Unit - 2

GRID SERVICES

- **Open Grid Services Architecture**
- The OGSA is an open source grid service standard jointly developed by academia and the IT industry under coordination of a working group in the Global Grid Forum (GGF).
- The standard was specifically developed for the emerging grid and cloud service communities. The OGSA is extended from web service concepts and technologies.
- OGSA Framework
- The OGSA was built on two basic software technologies: the Globus Toolkit widely adopted as a grid technology solution for scientific and technical computing, and web services (WS 2.0) as a popular standards-based framework for **business and network applications**.

- The OGSA is intended to support the creation, termination, management, and invocation of stateful, transient grid services via standard interfaces and conventions.
- The OGSA framework specifies the physical environment, security, infrastructure profile, resource provisioning, virtual domains, and execution environment for various grid services and API access tools.
- The OGSA is service-oriented.
- A service is an entity that provides some capability to its client by exchanging messages.
- The service oriented architecture (SOA) serves as the foundation of grid computing services. The individual and collective states of resources are specified in this service standard.
- An important point is that the architecture is not layered, where the implementation of one service is built upon modules that are logically dependent.

- **OGSA Interfaces**
- The OGSA is centered on grid services.
- These services demand special well-defined application interfaces.
- These interfaces provide resource discovery, dynamic service creation, lifetime management, notification, and manageability.
- While the OGSA defines a variety of behaviors and associated interfaces, all but one of these interfaces (the grid service) is optional.
- Two key properties of a grid service are **transience and statefulness.**
- These properties have significant implications regarding how a grid service is named, discovered, and managed.
- Transient means the service can be created and destroyed dynamically.
- Statefulness refers to the fact that one can distinguish one service instance from another.

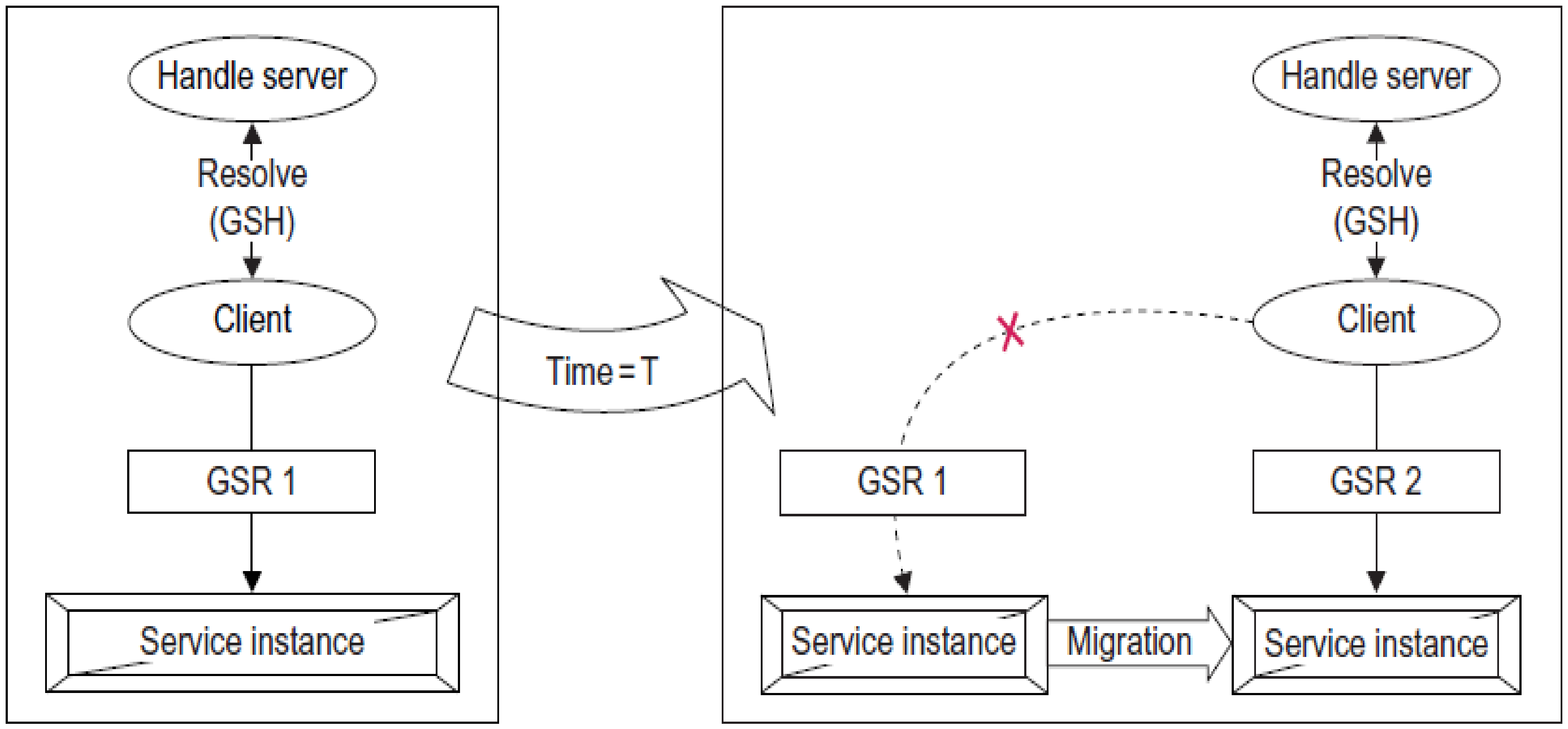
Table 7.3 OGSA Grid Service Interfaces Developed by the OGSA Working Group

Port Type	Operation	Brief Description
Grid service	Find service data	Query a grid service instance, including the handle, reference, primary key, home handle map, interface information, and service-specific information. Extensible support for various query languages.
	Termination time	Set (and get) termination time for grid service instance.
	Destroy	Terminate grid service instance.
Notification source	Subscribe to notification topic	Subscribe to notifications of service events. Allow delivery via third-party messaging services.
Notification sink	Deliver notification	Carry out asynchronous delivery of notification messages.
Registry	Register service	Conduct soft-state registration of Grid Service Handles (GSHs).
	Unregister service	Unregister a GSH.
Factory	Create service	Create a new grid service instance.
Handle map	Find by handle	Return the Grid Service Reference (GSR) associated with the GSH.

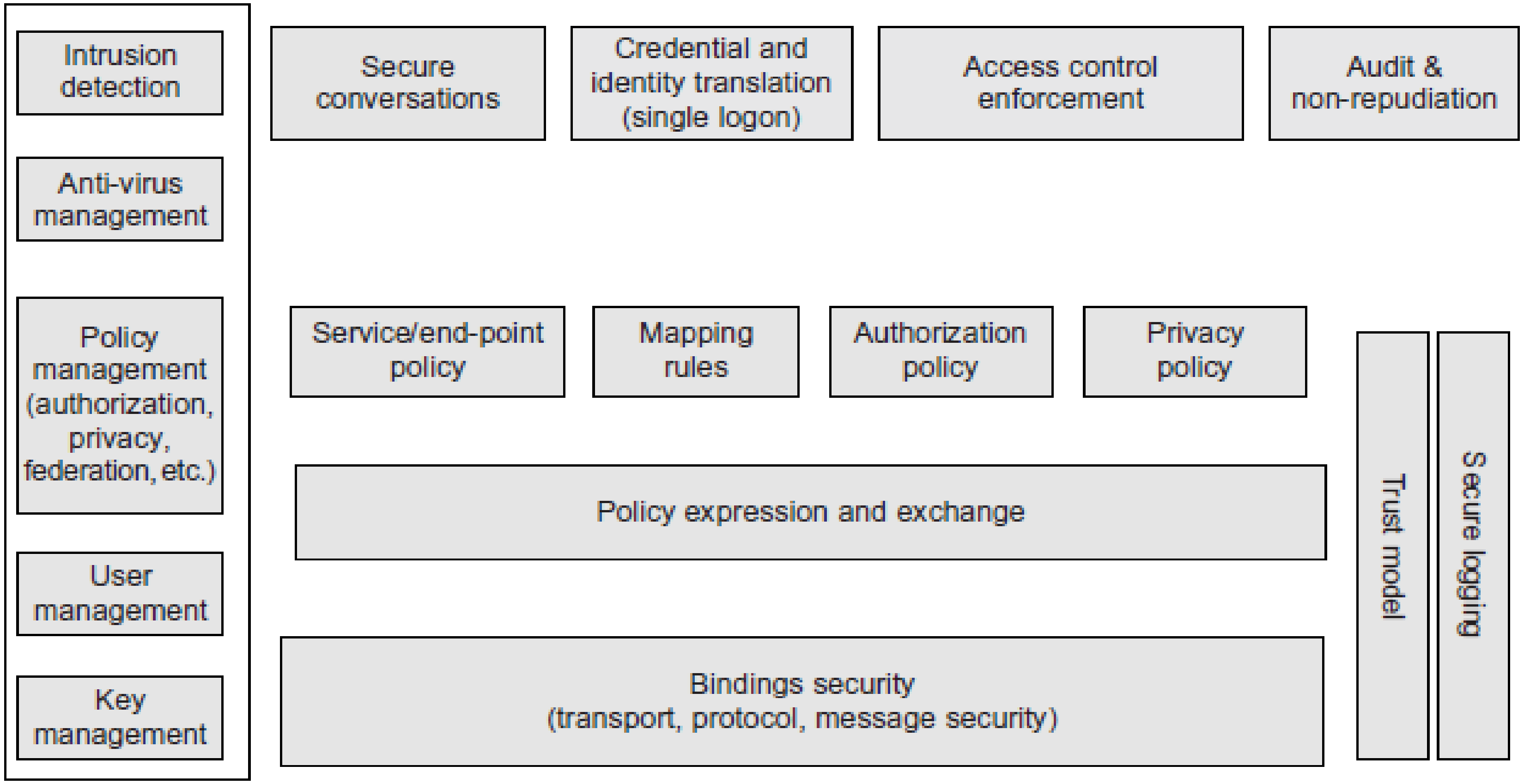
- Grid Service Handle
- A GSH is a globally unique name that distinguishes a specific grid service instance from all others.
- The status of a grid service instance could be that it exists now or that it will exist in the future.
- These instances carry no protocol or instance-specific addresses or supported protocol bindings.
- Instead, these information items are encapsulated along with all other instance-specific information.
- In order to interact with a specific service instance, a single abstraction is defined as a GSR.

- GSH, which is time-invariant.
- The GSR for an instance can change over the lifetime of the service.
- The OGSA employs a “handle-resolution” mechanism for mapping from a GSH to a GSR.
- The GSH must be globally defined for a particular instance.

- Grid Service Migration
- This is a mechanism for creating new services and specifying assertions regarding the lifetime of a service.
- The OGSA model defines a standard interface, known as a factor, to implement this reference.
- Any service that is created must address the former services as the reference of later services.
- The factory interface is labeled as a Create Service operation.
- This creates a requested grid service with a specified interface and returns the GSH and initial GSR for the new service instance.


Handle server
Resolve (GSH)
Client
GSR 1
Service instance
Time = T
Handle server
Resolve (GSH)
Client
GSR 1
GSR 2
Service instance
Migration
Service instance

- **OGSA Security Models**
- The OGSA supports security enforcement at various levels.


Intrusion detection
Anti-virus management
Policy management (authorization, privacy, federation, etc.)
User management
Key management
Secure conversations
Credential and identity translation (single logon)
Access control enforcement
Audit & non-repudiation
Service/end-point policy
Mapping rules
Authorization policy
Privacy policy
Policy expression and exchange
Bindings security (transport, protocol, message security)
Trust model
Secure logging

- The grid works in a heterogeneous distributed environment, which is essentially open to the general public.
- We must be able to detect intrusions or stop viruses from spreading by implementing secure conversations, single logon, access control, and auditing for nonrepudiation.
- At the security policy and user levels, we want to apply a service or endpoint policy, resource mapping rules, authorized access of critical resources, and privacy protection.
- At the Public Key Infrastructure (PKI) service level, the OGSA demands security binding with the security protocol stack and bridging of certificate authorities (CAs), use of multiple trusted intermediaries, and so on.
- Trust models and secure logging are often practiced in grid platforms.

- **FUNCTIONALITY REQUIREMENTS**
- The development of the OGSA document has been based on a variety of use case scenarios.
- The use cases have not been defined with a view to expressing formal requirements but have provided useful input to the definition process.
- **Basic Functionality Requirements**
- **Security Requirements**
- **Resource Management Requirements**
- **System Properties Requirements**
- **Other Functionality Requirements**

- **Basic Functionality Requirements**
- *Discovery and brokering*
- *Metering and accounting*
- *Data sharing*
- *Deployment*
- *Virtual organizations*
- *Monitoring*
- *Policy*

- **Security Requirements**
- *Multiple security infrastructures*
- *Perimeter security solutions*
- *Authentication, Authorization, and Accounting*
- *Encryption*
- *Application and Network-Level Firewalls*
- *Certification*

- **Resource Management Requirements**
- *Provisioning*
- *Resource virtualization*
- *Optimization of resource usage*
- *Transport management*
- *Access*
- *Management and monitoring*
- *Processor scavenging*
- *Scheduling of service tasks*
- *Load balancing*
- *Advanced reservation*
- *Notification and messaging*
- *Logging*
- *Workflow management*
- *Pricing*

- **System Properties Requirements**
- *Fault tolerance*
- *Disaster recovery*
- *Self-healing capabilities*
- *Strong monitoring*
- *Administration*
- *Agreement-based interaction*
- *Grouping/aggregation of services*

- **Other Functionality Requirements**
- *Platforms*
- *Mechanisms*
- *Administrative environments*

- **A PRACTICAL VIEW**
- OGSA aims at addressing standardization (for interoperability) by defining the basic framework of a grid application structure.
- The OGSA standard defines what grid services are, what they should be capable of, and what technologies they are based on.
- It is called an *architecture* because it is mainly about describing and building a well-defined set of interfaces from which systems can be built, based on open standards such as WSDL.
- The objectives of OGSA are
- Manage resources across distributed heterogeneous platforms
- Support QoS-oriented Service Level Agreements (SLAs).
- Define open, published interfaces and protocols for the interoperability of diverse resources.
- Exploit industry standard integration technologies

- OGSI document consists of specifications on how work is managed, distributed, and how service providers and grid services are described.
- The **Web services component** is utilized to facilitate the **distribution and the management** of work across the grid.
- WSDL provides a simple method of describing and advertising the Web services that support the grid’s application.
- Summarizing these observations, OGSA is the blueprint, OGSI is a technical specification, and Globus Toolkit is an implementation of the framework.
- OGSA describes and defines a Web-services-based architecture composed of a set of interfaces and their corresponding behaviors to facilitate distributed resource sharing and accessing in heterogeneous dynamic environments.

- OGSA relies on the definition of grid services in WSDL, which, as noted, defines, for this context, the operations names, parameters, and their types for grid service access.
- Based on the OGSI specification, a grid service instance is a Web service that conforms to a set of conventions expressed by the WSDL as service interfaces, extensions, and behaviors.
- Because the OGSI standard is based on a number of existing standards (XML, Web services, WSDL), it is an open and standards-based solution.
- The grid service interface is described by WSDL, which defines how to use the service. A new tag, **gsdl**, has been added to the WSDL document for grid service description.
- The UDDI registry and WSIL document are used to locate grid services. The transport protocol SOAP is used to connect data and applications for accessing grid services.

- The interfaces of grid services address discovery, dynamic service-instance creation, lifetime management, notification, and manageability.
- The standard interface of a grid service includes multiple bindings and implementations.
- Grid services, such as the ones just cited, can, therefore, be deployed on different hosting environments, even different operating systems.
- OGSA also provides a grid security mechanism to ensure that all the communications between services are secure.

Proposed OGSA grid service interfaces*

Port type	Operation	Description
GridService	FindServiceData	Query a variety of information about the grid service instance, including basic introspection information (handle, reference, primary key, home handle map: terms to be defined), richer per-interface information, and service-specific information (e.g., service instances known to a registry). Extensible support for various query languages.
	SetTermination Time	Set (and get) termination time for grid service instance
	Destroy	Terminate grid service instance.
Notification-Source	SubscribeTo-NotificationTopic	Subscribe to notifications of service-related events, based on message type and interest statement. Allows for delivery via third-party messaging services.
Notification-Sink	Deliver Notification	Carry out asynchronous delivery of notification messages.
Registry	RegisterService UnregisterService	Conduct soft-state registration of grid service handles. Deregister a grid service handle.
Factory	CreateService	Create new grid service instance.
Handle Map	FindByHandle	Return grid service reference currently associated with supplied grid service handle.

*Interfaces for authorization, policy management, manageability, and likely other purposes remain to be defined.

- The majority of Web services, OGSI services use WSDL as a service description mechanism.
- There are two fundamental requirements for describing Web services based on the OGSI.

1. The ability to describe interface inheritance—a basic concept with most of the distributed object systems.
2. The ability to describe additional information elements with the interface definitions.

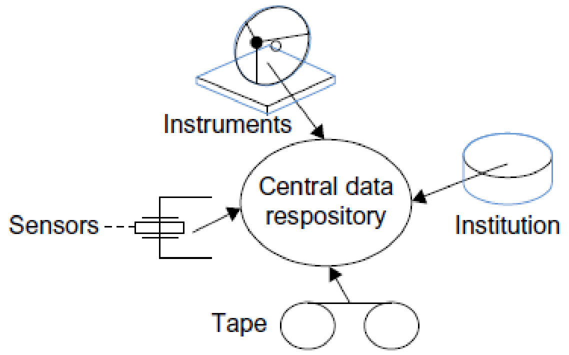
- **Data-Intensive Grid Service Models**
- Applications in the grid are normally grouped into two categories: computation-intensive and data intensive.
- For data-intensive applications, we may have to deal with massive amounts of data.
- The grid system must be specially designed to discover, transfer, and manipulate these massive data sets.
- Transferring massive data sets is a time-consuming task. Efficient data management demands low-cost storage and high-speed data movement.

- **Data Replication and Unified Namespace**
- This data access method is also known as caching, which is often applied to enhance data efficiency in a grid environment.
- By replicating the same data blocks and scattering them in multiple regions of a grid, users can access the same data with locality of references.
- Furthermore, the replicas of the same data set can be a backup for one another. Some key data will not be lost in case of failures.
- However, data replication may demand periodic consistency checks. The increase in storage requirements and network bandwidth may cause additional problems.

- Replication strategies determine when and where to create a replica of the data. The factors to consider include data demand, network conditions, and transfer cost.
- Replication can be classified into method types: **dynamic and static**.
- static method, the locations and number of replicas are determined in advance and will not be modified.
- static strategies cannot adapt to changes in demand, bandwidth, and storage availability.
- The replication strategy must be optimized with respect to the status of data replicas. For static replication, optimization is required to determine the location and number of data replicas.
- For dynamic replication, optimization may be determined based on whether the data replica is being created, deleted, or moved.

- **Grid Data Access Models**
- Multiple participants may want to share the same data collection. To retrieve any piece of data, we need a grid with a unique global namespace.
- Similarly, we desire to have unique file names. To achieve these, we have to resolve inconsistencies among multiple data objects bearing the same name.
- Access restrictions may be imposed to avoid confusion. Also, data needs to be protected to
- avoid leakage and damage.
- Users who want to access data have to be authenticated first and then authorized for access.

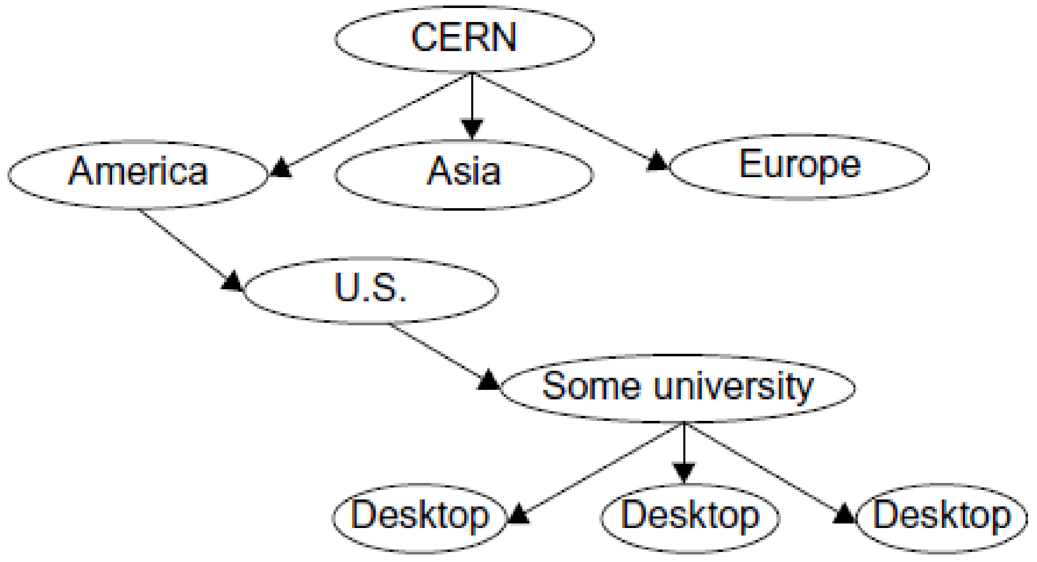
- There are four access models for organizing a data grid.
- Monadic model
- Hierarchical model
- Federation model
- Hybrid model
- **Monadic model**:



(a) Monadic

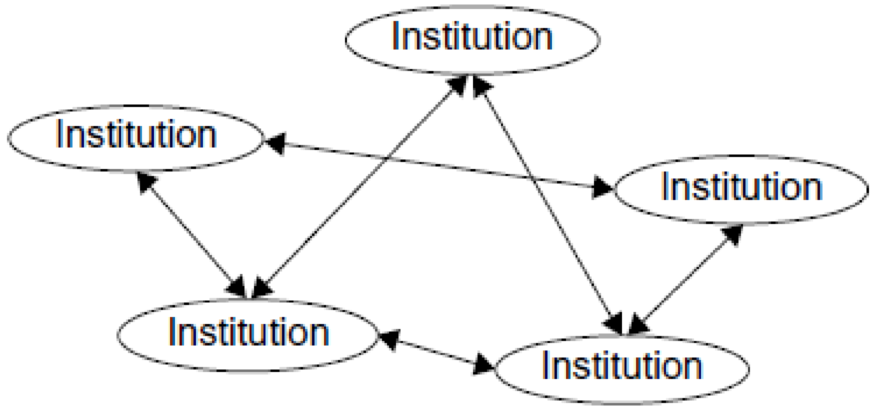
- This is a centralized data repository model.
- All the data is saved in a central data repository. When users want to access some data they have to submit requests directly to the central repository.
- No data is replicated for preserving data locality. This model is the simplest to implement for a small grid.

- The **hierarchical model**, is suitable for building a large data grid which has only one large data access directory.
- The data may be transferred from the source to a second-level center. Then some data in the regional center is transferred to the third-level center. After being forwarded several times, specific data objects are accessed directly by users.



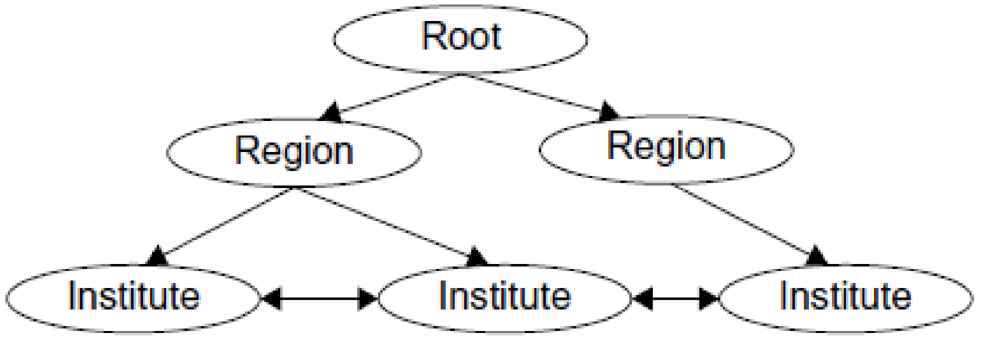
(b) Hierarchy

- **Federation model**
- This **data access model** is better suited for designing a data grid with multiple sources of data supplies. Sometimes this model is also known as a **mesh model**.
- The data sources are distributed to many different locations.
- Although the data is shared, the data items are still owned and controlled by their original owners.
- According to predefined access policies, only authenticated users are authorized to request data from any data source.



(c) Federation

- **Hybrid model**: The model combines the best features of the hierarchical and mesh models. Traditional data transfer technology, such as FTP, applies for networks with lower bandwidth.



(d) Hybrid

- **Parallel versus Striped Data Transfers**
- **parallel data transfer** opens multiple data streams for passing subdivided segments of a file simultaneously. Although the speed of each stream is the same as in sequential streaming, the total time to move data in all streams can be significantly reduced compared to FTP transfer.
- **Striped data transfer**, a data object is partitioned into a number of sections, and each section is placed in an individual site in a data grid. When a user requests this piece of data, a data stream is created for each site, and all the sections of data objects are transferred simultaneously.

- **OGSA/OGSI A MORE DETAILED VIEW**
- This section provides a more detailed view of OGSI based on the OGSI specification.
- The OGSA integrates key grid technologies with Web services mechanisms to create a distributed system framework based on the OGSI.
- A *grid service instance* is a (potentially transient) service that conforms to a set of conventions, expressed as WSDL interfaces, extensions, and behaviors, for such purposes as lifetime management, discovery of characteristics, and notification.
- OGSI defines a component model that extends WSDL and XML schema definition to incorporate the concepts of
  - Stateful Web services
  - Extension of Web services interfaces
  - Asynchronous notification of state change
  - References to instances of services
  - Collections of service instances
  - Service state data that augment the constraint capabilities of XML schema definition

- The OGSI specification (V1.0 at press time) defines the minimal, integrated set of extensions and interfaces necessary to support definition of the services that will compose OGSA.
- The OGSI V1.0 specification proposes detailed specifications for the conventions that govern how clients create, discover, and interact with a grid service instance.
- That is, it specifies

  (1) how grid service instances are named and referenced;

  (2) the base, common interfaces (and associated behaviors) that all grid services implement;

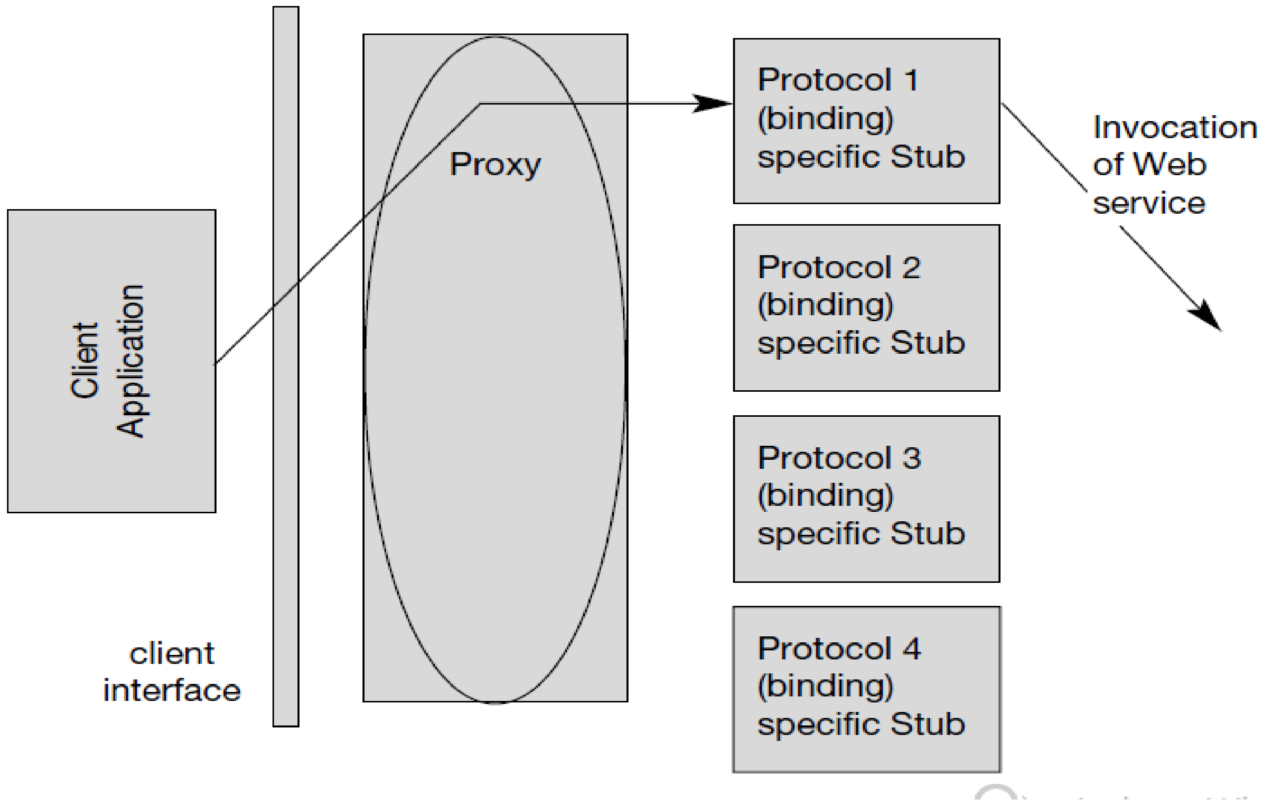
  (3) the additional (optional) interfaces and behaviors associated with factories and service groups.
- The specification does *not* address how grid services are created, managed, and destroyed within any particular hosting environment.

- Global Grid Form(GGF) calls OGSI the “base for OGSA”.
- ***Relationship to Distributed Object Systems***
- A given grid service implementation is an addressable and potentially stateful instance that implements one or more interfaces described by WSDL portTypes.
- Grid service factories can be used to create instances implementing a given set of portType(s).
- Each grid service instance has a notion of identity with respect to the other instances in the distributed grid.
- Grid service instances are made accessible to (potentially remote) client applications through the use of a grid service handle and a grid service reference (GSR).
- A client application can use a grid service reference to send requests, represented by the operations defined in the portType(s) of the target service description directly to the specific instance.

- The characteristics introduced above are frequently also cited as fundamental Characteristics of *distributed object-based systems*.
- OGSI does not adopt the term distributed object model or distributed object system when describing these concepts, but instead uses the term “open grid services infrastructure,” thus emphasizing the connections that are established with both Web services and grid technologies.
- **Client-Side Programming Patterns**
- Another important issue is how OGSI interfaces are likely to be invoked from client applications.
- OGSI exploits an important component of the Web services framework: the use of WSDL to describe multiple protocol bindings, encoding styles, messaging styles (RPC versus document oriented), and so on, for a given Web service.

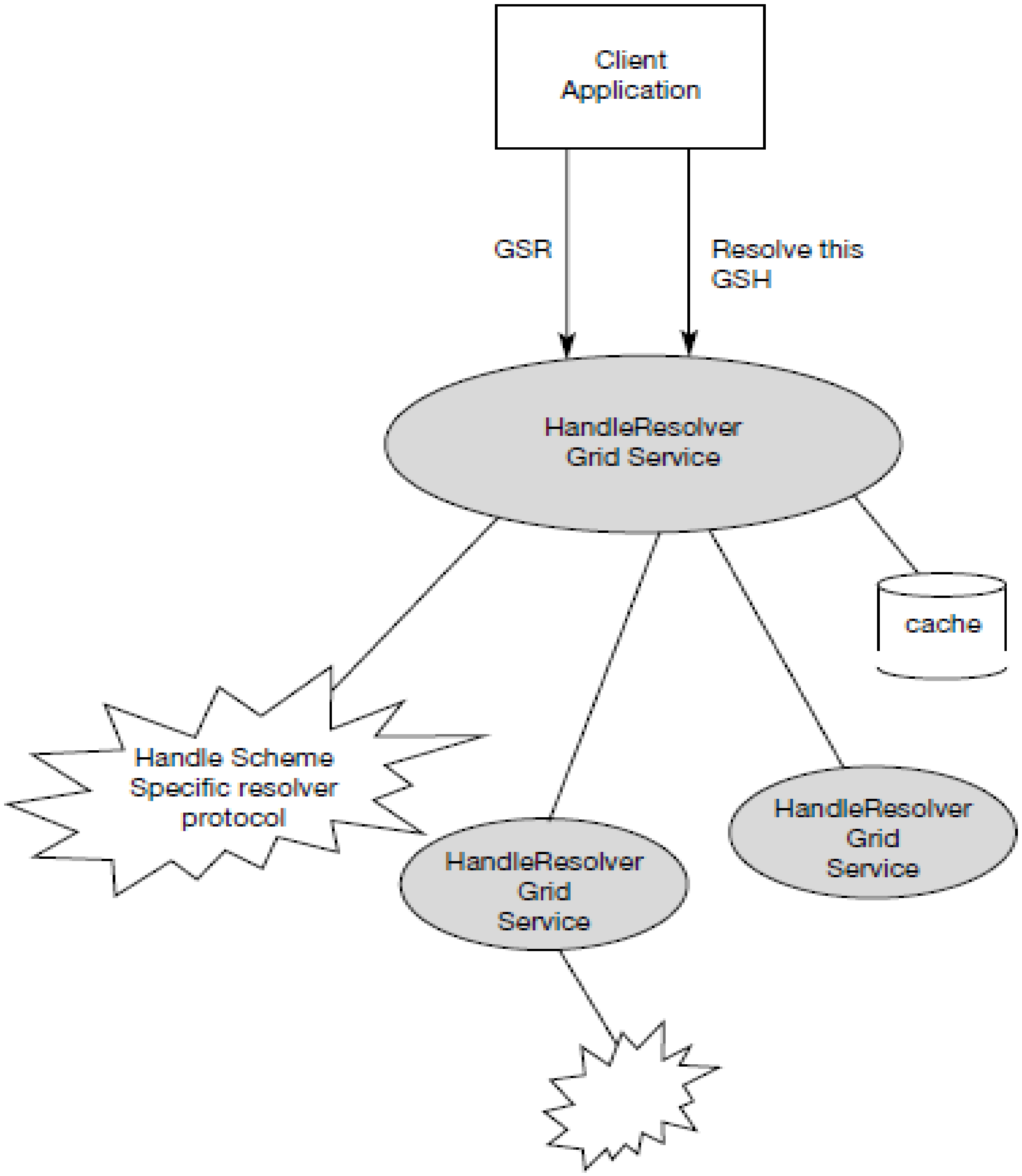
- The Web Services Invocation Framework (WSIF) and Java API for XML RPC (JAX-RPC) are among the many examples of infrastructure software that provide this capability.

Client-side architecture for OGSI

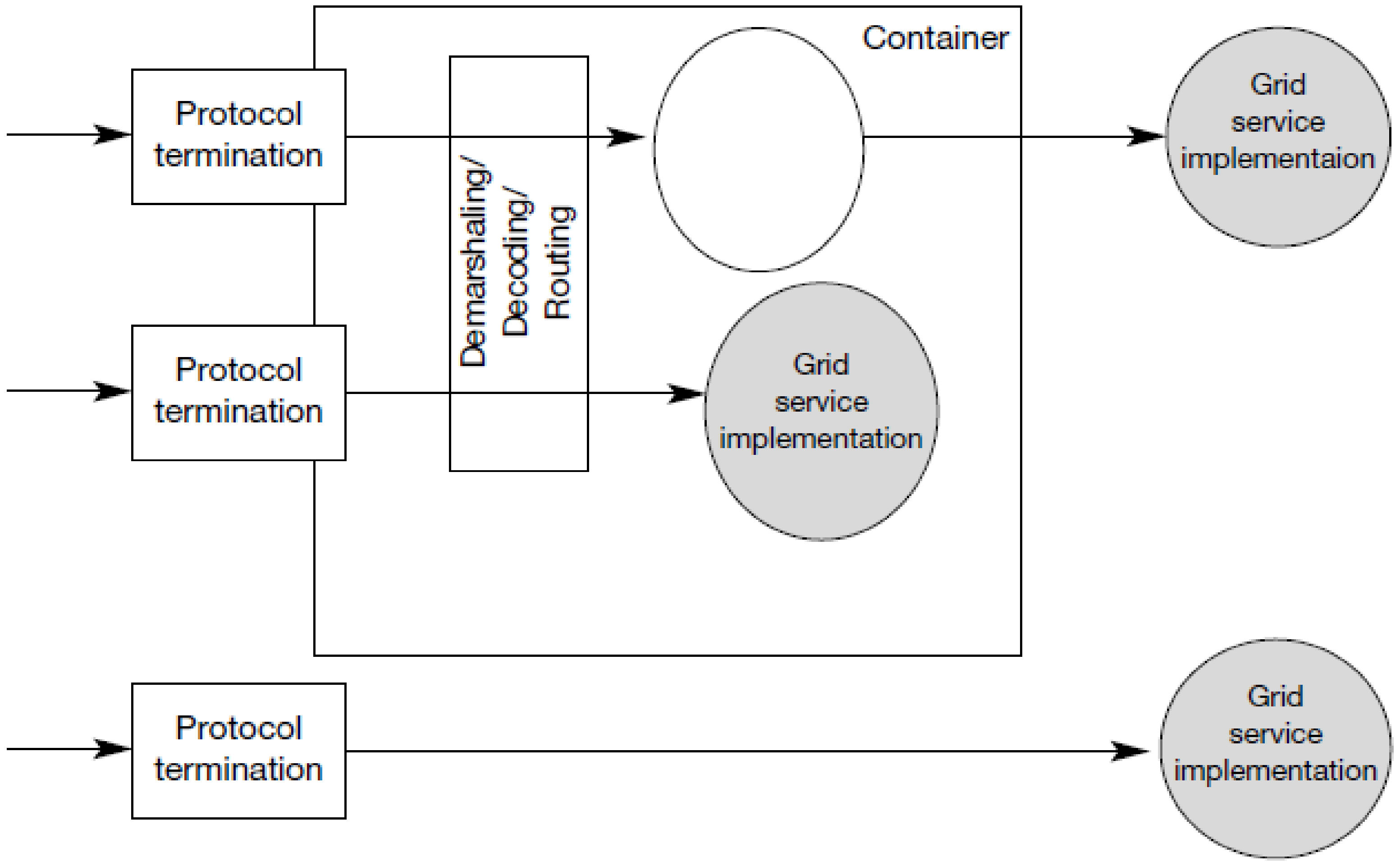

Client Application
client interface
Proxy
Protocol 1 (binding) specific Stub
Protocol 2 (binding) specific Stub
Protocol 3 (binding) specific Stub
Protocol 4 (binding) specific Stub
Invocation of Web service

- In this approach, a clear separation exists between the client application and the client-side representation of the Web service (proxy), including components for marshaling the invocation of a Web service over a chosen binding.
- Various tools can take the WSDL description of the Web service and generate interface definitions in a wide range of programming-language-specific constructs.
- This interface is a front end to specific parameter marshaling and message routing that can incorporate various binding options provided by the WSDL.
- Within the client application runtime, a *proxy* provides a client-side representation of remote service instance's interface.
- Proxy behaviors specific to a particular encoding and network protocol are encapsulated in a *protocol-specific (binding-specific) stub*.

- **Client Use of Grid Service Handles and References**
- A client gains access to a grid service instance through grid service handles and grid service references.
- A grid service handle (GSH) can be thought of as a permanent network pointer to a particular grid service instance. The GSH does not provide sufficient information to allow a client to access the service instance; the client needs to “resolve” a GSH into a grid service reference (GSR).
- The GSR contains all the necessary information to access the service instance. The GSR is not a “permanent” network pointer to the grid service instance.
- OGSI provides a mechanism, the Handle Resolver to support client resolution of a grid service handle into a grid service reference.


Client Application
GSR
Resolve this GSH
HandleResolver Grid Service
cache
Handle Scheme Specific resolver protocol
HandleResolver Grid Service
HandleResolver Grid Service

- ***Relationship to Hosting Environment***
- OGSI does not dictate a particular service-provider-side implementation architecture.
- A variety of approaches are possible, ranging from implementing the grid service instance directly as an operating system process to a sophisticated server-side component model.
- In the former case, most or even all support for standard grid service behaviors (invocation, lifetime management, registration, etc.) is encapsulated within the user process;
- These differences by showing two different approaches to the implementation of argument demarshaling functions.
- One can assume that, as is the case for many grid services, the invocation message is received at a network protocol termination point (e.g., an HTTP servlet engine) that converts the data in the invocation message into a format consumable by the hosting environment.


Container
Protocol termination
Protocol termination
Protocol termination
Demarshaling/ Decoding/ Routing
Grid service implementaion
Grid service implementation
Grid service implementation

- **Grid Service**
- The purpose of the OGSI document is to specify the (standardized) interfaces and behaviors that define a grid service.
- In brief, a grid service is a WSDL-defined service that conforms to a set of conventions relating to its interface definitions and behaviors.
- **Statements:**
- Introducing a set of WSDL conventions
- Defining *service data* that provide a standard way for representing and querying metadata and state data from a service instance.
- Introducing a series of core properties of grid service
- Defining grid service description and grid service instance
- Defining how OGSI models time
- Defining the life cycle of a grid service instance

- **WSDL Extensions and Conventions**
- OGSI is based on Web services; in particular, it uses WSDL as the mechanism to describe the public interfaces of grid services.
- **Service Data**

- The three lifetime declaration properties are:
- ogsi:goodFrom
- ogsi:goodUntil
- ogsi:availableUntil

- **OGSA SERVICES**
- **Handle Resolution**
- OGSI defines a two-level naming scheme for grid service instances based on **abstract**, **long-lived *grid service handles*** (GSHs) that can be mapped by HandleMapper services to concrete, but potentially less long lived, *grid service references* (GSRs).
- A client application can use a grid service reference to send requests (represented by the operations defined in the interfaces of the target service) directly to the specific instance at the specified network-attached service endpoint identified by that GSR.
- **Virtual Organization Creation and Management**
- VOs are a concept that supplies a "context" for operation of the grid that can be used to associate users, their requests, and resources.
- VO contexts permit the grid resource providers to associate appropriate policy and agreements with their resources.

- VO creation and management functions include
- Associating users/groups with a VO,
- manipulation of user roles (administration, configuration, use, etc.) within the VO.
- association of services (encapsulated resources) with the VO
- attachment of agreements and policies to the VO as a whole or to individual services within the VO

- **Service Groups and Discovery Services**
- GSHs and GSRs together realize a two-level naming scheme, with HandleResolver services mapping from handles to references.

- Other entities (both humans and applications) need other means for discovering services with particular properties, whether relating to interface, function, availability, location, policy, or other criteria.

- Traditionally, in distributed systems this problem is addressed by creating a third-level “human-readable” or “semantic” name space that is then mapped (bound) to abstract names (in this case, GSHs) via registry, discovery, metadata catalog, or other similar services.
- It is important that OGSA defines standard functions for managing such name spaces.
- Two types of such semantic name spaces are common—naming by attribute, and naming by path.

- **Choreography, Orchestration, and Workflow**
- Rich set of behaviors and associated operations and attributes for business process management.
- Definition of a job flow, including associated policies
- Assignment of resources to a grid flow instance
- Scheduling of grid flows (and associated grid services)
- Execution of grid flows (and associated grid services)
- Common context and metadata for grid flows (and associated services)
- Management and monitoring for grid flows (and associated grid services)
- Failure handling for grid flows
- Business transaction and coordination services

- **Transactions**
- **Metering Service**
- **Rating Service**
- **Accounting Service**
- accounting service can manage subscription users and accounts information, calculate the relevant monthly charges and maintain the invoice information.
- **Billing and Payment Service**
- Billing and payment service refers to the financial service that actually carries out the transfer of money; for example, a credit card authorization service.
- **Installation, Deployment, and Provisioning**
- Computer processors, applications, licenses, storage, networks, and instruments are all grid resources that require installation, deployment, and provisioning

- **Distributed Logging**
- Distributed logging can be viewed as a typical messaging application in which *message producers* generate *log artifacts*, (atomic expressions of diagnostic information) that may or may not be used at a later time by other independent *message consumers*.
- Logging services provide the extensions needed to deal with the following issues:
- *Decoupling*
- *Transformation and common representation*
- *Filtering and aggregation*
- *Configurable persistency*
- *Consumption patterns*

- **Messaging and Queuing**
- OGSA extends the scope of the base OGSI Notification Interface to allow grid services to produce a range of event messages, not just notifications that a serviceData element has changed.
- **Events**
- Events are generally used as asynchronous signaling mechanisms. The most common form is "publish/subscribe," in which a service "publishes" the events that it exports (makes available to clients). The service may publish the events as reliable or best effort. Clients may then "subscribe" to the event, and when the event is raised, a call-back or message is sent to the client.
- **Policy and Agreements**
- These services create a general framework for creation, administration, and management of policies and agreements for system operation, security, resource allocation, and so on, as well as an infrastructure for "policy aware" services to use the set of defined and managed policies to govern their operation.