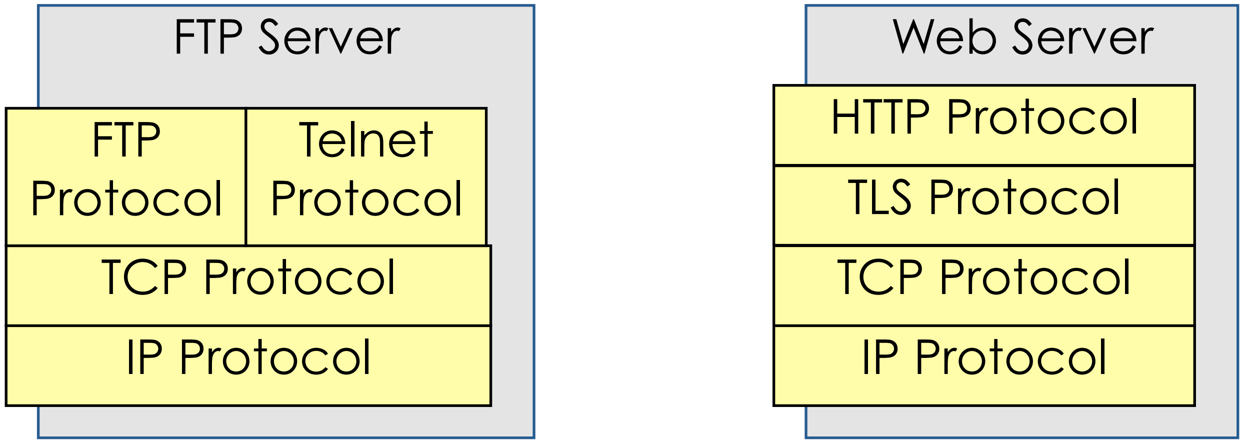
Grid Computing

Preliminary Definitions (1)

- **Resource**
  - An entity that may be shared
    - CPU, storage, data, software,…
  - Not necessarily a physical entity
    - Filesystem, bandwidth, thread pool…
  - Defined in terms of interfaces and capabilities
    - Open/close/read/write define the access methods to a filesystem
    - Copy/delete/move/create/cat define the methods to manipulate data
- **Protocol**
  - A formal description of messages format and a set of rules to exchange messages
    - Messages allow two or more resources to communicate
    - Rules may define a sequence of message exchanges
    - Message may change resources status and/or behavior
  - A good protocol does a single
    - Filesystem, bandwidth, thread pool…
  - Defined in terms of interfaces and capabilities (APIs)
    - Open/close/read/write define the access methods to a filesystem
    - Copy/delete/move/create/cat define the methods to manipulate data

Preliminary Definitions (2)

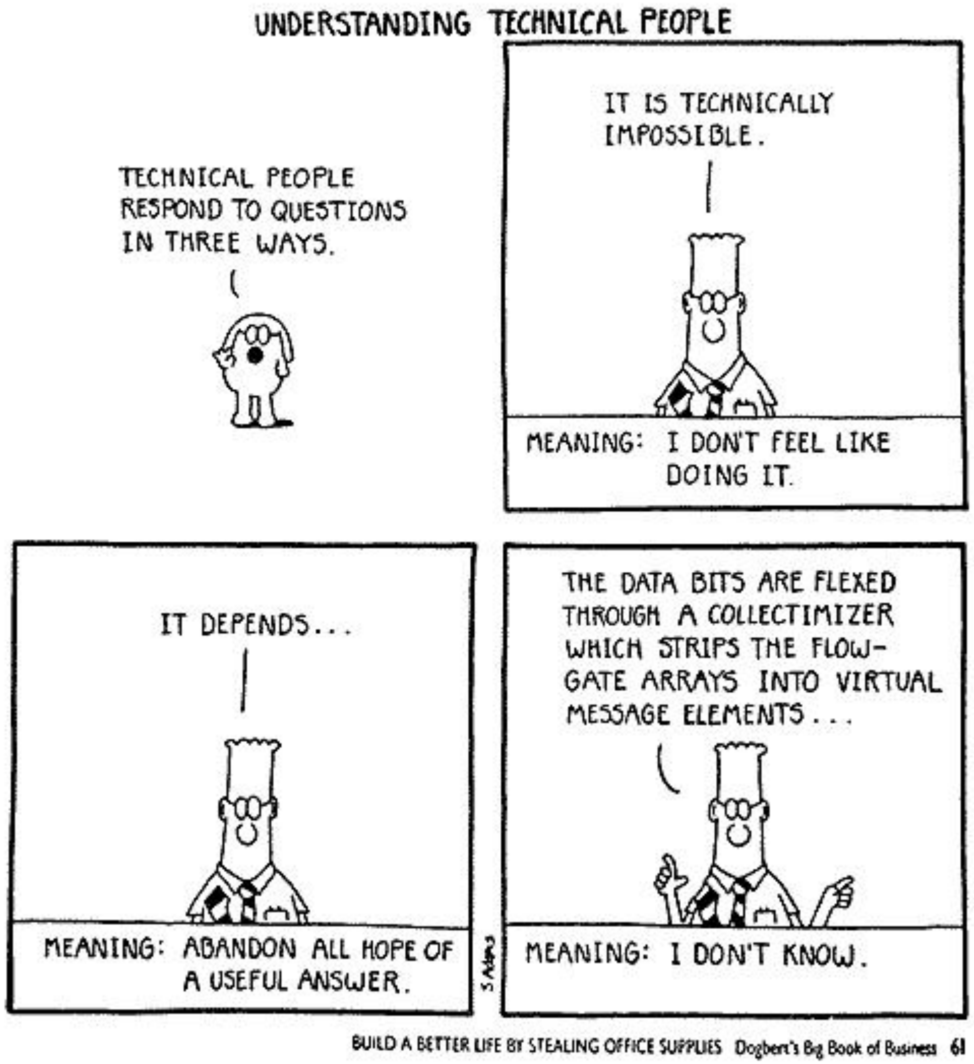
- **Service**
  - A server-side protocol implementation providing a set of capabilities
    - The protocol defines the interactions between a client and a server
    - A server implementing a protocol is the service
    - Every service needs a protocol to implement
  - A service can implement more than one protocol, but good services expose just one
  - Examples
    - FTP servers (ftp://)
    - Web servers (http://)
    - Mail servers (pop or imap)



Preliminary Definitions (3)

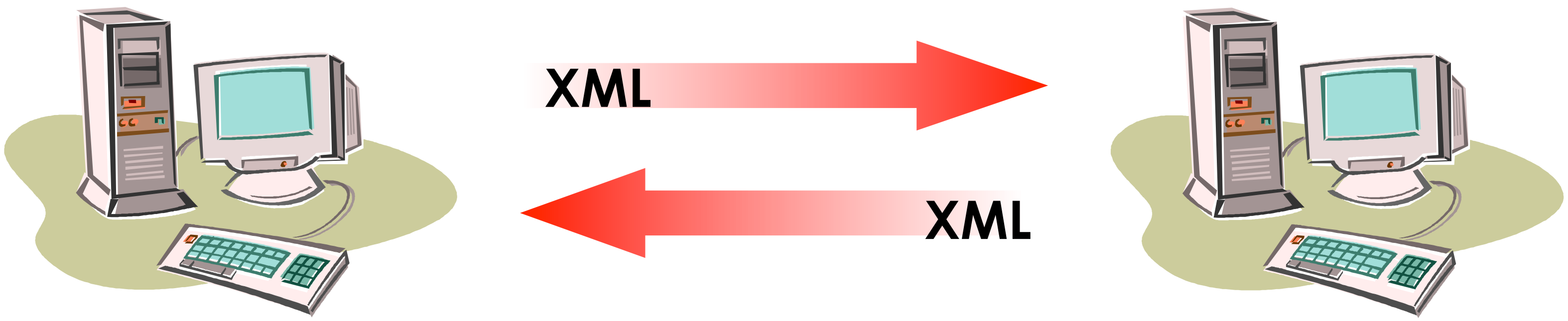
- **API (Application Programming Interface)**
  - Specifies a set of routines to facilitate the development of applications
    - Definition, NOT implementation
    - An API may have several implementations (e.g., MPI)
    - An API specifies behaviors and interfaces
  - APIs may be language-oriented
    - Mapping specification to language constructs
    - Name, number, order and type of parameters
- **SDK (Software Development Kit)**
  - A particular implementation of an API
  - Provides libraries and tools
  - Given an API we can have multiple SDK
    - e.g., LAM/MPI, MPICH, HP-MPI, Open MPI

- Standard API/SDK are important
  - They enable applications portability
  - But without standard protocols, interoperability is difficult
- Standard protocols are important
  - They enable applications interoperability
  - Programs using different APIs for the same protocol can communicate
    - Clients do not need to know server's API.
  - They allow shared infrastructures


UNDERSTANDING TECHNICAL PEOPLE
TECHNICAL PEOPLE RESPOND TO QUESTIONS IN THREE WAYS.
IT IS TECHNICALLY IMPOSSIBLE.
MEANING: I DON'T FEEL LIKE DOING IT.
IT DEPENDS...
MEANING: ABANDON ALL HOPE OF A USEFUL ANSWER.
THE DATA BITS ARE FLEXED THROUGH A COLLECTIMIZER WHICH STRIPS THE FLOW-GATE ARRAYS INTO VIRTUAL MESSAGE ELEMENTS...
MEANING: I DON'T KNOW.
BUILD A BETTER LIFE BY STEALING OFFICE SUPPLIES Dogbert's Big Book of Business 61

Example

- A web service is a service available on the Internet
- It allows creation of client/server applications.
- Platform and language independent protocol based on XML.
- Most use HTTP for transporting messages
- Lend themselves naturally to build *loosely coupled* distributed systems.



Computer A

Perl

Windows 2000

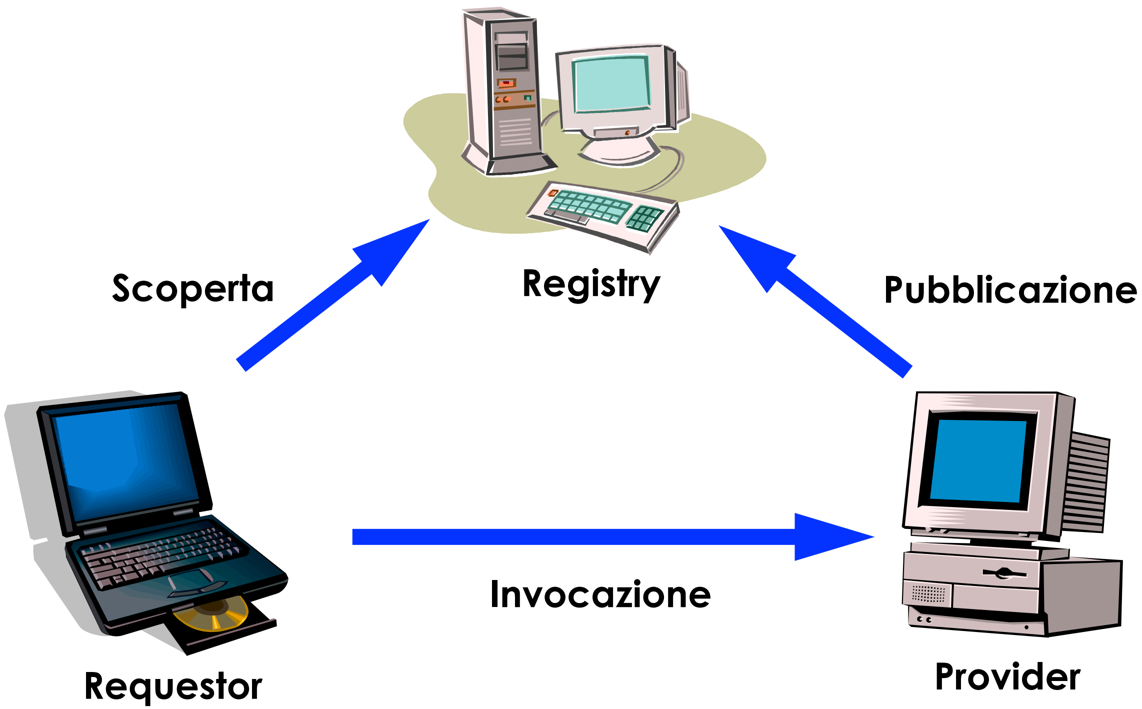
Computer B

Java

Linux

Roles

- **Service Provider**
  Implements the service and make it available on the Internet
- **Service Requestor**
  Service consumers use existing services opening a network connection, sending XML requests and receiving XML responses
- **Service Registry**
  The service registry provides a central point where service providers can publish their services and service requestors can look for existing services

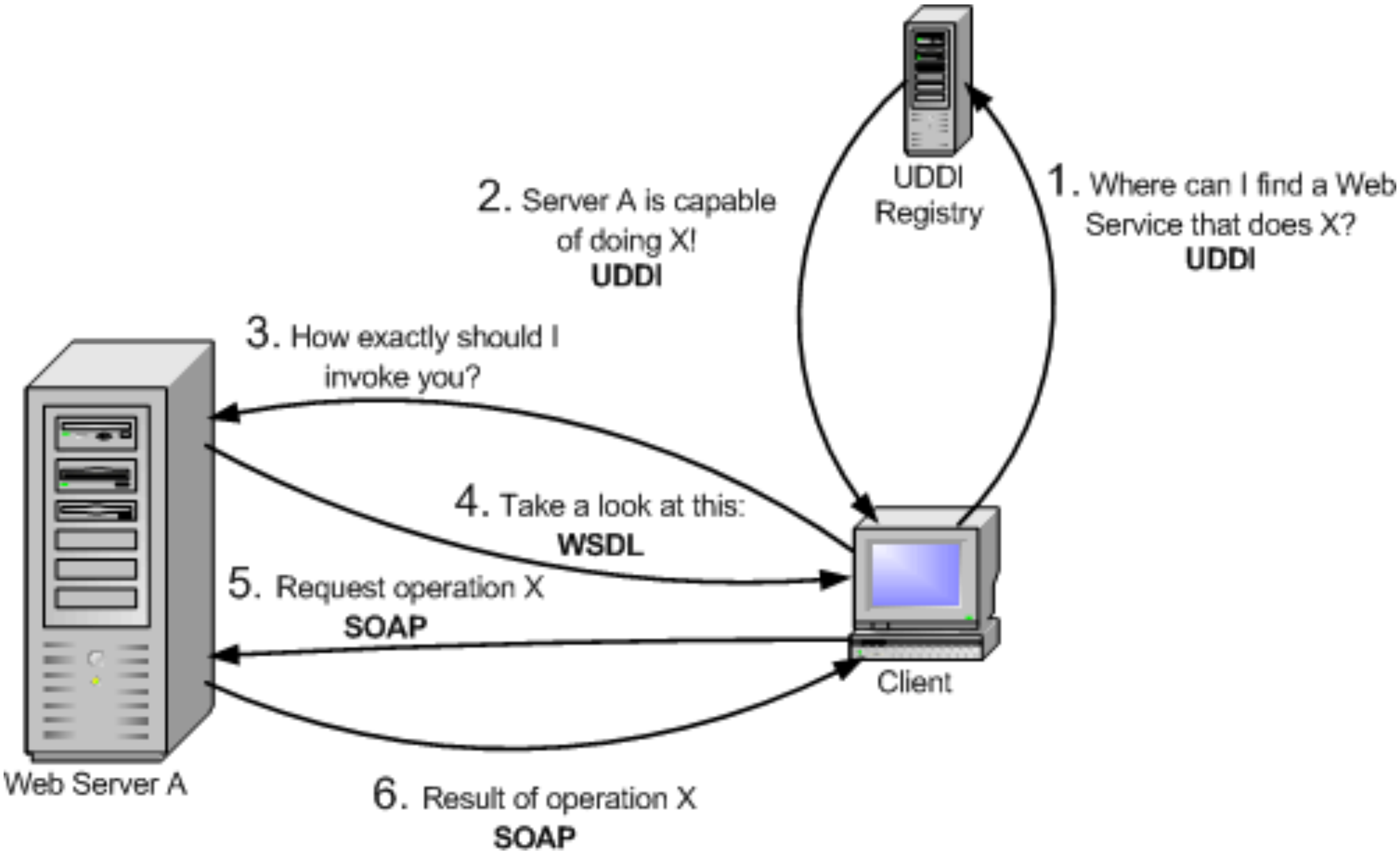


Protocols

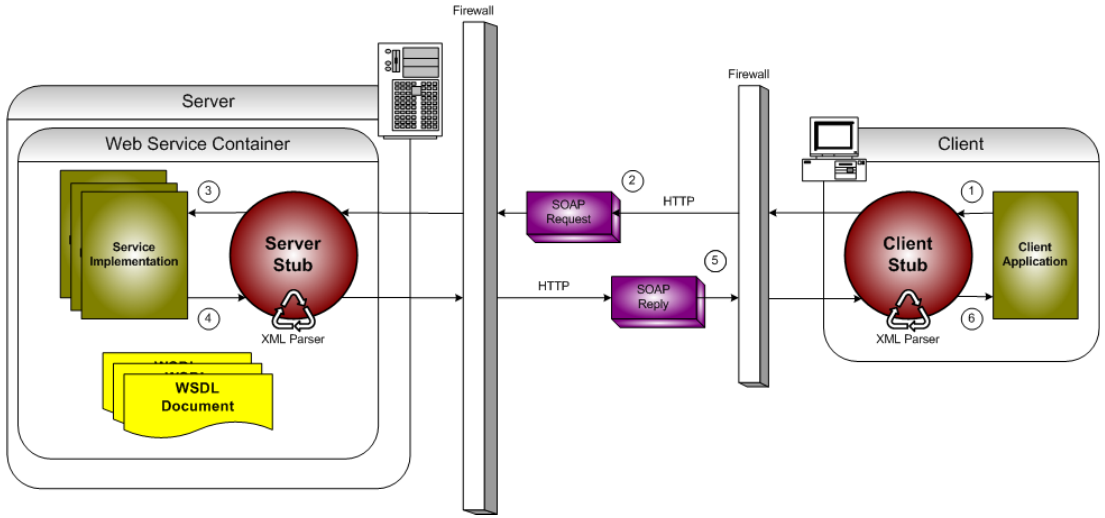
Discovery	**UDDI**
Description	**WSDL**
Invocation	**SOAP,XML-RPC**
Transpost	**HTTP,FTP**

The Web Services Architecture is specified and standardized by the World Wide Web Consortium, the same organization responsible for XML, HTML, CSS, etc.

Protocols in Action



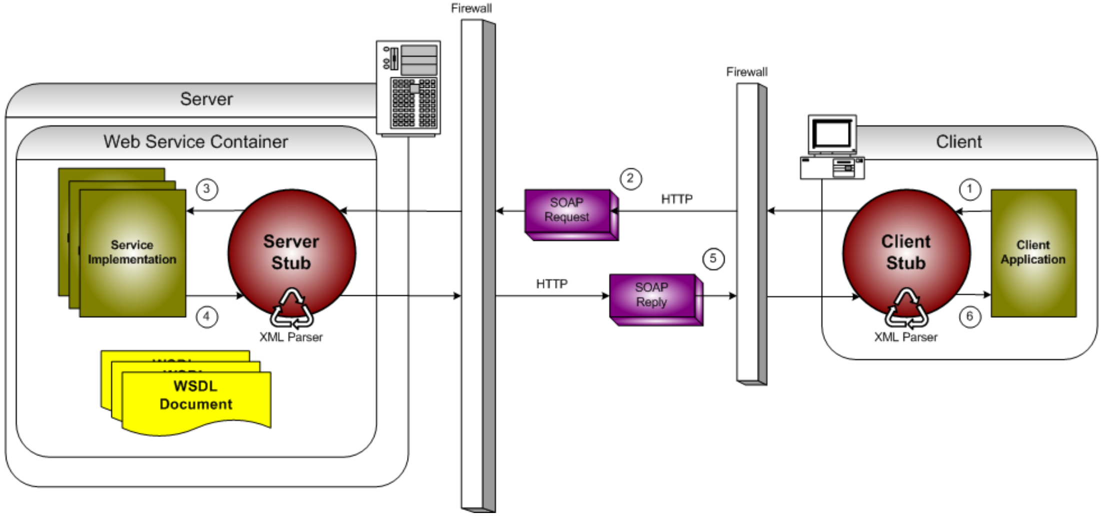
Web Service Invocation (1)



1. Client will call the client stub. The client stub will turn this 'local invocation' into a proper SOAP request. This is often called the *marshaling* or *serializing* process.

2. The SOAP request is sent over a network using the HTTP protocol. The server receives the SOAP requests and hands it to the server stub. The server stub will convert the SOAP request into something the service implementation can understand (this is usually called *unmarshaling* or *deserializing*)..

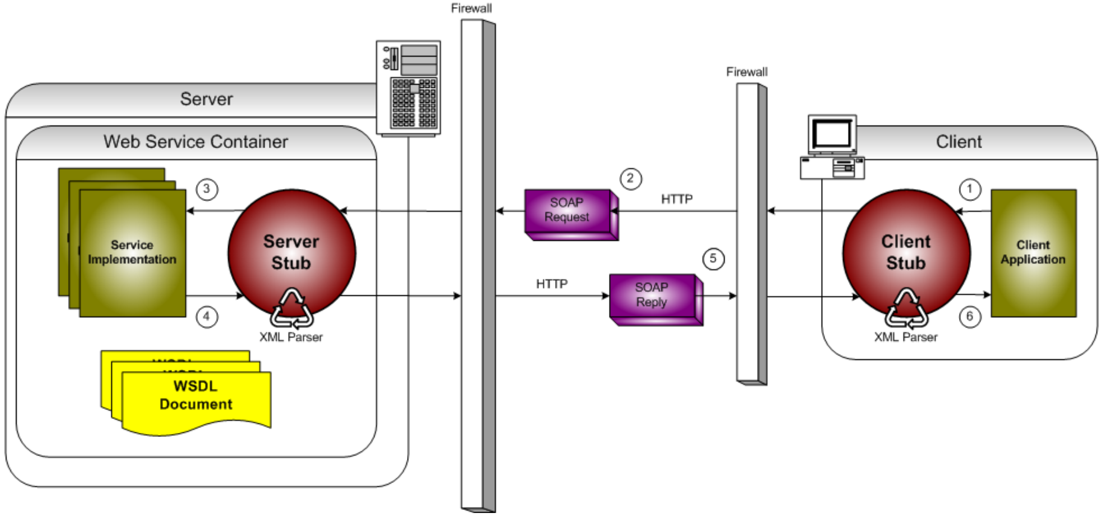
Web Service Invocation (2)



4. Once the SOAP request has been deserialized, the server stub invokes the service implementation, which then carries out the work it has been asked to do.

5. The result of the requested operation is handed to the server stub, which will turn it into a SOAP response.

Web Service Invocation (3)



6. The SOAP response is sent over a network using the HTTP protocol. The client stub receives the SOAP response and turns it into something the client application can understand.

7. Finally the application receives the result of the Web Service invocation and uses it.


BACK
TO THE FUTURE

Why?

- Large-scale resource sharing
  - Spanning administrative boundaries
- Multi-institutional environment
  - Dynamicity
  - Geographical distribution

- Grid computing is all about achieving performance and throughput by pooling and sharing resources on a local, national or world-wide level

Reading Assignments

- C. Kesselman, et al.,
  The Anatomy of the Grid: Enabling Scalable Virtual Organizations
  Internationall Journal of Supercomputing Applications, pp. 1-25, 2001.

  http://www.globus.org/alliance/publications/papers/anatomy.pdf

- I. Foster, et al.,
  The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration
  Globus Research, Work-in-Progress 2002.

  http://www.globus.org/alliance/publications/papers/ogsa.pdf

- Links provided at:

  http://www.cli.di.unipi.it/doku/doku.php/magistraleinformaticanetworking/cpa/start

Power Grid

Elements of Grid Computing

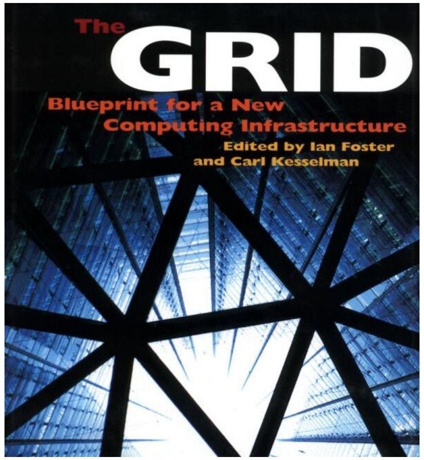
- Resource sharing
  - Computers, data, storage, sensors, networks, …
  - Sharing always conditional: issues of trust, policy, negotiation, payment, …
- Coordinated problem solving
  - Beyond client-server: distributed data analysis, computation, collaboration, …
- Dynamic, multi-institutional **virtual organizations**
  - Community overlays on classic org structures
  - Large or small, static or dynamic

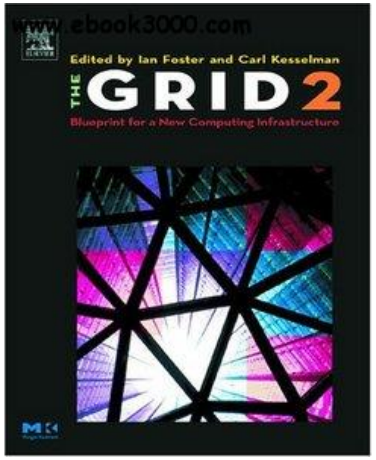
Virtual Organizations

- Dynamic set of individuals and/or institutions defined by a shared goal and a set of sharing rules
  - Example: several partners in a research project
- May vary in size, scope, duration and structure
  - Example: class students for cooperative lecture writing
    - Goal? Rules?
  - Example: industrial consortium building a new aircraft
    - Goal? Rules?
- The sharing is highly controlled, with resource providers and consumers defining clearly and carefully just what is shared

The Grid Vision

- Simple, transparent access to resources without central control
- Dynamic coordination and combination of services on demand
- Easy addition of resources
- Autonomic management of Grid components
- Complexity of the infrastructure is hidden from user or resource provider





Requirements

- Identity & authentication
- Authorization & policy
- Resource discovery
- Resource characterization
- Resource allocation
- (Co-)reservation, workflow
- Distributed algorithms
- Remote data access
- High-speed data transfer
- Performance guarantees
- Monitoring

- Adaptation
- Intrusion detection
- Resource management
- Accounting & payment
- Fault management
- System evolution
- and many more …

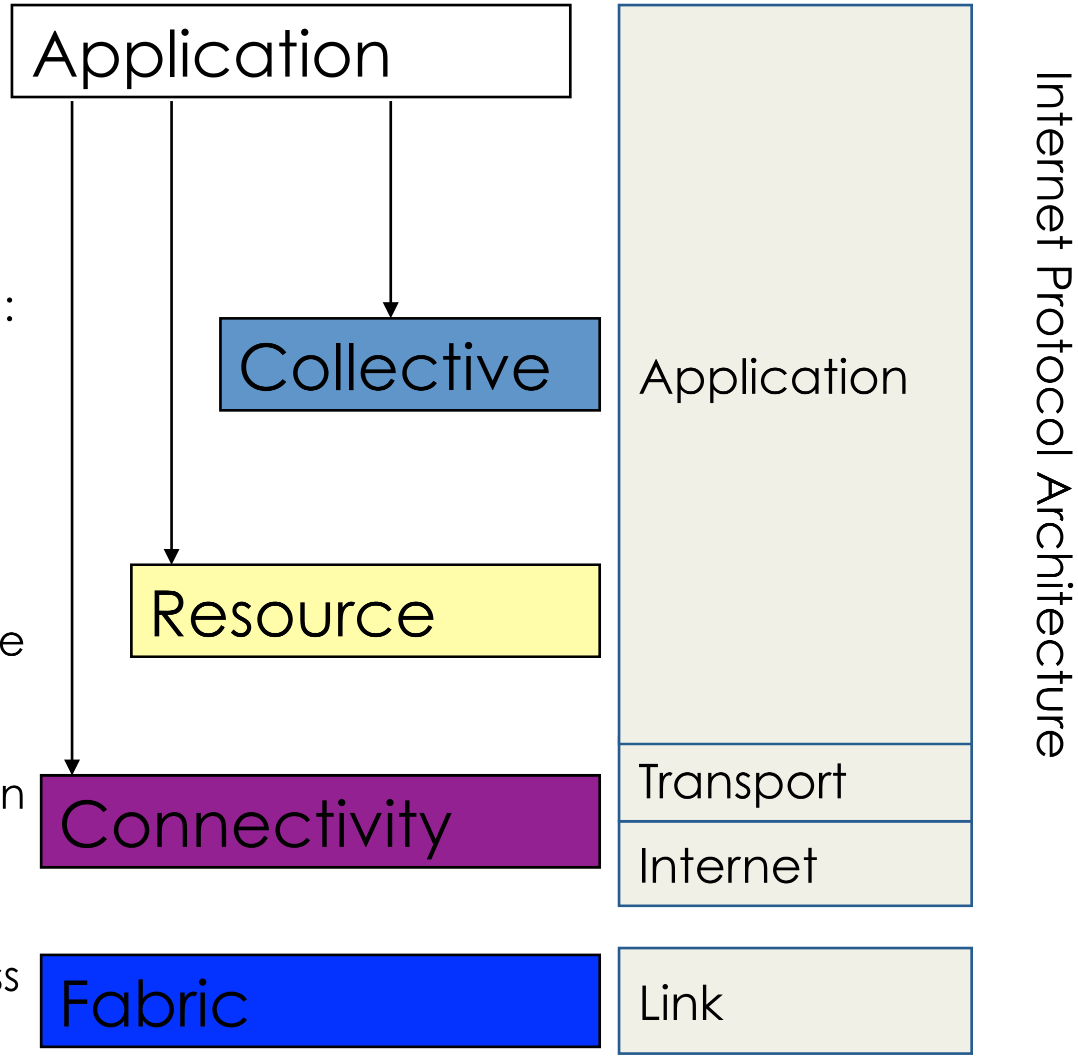
Grid Architecture

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

"Sharing single resources": negotiating access, controlling use

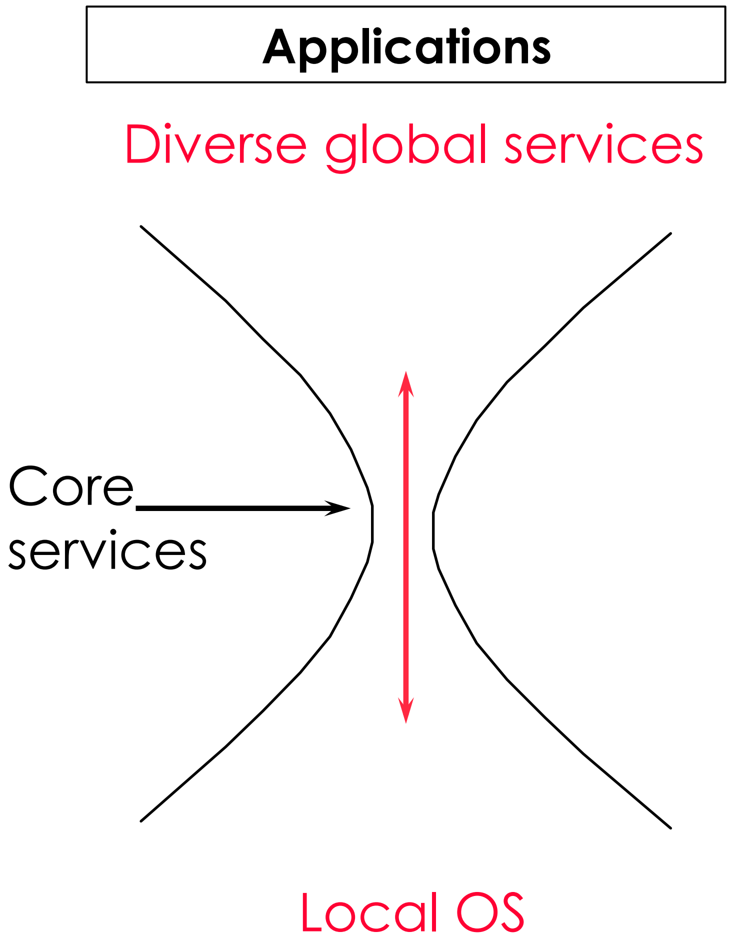
"Talking to things": communication (Internet protocols) & security

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



The Hourglass Model

- Focus on architecture issues
  - Propose set of core services as basic infrastructure
  - Use to construct high-level, domain-specific solutions
- Design principles
  - Keep participation cost low
  - Enable local control
  - Support for adaptation
  - "IP hourglass" model



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
  - OGF has an architecture working group (OGSA)
  - Technical specifications are being developed for architecture elements: e.g., security, data, resource management, information
  - Internet drafts submitted in security area

Fabric Layer: Protocol & Services

- 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.
- Few constraints on low-level technology: connectivity and resource level protocols form the "neck in the hourglass"
- Defined by interfaces not physical characteristics

Connectivity Layer: Protocol & Services

- 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, …

Resource Layer: Protocol & Services

- Grid Resource Allocation Management (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
- Others emerging: Catalog access, code repository access, accounting, etc.
- All built on connectivity layer: GSI & IP

Collective Layer: Protocol & Services

- Index servers a.k.a. meta-directory services
  - Custom views on dynamic resource collections assembled by a community
- Resource brokers
  - Resource discovery and allocation
- Replica catalogs
- Replication services
- Co-reservation and co-allocation services
- Workflow management services
- etc…

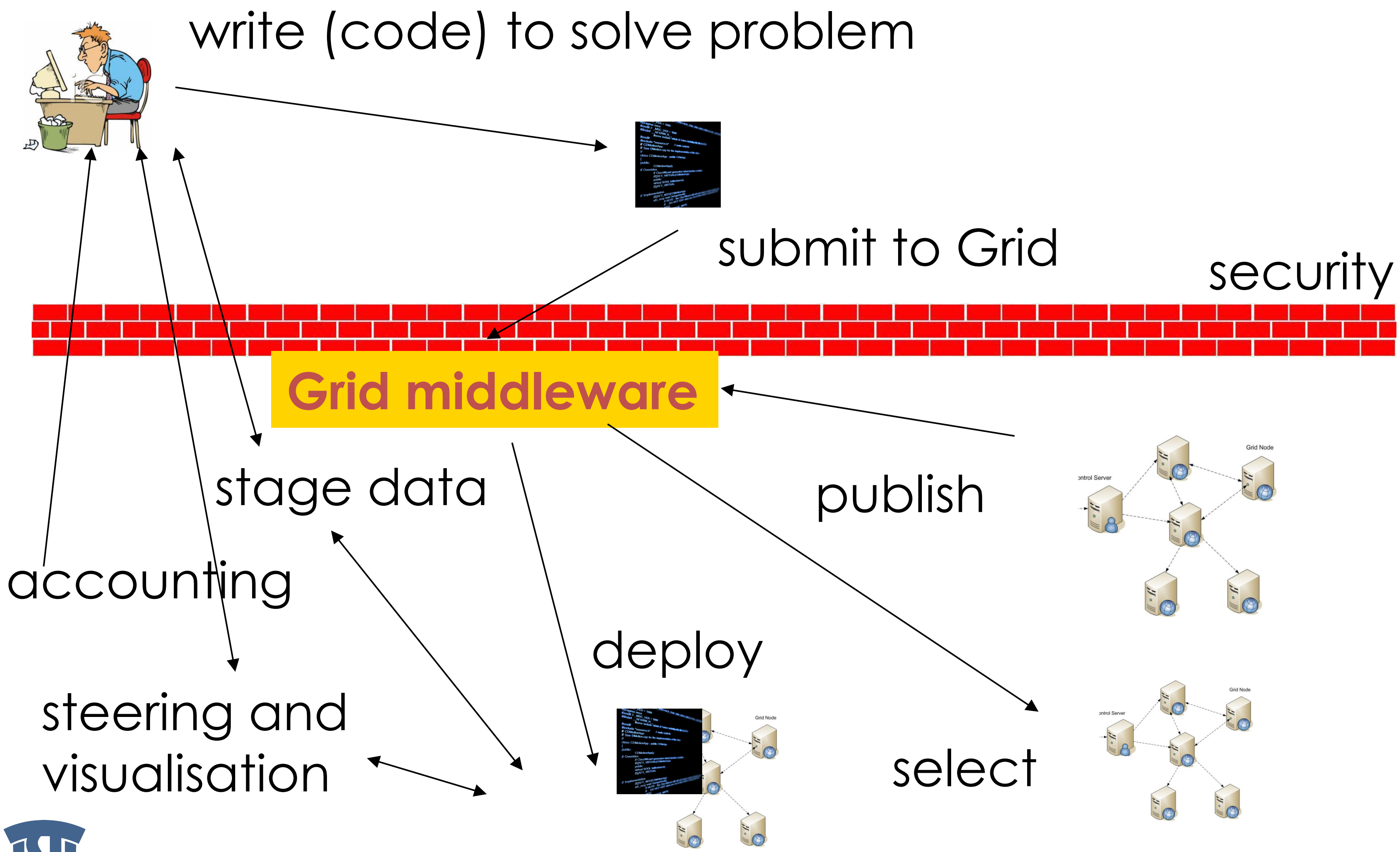
Example: High-Throughput Computing

App	High Throughput Computing System
Collective (App)	Dynamic checkpoint, job management, failover, staging
Collective (Generic)	Brokering, certificate authorities
Resource	Access to data, access to computers, access to network performance data
Connect	Communication, service discovery (DNS), authentication, authorization, delegation
Fabric	Storage systems, schedulers

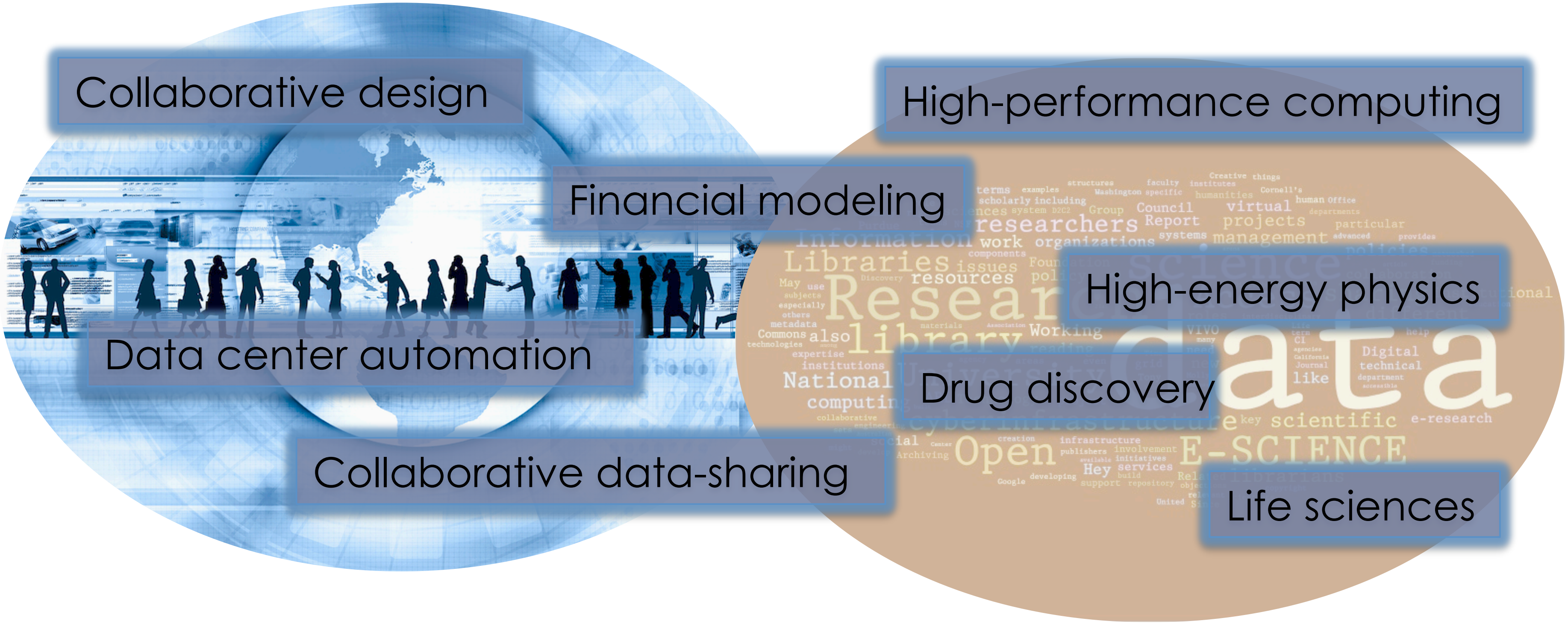
Example: Data Grid Architecture

App	Discipline-Specific Data Grid Application
Collective (App)	Coherency control, replica selection, task management, virtual data catalog, virtual data code catalog, …
Collective (Generic)	Replica catalog, replica management, co-allocation, certificate authorities, metadata catalogs, …
Resource	Access to data, access to computers, access to network performance data, …
Connect	Communication, service discovery (DNS), authentication, authorization, delegation
Fabric	Storage systems, clusters, networks, network caches, …

Using the Grid



How Grids are used?



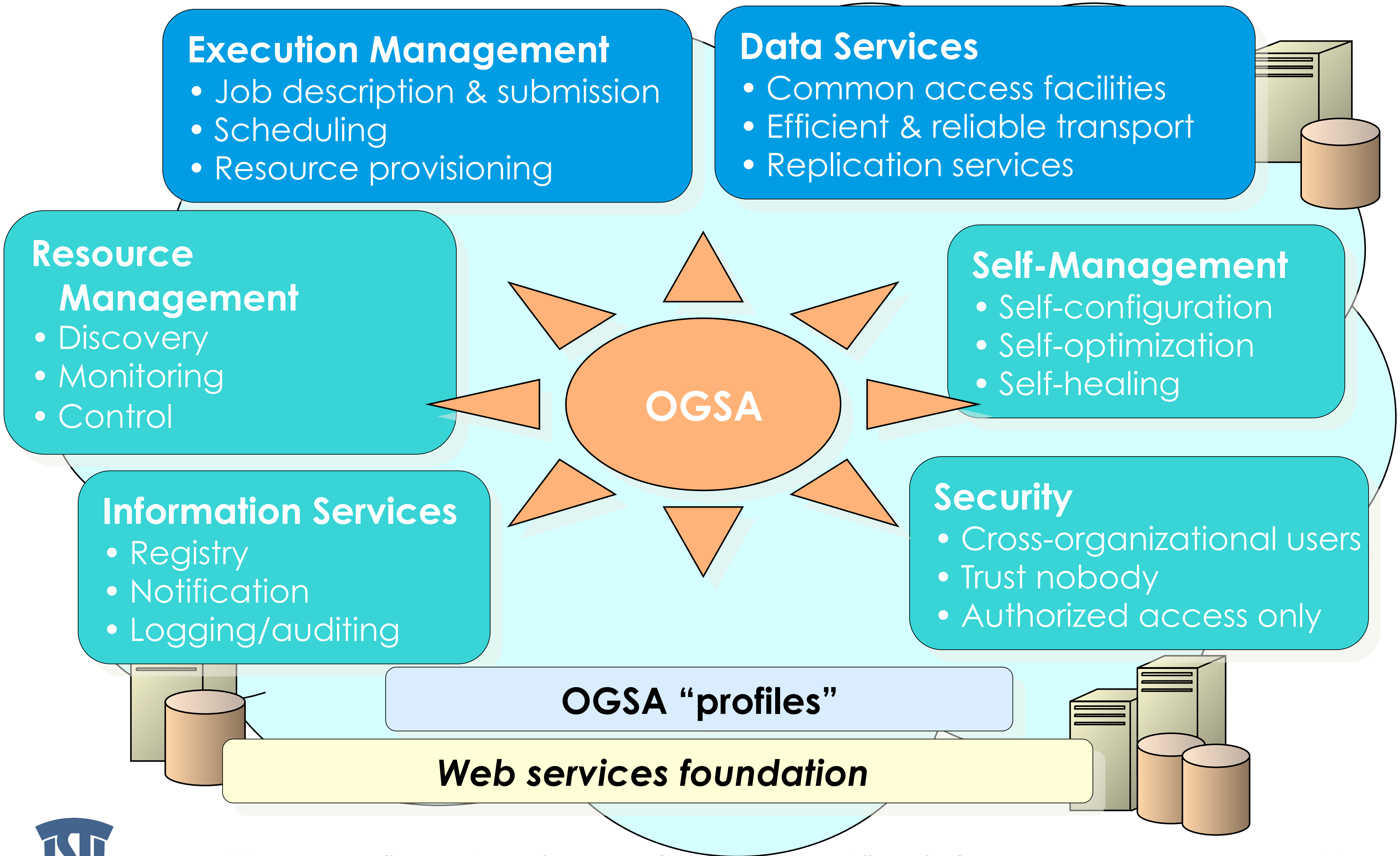
E-Business

E-Science

Open Grid Services Architecture

- Developed by the Global Grid Forum to define a common, standard, and open architectures for Grid-based applications.
  - Provides a standard approach to all services on the Grid.
    - VO Management Service.
    - Resource discovery and management service:
    - Job management service.
    - Security services.
    - Data management services.
- ***Built on top of and extends the Web Services architecture, protocols, and interfaces.***
- [***http://www.ogf.org/documents/GFD.80.pdf***](http://www.ogf.org/documents/GFD.80.pdf)

OGSA Capabilities



Grid Scenarios

- **Collaboration Grids**
  - Multiple institutions, secure, widely distributed, VOs
  - Collaborative agreements & commercial partnerships
  - Financial Model: Increase overall revenue
- **Data Center Grids** (evolving to Clouds)
  - Centralized management of multiple platforms
  - Aggregation of enterprise resources and applications
  - Financial Model: Reduce Total Cost Ownership (TCO)
- **Cluster Grids**
  - Networks of Workstations, Blades, etc.
  - Cycle scavenging, Homogeneous workload
  - Financial Model: Lower marginal costs

The Eight Fallacies of Distributed Computing (and the Grid)

- The *resources* are (network is) reliable
- *Resource* latency is zero
- *Resource* bandwidth is infinite
- The *resources are* (network is) secure
- *Resource* topology does not change
- There is one *resource* administrator
- Resource (transport) cost is zero
- The resources are (network is) homogeneous

Adapted from Deutsch & Gosling