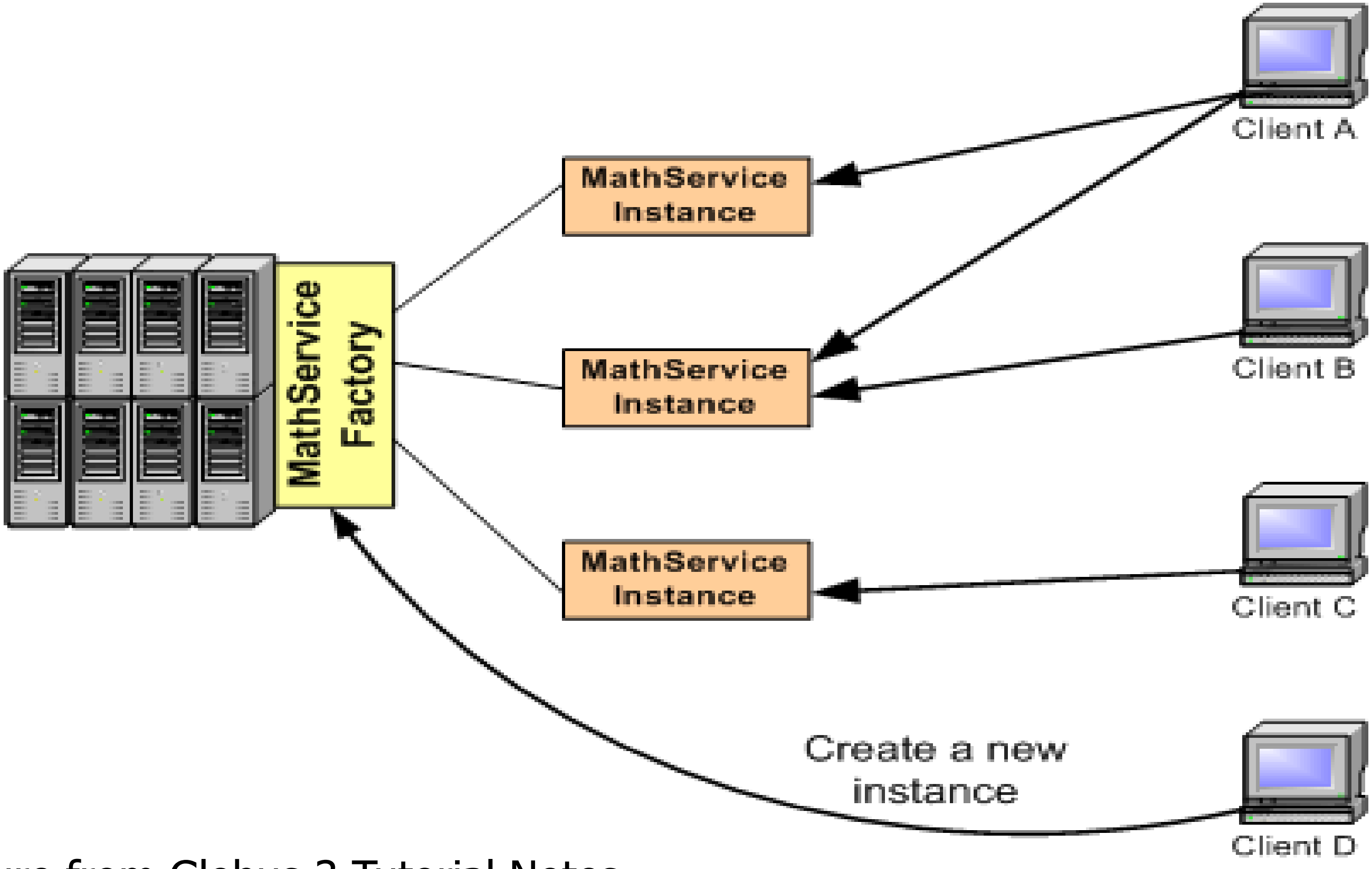
Services in the Web and the Grid

What is a Grid Service?

- It provides a set of well defined interfaces and that follows specific conventions.
- It is a web service with improved characteristics and services.
  - Improvement:
    - Potentially Transient
    - Stateful
    - Delegation
    - Lifecycle management
    - Service Data
    - Notifications
- *Examples* : computational resources, programs, databases…

Services in the Web and the Grid

Factories



Picture from Globus 3 Tutorial Notes

Services in the Web and the Grid

GSH & GSR

- GSH: Grid Service Handle (URI)
  - Unique
  - Shows the location of the service
- GSR: Grid Service Reference
  - Describes how to communicate with the service
  - As WS use SOAP, our GSR is a WSDL file.

Services in the Web and the Grid

Open Grid Services Architecture (OGSA)

- OGSA defines what Grid services are, what they should be capable of, what type of technologies they should be based on.
- OGSA does not give a technical and detailed specification. It uses WSDL.

Services in the Web and the Grid

Open Grid Services Infrastructure (OGSI)

- It is a formal and technical specification of the concepts described in OGSA.
- The Globus Toolkit 3 is an implementation of OGSI.
- Some other implementations are OGSI::Lite (Perl)1 and the UNICORE OGSA demonstrator2 from the EU GRIP project.
- OGSI specification defines grid services and builds upon web services.

Services in the Web and the Grid

OGSI

- OGSI creates an extension model for WSDL called GWSDL (Grid WSDL). The reason is:
  - Interface inheritance
  - Service Data (for expressing state information)
- Components:
  - Lifecycle
  - State management
  - Service Groups
  - Factory
  - Notification
  - HandleMap

Services in the Web and the Grid

Service Data Structure

```
<wsdl:definitions xmlns:tns="abc"
    targetNamespace="mynamespace">
  <gwsdl:portType name="AbstractSearchEngine">
    <wsdl:operation name="search" />
     ..........
    <sd:serviceData name="cachedURL" type="tns:
    cachedURLType“ mutability="mutable" nillable="true",
    maxOccurs="1"  minOccurs="0“ modifiable="true"/>
  </gwsdl:portType>
</wsdl:definitions>
```

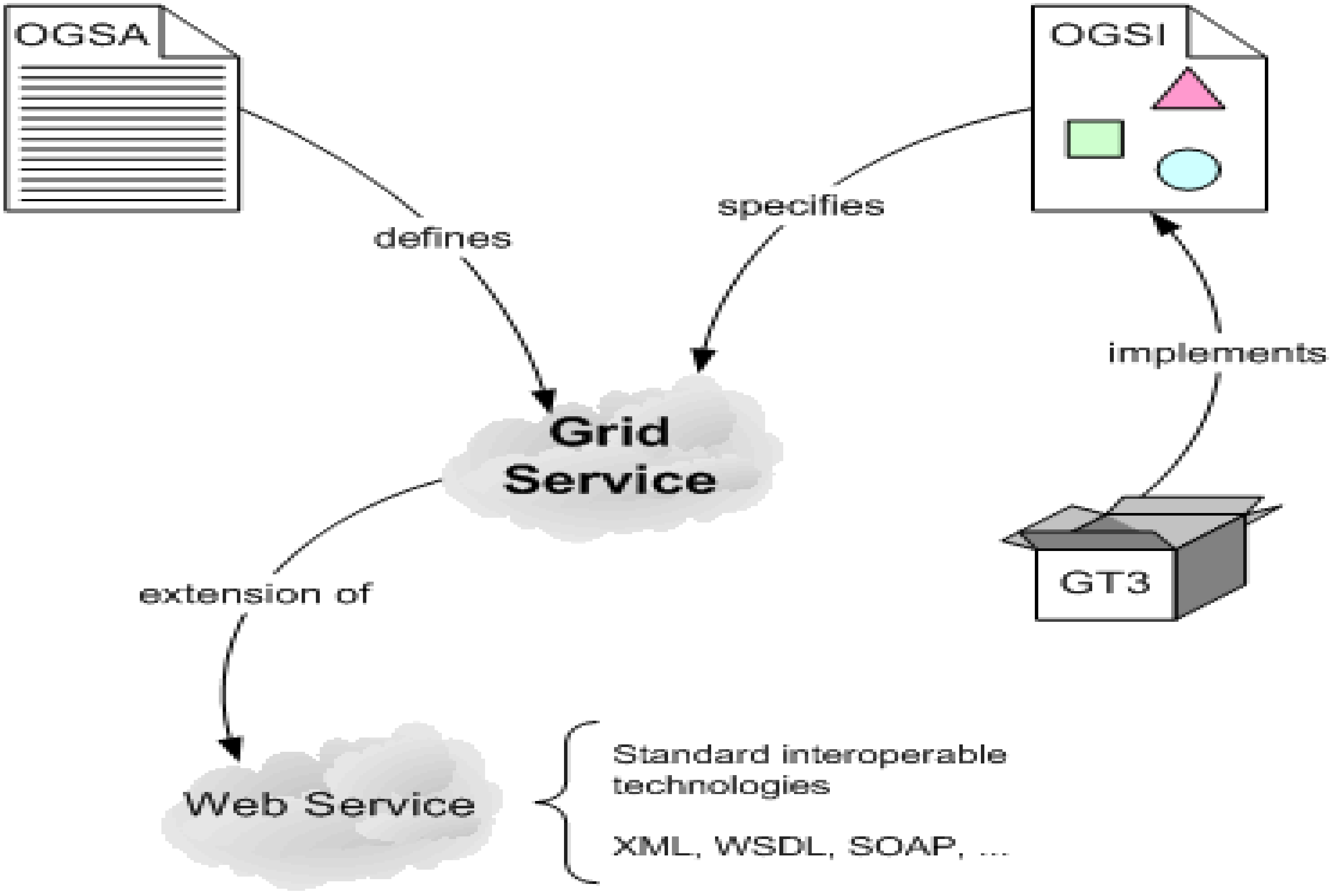
nillable: allows the element to have no value

modifiable: allows user override of the model element

mutable: service data element can change

Services in the Web and the Grid

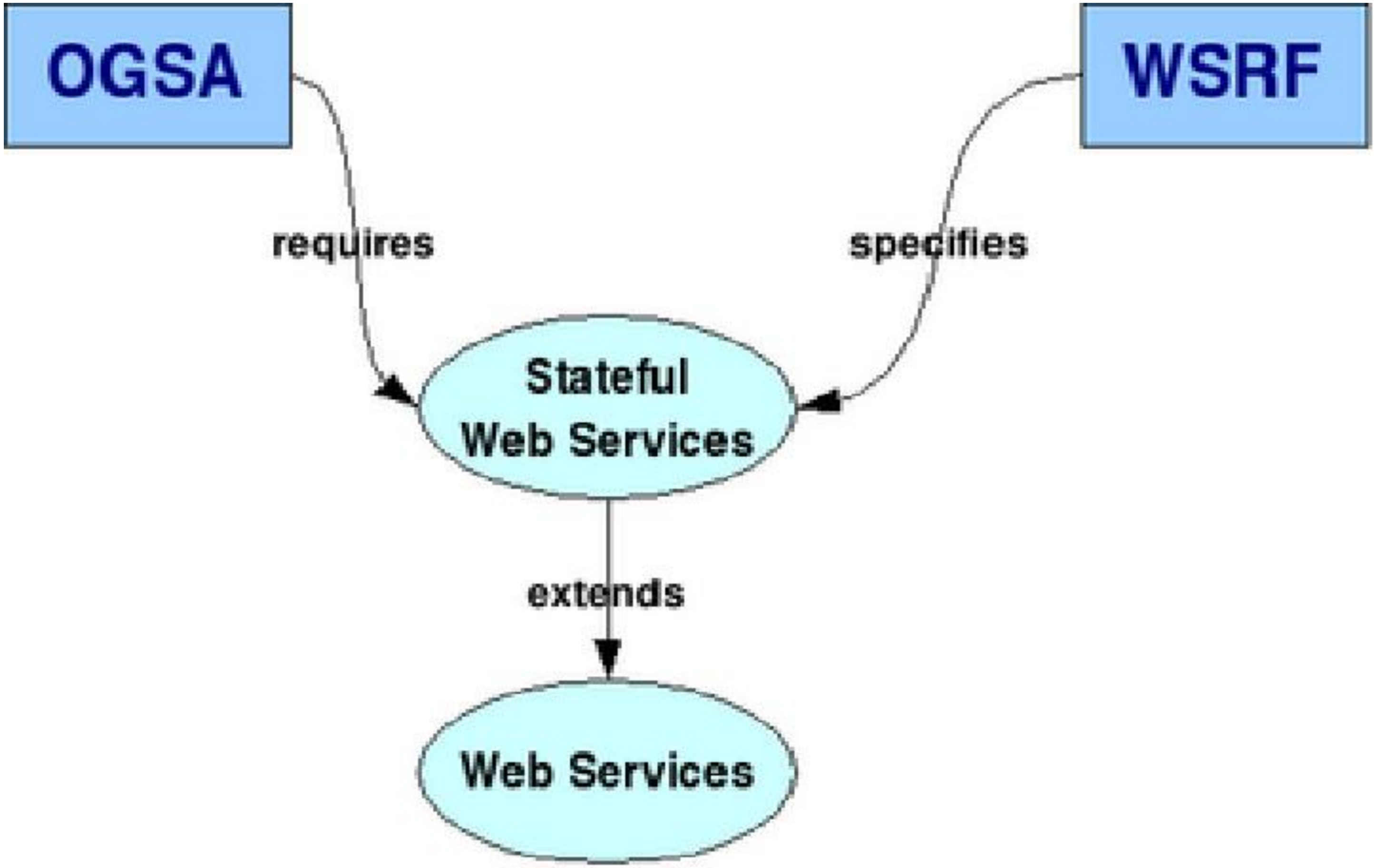
OGSA, OGSI, GT3



Picture from Globus 3 Tutorial Notes

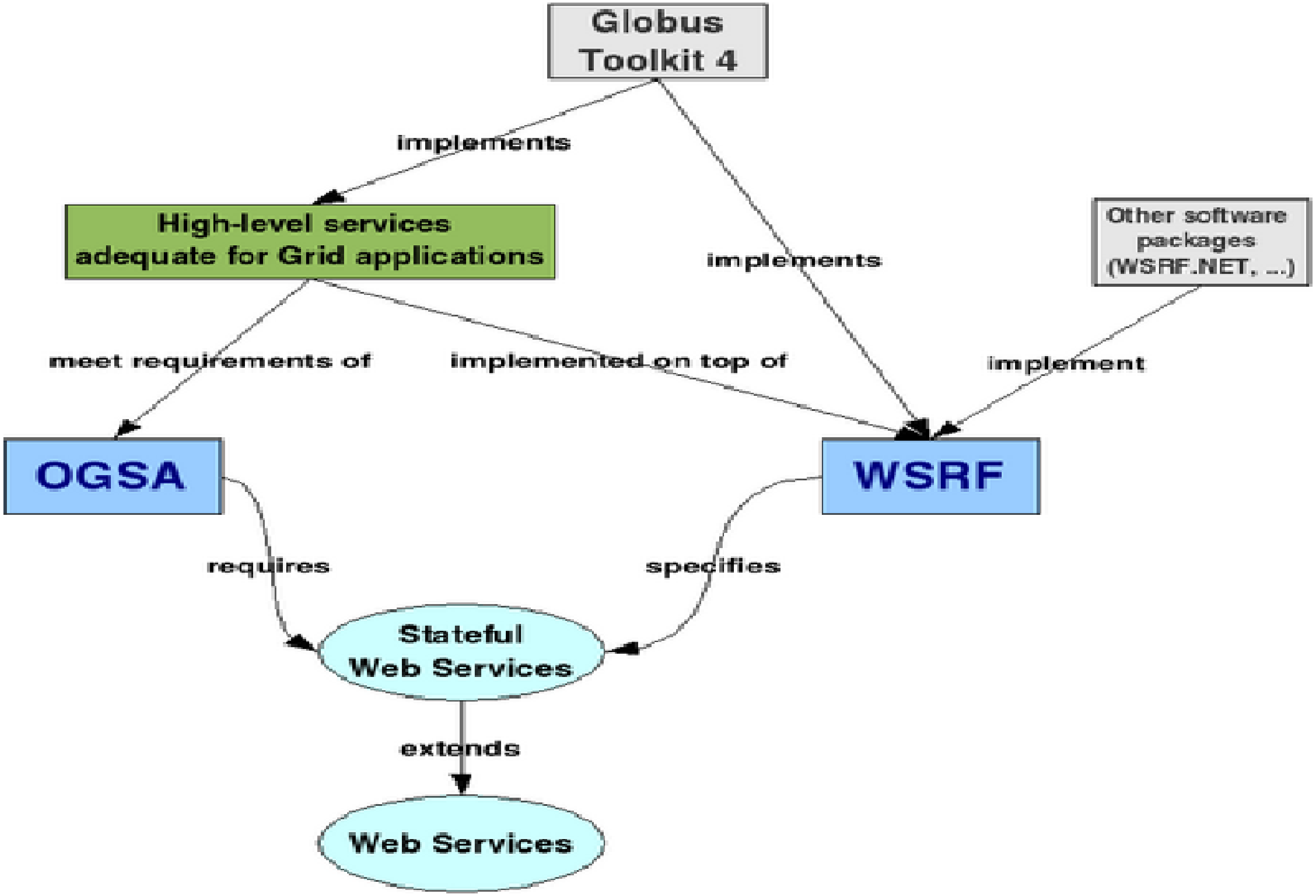
Services in the Web and the Grid

OGSA, WSRF



Web services and the Grid

OGSA, WSRF, GT4

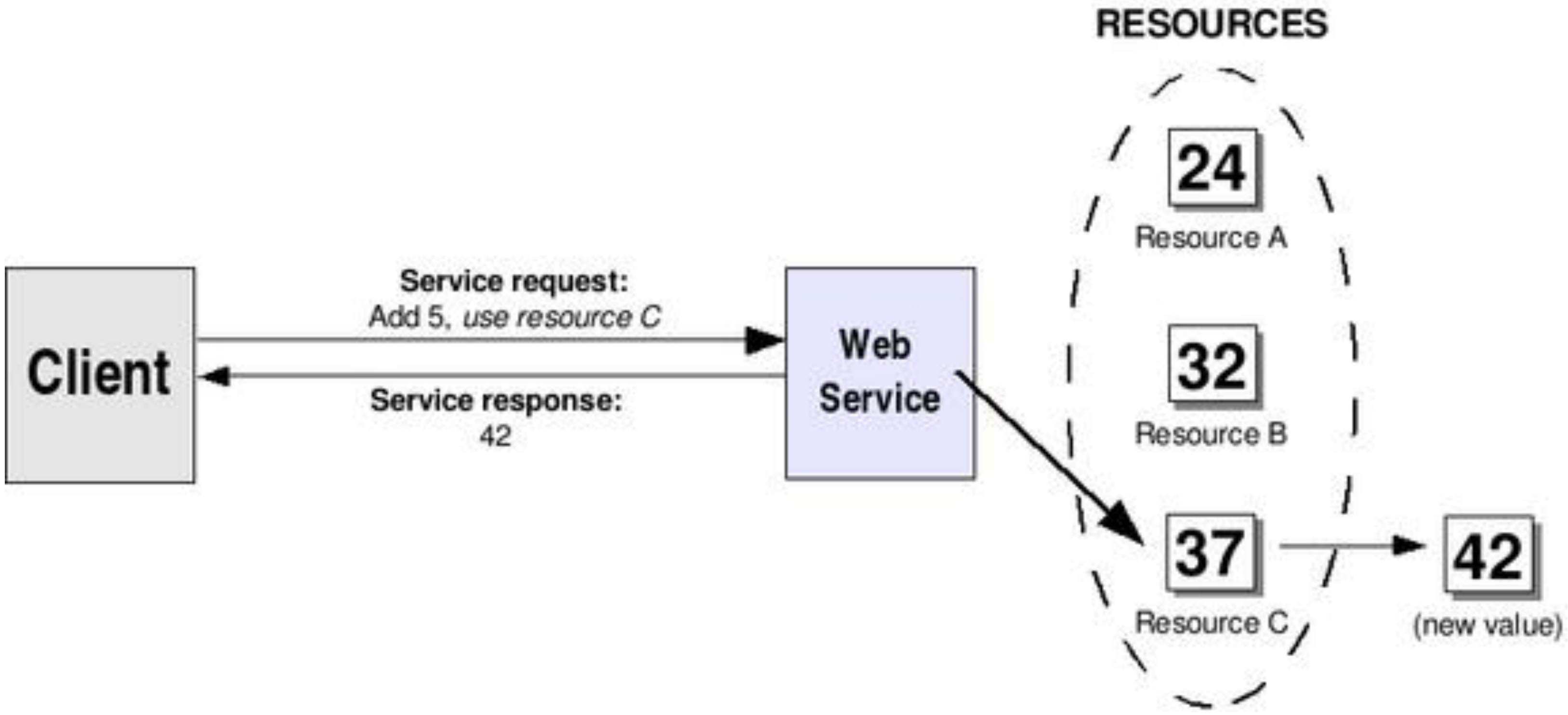


Web services and the Grid

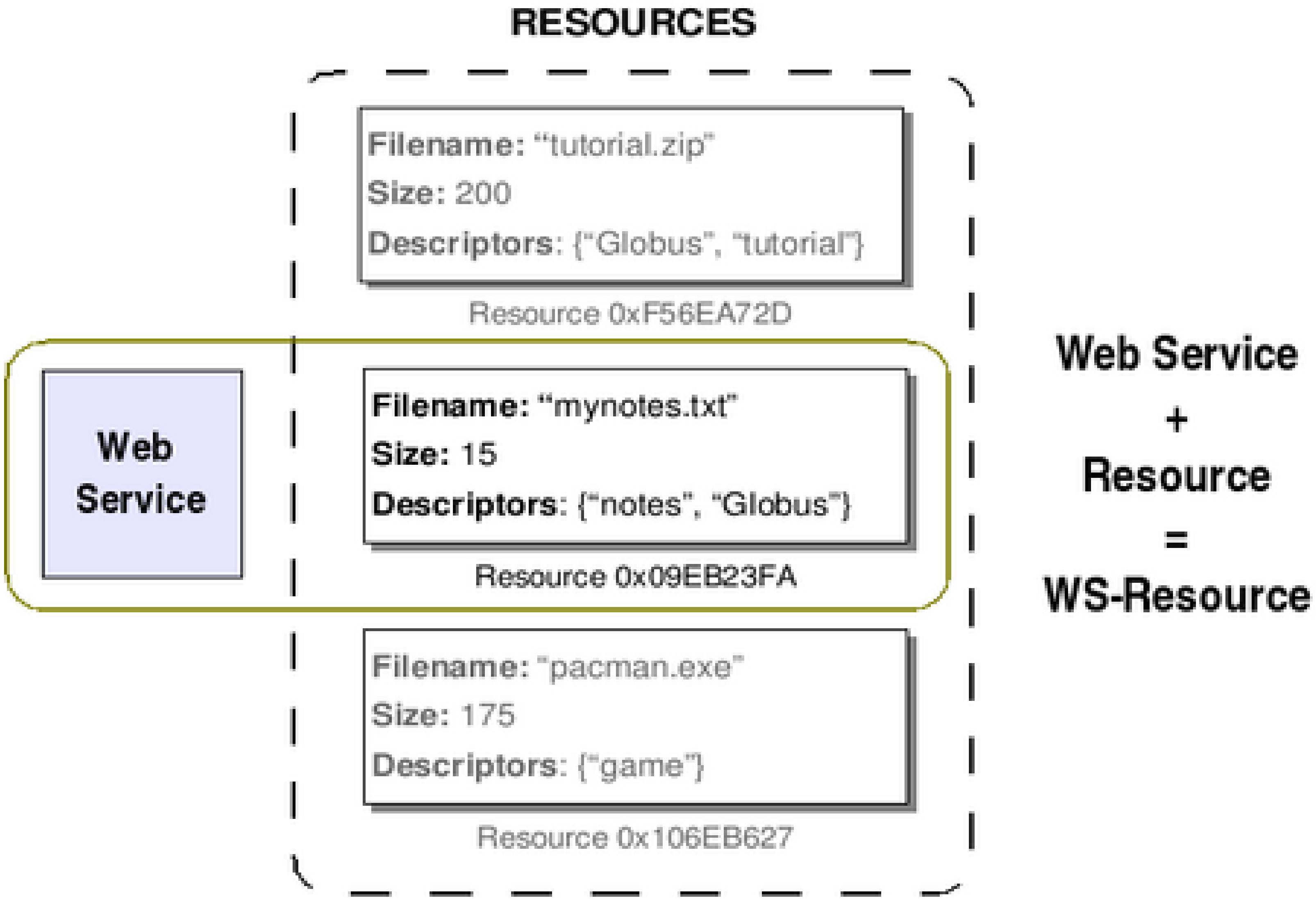
- GT4 replaced OGSI by WSRF (Web Service Resource Framework)
- Framework developed as a joint effort of WS and Grid groups
- GWSDL foi abandonada

How to model states using WS

- A resource is associated to each web service



WS-Resource



Filename: "tutorial.zip"
Size: 200
Descriptors: {"Globus", "tutorial"}

Resource 0xF56EA72D

Filename: "mynotes.txt"
Size: 15
Descriptors: {"notes", "Globus"}

Resource 0x09EB23FA

Filename: "pacman.exe"
Size: 175
Descriptors: {"game"}

Resource 0x106EB627

How to access a WS-Resource

- URI used to access the web service
- WS-Addressing used to access WS-Resource
- The address of a particular WS-Resource is called an *endpoint reference* in WS-Addressing lingo

WSRF

- a specification developed by OASIS http://www.oasis-open.org
- WSRF specifies how one can make Web Services stateful
- Differences between OGSI and WSRF:
  - – http://www.globus.org/wsrf/specs/ogsi_to_wsrf_1.0.pdf

WSRF

- 5 normative WSRF specifications:
  - WS-ResourceProperties
  - WS-ResourceLifetime
  - WS-RenewableReferences
  - WS-ServiceGroup
  - WS-BaseFault
  - WS-Notification family of specifications

WSRF

- WS-ResourceProperties: properties of resources. For example, a resource can have values of different types (properties)

WSRF

- WS-ResourceLifetime: a WS-Resource can be destroyed, either synchronously with respect to a destroy request or through a mechanism offering time-based (scheduled) destruction, and specified resource properties [WS-ResourceProperties] may be used to inspect and monitor the lifetime of a WS-Resource

WSRF

- WS-RenewableReferences: a Web service endpoint reference (WS-Addressing) can be renewed in the event the addressing or policy information contained within it becomes invalid or stale

WSRF

- WS-ServiceGroup: heterogeneous by-reference collections of Web services can be defined, whether or not the services are WS-Resources (for example, one can dynamically add a new resource to a group of resources)

WSRF

- WS-BaseFault: fault reporting can be made more standardized through the use of an XML Schema type for base faults and rules for how this base fault type is used and extended by Web services

WSRF

- WS-Notification family of specifications: Standard approaches to notification of changes

WSDL

- **Types**– a container for data type definitions using some type system (such as XSD).
- **Message**– an abstract, typed definition of the data being communicated.
- **Operation**– an abstract description of an action supported by the service.
- **Port Type**–an abstract set of operations supported by one or more endpoints.
- **Binding**– a concrete protocol and data format specification for a particular port type.
- **Port**– a single endpoint defined as a combination of a binding and a network address.
- **Service**– a collection of related endpoints.

Creation of a stateful web service

- MathService to perform operations:
  - Addition
  - Subtraction
- Have the ResourceProperties (RP):
  - Value (integer)
  - Last operation performed (string)
- Extra operation Get to get Value RP
- Once a new resource is created:
  - Value is set to zero
  - Last operation is set to “NONE”

5 steps

- **Define the service's interface**. This is done with *WSDL*
- **Implement the service**. This is done with *Java*.
- **Define the deployment parameters**. This is done with *WSDD* and *JNDI*
- **Compile everything and generate a GAR file**. This is done with *Ant*
- **Deploy service**. This is also done with *a GT4 tool*

WSDD: Web Service Deployment Descriptor

JNDI: Java Naming and Directory Interface

Step 1:

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions name="MathService"
    targetNamespace="http://www.globus.org/namespaces/examples/core/
    MathService_instance"
    xmlns ="http://schemas.xmlsoap.org/wsdl/"
    xmlns:tns="http://www.globus.org/namespaces/examples/core/MathService_
    instance"
    xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
    xmlns:wsrp="http://docs.oasis-open.org/wsrf/2004/06/wsrf-WS-
    ResourceProperties-1.2-draft-01.xsd" xmlns:wsrpw="http://docs.oasis-
    open.org/wsrf/2004/06/wsrf-WS-ResourceProperties-1.2-draft-01.wsdl"
    xmlns:wsdlpp="http://www.globus.org/namespaces/2004/10/WSDLPreproce
    ssor" xmlns:xsd="http://www.w3.org/2001/XMLSchema">
</definitions>
```

http://gdp.globus.org/gt4-tutorial/multiplehtml/apas01.html

More info

- http://www.globus.org/wsrf/specs/ogsi_to_wsrf_1.0.pdf
- http://docs.oasis-open.org/wsrf/wsrf-primer-1.2-primer-cd-02.pdf
- http://www.globus.org/wsrf/specs/ws-wsrf.pdf