



Grid Computing (Globus) Introduction

GUP Klausur

21.07.2004



Acknowledgments

- Slides Content taken from:
 - <http://www.globus.org/about/presentations>
 - <http://www.globus.org/about/events/US-tutorials/slides/index.html>



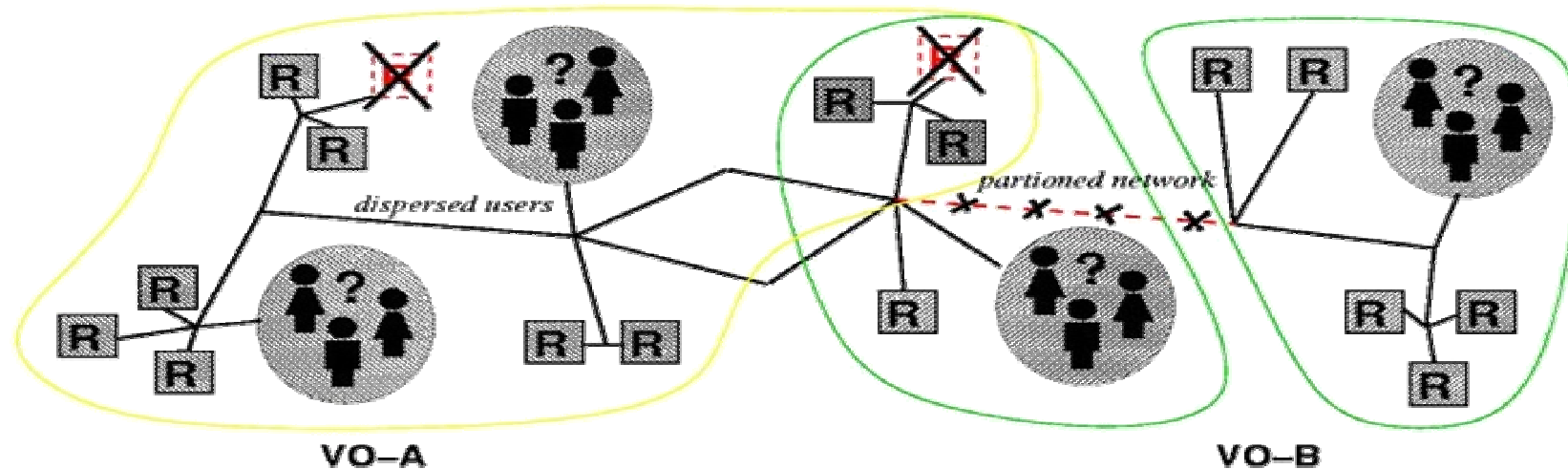
Introduction

Overview of Grid Computing



The Grid

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





The Grid

„A computational grid is a hardware and software infrastructure that provides dependable, consistent, pervasive, and inexpensive access to high-end computational capabilities.“

- 1) Enable integration of distributed resources
- 2) Using general-purpose protocols & infrastructure
- 3) To achieve better-than-best-effort service



The Grid (2)

- Dynamically link resources/services
- From collaborators, customers, eUtilities, ...
(members of evolving “virtual organization”)
- Into a “virtual computing system”
- Dynamic, multi-faceted system spanning institutions and industries
- Configured to meet instantaneous needs, for:
 - Multi-faceted QoX (Quality of Experience) for demanding workloads
- Security, performance, reliability, ...

Why Does the Globus Toolkit Exist?



- Problems were big enough that they required people in several organizations to collaborate & share computing resources, data, &/or instruments
- While helping to build/integrate a diverse range of applications, the same problems kept showing up over and over again
 - Different security systems
 - Different scheduling/execution mechanisms
 - Different storage systems
 - Different monitoring/status/event systems



What Kinds of Applications?

- Computation intensive
- Interactive simulation (climate modeling)
- Large-scale simulation (galaxy formation, gravity waves, battlefield simulation)
- Engineering (parameter studies, linked models)
- Data intensive
- Experimental data analysis (high energy physics)
- Image, sensor analysis (astronomy, climate)
- Distributed collaboration
- Online instruments (microscopes, x-ray devices)
- Remote visualization (climate studies, biology)
- Engineering (structural testing, chemical)



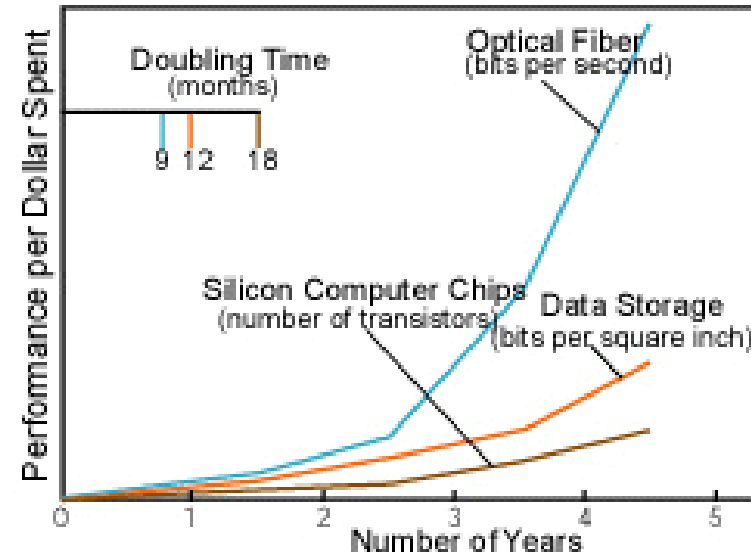
Why Now?

- Moore's law improvements in computing produce highly functional endsystems
- The Internet and burgeoning wired and wireless provide universal connectivity
- Changing modes of working and problem solving emphasize teamwork, computation
- Network exponentials produce dramatic changes in geometry and geography



Network Exponentials

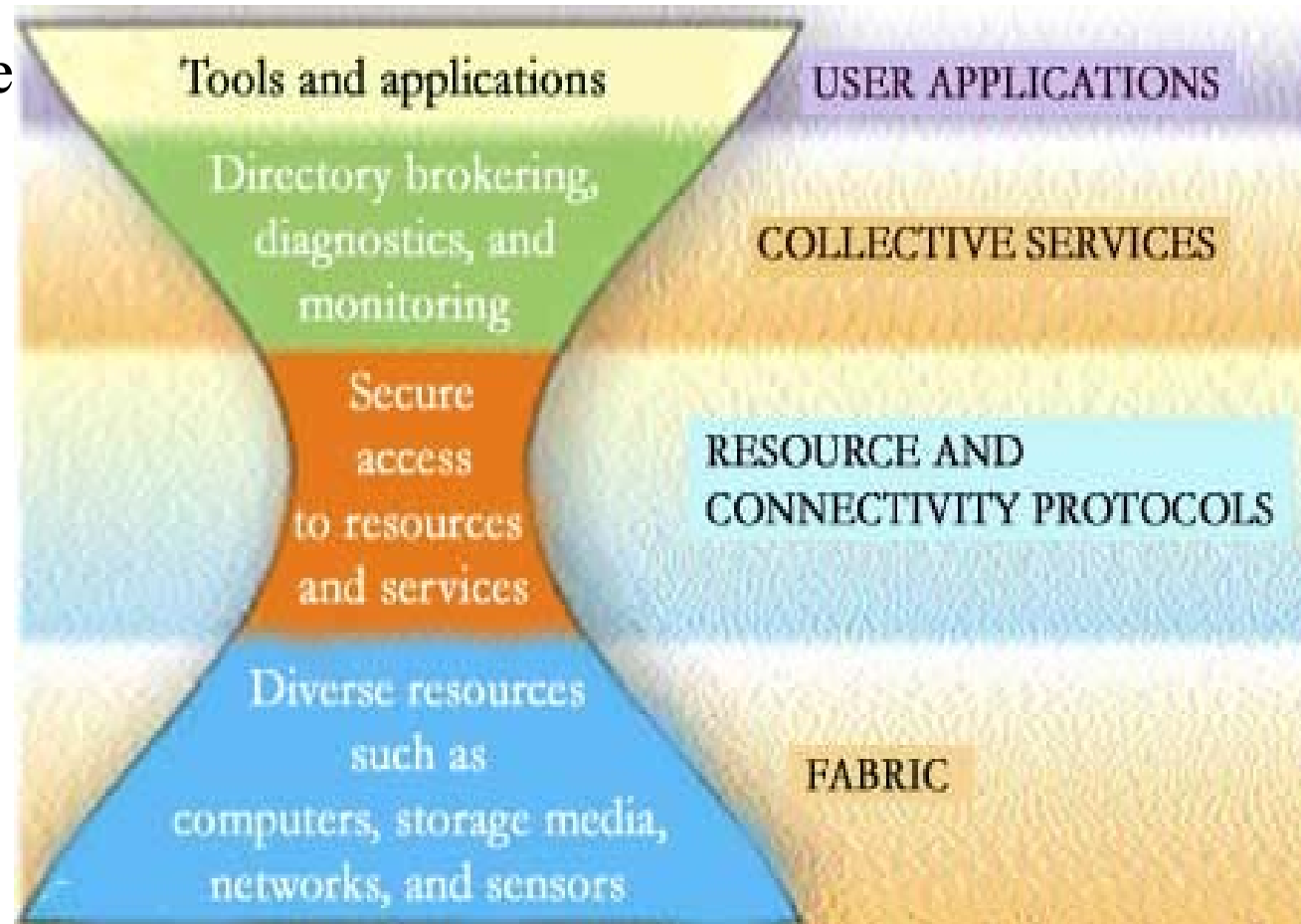
- Network vs. computer performance
 - Computer speed doubles every 18 months
 - Network speed doubles every 9 months
 - Difference = order of magnitude per 5 years
- 1986 to 2000
 - Computers: x 500
 - Networks: x 340,000
- 2001 to 2010
 - Computers: x 60
 - Networks: x 4000





Forget Homogeneity!

- Trying to force homogeneity on users is futile: everyone has their own preferences, sometimes even *dogma*
- The Internet provides the model ...





And thus, the Globus Toolkit

- The Globus Toolkit is a collection of solutions to problems that frequently arise when trying to build collaborative distributed applications



What Types of Problems?

- Your system administrators can't agree on a uniform authentication system, but you have to allow your users to authenticate once (using a single password) then use services on all systems, with per-user accounting
- You need to be able to offload work during peak times to systems at other companies, but the volume of work they'll accept changes from day-to-day



What Types of Problems?

- You and your colleagues have 6000 datasets from the past 50 years of studies that you want to start sharing, but no one is willing to submit the data to a centrally-managed storage system or database
- You need to run 24 experiments that each use six large-scale physical experimental facilities operating together in real time



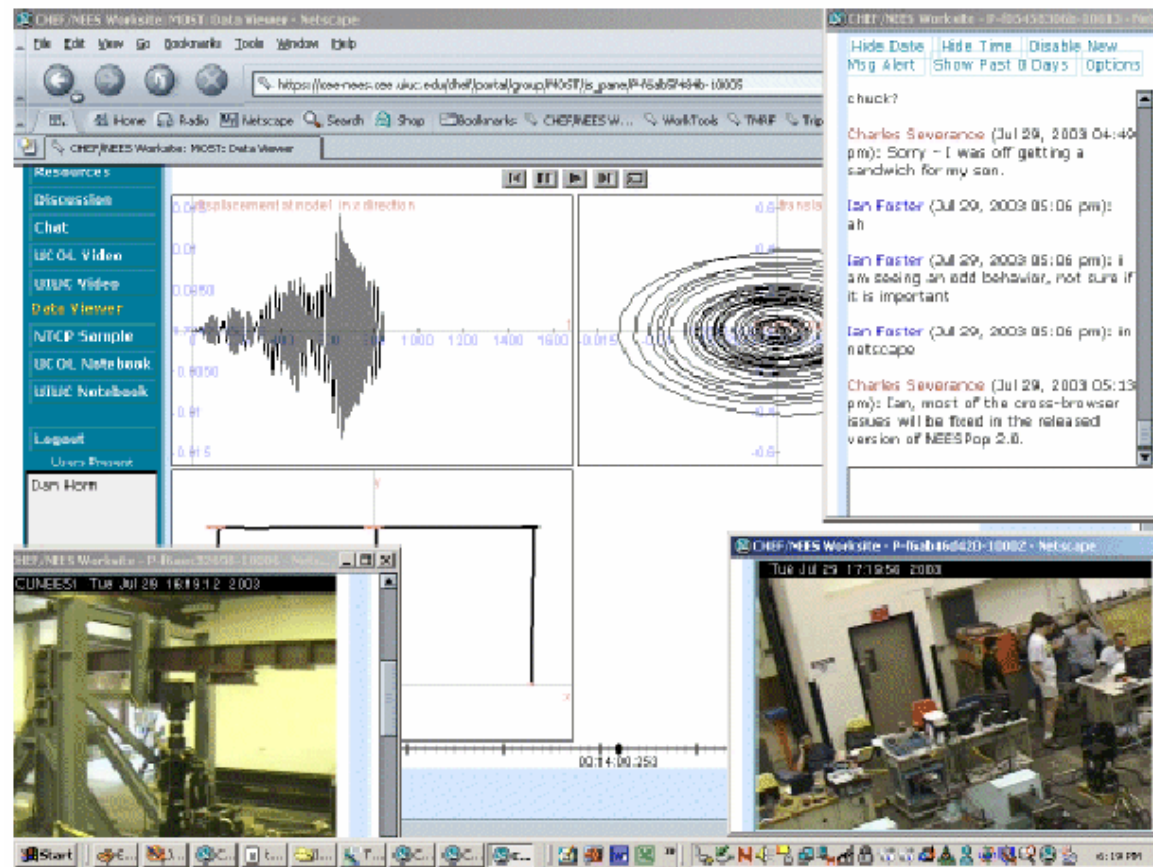
What Types of Problems?

- Security
- Monitoring/discovery
- Accessing computing/processing power
- Moving data
- Managing data
- Managing systems
- System packaging/distribution

What End Users Need



Secure,
reliable, on-
demand
access to
data,
software,
people, and
other
resources
(ideally all via
a Web
Browser!)





How it *Really* Happens

- Implementations are provided by a mix of
 - Application-specific code
 - “Off the shelf” tools and services
 - Tools and services from the Globus Toolkit
 - GT-compatible tools and services from the Grid community
- Glued together by ...
 - Application development
 - System integration
- Deployed and supported

Globus Toolkit:



“Standard Plumbing” for the Grid

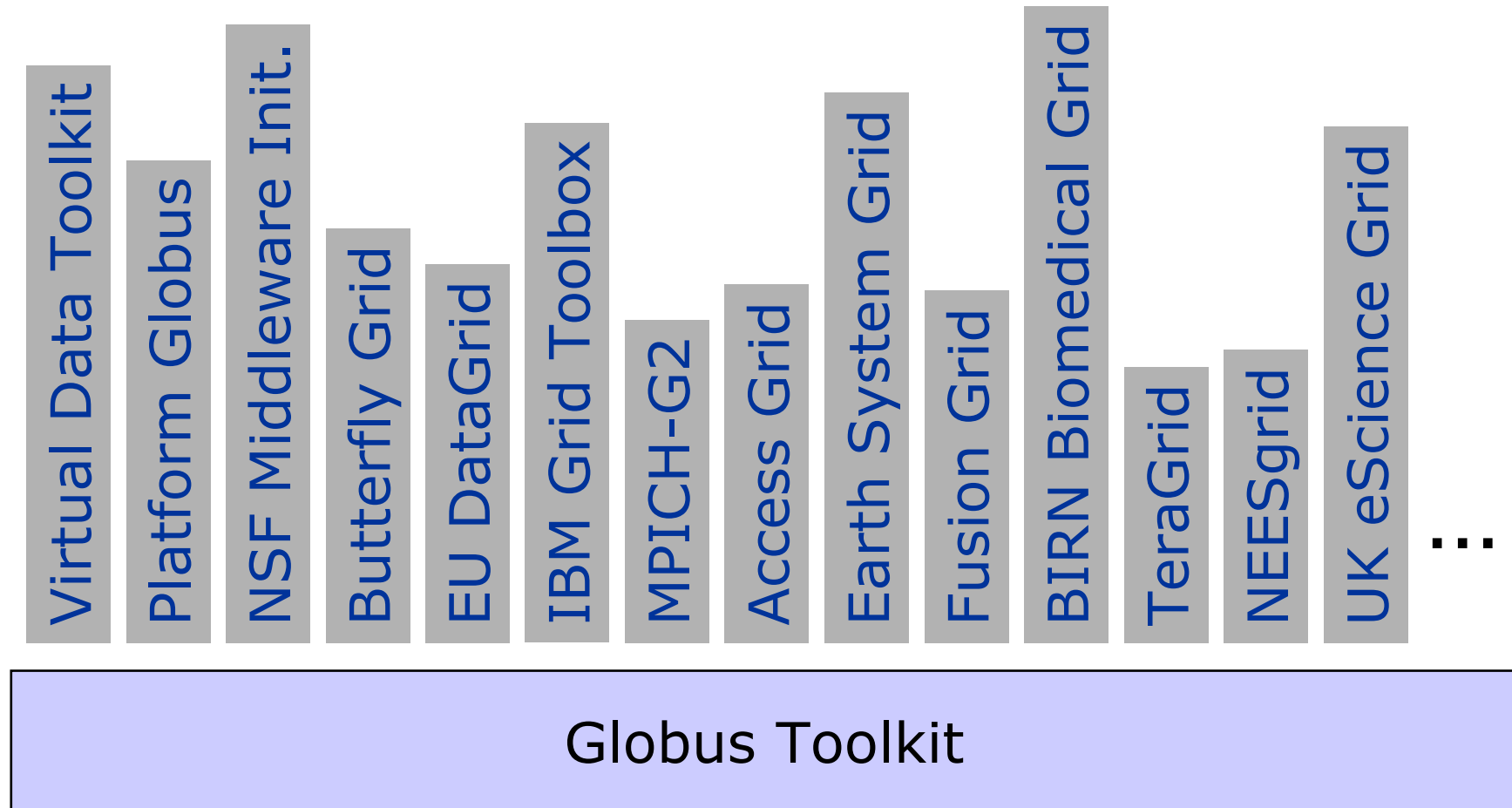
- *Not* turnkey solutions, but *building blocks* and *tools* for application developers and system integrators
 - Some components (e.g., file transfer) go farther than others (e.g., remote job submission) toward end-user relevance
- Since these solutions exist and others are already using them (and they’re free), it’s easier to reuse than to reinvent
 - And compatibility with other Grid systems comes for free!

What You Get in the Globus Toolkit



- OGSi(3.x)/WSRF(4.x) Core
 - Used to develop & run custom WS-based services (Java, C/C++)
- Basic Grid services
 - GRAM resource management, security, data management, registries, etc., etc.
 - Both WS & pre-WS implementations
- Developer APIs
 - C/C++ libraries and Java classes for building Grid-aware applications and tools
- Tools and examples
 - Tools & examples based on developer APIs

GT-Based Grid Tools & Solutions





Prerequisite Hardware/Software

- Most Grids make heavy use of Unix/Linux
 - Open source and easily extensible
 - Decent price/perf./reliability/native security
 - More familiar to original Grid community
- .NET good for OGSI/WSRF-based work
 - A benefit of being WS standards compatible
- Hardware depends on application goals
 - Clusters for compute-intensive apps, parallel work (including data striping), databases
 - Mass storage for data-intensive apps
 - Networking for communication-intensive apps (data transfer, videoconf., remote viz, etc.)



Important Planning Considerations

- All Grid technology is evolving rapidly
 - Web services standards
 - Grid interfaces
 - Grid implementations
 - Grid hosting services (ASP, SSP, etc.)
- Community is important!
- Best practices (GGF, OASIS, etc.)
- Open source (Linux, Axis, Globus, etc.)
- Application of community standards is vital
 - Increases leverage
 - Mitigates (a bit) effects of rapid evolution
 - Pave way for future integration/partnership



Selected Major Grid Projects







Name	URL & Sponsors	Focus
Access Grid	www.mcs.anl.gov/FL/accessgrid ; DOE, NSF	Create & deploy group collaboration systems using commodity technologies
BlueGrid	IBM	Grid testbed linking IBM laboratories
DISCOM	www.cs.sandia.gov/discom DOE Defense Programs	Create operational Grid providing access to resources at three U.S. DOE weapons laboratories
DOE Science Grid	sciencegrid.org DOE Office of Science	Create operational Grid providing access to resources & applications at U.S. DOE science laboratories & partner universities
Earth System Grid (ESG)	earthsystemgrid.org DOE Office of Science	Delivery and analysis of large climate model datasets for the climate research community
European Union (EU) DataGrid	eu-datagrid.org European Union	Create & apply an operational grid for applications in high energy physics, environmental science, bioinformatics

21.07.2004

Grid Computing Introduction



Selected Major Grid Projects

Name	URL/Sponso	Focus
EuroGrid, Grid Interoperability (GRIP)  <i>New</i>	eurogrid.org European Union	Create tech for remote access to supercomp resources & simulation codes; in GRIP, integrate with Globus Toolkit™
Fusion Collaborator  <i>New</i>	fusiongrid.org DOE Off. Science	Create a national computational collaboratory for fusion research
Globus Project™ 	globus.org DARPA, DOE, NSF, NASA, Msoft	Research on Grid technologies; development and support of Globus Toolkit™; application and deployment
GridLab  <i>New</i>	gridlab.org European Union	Grid technologies and applications
GridPP  <i>New</i>	gridpp.ac.uk U.K. eScience	Create & apply an operational grid within the U.K. for particle physics research
Grid Research Integration Dev. & Support Center  <i>New</i>	grids-center.org NSF	Integration, deployment, support of the NSF Middleware Infrastructure for research & education



Selected Major Grid Projects

Name	URL/Sponsor	Focus
Grid Application Dev. Software	hipersoft.rice.edu/grads ; NSF	Research into program development technologies for Grid applications
Grid Physics Network	griphyn.org NSF	Technology R&D for data analysis in physics expts: ATLAS, CMS, LIGO, SDSS
Information Power Grid	ipg.nasa.gov NASA	Create and apply a production Grid for aerosciences and other NASA missions
International Virtual Data Grid Laboratory	ivdgl.org NSF	Create international Data Grid to enable large-scale experimentation on Grid technologies & applications
Network for Earthquake Eng. Simulation Grid	neesgrid.org NSF	Create and apply a production Grid for earthquake engineering
Particle Physics Data Grid	ppdg.net DOE Science	Create and apply production Grids for data analysis in high energy and nuclear physics experiments



Selected Major Grid Projects

Name	URL/Sponsor	Focus
TeraGrid	teragrid.org NSF	U.S. science infrastructure linking four major resource sites at 40 Gb/s
UK Grid Support Center	grid-support.ac.uk U.K. eScience	Support center for Grid projects within the U.K.
Unicore	BMBFT	Technologies for remote access to supercomputers

Examples of



Production Grid Deployments

- Grid3/iVDGL (US) **P**
 - 22 sites, O(3000) CPUs, 2 countries
- LHC Computing Grid **P**
 - 25 sites, international
 - High energy physics
- NorduGrid **P**
 - 24 clusters, 724 CPUs, 6 countries; physics
- NASA IPG **P**
 - 4 sites, O(3000) CPUs
 - Aeronautics
- NEESgrid (prod. **W** 2004)
 - Instruments, data, compute, collaborative
 - Earthquake eng.
- TeraGrid (prod. Jan **P** 04)
 - 5 sites, expanding



The Globus Toolkit

Definitions



Some Important Definitions

- Resource
 - Network protocol
 - Network enabled service
 - Application Programmer Interface (API)
 - Software Development Kit (SDK)
 - Syntax
-
- Not discussed, but important: policies



Resource

- An entity that is to be shared
 - E.g., computers, storage, data, software
- Does not have to be a physical entity
 - E.g., Condor pool, distributed file system, ...
- Defined in terms of interfaces, not devices
 - E.g. scheduler such as LSF and PBS define a compute resource
 - Open/close/read/write define access to a distributed file system, e.g. NFS, AFS, DFS



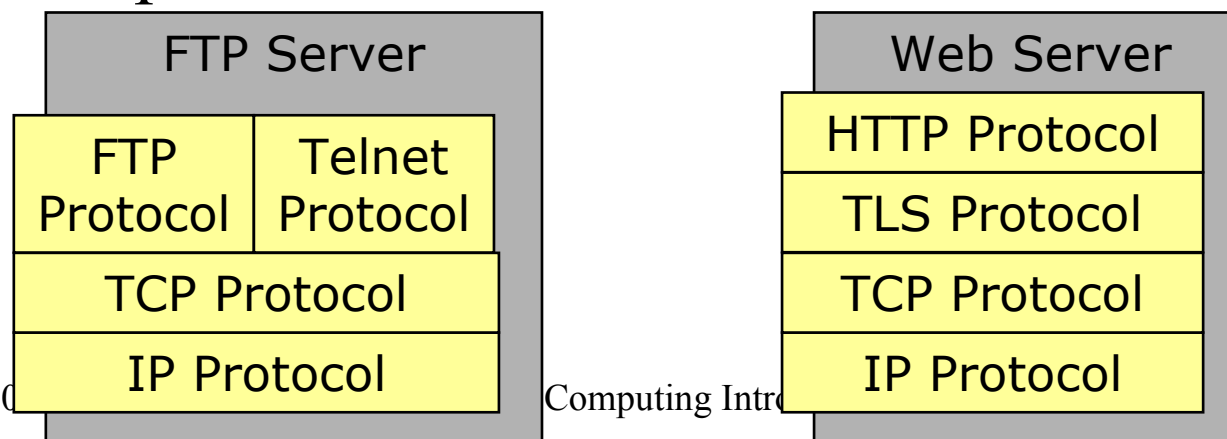
Network Protocol

- 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



Network Enabled Services

- Implementation of a protocol that defines a set of capabilities
 - Protocol defines interaction with service
 - All services require protocols
 - Not all protocols are used to provide services (e.g. IP, TLS)
- Examples: FTP and Web servers



Application Programming Interface



- A specification for a set of routines to facilitate application development
- Refers to definition, not implementation
- E.g., there are many implementations of MPI
- Spec often language-specific (or IDL)
- Routine name, number, order and type of arguments; mapping to language constructs
- Behavior or function of routine
- Examples
- GSS API (security), MPI (message passing)



Software Development Kit

- A particular instantiation of an API
- SDK consists of libraries and tools
 - Provides implementation of API specification
- Can have multiple SDKs for an API
- Examples of SDKs
 - MPICH, Motif Widgets



Syntax

- Rules for encoding information, e.g.
- XML, Condor ClassAds, Globus RSL
- X.509 certificate format (RFC 2459)
- Cryptographic Message Syntax (RFC 2630)
- Distinct from protocols
- One syntax may be used by many protocols (e.g., XML); & useful for other purposes
- Syntaxes may be layered
- E.g., Condor ClassAds -> XML -> ASCII
- Important to understand layerings when comparing or evaluating syntaxes

A Protocol can have Multiple APIs



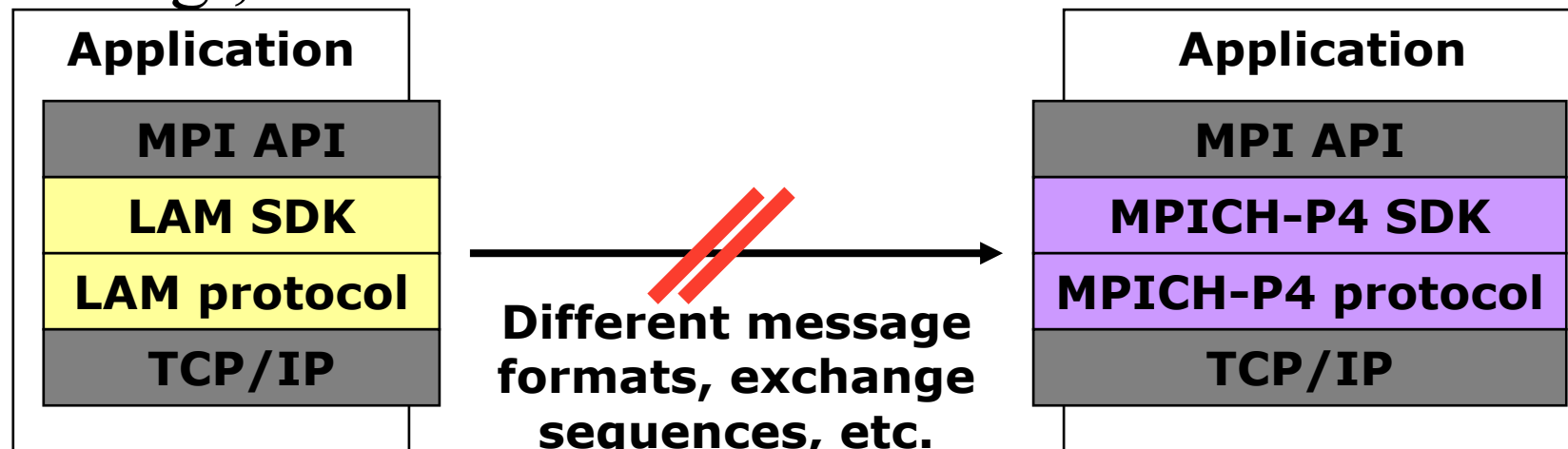
- TCP/IP APIs include BSD sockets, Winsock, System V streams, ...
- The protocol provides interoperability: programs using different APIs can exchange information
- I don't need to know remote user's API





An API can have Multiple Protocols

- MPI 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



21.07.2004

Grid Computing Introduction

APIs and Protocols are Both Important



- Standard APIs/SDKs are important
 - They enable application *portability*
 - But w/o standard protocols, interoperability is hard (every SDK speaks every protocol?)
- Standard protocols are important
 - Enable cross-site *interoperability*
 - Enable shared infrastructure
 - But w/o standard APIs/SDKs, application portability is hard (different platforms access protocols in different ways)



The Globus Toolkit

Grid Architecture



Why Discuss Architecture?

- Descriptive
 - Provide a common vocabulary for use when describing Grid systems
- Guidance
 - Identify key areas in which services are required
- Prescriptive
 - Define standard “Intergrid” protocols and APIs to facilitate creation of interoperable Grid systems and portable applications



One View of 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
- Etc.
- Etc.
- ...



Another View: “Three Obstacles to Making Grid Computing Routine”

1) New approaches to problem solving

- Data Grids, distributed computing, peer-to-peer, collaboration grids,

2) Structuring and writing programs

- Abstractions, tools

Programming Problem

3) Enabling resource sharing across distinct institutions

Systems Problem

- Resource discovery, access, reservation, allocation; authentication, authorization, policy; communication; fault detection and notification;

21.07.2004

Grid Computing Introduction

...

Programming & Systems Problems



- The programming problem
 - Facilitate development of sophisticated apps
 - Facilitate code sharing
 - Requires programming environments
 - APIs, SDKs, tools
- The systems problem
 - Facilitate coordinated use of diverse resources
 - Facilitate infrastructure sharing
 - e.g., certificate authorities, information services
 - Requires systems
 - protocols, services



The Systems Problem: Resource Sharing Mechanisms That ...

- Address security and policy concerns of resource owners and users
- Are flexible enough to deal with many resource types and sharing modalities
- Scale to large number of resources, many participants, many program components
- Operate efficiently when dealing with large amounts of data & computation



Aspects of the Systems Problem

- Need for interoperability when different groups want to share resources
 - Diverse components, policies, mechanisms
 - E.g., standard notions of identity, means of communication, resource descriptions
- Need for shared infrastructure services to avoid repeated development, installation
 - E.g., one port/service/protocol for remote access to computing, not one per tool/appln
 - E.g., Certificate Authorities: expensive to run
- A common need for protocols & services



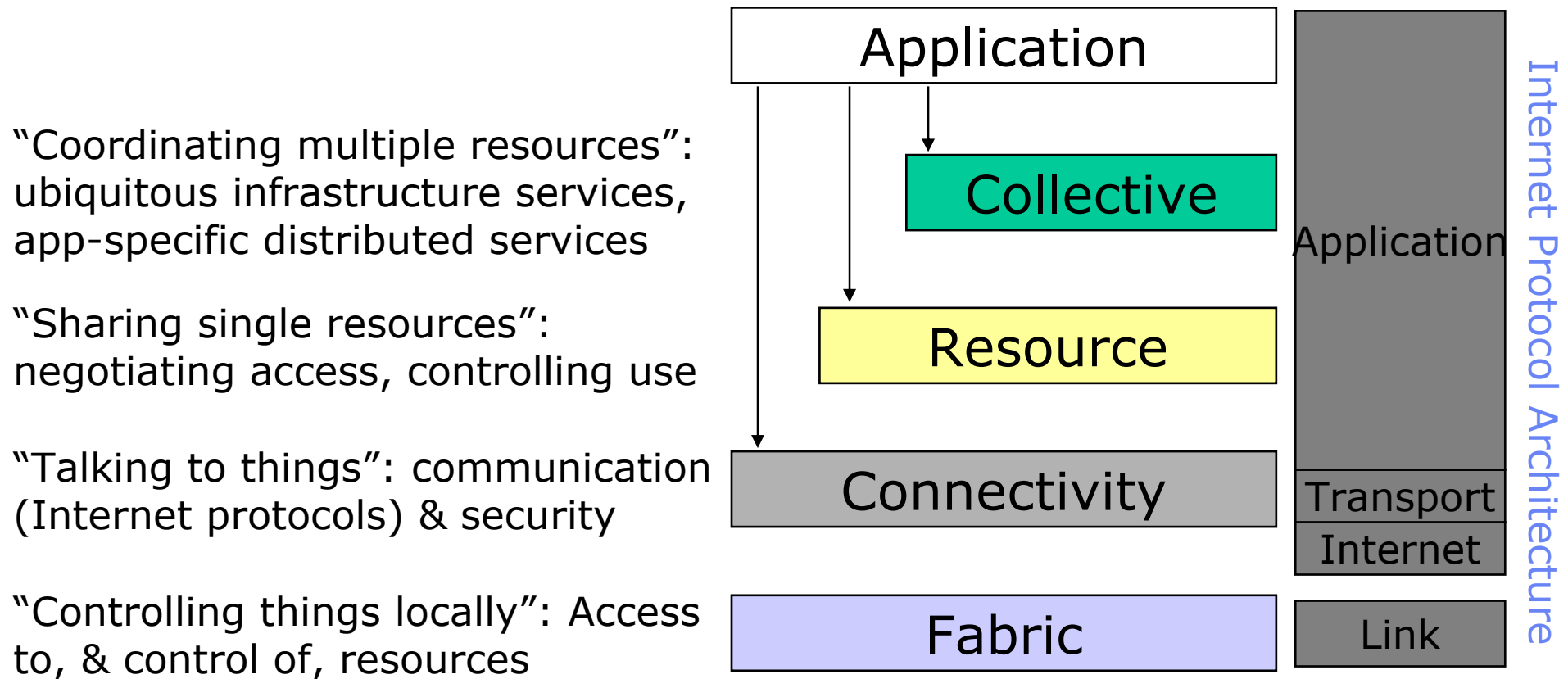
Hence, a Protocol-Oriented View of Grid Architecture, that Emphasizes ...

- Development of Grid protocols & services
 - Protocol-mediated access to remote resources
 - New services: e.g., resource brokering
 - “On the Grid” = speak Intergrid protocols
 - Mostly (extensions to) existing protocols
- Development of Grid APIs & SDKs
 - Interfaces to Grid protocols & services
 - Facilitate application development by supplying higher-level abstractions
- The (hugely successful) model is the Internet



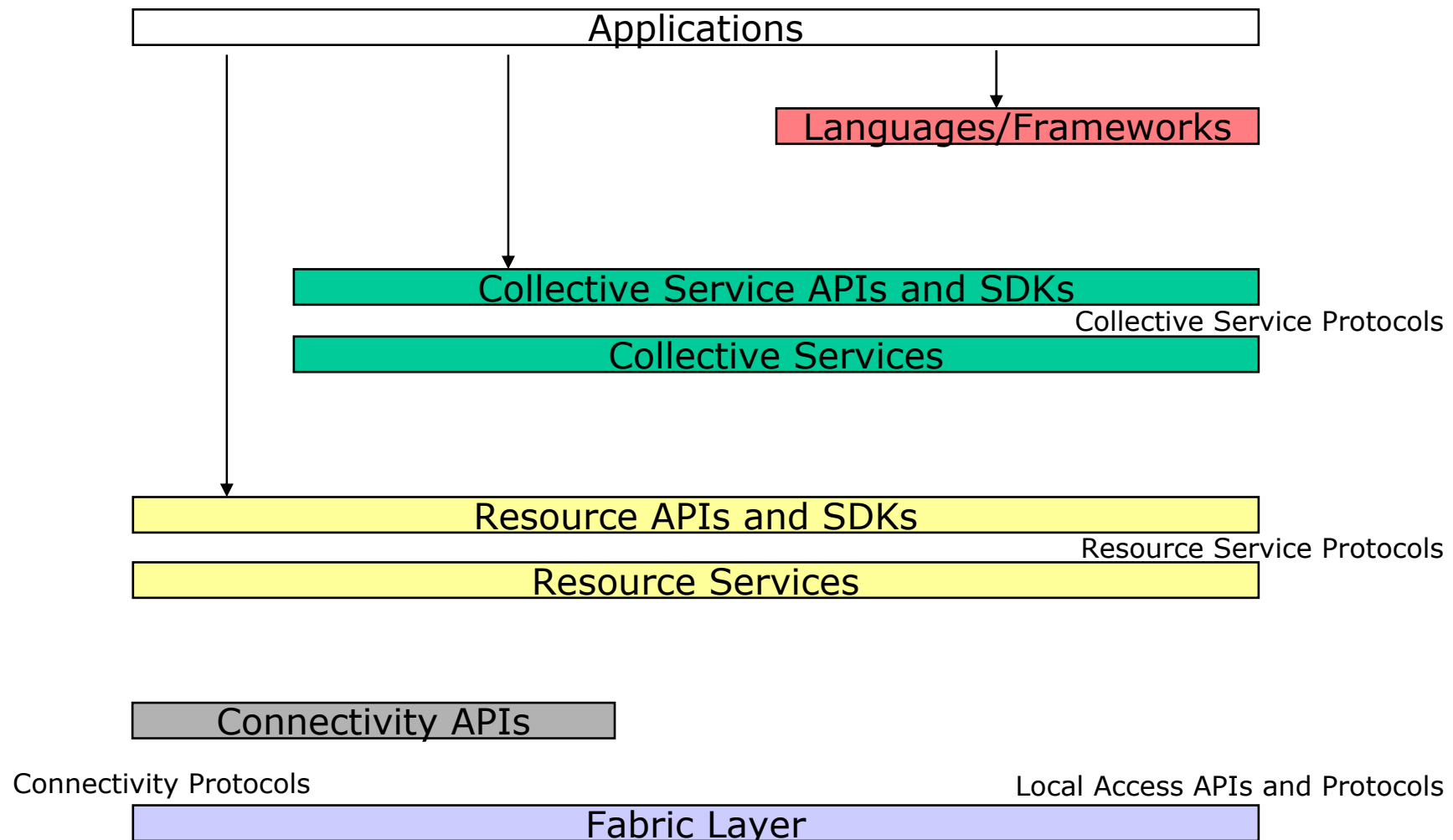
Layered Grid Architecture

(By Analogy to Internet Architecture)





Protocols, Services, and APIs Occur at Each Level





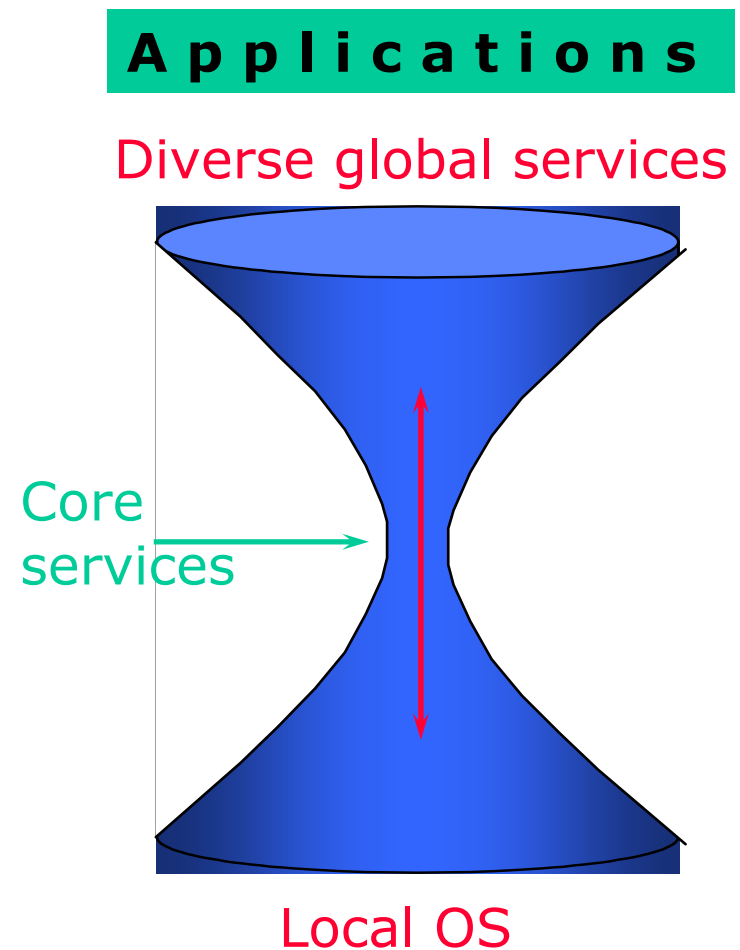
Important Points

- Built on Internet protocols & services
 - Communication, routing, name resolution, etc.
- “Layering” here is conceptual, does not imply constraints on who can call what
 - Protocols/services/APIs/SDKs will, ideally, be largely self-contained
 - Some things are fundamental: e.g., communication and security
 - But, advantageous for higher-level functions to use common lower-level functions



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



Fabric Layer

Protocols & 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., 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 Protocols & 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

Protocols & Services

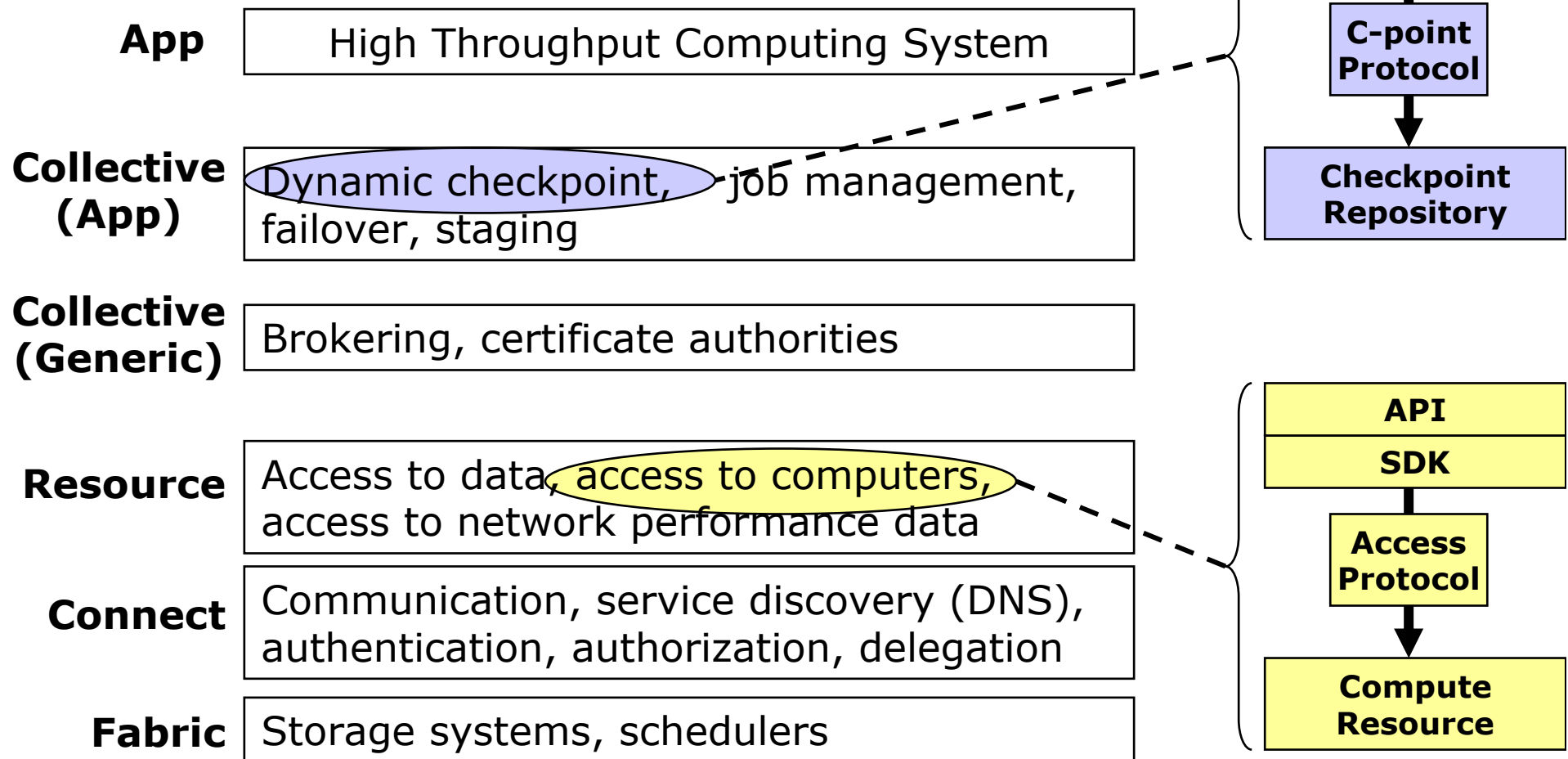
- Grid Resource Allocation Management (GRAM)
- Remote allocation, reservation, monitoring, control of compute resources
- GridFTP protocol (FTP extensions)
- High-performance data access & transport
- Grid Resource Information Service (GRIS)
- Access to structure & state information
- Others emerging: Catalog access, code repository access, accounting, etc.
- All built on connectivity layer: GSI & IP



Collective Layer Protocols & Services

- Index servers aka metadirectory services
 - Custom views on dynamic resource collections assembled by a community
- Resource brokers (e.g., Condor Matchmaker)
 - Resource discovery and allocation
- Replica catalogs
- Replication services
- Co-reservation and co-allocation services
- Workflow management services
- Etc.

Example: High-Throughput Computing System





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, ...



The Globus Toolkit

The Programming Problem



The Programming Problem

- But how do I develop robust, secure, long-lived, well-performing applications for dynamic, heterogeneous Grids?
- I need, presumably:
 - Abstractions and models to add to speed/robustness/etc. of development
 - Tools to ease application development and diagnose common problems
 - Code/tool sharing to allow reuse of code components developed by others



Grid Programming Technologies

- “Grid applications” are incredibly diverse (data, collaboration, computing, sensors, ...)
 - Seems unlikely there is one solution
- Most applications have been written “from scratch,” with or without Grid services
- Application-specific libraries have been shown to provide significant benefits
- No new language, programming model, etc., has yet emerged that transforms things
 - But certainly still quite possible



Examples of Grid Programming Technologies

- MPICH-G2: Grid-enabled message passing
- CoG Kits, GridPort: Portal construction, based on N-tier architectures
- GDMP, Data Grid Tools, SRB: replica management, collection management
- Condor-G: workflow management
- Legion: object models for Grid computing
- Cactus: Grid-aware numerical solver framework
 - Note tremendous variety, application focus

MPICH-G2: A Grid-Enabled MPI



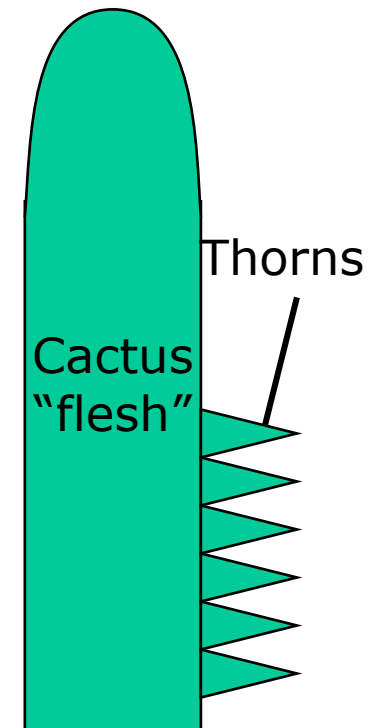
- A complete implementation of the Message Passing Interface (MPI) for heterogeneous, wide area environments
- Based on the Argonne MPICH implementation of MPI (Gropp and Lusk)
- Requires services for authentication, resource allocation, executable staging, output, etc.
- Programs run in wide area without change
- Modulo accommodating heterogeneous communication performance
- See also: MetaMPI, PACX, STAMPI, MAGPIE



Cactus

(Allen, Dramlitsch, Seidel, Shalf, Radke)

- Modular, portable framework for parallel, multidimensional simulations
- Construct codes by linking
 - Small core (flesh): mgmt services
 - Selected modules (thorns): Numerical methods, grids & domain decomp, visualization and steering, etc.
- Custom linking/configuration tools
- Developed for astrophysics, but not astrophysics-specific



High-Throughput Computing and Condor



- High-throughput computing
 - CPU cycles/day (week, month, year?) under non-ideal circumstances
 - “How many times can I run simulation X in a month using all available machines?”
- Condor converts collections of distributively owned workstations and dedicated clusters into a distributed **high-throughput** computing facility
- Emphasis on policy management and reliability



Object-Based Approaches

- Grid-enabled CORBA
 - NASA Lewis, Rutgers, ANL, others
 - CORBA wrappers for Grid protocols
 - Some initial successes
- Legion
 - U.Virginia
 - Object models for Grid components (e.g., “vault”=storage, “host”=computer)



Portals

- N-tier architectures enabling thin clients, with middle tiers using Grid functions
- Thin clients = web browsers
- Middle tier = e.g. Java Server Pages, with Java CoG Kit, GPDK, GridPort utilities
- Bottom tier = various Grid resources
- Numerous applications and projects, e.g.
- Unicore, Gateway, Discover, Mississippi Computational Web Portal, NPACI Grid Port, Lattice Portal, Nimrod-G, Cactus, NASA IPG Launchpad, Grid Resource Broker, ...