DAP2 and DAP4 Protocol Services In the Thredds Server

Unidata NetCDF Workshop

October 25, 2018

Dennis Heimbigner
Unidata

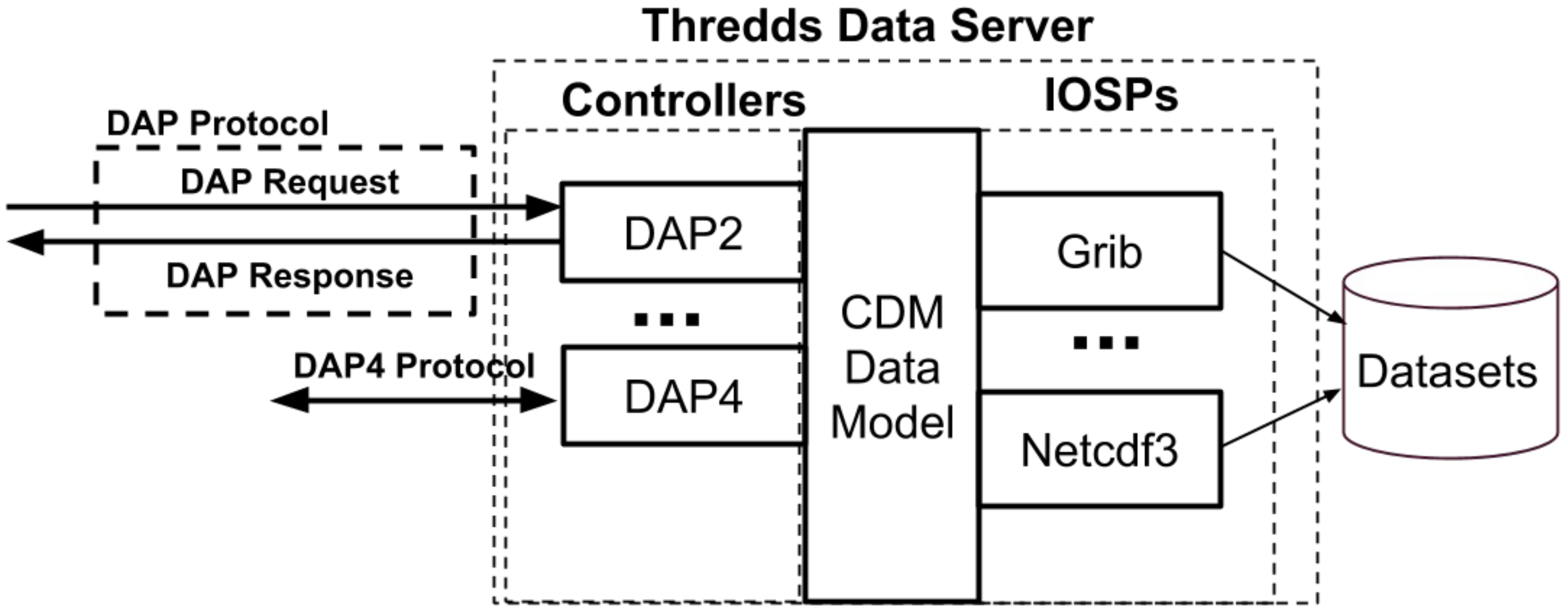
Resources For This Session

- Web Browser
- Test Servers used to test DAP2 and DAP4 Protocols
    - DAP2 Test Server: http://149.165.169.123:8080/dts/
    - DAP4 Test Server: http://149.165.169.123:8080/d4ts/
    - Thredds Server: https://thredds-test.unidata.ucar.edu/thredds/

What are DAP2 (OPeNDAP) and DAP4?

- DAP version 2 (aka DAP2, aka OPeNDAP) is a widely supported protocol and standard data format for accessing remote data
- The DAP2 protocol was expressly designed to serve as intermediate format for accessing a wide variety of data sources
- The newer DAP version 4 protocol (DAP4) provides a richer data model and a more powerful constraint (subsetting) language than DAP2.
- The DAP2 and DAP4 specifications can be obtained from the OPenDAP website.
- DAP Version 2: http://opendap.org/pdf/ESE-RFC-004v1.2.pdf
- DAP Version 4: http://docs.opendap.org/index.php/OPULS_Development#DAP4_Specification

DAP In the Thredds Architecture



Specifying a DAP 2/4 Request

- A DAP2 request is a URL to be sent to the server (via e.g. ncdump)

https://thredds-dev.unidata.ucar.edu/thredds/dodsC/casestudies/harvey/goes16/CONUS/Channel01/20170821/GOES16_CONUS_20170821_020218_0.47_1km_33.3N_91.4W.nc4.dds

- DAP4 equivalent
  - Note the use of **/dap4/** instead of **/dodsC/** in the URL path to distinguish a DAP2 request from a DAP4 request
  - Note the use of **.dap** versus **.dods** to specify the format of the response

https://thredds-dev.unidata.ucar.edu/thredds/dap4/casestudies/harvey/goes16/CONUS/Channel01/20170821/GOES16_CONUS_20170821_020218_0.47_1km_33.3N_91.4W.nc4.dmr.xml

Formats for a DAP2 Request/Response

- For DAP2, there are three core kinds of responses:
  1. **.dds** – Return just the meta-data for the requested dataset.
  2. **.das** – Return just the attributes of the requested dataset; additional attributes may be added that are not in the original dataset.
  3. **.dods** – Return the metadata followed by the actual contents of the dataset encoded in DAP2 format (basically XDR encoding)

- Additional possible responses:
  1. **.asc** – Return the .dods information in ascii format**.**
  2. **.html** – Provide a form for accessing subsets of a dataset.

Formats for a DAP4 Request/Response

- DAP4 responses are simpler (sort-of):
  1. **.dmr** – Equivalent to .dds + .das.
  2. **.dap** – Equivalent to .dods

- Additional response:
  1. **.dsr** – (New) Returns the "dataset services" that describes how to access the dataset.
- The DAP4 controller chooses the by examining the final extensions of the request path. The default is **.dmr.xml**. Other formats are possible if the server supports them: **.dmr.json**, for example.

Components of a Request URL

- DAP2 Example https://thredds-test.unidata.ucar.edu:8080/thredds/dodsC/.../WEST-CONUS_4km_3.9_20181002_0000.gini.dods?IR[0][0:4][0:4],x[0:4],y[0:4]
- The URL has four parts
  1. Protocol: https or http
  2. Host+Port: thredds-test.unidata.ucar.edu+8080 (8080 is default)
  3. Path:/thredds/dodsC/.../WEST-CONUS_4km_3.9_20181002_0000.gini.dods
     a. DAP4: change .dods => .dap
  4. Query (aka Constraint): ?IR[0][0:4][0:4],x[0:4],y[0:4]
     a. DAP4: change to ?dap4.ce=/IR[0][0:4][0:4];/x[0:4];/y[0:4]

DAP Processing

1. Thredds gets incoming request as a URL
2. Looks for “/thredds/XXX” in the path of the URL
3. Uses XXX to choose a controller to process the request
    a. dodsC => DAP2 controller
    b. dap4 => DAP4 controller
4. Controller converts the path (minus e.g. .dods) to an actual dataset on the Thredds server
5. Controller opens that dataset as a NetcdfDataset object
6. Controller accesses the dataset and translatethe CDM representation and data to the “equivalent” DAP format (e.g. .dds, .dmr, etc)
    a. Especially taking the query/contraint into consideration
7. Controller serializes the translation and returns it to the requestor

Introduction to DAP2 Subsetting (aka Constraints)

- DAP2 provides a ***constraint*** notation for requesting a subset of a dataset.
  - Essential for performance by avoiding downloading whole dataset
- The constraint is contained in the query part of a URL
  - The part starting with ‘?’
  - Format is not a standard URL query
- Basically, provide a list of variables slices to specify a subset of each variable
- Example:
  https://thredds-test.unidata.ucar.edu/thredds/dodsC/satellite/3.9/WEST-CONUS_4km/20181020/WEST-CONUS_4km_3.9_20181020_0000.gini.dds?IR[0][0:4][0:4],x[0:4],y[0:4]

Introduction to DAP2 Subsetting (cont.)

- Consider ?IR[0][0:4][0:4],x[0:4],y[0:1:4]

- The forms of a ***slice*** constraint are:
  - [start-index:stride:last-index] (most general)
  - [start-index:last-index] (stride == 1)
  - [start-index] (last-index == start-index)

- DAP2 constraints are considerably more complex than just slices
  - E.g. DAP2 also has a mechanism for accessing parts of ***Sequences***
- Refer to tutorials at opendap.org

Introduction to DAP4 Subsetting

- DAP4 has a constraint notation that is a superset of the DAP2 notation
  - And even more complex
- The insertion into a URL looks somewhat different.
- Previous Example In DAP4 form

  https://thredds-test.unidata.ucar.edu/thredds/dap4/satellite/3.9/WEST-CONUS_4km/20181020/WEST-CONUS_4km_3.9_20181020_0000.gini.dmr.xml?dap4.ce=/IR[0][0:4][0:4];/x[0:4];/y[0:4]

Introduction to DAP4 Constraints (cont.)

- ?dap4.ce=/IR[0][0:4][0:4];/x[0:4];/y[0:4]

- Note differences:
    - Use of semicolon instead of comma
    - The use of a fully qualified name: e.g. /IR
        - Because DAP4 data model has groups
    - Use of key name “dap4.ce”
        - Because the DAP4 query can specify more than just constraints
        - It conforms more closely to standard URL ?key=value… format

- The details can be found in the DAP4 specification.

Questions?