Workflow-Oriented Cyberinfrastructure for Sensor Data Analytics

Arcot Rajasekar
University of North Carolina
Chapel Hill, NC, USA
rajasekar@unc.edu

John Orcutt
University of California
San Diego, CA, USA
jorcutt@ucsd.edu

Frank Vernon
University of California
San Diego, CA, USA
flvernon@ucsd.edu

ABSTRACT

The iRODS middleware provides federation, virtualization, metadata integration and policy-oriented data management for static files. Real-time data streams (RTDS) from sensors and other sources pose a different challenge compared to static files. They are infinite in length and time line, comprise discrete packets of information which are time-specific, and the concept of byte-oriented i/o is not at all suited for accessing and managing real-time data streams. Moreover, because of the nature of the sources, the volume and velocity of the flow can be very low (temperature data) to very high (HDTV). They are also time-sensitive in two ways. The data must be captured immediately or they will be lost forever; and in many cases, the analysis has to be done immediately as the decision making can be time sensitive (eg. earthquake or tsunami detection). We have developed new features in the iRODS system to capture, store and archive RTDS. Our model captures RTDS into a continuum of discrete files and archives them in a few different standardized formats. It also provides packet-based and time-oriented access for replay of sensor data. By folding in the management of RTDS into iRODS we have extended the four main functionalities - federation, virtualization, metadata integration and policy-oriented automation – for real-time data management.

Keywords

Real-time data streams, iRODS, Datanet Federation Consortium, sensors.

INTRODUCTION

Sensors are electronic devices which detect or measure a physical property and convert them to electrical signals suitable for processing to be consumed by other electronic circuitry. Commonly detectable properties include biological, environmental, chemical, electrical, electromagnetic, mechanical, optical, radioactivity, etc. Sensors have been used for monitoring the environment [1,2,3,4], biological systems [5] and the human body [6]. A sensor grid [7,8,9] is a system that integrates sensor networks with grid computing and data grids. Sensor grids provide seamless access to distributed and heterogeneous sensors in a pervasive manner. It allows for applying large-scale computational power for analyzing sensor data, data fusion across multiple sensors, and developing novel algorithms for pattern recognition, sensor data discovery and decision making, using advanced techniques such as deep learning, machine learning, deep indexing, **`data mining`**, and distributed database processing. With the coming prominence of Internet of Things (IoT), more and more common everyday physical devices, buildings vehicles, and appliances are being embedded with electronic sensors leading to an enormous data volume that can be collected and processed [10,11,12]. Hence, there is a need for integrated data management of sensor data for collecting, storing, analyzing, sharing, discovery, and curating sensor data.

With the advent of IoT, Wearable Computers, Smart Cities and Connected Communities, and with large numbers of Science instruments being deployed, the amount of sensor-generated data is growing at a very fast pace. As of now, most of the sensor data are gathered, and analyzed in near-real-time, in situ or very close to the source and seldom archived for the long term (there are exceptions such as Incorporated Research Institutions for Seismology (IRIS),

iRODS UGM 2017, June 13-15, 2017, Utrecht, Netherlands

Integrated Ocean Observing System (IOOS), NASA, etc.) This does not allow many options for reuse, replay and reanalysis of sensor streams. No sensor data management system is currently available, which can deal with millions of sensors, to store, access, analyze and manage sensor streams as is done with file storage and archives. The main reason is a lack of an ability to transport, store and retrieve sensor stream data at scale. When sensor streams are stored for later analysis, as in seismic data at the IRIS, or oceanographic data at IOOS, they are stored as files and access is to individual files through web links and ftp. Hence the problem is not only with storage, but also in retrieval and transport to users and sensor applications. The storage problem is compounded when dealing with millions of sensors.

Over the past 15 years there has been an explosion of sensor network data from many scientific disciplines including satellites in space observing magnetic fields and solar wind, meteorological networks for real-time forecasting and climate research, geophysical observations of earthquakes and tectonic motion, and physical oceanographic measurements of currents, temperature/salinity, waves and acoustic tomography/thermometry. With the advent of virtually ubiquitous networking, environmental sensor data are being streamed to a variety of locations for immediate application (e.g. tsunami detection). Today, seismic and environmental scientists continue to work with file-based systems including the extraction of data from the field and storage of sensor data. For example, field sensor data are first written to files, and at some later time the data are transferred to larger, community storage by copying the files and metadata over the Internet or actually mailing original physical media. The bulk of today's sensor data are managed through file-based systems, but streaming data analysis is quickly replacing the file-based approach even though the software continues to rely upon the traditional file approach. This fundamental mismatch needs to be addressed to meet the growing reliance on sensor stream data through research and development of a sensor-centric data system that provides end-to-end optimal performance.

The integrated Rule Oriented Data Systems (iRODS) [13,14,15,16] is a second-generation data grid that provides a collaboration environment for large-scale data oriented enterprises. iRODS promotes four concepts: virtualization, federation, automation and discovery (see Side Bar A). iRODS provides a rich client interface that supports a range of user-friendly interfaces for accessing data, ingesting and querying metadata, and for performing discipline-centric analysis and visualization through emerging social networking web applications. iRODS integrates and virtualizes distributed and heterogeneous data resources into a single logical file system (called the collection hierarchy) and provides a modular but uniform application processing interface to integrate new client-side applications as well as server-side data and compute resources.

A. iRODS Concepts

Virtualization allows users to create collections of dispersed data residing in distributed, heterogeneous resources and uniformly access them through single sign-on mechanisms. The data resources, the users and the access mechanism are represented by virtual name spaces that are mapped onto real objects. Virtualization implements technology independence and enables seamless access to data while hiding practical problems with authentication, authorization, arbitration and access to independently managed, heterogeneous resources. Virtualization also extends to compute services through containerization integrated with execution and data management. The iRODS system provides virtualization through mapping name spaces for users, resources, data, collections and micro-services (apps).

Policy-based Automation enables customization and realization of complex resource management services at a fundamental level through computer actionable rules. Data resource managers, project leaders, and individual users can chain basic operations (micro-services) in order to define their own access pipeline, life-cycle management, sharing and disposition. The iRODS system used a rule engine to enforce policies stored in a rule base.

Federation interconnects third party data, compute servers, and resources through the virtualization mechanisms. This enables a robust extensible framework for sharing resources owned and operated by third parties in a seamless manner. Multiple levels of interconnection are based on trust models and protocol brokering. iRODS system deployed tight and loosely coupled models as well as asynchronous federation mechanisms.

Metadata integration is essential for discovery and sharing. Multiple types of metadata can be associated with data collections and need to be integrated through a common query mechanism. Metadata can range from key-value pairs, RDF and relational data to semi-structured and unstructured data. The iRODS system provided a common query mechanism through its logic-based QueryArrow system which provides a virtual query interface for both SQL and NoSQL databases.

The iRODS system has been used in multiple large-scale projects [17,18,19,20] and easily scales to 100s of millions of data objects in Petabyte storage systems and supports high-speed data transport including parallel streaming. In performing distributed data management, iRODS acts as a third-party intermediary providing authentication, authorization and auditing, and other functionalities that may or may not be supported by the underlying data resources. iRODS also provides optimized data movement protocols and rich support for metadata for data files as well as data collections. iRODS provides automation for data management as well as support for user-defined processing pipelines through its built-in distributed rule-engine. Administrators and collection owners can encode pipelines and policies as rules for managing and analyzing their data collections. The rule engine in iRODS provides a way to customize the community policies to meet the demands of each discipline and also encode trust relationships for sharing data across disciplines. Using the policy-based data management, one can encode data preservation and access functionalities such as data accession workflows, archival processes, dissemination processes, and analyses and access provisioning – all functions needed by large-scale digital sharing and curation systems. iRODS also provides a full range of services for long-term data management, including tracking replicas, versions, backup, and restoration. iRODS uses checksums to validate the integrity and supports automatic checking and repair of corrupted copies at user-defined intervals.

We have implemented a scalable sensor grid architecture that can be used to dynamically access packets of data in a stream from multiple sensors, and perform synthesis and analysis across a distributed network. Our system is based on the integrating iRODS with a new type of resource called the Antelope Real Time Data System (ARTS) [21], and providing virtualized access and handling to collections of data streams. The iRODS system brings to sensor processing features and facilities such as single sign-on, third party access control lists (ACLs), location transparency, logical resource naming, and server-side modeling capabilities while reducing the burden on sensor network operators. Rich integrated metadata support also makes it straightforward to discover data streams of interest and maintain data provenance. The workflow support in iRODS readily integrates sensor processing into any analytical pipeline. APIs for selecting, opening, reaping and closing sensor streams are provided, along with other helper functions to associate metadata and convert sensor packets into NetCDF and JSON formats. Near real-time sensor data including seismic sensors, environmental sensors, LIDAR and video streams are available through this interface [22]. We discuss the implementation of these features in some detail in the rest of the paper.

SENSOR DATA REQUIREMENTS

Sensor data have some peculiar properties compared to static data.. Unlike files, the sensors are highly distributed and their geographic location is an important property of their metadata and need to be captured. Another important feature of sensor data is that they are potentially infinite, but produced at discrete time intervals and referenced to a canonical time system. Hence the data stream from a sensor, unlike that from a file, can be unlimited and growing and cannot be defined in terms of bytes. Sensor streams comprise quanta of bytes called ‘packets’ and we can view a sensor data stream as a time-stamped series of packets. Hence, when accessing and storing data from sensors, one needs to deal with packets instead of byte buffers. These packets consist of a header, which can be used as an identifier (there are other metadata also part of the header) and a body (or payload) which contains the data. Moreover, a packet can be a complex, concentrated structure, having multiple sensor measurements either from many sensors collected over a period of time, or measurements taken by several co-located sensors at the same time and packed together in a single packet. Also, in many cases, the datum coming from the sensor can be a representation, example and electrical voltage value, which may be converted to the actual measurement of the physical entity before it can be stored or used in analysis. Hence, dealing with the ingestion and storage of each sensor stream may need to be customized and pre-processed.

Access to sensor data is normally done directly from sensors or at a concentrator (such as the ARTS systems). Currently most sensor processing is done in (almost) real time and seldom done afterward. When anyone wants to do post-facto analysis of archived sensor data, they are provided access to one or more files that contain the sensor data (stored in a standard format) and they have to deal with how to unpack that file and extract the time-series sensor data.

Replay and fast replay of sensor streams is very rare, but such a relay capability will be highly necessary if the interpreter wants to compare real time data with archived data.

If one wants to use data from multiple sensors in their analysis, unless they are available through the same concentrator, it is almost impossible to perform such analysis; this will need access to multiple sensor systems or concentrators (many of them have proprietary access control and authentication) as well as perform application-level time-alignment and data fusion. Hence providing capabilities for easy fusion of data from diverse and distributed sensor streams will be very useful for complex data analysis.

In our Datanet federation Consortium (DFC) [23,24,25] project, multiple usage models have been identified. The type of sensors that we need to access include marine, seismic, hydro and other environmental sensors, engineering sensors (as from smart buildings as well as infrastructures such as bridges), biological sensors, and diagnostic sensors such as MRI. The main needs of this group of scientists include ease of access to sensor data, export to standard formats so that it is easy to manipulate, access for archived sensor data, synchronized playback and integrated metadata for discovery and ease of integration into workflows and access through tools and applications. In order to meet these needs, we have developed an extension to the iRODS system enabling access, store, archive, discover, replay and analyze real-time data streams.

ANTELOPE REAL TIME SYSTEMS

The Antelope Real Time Systems (ARTS) [21] sensor data concentrator is used by multiple projects. ARTS (Figure 1) uses the concept of Object Ring Buffer (ORBs) to implement sensor data acquisition, transport, buffering, processing, archiving and distribution of environmental monitoring information. Antelope provides real time automated data processing and non-real time batch mode and interactive data processing. It has a built-in relational database for holding all raw data as well as processing results and other meta information. Antelope provides a comprehensive list of field interface modules for connecting with field sensor/digitizer/datalogger hardware to acquire data as well as state of health information and to control the field units. Antelope also has extensibility options in which application specific real time processing modules can be integrated easily for extracting information and knowledge from the raw data. Processing results are stored back into the same object oriented ring buffers as the raw data. These applications can be used for triggering new actions based on conditions/events emanating from one or more data streams. We use this facility to extract metadata automatically from a data stream. The Antelope System uses a relation-based database called Datascope which stores metadata in a relational schema as well as storing sensor data. It exposes a relational view of accessing sensor data as rows and allows time-interval querying. The ARTS system is used by multiple projects [26-29], including the SciON project [30], part of the DFC.

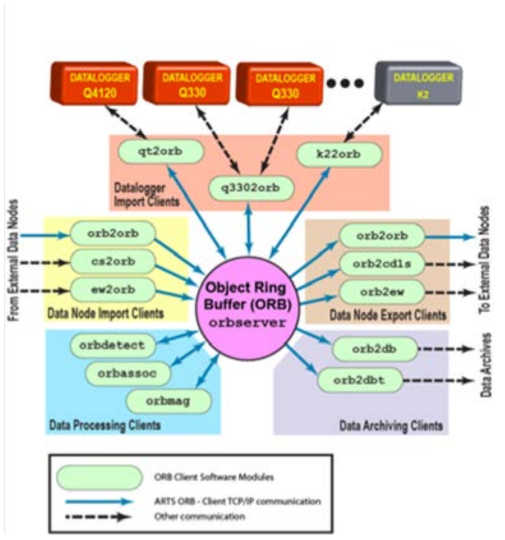

Figure 1 Antelope Real Time System

IRODS SENSOR INTEGRATION

The integration of the sensor data management in iRODS is done through an implementation of series of micro-services. The micro-services interact with the ARTS system and perform operations that provide access to sensor data from ARTS. Other micro-services were implemented to manipulate packet data, perform conversion and store them into the iRODS system. We discuss them below.

ARTS Micro-services

As mentioned earlier, ARTS stores and disseminates sensor packets in an object ring buffer. The ORBs store packets from multiple sensor streams. In order to access sensor packets, we need to connect to the ARTS systems at the host where it is running and perform stream-oriented operations. We have implemented micro-services for connecting to an ORB and disconnecting from it. The stream-oriented micro-services provide means to select a particular stream from the ORB and position the cursor for starting the read operation, using seek and position operations. The selection can also select one or more streams at the same time and can be used with wildcards for accessing data from multiple sensors, in interleaved time-series mode. Micro-services for getting the current packet, next packet in the time-series as well as installing a packet into the ORB are also implemented. Using these micro-services, one can open one or more streams and access packets in a row and then close the stream. Apart from these low-level micro-services for interacting with the ORB, two other micro-services called the *msiAntelopeGet* and the *msiAntelopePut* provide high level access for getting a stream in bulk mode and ingesting a stream back into the ORB systems. We also provide some heartbeat monitoring micro-services for checking the status of the ORB.

As mentioned before, a packet from a sensor stream can be quite complex with header and payload and the payload can also be quite complex. Micro-services are provided to decode and encode packets as well as unpack and repack a packet payload (using stuff and unstuff terminology which comes from ARTS). These micro-services along with another one which can perform format conversions of sensor data measurement form a suite of packet manipulation services that can be used in an iRODS workflow. Side Bar B provides brief explanations for these real-time sensor micro-services.

B. ARTS Micro-services

- Single Packet Microservices
  - msiAntelopeGet - get a packet
  - msiAntelopePut - put a packet
- Connection Microservices
  - msiOrbOpen - open an orb
  - msiOrbClose - close an orb
  - msiOrbTell - redirect to another orb
- Stream-level Microservices
  - msiOrbSelect - select streams
  - msiOrbReject - reject streams
  - msiOrbPosition - position read pointer by packetid
  - msiOrbSeek - position read pointer by skipping *n* packet
  - msiOrbAfter - position by time
  - convertExec - format conversion
- Packet Low-level Access Microservices
  - msiOrbGet - get current packet
  - msiOrbReap - get next packet
  - msiOrbReapTimeout - get next with timeout (return after timeout)
  - msiOrbPut - push a packet into stream
- Packet Manipulation Microservices
  - msiOrbUnstuffPkt - unpack a packet
  - msiOrbDecodePkt - decode a packet
  - msiOrbStuffPkt - pack a new packet
  - msiOrbENcodePkt - encode a packet
- Heartbeat Microservices
  - msiOrbStat – get info on streams
  - msiOrbPing – check on an ORB

Storage Formats

The sensor data stream is stored in the iRODS system as files. We provide three different formats for storing the time-series data. The first format is the compact format where the data packets are stored "as is". This format conforms to the one exported by the ARTS ORB system and can be easily ingested back into the ORB if needed for playback. The space needed for this is also quite low compared to other formats. A second format of storage is in the netCDF/HDF5 format called Common Data Language (CDL) format. This is human readable and is self-describing. We also provide storage in a third format using JSON. This format is useful for web-based apps. Figures 2 and 3 show some sample sensor data streams in the CDL and JSON formats. The schema for these two formats are defined such that the data can store a vector or a singleton measurement. Moreover, they are extensible as new data can be easily appended to the tail of these files.

```
netcdf barometric_pressure {
types:
 compound pressure_vector_t {
   double  timestamp;
   float pressure ;
   float infrasound ;
  }; // barometric_vector_t
dimensions:
      time = UNLIMITED;
variables:
            pressure_vector_t barometric(time) ;
             barometric:standard_name = "two vector barometric pressure
data" ;
             barometric:long_name = "Barometric" ;
// global attributes:
      :srcname = "TA_O03E/MGENC/EP1";
      :packettype = "waveform";
      :net = "TA";
      :sta = "O03E";
      :chan = "LDO";
      :loc = "EP";
      :sampratepersec = " 1.000";
      :calib = "          1";
      :calper = "-1.000";
      :segtype = "5s";
      :nsamps = "120";
      :epochtime = "1446064294.9710000";
      :epochstarttime = "Wed 2015-301 Oct 28 20:31:34.97100";
      :epochendtime = "20:33:34.97100";
data:
barometric =
      {1446064294.9710000,  717022,  10159},
      {1446064295.9710000,  717021,   8821},
      {1446064296.9710000,  717023,  15918},
      {1446064297.9710000,  717026,  21402},
```

Figure 2 CDL Format: Pressure Data

```
{
      "packets":[
      {
            "srcname":"TA_J01E/MGENC/SM100",
            "pkttime":" 6/25/2015 (176)  0:30:23.968",
            "bytes":"535",
            "packettype":"waveform",
            "channels":[
            {
                  "channum":" 0",
                  "net":"TA",
                  "sta":"J01E",
                  "chan":"HNZ",
                  "loc":"",
                  "sampratepersec":"100.000",
                  "calib":"          1",
                  "calper":"-1.000",
                  "segtype":"5s",
                  "nsamps":"100",
                  "epochtime":"1435192223.9683931",
                  "epochstarttime":"Thu 2015-176 Jun 25
                                    0:30:23.96839",
                  "epochendtime":" 0:30:24.96839",
                  "data":[
                        {"v":"  -52727"},
                        {"v":"  -52729"},
                        {"v":"  -52729"},
                        {"v":"  -52731"},
```

Figure 3JSON Format: Seismic VData

Demonstrating iRODS Sensor Workflows

The micro-services implemented for real-time data access and manipulation can be used along with other object-oriented micro-services in iRODS to perform many types of applications. Using the iRODS rule language we, can write applications that can be run interactively from the client or on the server side for continuous data reaping operations. We have developed several workflow programs to demonstrate the system capabilities. These include applications for (a) reaping *n* packets and storing them directly in the compact format in iRODS; (b) archiving one or more packets from a data stream in JSON format; (c) same as for CDL format; (d) ingesting packets into a sensor stream; (e) perform an orb2orb copy of a sensor data stream. (f) Access data from iRODS-stored files in CDL format through the iRODS Cloud Browser; (g) Show plots of data streams using the HDFViewer. Appendix A shows a few of the workflows that we have developed to showcase the application of these micro-services.

Demonstrating Real-time Sensor Data Access

One of the requirements for DFC is to show the streaming access of data from the iRODS system for sensor data. By its nature, the iRule command does not do streaming output and cannot deal with continuous data access. To perform this operation, we implemented a new iCommand called *isense* that can continuously reap packets from the iRODS-ARTS integration and show it on the screen.

isense   *orbHost*   *sensorName*

The command takes the ORB hostname, and a sensor name to continuously display the datapacket

```
isense "anfexport.ucsd.edu:cascadia" "TA_J01E/MGENC/SM100"
```

The hostname maps to a particular ORB and the stream identified by the second parameter, in this case, is a seismic stream coming from the Anza Seismic Network. One can easily pipe this output to a stream processing application and perform real-time operations on the sensor stream.

We have used the isense command to reap sensor data and have piped it over the web using the websocketd [31] command-line tool to send a stream of data over to the web. At the web client side, we used the SmoothieChart [32] to continuously plot the sensor data stream on a web browser. This setup is shown in Figure 4.

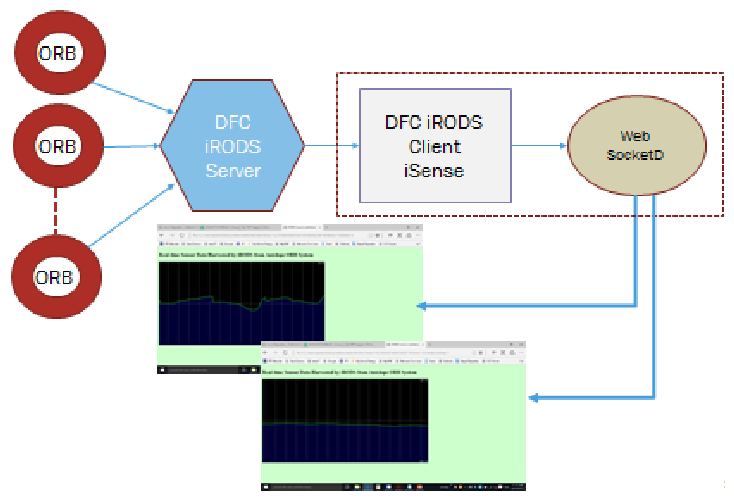

Figure 4 Web-access to Real-time Data Stream through iRODS

CONCLUSION

We have developed new features in the iRODS system to capture, store and archive RTDS. Our model captures RTDS into a continuum of discrete files and archives them in a few different standardized formats. It also provides packet-based and time-oriented access for replay of sensor data. By folding in the management of RTDS into iRODS we have extended the four main functionalities - federation, virtualization, metadata integration and policy-oriented data management – for real-time data. This extension to the iRODS system brings to sensor processing features and facilities such as single sign-on, third party access control lists (ACLs), location transparency, logical resource naming, and server-side modeling capabilities while reducing the burden on sensor network operators. Rich integrated metadata support also makes it straightforward to discover data streams of interest and maintain data provenance. The workflow support in iRODS readily integrates sensor processing into any analytical pipeline. APIs for selecting, opening, reaping and closing sensor streams are provided, along with other helper functions to associate metadata and convert sensor packets into NetCDF/HDF5 and JSON formats. With this extension, iRODS is well on its way to being a urban platform for smart cities and connected communities.

ACKNOWLEDGMENTS

This research is partially supported by NSF grant #0940841 “DataNet Federation Consortium”.

REFERENCES

[1] Akyildiz, A, W. Su, Y. Sankarasubramaniam, E. Cayirci, “Wireless sensor networks: a survey,” Computer Networks, Volume 38, Issue 4, 15 March 2002, pp. 393–422.

[2] Ho, C.K., A. Robinson, D. Miller and M. Davis, “Overview of Sensors and Needs for Environmental Monitoring,” Sensors 2005, 5, pp 4-37.

[3] Ituen, I., G. Sohn, “The Environmental Applications of Wireless Sensor Networks,” International Journal of Contents 2007, 3:4, pp 1-7, doi: 10.5392/IJoC.2007.3.4.001.

[4] Oliveira, L., J. Rodrrgues, “Wireless Sensor Networks: a Survey on Environmental Monitoring,” JOURNAL OF COMMUNICATIONS, VOL. 6, NO. 2, APRIL 2011.

[5] Alemdar, H., C. Ersoy, “Wireless sensor networks for healthcare: A survey”, Computer Networks, Volume 54, Issue 15, 28 October 2010, pp. 2688–2710.

[6] E. Egbogah, A. Fapojuwo, “A Survey of System Architecture Requirements for Health Care-Based Wireless Sensor Networks,” Sensors 2011, 11, 4875-4898; doi:10.3390/s110504875.

[7] J. Yick, B. Mukherjee, D. Ghosal, “Wireless sensor network survey,” Computer Networks, Volume 52, Issue 12, 22 August 2008, Pages 2292–2330.

[8] H.B. Lim, et al. Sensor Grid: Integration of Wireless Sensor Networks and the Grid, In Proc. of the IEEE Conference on Local Computer Networks 30th Anniversary (LCN’05), October 2005.

[9] M Richards; M Ghanem; M Osmond; Y Guo; J Hassard (2006), "Grid-based analysis of air pollution data", Ecological modelling, 194 (1-3).

[10] Xia, F., A. Yang, L. Wang, A. Vinel, “Internet of Things,” Editorial, INTERNATIONAL JOURNAL OF COMMUNICATION SYSTEMS Int. J. Commun. Syst. 2012; 25:1101–1102.

[11] Atzori, L., A. Lera, G. Marabito, “The Internet of Things: A survey,” Computer Networks, Volume 54, Issue 15, 28 October 2010, Pages 2787–2805.

[12] Bonomi, F., R. Milito, J. Zhu, S. Addepalli, “Fog computing and its role in the internet of things,” Proceedings of the first edition of the MCC workshop on Mobile cloud computing, 2012, pp. 13-16.

[13] Moore, R., A. Rajasekar, “iRODS: Data Sharing Technology Integrating Communities of Practice”, Proceedings of the 2010 IEEE International Geoscience and Remote Sensing Symposium, July 25, 2010.

[14] Moore, R.W., H. Xu, M. Conway, A. Rajasekar, J. Crabtree, H. Tibbo, "Trustworthy Policies for Distributed Repositories², Synthesis Lectures on Synthesis Lectures On Information Concepts, Retrieval, and Services, Vol 8 Issue 3, pp. 1-133.

[15] Moore, R., A. Rajasekar, H. Xu, “Extensible Generic Data Management Software”, Journal of Open Research Software, July 2014.

[16] Rajasekar, A., M. Wan, R. Moore, W. Schroeder, S.-Y. Chen, L. Gilbert, C.-Y. Hou, R. Marciano, P. Tooby, A. de Torcy, B. Zhu, “iRODS – integrated Rule Oriented Data System”, book in Synthesis Lectures on Information Concepts, Retrieval, and Services, Editor Gary Marchionini, Morgan Claypool Publishers, 2010.

[17] Cyverse, transforming science through data-driven discovery, **http://cyverse.org**.

[18] Hydroshare: Share and Collaborate, https://www.hydroshare.org.

[19] Chiang, G-T., P. Clapham, G. Qi, K. Sale, G. Coates, “Implementing a genomic data management system using iRODS in the Wellcome Trust Sanger Institute,” BMC Bioinformatics, 2011, Volume 12, Number 1, Page 1.

[20] XSEDE, Extreme Science and Engineering Discovery Environment, **https://www.xsede.org/.**

[21] ARTS: Antelope Real Time Systems. **http://www.brtt.com/docs/ARTS.html**.

[22] Lindquist, K.G. and F.L. Vernon, D. Quinlan, J. Orcutt, A. Rajasekar, T.S. Hansen, S. Foley (2007). "The Data Acquisition Core of the ROADNet Real-Time Monitoring System". In Proceedings from Data Sharing and Interoperability on the World-wide Sensor Web (DSI 2007), Cambridge, MA (April 24, 48 pp.).

[23] DFC: Datanet Federation Consortium, **www.datafed.org**

[24] Billah, M., J. Goodall, U. Narayan, B. Essawy, V. Lakshmi, A. Rajasekar, R. Moore, “Using a data grid to automate data preparation pipelines required for regional-scale hydrologic modeling”, Environmental Modeling and Software 78:31-39, March 2016.

[25] Moore, R., A. Rajasekar, “Reproducible Research within the DataNet Federation Consortium”, International Environmental Modeling and Software Society 7th International Congress on Environmental Modeling and Software, San Diego, California, June 2014, http://www.iemss.org/society/index.php/iemss-2014-proceedings.

[26] USArray/Earthscope, **http://www.earthscope.org/**

[27] Anza Seismic Network, **http://eqinfo.ucsd.edu/deployments/anza.html**.

[28] HPWREN: High Performance Wireless Research and Education Network, **https://hpwren.ucsd.edu/**

[29] NEES: National Earthquake Engineering Simulation. **http://www.nees.org**

[30] SciON: Scientific Observatory Network, **http://scion-network.ucsd.edu/**

[31] WebSockets: the UNIX way, **http://websocketd.com/**

[32] Smoothie Charts: A JavaScript Charting Library for Streaming Data, **http://smoothiecharts.org/**

APPENDIX A: SINGLE PACKET REAP

```
reapAndConvertAntelopPacket {
#Get Packet
       msiOrbOpen(*orbHost,*orbParam, *orbId);
       msiOrbSelect(*orbId, *Sensor,*sresOut);
       msiOrbReap(*orbId, *pktId, *srcName, *oTime, *pktOut, *nBytes, *resOut);
       msiOrbDecodePkt(*orbId, *modeIn, *srcName, *oTime, *pktOut, *nBytes,
                     *decodeBufInOut);
       msiOrbClose(*orbId);
#Store Packet
       *SColl = *Coll ++ "/" ++ *Sensor
       *SFile = *SColl ++ "/" ++ "waveform.data";
       msiCollCreate(*SColl,"1",*STAT_1);
       openForAppendOrCreate(*SFile, *Resc, *D_FD);
       msiDataObjWrite(*D_FD, *decodeBufInOut, *WR_LN);
       msiDataObjClose(*D_FD,*STAT_2);
}

openForAppendOrCreate(*SFile, *Resc, *D_FD) {
# Sub Rule for appending if file already exists or creating header otherwise
       *SObj = "objPath=" ++ *SFile ++ "++++openFlags=O_RDWR";
       msiDataObjOpen(*SObj, *D_FD);
       msiDataObjLseek(*D_FD, *Offset,*Loc,*Status1);
}
openForAppendOrCreate(*SFile, *Resc, *D_FD) {
       msiDataObjCreate(*SFile, *Resc, *D_FD);
}
INPUT Coll="/rajaanf/home/rods/newsenstest",
       *Resc="destRescName=anfdemoResc++++forceFlag=", *Sensor= "TA_J01E/MGENC/SM100",
       *orbHost="anfexport.ucsd.edu:cascadia", *orbParam="", *modeIn=2, *Offset="0",
       *Loc="SEEK_END"
OUTPUT *pktId, *srcName, *oTime, *nBytes, *pktOut, *decodeBufInOut, ruleExecOut
```

APPENDIX B: ORB2ORB

```
orb2OrbReapedPacketIngestion {
#get a MGENC packet  from cascadia ORB and put it in demo  ORB
# also write also in a file to compare

# get the packet and the write into file
      msiAntelopeGet(*pktSelectInfo, *firstPktId, *lastPktId, NumOfPkts,*outBufParam);
     *SColl = *Coll ++ "/" ++ *Sensor
     *SFile = *SColl ++ "/" ++ "*firstPktId" ++ "_" ++ "*lastPktId" ++ ".data";
     msiCollCreate(*SColl,"1",*STAT_1);
     msiDataObjCreate(*SFile, *Resc, *D_FD);
     msiDataObjWrite(*D_FD, *outBufParam, *WR_LN);
     msiDataObjClose(*D_FD,*STAT_2);

# write to orb
     msiAntelopePut(*orbName, *srcName, *timeStamp, *outBufParam);
}
INPUT *pktSelectInfo="<ORBHOST>anfexport.ucsd.edu:cascadia</ORBHOST>
       <ORBSELECT>TA_J01E/MGENC/SM1</ORBSELECT><ORBWHICH>ORBOLDEST</ORBWHICH>
       <ORBNUMOFPKTS>1</ORBNUMOFPKTS><ORBNUMBULKREADS>1</ORBNUMBULKREADS>
       <ORBPRESENTATION>ONEPKT</ORBPRESENTATION>",
       *Resc="destRescName=anfdemoResc++++forceFlag=",
       *Coll="/rajaanf/home/rods/SensorData", *Sensor="TA_J01E_MGENC_SM1",
       *orbName="anfdevl.ucsd.edu:demo", *srcName="DFC_UNC/MGENC/T1", *timeStamp=""
OUTPUT *outBufParam, *firstPktId, *lastPktId, *NumOfPkts,  ruleExecOut
```

APPENDIX C: CONTINUOUS REAP

```
continuousReap {
  delay("<PLUSET>30s</PLUSET><EF>10m</EF>") {
       msiAddKeyVal(*KVP,"selectCriteria",*pktSelectInfo);
       msiAntelopeGet(*pktSelectInfo, *firstPktId, *lastPktId,
            *NumOfPkts,*outBufParam);
       *SColl = *Coll ++ "/" ++ *Sensor
       *SFile = *SColl ++ "/" ++ "*firstPktId" ++ "_" ++ "*lastPktId" ++ ".data";
       msiCollCreate(*SColl,"1",*STAT_1);
       msiDataObjCreate(*SFile, *Resc, *D_FD);
       msiDataObjWrite(*D_FD, *outBufParam, *WR_LN);
       msiDataObjClose(*D_FD,*STAT_2);
       msiAddKeyVal(*KVP,"firstPktId","*firstPktId");
       msiAddKeyVal(*KVP,"lastPktId","*lastPktId");
       msiAddKeyVal(*KVP,"numOfPkts","*NumOfPkts");
       msiAssociateKeyValuePairsToObj(*KVP, *SFile, "-d");
    }
    writeLine("stdout", "Delayed Rule Launched");
}
INPUT *pktSelectInfo="<ORBHOST>anfexport.ucsd.edu:cascadia</ORBHOST>
       <ORBSELECT>TA_M04C/MGENC/EP40</ORBSELECT><ORBWHICH>ORBOLDEST</ORBWHICH>
       <ORBNUMOFPKTS>8</ORBNUMOFPKTS><ORBNUMBULKREADS>4</ORBNUMBULKREADS>",
       *Resc="destRescName=anfdemoResc++++forceFlag=",
       *Coll="/rajaanf/home/rods/SensorData",
       *Sensor= "TA/M04C/MGENC/EP40"
OUTPUT ruleExecOut
```