Astronomical Data Analysis Software and Systems XII
ASP Conference Series, Vol. 295, 2003
H. E. Payne, R. I. Jedrzejewski, and R. N. Hook, eds.

National Virtual Observatory Efforts at SAO

Mark Cresitello-Dittmar, Janet DePonte Evans, Ian Evans, Michael Harris, Stephen Lowe, Jonathan McDowell

Harvard-Smithsonian Center for Astrophysics

Michael Noble

Massachusetts Institute of Technology

Abstract. The National Virtual Observatory (NVO) project is an effort to federate astronomical resources, to provide seamless access to heterogeneous data at various centers throughout the world, and make them appear to the user as a homogeneous set. The NVO will reduce the user's need to obtain, recall and manage details such as passwords, band coverage, instrument specificity and access methodologies for each archive site in order to get and analyze data. The project will employ Grid technology and distributed computing techniques to manage enormous data volumes and processing needs.

At the Harvard-Smithsonian Center for Astrophysics (CfA), we are developing a small scale prototype implementation of the NVO paradigm. This demonstration will illustrate the directions being pursued toward this goal by allowing a user to request data from various resources, display the returned data, and interactively perform analysis on that data.

1. Portal

At the Harvard-Smithsonian Center for Astrophysics, we are defining the datamodel protocol for the NVO project. As a test case, we have implemented an end-to-end prototype of an NVO Portal. This portal will aid in defining/refining the datamodel by applying the concepts to a real-world application.

This NVO portal allows the user to query and retrieve data from a variety of archive centers for a particular object or location in the sky. The returned data may then be displayed as a multi-color overlay image on which the user selects a region of interest and executes an analysis task to determine the flux from that region in various wavebands.

The GUI itself is written in S-lang using GTK for the graphics. It manages the communication between the user and the various modules used to perform this task. It knows how to find and communicate with a service registry, but does not have prior knowledge of which services are available.

Figure 1. NVO Portal GUI

2. Resource Discovery

This portal makes use of a Java tuplespace as the service registry. The services and the client communicate to each other through this space. (For more information about tuple space, see Noble, this volume3.) A service must advertise its availability by leasing a "tuple" in the tuplespace. These tuples expire after a certain amount of time unless the service renews this lease to keep them active. When the user clicks the 'Find Archives' button of the portal, a query is sent to the tuplespace asking which archive services are available for searching. The server responds with the list of currently active services (Figure 2). If a service provider has hardware issues or simply needs their resources, they can stop renewing the leased tuple and the service will no longer be visible to the outside world.

3. Query

The user may either input an object name and use the namespace resolver service or enter the coordinates manually (in decimal degrees). If an object name is entered, the portal queries the tuplespace server to determine if a name resolver service is available. If so, the portal constructs a query request for the name resolver service and sends that request to the tuplespace. The name resolver service notices that the request has been registered and processes it, returning the coordinates of the object (Figure 3).

When the user clicks 'Find Images,' the portal forms queries for each of the archive services that were selected by the user. These requests are in a standard form and are submitted individually to the tuplespace server. Each archive service retrieves its tuple, performs the search, and returns the tuple with the results. The tuple server conveys the results back to the user (Figure 4).

The query services return Metadata describing the data and the means to retrieve the file (e.g., a URL or ftp link). The interface collects this information and indicates to the user which images were found. The user then selects which images to retrieve.

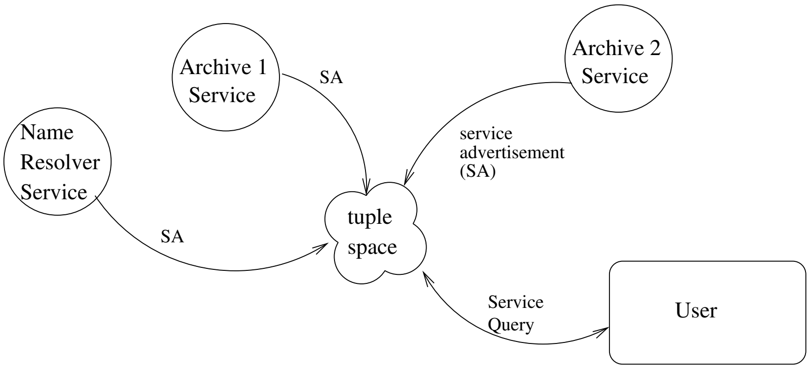


Figure 2. Services advertise to tuple space, client queries for available services.

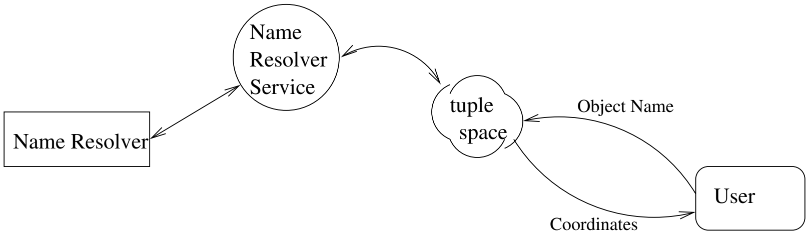


Figure 3. Communication flow for name resolver service.

4. Data Retrieval

Peer-to-Peer Data Transfer Each archive search delivers an information packet to the portal that is sufficient for the portal to retrieve the data file directly from its hosting site. Since this transfer involves large amounts of data, the retrieval process is done peer-to-peer between the portal and the data-hosting site, bypassing the tuplespace registry.

Site-Specific Extractors (SSEs) The data stored at the hosting sites exist in a variety of formats. For interoperability purposes, data used in VO applications should be delivered in a uniform format. In the VO paradigm, each archive service will create a Site-Specific Extractor (SSE), which is a processing pipeline to transform the requested data from its native format to the VO standard format. Since these VO format standards are still being developed and for reasons of convenience, for the present demo, we have written stand-ins for these components which run on the local portal.

5. Data Analysis

While not specifically “VO,” this segment highlights the interoperability advantage that VO formatted data can provide. We created some simple tools to operate on VO formatted data. These tools are local to the user’s computer.

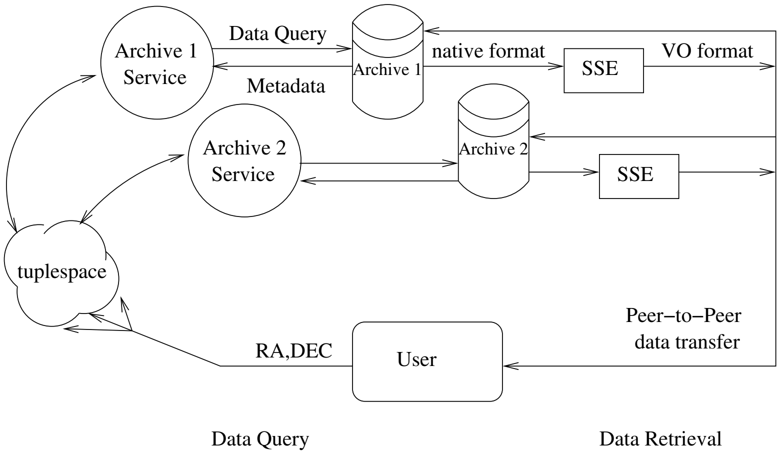


Figure 4. Archive query

Generate Composite Image: An IDL tool to generate a tri-color image of the returned data sorted by energy band. The Lowest=red, Mid=green, Highest=blue. The resulting image is displayed in a DS9 window where the user can select a region on which to perform further analysis.

Display Individual Images: Simple invocation of DS9 with multiple frames so the user can view the individual images.

Flux Calculation: Executes routines that calculate the flux within the specified region for all retrieved data and displays a plot of the results.

6. Future Work

Since this demo provides an end-to-end thread, it will be very helpful in developing and refining the VO Datamodel, which is the focus of our work.

This software was developed with a very modular design. As VO standards for VO file format, query model, data model, etc. evolve, we will replace existing modules with ones conforming to those standards.

Several enhancements to the interface will be needed as we pull more and varied services into the thread.

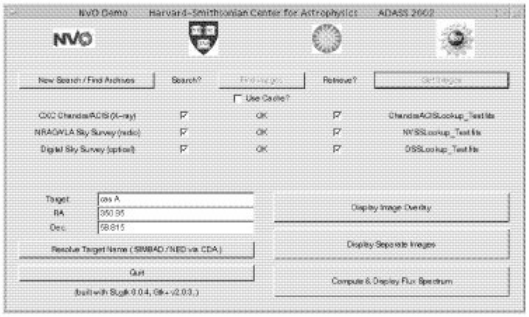
For more information about the NVO project, please refer to these web sites: CfA NVO home,[1] US National VO home,[2] and International VO home.[3]

Acknowledgments. This material is based upon work supported by the National Science Foundation under Cooperative Agreement No. AST-0122449. This project is supported by the Chandra X-ray Center under NASA contract NAS8-39073

[1] http://cfa-www.harvard.edu/nvo

[2] http://www.us-vo.org/

[3] http://www.ivoa.net/


NVO Demo Harvard-Smithsonian Center for Astrophysics ADASS 2002
NVO
New Search / Find Archives
Search?
Find Images
Retrieve?
Get Images
Display Image Overlay
Display Separate Images
Resolve Target Name (SIMBAD / NED via CDA)
Quit
Compute & Display Flux Spectrum

□ Use Cache?

	Search?		Retrieve?	
CXC Chandra/ACIS (X-ray)	☑	OK	☑	ChandraACISLookup_Test.file
NRAO/VLA Sky Survey (radio)	☑	OK	☑	NVSSLookup_Test.file
Digital Sky Survey (optical)	☑	OK	☑	DSSLookup_Test.file

Target: cas A
RA: 350.95
Dec: 58.815

(built with SLgtk 0.0.4, Gtk+ v2.0.3.)

Figure 1. NVO Portal GUI

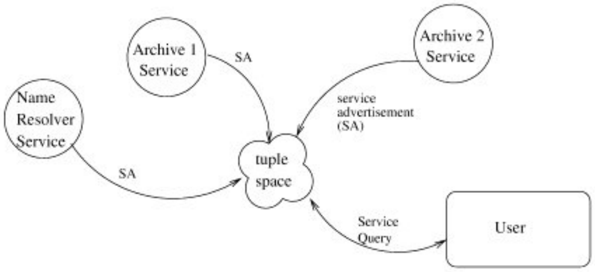


Figure 2. Services advertise to tuple space, client queries for available services.

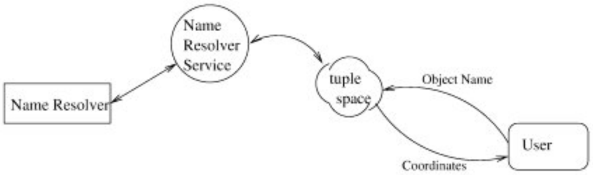


Figure 3. Communication flow for name resolver service.

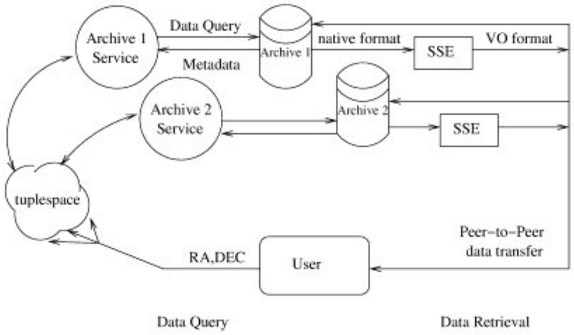


Figure 4. Archive query