

Introduction to the Open Grid Services Architecture

Keith R. Jackson
Lawrence Berkeley National Laboratory
krjackson@lbl.gov

Overview

- Background
 - The Grid Problem
 - The Globus Approach
 - OGSA & OGSi
 - Globus Toolkit
- GT3 Architecture and Functionality: The Latest Refinement of the Globus Toolkit
 - Core
 - Base Services
 - User-Defined Services
 - Future Directions
- PyOGSi
 - Client Bindings
 - Security
 - Server Support
 - Status
- Acknowledgements

A Story of Evolution

- Definition of Grid problem has been stable since original Globus Project proposal in 1995
 - Though we've gotten better at articulating it
- But our approach to its solution has evolved:
 - From APIs and custom protocols...
 - to standard protocols...
 - to Grid services (OGSA).
- Driven by experience implementing and deploying the Globus Toolkit, and building real applications with it

What is a Grid?

- We believe there are three key criteria:
 - Coordinates distributed resources ...
 - using standard, open, general-purpose protocols and interfaces ...
 - to deliver non-trivial qualities of service.
- What is not a Grid?
 - A cluster, a network attached storage device, a scientific instrument, a network, etc.
 - Each is an important component of a Grid, but by itself does not constitute a Grid

Challenging Technical Requirements

- Dynamic formation and management of virtual organizations
- Discovery & online negotiation of access to services: who, what, why, when, how
- Configuration of applications and systems able to deliver multiple qualities of service
- Autonomic management of distributed infrastructures, services, and applications
- Management of distributed state
- Open, extensible, evolvable infrastructure

The Globus Project™

- Close collaboration with real Grid projects in science and industry
- The Globus Toolkit®: Open source software base for building Grid infrastructure and applications
- Development and promotion of standard Grid protocols to enable interoperability and shared infrastructure
- Development and promotion of standard Grid software APIs to enable portability and code sharing
- Global Grid Forum: We co-founded GGF to foster Grid standardization and community

From APIs & Custom Protocols, To Standard Protocols

Application Programming Interface

- A specification for a set of routines to facilitate application development
 - Refers to definition, not implementation
- Often language-specific (or IDL)
 - Routine name, number, order and type of arguments; mapping to language constructs
 - Behavior or function of routine
- Examples of APIs
 - GSS-API (security), MPI (message passing)

Network Protocol

- A formal description of message formats and a set of rules for message exchange
 - Rules may define sequence of message exchanges
 - Protocol may define state-change in endpoint, e.g., file system state change
- Good protocols designed to do one thing
 - Protocols can be layered
- Examples of protocols
 - IP, TCP, TLS (was SSL), HTTP, Kerberos

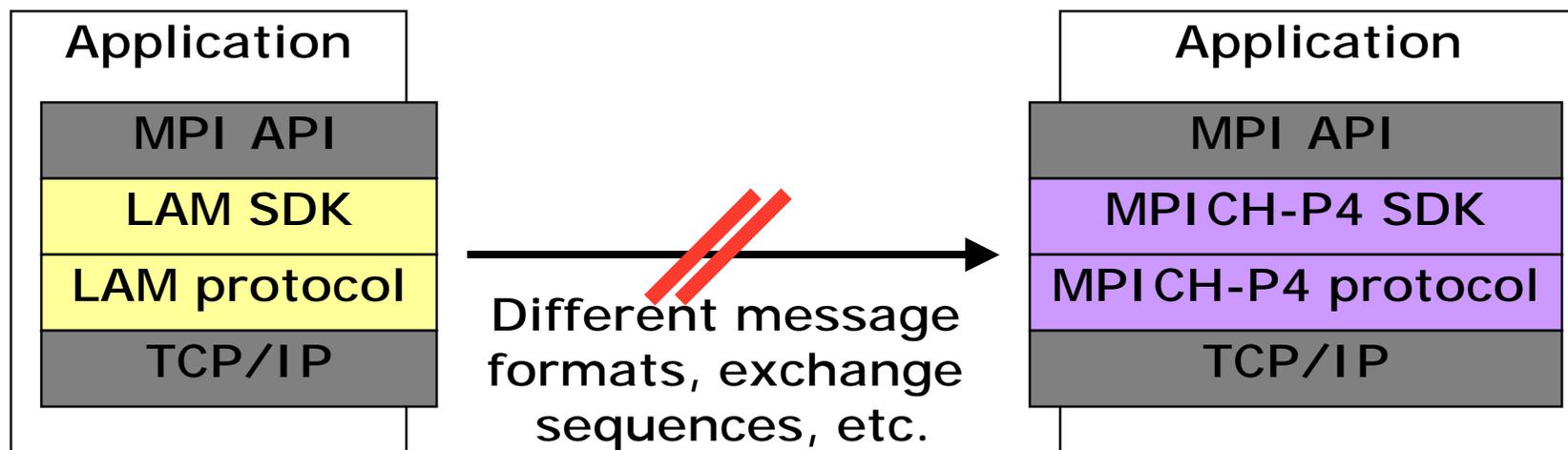
A Protocol can have Multiple APIs

- TCP/IP APIs include BSD sockets, Winsock, System V streams, ...
- The protocol provides **interoperability**: programs using different APIs can exchange information
- I don't need to know remote user's API



An API can have Multiple Protocols

- An API provides **portability**: any correct program compiles & runs on a platform
- Does not provide interoperability: all processes must link against same SDK
 - E.g., MPICH and LAM versions of MPI



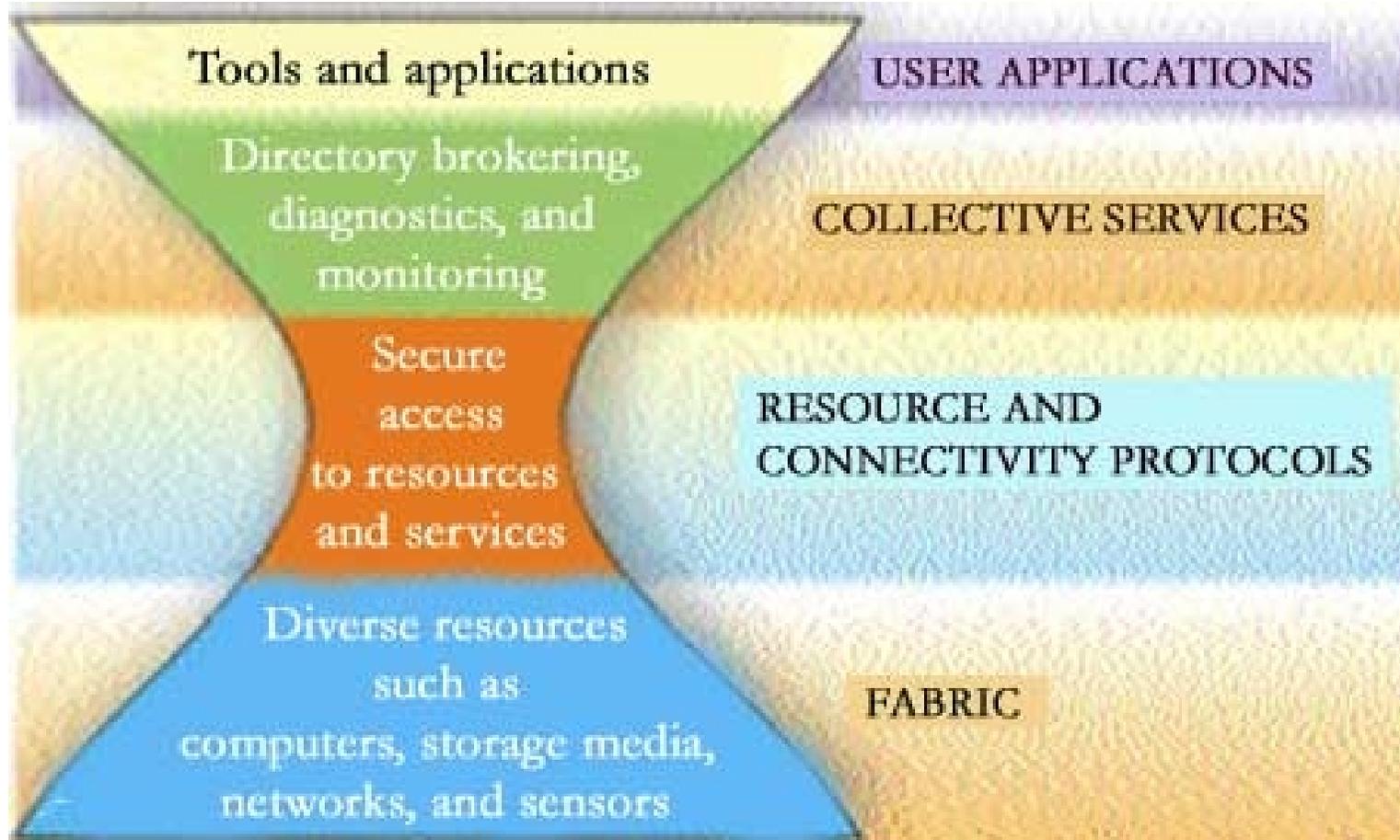
Initial Focus On APIs and Custom Protocols

- Primary concern was allowing Grid applications to be built quickly, in order to demonstrate feasibility
- Good development APIs and SDKs mattered most
- Protocols were a means to an end
 - We borrowed and extended standard protocols to make life easier (e.g. LDAP)
 - We defined custom protocols (e.g. GRAM)

But Focus Shifted To Protocols

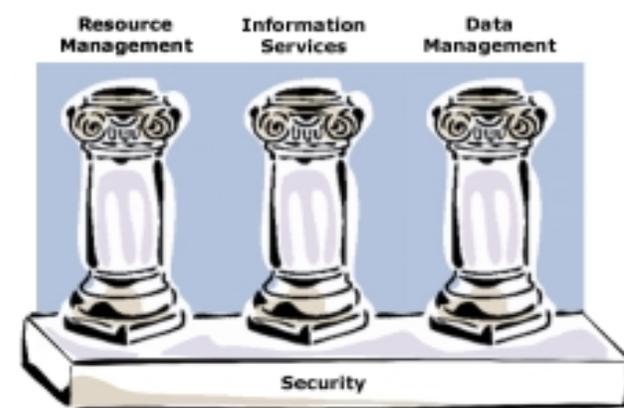
- As demand grew, customers worried about:
 - compatibility between versions (i.e. Stop changing the protocols!)
 - independent implementations of some components (i.e. What are the protocols?)
- Ubiquitous adoption demands open, standard protocols
 - Internet and Web as guides
 - Enables innovation/competition on end points
 - Avoid product/vendor lock-in

Layers of Grid Architecture



GT2

Key Protocols



- The Globus Toolkit v2 (GT2) centers around four key protocols
 - Connectivity layer:
 - *Security*: Grid Security Infrastructure (GSI)
 - Resource layer:
 - *Resource Management*: Grid Resource Allocation Management (GRAM)
 - *Information Services*: Grid Resource Information Protocol (GRIP)
 - *Data Transfer*: Grid File Transfer Protocol (GridFTP)
- Also key collective layer protocols
 - Info Services, Replica Management, etc.

From Standard Protocols, To Grid Services

But Along The Way...

- Heterogeneous protocol base was hurting us
- Increasing number of virtual services that needed to be managed
- Web services (WSDL, SOAP) appeared

Web Services

- At the heart of Web services is:
 - WSDL: Language for defining abstract service interfaces
 - SOAP (and friends): Binding from WSDL to bytes on the wire
- Web services appears to offer a fighting chance at ubiquity (unlike CORBA)
- 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, subscription
- Significant implications for how services are managed, named, discovered, and used
 - In fact, much of Grid is concerned with the management of service instances

Standard Interfaces & Behaviors: Four Interrelated Concepts

- Naming and bindings
 - Every service instance has a unique name, from which can discover supported bindings
- Lifecycle
 - Service instances created by factories
 - Destroyed explicitly or via soft state
- Information model
 - Service data associated with Grid service instances, operations for accessing this info
 - Basis for service introspection, monitoring, discovery
- Notification
 - Interfaces for registering existence, and delivering notifications of changes to service data

Grid Evolution: Open Grid Services Architecture

- Refactor Globus protocol suite to enable common base and expose key capabilities
- Service orientation to virtualize resources and unify resources/services/information
- Embrace key Web services technologies for standard IDL, leverage commercial efforts
- Result: standard interfaces & behaviors for distributed system management: the Grid service

OGSA Structure

- A standard substrate: the Grid service
 - OGSi = Open Grid Service Infrastructure
 - Standard interfaces and behaviors that address key distributed system issues
 - Much borrowed from GT abstractions
- ... supports standard service specifications
 - Resource mgt, dbms, workflow, security, ...
 - Target of current & planned GGF efforts
- ... and arbitrary application-specific services based on these & other definitions

OGSI Grid Service Specification

- Defines WSDL conventions and GSDL extensions
 - For describing and structuring services
 - Working with W3C WSDL working group to drive GSDL extensions into WSDL
- Defines fundamental interfaces (using WSDL) and behaviors that define a Grid Service
 - A unifying framework for interoperability & establishment of total system properties

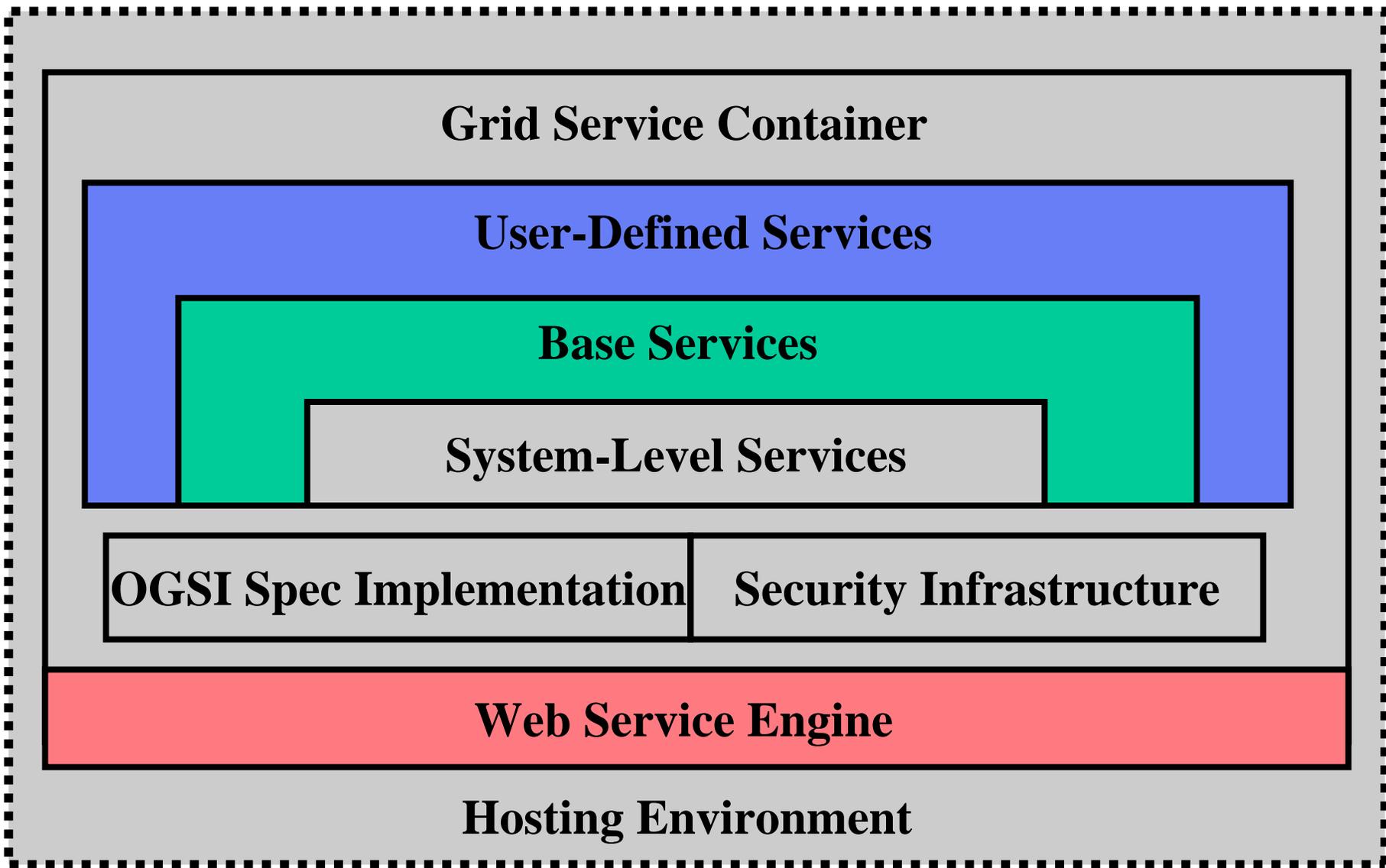
OGSA and the Globus Toolkit

- Technically, OGSA enables
 - Refactoring of protocols (GRAM, MDS, GridFTP), *while preserving all GT concepts/features!*
 - Integration with hosting environments: simplifying components, distribution, etc.
 - Greatly expanded standard service set
- Pragmatically, we are proceeding as follows
 - Develop open source OGSA implementation
 - Globus Toolkit 3.0; supports Globus Toolkit 2.0 APIs
 - Partnerships for service development
 - Also expect commercial value-adds

GT2 Evolution To GT3

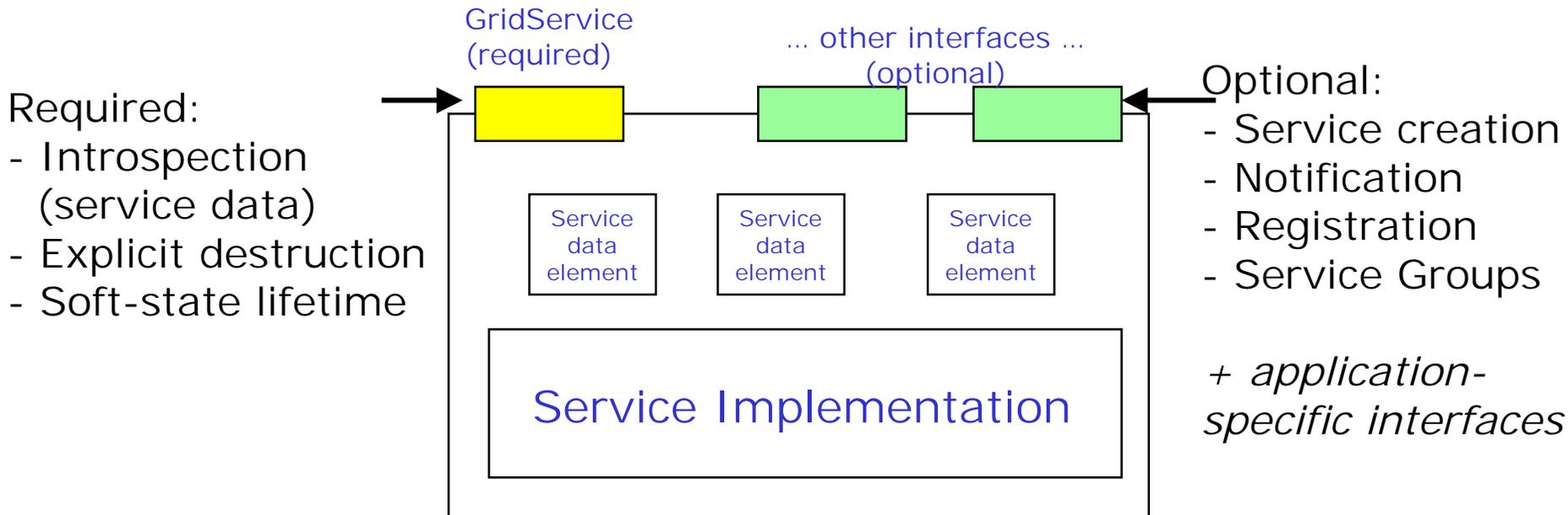
- What happened to the GT2 key protocols?
 - Security: Adapting X.509 proxy certs to integrate with emerging WS standards
 - GRIP/LDAP: Abstractions integrated into OGSI as serviceData
 - GRAM: ManagedJobFactory and related service definitions
 - GridFTP: Unchanged in 3.0, but will evolve into OGSI-compliant service in 2004
- Also rendering collective services in terms of OGSI: RFT, RLS, etc.

GT-OGSA Grid Service Infrastructure



OGSI Specification

The Specification Defines how Entities can Create, Discover and Interact with a Grid Service



GT3 Core: OGSI Spec Implementation

- GT3 includes a set of primitives that implement the interfaces and behaviors defined in the latest version of the OGSI Specification
- The implementation supports a declarative programming model in which GT3 users can compose OGSI-Compliant grid services by plugging the desired primitives into their implementation

OGSI Specification (cont.)

GridService portType

- Defines the fundamental behavior of a Grid Service
 - Introspection
 - Discovery
 - Soft State Lifetime Management
- Mandated by the Spec

OGSI Specification (cont.)

Factory portType

- Factories create services
- Factories are typically persistent services
- Factory is an optional OGSI interface

(Grid Services can also be instantiated by other mechanisms)

OGSI Specification (cont.)

Notification portTypes

- A subscription for notification causes the creation of a NotificationSubscription service
- NotificationSinks are not required to implement the GridService portType
- Notification portTypes are optional

OGSI Specification (cont.)

Service group portTypes

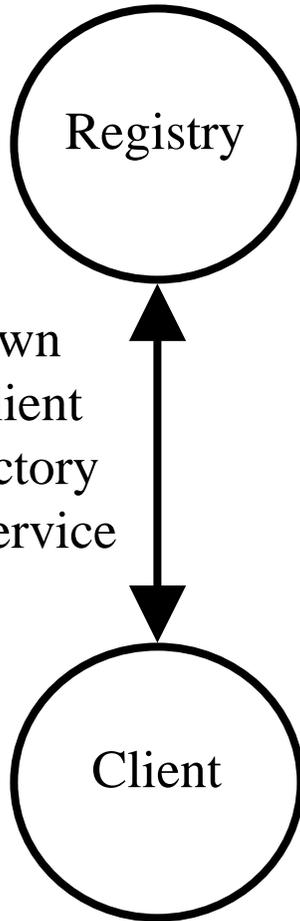
- A ServiceGroup is a grid service that maintains information about a group of other grid services
- The classic registry model can be implemented with the ServiceGroup portTypes
- A grid service can belong to more than one ServiceGroup
- Members of a ServiceGroup can be heterogenous or homogenous
- Each entry in a service group can be represented as its own service
- Service group portTypes are optional OGSI interfaces

OGSI Specification (cont.)

HandleResolver portType

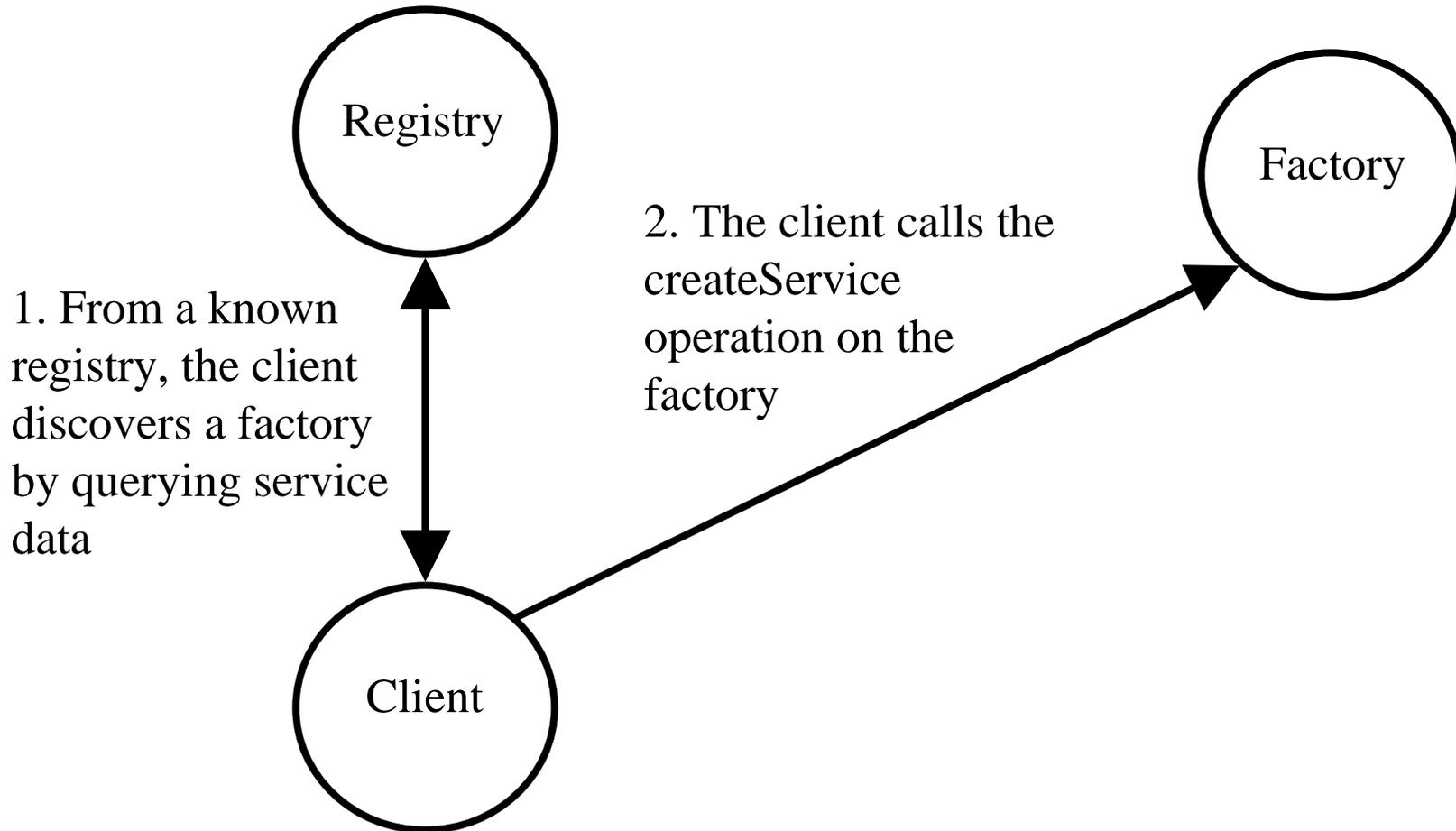
- Defines a means for resolving a GSH (Grid Service Handle) to a GSR (Grid Service Reference)
 - A GSH points to a Grid Service
(GT3 uses a hostname-based GSH scheme)
 - A GSR specifies how to communicate with the Grid Service
(GT3 currently supports SOAP over HTTP, so GSRs are in WSDL format)
- HandleResolver is an optional OGSI interface

A Service Creation Scenario

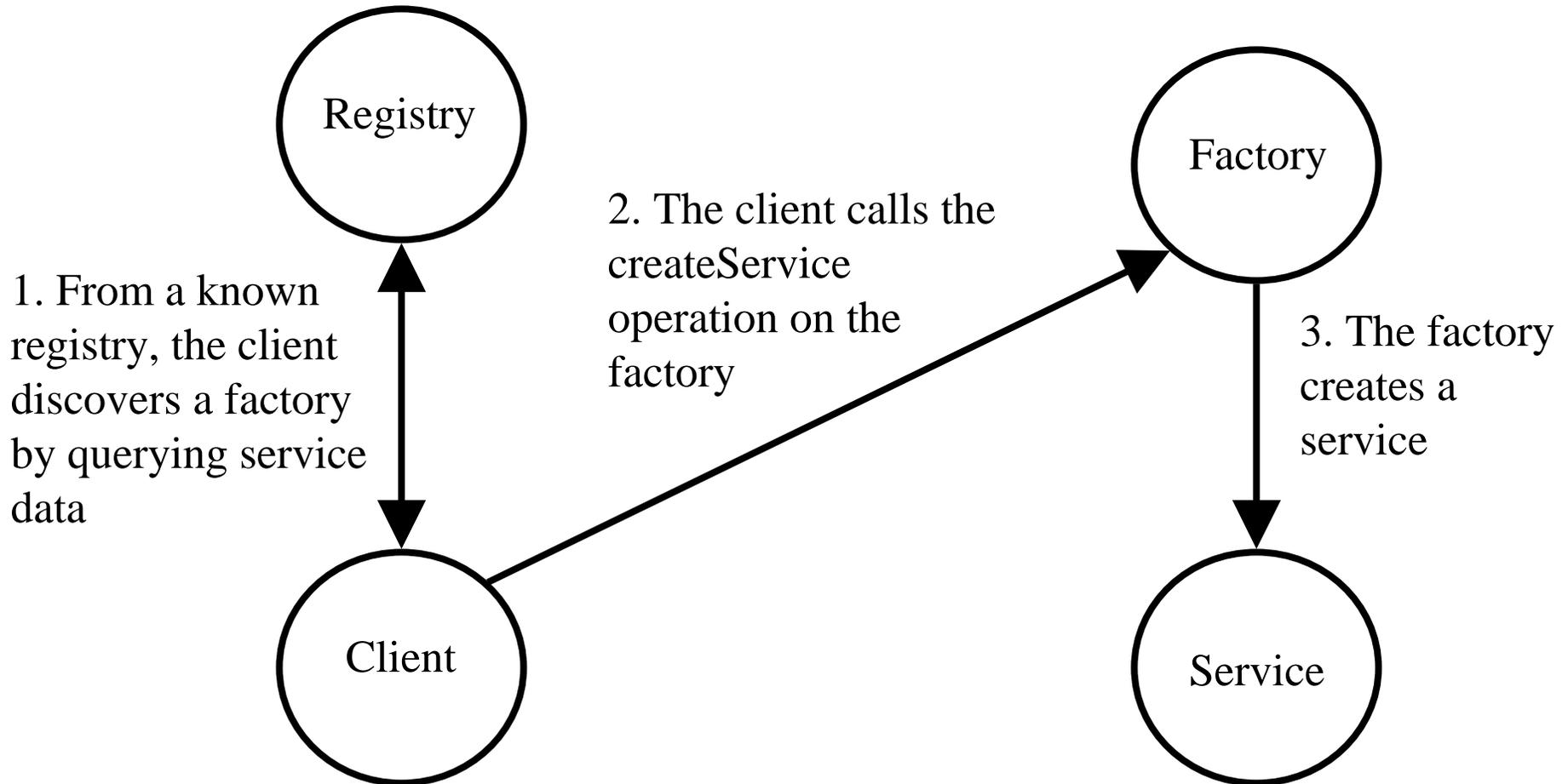


1. From a known registry, the client discovers a factory by querying service data

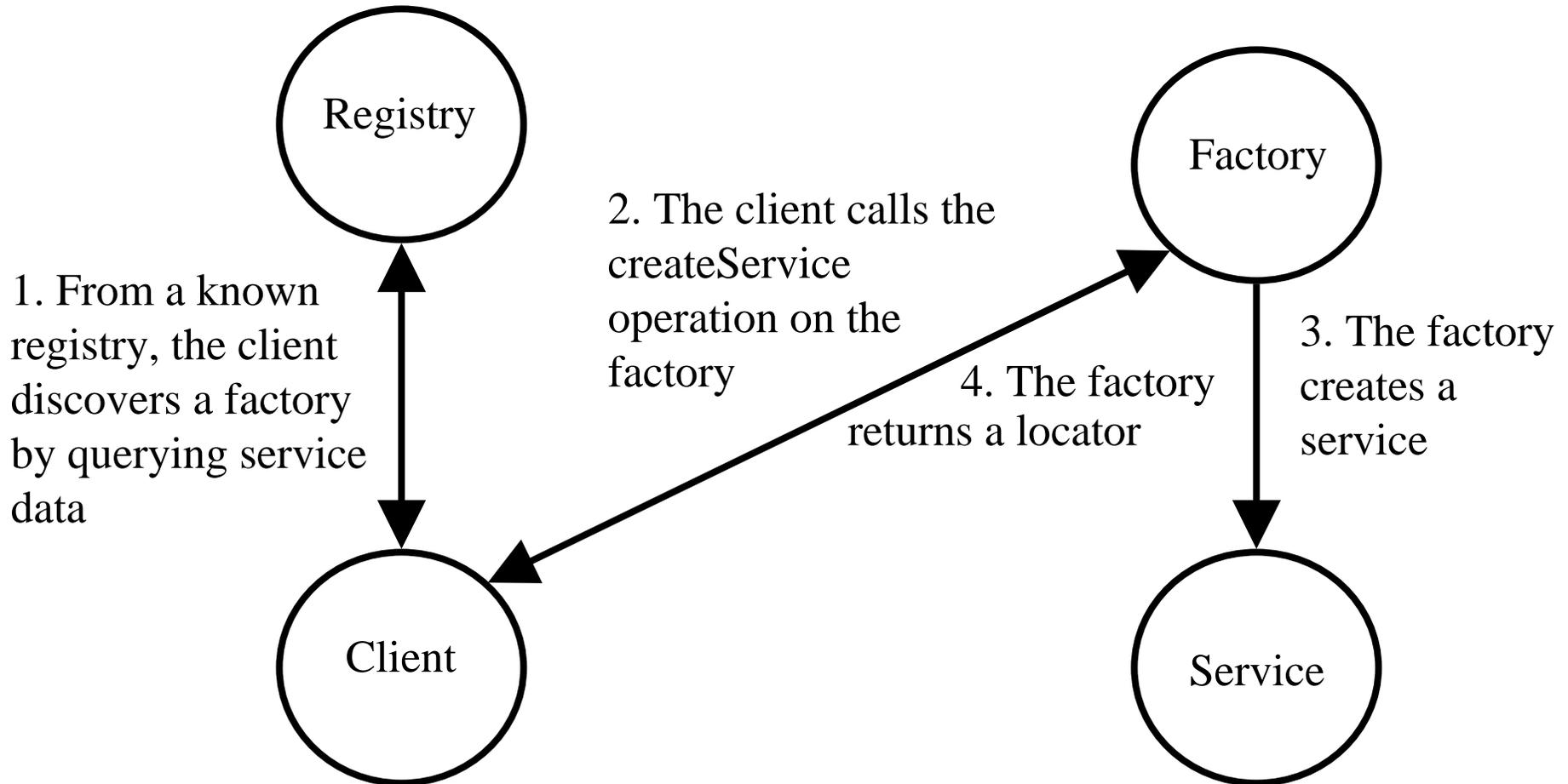
A Service Creation Scenario



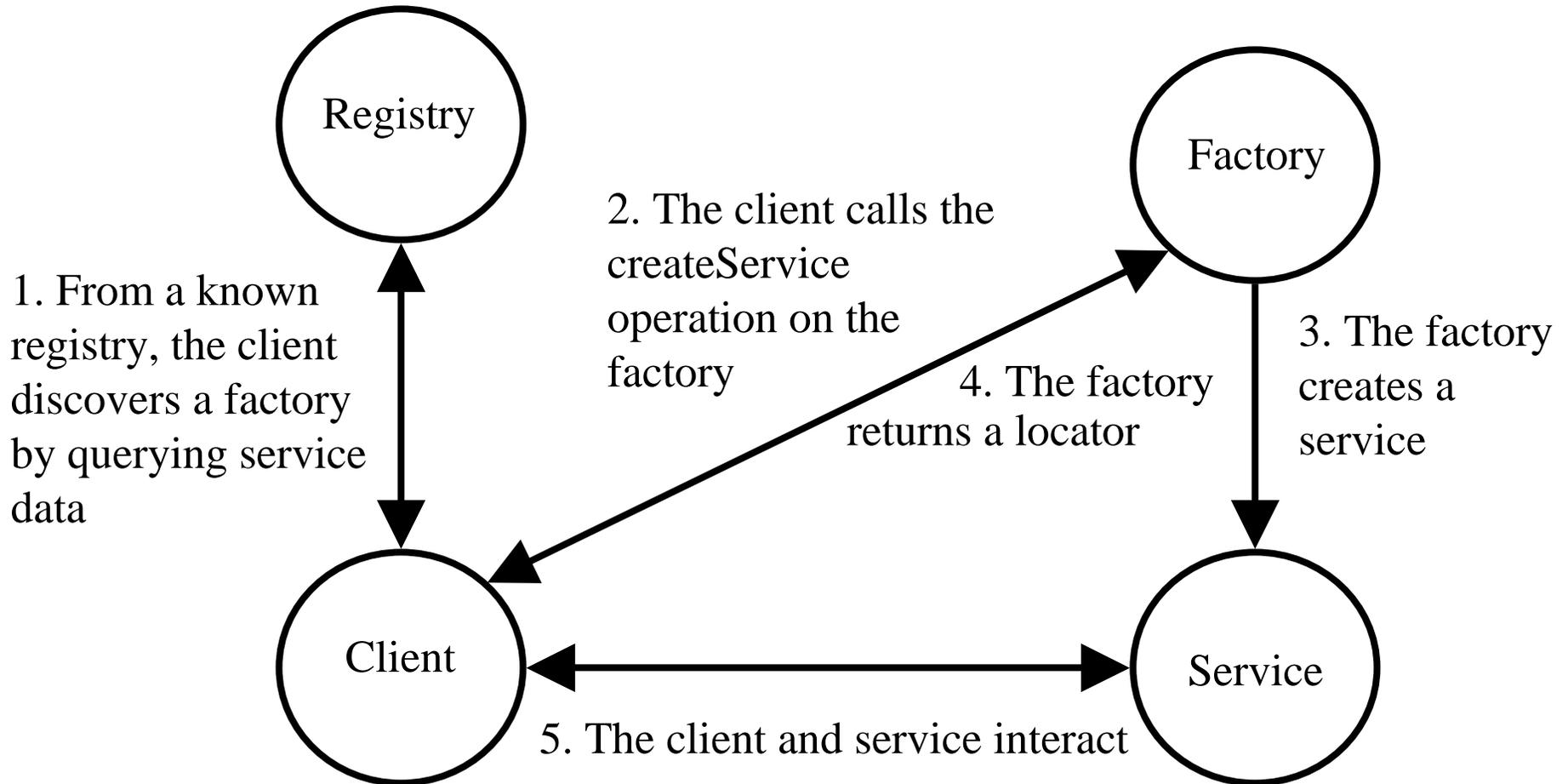
A Service Creation Scenario



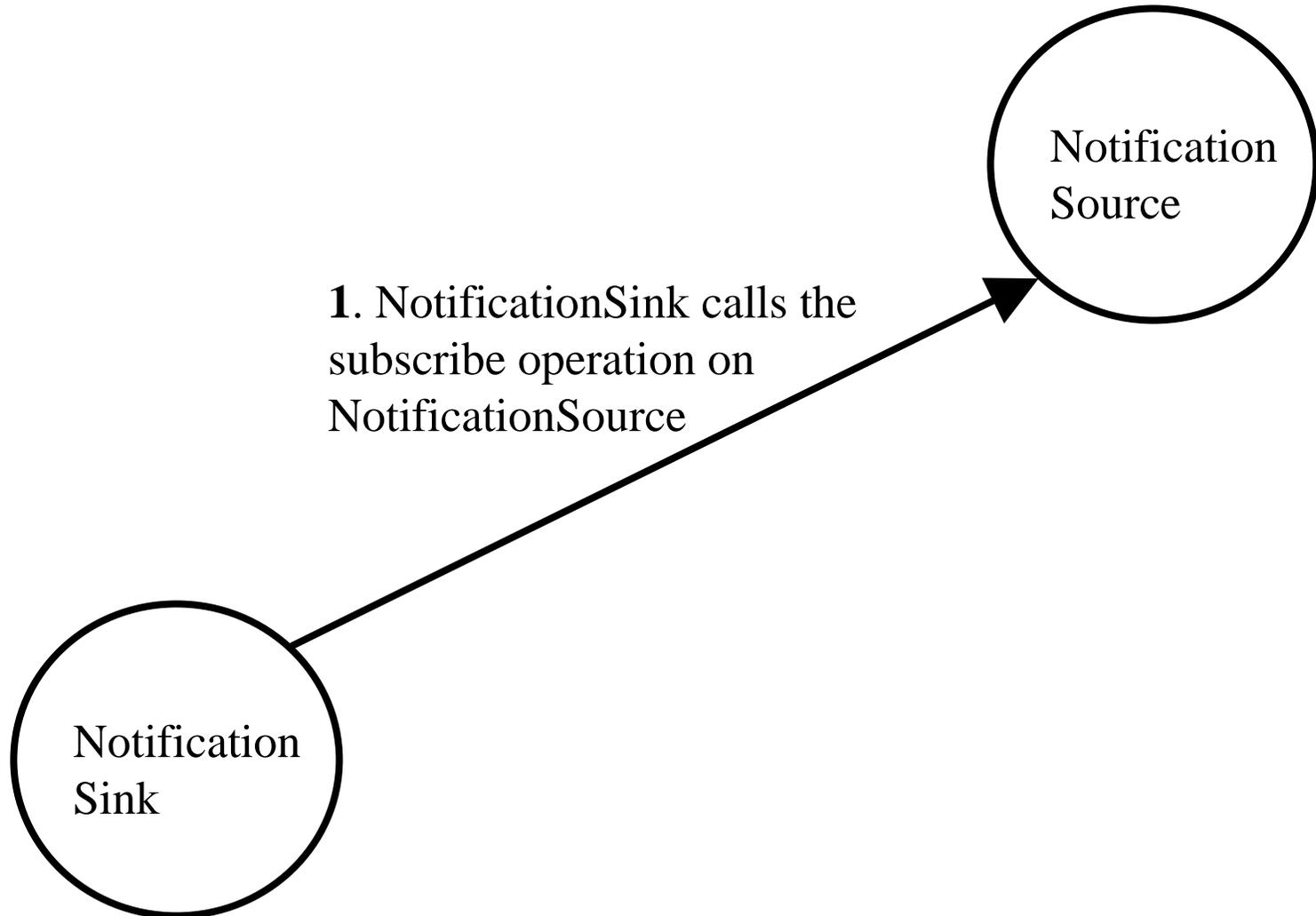
A Service Creation Scenario



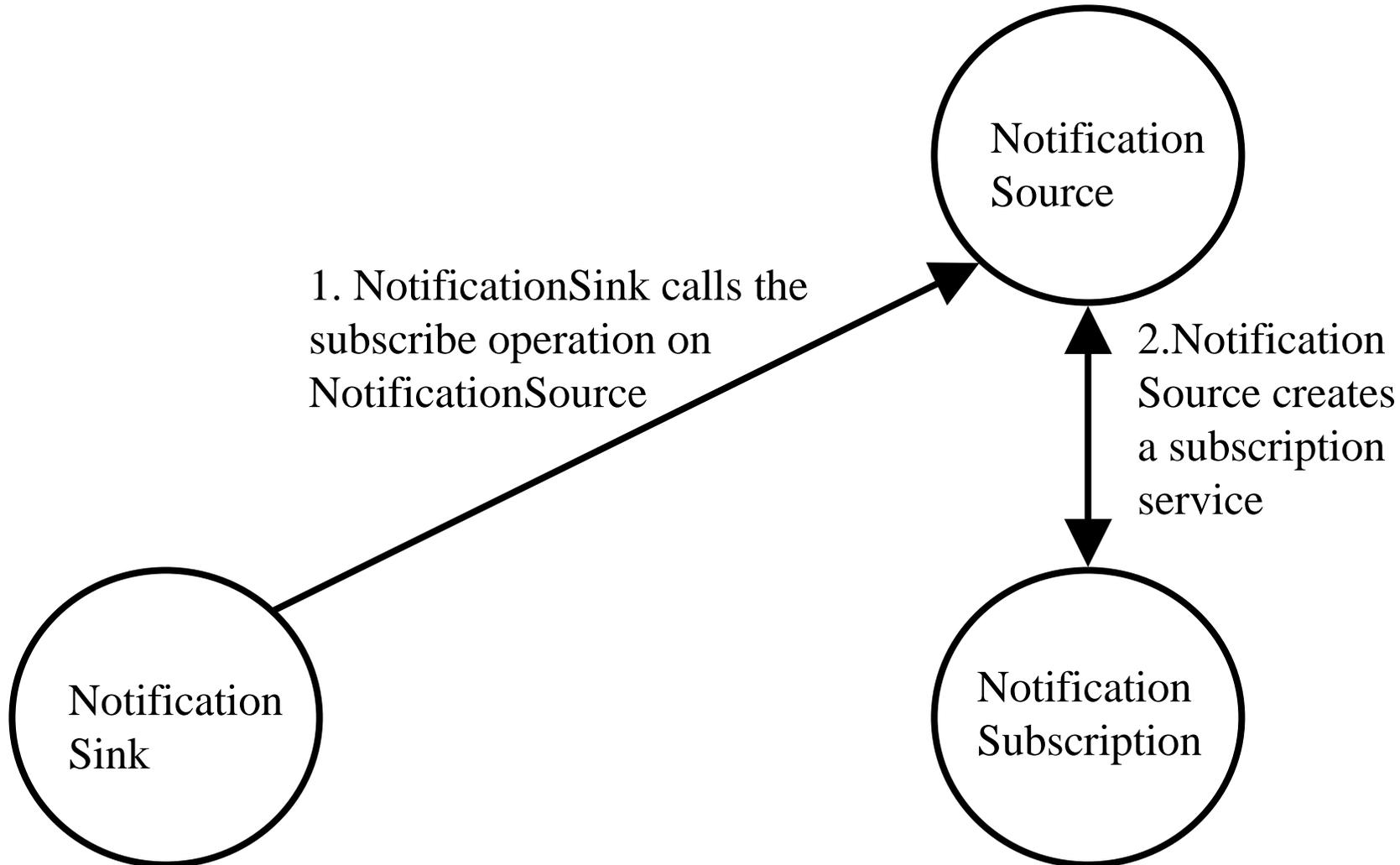
A Service Creation Scenario



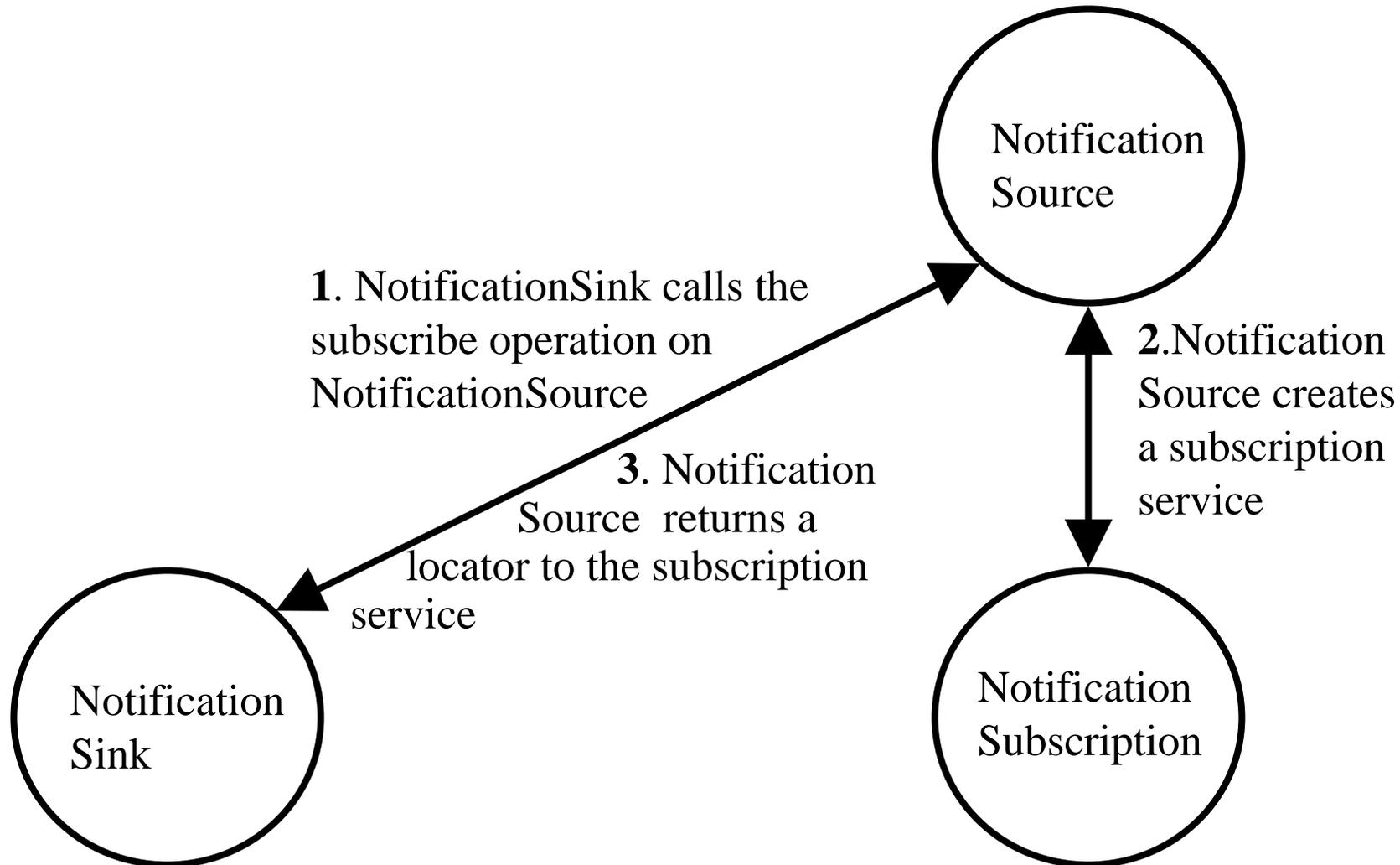
A Notification Scenario



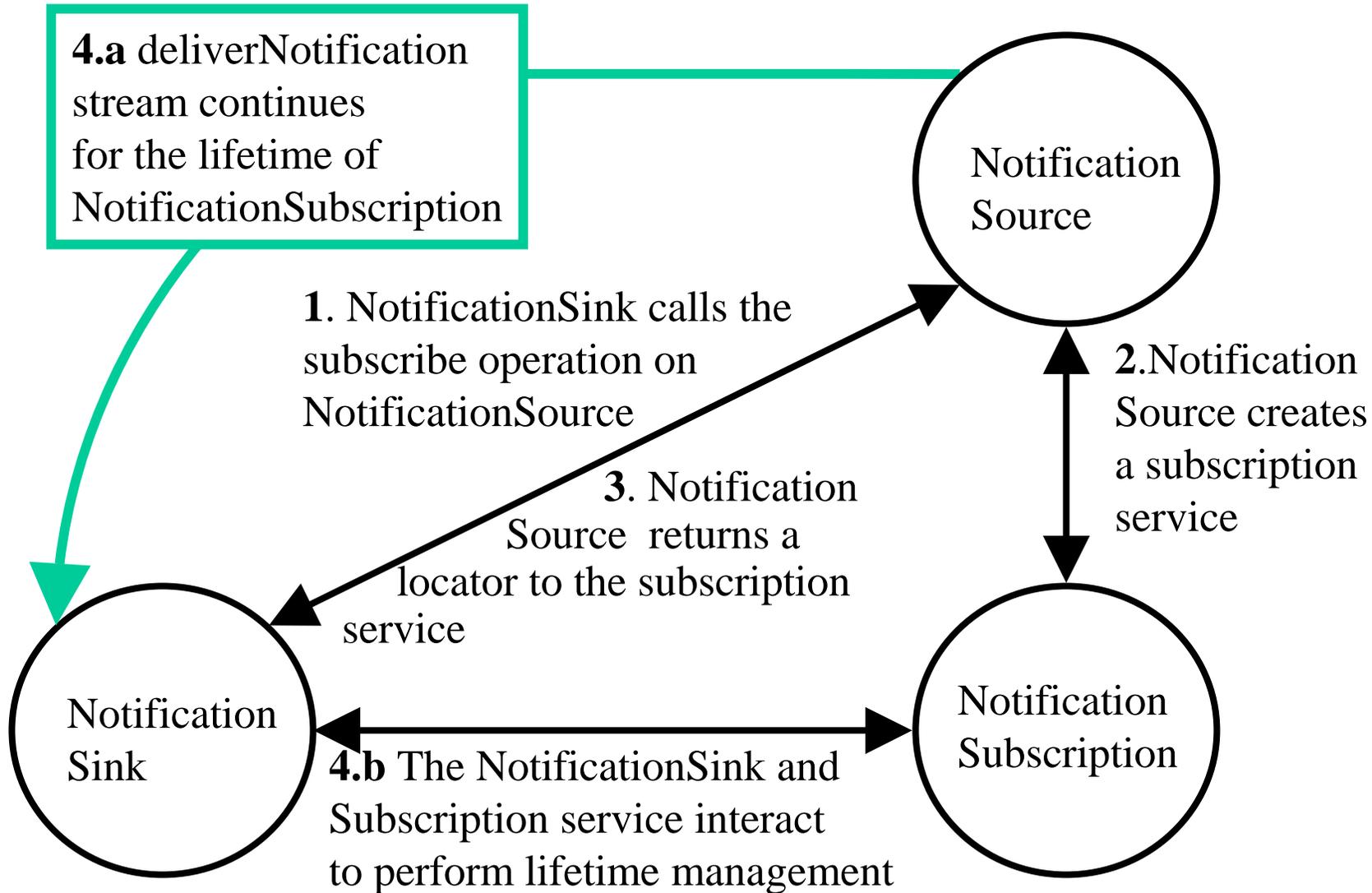
A Notification Scenario



A Notification Scenario



A Notification Scenario



GT3 Core: Security Infrastructure

- Transport Layer Security/Secure Socket Layer (TLS/SSL)
 - To be deprecated
- SOAP Layer Security
 - Based on WS-Security, XML Encryption, XML Signature
- GT3 uses X.509 identity certificates for authentication
- It also uses X.509 Proxy certificates to support delegation and single sign-on, updated to conform to latest IETF/GGF draft

WS-Secure Conversation

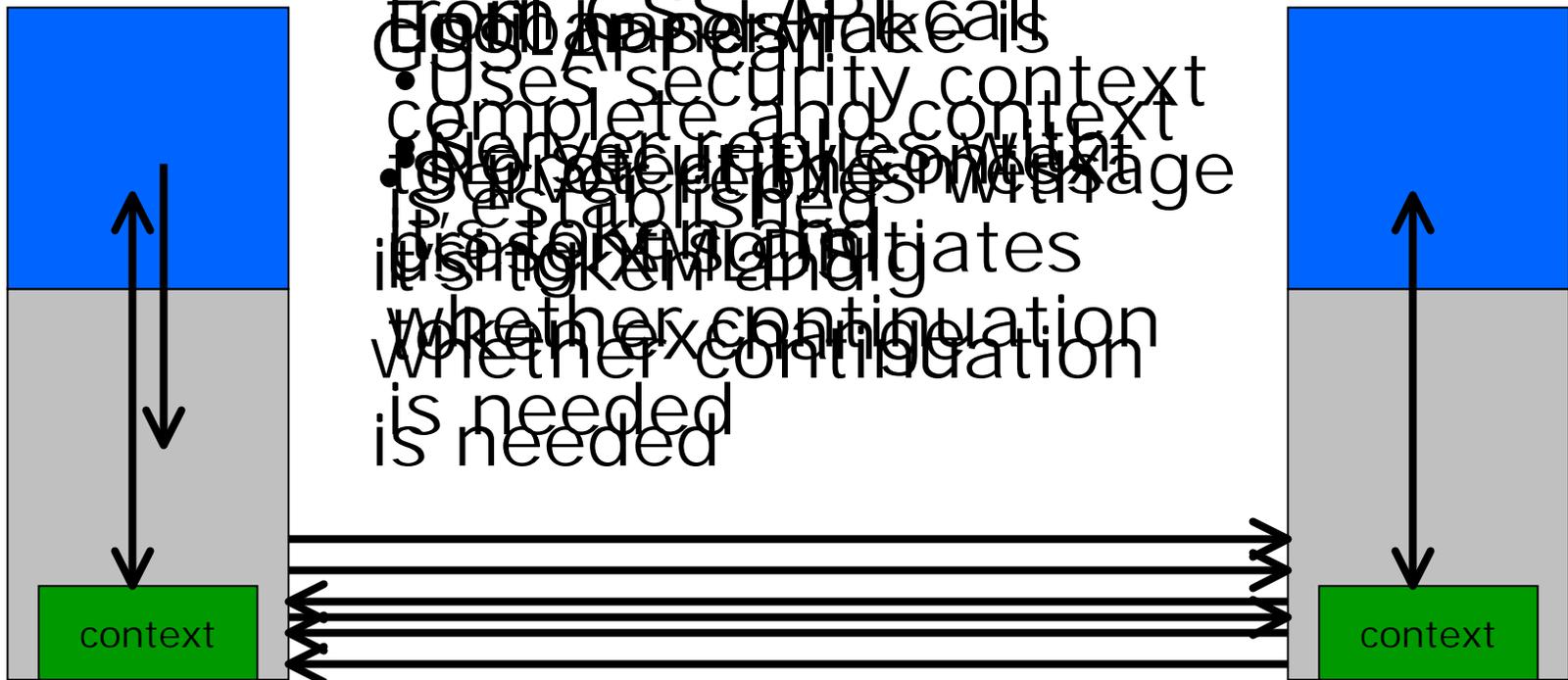
- Used to establish a security context between two end points
- Context establishment is done at the message layer
 - Allows for SOAP intermediaries
- Initial implementation based on Grid Security Infrastructure
 - TLS handshake is used for mutual authentication and context establishment
 - Supports the ability to delegate X.509 credentials
 - Uses GSS-API

WS-Secure Conversation

- Calls
- Calls
- continueContextExchange
- contextExchange
- applyWithOutput
- fromSOAPCall
- uses security context
- complete and context
- never replies message
- established with
- initiates
- whether continuation
- is needed

client

server



GT3 Core: Grid Service Container

Includes the OGSI Implementation, security infrastructure and system-level services, plus:

- Service activation, deactivation, construction, destruction, etc.
- Service data element placeholders that allow you to dynamically fetch service data values at query time
- Evaluator framework (supporting ByXPath and ByName notifications and queries)
- Interceptor/callback framework (allows one to intercept certain service lifecycle events)

GT3 Core: Hosting Environment

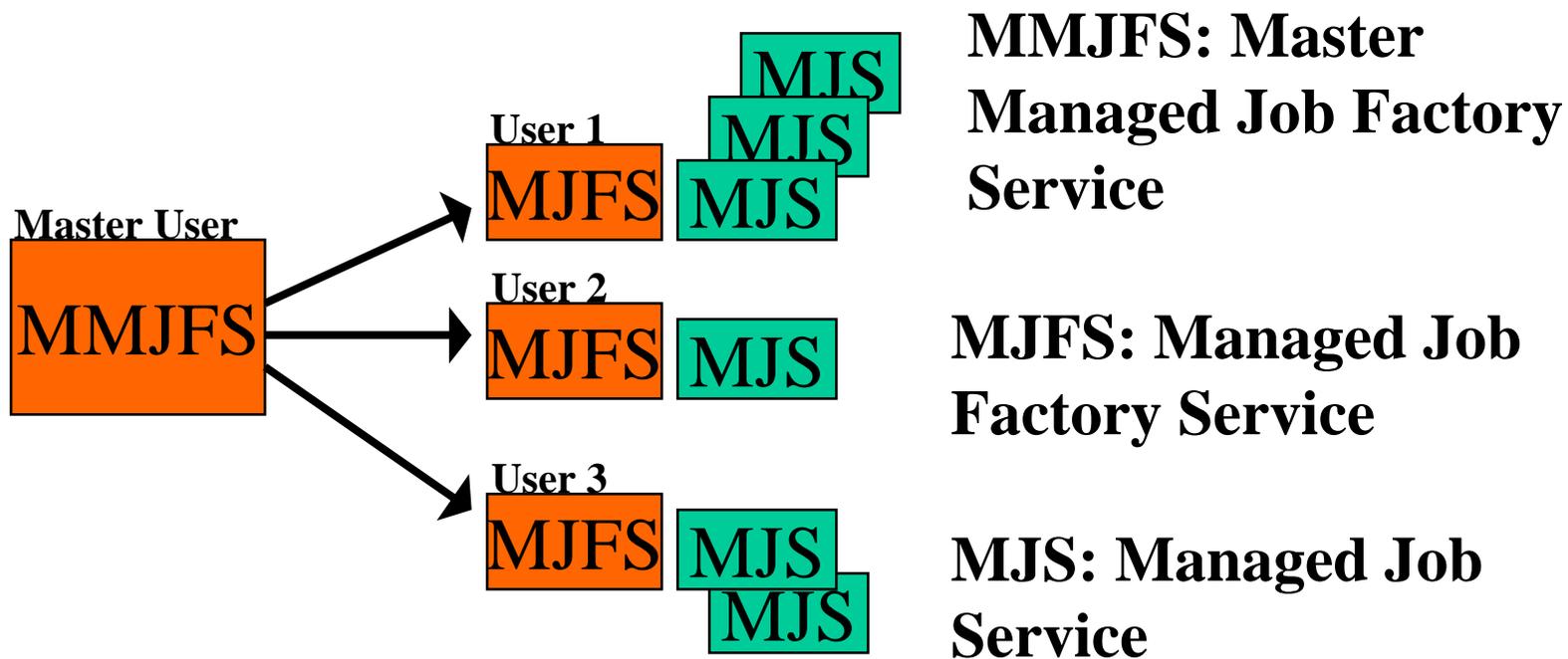
GT3 currently offers support for four Java Hosting Environments:

- Embedded
- Standalone
- Servlet
- EJB

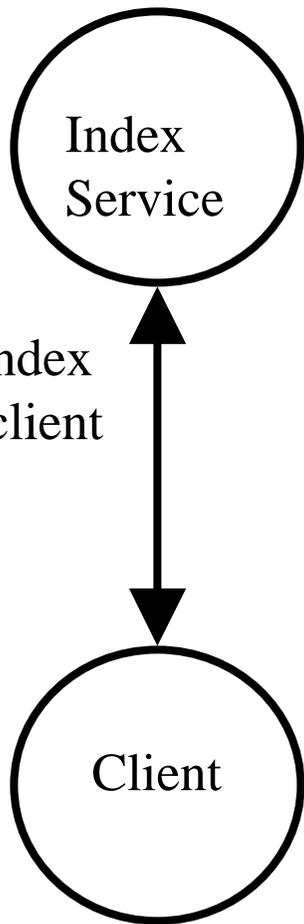
GT3 Base Services

GT3 Base: Resource Management

- GRAM Architecture rendered in OGSA
- The MMJFS runs as an unprivileged user, with a small highly-constrained setuid executable behind it.

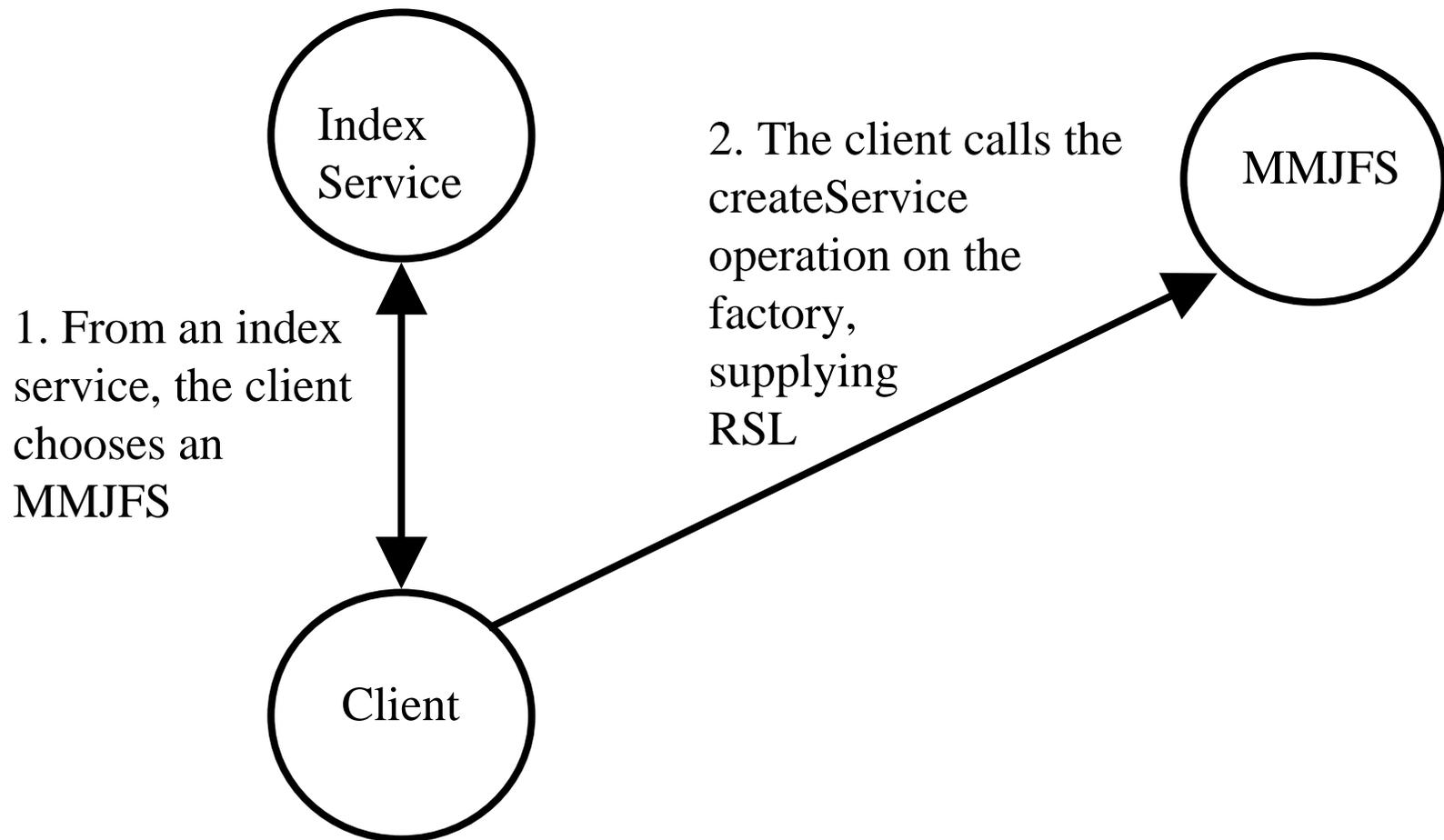


GRAM Job Submission Scenario

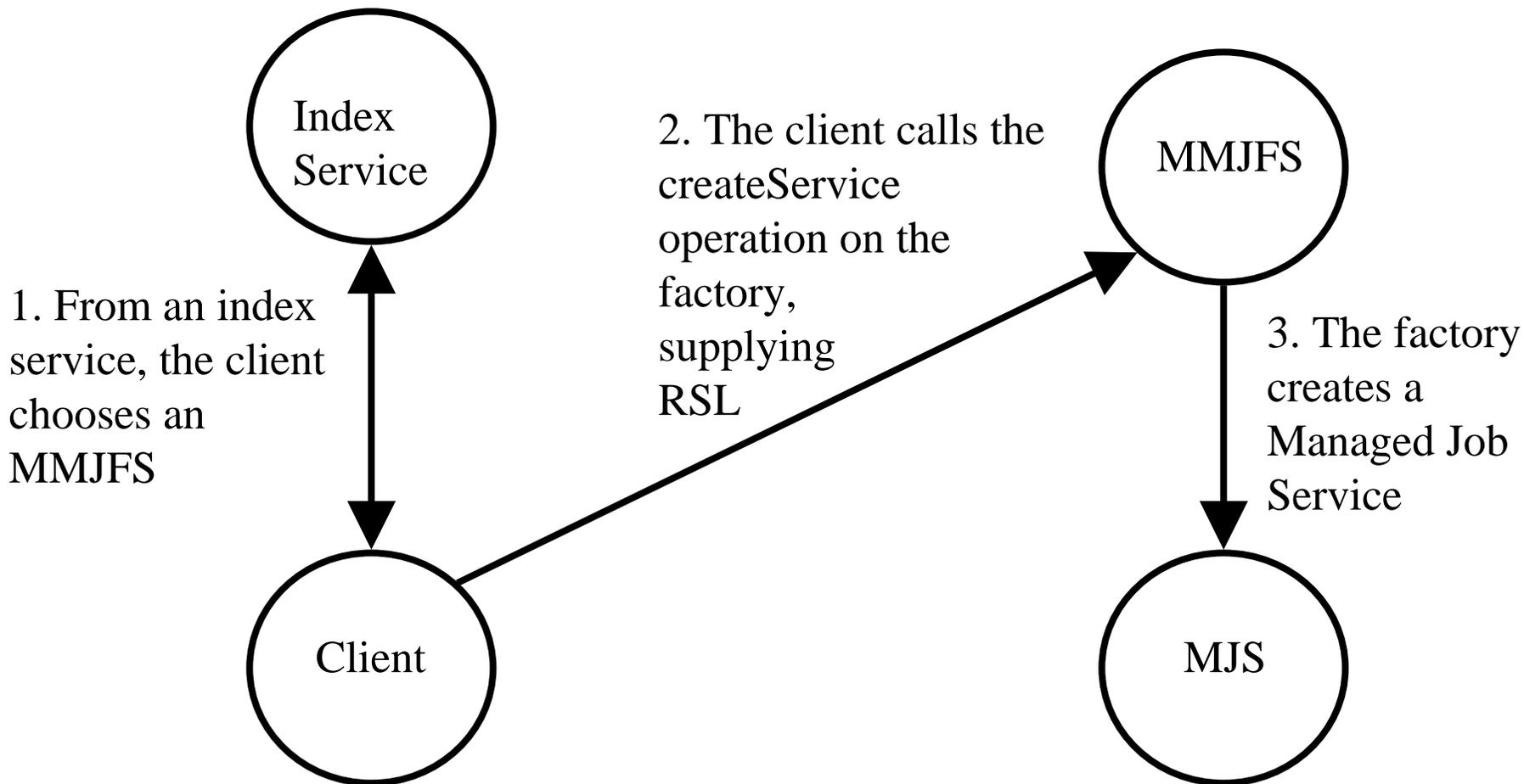


1. From an index service, the client chooses an MMJFS

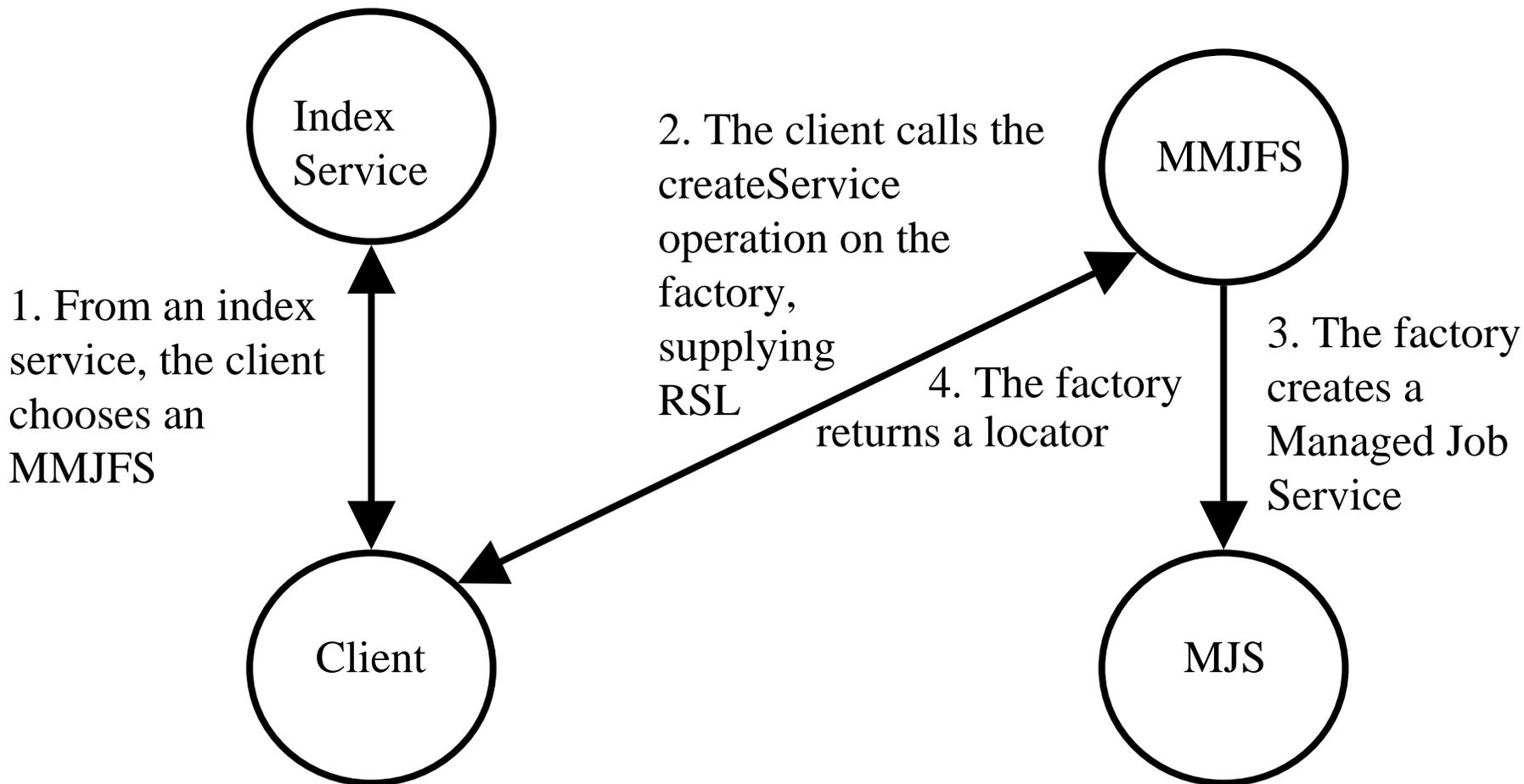
GRAM Job Submission Scenario



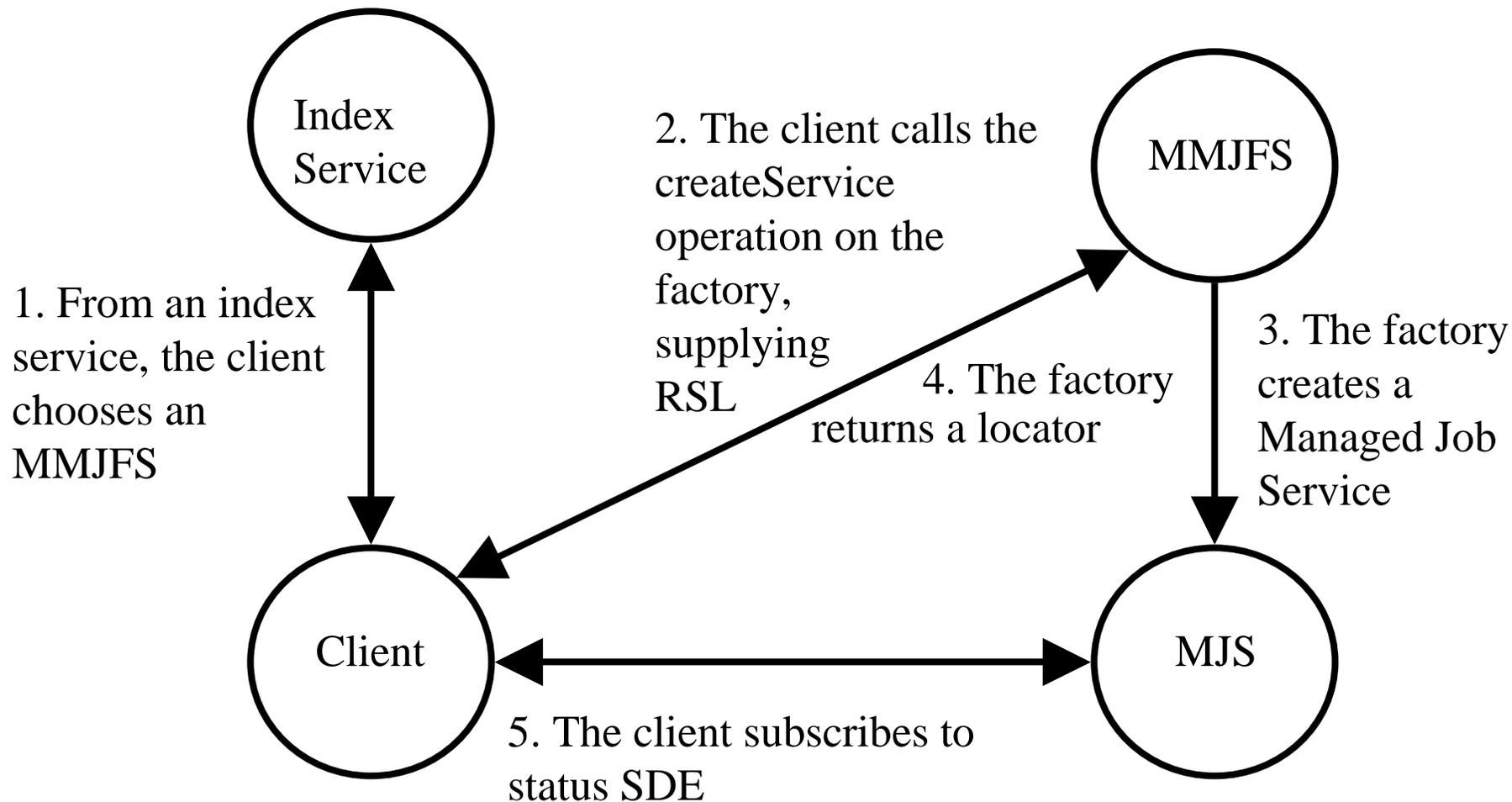
GRAM Job Submission Scenario



GRAM Job Submission Scenario



GRAM Job Submission Scenario

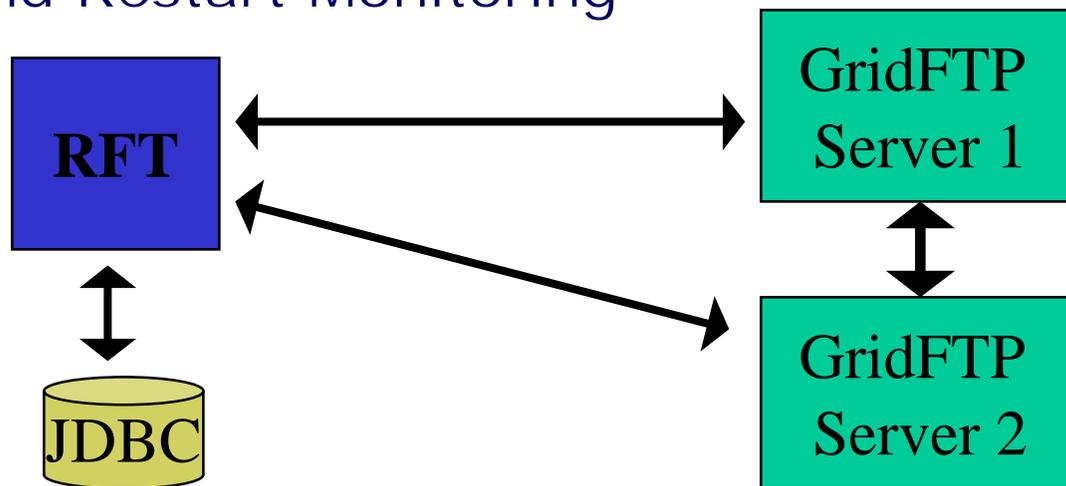


GT3 Base: Information Services

- Index Service as Caching Aggregator
 - Caches service data from other grid services
- Index Service as Provider Framework
 - Serves as a host for service data providers that live outside of a grid service to publish data

GT3 Base: Reliable File Transfer

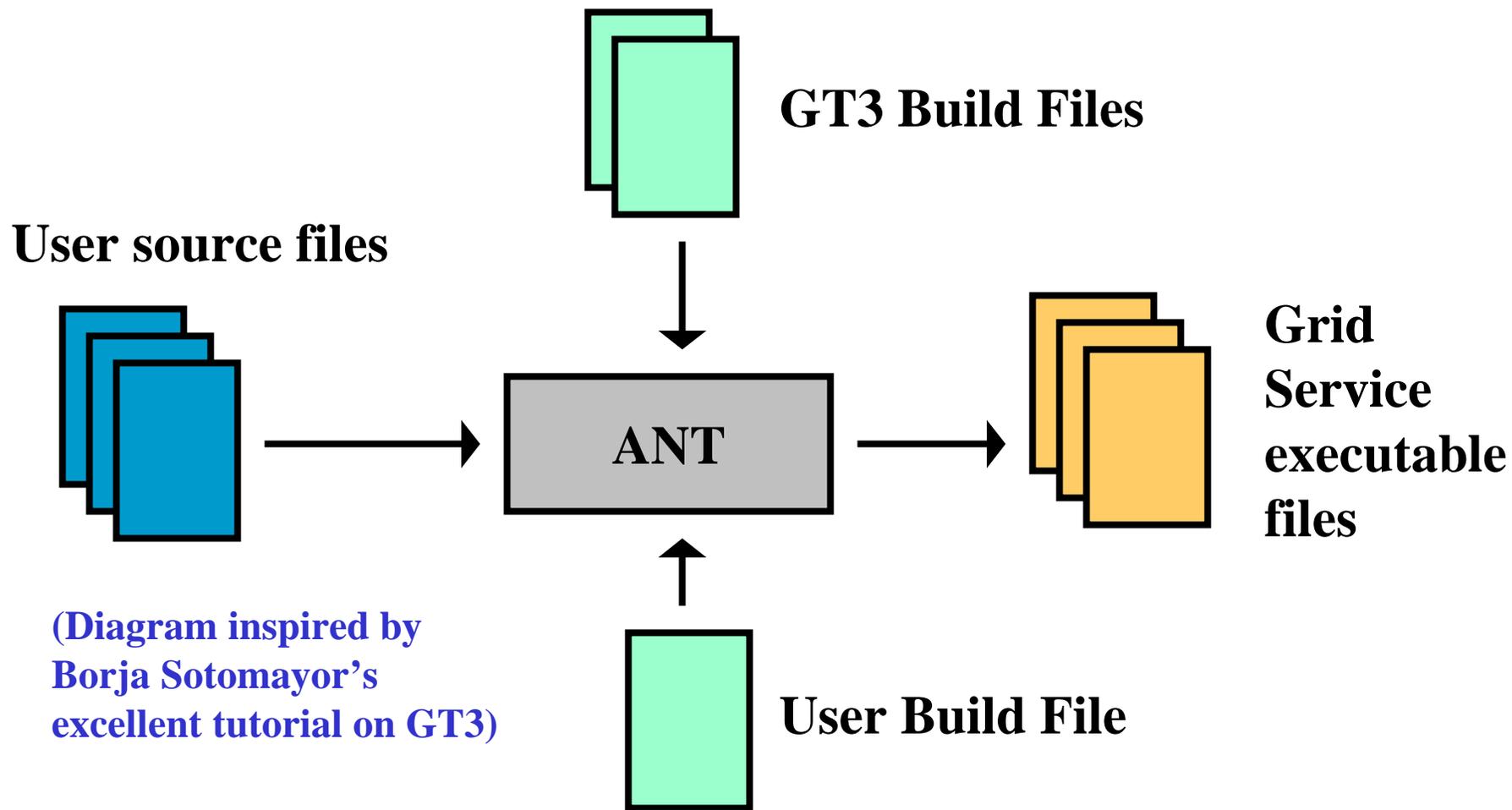
- Reliably performs a third party transfer between two GridFTP servers
- OGSI-compliant service exposing GridFTP control channel functionality
- Recoverable Grid Service
 - Automatically restarts interrupted transfers from the last checkpoint
- Progress and Restart Monitoring



GT3 User-Defined Services

- GT3 can be viewed as a Grid Service Development Kit that includes:
 - Primitives designed to ease the task of building OGSI-Compliant Services
 - Primitives for provisioning security
 - Base services that provide an infrastructure with which to build higher-level services

GT3 User-Defined Services (cont.)



(Diagram inspired by Borja Sotomayor's excellent tutorial on GT3)

Future Directions of GT

- Standardization of container model
- Development of lightweight container/api
- Adding rich support for queries
- Further refinements of Base Service designs
- Pushing on standardizing at a higher level than OGSI

pyOGSI

pyOGSI

- Developing a full Open Grid Services Architecture implementation
- ZSI library is used for SOAP parsing
- Twisted application server for the hosting environment
 - Also support standalone Grid Services

pyOGSI Client Bindings

- Builds on the automatic wsdl2python generator we've built to use with ZSI
- Adds support for transparent GSH to GSR mapping
- Notification Sink support
 - Requires ability to support service invocation on the client
 - Uses SimpleHTTPServer
- Security support automatically added into generated code

pyOGSI Security

- Support normal TLS protocol (https)
- Support GSI enable TLS protocol (httpg)
- Adding support for WS-Security message level security

WS-Security

- Implementing the XML Digital Signature standard
 - Only doing the required parts to implement signed SOAP messages
 - Implementing the enveloped transform
 - Canonicalization is already part of PyXML
- May implement part of the XML Encryption standard if needed

pyOGSI Server Support

- Support both a OGSI “hosting environment” and standalone OGSI servers
- Standalone server will:
 - Use the standard Python SimpleHTTPServer
 - Provide a super-class that contains all of the necessary Grid Service code
 - Lifecycle management
 - Service Data/Notification
 - Security
 - Factory

pyOGSI Server Support (cont.)

- Will provide a container to host Grid Services
 - Based on the Twisted project (<http://twistedmatrix.com/products/twisted>)
 - Provides a base-class to encapsulate all required Grid Service functionality
 - Will support exposing legacy Fortran/C/C++ codes as Python OGSI components
 - Will provide automatic server stub generation from WSDL/GSDL document

pyOGSI Status

- Client bindings currently available from CVS
 - Working to integrate the common Web Service code back into the ZSI project
 - Still working on the border cases with complex type encoding
- Server side support is under development
 - Working on automated server stub generation
 - Currently have working Web Service code, working on adding required Grid Service port types
- Code released under the standard BSD license

Acknowledgement

- The Python Grid Service Work is funded by the U.S. Department of Energy Office of Science
- Thanks to the Globus project
- More information can be found at
 - <http://www-itg.lbl.gov/gtg/projects/pyOGSI/>
 - <http://www.globus.org/>
 - <http://www.globus.org/ogsa>
 - <http://www.ggf.org/>
- Bug submission
 - <http://www-itg.lbl.gov/bugzilla/>
 - <http://bugzilla.globus.org>
- Email:
 - krjackson@lbl.gov