Resource Discovery and Information Services

John Watt

http://csperkins.org/teaching/2004-2005/gc5/

Overview

- Introduction
- Requirements
- Architecture
- MDS2
  - GRISs and GIISs
- MDS3
  - serviceData
- Index Service
- Summary

Information Services

- System information is critical to the operation of the grid and construction of applications
  - What resources are available?
    - Resource discovery
  - What is the “state” of the grid?
    - Resource selection
  - How to optimize resource use
    - Application configuration and adaptation

- We need a general information infrastructure to answer these questions

Information Services

- Any grid software infrastructure should provide fundamental mechanisms for
  - Discovery
  - Monitoring
  - Planning
  - Adapting application behaviour
- Design to support the initial discovery and ongoing monitoring of the existence and characteristics of resources, services, computations, and other entities.

Requirements

- Distribution
  - Information sources are necessarily distributed and individual sources are subject to error/failure
  - Any information delivered will be old

- Information producers should use timestamps and time-to-live metadata
- Information services should transport information as rapidly and efficiently as possible

Requirements

- Failure management
  - Information services should behave robustly in the face of failure of any component
  - Failure should not prevent users from gaining information, even if it is partial or inconsistent

- Information services should be as distributed as possible.
- Information services components should be built under the assumption that they will fail!!

Requirements

- Diversity
  - A new virtual organization may involve many entities and have unique requirements for discovery and monitoring

- Define the standard discovery and enquiry mechanisms that must be supported by any grid entity

Requirements

- Security
  - Information is often provided with restrictions

- Must have robust authentication and authorisation mechanisms that information owners will trust
- Providers may wish to assert policy over which virtual organisations they are prepared to join

Architecture

- There are two fundamental entities in information services

- Information Providers
  - Common VO-neutral infrastructure
  - Providing access to detailed dynamic information about Grid entities
- Specialised aggregate directory services
  - Specialised VO-specific views of federated resources, services etc.

Security Issues

- The information providers and aggregate directory have the same access policy (providers trust the directory)
  - Information providers limit the information that is available to an aggregate directory
  - Information provider makes no other information known, other than its existence
  - Information provider places no restrictions on the information given out

LDAP

- Lightweight Directory Access Protocol
  - Version of X.500 DAP protocol
  - Supports distributed storage/access (referrals)
  - Supports authentication and authorisation
- Defines
  - Network protocol for accessing directory contents
  - Information model defining form of information
  - Namespace defining how information is referenced and organised
    - e.g. Domain Name Service (DNS)

LDAP

- Information model and namespace are based on entries
- An entry is used to store attributes
- An attribute is an associated type and can have one or more values
- Each entry in a namespace has a distinguished name which allows it to be identified easily

Implementations

- Globus toolkit contains an implementation of a grid information service
  - Meta Directory Service (MDS)
    - A basis for configuration and adaptation in heterogeneous, dynamic environments
  - Uniform, flexible access to information
  - Scalable, efficient access to dynamic data
  - Access to multiple information sources
  - Decentralised maintenance

MDS Protocols

- Grid Information Protocol (GRIP)
  - Users, aggregate directories and applications use GRIP to obtain information from an information provider about the entities the provider possesses
  - Discovery and Enquiry are supported
  - Implemented using OpenLDAP

MDS Protocols

- Grid Registration Protocol
  - Used by information provider to notify the aggregate directory of its availability for indexing
  - Or used by aggregate directory to invite an information provider to join a VO
  - GRRP is a “soft-state” protocol
    - Information can be discarded unless refreshed by a stream of subsequent notifications
    - Makes it resistant to failure

GRIS

- GRIS stands for “Grid Resource Information Service”
  - Implemented as a server which runs on each resource
    - Given the resource DNS, one can find the GRIS server (implemented as an OpenLDAP server backend)
  - Provides resource specific information by parsing each GRIP request
    - Much of this information may be dynamic
      - Load, process information, storage info etc
      - GRIS gathers this info “on-demand”

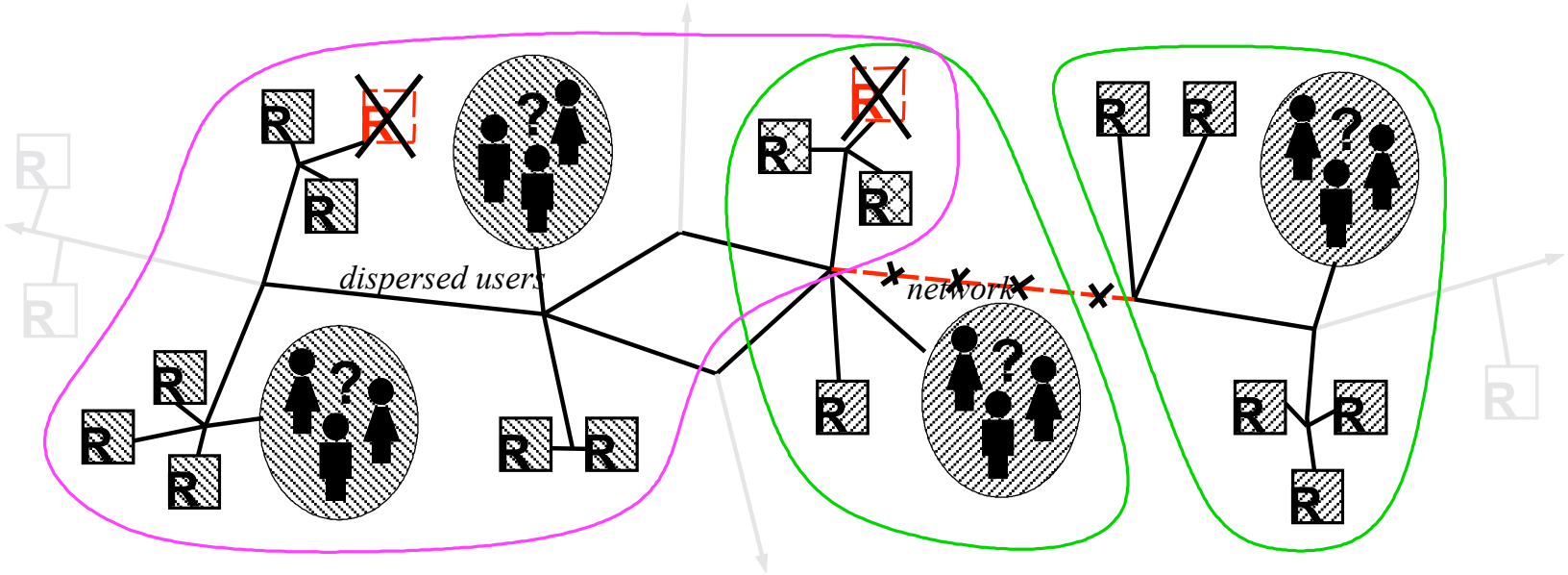
GRIS

- “White Pages” lookup of resource information
  - How much memory does the machine have?
- “Yellow Pages” lookup of resource options
  - Which queues on machines allow large jobs?

GIIS

- GIIS stands for Grid Index Information Service
  - GIIS describes a class of servers
    - Gathers information from multiple GRIS servers via GRRP messages
    - Each GIIS is optimised for particular queries
    - Relative to web search engines
  - GIIS provides a “caching” service much like a web search engine. Resources register with GIIS and GIIS pulls information from them when requested by a client and the cache has expired.
  - GIIS provides the collective-level indexing/searching function.

MDS2 Architecture

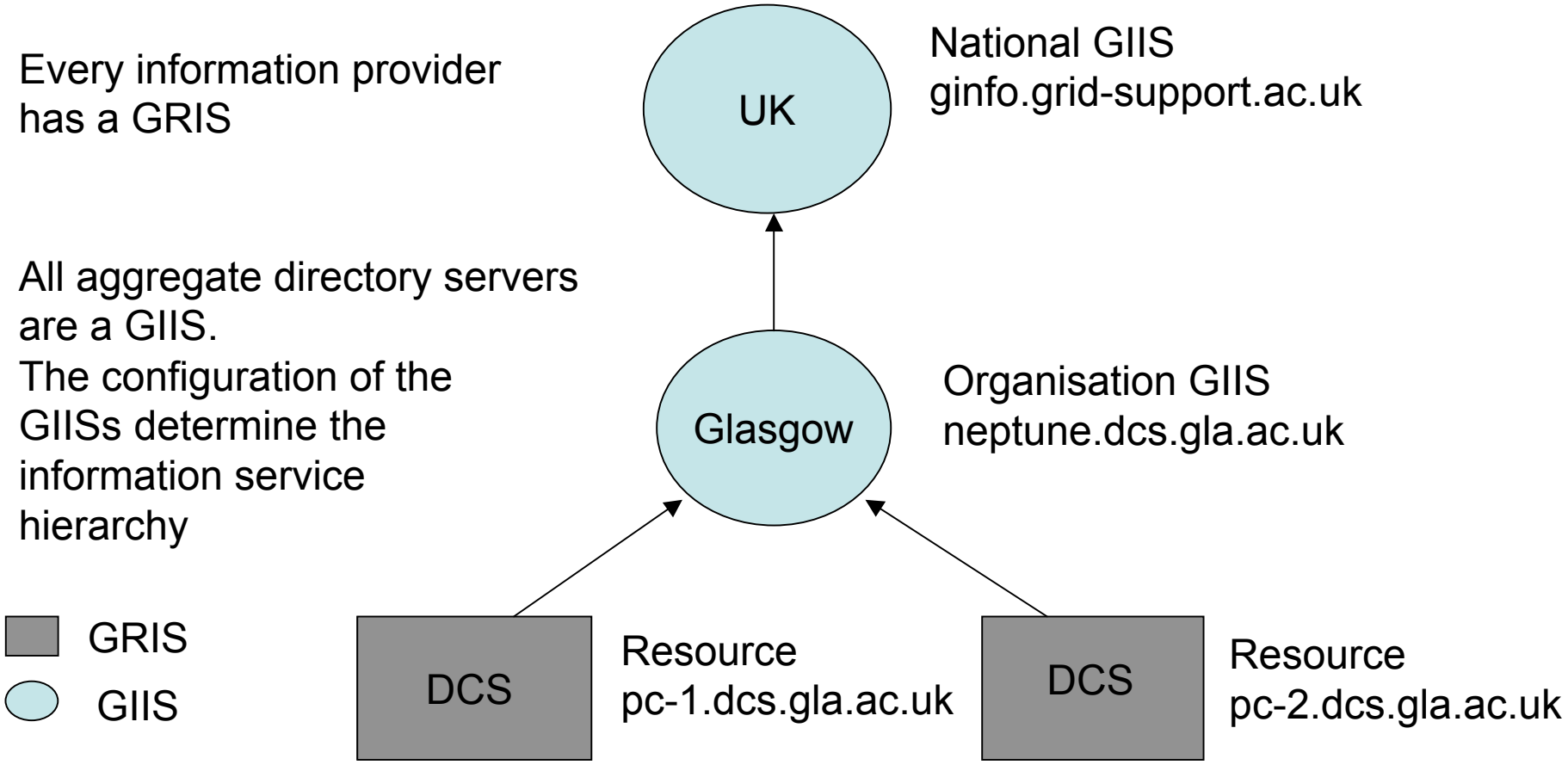


- GRIS – resource
- GIIS – multi-resource index
- LDAP protocols (OpenLDAP)
- Pull only

GRIS/GIIS Hierarchy

Every information provider has a GRIS

All aggregate directory servers are a GIIS.
The configuration of the GIISs determine the information service hierarchy



MDS3

- MDS3 is the Information Service for the Globus Toolkit 3
- Main interesting areas of MDS3
  - serviceData
  - Query mechanisms
  - XPath query support
  - Host status info
  - Index Service
  - Hosting Environment components

MDS3

- MDS3 is built on the Open Grid Services Infrastructure
- Recap:
  - Grid services add extra features to Web Service standards (WSDL, SOAP)
    - But still can use standard web service components to build grid services (e.g. Using Apache Axis with GT3)
  - Most important extra feature for MDS is “serviceData”

serviceData

- serviceData is XML data published by every grid service that provides some representation of its internal state
  - Each piece of serviceData is called a “serviceData Element” (SDE)
    - (XML of arb. complexity)
  - A grid service has a logical entity called the “serviceData set” which is a collection of all the serviceData Elements
    - e.g. State of a host (or cluster) exposed as a single SDE (by GRAM), or job status
    - Similar to MDS2 GRIS, but in each service rather than once per resource

Getting the serviceData

- Two ways to retrieve service data
  - pull or push.

- Pull (send 1 query and receive 1 response only)
  - findServiceData operation
    - OGSI wants it extensible and support “queryByName”
    - GT3 has defined “queryByXPath” – coming up

Getting the serviceData

- Push (send a subscription expression and receive notifications [callbacks] at appro. times)
  - subscribeByServiceDataNames
  - Extensible – BUT optional in OGSI

- A client can discover which query and subscription mechanism is supported by a service by additional query listed in well-known SDEs
  - findServiceDataExtensibility
  - subscribeExtensibility

serviceDataNames

- The basic query and subscription types for OGSI
  - queryByServiceDataNames
    - guaranteed to be supported by ANY Grid Service
  - subscribeByServiceDataNames
- Parameter returned is the list of SDE names
  - Plus some additional info for subscriptions
    - Min/max frequency, sink, ttl
  - Entire SDE is returned, so if SDE is large, the return/notification is large also
    - e.g. if we have a cluster SDE representing status of 300 nodes, every notification must include the entire state of all 300 nodes
    - RFT has SDE of all outstanding jobs, so queryByName would give us the state of ALL jobs each time

XPath queries

- So findServiceData is good for small SDEs, bad for large SDEs.

- How can we get smaller results from findServiceData?
  - Use a different query type
  - No other type is described in the OGSI specification
  - Globus have built their own query type using XPath

XPath queries

- XPath is a W3C XML query language
  - Based on XALAN XPath library
  - Any service built on the GT3 OGSA core has support for this functionality

- Globus have defined a query type
  - Input parameters are an SDE and an XPath query
  - Output is the result of evaluating the XPath query against the SDE set
    - So you get a list of 0 or more elements that matched the query

XPath queries

- Query
  - `//Host[@Name="lab.nesc.ac.uk"]/ProcessorLoad`
- Result
  - `<ProcessorLoad Last1Min="00" Last5Min="00" Last15min="00"/>`

- An example MDS3 query
  - Returns the processor load element of lab.nesc when applied to a cluster status SDE
    - Works by selecting ALL the host elements
    - Then selects the subset with name "lab.nesc.ac.uk" (could be many)
    - Then from each of these, selects the Processor Load element (could be many again)

Future work on queries

- Other XML query languages?
  - XQuery, XSLT (both using XPath as a base)
  - May allow more interesting XML processing on the server side
    - But don't want too many in case the wrong one is chosen and needs to be scrapped when it gets widely used (sound familiar??!)
- XPath subscription?
  - "when query result changes, send me an update"
  - Could be quite simple to implement…
    - But only if we are interested ONLY in a subset of the SDE data

Future work on queries

- Partial SDE notification?
  - Currently, if you want to subscribe to a large SDE, your only choice is to get the whole SDE back at one time.
  - XPath subscription would help if we are ONLY interested in a subset of the data, not the whole SDE
  - Partial notification could generate XML that describes the difference between the old and new values of the SDE
    - We could generate difference XMLs so some notification streams could be missed, but we still get reasonable results
      - Sounds difficult? It's VERY difficult for arbitrary XML data

Host Status

- Also want to retrieve information about the status of the host (not just the Grid Service)
  - Filesystem info, memory usage, CPU load, network adapter info, etc…
  - This is the info that comes out of MDS2 GRIS by default
  - XML schema based on GLUE
    - Better for representing clusters
  - Single host MDS2 providers ported to XML output
    - Other people working on implementation with Ganglia

Host status example

```
<Cluster Name="pygar.isi.edu" UniqueID="pygar.isi.edu">
  <SubCluster Name="pygar.isi.edu" UniqueID="pygar.isi.edu">
    <Host Name="pygar.isi.edu" UniqueID="pygar.isi.edu">

<Processor
  Vendor=" GenuineIntel"  Model=" Intel(R) XEON(TM) CPU 2"  Version="15.2.4"
  ClockSpeed="2193"  CacheL2="512"/>

<MainMemory VirtualSize="2047" RAMSize="1004" RAMAvailable="119"
   VirtualAvailable="1716" />

<OperatingSystem Name="Linux" Release="2.4.7-10"  Version="#1 Thu Sep 6
   17:27:27 EDT 2001" />

<FileSystem  Name="/"   Size="23510"  AvailableSpace="650"  Root="/"
   Type="unavailable"  ReadOnly="false" />

<NetworkAdapter  Name="eth0"  IPAddress="128.9.72.46"  InboundIP="True"
   OutboundIP="True"  MTU="1500"/>

<ProcessorLoad  Last1Min="00"  Last5Min="00"  Last15Min="00" />

    </Host>
  </SubCluster>
</Cluster>
```

Information Service Development

- Useful hosting environment components
  - OGSI Core
    - Provides basic findServiceData call and subscription handling
  - Can store service data either in memory or in Xindice
    - Xindice is XML database that gives PERSISTENT service data (i.e. you can reboot services and the data will still be there)
    - Switchable per service at runtime (decide what works best for your application)
  - Aggregator mechanism
    - Acts as notification sink, and receives notifications from other services. Republishes under one name so query against this rather than a load of separate ones
  - Provider component
    - Lets you plug in Java or UNIX script service data providers

The GT3 Index Service

- Take all these components, stick them together and configure
  - GT3 IndexService
  - End up like MDS2 GIIS
    - Can retrieve data from various sources
    - Publishes it through ONE service
  - Query the index in the same way as any other service data (e.g. resources)
    - i.e. using findServiceData or subscription
    - Like MDS2 where query a GIIS the same way as a GRIS
    - Arbitrary XML data – no schema

Summary

- The GT3 Index Service is an implementation which provides
  - An interface for connecting service instances to external Service Data Providers
    - Can generate dynamic service data using external programs (either using GT3 or a custom provider)
  - A generic framework for aggregation of service data
    - Can index services from different service providers and support their subscription, notification, polling…
  - A registry of Grid Services
    - Soft-state registration, can support queries on that service