ARL-CR-0780 ● SEP 2015

US Army Research Laboratory

High-Bandwidth Tactical-Network Data Analysis in a High-Performance-Computing (HPC) Environment: Device Status Data

prepared by Brian Panneton
Technical and Project Engineering, LLC
Alexandria, VA

Brendan Tauras, Christopher Wancowicz, and Sean Coyne
US Army Aberdeen Test Center
Aberdeen Proving Ground, MD

under contract W91CRB-11-D-0007

NOTICES

Disclaimers

The findings in this report are not to be construed as an official Department of the Army position unless so designated by other authorized documents.

Citation of manufacturer's or trade names does not constitute an official endorsement or approval of the use thereof.

Destroy this report when it is no longer needed. Do not return it to the originator.

ARL-CR-0780 ● SEP 2015

US Army Research Laboratory

High-Bandwidth Tactical-Network Data Analysis in a High-Performance-Computing (HPC) Environment: Device Status Data

prepared by Brian Panneton
Technical and Project Engineering, LLC
Alexandria, VA

Brendan Tauras, Christopher Wancowicz, and Sean Coyne
US Army Aberdeen Test Center
Aberdeen Proving Ground, MD

under contract W91CRB-11-D-0007

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED (From - To)
September 2015	Final	July 2012–December 2014

4. TITLE AND SUBTITLE
High-Bandwidth Tactical-Network Data Analysis in a High-Performance-Computing (HPC) Environment: Device Status Data

5a. CONTRACT NUMBER
W91CRB-11-D-0007

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)
Brian Panneton, Brendan Tauras, Christopher Wancowicz, and Sean Coyne

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Technical and Project Engineering, LLC
Alexandria, VA

US Army Aberdeen Test Center
Aberdeen, MD

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
US Army Research Laboratory
ATTN: RDRL-CIH-C
Aberdeen Proving Ground, MD 21005

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)
ARL-CR-0780

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT
Well-designed network traffic collection provides a wealth of information regarding what a network-centric system is doing. Analysis tools examine the data and readily help answer "what," but "why" and "how" may not be attainable, even with advanced analytics. Collecting device status information can augment network traffic information to help answer "why" and "how." To answer these questions, the Aberdeen Test Center developed an active device status collector and collaborated with the US Army Research Laboratory to develop a high-performance-computing solution for correlating device status data with network traffic data.

15. SUBJECT TERMS
tactical networks, data reduction, high-performance computing, data analysis, big data

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Kenneth Renard
Unclassified	Unclassified	Unclassified	UU	22	19b. TELEPHONE NUMBER (Include area code) 410-278-4678

Contents

List of Figures

List of Tables

1. Introduction

Device status collection configuration is usually more involved than network traffic collection configuration because it is usually *active* collection. Active means the collector device is a host on the network under test, and it usually requires configuration of the system under test to allow the collector to interact with it over network protocols. The load incurred by directly polling devices must be carefully considered to ensure the active collection does not affect the system under test. While trap-type paradigms[1] would help reduce the load on devices, hardware vendors may not support and typically do not allow trap-type paradigms. Therefore, polling methods are employed for maximum compatibility. The US Army Aberdeen Test Center (ATC) developed a software package called Hydra to provide this capability.

Device status data processing mainly involves grouping related data and applying transformations, such as interpreting or cross-referencing poll results with test plan information and/or expected values. ATC collaborated with the US Army Research Laboratory (ARL) to develop a distributed computing software framework that provides a basis for processing device status data collected by Hydra.

2. Device Status Data

Hydra collects several different sources of device status data. Each different type of data is typically associated with the network protocol employed to collect it.

2.1 SNMP

SNMP (Simple Network Management Protocol) is a set of standards for inspecting and altering devices on networks.[2] SNMP specifies an architecture that includes an application layer protocol for communication, a Structure of Management Information (SMI) for defining objects, and a Management Information Base for containing objects. SNMP is often used to monitor or alter system configuration data on devices, such as routers, switches, and servers. SNMP data are typically collected by Hydra to show the direct status of individual devices that comprise a system.

2.2 NMS

NMS (Network Management System) stores statuses and statistics of devices connected to a network into a relational database. The database can be queried using Structured Query Language (SQL) to find out a status of a particular device or find

out how a device is operating. NMS data are typically collected by Hydra to show a system's perspective of how its devices and other components are functioning.

2.3 ICMP Ping

ICMP (Internet Control Message Protocol) ping is a standard of providing diagnostic and error messages across a network.[3] ICMP ping uses ICMP message types 8 (echo request) and 0 (echo reply) to determine whether a host is reachable on a network. ICMP ping also provides sequence numbers and round-trip times for echo request and reply pairs. ICMP ping data are typically collected by Hydra to determine whether devices that comprise a system are reachable to help diagnose issues if system services are down.

3. Data Collection

Hydra is an ATC-developed Linux software package used to collect SNMP and NMS data (Fig. 1). Hydra is an active collection system in the sense that it polls network devices at a user-specified interval and they respond with an answer.

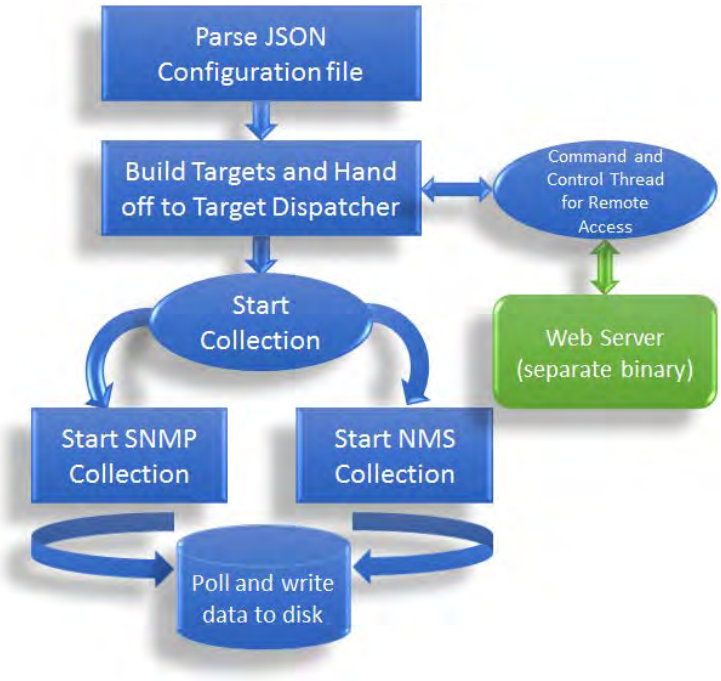

Fig. 1 Hydra program flow

Using the Net-SNMP[1] library Hydra can poll SNMP agents using version 2c or version 3 requests. It can perform SNMP Walk's and Get's with any valid SNMP Object Identifier (OID).

The second part of Hydra collection, MySQL database querying, can be performed on any MySQL database.[4] Hydra can use custom select statements on any database table or select the entire table at a user-specified interval.

Hydra records its data in binary large object (BLOb) files.[5] The data are organized into cuts inside the BLOb files. Typically each cut represents one poll on the network.

4. Hydra Configuration

Hydra uses a JavaScript Object Notation (JSON) file to hold its configuration information.[6] This file holds information on what targets are to be polled. An SNMP target consists of the name of the device to be polled (this will be used in the name of the BLOb file created), the IP address of the device to poll, and the SNMP credentials needed to access the SNMP agent. In the configuration file (Fig. 2) each target device will typically have several queries associated with it. Queries are made up of a query name, interval at which to poll the device in seconds, the type of SNMP request, and SNMP OIDs to send to the device's SNMP agent. Hydra can perform 2 types of SNMP requests: Get's and Walk's. A Get request is a one-to-one poll. One OID gets sent to the SNMP agent and one response is received. A Walk is used to get the contents of a table from the SNMP agent. One OID is sent and many responses are received.

```
{
      "DEVICE NAME": "XT2S”,
      "VERSION”: "3”,
      “TARGET_IP”: “9.9.9.9”
      "AUTHPASS": “PASSL”,
      "AUTHPROTO": "SHA:”,
      "PRIVPASS": “PASS2”,
      "PRIVACY": "DES”,
      "SECURITYLEVEL": "AUTHPRIV”,
      'USER_ COMUNITY'': "USERL'',
      "QUERIES": [
       {
        "QUERY_NAME'':  "WANINTERFACES",
        "INTERVAL": 120,
        "OIDS': [
           "IF-MIB:: IFTABLE''
        ],
        "TYPE": "WALK"
       }
}
```

Fig. 2 Snippet of Hydra SNMP configuration file

The configuration section for a MySQL database target (Fig. 3) follows the same paradigm as the SNMP section. A MySQL target consists of a server that contains the IP address of the database, the credentials needed to access the database, and queries to be executed on the target. These queries consist of a query name, database table to poll, and the interval at which to poll the table.

```
“NMS”:{
  “SERVERS”:[
  {
    “SERVER_NAME”:”XOEM”,
    “PASSWORD”:”XXXX”,
    “TARGET_IP”:”9.9.9.9”,
    “TYPE”:”MYSQL”,
    “USER”:”UUSER123”,
    “QUERIES”:[
      {
        “QUERY_NAME”:”QUERYNAME1”,
        “DATABASE”:”DATA12”,
        “INTERVAL”:60,
        “TABLE”:”TABLE1”
      }
      {
        “QUERY_NAME”:”QUERYNAME2”,
        “DATABASE”:”DATA12”,
        “INTERVAL”:60,
        “TABLE”:”TABLE2”
      }
    ]
  }
  ]
}
```

Fig. 3 Hydra MySQL section snippet

A BLOb file will be created for each query in the configuration file. When the polling interval is up, Hydra will poll the network device and write whatever data it receives back into a data cut. A cut is made up of a cut header, which contains metadata about the cut and the data returned from the poll.

4.1 Status Codes

Each cut contains a field with the cut status code (Tables 1 and 2). This code is used to determine if a poll was successful or if an error occurred. There are several errors that can occur, and these are reflected by several status codes.

Table 1 MySQL status codes

Code	Explanation
0	Success
E	Library error
C	Connection error
F	MySQL database Permissions needed

Table 2 SNMP status codes

Code	Explanation
0	Success.
M	Timeout. SNMP agent could not be reached.
4	Empty table.
P	Wrong authentication username or password.
U	Unknown SNMP username.
D	Incorrect encryption protocol.
S	Incorrect security level.

Each cut contains a sequence number. This number is incremented each time a data cut is written to the BLOb. This sequence number is used to ensure that each poll that was sent is recorded in the BLOb file. It also helps to ensure all data cuts have been accounted for when the BLOb files are being processed post-collection. Hydra BLObs use the naming paradigm of "<devicename>_<queryname> _fileopentime.msb". Therefore, all BLObs with this device and query name can be thought of as one file stream. The number that the active file stream Hydra is writing to depends on the number of queries defined in the configuration file. During data collection, Hydra will close a BLOb file and open a new one when either 1,800 s have passed since the file has been opened or the file has reached 500 MB in size. When Hydra closes a file and opens another one, the sequence number in the new file does not start back at zero but continues incrementing from where it left off in the previous file.

4.2 Request Time

Each cut header also contains a poll request time. This is the time in Coordinated Universal Time (UTC) when the poll was sent out. Associating a request time with each poll allows the opportunity to create a "time window" an analyst could focus on during postprocessing.

4.3 Hydra BLOb Metadata

When a new BLOb file is created, a section of XML-formatted metadata is written to the file. This metadata contains all the information necessary for successful postprocessing of the BLOb file.

5. Data Processing

5.1 Hydra Data Processing Framework

All device status data collected by Hydra can be processed in a similar way that involves grouping related data and applying transformations. The Hydra data processing framework provides common functionality for processing Hydra data, and it enables developers to quickly implement modules for specific types of Hydra data.

The Hydra data processing framework is used when defining cut modules to process Hydra data for the broader-scope high-performance-computing (HPC) framework, discussed in Panneton and Adametz's framework report.[7] Processing for specific data simply extends and overrides the Hydra data framework as needed. As with other HPC cut modules, the new specific Hydra data processing cut modules are automatically detected and used at runtime.

5.1.1 Basic Components

The Hydra data processing framework provides an object-oriented hierarchy for organizing data processing within an HPC framework cut module. The components, from highest to lowest level, are the Hydra data type, table group, table, and table row components (Fig. 4).

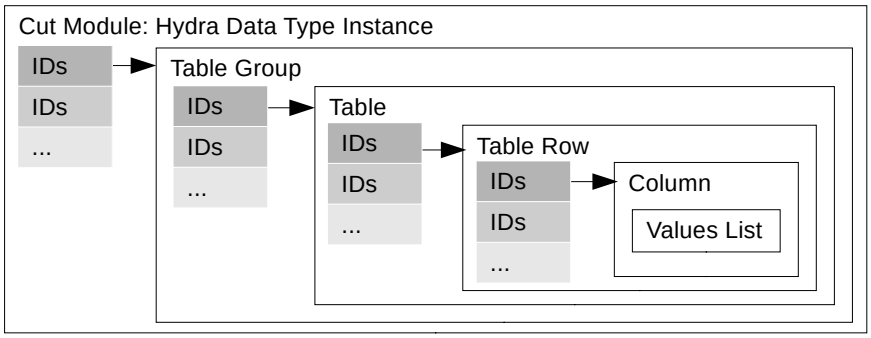

Fig. 4 Hydra data framework components. Incoming data are handled by the specific module instance for a Hydra data type, being passed to its proper table group, table, and then table row.

The Hydra data type framework component is a framework cut module for Hydra data cuts with a specific type of Hydra data. The component provides base definitions for configuration-based options, detecting and loading all its table group components, loading information about files it is reading, filtering incoming Hydra data cuts based on Hydra data types, processing data, communicating for distributed processing, and writing data outputs. Configuration-based options are passed down to the lower-level components. Specific modules must define what type of Hydra data they accept, must define how to partition data into table groups, may provide specific code for processing its type of data, and may override other methods or fields as necessary.

The table group framework component contains several related tables of Hydra data. The component provides base definitions for detecting and loading all its table components, processing data, and writing data outputs. Specific modules must define how to partition data into tables and may override other methods or fields as necessary.

The table framework component provides base definitions for finding and creating table rows and writing data outputs. Specific modules must define how to partition data into rows and process data.

The table row framework component provides base definitions for processing data, setting column values, determining whether all columns have a value, and writing data outputs. Specific modules should override how to determine whether data are applicable to the row and whether data are valid, and they may override how data points are mapped to columns.

At all levels, the Hydra data processing framework has common methods for outputting data to HDF (Hierarchical Data Format) or CSV (Comma-Separated Values) format files. The default is HDF, but output formats for each Hydra module can be selected independently.

5.1.2 Map Component

The Hydra data processing framework provides a map component for applying transformations, such as look-up, translation, and calculation functionality (Fig. 5). The map component is a member of the table framework component because all table rows share a map component with its defined transformations. The map component inherits common methods from the Hydra data processing framework and can override and extend the definition for the specific table. The map methods are easy to reuse as they are referenced by name and can be chained together.

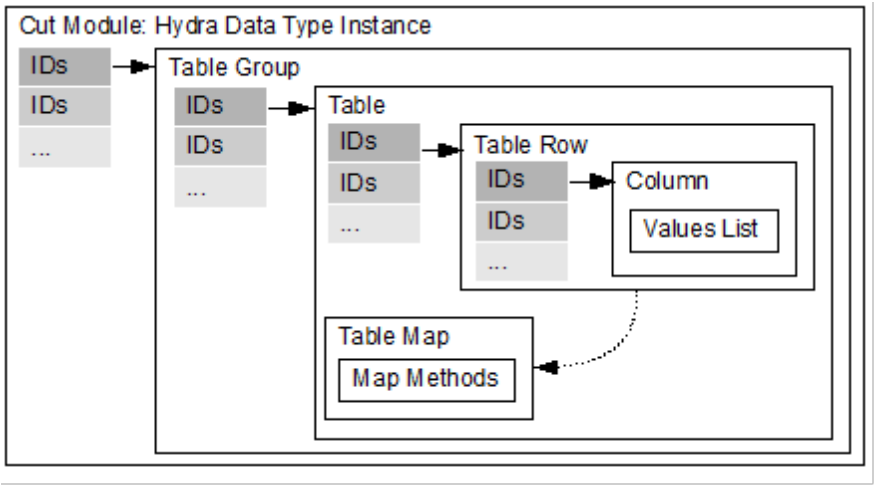

Fig. 5 Hydra data framework, including map components. The postmap methods can be called from any level to apply transformations across columns in a row, rows in a table, or tables in a group.

If one or more map methods are specified for a column in a table row, then a chain of map methods is executed when assigning a value. The chain includes one input method that gathers the appropriate data to start the chain and one or more transformation methods that comprise the chain. The output of the final transformation in the chain is used as the final column value for data outputs.

The map framework also supports several features. Map input methods can be provided as default values to start the chain in case it cannot gather the appropriate data. The map methods also can reference framework configuration entries to change behavior or reference default values. Values along each step of the chain are saved to support methods that also require previous values, such as minimum and maximum functions.

5.1.3 Postmap Methods

The Hydra data processing framework provides postmap methods to support column values that depend on other data. Each framework level has a postmap method to allow resolving data dependencies that span multiple columns in a row, rows in a table, or tables in a table group. The most common case is a derived column that depends on the values of one or more other columns.

Postmap methods must be overridden by the specific modules. Postmap methods are executed after all data have been assigned to values such that any derived values that can be determined are assigned.

5.1.4 Data Flow

During the HPC framework's registration process in a specific Hydra data cut module, Hydra table instances report which identifiers they process to their corresponding table groups. The table group instances accumulate all the identifiers from their table instances and report the aggregate to the top-level instance for the Hydra data type. The Hydra data type instance will report the cut identifier for Hydra data up to the HPC framework.

The Hydra data processing framework's highest-level component, a specific Hydra data type instance, first receives a Hydra data cut from the HPC framework. The specific module will only continue processing if the Hydra cut data type is the type it accepts. It may optionally include specific data processing definitions before selecting a table group to continue processing the data cut.

A table group framework instance may further parse the data cut identifier to determine which of its tables (one or more) should continue processing the data cut. A table instance performs a function similar to the table group, except that it also finds or creates a new table row instance to continue processing the data.

A table row instance will determine which column the data should be used for and whether the data is valid. The row will assign a value to the column, optionally sending the data through the table's map instance to invoke any map methods that were specified for that column. The map process, as described in the previous section, will use a map inputs method and then a chain of one or more map methods. The final output of the map chain will be the output value for the column.

On the way back up the hierarchy, postmap methods are invoked at each level. The table row instance will invoke any postmap methods to fill any columns that depend on one or more other columns in that row. The table will invoke any postmap methods to fill in any columns that depend on values in columns from other rows in the table. Lastly, the table group will invoke any postmap methods to fill in any columns that depend on values in other columns from other tables.

5.1.5 Distributed Processing Considerations

Since the HPC framework distributes processing on a per-file basis, the Hydra data processing framework behaves as described in the previous section for each cut on a per-file basis. Hydra data are typically structured such that all related data cuts are in a single file. The top-level Hydra data–type modules can be configured to handle the case where related data cuts are split across files.

A worker by default writes all outputs, whether the rows are complete or incomplete, after it has accumulated all the data from a single input file. Complete means there is a value for each column in the row. Alternately, the Hydra data processing framework can be configured such that workers output complete rows and send the incomplete rows to the receiver. In this case, the receiver will merge the data from corresponding incomplete rows, then run the postmap methods to assign values for any derived columns. If there is a collision when merging data, then a warning is logged, and the value with the latest timestamp is selected. Finally, the receiver will output all rows it received.

5.2 Specific Hydra Data Processing

Specific Hydra data processing modules inherit functionality from the Hydra data processing framework described previously and add specific functionality to process a specific type of data.

5.2.1 SNMP Data Processing

5.2.1.1 Data Description

Section 2 introduced SNMP device status data. Hydra SNMP data cuts, introduced in Section 4, contain the common metadata, SNMP-specific metadata, and a list of one or more SNMP data variables.

The common cut metadata includes poll request and response times, a status code, and a sequence number as described in Section 4. SNMP-specific metadata includes a query identifier string to match each cut with the Hydra configuration and a subsequent number that distinguishes cuts that are part of a single SNMP walk iteration for a sequence number.

SNMP data variables are name-value pairs defined by the SNMP SMI, interpreted and output by the Net-SNMP library running on Hydra. Names are SNMP OIDs. Values may be data entries optionally labeled with a data type, hints, and units; or values may be error or warning messages.

5.2.1.2 Processing Components

The SNMP processing has its own base library that inherits from the common Hydra data processing framework and provides the basis for all its table groups, tables, table rows, and table maps. The top-level specific module adds functionality to request Hydra SNMP cuts and uses the query identifier strings to pass cuts to their appropriate table groups.

The SNMP table group base also uses the query identifier strings to pass cuts to their appropriate tables. The specific table groups define their table group names and may also implement postmap methods for handling data dependencies across tables.

The SNMP table base defines how to split up most SNMP OIDs into components that identify the table row and table column. It also updates a map that tracks the minimum response time for a given sequence number from a specific Hydra collector, power cycle, and device with applicable data. This minimum response time is the closest single time value to when the data were last updated on the device; this value is used for the response timestamp in SNMP outputs.

The specific SNMP tables request which query identifier strings to use and their table names. They may override how to split SNMP OIDs into row and column components or implement postmap methods for handling data dependencies across table rows.

The SNMP table row base defines the methods to determine whether SNMP variables are valid and applicable. Note that error and warning messages and other data anomalies in well-formed SNMP response variables must be detected in these methods. The timestamp column is overridden to use a map to get the minimum response time for that row.

The specific SNMP table rows define all their specific columns, along with their column identifiers, output names, data types, overwrite behaviors, and maps. They may also define postmap methods for handling data dependencies across columns.

The SNMP map base defines maps that are specific to interpreting and converting SNMP datatypes to desired column output values. Specific SNMP map instances may add additional maps that are specific to data for their associated tables.

5.2.2 NMS Data Processing

5.2.2.1 Data Description

Section 2 introduced NMS device status data. Hydra SQL data cuts, used for NMS data processing, contain the common metadata, SQL-specific metadata, and a list of the SQL response entries.

Each time an SQL database table is polled, a “snapshot” of the database table is returned. Each new poll will have the previous data in the table (if it was not removed by the database) in addition to any new rows in the table.

A Hydra NMS file is broken into 3 sections. The beginning of each Hydra NMS file has a config section that contains the name of the SQL database table that was

queried. The data section in the middle of the file comprises data cuts that hold the queried output from the NMS polls. The output contains any rows from the database table query, written in CSV format. The end of each Hydra NMS file may have an EXT_CONFIG metadata tag that describes each column name and the database data type of its value.

5.2.2.2 Processing Components

The NMS process uses the previously discussed Hydra framework and is further broken down into table groupings. Each table grouping defines one or more associated tables, which are python modules that are each responsible for reducing data from a single database table. Each python module specifies its outputs by selecting values from the data cuts.

Hydra NMS data cuts are sent to python modules responsible for processing that database table's data, determined by table name in the file EXT_CONFIG metadata. If the table name is not defined for processing, a warning message is issued and that data are not processed.

Every NMS python module has a set of default column headers and value type pairs given in the order the values are written in the Hydra NMS data cut CSV. These default column and value types are loaded from a JSON file. The default column headers are used to identify the values in the cut data in the event that the column headers are not defined in the EXT_CONFIG. Column headers typically do not change during a test; using a set of default columns allows processing of the data even if the columns are not explicitly in the EXT_CONFIG metadata.

Once the column headers and data types are established from the JSON defaults or the EXT_CONFIG, data values are mapped to column names and data types. The column names are in respective order to the values in the Hydra NMS data cut. Each value is paired with its column header and converted to the defined data type.

The python table module can apply a postmap function on the value if additional processing is required before writing to the outputs.

6. Conclusion

ATC developed an active device status collector and collaborated with ARL to develop an HPC solution for correlating device status data with network traffic data. Correlating device status and network status data enables analysts to see both network traffic data (what happened) and related device status data (why it happened) to provide more insights into network performance and reliability.

7. References and Notes

1. These are event-driven reporting methods. When a specific type of event occurs (trap), a message is generated that can be recorded. Refer to RFC-1215 (Rose M, editor. A convention for defining traps for use with the SNMP. Fremont (CA): Internet Engineering Task Force (IETF); 1991 Mar [accessed 2015 Jan 15]. http://www.ietf.org/rfc/rfc1215.txt?number=1215.) and Net-SNMP (2013 Feb 26 [accessed 2015 Jan 15]. http://www.net-snmp.org/) for more details.

2. Case J, Fedor M, Schoffstall M, Davin J. A simple network management protocol (SNMP). Fremont (CA): Internet Engineering Task Force; 1990 May [accessed 2015 Jan 15]. http://www.ietf.org/rfc/rfc1157.txt?number=1157.

3. Postel J. Internet control message protocol. Fremont (CA): Internet Engineering Task Force; 1981 Sep [accessed 2015 Jan 29]. http://tools.ietf.org/html/rfc792.

4. Oracle Corp. MySQL documentation: MySQL reference manuals. Reston (VA): Oracle Corp; 2015 [accessed 15 Jan 2015]. http://dev.mysql.com/doc/.

5. Army Aberdeen Test Center (US). VISION BLOb description. Aberdeen Proving Ground (MD): Army Aberdeen Test Center (US); 2014.

6. JSON.org. Introducing JSON; n.d. [accessed 2015 Jan 15]. http://www.json.org/.

7. Panneton B, Adametz J. High-bandwidth tactical-network data analysis in a high-performance-computing (HPC) environment: data reduction framework. Aberdeen Proving Ground (MD): Army Research Laboratory (US); 2015 Sep. Report No.: ARL-CR-0777.

List of Symbols, Abbreviations, and Acronyms

ARL	US Army Research Laboratory
ATC	US Army Aberdeen Test Center
BLOb	binary large object
CSV	Comma-Separated Value
HDF	Hierarchical Data Format
HPC	High-Performance Computing
IP	Internet Protocol
JSON	JavaScript Object Notation
NMS	Network Management System
OID	Object Identifier
SMI	Structure of Management Information
SNMP	Simple Network Management Protocol
SQL	Structured Query Language

1 (PDF)	DEFENSE TECHNICAL INFORMATION CTR DTIC OCA
2 (PDF)	DIRECTOR US ARMY RESEARCH LAB RDRL CIO LL IMAL HRA MAIL & RECORDS MGMT
1 (PDF)	GOVT PRINTG OFC A MALHOTRA
1 (PDF)	TECH AND PROJ ENGR LLC B PANNETON
2 (PDF)	USATC B TAURAS S COYNE
1 (PDF)	DIR USARL RDRL CIH C K RENARD

INTENTIONALLY LEFT BLANK.