



the globus project™
www.globus.org

The Taming of the Grid: Services and Applications

Kate Keahey

The Globus Project™

Mathematics and Computer Science Division

Argonne National Laboratory

Overview

- † Computational Grids
 - Examples of Grid applications
 - Review of requirements
- † Review of existing technologies
 - Globus
 - Web Services
- † Overview of OGSA
- † Benefits of service-oriented architecture:
Application Services



“Computing Power on Tap”

“Resource sharing & coordinated problem solving in dynamic, multi-institutional virtual organizations”

- On-demand, ubiquitous access to computing, data, and services
- New capabilities constructed dynamically and transparently from distributed service

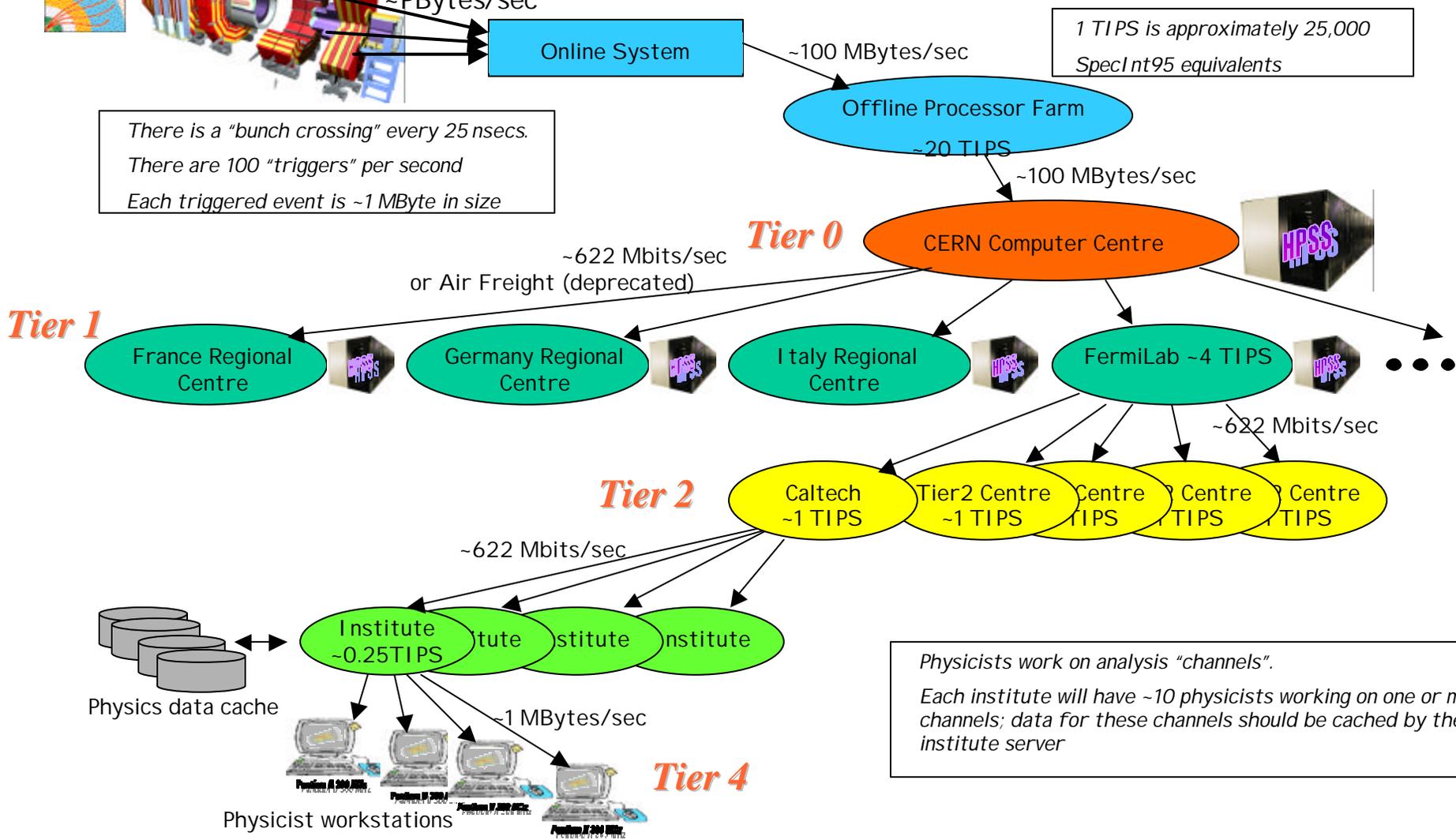


Grid Communities & Applications: Data Grids for High Energy Physics



There is a "bunch crossing" every 25 nsecs.
There are 100 "triggers" per second
Each triggered event is ~1 MByte in size

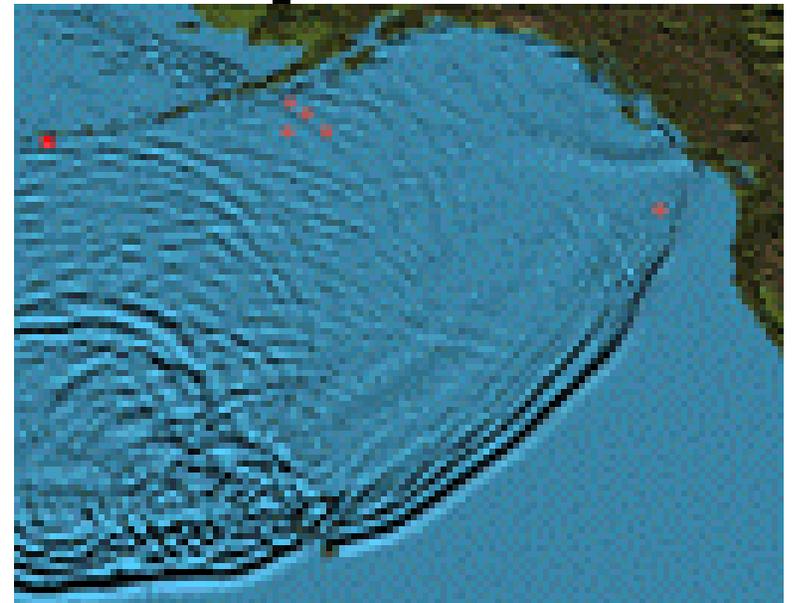
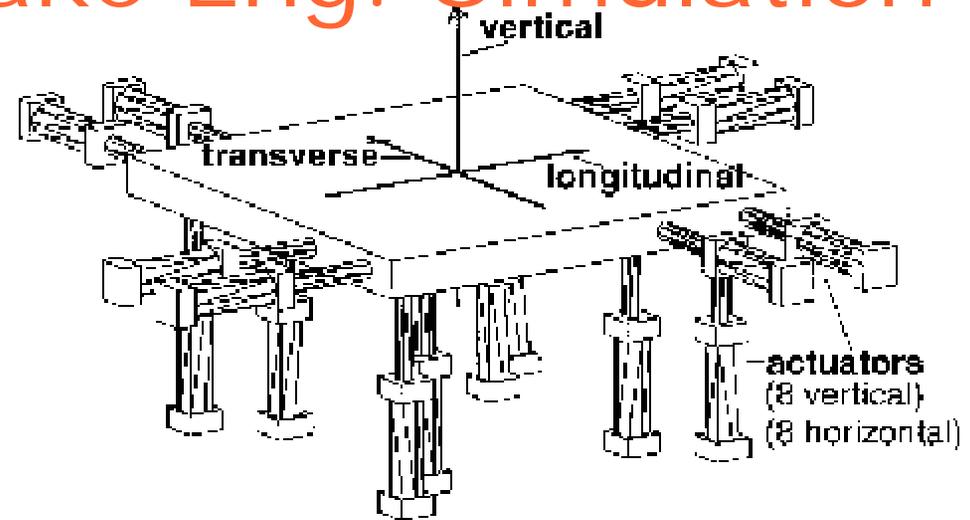
1 TIPS is approximately 25,000
SpecInt95 equivalents



Physicists work on analysis "channels".
Each institute will have ~10 physicists working on one or more channels; data for these channels should be cached by the institute server

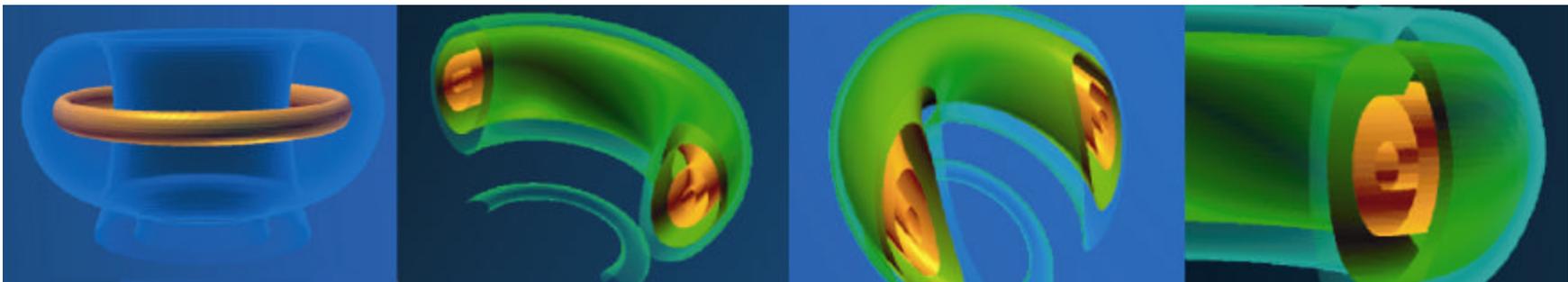
Grid Communities and Applications: Network for Earthquake Eng. Simulation

- † NEESgrid: US national infrastructure to couple earthquake engineers with experimental facilities, databases, computers, & each other
- † On-demand access to experiments, data streams, computing, archives, collaboration



The National Fusion Collaboratory

- † Fusion experiments
 - A series of pulses 15-20 mins apart
 - Need to run codes between pulses
 - QoS-based execution
- † Fusion codes
 - Hard to port and maintain, but critical to update





Requirements Include ...

- † Online negotiation of access to services and resources: who, what, why, when, how
- † Establishment of applications and systems able to deliver multiple qualities of service
- † Other:
 - Dynamic formation and management of Virtual Organizations (VOs)
 - Autonomic management of infrastructure elements
- † In short: open, extensible, evolvable infrastructure

The Globus Toolkit

- † Tools enabling resource sharing within VOs
 - GRAM (Grid Resource Allocation and Management)
 - > Tool for remote job and resource management
 - GSI (Grid Security Infrastructure)
 - > Authentication based on Grid-wide credential
 - > Single sign-on, delegation
 - > Authorization
 - MDS (Monitoring and Discovery Service)
 - > Grid-wide information on the state of resources
 - Data Grid Technologies
 - > GridFTP
 - > Replica Management
 - > Virtual data
- † Protocols and APIs

Globus Toolkit: Evaluation

† Plus

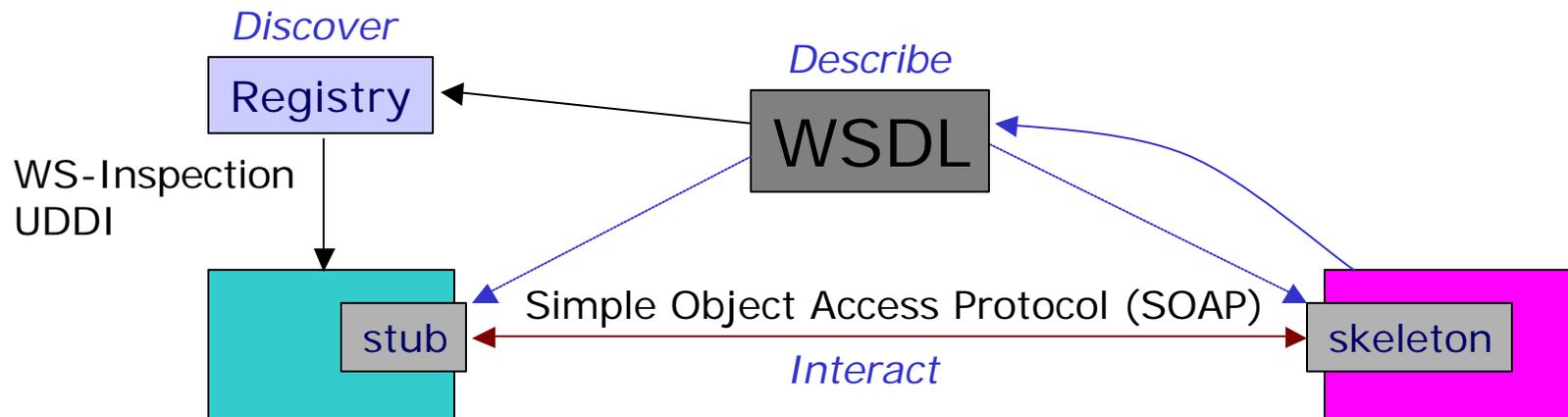
- Good technical solutions providing key capabilities
- Good quality reference implementation
 - > interfaces to many systems
 - > industrial support (very important)
- Growing community code/user base using these tools

† Minus

- Heterogeneous protocol basis: HTTP, LDAP, FTP
- No standard means of invocation, notification, error propagation, authorization, lifecycle, ...
- Significant missing functionality, e.g.
 - > Databases, sensors, instruments, workflow, ...
 - > Virtualization of end systems (hosting envs.)
- Little work on total system properties, e.g.
 - > Dependability, end-to-end QoS, ...

Web Services

- † A set of standards for network applications
 - W3C standardization; Microsoft, IBM, Sun, others



- † Benefits:
 - Platform and programming language independent
 - Focus on interface
 - > Minimal shared understanding between service provider and requestor
 - Enables Modular design, reusability, etc.

Open Grid Services Architecture

- † A unifying framework for the Grids
- † Based on widely accepted industry standards
- † From Web services
 - Standard interface definition mechanisms: multiple protocol bindings, local/remote transparency
- † From Grids
 - Service semantics, reliability and security models
 - Lifecycle management, discovery, other services
- † Multiple “hosting environments”
 - **C/C++**, J2EE, .NET, ...
- † Evolving even as we speak...



The Grid Service = Interfaces + Service Data

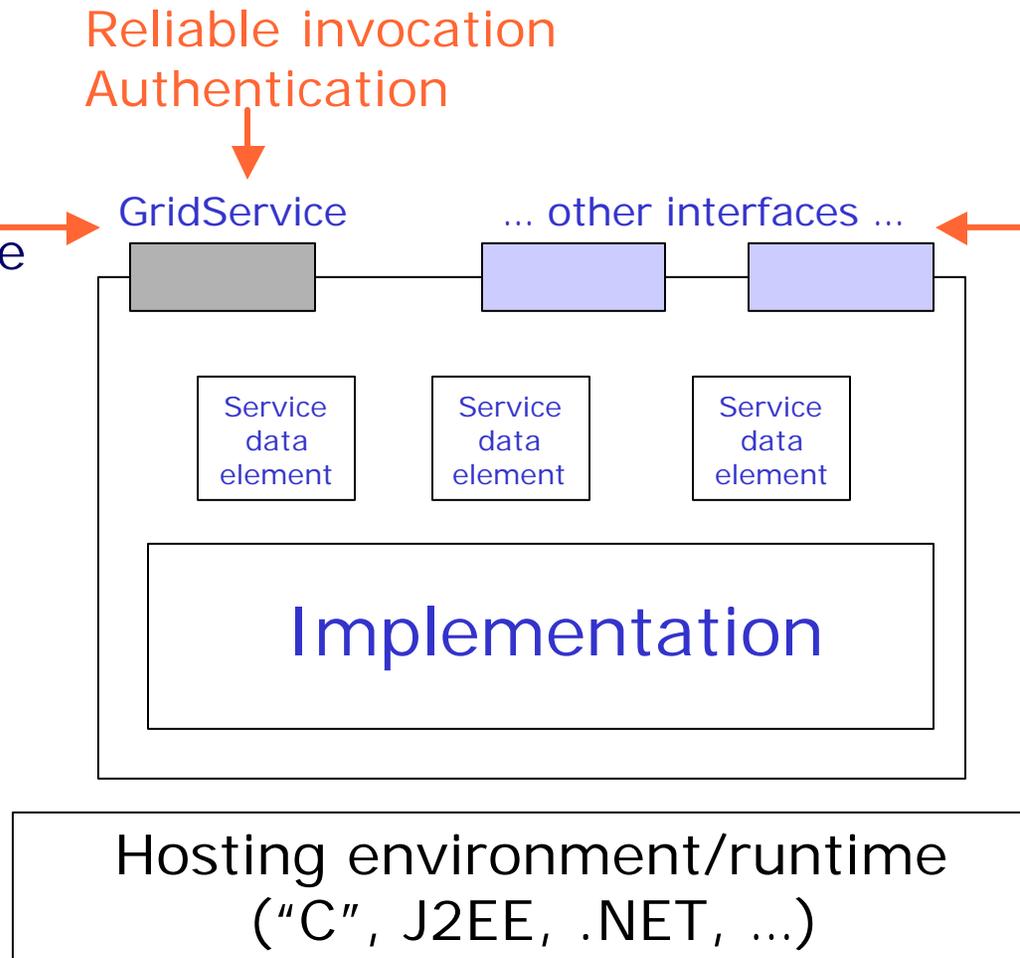
Required:

FindServiceData
SetTerminationTime
Destroy

Reliable invocation
Authentication

Optional:

Notification
Authorization
Service creation
Service registry
Manageability
Concurrency



Standard Interfaces & Behaviors: Four Interrelated Concepts

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

Naming and Bindings

- † Every service instance has a unique and immutable name: Grid Service Handle (GSH)
 - URI: encodes information on how to resolve it
 - > Can be resolved by the client itself
 - > HandleResolver
- † Handle must be converted to a Grid Service Reference (GSR) to use service
 - Includes all the information a client needs to communicate with the service
 - Binding-specific
 - > SOAP: WSDL document
 - > RMI/IIOP: IOR
 - may expire during the lifetime of the service
- † Separation of name from implementation facilitates service evolution

Service Data

- † A Grid service instance maintains a set of service data elements
 - Meta-data (static information: info about types, etc.)
 - State data (dynamic properties: lifetime, application-specific, etc.)
- † **Attributes**
 - Name and type
 - Lifetime declarations:
 - > goodFrom, goodUntil, notGoodAfter
- † **FindServiceData** operation (GridService interface) queries this information
 - Input: query types and query expressions
 - Output: result of the query



Lifetime Management

- † Creation of Grid Service instances
 - manually
 - using a factory
 - > input: termination time and service-specific parameters
 - > output: ServiceLocator -> GSH
- † Destruction
 - **Destroy** operation for explicit destruction
 - **SetTerminationTime** enables soft state destruction
- † Soft state lifetime management avoids
 - Explicit client teardown of complex state
 - Resource “leaks” in hosting environments
 - Termination time can be extended by the client



Notification Interfaces

- † **NotificationSource** for client subscription
 - One or more *notification generators*
 - > Generates notification message of a specific type
 - > Typed *interest statements*: E.g., Filters, topics, ...
 - > Supports messaging services, 3rd party filter services, ...
 - Soft state subscription to a generator
 - PortType: Subscribe, Unsubscribe
- † **NotificationSink** for asynchronous delivery of notification messages
 - DeliverNotification
- † A wide variety of uses are possible
 - E.g. Dynamic discovery/registry services, monitoring, application error notification, ...

Registry

- † The **Registry** interface may be used to register Grid service instances with a registry
 - A set of Grid services can periodically register their GSHs into a registry service, to allow for discovery of services in that set
- † Registrations maintained in a service data element associated with Registry interface
 - Standard discovery mechanisms can then be used to discover registered services
 - Returns a WS-Inspection document containing the GSHs of a set of Grid services



Service-Oriented View of the Grid

- † Service-orientation facilitates virtualization
- † For example: Grid as a collection of application services rather than resources
- † Resources are “invisible” to the end user
 - User can focus on the application
 - Service provider maintains the service and its complexities
 - Requires providing application-level QoS
- † **Examples:**
 - Hard to port applications (TRANSP in NFC)
 - > Need to be updated frequently
 - Resource thirsty applications (Cactus)
- † One answer to “computing power on tap”

Motivating Example: TRANSP

† Characteristics of the software

- Software is hard/expensive to port and maintain (complex)
- Needs updating frequently and consistently (physics changes)
- “Software Grid” as important as “Hardware Grid”

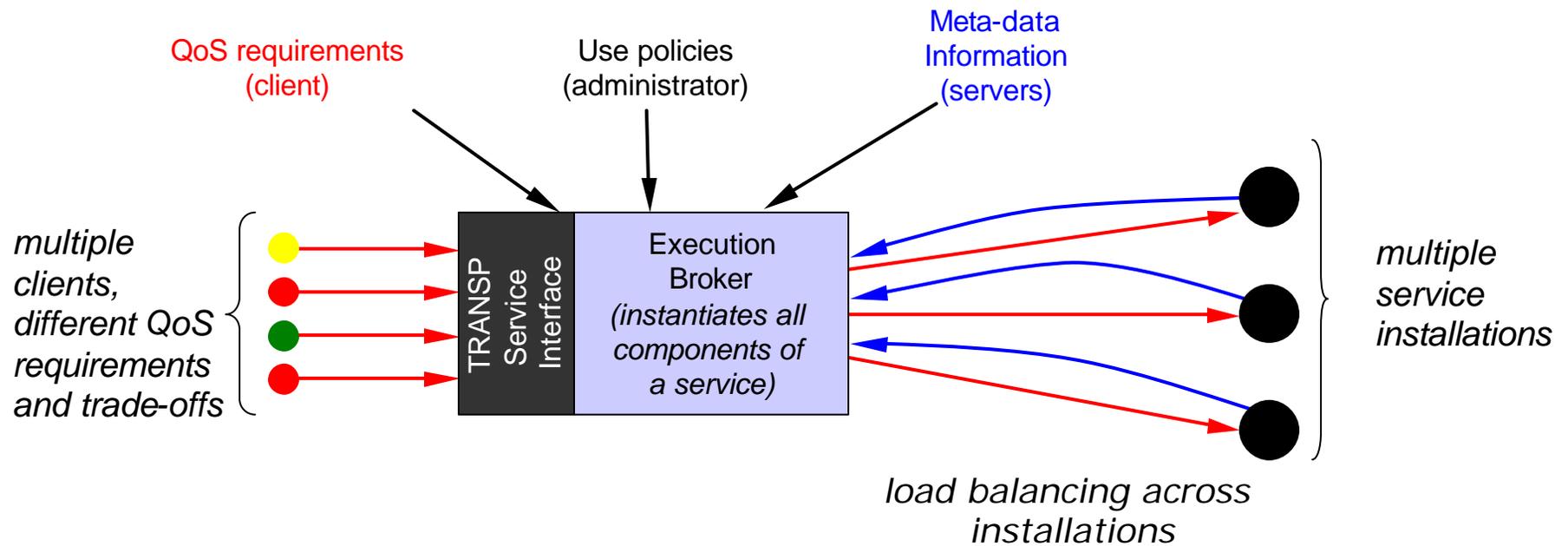
† Scenario:

- Requirement: run betw. experimental pulses in less than 10 mins
- TRANSP inputs can be experimentally configured beforehand to determine how its execution time relates to them
 - > Loss of complexity (“physics”) to gain time
- The scientist makes a reservation for the time of the experiment
- Multiple executions of TRANSP, initiated by different clients and requiring different QoS guarantees can co-exist on the cluster at any time, but when a reservation is claimed, the corresponding TRANSP execution claims full CPU power

† Solution: provide TRANSP as a “Network Service”



TRANSP as a "Network Service"



- † **Status: OGSA-based prototype!**
 - OGSA leverage: Factories, Service Data
 - GARA-inspired solutions: DSRT, preemption
- † **Joint work with Kal Motawi**

What Next?(Capability Challenges)

† Issues of Trust

- How do I enter into contract for resource usage?
- How do I ensure that this contract is observed?
- Will my code get priority when it is needed?
- How do I deal with a dynamic set of users?

† Issues of Application Management

- Orchestration: combining many application services
- Application-specific meta-data
 - > Dependency management
 - > Self-deployable applications
- Application-level QoS

† Issues of Resource Management

- The problem of “shared environment”



What Next?(Technical Challenges)

- † Hosting Environments
 - Bindings in C/C++, Python...
- † Emphasis on high-performance
 - Protocol bindings for efficient communication
- † Common Services
 - Geared towards the needs of STC community
- † DOE Grid Services proposal
 - Co-PI with Keith Jackson
- † Building support partnerships

Conclusions

- † The Globus Project leads the exploration of computational Grids
- † OGSA: leveraging emerging technologies towards a better Grid infrastructure
 - Providing an architecture for the Grids
 - Creating opportunity for new capabilities
- † A service-oriented view of the Grids provides one answer to “computing power on tap”
 - Virtualization in terms of applications
 - Virtualization in terms of resources