Applications of SOA and Web Services in Grid Computing

Thái Duy Cường – 09070425
Nguyễn Văn Long – 09070450
GVHD: TS.Phạm Trần Vũ

Outline

- SOA – Service-Oriented Architecture
- Web Services
- Open Grid Services Architecture (OGSA)
- Web Service Resource Framework (WSRF)

Service-Oriented Architecture

- Architecture
- Service
- SOA

Architecture

- A formal description of a system, defining its purpose, functions, externally visible properties, and interfaces. It also includes the description of the system's internal components and their relationships, along with the principles governing its design, operation, and evolution.

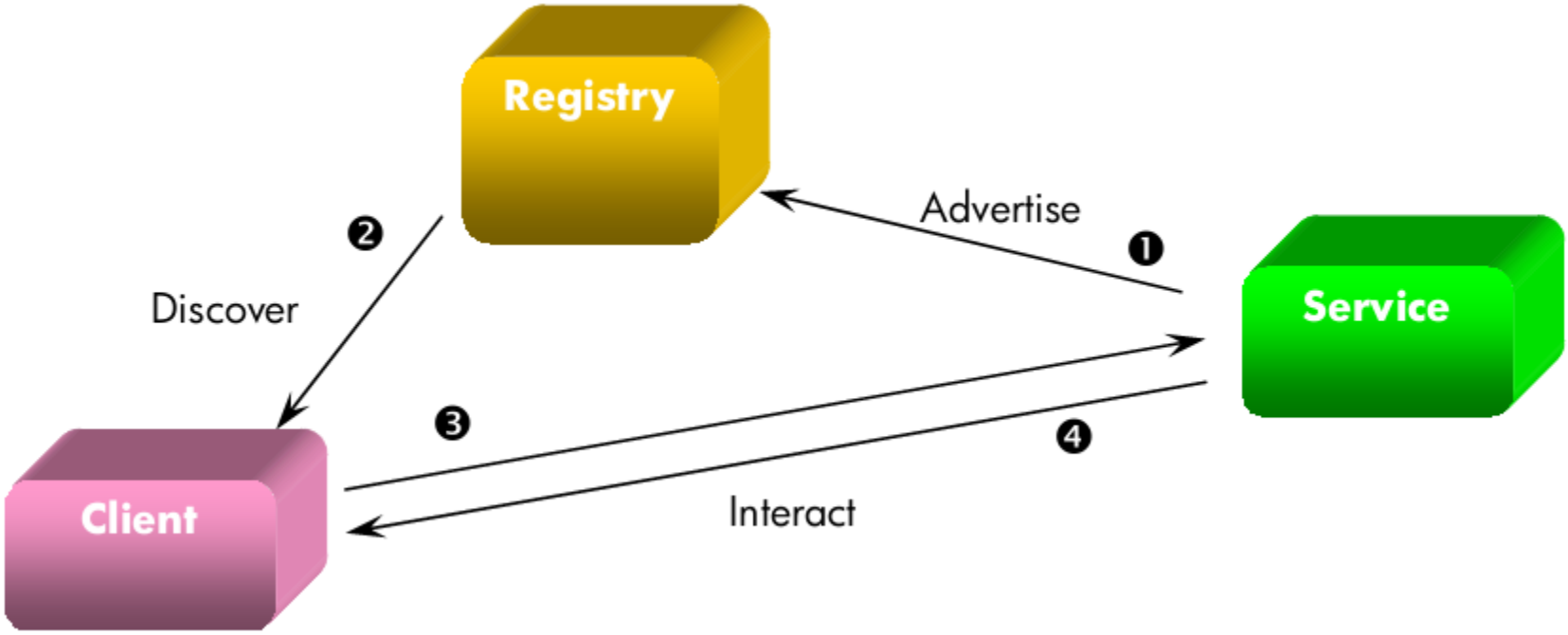
Service

- A software component that can be accessed via a network to provide functionality to a service requester.

Service's Characteristics

- Services may be individually useful, or can be integrated to provide higher-level services.
- Services communicate with their clients by exchanging messages
- Services can participate in a workflow
- Services may be completely self-contained, or they may depend on the availability of other services, or on the existence of a resource such as a data base.
- Services advertise details
- Implementation details are of no concern to clients, and are not revealed

Service Interaction



Service-Oriented Architecture

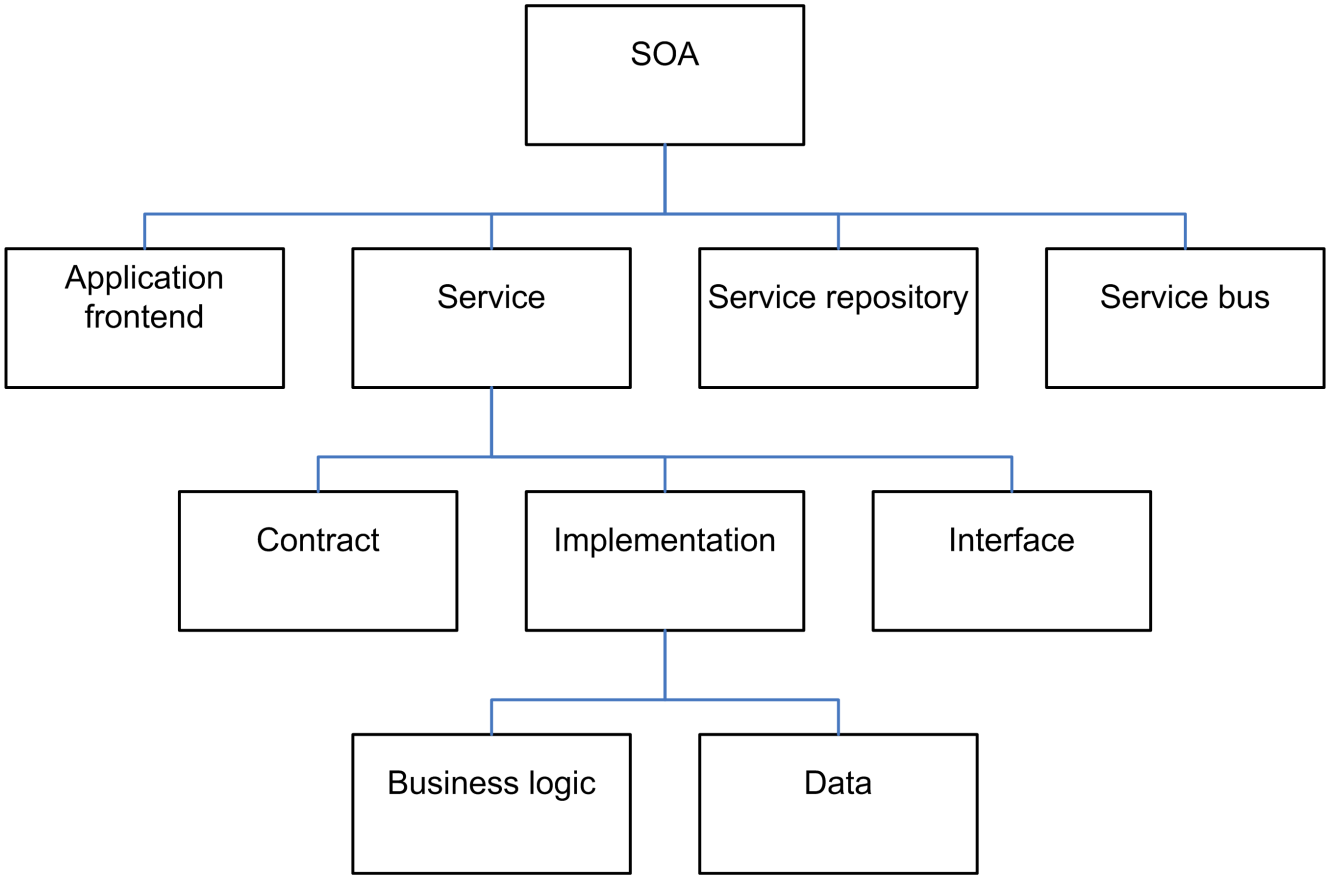
- In software engineering, a SOA is a set of principles and methodologies for designing and developing software in the form of interoperable services.

Service-Oriented Architecture

- OASIS: A paradigm for organizing and utilizing distributed capabilities that may be under the control of different ownership domains. It provides a uniform means to offer, discover, interact with and use capabilities to produce desired effects consistent with measurable preconditions and expectations.
- Thomas Erl: SOA represents an open, agile, extensible, federated, composable architecture comprised of autonomous, QoS-capable, vendor diverse, interoperable, discoverable, and potentially reusable services, implemented as Web services.

Service-Oriented Architecture

- SOA Elements



Service-Oriented Architecture

- Application frontends: are active elements of the SOA, delivering the value of SOA to the end users.
  - They initiate and control all activity of the enterprise system.
  - Web application, application with GUI, or a batch application.
- Service: a software component that encapsulates a high level business concept.
- Contract: provides a specification of the purpose, functionality, constraints, and usage of services.
- Interface: functionality of the service exposed by the service to the clients that are connected to the service.

Service-Oriented Architecture

- Implementation: provides the required business logic and appropriate data. It contains one or more of the artifacts: programs, configuration, data and databases.
- Business logic: business process represented by the service.
- Data: data represented in the service/used by the service.
- Service repository: it registers the services and their attributes to facilitate the discovery of services; operation, access rights, owner, qualities, etc.
- Service Bus (ESB): A flexible infrastructure for integrating applications and services by : routing messages, transforming protocols between requestor and service, handling business events and delivering them, providing QoS, mediation and security, and managing the interaction among services.

Service-Oriented Architecture

- Benefits
  - Loose Coupling
    - Flexibility
    - Scalability
    - Replaceability
    - Fault tolerance
  - Distributed
    - Manage load
    - Failover for reliability
    - May be geographically distributed
  - Asset Management
    - Leverages existing resources
    - Creates assets
    - Separate teams

Service-Oriented Architecture

- Disadvantage
  - Overhead: using XML – easy to parse, but larger size
  - Reliability:
    - Multiple points of failure: machine, router, link, process
    - Multiple points of infrastructure failure: power, physical security, environment (storm, flood, earthquake)
  - Security
  - Programming Complexity : Programming SOA is easy, but doing it well is hard.
  - Configuration Management
  - Governance

Web Services

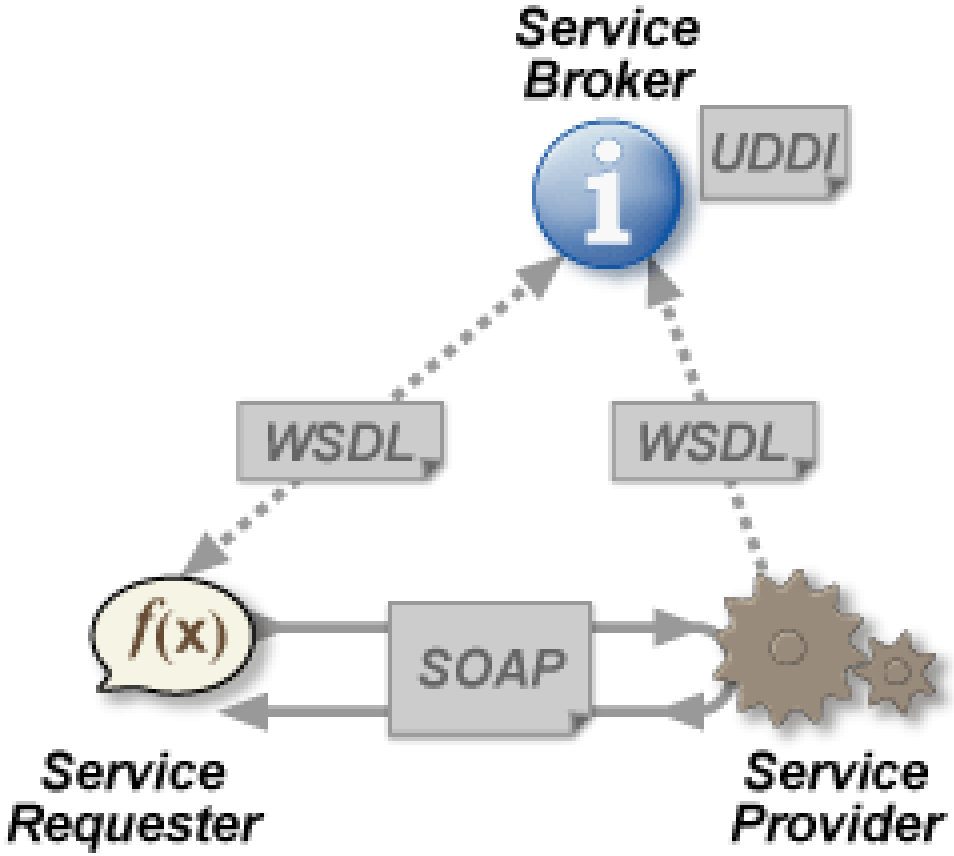
- Definitions
- XML
- SOAP
- WSDL

Web Services

- Definitions
  - Wiki: a method of communication between two electronic devices over the web
  - W3C: a software system designed to support interoperable machine-to-machine interaction over a network
  - A standardized way of integrating Web-based applications using the XML, SOAP, WSDL and UDDI open standards over an Internet protocol backbone

Web Services

- Web services architecture



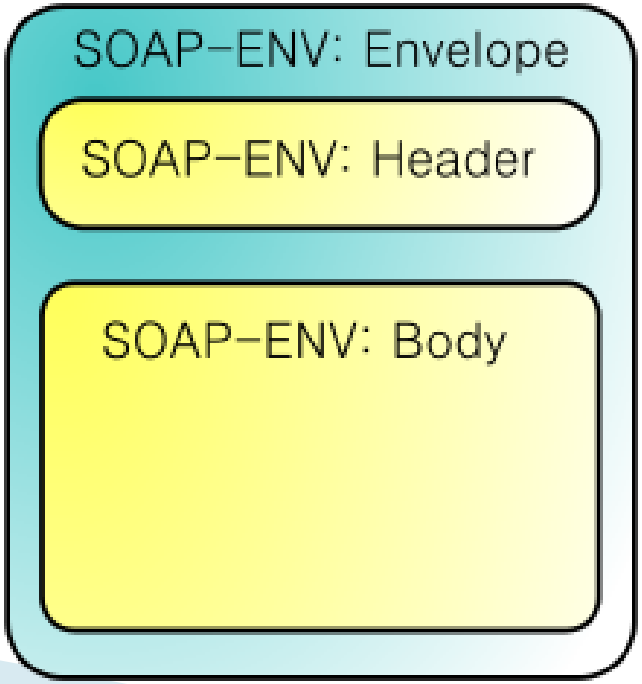
Web Services

- XML – eXtensible Markup Language
  - a markup language for formatting and exchanging structured data
  - It allows designers to create their own customized tags, enabling the definition, transmission, validation, and interpretation of data between applications and between organizations.

```
<note>
    <to>Tove</to>
    <from>Jani</from>
    <heading>Reminder</heading>
    <body>Don't forget me this weekend!</body>
</note>
```

Web Services

- SOAP – Simple Object Access Protocol
  - an XML-based protocol for specifying envelope information, contents and processing information for a message



Web Services

- SOAP – Simple Object Access Protocol
  - A SOAP Request

```
POST /InStock HTTP/1.1
Host: www.example.org
Content-Type: application/soap+xml; charset=utf-8
<?xml version="1.0"?>
<soap:Envelope>

<soap:Body xmlns:m="http://www.example.org/stock">
  <m:GetStockPrice>
    <m:StockName>IBM</m:StockName>
  </m:GetStockPrice>
</soap:Body>

</soap:Envelope>
```

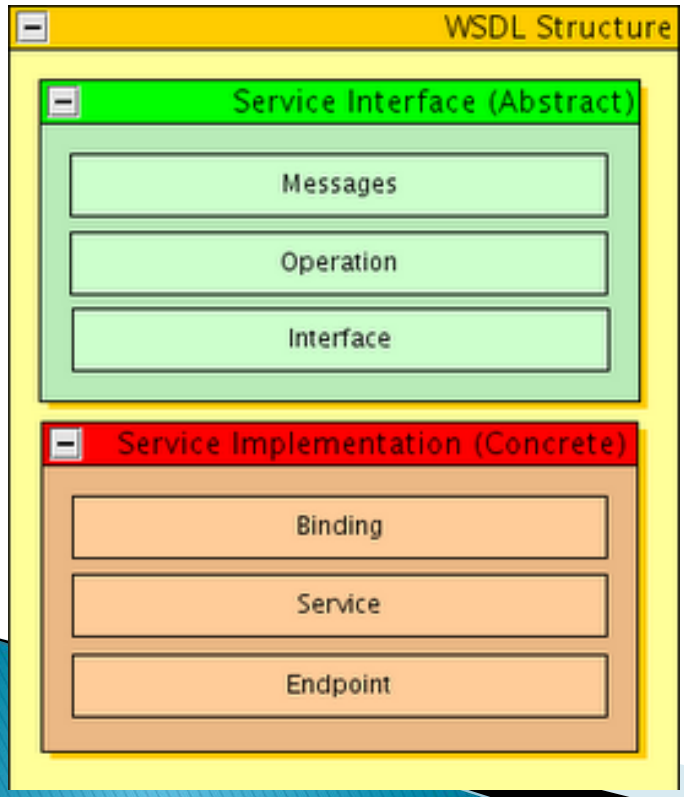
Web Services

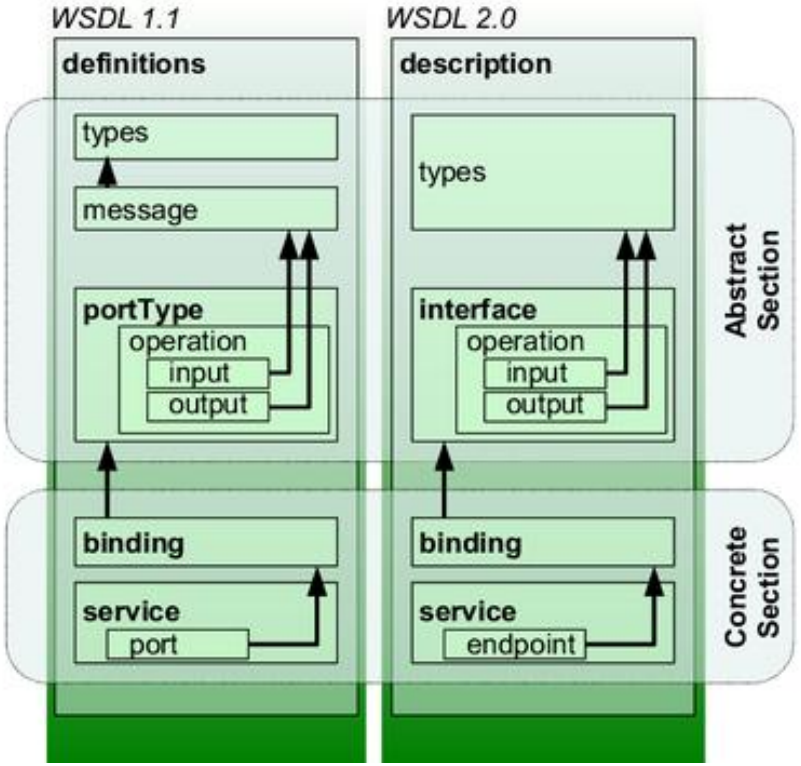
- SOAP – Simple Object Access Protocol
  - A SOAP Response

```
HTTP/1.1 200 OK
Content-Type: application/soap+xml; charset=utf-8

<?xml version="1.0"?>
<soap:Envelope>

<soap:Body xmlns:m="http://www.example.org/stock">
  <m:GetStockPriceResponse>
    <m:Price>34.5</m:Price>
  </m:GetStockPriceResponse>
</soap:Body>

</soap:Envelope>
```

Web Services

- WSDL – Web Services Description Language
  - an XML-based language that is used for describing the functionality offered by a Web service





Web Services

- UDDI – Universal Description, Discovery and Integration
  - a directory service where businesses can register and search for Web services
  - a platform-independent framework for describing services, discovering businesses, and integrating business services by using the Internet

Web Services

- UDDI – Universal Description, Discovery and Integration

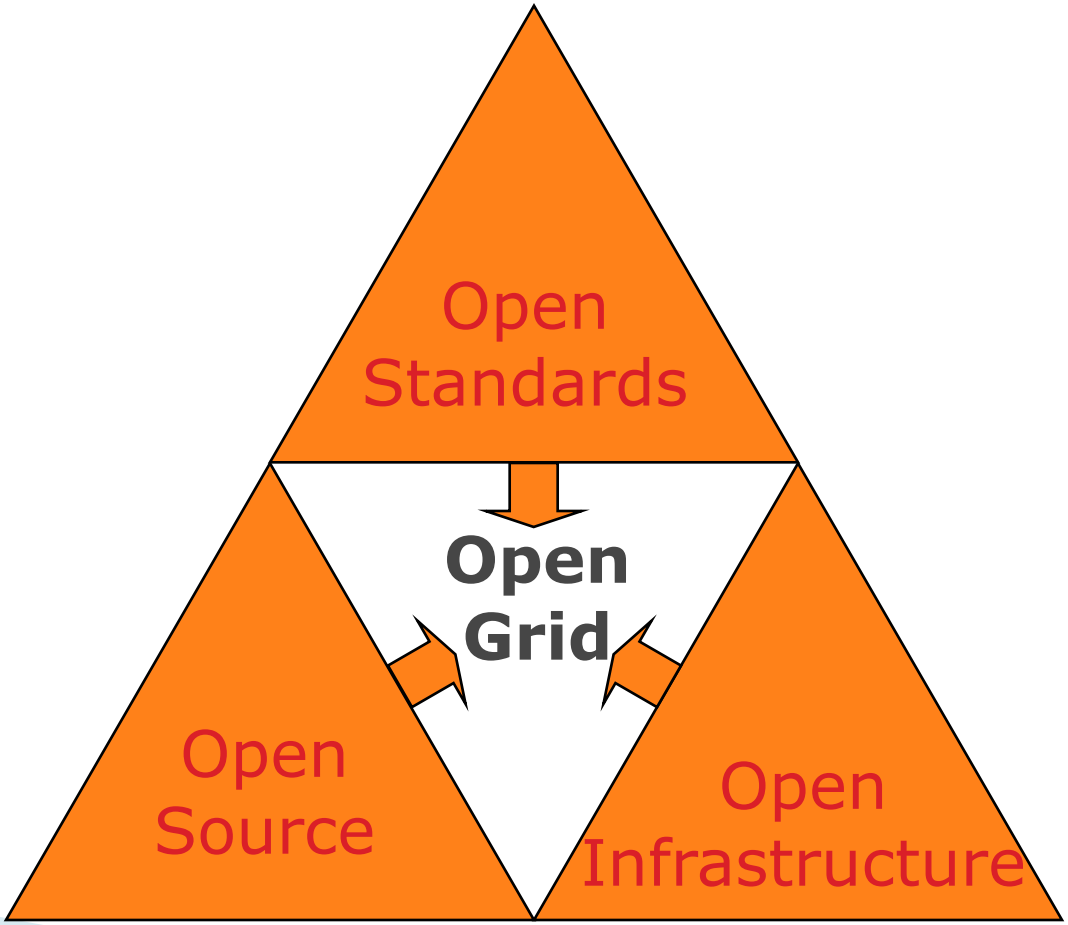
SOA and Web Services: Style vs. Implementation

- Service-orientation is an architectural style
- Web services are an implementation technology
- The two can be used together, and they frequently are, but they are not mutually dependent

Open Grid Services Architecture

- OGSA have introduced by Globus and IBM, 2002
- The Open Grid Services Architecture (OGSA) represents an evolution towards a Grid system architecture based on Web services concepts and technologies.

Building an Open Grid



Grids and Open Standards

- Open Grid Service Architecture
  - everything is represented as a service
  - defines what is a Grid Service
    - definition of standard service interfaces
    - identification of the protocol(s)

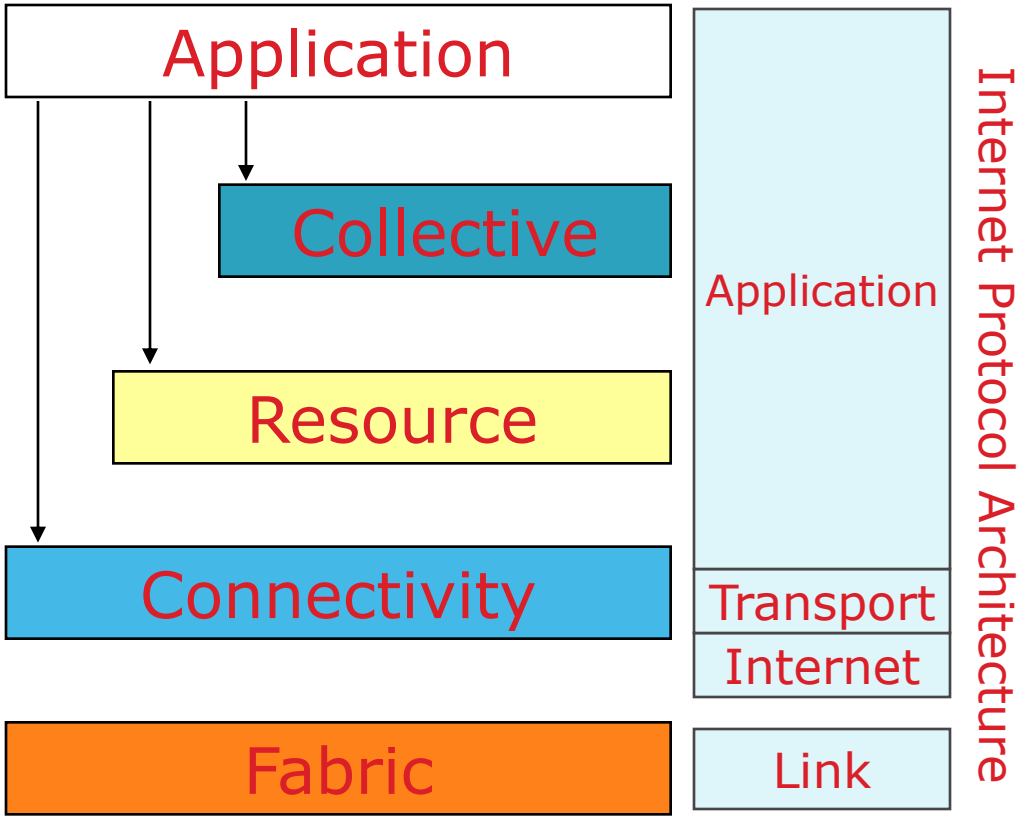
Layered Grid Architecture

"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



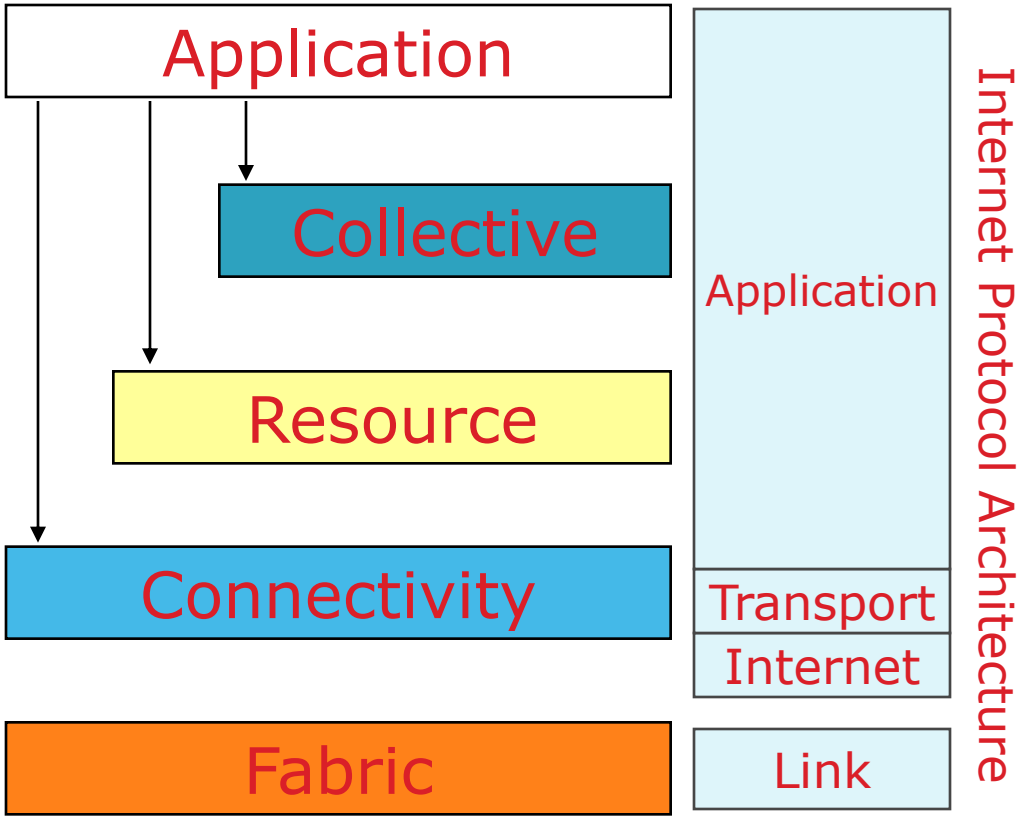
Layered Grid Architecture

"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



From Resources to Services: Managing Virtual Services

- Trying to manage total system properties
  - E.g. Dependability, end-to-end QoS
- "Resource" tends to connote a tangible entity to be consumed: CPU, storage, bandwidth, …
- But many interesting services may be decoupled from any particular resource
  - E.g. virtual data service, data analysis service
  - A service consumes resources, but how that happens is irrelevant to the client
- "Service" forms a better base abstraction
  - Can apply to physical or virtual

Open Grid Services Architecture

- Service-oriented architecture
  - Key to virtualization, discovery, composition, local-remote transparency
- Leverage industry standards
  - Internet, Web services
- Distributed service management
  - A “component model for Web services” (or: a “service model for the Grid”)
- A framework for the definition of composable, interoperable services

Web Services

- A simple but powerful distributed system paradigm, that allows one to:
  - Describe a service (WSDL)
  - Invoke a service (SOAP)
  - Discover a service (various)
- Web services appears to offer a fighting chance at ubiquity (unlike CORBA)
  - Sophisticated tools emerging from industry
- But Web services does not go far enough to serve a common base for the Grid …

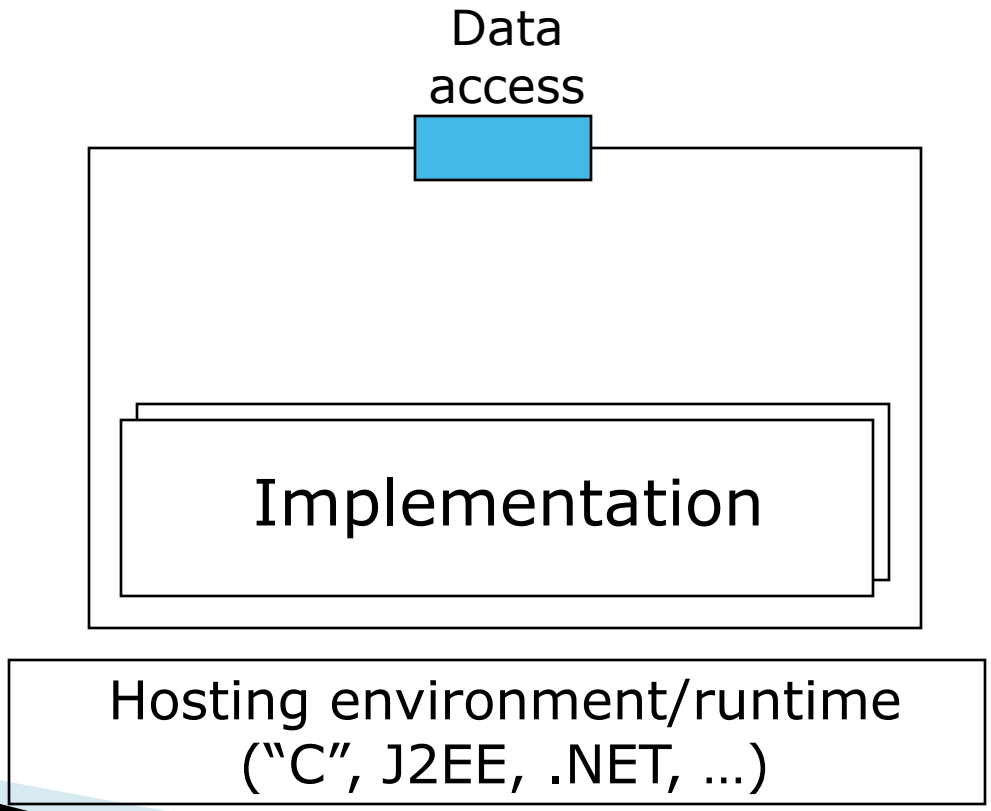
Transient Service Instances

- "Web services" address discovery & invocation of persistent services
  - Interface to persistent state of entire enterprise
- In Grids, must also support transient service instances, created/destroyed dynamically
  - Interfaces to the states of distributed activities
  - E.g. workflow, video conf., dist. data analysis
- Significant implications for how services are managed, named, discovered, and used
  - In fact, much of Grid is concerned with the management of service instances

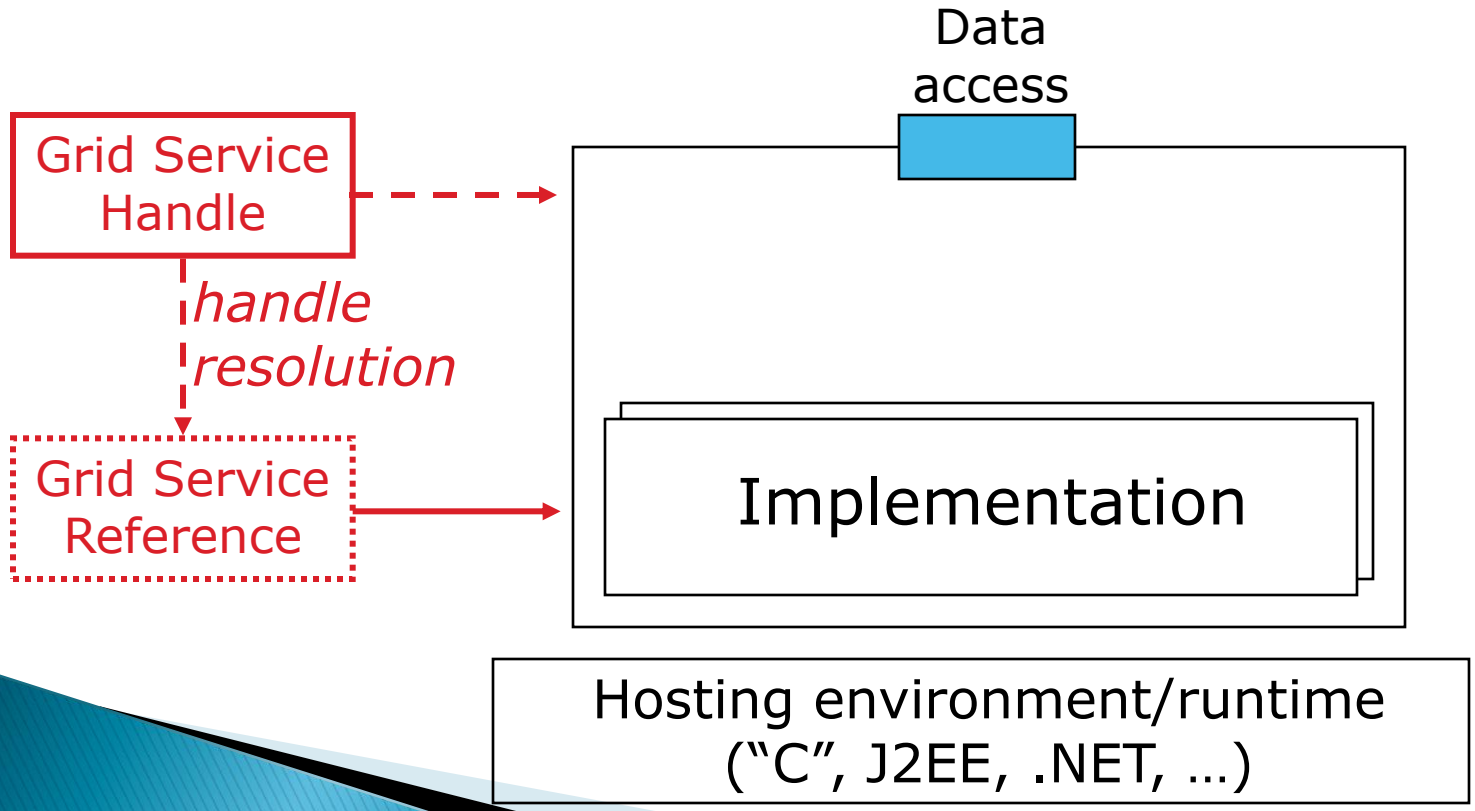
OGSA Structure

- A standard substrate: the Grid service
  - Standard interfaces and behaviors that address key distributed system issues
  - A refactoring and extension of the Globus Toolkit protocol suite
- … supports standard service specifications
  - Resource management, databases, workflow, security, diagnostics, etc., etc.
  - Target of current & planned GGF efforts
- … and arbitrary application-specific services based on these & other definitions

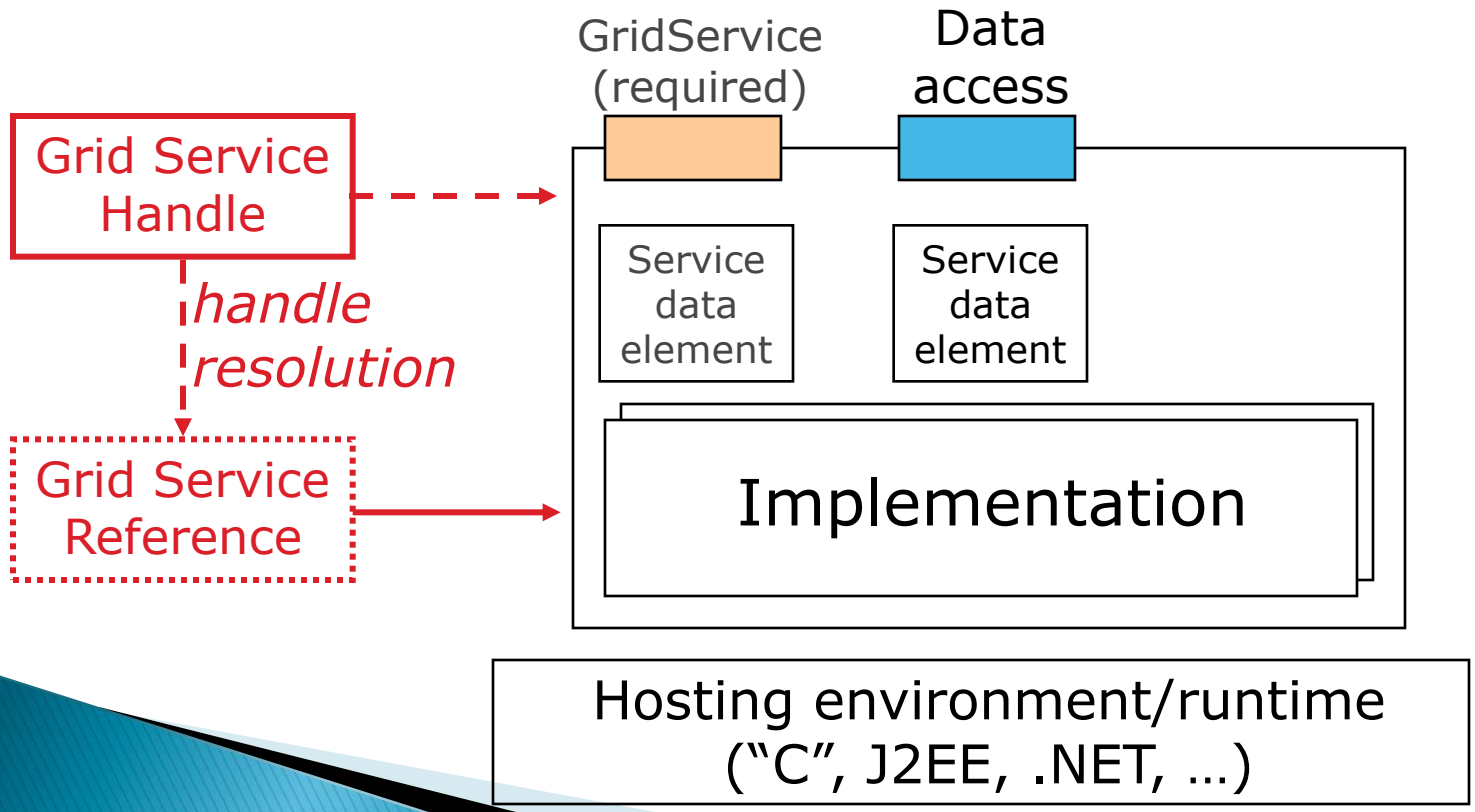
Open Grid Services Infrastructure



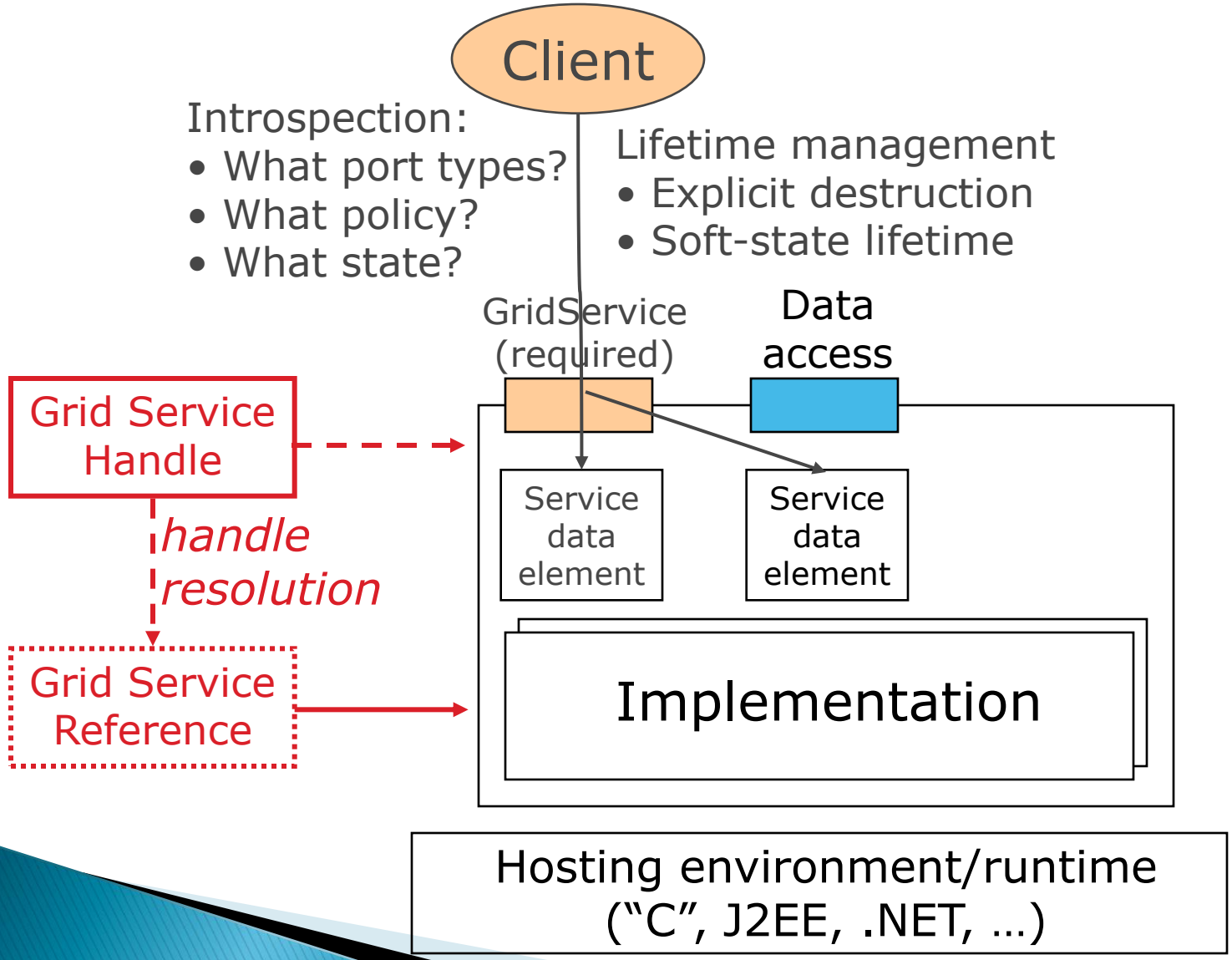
Open Grid Services Infrastructure



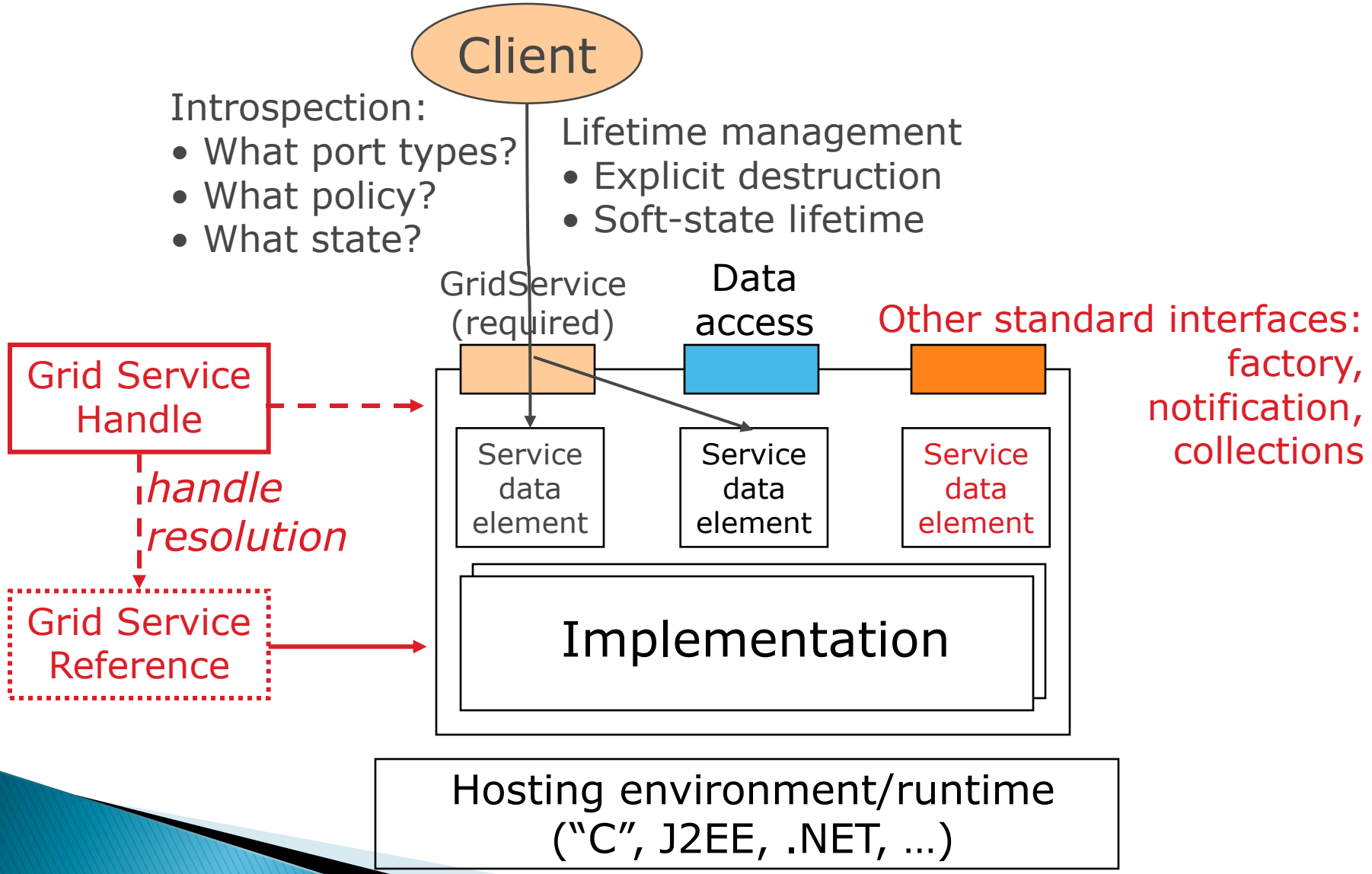
Open Grid Services Infrastructure



Open Grid Services Infrastructure

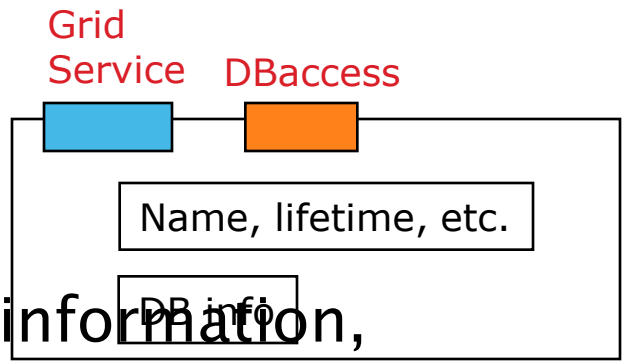


Open Grid Services Infrastructure



Grid Service Example: Database Service

- A DBaccess Grid service will support at least two portTypes
  - GridService
  - DBaccess
- Each has service data
  - GridService: basic introspection information, lifetime, …
  - DBaccess: database type, query languages supported, current load, …, …



Lifetime Management

- GS instances created by factory or manually; destroyed explicitly or via soft state
  - Negotiation of initial lifetime with a factory (=service supporting Factory interface)
- **GridService** interface supports
  - **Destroy** operation for explicit destruction
  - **SetTerminationTime** operation for keepalive
- Soft state lifetime management avoids
  - Explicit client teardown of complex state
  - Resource “leaks” in hosting environments

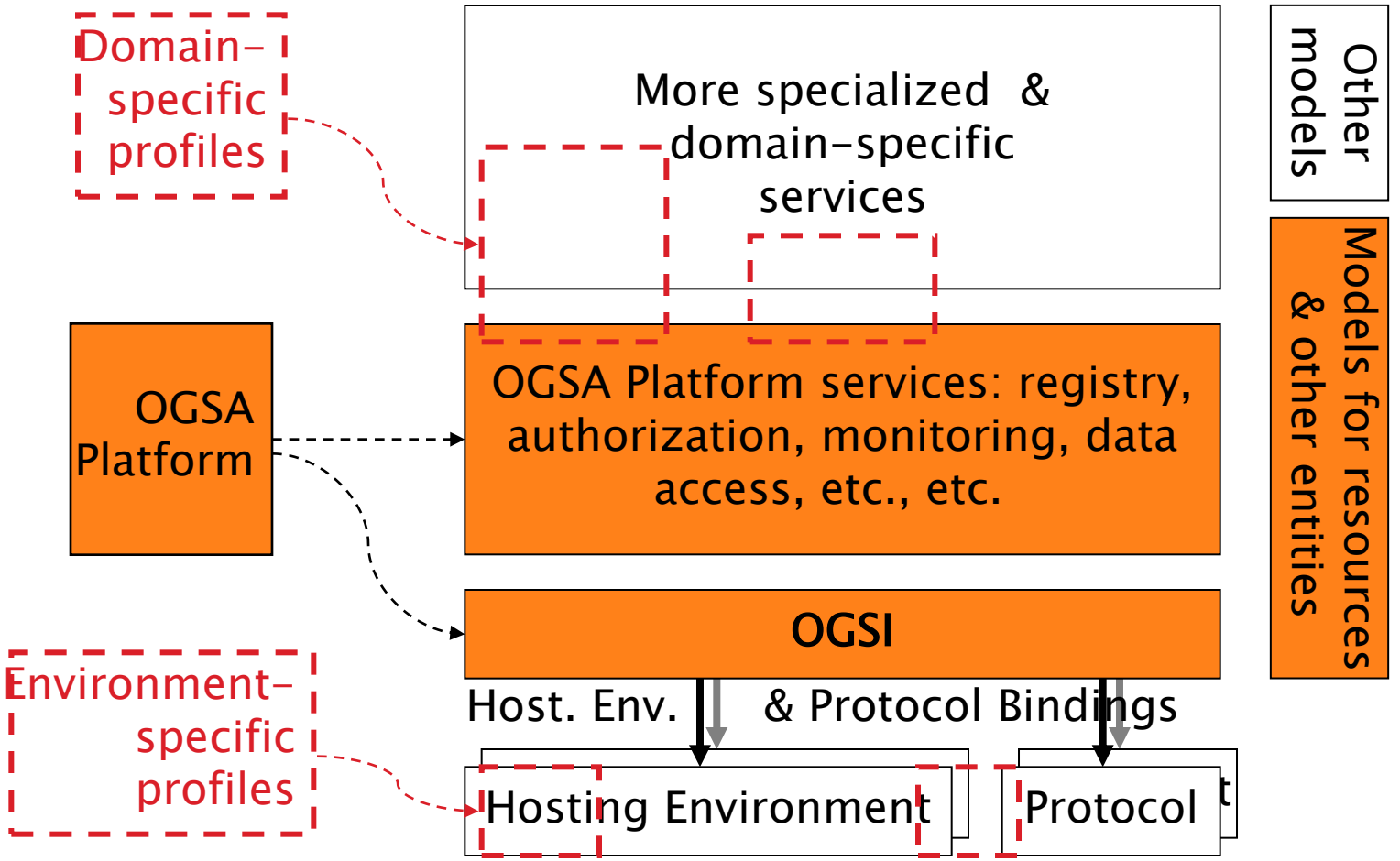
Factory

- **Factory** interface's **CreateService** operation creates a new Grid service instance
  - Reliable creation (once-and-only-once)
- **CreateService** operation can be extended to accept service-specific creation parameters
- Returns a **Grid Service Handle** (**GSH**)
  - A globally unique URL
  - Uniquely identifies the instance for all time
  - Based on name of a home handleMap service

Realizing a Service-Oriented Architecture: How Do I

- Create, name, manage, discover services?
- Render resources, data, sensors as services?
- Negotiate service level agreements?
- Express & negotiate policy?
- Organize & manage service collections?
- Establish identity, negotiate authentication?
- Manage VO membership & communication?
- Compose services efficiently?
- Achieve interoperability?

The OGSA Platform



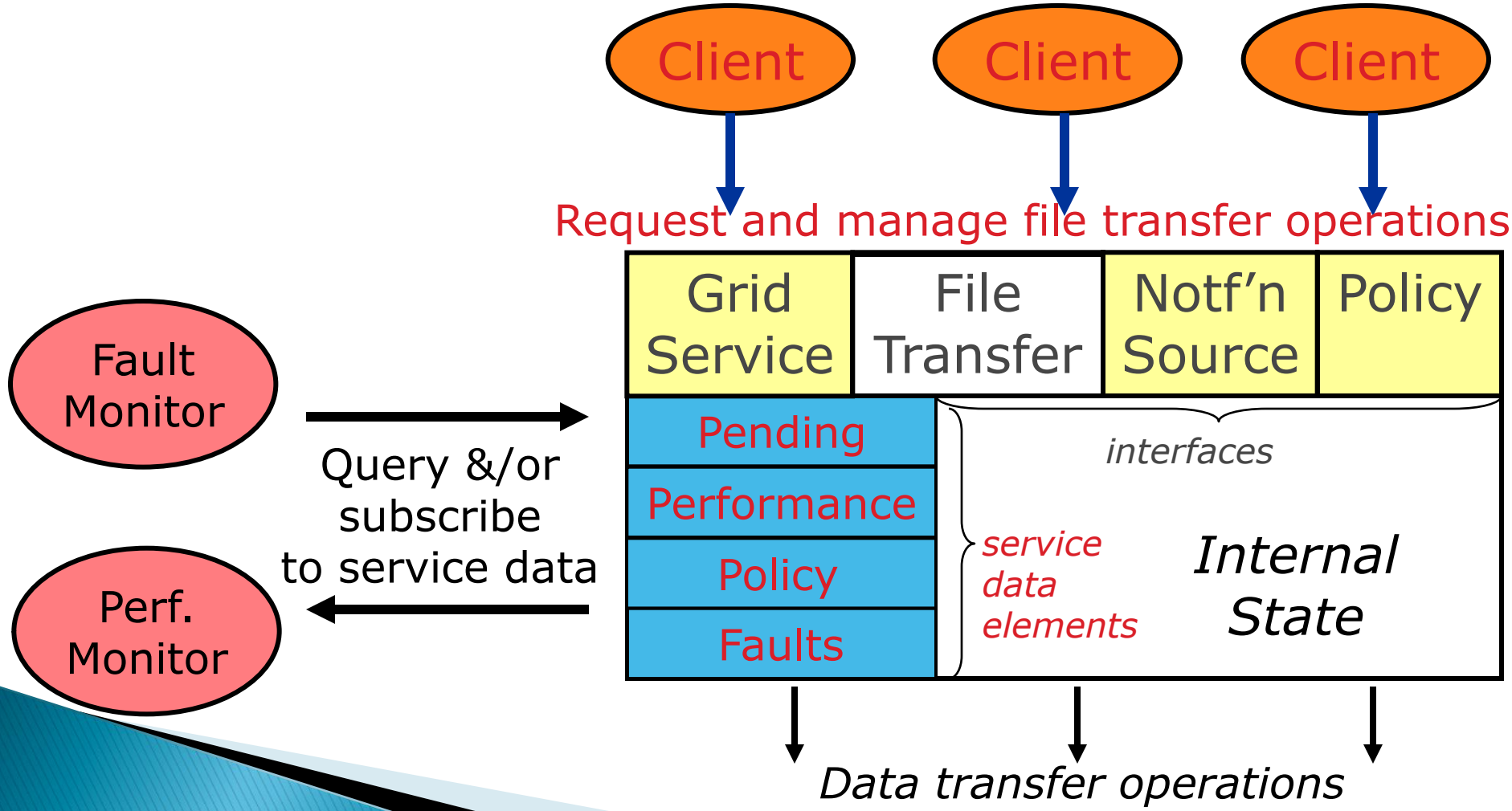
OGSA Definition Activities (Underway or Pending)

- Data Access and Integration
- Data Replication
- Security
- SLA Negotiation
- Common Management Model
- And others…

Open Grid Service Architecture: Next Steps

- ✓ Technical specifications
  - Open Grid Services Infrastructure is complete
  - Security, data access, Java binding, common management models, etc., in the pipeline
- ✓ Implementations and compliant products
  - OGSA-based Globus Toolkit v3, pyGlobus, …
  - IBM, Avaki, Platform, Sun, NEC, Oracle, …
- ⌛ Rich set of service defns & implementations
  - Time to start on OGSI-compliant services!

Example: Reliable File Transfer Service



Globus Toolkit v3 (GT3) Open Source OGSA Technology

- Implement core OGSI interfaces
- Support primary GT2 interfaces
  - High degree of backward compatibility
- Multiple platforms & hosting environments
  - J2EE, Java, C, .NET, Python
- New services
  - SLA negotiation (GRAM-2), registry, replica location, community authorization, data, …
- Growing external contributions & adoption

GT Timeline

- GT 1.0: *1998*
  - GRAM, MDS
- GT 2.0: *2001*
  - GridFTP, packaging, reliability
- GT3 Technology Preview: *Apr-Dec 2002*
  - Tracking OGSI definition
- GT3.0 Alpha: *Jan 2003*
  - OGSI Base, GT2 functionality
- GT3.0 Production: *June 2003*
  - Tested, documented, etc.

Globus Toolkit Contributors: GT2

- Grid Packaging Technology (GPT) NCSA
- Persistent GRAM Jobmanager Condor
- GSI/Kerberos interchangeability Sandia
- Documentation NASA, NCSA
- Ports IBM, HP, Sun, SDSC, …
- MDS stress testing EU DataGrid
- Support IBM, Platform, UK eScience
- Testing and patches Many!
- Interoperable tools Many!
- $$ DARPA, DOE, NSF, NASA, Microsoft, EU

Globus Toolkit Contributors: GT3

Replica location service	EU DataGrid
Python hosting environment	LBNL
Data access & integration	UK eScience
Data mediation services	SDSC
Tooling, Xindice, JMS	IBM
…	
…	
…	

OGSA Misconceptions

- OGSA means you have to code in Java
  - No: C client bindings now, C server side eventually (but not needed for current apps)
- OGSA means all programs must be services
  - No: You can write services if you want, but GT2-style GRAM behavior is still supported (GRAM is just a server)
- OGSA is a silver bullet for Grid computing
  - No, it makes some things easier, but it's only interfaces and behaviors, after all!

Summary

- OGSA: standards-based Grid technology
  - From Web services: standard IDL, discovery, binding independence, other desirable features
  - From Grid: naming, state, lifetime management, etc., etc.
- Rapid progress on definition & implementation
  - OGSI is defined, GT3 implements it (and other things), multiple groups coding to it
  - Much more happening, much more to be done!
- No silver bullet, but a good incremental step forward to our ultimate Grid software goals

Introduction to WSRF

- Introduction to WSRF
- What it is, why it was developed
- Relations to OGSI, OGSA
- Definitions

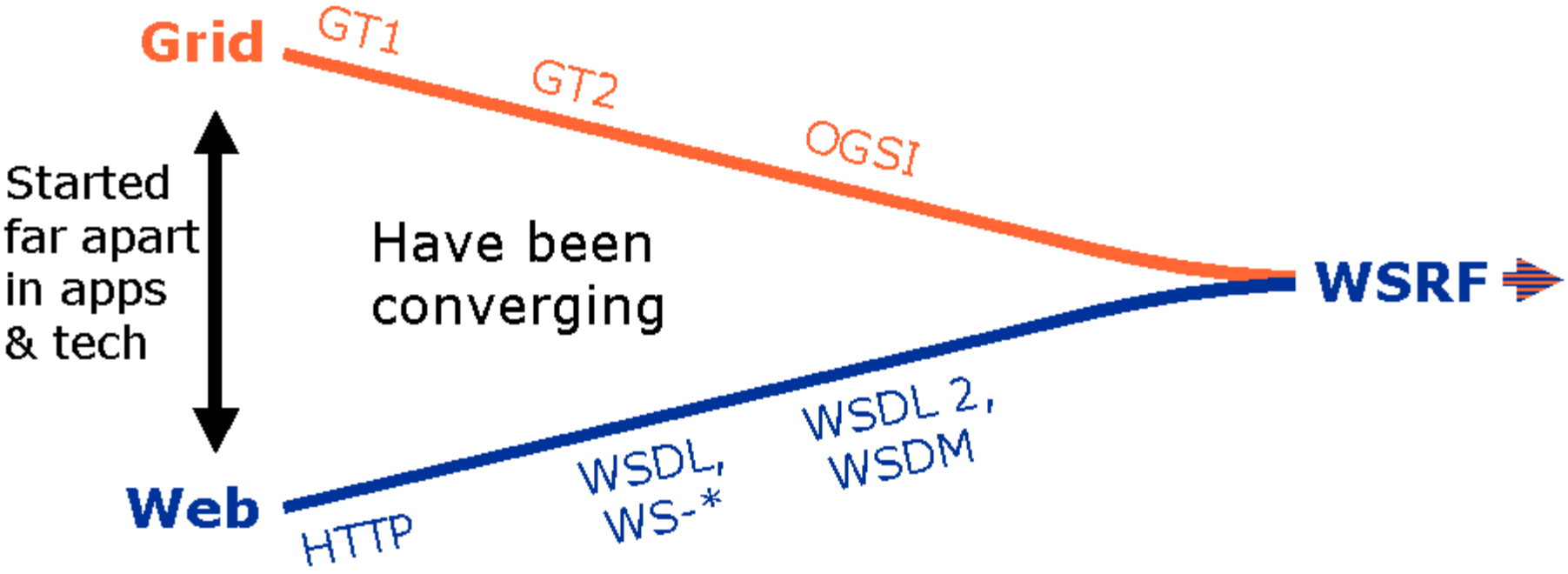
What it is?

- announced at GlobusWorld 04 by the Globus Alliance, IBM and HP

- WSRF is a set of five Web services specifications to model and manage state in a Web services context
  - ResourceLifetime
  - ResourceProperties
  - BaseFaults
  - RenewableReferences
  - ServiceGroup

  ... which together with the Notification Spec retain all of the essential functional capabilities present in OGSI

Why it was developed?



WSRF effectively completes the convergence of the Web service and Grid computing communities

Why it was developed?

- Criticisms of OGSI from the Web services community:
  - **Too much stuff in one spec**
    => functionality partitioned into a family of composable specifications
  - **Does not work well with existing Web services tooling**
    => WSRF tones down the usage of XML Schema
  - **Too object oriented**: OGSI v1.0 models a stateful resource as a Web service that encapsulates the resource's state, with the identity and lifecycle of the service and resource state coupled
    => WSRF makes an explicit distinction between the "service" and the stateful entities acted upon by that service

Relation from WSRF to …

- OGSA: WSRF mechanisms will enable OGSA
- OGSI: WSRF restates OGSI concepts in WS terms

OGSI	WSRF
Grid Service Reference (GSR)	*WS-Addressing* Endpoint Reference
Grid Service Handle (GSH)	*WS-Addressing* Endpoint Reference
HandleResolver portType	**WS-RenewableReferences**
Service data elements (SDE)	**WS-ResourceProperties**
GridService lifetime managementt	**WS-ResourceLifeCycle**
Notification portTypes	**WS-Notification**
Factory portType	Treated as a pattern
ServiceGroup portTypes	**WS-ServiceGroup**
Base fault type	**WS-BaseFaults**

Definitions in WSRF

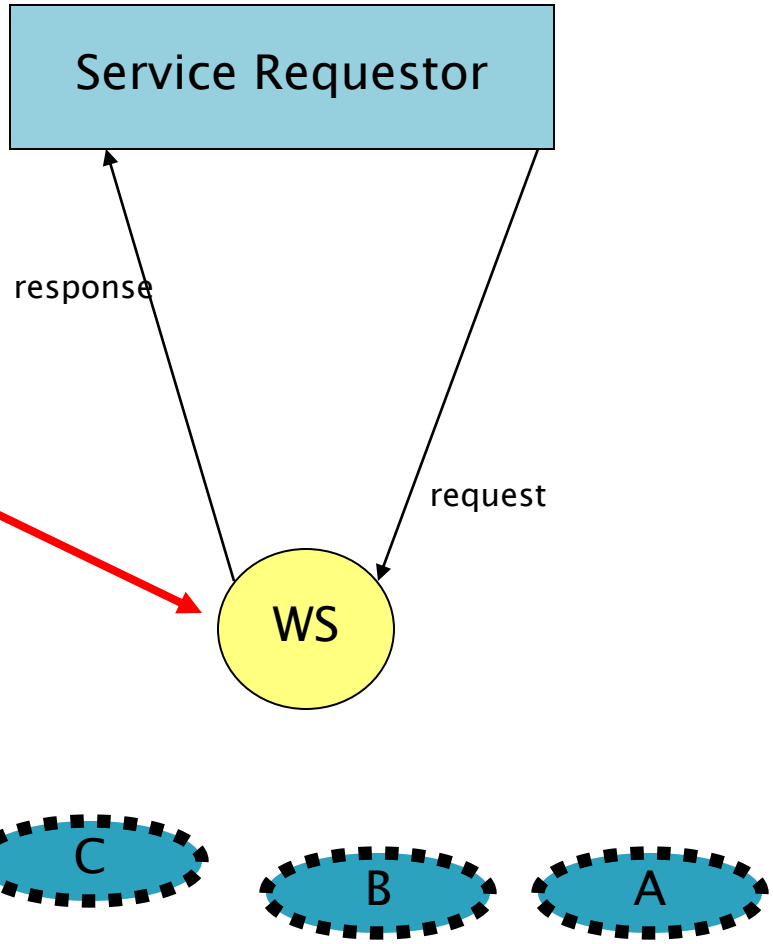
- **WS-Resource** = Web Service + stateful resource which is used in the execution of message exchanges
- **Stateful resource**:
  - Specific set of state data expressible as XML doc
  - Well defined lifecycle
  - Known to and acted upon by one or more web services
- **Implied resource pattern** = specific kind of relationship between web service and stateful resource
  - Stateful resource implicit input for the execution of the message request (static or dynamic)
  - Pattern means that relationship is codified by a set of conventions – in particular XML, WSDL and WS-Addressing

WSRF in detail

- WSRF Concepts in Detail
  - how WS-Addressing is used
  - have a closer look on the specs

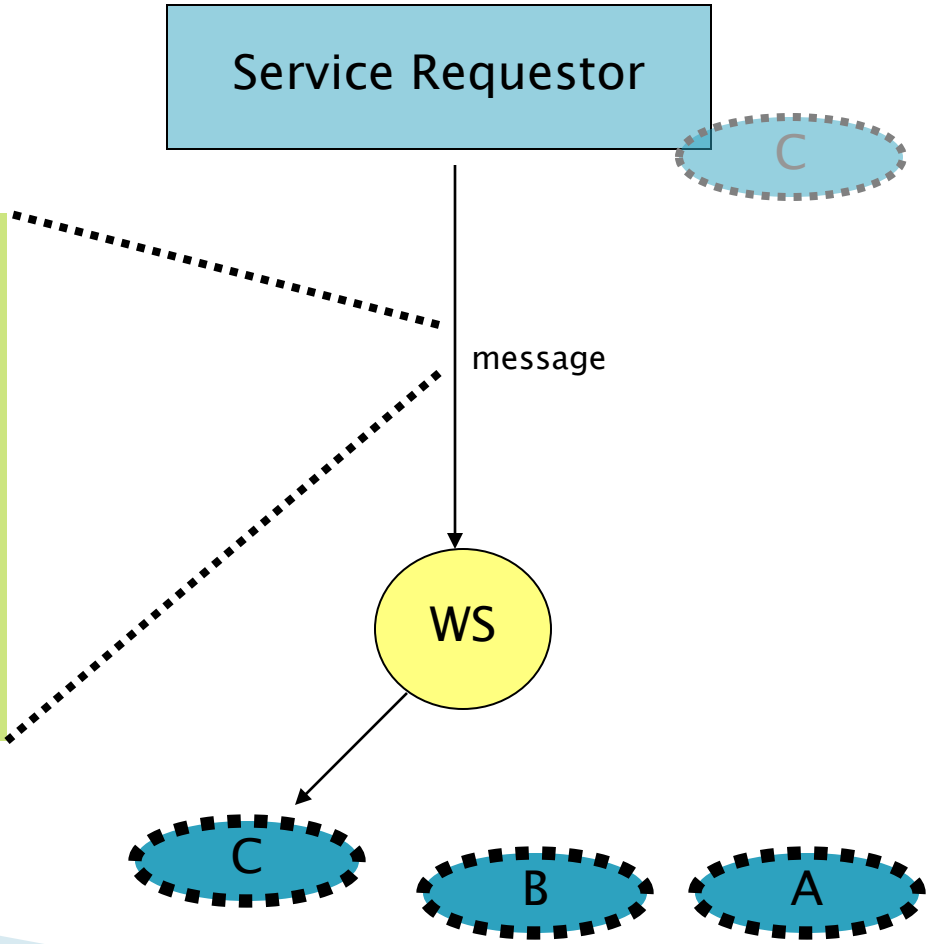
Usage of WS-Adressing I

```
<wsa:EndpointReference>
  <wsa:Address>
    http://someOrg.com/aWebService
  </wsa:Address>
  <wsa:ReferenceProperties>
    <tns:resourceID> C </tns:resourceID>
  </wsa:ReferenceProperties>
</wsa:EndpointReference>
```



Usage of WS-Adressing II

```
<soap:Envelope>
  <soap:Header>
    <tns:resourceID> C </tns:resourceID>
  </soap:Header>
  <soap:Body>
      … some message
  </soap:Body>
</soap:Envelope>
```



Resource-Lifecycle I

- The lifecycle of a WS-Resource is defined as the period between its instantiation and its destruction.
- Creation of a WS-Resource:
  - trough any Web service capable of bringing one or more WS-Resources into existence
  - response message typically contains at least one endpoint reference that refers to the new WS-Resource or places it into a registry for later retrival
  - a message exchange is only considered a WS-Resource factory operation if it results in the actual creation of the WS-Resource referred to in the returned WSResource-qualified endpoint reference

Resource-Lifecycle II

- immediate destruction

  request message: <wsrl:DestroyRequest />

  response message: <wsrl:DestroyResponse />

- scheduled destruction mechanisms uses properties of the WS-Resource to
  - query current time
  - Determine current termination time

Resource Lifecycle III

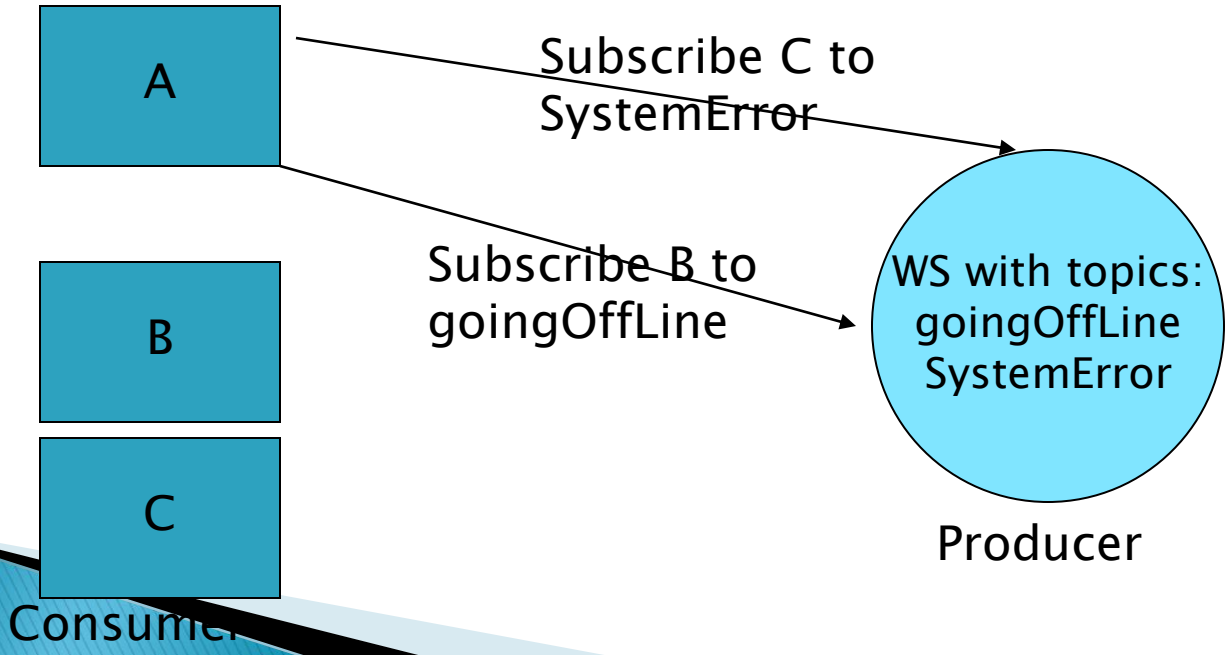
- Setting initial termination Time
  - via special XML element in the creation request message
- Requesting Change to Termination Time
  - SetTerminationTimeRequest message
- Notification of Resource Destruction
  - via subscription to topic ResourceTermination
- All time specifications are in UTC

Notifications I

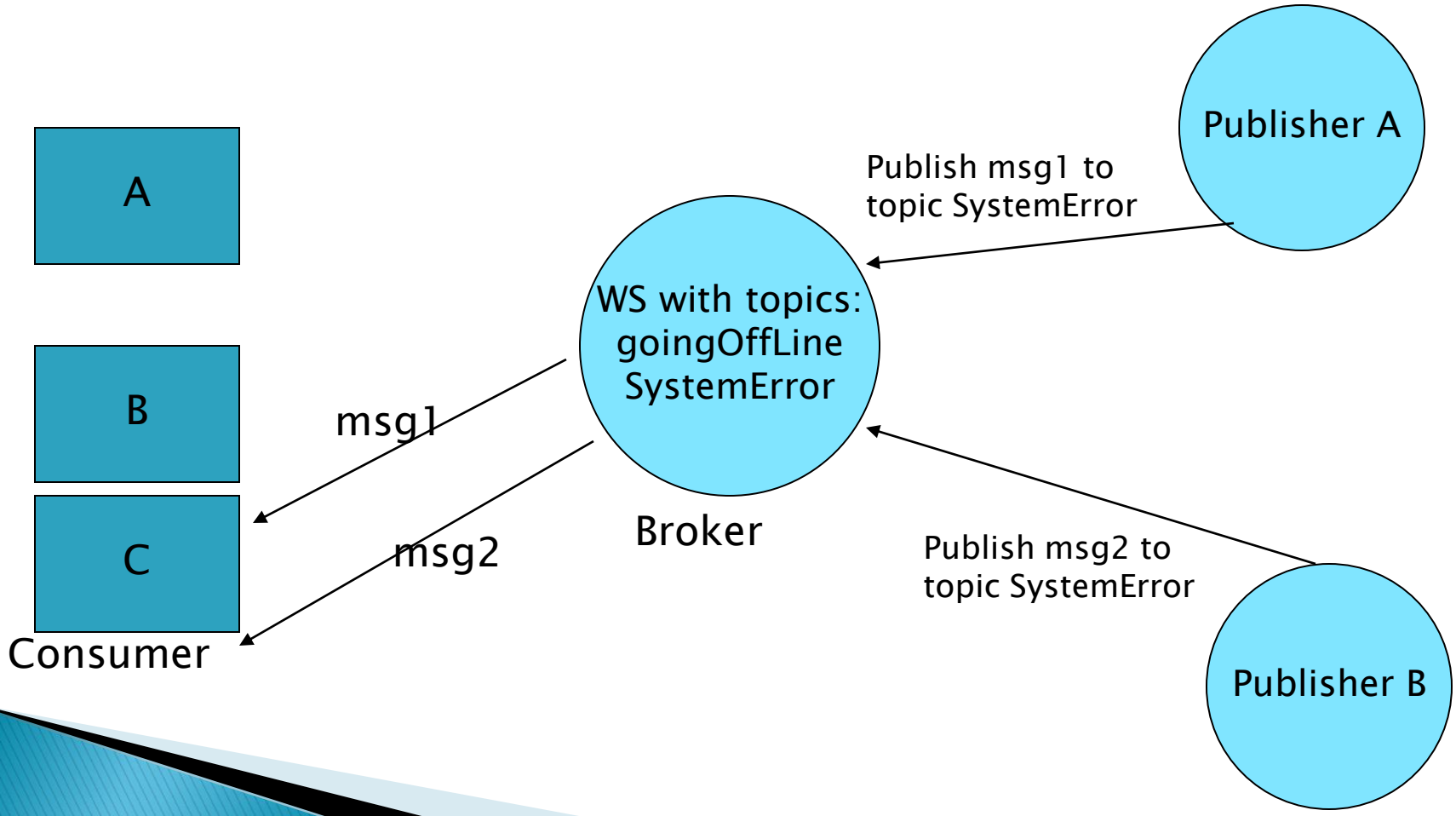
- notification using a topic-based publication/subscription pattern
- standard set of message exchanges that define the roles of NotificationProducer and NotificationConsumer
- standard way to name and describe Topics

Notifications II

- Topic = categorize Notifications and their related NotificationMessage schemas
  - part of the matching process



Notifications III



Notifications IV

- Broker interface:
  - intermediary Web Service that decouples NotificationConsumers from Publishers
- Demand-based publishing:
  - producing notifications may be costly
  - Broker subscribes to the Publisher
  - When no subscribers for the messages
    - it pauses its subscription
    - resumes when there are subscribers

Resource Properties I

- defines the type and values of a WS-Resource's state that can be viewed and modified
- Resource properties document acts as a view on the actual state
- Described using XML Schema

Resource Properties II

- Defined Messages:
  - *GetResourceProperty*
  - *GetMultipleResourceProperties*
  - *SetResourceProperties*
    - Insert,update,delete
  - *QueryResourceProperties*
    - Using a query expression such as Xpath

Base Fault I

- Target: specifying Web services fault messages in a common way
- defines an XML Schema type for a base fault, along with rules for how this fault type is used

Base Fault II

```
<BaseFault>
  <Timestamp>xsd:dateTime</Timestamp>
  <OriginatorReference>
     wsa:EndpointReferenceType
  </OriginatorReference> ?
  <ErrorCode
  dialect="anyURI">xsd:string</ErrorCode> ?
  <Description>xsd:string</Description> *
  <FaultCause>wsbf:BaseFault</FaultCause> *
</BaseFault>
```

Service Groups I

- defines means by which WS can be grouped together for a domain specific purpose
- ServiceGroup is a WS-Resource, which represents a collection of other Web services
- **MembershipContentRule**: constraints on *membership* of the *service group*
  - E.g. membership can be restricted to members that implement a particular interface
  - no MembershipContentRule elements are specified, the members of the ServiceGroup are unconstrained.

```
<wssg:MembershipContentRule
            MemberInterface="QName"?
            ContentElements="list of QName"
   />
```

Service Groups II

- ServiceGroupRegistration interface defines the message exchanges allow a requestor to add entries to a ServiceGroup (Add Operation)
- Notification of ServiceGroup Modification
  - Topic ServiceGroupModification
  - Notification Messages
    - EntryAdditionNotification
    - EntryRemovalNotification

Renewable Reference

- No specification yet!
- define mechanisms that can be used to renew an endpoint reference that has become invalid
  - reference may contain not only addressing but also policy information concerning interactions with the service
- How?
  - Decorating endpoint references with information necessary to retrieve a new endpoint reference

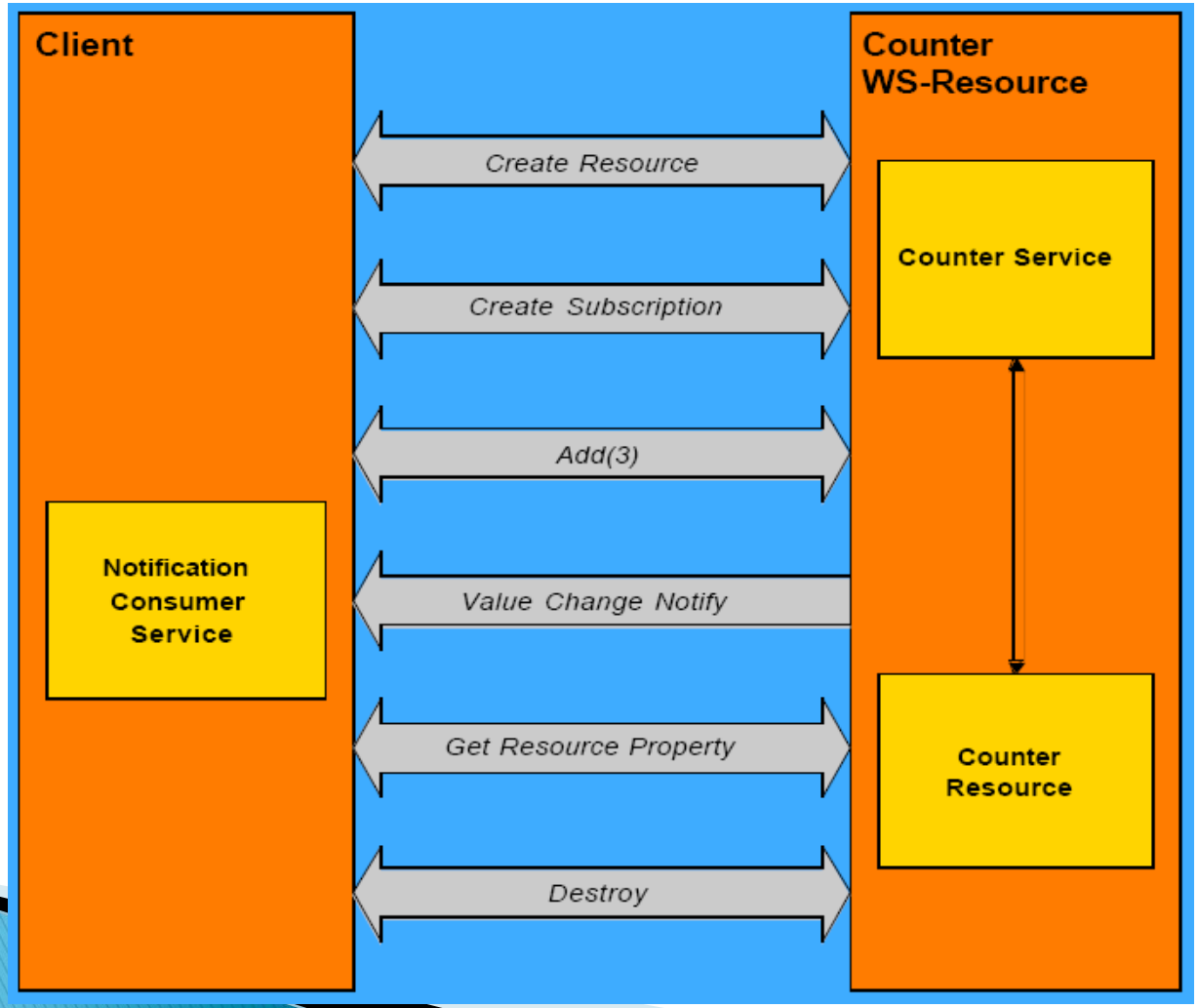
Globus WSRF Preview

- early preview of the Java WSRF Core implementation
  - none of the higher-level services
- GT 4.0 based on WSRF should become available in Quartal 4 of 2004

Example I

- What is required to implement a new service?
  - WSDL
  - Service impl.
  - Resource impl.
  - ResourceHome
  - *Client*
  - *Configuration/Installation*

Example II – Counter Scenario



WSDL I – Properties

```
<types>
   <xsd:schema targetNamespace="http://counter.com"
      xmlns:tns="http://counter.com"
      xmlns:xsd="http://www.w3.org/2001/XMLSchema">
…
      <xsd:element name="Value" type="xsd:int"/>

      <xsd:element name="CounterRP">
         <xsd:complexType>
            <xsd:sequence>
               <xsd:element ref="tns:Value"
                  minOccurs="1" maxOccurs="1"/>
            </xsd:sequence>
         </xsd:complexType>
      </xsd:element>

   </xsd:schema>
</types>
```

WSDL II – Interface

```
<portType name="CounterPortType"
      gtwsdl:implements="wsnt:NotificationProducer
      wsrl:ImmediateResourceTermination"
      wsrp:ResourceProperties ="tns:CounterRP">

      <operation name="createCounter">
         <input message="tns:CreateCounterRequest"/>
         <output message="tns:CreateCounterResponse"/>
      </operation>

      <operation name="add">
         <input message="tns:AddInputMessage"/>
         <output message="tns:AddOutputMessage"/>
      </operation>

 </portType>
```

Service Implementation

```
public _createCounterResponse createCounter(_createCounterRequest request)
{
   ResourceContext ctx = null;
   CounterHome home = null;
   ResourceKey key = null;

   ctx = ResourceContext.getResourceContext();
   home = (CounterHome) ctx.getResourceHome();
   key = home.create();

   EndpointReferenceType epr = AddressingUtils.createEndpointReference(ctx,
    key);

    _createCounterResponse response = new _createCounterResponse();
    response.setEndpointReference(epr);
    return response;
}
```

Service Implementation – add

```
public int add(int arg0) throws RemoteException
{
    Object resource =
            ResourceContext.getResourceContext().getResource();

   Counter counter = (Counter) resource;
    int result = counter.getValue();
    result += arg0;
    counter.setValue(result);
    return result;
 }
```

Resource Implementation

```
public class PersistentCounter
    extends Counter implements PersistentResource {

    public void setValue(int value) {
        super.setValue(value);
        store();
    }

    public Object create() throws Exception {
        Object key = super.create();
        store();
        return key;
    }
    public void load(ResourceKey key) throws ResourceException { …}
    public void store() throws ResourceException { … }
    public void remove() throws ResourceException { … }
}
```

ResourceHome

```
public class CounterHome extends PersistentResourceHome {

    public ResourceKey create() throws Exception {
        Counter counter = (Counter)createNewInstance();
        counter.create();
        ResourceKey key =
            new SimpleResourceKey(keyTypeName, counter.getID());
        this.resources.put(key, counter);
        return key;
    }
}
```

Conclusions

- WSRF refactors OGSA concepts
  - some parts are still missing
  - Grid and Web communities can move forward on a common base

- WS-Resource:

  Web service that acts upon stateful resources

Reference

- An Overview of Service-oriented Architecture, Web Services and Grid Computing
- Web Services Implementation Methodology for SOA Application
- Elements of Service-Oriented Architecture – B. Ramamurthy
- wikipedia.org
- www.w3.org
- www.webservices.org

- IBM: http://www.ibm.com/developerworks/library/ws-resource/
- Globus Java WSRF Core 3.9.1 http://www.globus.org/toolkit/