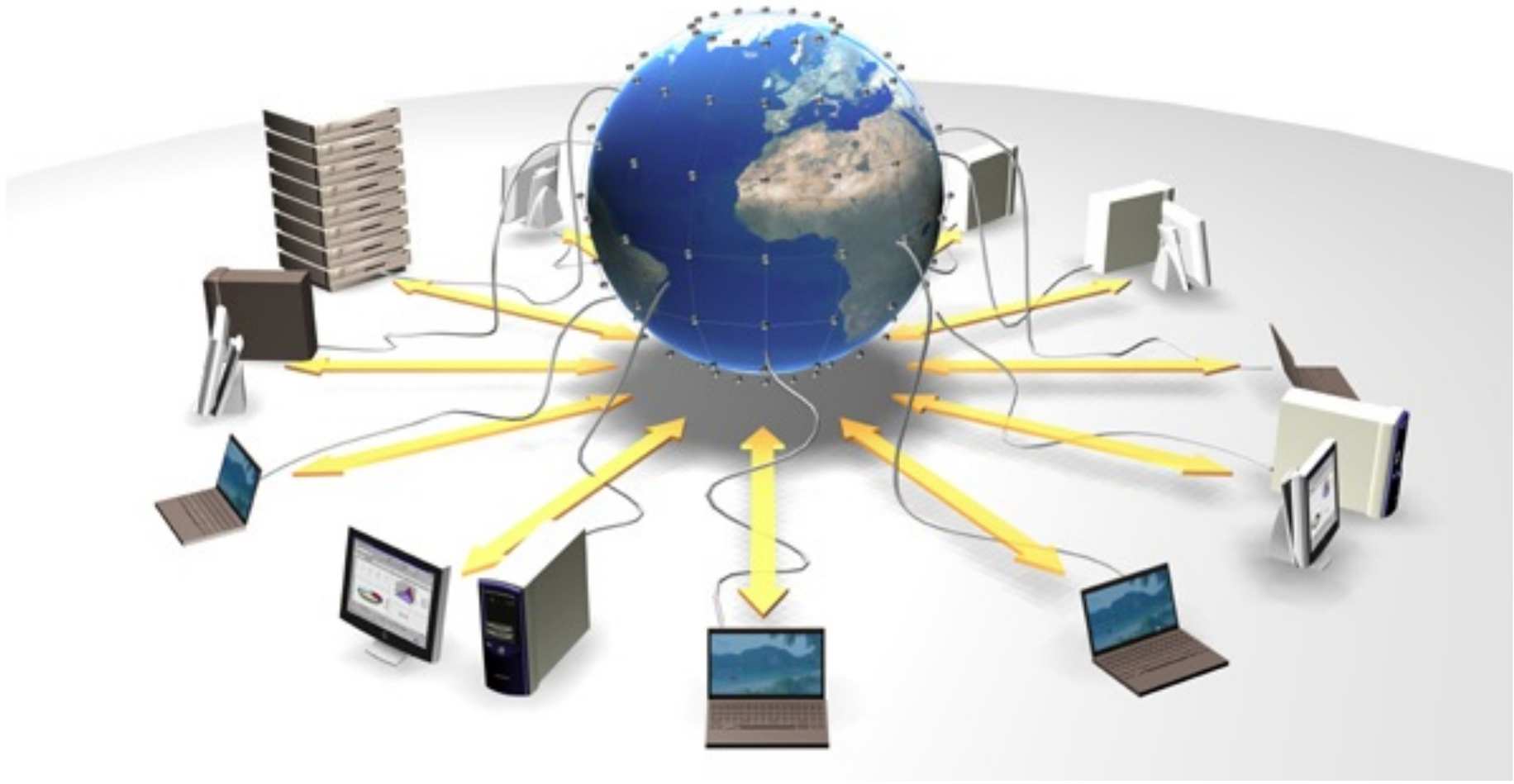
Grid Computing



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

Definitions

We define a **Grid** as a system that

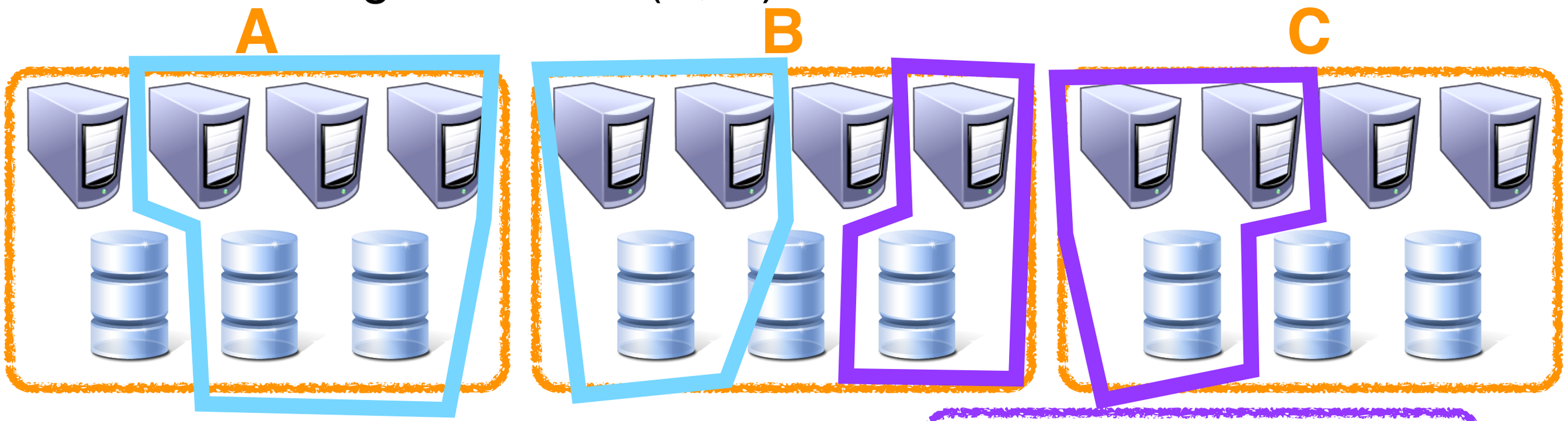
- **coordinates distributed resources**
  - integrating and coordinating resources and users that live within different control domains
  - addressing the issues of security, policy, payment, membership, and so forth that arise in these settings
  - Otherwise, we are dealing with a local management system.
- **using standard, open, general-purpose protocols and interfaces**
  - built from multipurpose protocols and interfaces that address such fundamental issues as authentication, authorization, resource discovery, and resource access.
  - Otherwise, we are dealing with an application-specific system.
- **to deliver nontrivial qualities of service**
  - resources to be used in a coordinated fashion to deliver various qualities of service (e.g., response time, throughput, availability, and security)

Virtual Organizations

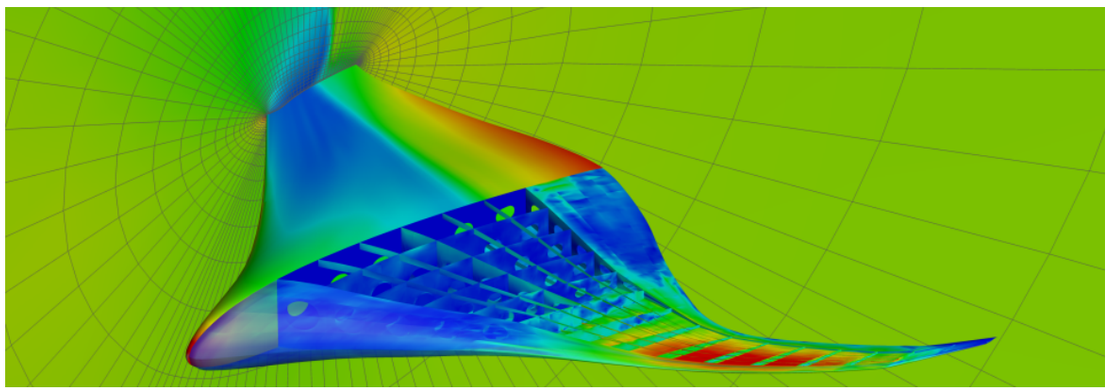
- Components
  - set of individual/institutions
  - set of resources
  - set of sharing rules
- Dynamic set of individuals and/or institutions defined by a shared goal and a set of sharing rules
- May vary in size, scope, duration and structure
  - Example: class students for cooperative lecture writing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

Example of VOs

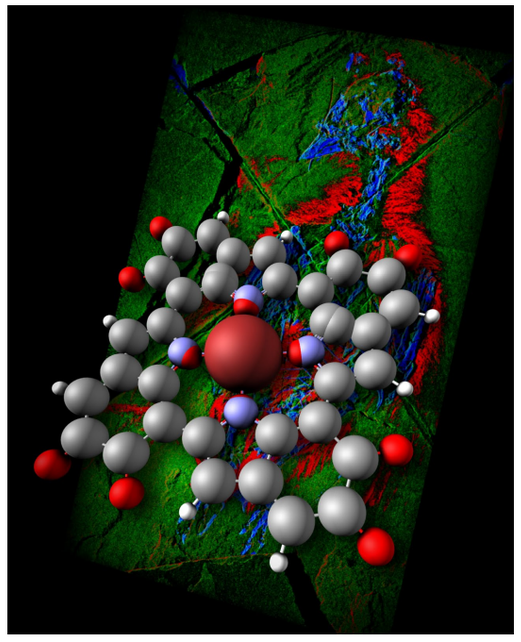
- Three physical organizations (A, B, C)
- Two virtual organizations (X, Y)



X



Multidisciplinary Design



Joint Drug Analysis

Y

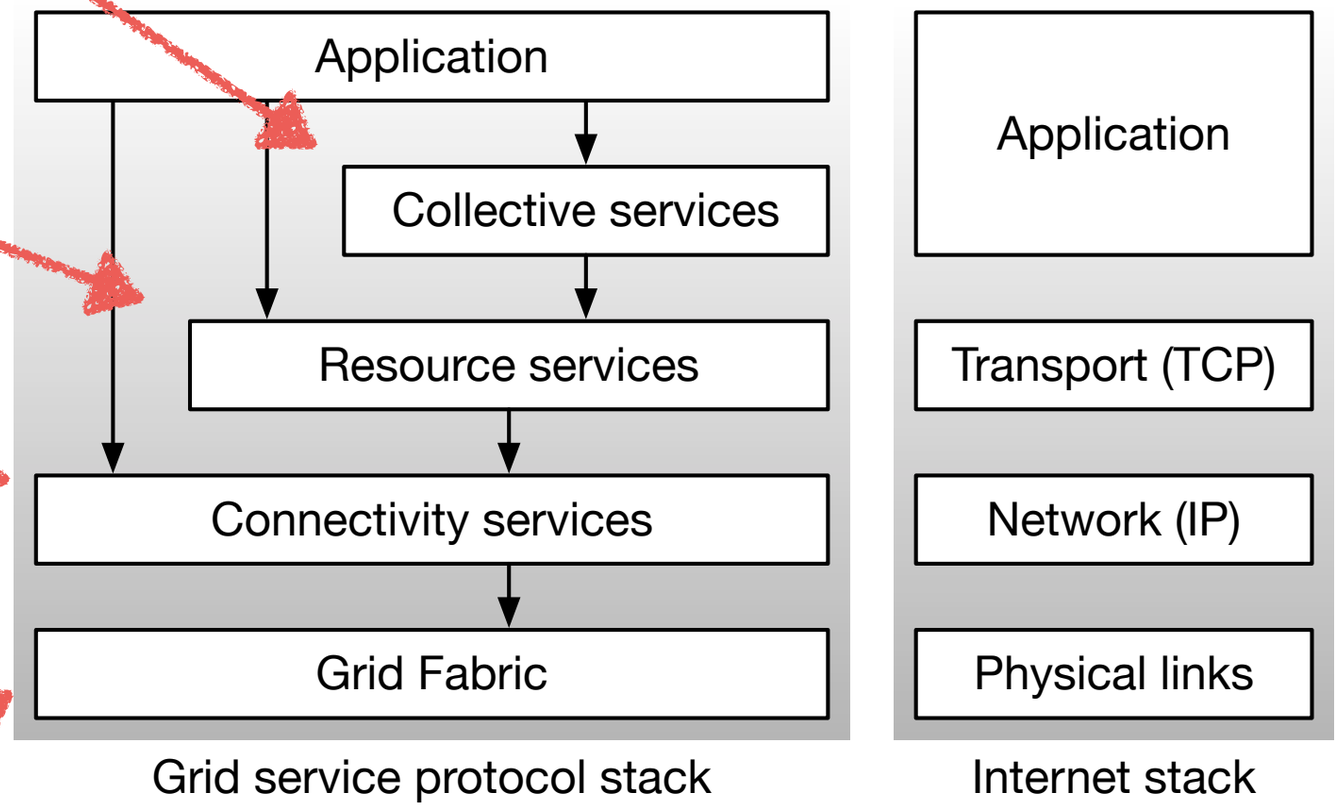
Grid Architecture

"Coordinating multiple resources": Ubiquitous infrastructure services, application-specific distributed services

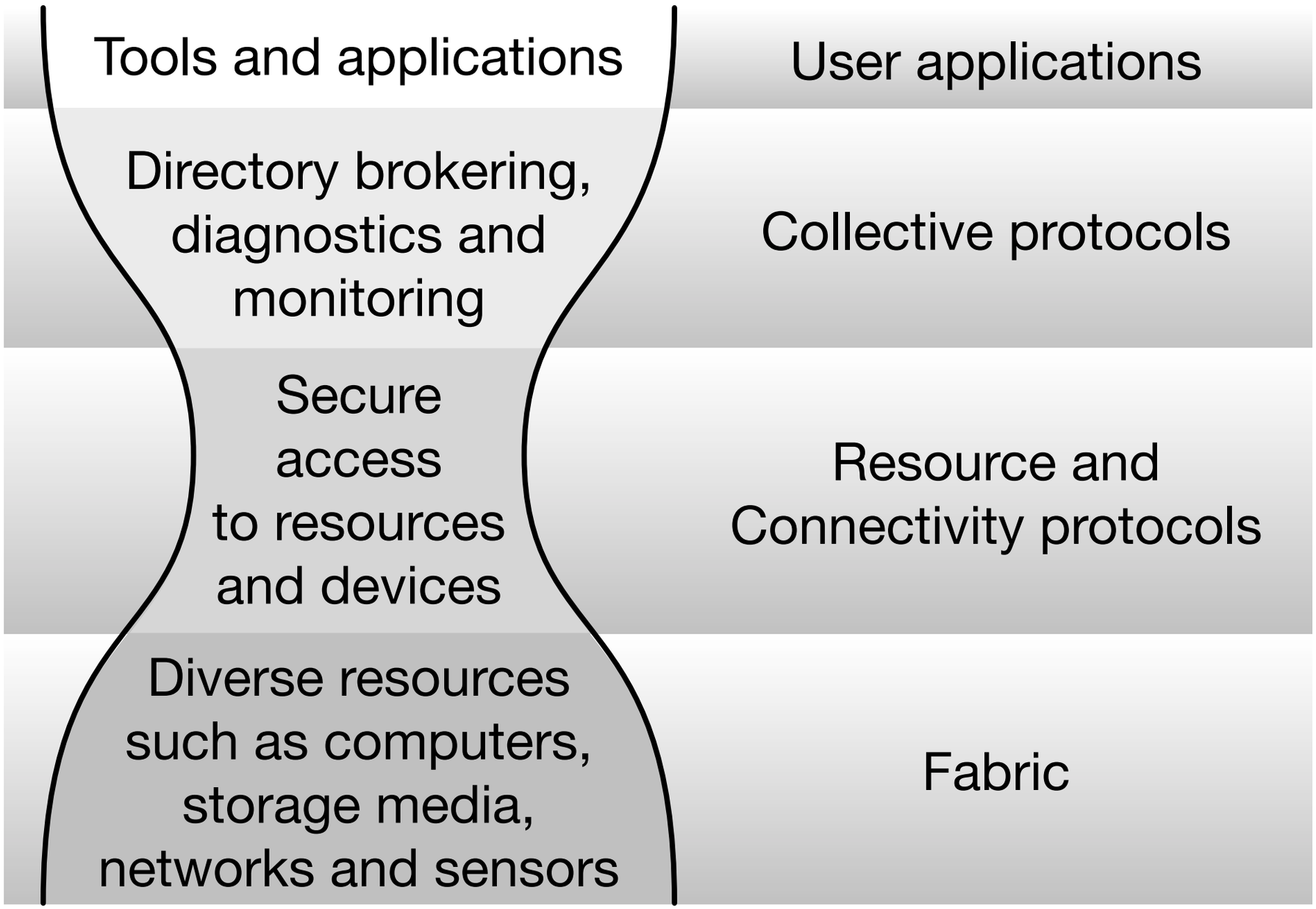
"Sharing single resources": Negotiating access, controlling use

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

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



The Hourglass Model



Fabric Layer

- 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

- 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

- 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

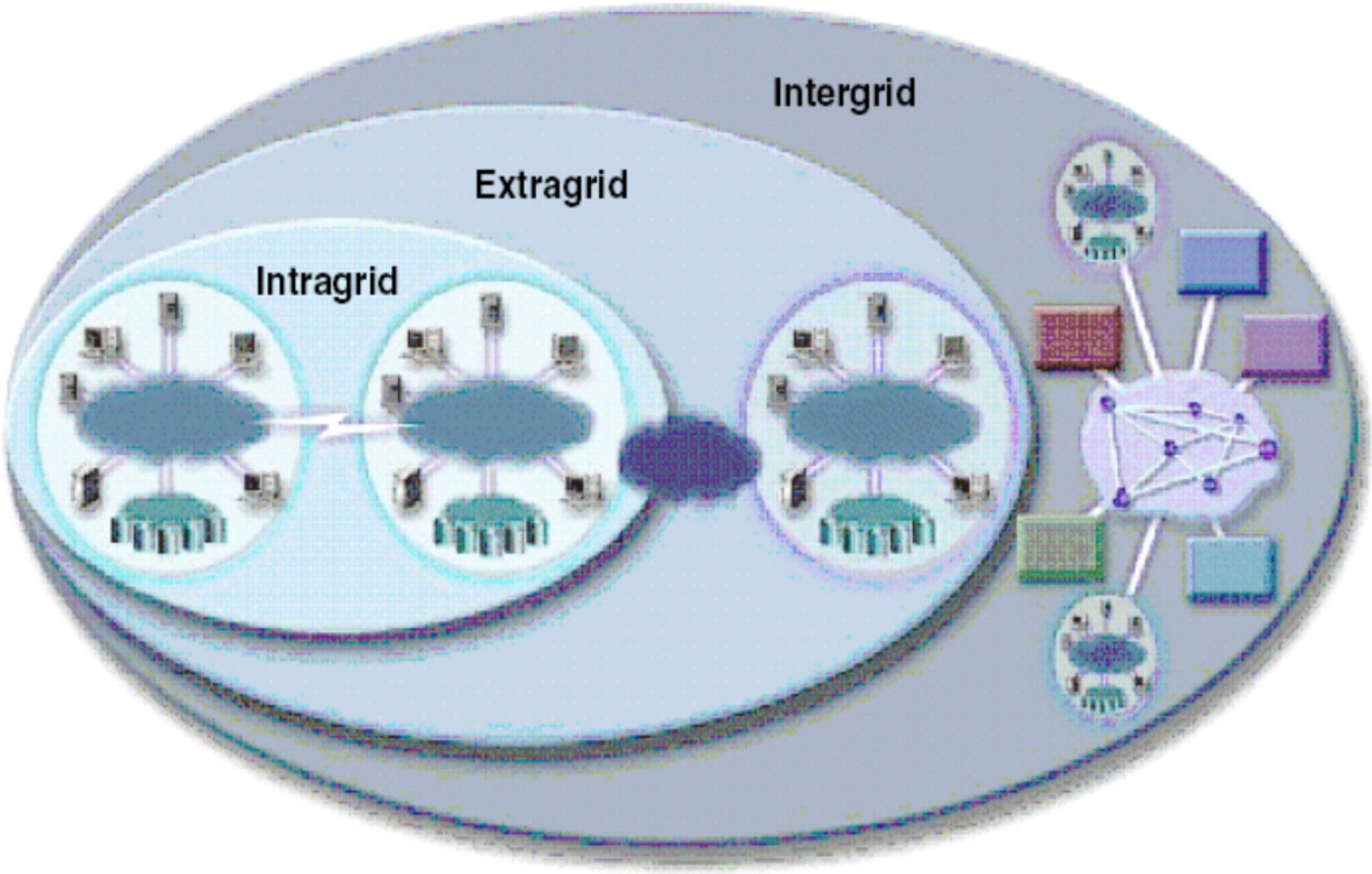
Collective Layer

- 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…

Grid Architectural Models

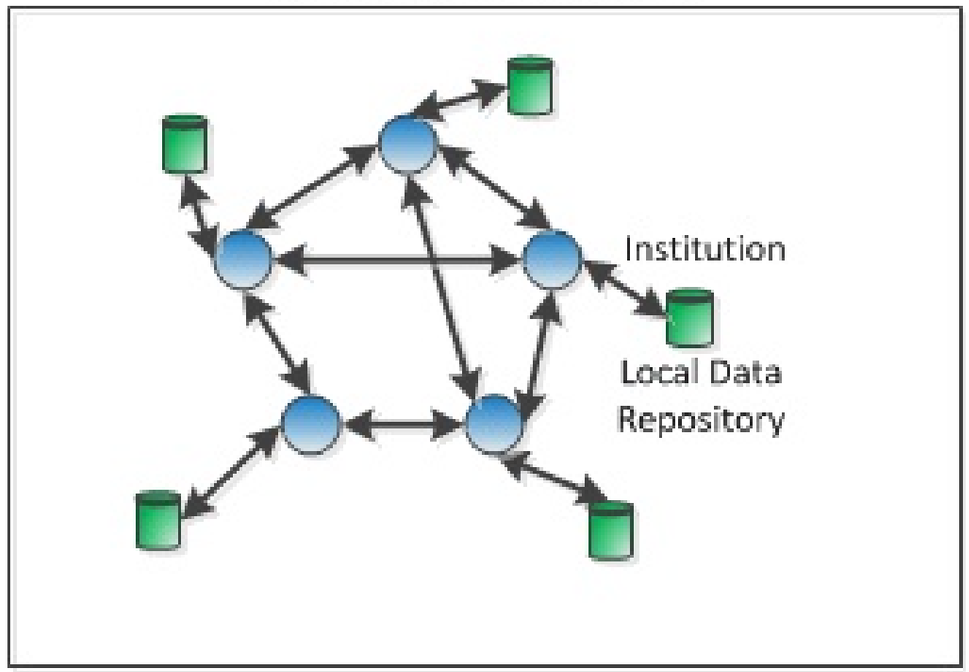
- **Computational Grids**
  - A computational Grid aggregates the processing power from a distributed collection of systems
  - This type of Grid is primarily composed of low powered computers with minimal application logic awareness and minimal storage capacity
  - The primary benefits of computational Grids are a reduced Total Cost of Ownership (TCO), and shorter deployment life cycles
  - Example: SETI@Home
- **DataGrids**
  - Data Grids focus on providing secure access to distributed, heterogeneous pools of data
  - Through collaboration, data Grids can also include a new concept such as a federated database
  - Data Grids also harness data, storage, and network resources located in distinct administrative domains, respect local and global policies governing how data can be used, schedule resources efficiently, again subject to local and global constraints, and provide high speed and reliable access to data
  - Example: CERN's LHC Data Grid

Computational Grid topologies

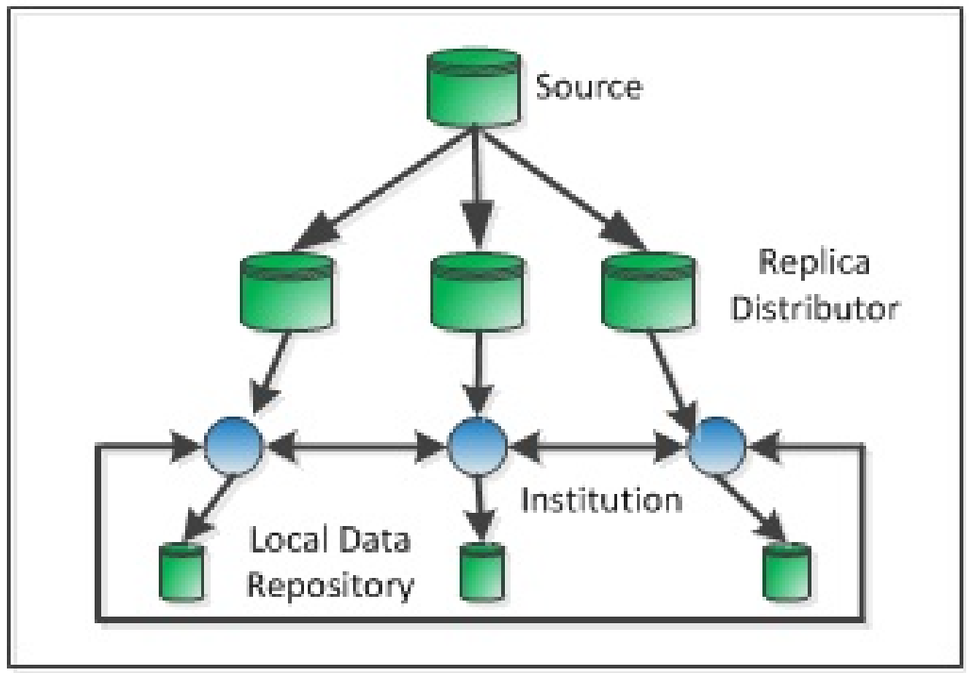


Data Grid topologies

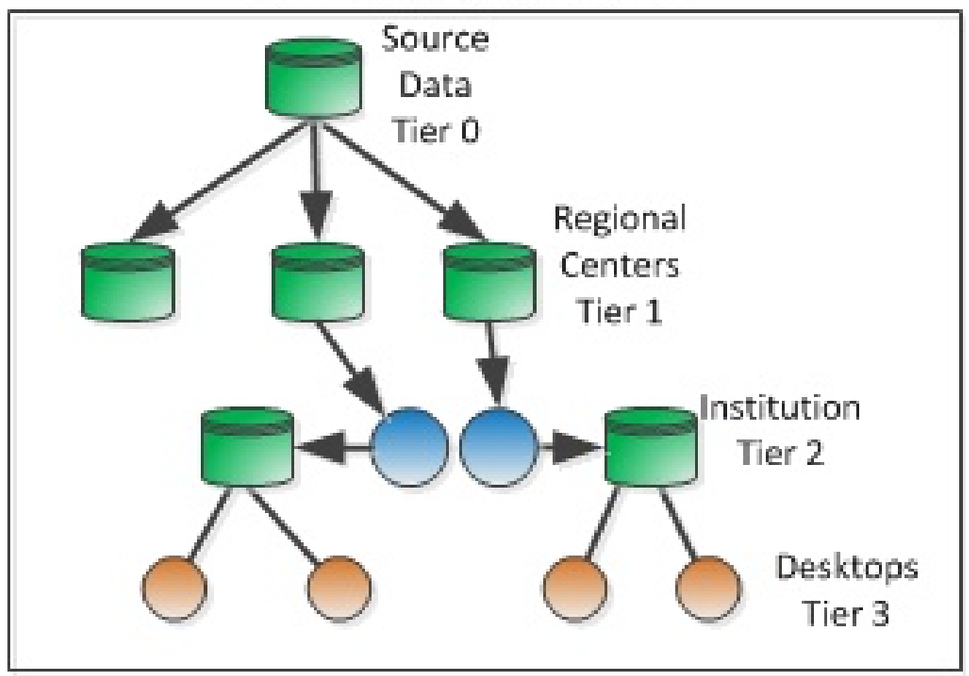
Federation Topology



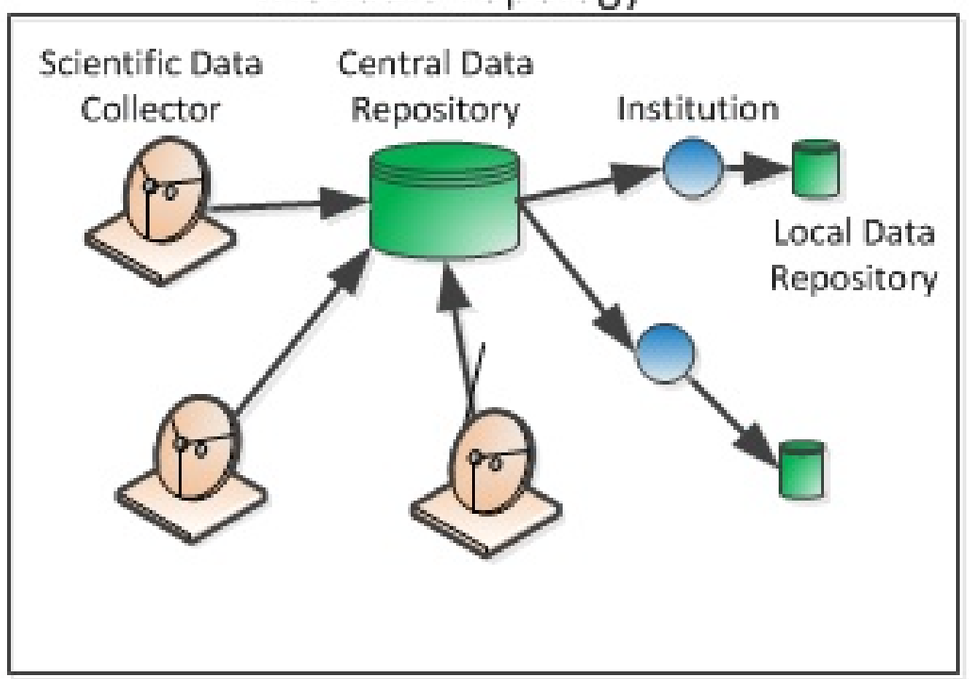
Hybrid Topology



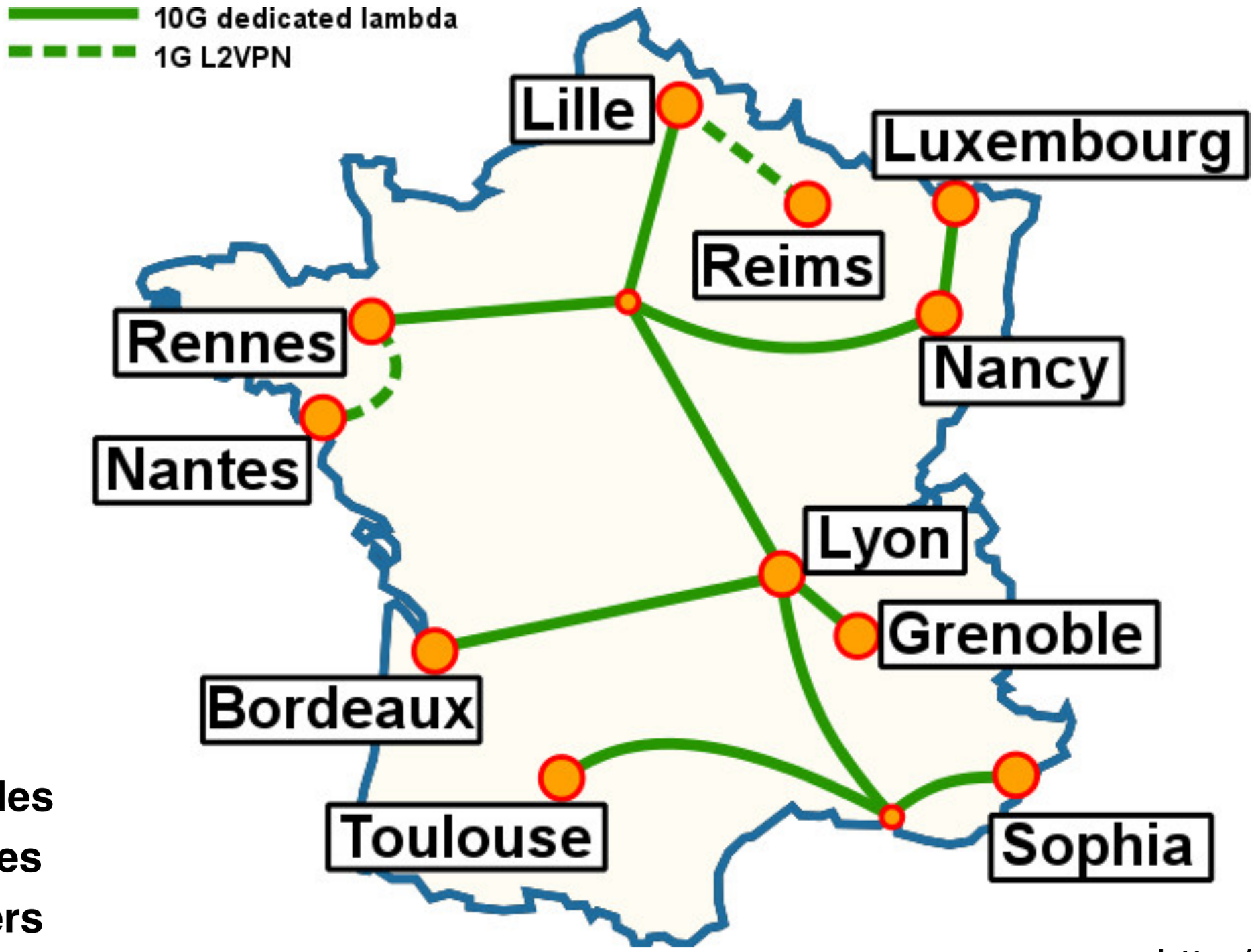
Hierarchical Topology



Monadic Topology



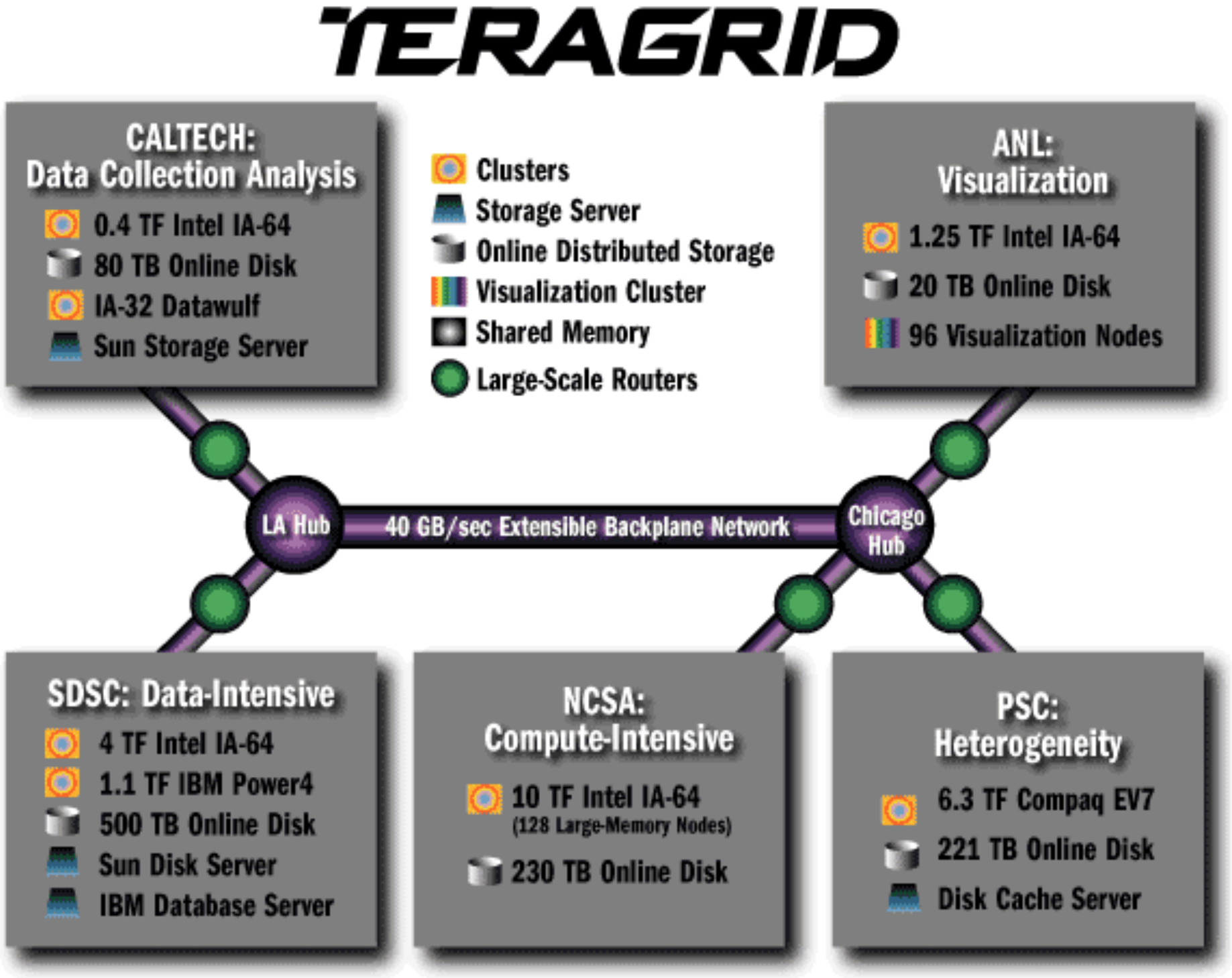
Grid 5000 Example



- 1000 nodes
- 8000 cores
- 500+ users

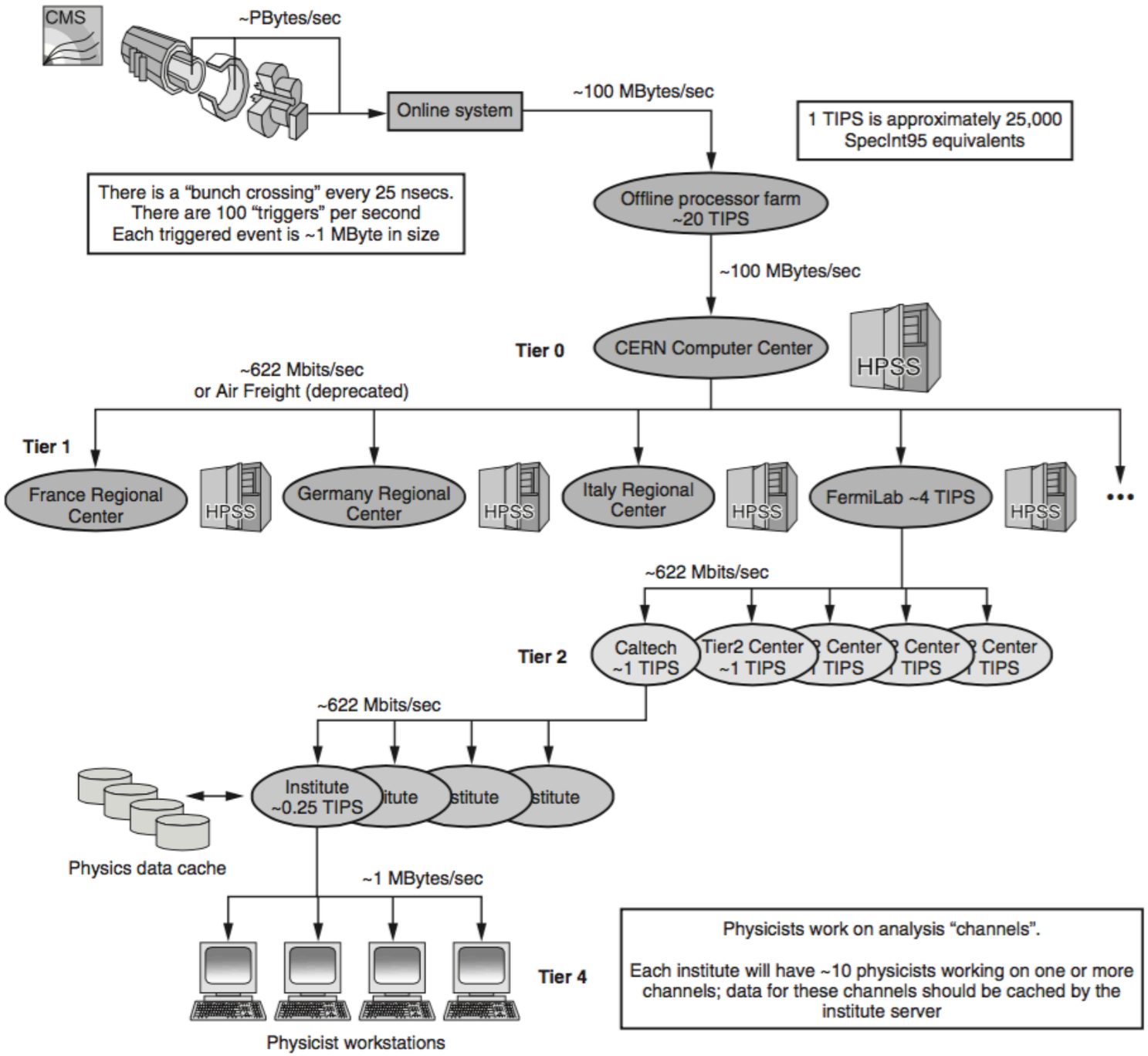
http://grid5000.fr

TeraGrid Example



sede.org

European Data Grid Example



http://eu-datagrid.web.cern.ch/eu-datagrid/

Globus Toolkit

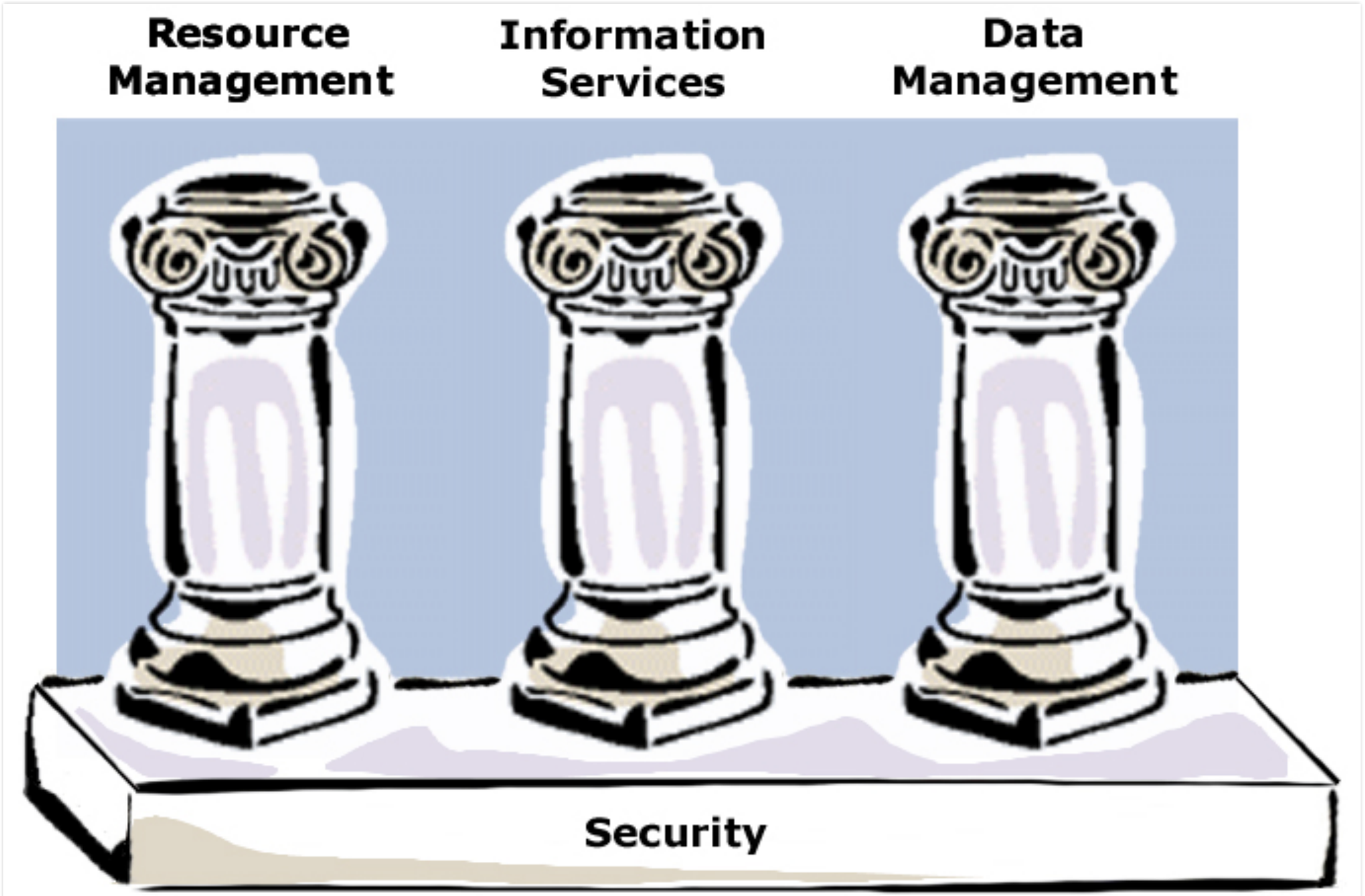
- An example Grid middleware

http://www.globus.org/toolkit/

- A software toolkit addressing key technical problems in the development of Grid enabled tools, services, and applications
  - Offer a modular “bag of technologies”
  - Enable incremental development of Grid-enabled tools and applications
  - Implement standard Grid protocols and APIs (the “core” of the hourglass)
  - Is available under liberal open source license

- Now is evolving to Cloud middleware

Key Protocols



GT2 Protocols and Services