OPENDAP: ACCESSING DATA IN A DISTRIBUTED, HETEROGENEOUS ENVIRONMENT

P. Cornillon[*], *J. Gallagher*[*], *T. Sgouros*[†]

[*]*Dept. of Physical Oceanography, Graduate School of Oceanography, University of Rhode Island, Narragansett, RI, 02882 Email: pcornillon@gso.uri.edu*
[†]*Manual Writing, Providence, RI, 02903-3011, Email: tomss@ids.net*

ABSTRACT

In the process of implementing a protocol for the transport of science data, the Open Source Project for a Network Data Access Protocol (OPeNDAP) group has learned a considerable amount about the internal anatomy of what are commonly considered monolithic concepts. In order to communicate among our group, we have adopted a collection of definitions and observations about data and the metadata that make them useful: differentiating between "semantic" and "syntactic" metadata, as well as "translational" and "use" metadata. We share the definitions and categorizations here in the hope that others will find them as useful as we do.

Keywords Metadata, data transport, syntax, semantics, data archive, data model, data attributes.

1 INTRODUCTION

In the distribution of science data, issues of format incompatibility have for years been the most significant obstacle, preventing scientists from freely sharing their data, and impeding or discouraging correspondence with central data archives. To begin to address these issues for oceanographic data, scientists at URI and MIT undertook a project called the Distributed Oceanographic Data System (DODS) in 1993 (Sgouros, 1996-2002). The objective was to make data sharing seamless, and allow scientists to import data into their data analysis software directly from remote sites, regardless of the format in which the data were stored.

no. This is mechanical. Semantic interoperability is the larger problem

The DODS effort consisted of two fundamental parts: a part that focussed on how data were to be moved over the network within the system, the data access protocol or DAP, and a part that focussed on the data sets served by the system, specialized applications for the use of these data, a directory of these data sets, etc. In that the system was to be used for all classes of oceanographic data and because these data span such a large range of data types, formats and organizational structures, the DAP was by necessity designed to operate at a very low level. The result was that although developed for the oceanographic community, there was nothing in the DAP (based on C^{++} data types) that was oceanography-specific; i.e., the DAP could be used to move information about art objects as readily as information about the ocean. The two basic parts of DODS therefore consisted of a discipline-neutral portion, the DAP, and a discipline-specific portion, those parts that were oceanography-specific—data set population, specialized clients, etc.

currently a liability to the design

The neutrality with regard to scientific discipline of the DAP was quickly recognized by scientists in other fields who began to adopt the DAP in their data system development efforts. However, there was a concern among these same groups that with the primary focus of the DODS effort being on oceanographic data system needs, evolution of the core software might diverge from their needs. As a result, specializations of the DAP began to appear that addressed problems faced by all but that took different approaches The result was an undesirable duplication of effort. In order to address this, the Open Source Project for a Network Data Access Protocol (OPeNDAP) was established in 2000 for the development and promotion of software that facilitates access to data via the network. OPeNDAP is a non-profit corporation operating in Rhode Island. At the same time that OPeNDAP was created, the oceanography-specific part of DODS was named National Virtual Ocean Data System (NVODS). NVODS uses the OPeNDAP data access protocol. OPeNDAP has a broader mandate than DODS in that it focuses on network access to data in general. Particular attention within OPeNDAP is on the development of a robust transport protocol that addresses a wide variety of data needs. The present account concerns the OPeNDAP data access protocol. ?

Although the objective of seamless access to data over the network has to a large extent been met by the DAP, some unexpected obstacles have turned up while other anticipated roadblocks failed to materialize. In the process of implementing real solutions to these problems we have learned a considerable amount about the internal anatomy of what are commonly considered monolithic concepts. In order to communicate among our group, we have adopted a collection of definitions and observations about data and the metadata that make them useful. We share these here in the hope that others will find them useful.

2 LAYERS OF INTEROPERABILITY

The essential problem addressed by OPeNDAP is to have a computer program operate on some data provided by some other institution, stored in some unknown format on some remote computer. One important outcome of the work on OPeNDAP is the observation that this is not a binary problem. That is, there are different *degrees* of success in getting intelligible data from a server to a client.

Part this realization came from a subtle restating of the problem. It is often the case that OPeNDAP users want to compare data from different sources. In that case, the user generally wants to see comparable data presented in identical fashion. This means that it is not enough simply to be able to receive data from remote files. It means that data from those remote files should appear uniform, to some degree. The more uniform data from different sources appear, the easier it is to compare them, and the more likely it is that some data processing can be automated. This is what is meant by rendering data "interoperable." We sometimes use "machine-to-machine interoperability" to make even more explicit the degree of uniformity required to be considered interoperable.

As there can be said to be differing degrees of uniformity, there are also differing degrees of being able to operate on remote data. As a matter of convenience in discussing these degrees, we have defined several "layers of interoperability," shown in Figure 1. These layers are conceptually similar to the layers of abstraction defined in internet design documents, and may be thought of as a somewhat more elaborate subdivision of the "presentation" and "application" layers defined in the OSI (ISO) model of the internet.[1] We have found the standard division of the layers do not reflect the situations we have encountered. Our model presented here should not be viewed as the last word for all situations, and there is substantial overlap between the standard model and ours. But we think of our model as a better description of a situation where the sources of data, the types of data, and the kinds of clients are somewhat more heterogeneous than is usually considered normal in other discussions of the internet.

An item of scientific data is accorded some meaning solely by its relationship to other items. A number in some time series is meaningful only to the extent it can be accurately placed within that series. Another number in some three-dimensional grid of measurements only has meaning in the relation it bears to its

[1] Internet design documents using this model define six distinct layers of abstraction above the basic level of oscillating voltages connecting two computers. Roughly, they are: the physical layer (the kind of cable connecting neighboring machines), the data link layer (the kind of communication between one machine and its neighbor), the network layer (how the computer network is laid out), the transport layer (how do data get from one computer to another), the session layer (how does a program on one computer cooperate with a program running on another), the presentation layer (what kind of data are being received), and the application layer (how should the data be displayed to the user). See IEC, 1994; Stevens, 1999.

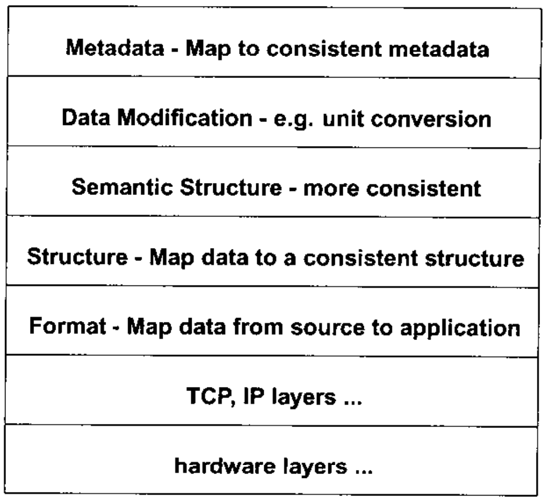
Metadata - Map to consistent metadata
Data Modification - e.g. unit conversion
Semantic Structure - more consistent
Structure - Map data to a consistent structure
Format - Map data from source to application
TCP, IP layers ...
hardware layers ...

Figure 1. Layers of Interoperability

six neighbors, and its position in that grid. These are somewhat more intricate forms of context than is usually relevant in discussions of data transmission. For example, a word transmitted in a piece of text also bears meaning in relation to its context, but the relevant context—at least insofar as low-level issues of transmission format are concerned—is a fairly simple nearest-neighbor sort of context. That is, so long as the words and formatting directives in some HTML file are transmitted in the correct order, they have a fair chance of being understood at the other end.

Transmitting graphic images bears some relation to the problems of transmitting science data. For example, a pixel of some picture, especially one encoded in a progressive format like a progressive GIF, bears a similarly complex relation to its neighboring pixels, which may be encoded in distant parts of the file. But these formats avoid the problems of defining context by subscribing to a much more strictly defined Application Program Interface (API) than is possible for a file format meant to be useful for more than one purpose.

The norm in the analysis of scientific data is that data are stored in a wide variety of different file formats, read and written each by their own specialized API. Some of these are widely used standards, while others are somewhat more *ad hoc*. Few are compatible with each other. The netCDF and HDF formats are two standards popular in the world of earth sciences.[2] The range of data types as well as relationships between these data types and permissible operations associated with them defines the "data model" of a given API. The goal of the OPeNDAP project is to allow users to request data from remote sources, and to allow them to imagine the remote data to be stored in whatever format is most convenient for them. They should be able to download the data directly into the data analysis programs they are using. OPeNDAP provides both server software, to make data available to remote users, and client software, to access those data. Some of the client software can be used to convert programs written for a data access API like netCDF into network-ready "browsers" of remote OPeNDAP data.

what's wrong with HDF?

The first step in achieving this goal was to address the "format" layer, the lowest layer of the diagram in Figure 1. This refers to the ability of a program designed for some API simply to read data from some remote file, no matter what that remote file's format is. To do this, the OPeNDAP server software re-formats data into a well-defined transmission format (see Section 3), which is received by the client

[2]See Rew *et al.*, 1993 or Rew & Davis, 1990 for information about netCDF. See NCSA, 2001 about HDF.

software and re-formatted again to be digestible by that client software.

This works, but the difficulty is that if an API is powerful enough to be useful to a wide variety of researchers, it is also powerful enough to be used in many different ways to solve the same problem. A trivial example here would be two different records of sea surface temperature, recorded on one-degree grid squares, with one using latitude and the other longitude as the X dimension. If a user of OPeNDAP client software was interested in comparing data from these two sources, the fact that they were both transmitted perfectly from their respective servers would be somewhat beside the point. That is, the arrays are inherently incompatible with one another, even though they share (or may be made to appear to share) the same file format.

For a more common example, consider a time series of two-dimensional satellite measurements. One scientist might reasonably choose to keep such a time series as a single netCDF file containing a three-dimensional grid of data, while another group might record the data as a collection of individual files, with one two-dimensional record in each file. Again, the fact that the two data sets are (or can be made to appear) stored in compatible formats is of little relevance to the problem of trying to compare data from the two sources.

With adequate metadata, both situations can be resolved mechanically, and in fact, the OPeNDAP project now supports an "Aggregation Server" to address situations like the second example. But before further discussion of the interaction of different kinds of metadata, it will be useful to present a cursory overview of the OPeNDAP data transmission protocol.

3 THE DATA ACCESS PROTOCOL

DAP is an ISO standard For Directory Access Protocol. why create confusion

The OPeNDAP Data Access Protocol (DAP) defines how an OPeNDAP client and server communicate with one another. The DAP consists of ~~four~~ components:

1. An intermediate data representation. This is used to transport data from the remote source to the client. The data types that make up this representation may be thought of as the OPeNDAP data model, though the array of available types is designed to accommodate many other data models.
2. A format for the ancillary data needed to translate a data set into the intermediate representation, and to translate the intermediate representation into the target data model. This format provides a description of the shape and size of the various data types stored in some given data set and is called the Data Descriptor Structure (DDS).
3. A procedure for retrieving data and ancillary data from remote platforms.
4. An API consisting of OPeNDAP classes and data access calls designed to implement the protocol,

The last two components are not important to the discussion in the present article, and will not be treated further here. The intermediate data representation itself consists of several components:

- A set of simple data types. These include bytes, integers, strings, floating point numbers (all to several different precisions), as well as a specialization of a "string" meant to hold an internet Universal Resource Locator (URL).
- A set of "constructor" data types, constructed from other data types (including other constructor data types). These include simple collections (structures), ordered collections (arrays), relational tables (called "Sequences" here), and "Grids", a structure containing a multidimensional array and a set of one-dimensional arrays to describe its dimensions.

The intermediate data representation also consists of a set of operators for selecting different segments of the data in some data set, and a definition of the external data format for transmitting simple data types (OPeNDAP uses Sun's XDR definitions, see Sun Microsystems, Inc., 1987), but these are also not relevant to the discussion here.

XDR has significant limitations

3.1 The Data Descriptor Structure (DDS) and the Data Attribute Structure (DAS)

The OPeNDAP system uses two different kinds of messages to describe the data it sends. One, the DDS, mentioned above, describes the data structures to be filled by the accompanying data. The other, the DAS contains whatever other descriptions are necessary for a client to make sense of the data. An example DDS and DAS are shown in Figure 2.[3]

why not in explain ASN.1

```
Dataset {
  Int32 catalog_number;
  Sequence {
    String experimenter;
    Int32 time;
    Structure {
      Float64 longitude;
      Float64 latitude;
    } location;
    Sequence {
      Float64 depth;
      Float64 salinity;
      Float64 temperature;
    } cast;
  } station;
} data;
```

```
Attributes {
  catalog_number {
    String type "UN-format";}
  station {
    String shipName "Lollipop";
    Int32 stationID 4533;
    time {
      String format "Seconds since 1/1/1970";}
    location {
      latitude
        Float32 actual_range 0, 30.0;}
      longitude {
        Float32 actual_range -50.0, -48.0;}
    cast {
      salinity {
        String units "ppt";}
      temperature {
        String units "deg Celsius";}}}}}
```

Figure 2. An example OPeNDAP DDS (left) and DAS (right).

The data set described by the DDS in Figure 2 are stored as a catalog number and a Sequence (a relational table with no set length), with each record containing the name of the experimenter, a 32-bit integer containing the time, a Structure containing latitude and longitude values, and another Sequence containing measurements of depth, salinity, and temperature.

The DAS in Figure 2 shows additional information that is helpful to make sense of the data that arrives with the DDS. The DDS indicates that time is sent as an integer, but it is the DAS that explains how the time is encoded in that integer. The DDS explains that salinity values are encoded as 64-bit floating-point values enclosed in a Sequence that also includes depth and temperature values. The DAS defines the units of those salinity values.

Both the DDS and the DAS contain metadata: information about the data the, together with that data, make up the message.

We refer to the metadata held in the DDS as "syntactic" metadata, information that describe the syntax of the data set, and to the metadata held in the DAS as "semantic" metadata, information about the semantics, or meaning, of the data values.

[3]The next version of the DAP will replace the DDS and DAS formats shown here with an XML rendering of the same information. The general principles outlined in this article will not change, nor will the terminology we use to describe them.

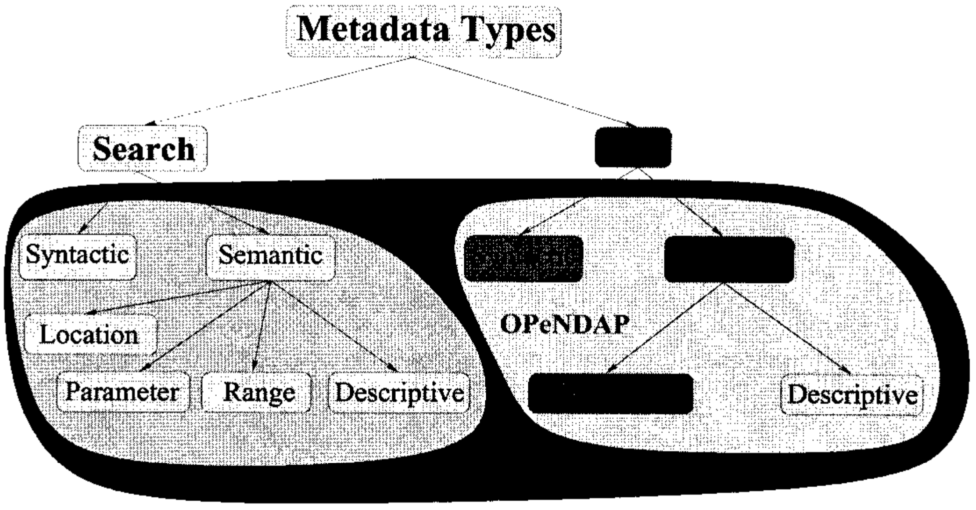

Figure 3. A schematic diagram of different metadata types.

need to discuss. ontologies and how semantic mapping will be handled

contains the ranges of variables within the data set. In most existing directory systems, only the ranges for time and space are included, but in future systems the range of all data values may be included as search metadata. Location search metadata is effectively a pointer to the data, a URL in the OPeNDAP case. Descriptive search metadata contains other information associated with the data set such as a generic description of the sensor used.

6 THE LAYER MODEL, AGAIN

Referring again to Figure 1, the metadata taxonomy introduced above makes explication of the layers simpler. The first structural layer refers to matching syntactic use metadata. Consider the two sea surface temperature records mentioned earlier: one is stored in a three-dimensional grid, with the dimensions of latitude, longitude and time, and the other is stored in many individual files, each containing a two-dimensional grid. These two data sets may be compatible on the format level, but not at the structural level – one consists of one 3-d object and the other of a number of 2-d objects. At the syntactic structural level, these data are all delivered as three dimensional objects. But even the fact that the two data sets have the same general structure does not mean that the structures are necessarily organized identically. For example, one data set could be organized as a longitude, latitude, time matrix while the other might be a time, longitude, latitude matrix. These data objects are syntactically similar, but not semantically similar. We therefore refer to the first structural level as the syntactic structural level. The OPeNDAP project supports a server capable of making the two data sources mentioned above appear to have a similar structure. (The is the Aggregation Server. See Caron & Sgouros, 2002.)

The semantic structural level addresses the semantic incompatibility of structures. In the example cited above, the order of dimensions within the arrays would be altered to one that is consistent for all similar data sets within the system.

It is important to keep in mind that in general each level in the system involves more potential modification of the data. At the format level, only the format is modified (as long as the data type required by the client