arts++

Daniel McRobb, CAIDA Original Aug 1998 ; last modified Nov 22, 1999

This document describes arts++, a C++ class library and set of simple utilities for processing ARTS files produced by CAIDA software.

Contents

1 Overview

`arts++` is a set of C++ classes and applications for handling ARTS data files produced by CAIDA software (`cflowd` and `skitter`). Specifically, the `arts++` package handles the following ARTS data types:

- AS matrix (version 0)
- net matrix (version 2)
- port table (version 2)
- port matrix (version 0)
- selected port table (version 0)
- protocol table (version 2)
- TOS table (version 0)
- interface matrix (version 0)
- nexthop table (version 0)
- forward IP path and RTT (version 0)
- BGP4 route table (version 0)
- RTT time series table (version 0)

2 ARTS Data Types Handled by `arts++`

There are many types of data stored in ARTS files. `arts++` handles a subset of ARTS data types.

2.1 AS matrix (version 0)

The AS matrix contains counters for traffic (packets and bytes) from source ASes to destination ASes. It is a sparse matrix, having only entries for which traffic information is stored. Since this data is typically collected by `cflowd`, an AS matrix normally contains counters for sourceAS:destinationAS pairs for which a particular Cisco forwarded traffic.

When AS matrix data is stored in a file, it is stored with a period attribute which represents the time interval in which the traffic was seen. It is also stored with a host attribute indicating the router from which the data was collected. Finally, it may also have an interface index attribute indicating the input interface on which the data was seen, an interface description attribute indicating the name of the input interface, and an interface IP address attribute indicating the IP address of the input interface.

2.2 net matrix (version 2)

The net matrix contains counters for traffic (packets and bytes) from source networks to destination networks. Networks are identified by network number and netmask length. It is a sparse matrix, having only entries for which traffic information is stored. Since this data is typically collected by `cflowd`, a net matrix normally contains counters for sourceNetwork:destinationNetwork pairs for which a particular Cisco forwarded traffic.

When net matrix data is stored in a file, it is stored with a period attribute which represents the time interval in which the traffic was seen. It is also stored with a host attribute indicating the router from which the data was collected. Finally, it may also have an interface index attribute indicating the input interface on which the data was seen, an interface description attribute indicating the name of the input interface, and an interface IP address attribute indicating the IP address of the input interface.

2.3 port table (version 2)

The port table contains counters for input and output traffic (packets and bytes) versus transport layer port number. Input counters represent traffic destined for the port while output counters represent traffic sourced from the port. For example, the input counters for port 80 would normally indicate the amount of traffic sent from Web browsers to Web servers, while the output counters for port 80 would normally indicate the amount of traffic sent from Web servers to Web browsers.

The table is sparse; there are no entries for ports on which no traffic was seen. Since this data is typically collected by `cflowd`, a port table normally contains counters for traffic forwarded by a particular Cisco.

When port table data is stored in a file, it is stored with a period attribute which represents the time interval in which the traffic was seen. It is also stored with a host attribute indicating the router from which the data was collected. Finally, it may also have an interface index attribute indicating the input interface on which the data was seen.

2.4 port matrix (version 0)

The port matrix contains counters for traffic from source ports to destination ports. Unlike the port table, this object retains the source to destination port relationship.

When port matrix data is stored in a file, it is stored with a period attribute which represents the time interval in which the traffic was seen. It is also stored with a host attribute indicating the router from which the data was collected. Finally, it may also have an interface index attribute indicating the input interface on which the data was seen, an interface description attribute indicating the name of the input interface, and an interface IP address attribute indicating the IP address of the input interface.

2.5 selected port table (version 0)

The selected port table contains counters for input and output traffic (packets and bytes) versus transport layer port number for a set of ports, plus one additional counter for all other ports (summed traffic across all other ports). Input counters represent traffic destined for the port while output counters represent traffic sourced from the port. For example, the input counters for port 80 would normally indicate the amount of traffic sent from Web browsers to Web servers, while the output counters for port 80 would normally indicate the amount of traffic sent from Web servers to Web browsers.

The table is sparse; there are no entries for ports on which no traffic was seen. Since this data is typically collected by `cflowd`, a port table normally contains counters for traffic forwarded by a particular Cisco.

The difference between this object and a plain port table: the selected port table contains inidividual port entries for only selected ports, and lumps all other data under the port 0 entry. Inside the selected port table, there is an `ArtsPortChooser` object which contains the ports chosen when the object was created. Data is only counted *once*: we put it under the lower of the source or destination ports that is in the `ArtsPortChooser`, or put it under port 0 if neither the source nor destination port was in the `ArtsPortChooser`.

This object is usually generated by using port matrix objects and an `ArtsPortChooser` object as input to the `ArtsPortMatrixAggregator` class. This permits a simple user configurable means of boiling port matrix data down to data usable for tracking per-application traffic patterns. This functionality is available in the *artsportmagg(l)* utility.

When selected port table data is stored in a file, it is stored with a period attribute which represents the time interval in which the traffic was seen. It is also stored with a host attribute indicating the router from which the data was collected. Finally, it may also have an interface index attribute indicating the input interface on which the data was seen, an interface description attribute indicating the name of the input interface, and an interface IP address attribute indicating the IP address of the input interface.

2.6 protocol table (version 2)

The protocol table contains counters (packets and bytes) versus IP protocol (TCP, UDP, ICMP, IGMP, et. al.). The table is sparse; there are no entries for protocols that were not seen in the measured traffic. Since this data is typically collected by `cflowd`, a protocol table normally contains counters for traffic forwarded by a particular Cisco.

When protocol table data is stored in a file, it is stored with a period attribute which represents the time interval in which the traffic was seen. It is also stored with a host attribute indicating the router from which the data was collected. Finally, it may also have in interface index attribute indicating the input interface on which the data was seen, an interface description attribute indicating the name of the input interface, and an interface IP address attribute indicating the IP address of the input interface.

2.7 TOS table (version 0)

The TOS table contains counters (packets and bytes) versus IP TOS (Type Of Service). The table is sparse; there are no entries for TOS values that were not seen in the measured traffic. Since this data is typically collected by `cflowd`, a TOS table normally contains counters for traffic forwarded by a particular Cisco.

When TOS table data is stored in a file, it is stored with a period attribute which represents the time interval in which the traffic was seen. It is also stored with a host attribute indicating the router from which the data was collected. Finally, it may also have an interface index attribute indicating the input interface on which the data was seen, an interface description attribute indicating the name of the input interface, and an interface IP address attribute indicating the IP address of the input interface.

2.8 interface matrix (version 0)

The interface matrix contains counters (packets and bytes) for traffic from input interfaces to output interfaces. Input and output interfaces are identified by their index (ifIndex); the matrix can be viewed as having rows for input interfaces and columns for output interfaces.

When interface matrix data is stored in a file, it is stored with a period attribute which represents the time interval in which the traffic was seen. It is also stored with a host attribute indicating the router from which the data was collected. Finally, it may also have an interface index attribute indicating the input interface on which the data was seen, an interface description attribute indicating the name of the input interface, and an interface IP address attribute indicating the IP address of the input interface.

2.9 nexthop table (version 0)

The nexthop table contains counters (packets and bytes) versus IP nexthop. Since this data is typically collected by `cflowd`, a nexthop table normally contains counters for traffic forwarded by a particular Cisco.

When nexthop table data is stored in a file, it is stored with a period attribute which represents the time interval in which the traffic was seen. It is also stored with a host attribute indicating the router from which the data was collected. Finally, it may also have an interface index attribute indicating the input interface on which the data was seen, an interface description attribute indicating the name of the input interface, and an interface IP address attribute indicating the IP address of the input interface.

2.10 forward IP path and RTT (version 0)

2.11 BGP4 route table (version 0)

The BGP4 route table contains BGP4 information for a set of unique IPv4 network prefixes (usually the 'chosen' routes, i.e. those used for forwarding). This object is currently experimental and is used by some analysis and visualization tools in the `skitter` package with data collected by `mrtd`. [1]

When BGP4 route table data is stored in a file, it is stored with a creation attribute which represents the time at which the 'snapshot' of the BGP4 route information was taken. It is also stored with a host attribute indicating the router from which the data was obtained.

2.12 RTT time series table (version 0)

The RTT time series table contains round-trip time information for a source and destination host. This object also stores dropped packet information (packet loss).

When RTT time series table data is stored in a file, it is stored with a creation attribute which represents the time at which the first RTT measurement was taken. It is also stored with a host pair attribute indicating the source and destination of the RTT measurement.

3 C++ Class Library

The `arts++` package includes a C++ class library which provides an API for ARTS data. The class library contains a top-level `Arts` class which may contain any type of ARTS data. The class library also contains derived classes for each of the ARTS data objects used by CAIDA.

3.1 The `Arts` Class and Contained Classes

The `Arts` class is a top level class in the class heirarchy. It is used to hold any of the ARTS data types.

[1] mrtd is part of the Multi-Threaded Routing Toolkit. See http://www.merit.edu/ mrt/

All `Arts` objects contain a header section represented by a contained `ArtsHeader` object. `Arts` objects also normally contain one or more attributes in a vector of `ArtsAttribute` objects (literally a `vector<ArtsAttribute>` using STL). Finally, each `Arts` object contains a data portion. The data portion is where the specific type of data (AS matrix, net matrix, etc.) is stored.

You may obtain a reference to the `ArtsHeader` in an `Arts` object by calling the `Header()` member from the `Arts` class.You may obtain a reference to the vector of attributes in an `Arts` object or one of the `Arts` descendants by calling the `Attributes()` member from the `Arts` class.

The type of the data in the data portion of an `Arts` object can be determined by looking at the `Header().Identifier()` in the object. The class library will handle the following values:

Header().Identifier()	*Description*
`artsC_OBJECT_AS_MATRIX`	An AS matrix object.
`artsC_OBJECT_NET`	A net matrix object.
`artsC_OBJECT_PORT`	A port table object.
`artsC_OBJECT_PORT_MATRIX`	A port matrix object.
`artsC_OBJECT_SELECTED_PORT`	A selected port table object.
`artsC_OBJECT_PROTO`	A protocol table object.
`artsC_OBJECT_TOS`	A TOS table object.
`artsC_OBJECT_IP_PATH`	An IP forward path and RTT object.
`artsC_OBJECT_INTERFACE_MATRIX`	An interface matrix object.
`artsC_OBJECT_NEXT_HOP`	An IP nexthop table object.
`artsC_OBJECT_BGP4`	A BGP4 route table object.
`artsC_OBJECT_RTT_TIME_SERIES`	A RTT time series object.

3.1.1 The `ArtsHeader` Class

The `ArtsHeader` class is used inside each `Arts` object to hold information which is generic to all ARTS data types. For example, the object identifier, the version of the object, or the number of attributes in the object.

3.1.2 The `ArtsAttribute` Class

A vector of `ArtsAttribute` objects is used inside an `Arts` object to hold various attributes which may apply to a variety of ARTS data types. For example, a period attribute is used in several objects to hold a time interval for which the ARTS data holds information. A host attribute is frequently used to hold the IP address of a host or router from which the ARTS data was collected.

3.1.3 The `ArtsAsMatrixData` Class

The `ArtsAsMatrixData` class is used to hold AS matrix data within an `Arts` object. When the `Header().Identifier()` of an `Arts` object is `artsC_OBJECT_AS_MATRIX`, you may obtain a pointer to the `ArtsAsMatrixData` object in the `Arts` object by calling the `AsMatrixData()` member from the `Arts` class.

An `ArtsAsMatrixData` object contains a vector of `ArtsAsMatrixEntry` objects.

The `ArtsAsMatrixEntry` Class The `ArtsAsMatrixEntry` class contains a source AS number, a destination AS number, the packets sent from the source AS to the destination AS, and the bytes sent from the source AS to the destination AS.

3.1.4 The ArtsNetMatrixData Class

The `ArtsNetMatrixData` class is used to hold net matrix data within an `Arts` object. When the `Header().Identifier()` of an `Arts` object is `artsC_OBJECT_NET`, you may obtain a pointer to the `ArtsNetMatrixData` object in the `Arts` object by calling the `NetMatrixData()` member from the `Arts` class.

An `ArtsNetMatrixData` object contains a vector of `ArtsNetMatrixEntry` objects.

The `ArtsNetMatrixEntry` Class The `ArtsNetMatrixEntry` class contains a source network address and netmask length, a destination network address and netmask length, the packets sent from the source network to the destination network, and the bytes sent from the source network to the destination network.

3.1.5 The ArtsPortTableData Class

The `ArtsPortTableData` class is used to hold port table data within an `Arts` object. When the `Header().Identifier()` of an `Arts` object is `artsC_OBJECT_PORT`, you may obtain a pointer to the `ArtsPortTableData` object in the `Arts` object by calling the `PortTableData()` member from the `Arts` class.

An `ArtsPortTableData` object contains a vector of `ArtsPortTableEntry` objects.

The `ArtsPortTableEntry` Class The `ArtsPortTableEntry` class contains a port number, the packets sent with a source port equal to the port number, the bytes sent with a source port equal to the port number, the packets sent with a destination port equal to the port number and the bytes sent with a destination port equal to the port number.

3.1.6 The ArtsPortMatrixData Class

The `ArtsPortMatrixData` class is used to hold port matrix data within an `Arts` object. When the `Header().Identifier()` of an `Arts` object is `artsC_OBJECT_PORT_MATRIX`, you may obtain a pointer to the `ArtsPortMAtrixData` object in the `Arts` object by calling the `PortMatrixData()` member from the `Arts` class.

An `ArtsPortMatrixData` object contains a vector of `ArtsPortMatrixEntry` objects.

The `ArtsPortMatrixEntry` Class The `ArtsPortMatrixEntry` class contains a source port number, a destination port number, the number of packets sent from the source port number to the destination port number, and the number of bytes sent from the source port number to the destination port number.

3.1.7 The ArtsSelectedPortTableData Clsss

The `ArtsSelectedPortTableData` class is used to hold selected port table data within an `Arts` object. When the `Header().Identifier()` of an `Arts` object is `artsC_OBJECT_SELECTED_PORT`, you may obtain a pointer to the `ArtsSelectedPortTableData` object in the `Arts` object by calling the `SelectedPortTableData()` member from the `Arts` class.

The data portion of this object is stored just like that in the `ArtsPortTableData` class: a vector of `ArtsPortTableEntry` objects, accessible via the `PortEntries()` member function. However, the interpretation of the data is different than that of the `ArtsPortTableData` class. Only selected ports (which are held in an `ArtsPortChooser` object in the `ArtsSelectedPortTableData` class) are allowed to have inidividual entries, and traffic for ports which aren't in the selection list is lumped unoder an entry for port 0. Unlike the `ArtsPortTableData` class, we only count traffic once, for the lower of the source or destination port that was in the selection list (or in the entry for port 0).

3.1.8 The `ArtsProtocolTableData` Class

The `ArtsProtocolTableData` class is used to hold protocol table data within an `Arts` object. When the `Header().Identifier()` of an `Arts` object is `artsC_OBJECT_PROTO`, you may obtain a pointer to the `ArtsProtocolTableData` object in the `Arts` object by calling the `ProtocolTableData()` member from the `Arts` class.

An `ArtsProtocolTableData` object contains a vector of `ArtsProtocolTableEntry` objects.

The `ArtsProtocolTableEntry` Class The `ArtsProtocolTableEntry` class contains a protocol number, the number of packets sent using the protocol and the number of bytes sent using the protocol.

3.1.9 The `ArtsTosTableData` Class

The `ArtsTosTableData` class is used to hold TOS table data within an `Arts` object. When the `Header().Identifier()` of an `Arts` object is `artsC_OBJECT_TOS`, you may obtain a pointer to the `ArtsTosTableData` object in the `Arts` object by calling the `TosTableData()` member from the `Arts` class.

An `ArtsTosTableData` object contains a vector of `ArtsTosTableEntry` objects.

The `ArtsTosTableEntry` Class The `ArtsTosTableEntry` class contains a TOS value, the number of packets sent using the TOS and the number of bytes sent using the TOS.

3.1.10 The `ArtsInterfaceMatrixData` Class

The `ArtsInterfaceMatrixData` class is used to hold interface matrix data within an `Arts` object. When the `Header().Identifier()` of an `Arts` object is `artsC_OBJECT_INTERFACE_MATRIX`, you may obtain a pointer to the `ArtsInterfaceMatrixData` object in the `Arts` object by calling the `InterfaceMatrixData()` member from the `Arts` class. The interface matrix data is held in a vector of `ArtsInterfaceMatrixEntry` objects inside the `ArtsInterfaceMatrixData` class.

The `ArtsInterfaceMatrixEntry` Class The `ArtsInterfaceMatrixEntry` class contains a source (input) interface index, a destination (output) interface index, the number of packets sent from the source interface to the destination interface and the number of bytes sent from the source interface to the destination interface.

3.1.11 The `ArtsNextHopTableData` Class

The `ArtsNextHopTableData` class is used to hold nexthop table data within an `Arts` object. When the `Header().Identifier()` of an `Arts` object is `artsC_OBJECT_NEXT_HOP`, you may obtain a pointer to the `ArtsNextHopTableData` object in the `Arts` object by calling the `NextHopTableData()` member from the `Arts` class. The nexthop table data is held in a vector of `ArtsNextHopTableEntry` objects inside the `ArtsNextHopTableData` class.

The `ArtsNextHopTableEntry` Class The `ArtsNextHopTableEntry` class contains the IP address of the nexthop and counters for the packets and bytes of traffic sent via the nexthop.

3.1.12 The `ArtsIpPathData` Class

Not yet documented, used by the `skitter` package.

3.1.13 The `ArtsBGP4RouteTableData` Class

Not yet documented, used by the `skitter` package.

3.1.14 The `ArtsRttTimeSeriesTable` class

Not yet documented, used by `skping` and `skpingd` from the `skitter` package.

3.2 Derived Classes

For each of the ARTS data types, there is a class derived from the `Arts` class to represent the data. An applicaiotn would typically use one of the derived classes when it knows what type of data to be stored in a given object. Typically these classes are no different than the `Arts` class other than having specialized constructors and shortcut calls to members of the data classes held inside an ARTS object.

3.2.1 ArtsAsMatrix

This class can be used to hold AS matrix data. An instantiation of this object is no different than an instantiation of an `Arts` object except that the `Header().Identifier()` should always be `artsC_OBJECT_AS_MATRIX`.

3.2.2 ArtsNetMatrix

This class can be used to hold net matrix data. An instantiation of this object is no different than an instantiation of an `Arts` object except that the `Header().Identifier()` should always be `artsC_OBJECT_NET`.

3.2.3 ArtsPortTable

This class can be used to hold port table data. An instantiation of this object is no different than an instantiation of an `Arts` object except that the `Header().Identifier()` should always be `artsC_OBJECT_PORT`.

3.2.4 ArtsPortMatrix

This class can be used to hold port matrix data. An instantiation of this object is no different than an instantiation of an `Arts` object except that the `Header().Identifier()` should always be `artsC_OBJECT_PORT_MATRIX`.

3.2.5 ArtsSelectedPortTable

This class can be used to hold selected port table data. An instantiation of this object is no different than an instantiation of an `Arts` object except that the `Header().Identifier()` should always be `artsC_OBJECT_SELECTED_PORT`.

3.2.6 ArtsProtocolTable

This class can be used to hold protocol table data. An instantiation of this object is no different than an instantiation of an `Arts` object except that the `Header().Identifier()` should always be `artsC_OBJECT_PROTO`.

3.2.7 ArtsTosTable

This class can be used to hold TOS table data. An instantiation of this object is no different than an instantiation of an `Arts` object except that the `Header().Identifier()` should always be `artsC_OBJECT_TOS`.

3.2.8 ArtsInterfaceMatrix

This class can be used to hold interface matrix data. An instantiation of this object is no different than an instantiation of an `Arts` object except that the `Header().Identifier()` should always be `artsC_OBJECT_INTERFACE_MATRIX`.

3.2.9 ArtsNextHopTable

This class can be used to hold IP nexthop table data. An instantiation of this object is no different than an instantiation of an `Arts` object except that the `Header().Identifier()` should always be `artsC_OBJECT_NEXT_HOP`.

3.2.10 ArtsIpPath

This class can be used to hold IP forward path data. An instantiation of this object is no different than an instantiation of an `Arts` object except that the `Header().Identifier()` should always be `artsC_OBJECT_IP_PATH`.

3.2.11 ArtsBgp4RouteTable

This class can be used to hold BGP4 route table data. An instantiation of this object is no different than an instantiation of an `Arts` object except that the `Header().Identifier()` should always be `artsC_OBJECT_BGP4`.

3.3 Helper Classes

3.3.1 The `ArtsAsMatrixAggregator` Class

The `ArtsAsMatrixAggregator` class may be used to aggregate AS matrix data in the time domain.

3.3.2 The `ArtsNetMatrixAggregator` Class

The `ArtsNetMatrixAggregator` class may be used to aggregate net matrix data in the time domain.

3.3.3 The `ArtsPortTableAggregator` Class

The `ArtsPortTableAggregator` class may be used to aggregate port table data in the time domain.

3.3.4 The `ArtsPortMatrixAggregator` Class

The `ArtsPortMatrixAggregator` class may be used to aggregate port matrix data in the time domain, generating `ArtsPortMatrix` or `ArtsSelectedPortTable` objects as aggregate output.

3.3.5 The `ArtsProtocolTableAggregator` Class

The `ArtsProtocolTableAggregator` class may be used to aggregate protocol table data in the time domain.

3.3.6 The `ArtsTosTableAggregator` Class

The `ArtsTosTableAggregator` class may be used to aggregate TOS table table data in the time domain.

3.3.7 The `ArtsInterfaceMatrixAggregator` Class

The `ArtsInterfaceMatrixAggregator` class may be used to aggregate interface matrix data in the time domain.

3.3.8 The `ArtsNextHopTableAggregator` Class

The `ArtsNextHopTableAggregator` class may be used to aggregate nexthop table data in the time domain.

3.4 More Information

For more detailed information, see the arts++ Class Library Reference.

4 Applications

4.1 Data Viewers

A handul of simple utilities for viewing the contents of ARTS files are included with `arts++`. These tools simply read the contents of ARTS files and produce output on stdout. They're really just simple examples of using the class library. However, some users may find them useful in their own right, particularly when working with aggregate data.

4.1.1 artsdump

usage: `artdsump [-v] infile(s)`

`artsdump` is a general utility for viewing the contents of an ARTS file. `artsdump` reads all ARTS data from the files specified on the command line and dumps it to stdout in a human-readable form.

Note that the output format is verbose and not amenable to post-processing. The output is not intended to be fed to other programs. `artsdump` is intended only to aid the user in quickly verifying or viewing the entire contents of an ARTS file.

See the manpage (*artsdump(l)*) for more information.

4.1.2 artstoc

usage: `artstoc [-v] infile(s)`

`artstoc` is a utility that reads ARTS objects from ARTS files and displays a one-line summary for each object on stdout. This utility is typically used to quickly check the type and quantity of objects stored in an ARTS file. See the manpage (*artstoc(l)*) for more information.

Example output:

```
% artstoc arts.19980817
  192.172.226.1  08/17/1998 05:30:44 - 08/17/1998 05:37:42 EDT  protocol table
  192.172.226.1  08/17/1998 05:30:44 - 08/17/1998 05:37:42 EDT  port table
  192.172.226.1  08/17/1998 05:30:44 - 08/17/1998 05:37:42 EDT  AS matrix
  192.172.226.1  08/17/1998 05:38:00 - 08/17/1998 05:48:03 EDT  protocol table
  192.172.226.1  08/17/1998 05:38:00 - 08/17/1998 05:48:03 EDT  port table
  192.172.226.1  08/17/1998 05:38:00 - 08/17/1998 05:48:03 EDT  net matrix
  192.172.226.1  08/17/1998 05:38:00 - 08/17/1998 05:48:03 EDT  AS matrix
  192.172.226.1  08/17/1998 05:48:03 - 08/17/1998 05:52:54 EDT  protocol table
  192.172.226.1  08/17/1998 05:48:03 - 08/17/1998 05:52:54 EDT  port table
  192.172.226.1  08/17/1998 05:48:03 - 08/17/1998 05:52:54 EDT  net matrix
  192.172.226.1  08/17/1998 05:48:03 - 08/17/1998 05:52:54 EDT  AS matrix
```

4.1.3 artsases

usage: `artsases [-v] [-p] [-i ifIndexList] [-l maxEntries] [-t AS] [-r] [-s srcAS] [-d dstAS] infile(s)`

`artsases` displays a summary of each AS matrix object in ARTS files. See the manpage (*artsases(l)*) for more information.

Example output:

```
% artsases arts.19990819
router:  192.172.226.1
ifIndex: 5 (Fddi0/1/0 198.17.46.56)
period:  08/19/1999 13:16:46 - 08/19/1999 13:23:10 EDT
  Src AS  Dst AS           Pkts       Pkts/sec          Bytes       Bits/sec
  ------  ------  -------------  -------------  -------------  -------------
       0       0          37034        96.4427       34022843         708809
     194    1909          15806        41.1615       10086355         210132
    3967    1909           9051        23.5703        8395655         174909
    1224    1909          10086        26.2656        7195383         149904
    6461    1909           4537        11.8151        4977144         103690
    5050    1909          12347        32.1536        4374027        91125.6
    3549    1909           5532        14.4062        3264083        68001.7
   10487    1909           2348        6.11458        2102355        43799.1
    1740    1909           8332        21.6979        1419569        29574.4
      38    1909          11848        30.8542        1308203        27254.2
     297    1909            592        1.54167        1229825        25621.4
    6227    1909            826        2.15104        1168984        24353.8
```

4.1.4 artsnets

usage: `artsnets [-v] [-r] [-p] [-i ifIndexList] [-l maxEntries] [-s srcNet/mask] [-d dstNet/mask] infile(s)`

`artsnets` displays a summary of each net matrix object in ARTS files. See the manpage (*artsnets(l)*) for more information.

Example output:

```
% artsnets arts.19990819
router:  192.172.226.1
ifIndex: 5 (Fddi0/1/0 198.17.46.56)
period:  08/19/1999 13:16:46 - 08/19/1999 13:23:10 EDT
         Src Network         Dst Network           Pkts          Bytes
  ------------------  ------------------  -------------  -------------
   128.223.161.22/32    224.2.128.143/32          16407       21366381
     192.52.106.0/24  192.172.226.128/27          15606        9822154
      141.142.0.0/16  192.172.226.128/27           9874        6936648
        209.1.0.0/16  192.172.226.128/27           4523        4434591
     128.182.64.0/18  192.172.226.128/27          12347        4374027
      209.249.0.0/16  192.172.226.128/27           2332        2926264
   209.133.123.90/32    224.2.172.238/32           3532        2365867
   128.197.160.29/32    224.2.177.155/32           3740        2310318
  128.223.215.225/32    224.2.176.111/32           1924        2218562
      209.132.0.0/17  192.172.226.128/27           2282        2096629
      128.174.0.0/16  192.172.226.128/27          11841        1307811
      216.200.0.0/16  192.172.226.128/27           1137        1220639
     204.71.192.0/20  192.172.226.128/27           1733        1206023
      63.224.64.0/19    192.172.226.0/26            826        1168984
     216.32.160.0/19   192.172.226.64/26            771         978894
    206.132.128.0/18  192.172.226.128/27           1004         939426
      192.215.0.0/16  192.172.226.128/27            844         926372
    192.203.230.0/24  192.172.226.128/27            326         904292
    216.184.128.0/17  192.172.226.128/27            680         856372
     192.92.126.0/24  192.172.226.128/27           6920         827154
    204.212.128.0/19  192.172.226.128/27           6863         815667
     216.121.96.0/19  192.172.226.128/27            543         791862
   205.253.57.105/32    224.2.177.155/32            783         771311
      209.133.0.0/17  192.172.226.128/27            773         725231
      163.178.0.0/16  192.172.226.128/27           5471         724441
   130.235.133.92/32    224.2.172.238/32            717         704138
   128.223.161.21/32     224.2.244.41/32            620         682022
    207.246.128.0/20  192.172.226.128/27            738         680724
        62.81.0.0/16  192.172.226.128/27           5493         665011
    130.240.60.42/32    224.2.172.238/32           1887         655525
     216.33.160.0/19  192.172.226.128/27            516         645244
       216.49.0.0/18  192.172.226.128/27            680         602690
  141.142.220.101/32    224.2.177.155/32            792         579586
      209.100.0.0/16    192.172.226.0/26            523         530971
     207.82.250.0/23  192.172.226.128/27            792         503945
      203.176.0.0/18  192.172.226.128/27           4477         463073
```

4.1.5 artsportms

usage: `artsportms [-v] [-p] [-i ifIndexList] [-l maxEntries] [-s srcPort] [-d dstPort] infile(s)`

`artsportms` displays a summary of each port matrix object in ARTS files. See the manpage (*artsportms(l)*) for more information.

Example output:

```
% artsportms arts.19990819
router:  192.172.226.1
ifIndex: 5 (Fddi0/1/0 198.17.46.56)
period:  08/19/1999 13:16:46 - 08/19/1999 13:23:10 EDT
  srcPort  dstPort           Pkts       Pkts/sec          Bytes       Bits/sec
  -------  -------  -------------  -------------  -------------  -------------
     1038    50562          16318        42.4948       21357457         444947
     3128     4104           2071        5.39323        3057562        63699.2
     3128     4023           1851        4.82031        2731545        56907.2
     2293    51482           3452        8.98958        2351575        48991.1
     1058    62268           1921         5.0026        2218278        46214.1
     1103    55524           2462        6.41146        2014681        41972.5
     3130     3130          12752        33.2083        1317970        27457.7
       80     3373            798        2.07812        1179520        24573.3
    30385      515            778        2.02604        1166636        24304.9
     3128     4125            217       0.565104         875645        18242.6
       80     3599            680        1.77083         856372        17841.1
       80     1495            836        2.17708         853562        17782.5
       80     2133            540        1.40625         791643        16492.6
       80     4089            785        2.04427         778601        16220.9
     1058    55524            783        2.03906         771311          16069
     1071    51482            717        1.86719         704138        14669.5
     1040    52382            614        1.59896         681386        14195.5
     8081     3130           6134         15.974         639488        13322.7
    38566    51482           1796        4.67708         637849        13288.5
       80     1664            538        1.40104         625192        13024.8
     3128     1088            167       0.434896         550598        11470.8
       80     1458            384              1         547682          11410
       80     1501            436        1.13542         513379        10695.4
     7213    55524            542        1.41146         502022        10458.8
       80     1204            415        1.08073         455316        9485.75
     3128     3983            375       0.976562         377234        7859.04
     3100    42616            250       0.651042         368798        7683.29
       80     2824            287       0.747396         349265        7276.35
       80     2488            268       0.697917         332975        6936.98
     8888     1974            320       0.833333         329596        6866.58
       80     4981            247       0.643229         323687        6743.48
     1057    28262            248       0.645833         321873        6705.69
     3128     3101            131       0.341146         300211         6254.4
     1616    55524            389        1.01302         287638        5992.46
       80     2568            208       0.541667         278268        5797.25
```

4.1.6 artsports

usage: `artsports [-v] [-n] [-p] [-t] [-i ifIndexList] [-l maxEntries] infile(s)`

`artsports` displays a summary of each port table object and selected port table object in ARTS files. See the manpage (*artsports(l)*) for more information.

Example output:

```
% artsports artsagg.19990819
router:  192.172.226.1
ifIndex: 5 (Fddi0/1/0 198.17.46.56)
period:  08/19/1999 13:16:46 - 08/19/1999 15:38:09 EDT
selected ports: 1-1024,6000,7070
         Port        InPkts       InBytes       OutPkts      OutBytes
        -----  ------------- ------------- ------------- -------------
            0       3842538    1558961934           183        135428
         http        108412       7767342       1156629    1213921483
      printer        103687     154890263             0             0
          ssh         36314       1861912        163627      54043036
       domain           741         47626         24727       6272662
         snmp         37196       3746994            93         15813
     ftp-data          7017        280892          1352       1991027
       svrloc             0             0          5153        831957
         6000          4513        753016             0             0
         smtp          1996        445836          2036        135266
       gopher             0             0           776        418897
          ftp          2805        161339          2290        188024
         pop3             0             0           668        347934
   netbios-ns             0             0          4230        329940
          ntp             0             0          2349        189996
         xfer             0             0            85         96798
           88             0             0           145         70664
          bgp           849         56001            79          3160
       sunrpc           434         58380             0             0
         1024            78         34979           117         17802
         nntp             0             0            60         41470
         auth           189          9250           312         15075
       bootps            74         24272             0             0
        https            10           976            84         23028
    hosts2-ns             0             0            33         21952
          ctf             0             0            17         18517
      nicname             0             0            20          9064
        dnsix             0             0            48          8920
   nameserver             0             0           149          7031
        shell             0             0            37          3718
        login             0             0            25          2992
      ocs_amu            12           756             0             0
     kerberos             4           432             0             0
         7070             0             0             4           339
  netbios-ssn             5           296             0             0
```

4.1.7 artsprotos

usage: `artsprotos [-v] [-n] [-p] [-i ifIndexList] infile(s)`

`artsprotos` is a utility that reads protocol table objects from ARTS files and displays a summary of their contents on stdout. See the manpage (*artsprotos(l)*) for more information.

Example output:

```
% artsprotos arts.19990819
router:  192.172.226.1
ifIndex: 1 (Ethernet0/0/0 192.172.226.1)
period:  08/19/1999 13:16:46 - 08/19/1999 13:23:10 EDT
Protocol          Pkts       Pkts/sec          Bytes       Bits/sec
--------  -------------  -------------  -------------  -------------
     udp         10155             26        7278343         151632
    icmp         34042             88        1776480          37010
     tcp          3344              8         361827           7538

router:  192.172.226.1
ifIndex: 5 (Fddi0/1/0 198.17.46.56)
period:  08/19/1999 13:16:46 - 08/19/1999 13:23:10 EDT
Protocol          Pkts       Pkts/sec          Bytes       Bits/sec
--------  -------------  -------------  -------------  -------------
     tcp        128780            335       62367956        1299332
     udp         66574            173       37321072         777522
    icmp         20913             54        1267518          26406
    igmp             3              0            612             12

router:  192.172.226.1
ifIndex: 7 (FastEthernet2/0/0 192.172.226.65)
period:  08/19/1999 13:16:46 - 08/19/1999 13:23:10 EDT
Protocol          Pkts       Pkts/sec          Bytes       Bits/sec
--------  -------------  -------------  -------------  -------------
     tcp         16977             44        2235522          46573
     udp           112              0          11541            240
    icmp             9              0            556             11

router:  192.172.226.1
ifIndex: 8 (Fddi2/1/0 192.172.226.129)
period:  08/19/1999 13:16:46 - 08/19/1999 13:23:10 EDT
Protocol          Pkts       Pkts/sec          Bytes       Bits/sec
--------  -------------  -------------  -------------  -------------
     tcp        137215            357       98659010        2055396
     udp         27301             71        2878374          59966
    icmp          1803              4         133854           2788
```

4.1.8 artstos

usage: `artstos [-v] [-n] [-p] [-i ifIndexList] infile(s)`

`artstos` is a utility that reads TOS table objects from ARTS files and displays a summary of their contents on stdout. See the manpage (*artstos(l)*) for more information.

Example output:

```
% artstos arts.19990819
router:  192.172.226.1
ifIndex: 5 (Fddi0/1/0 198.17.46.56)
period:  08/19/1999 13:23:13 - 08/19/1999 13:28:10 EDT
     TOS           Pkts       Pkts/sec          Bytes       Bits/sec
--------  -------------  -------------  -------------  -------------
       0         179070            602       83042852        2236844
       8           1751              5        1040714          28032
      16           4377             14         689586          18574
     192          10593             35         600186          16166
     160             79              0         108484           2922
     136              6              0            682             18
      41             12              0            656             17
     224              8              0            616             16
      96              3              0            164              4
```

4.1.9 artsintfms

usage: `artsintfms [-v] [-p] [-i ifIndexList] [-s srcIntfList] [-d dstIntfList] infile(s)`

`artsintfms` displays a summary of each interface matrix object in ARTS files. See the manpage (*artsintfms(l)*) for more information.

Example output:

```
% artsintfms -i 1,5 arts.19990819
router:  192.172.226.1
ifIndex: 1 (Ethernet0/0/0 192.172.226.1)
period:  08/19/1999 13:16:46 - 08/19/1999 13:23:10 EDT
  SrcIntf  DstIntf           Pkts       Pkts/sec          Bytes       Bits/sec
  -------  -------  -------------  -------------  -------------  -------------
        1        0           8135        21.1849        7076762         147433
        1        5          38898        101.297        2245081        46772.5
        1        8            508        1.32292          94807        1975.15

router:  192.172.226.1
ifIndex: 5 (Fddi0/1/0 198.17.46.56)
period:  08/19/1999 13:16:46 - 08/19/1999 13:23:10 EDT
  SrcIntf  DstIntf           Pkts       Pkts/sec          Bytes       Bits/sec
  -------  -------  -------------  -------------  -------------  -------------
        5        8         147173        383.263       60293921    1.25612e+06
        5        0          38888        101.271       34223634         712992
        5        1          23364        60.8438        3248003        67666.7
        5        7           6845        17.8255        3191600        66491.7
```

4.1.10 artsnexthops

usage: `artsnexthops [-v] [-p] [-i ifIndexList] [-l maxEntries] infile(s)`

`artsnexthops` is a utility that reads nexthop table objects from ARTS files and displays a summary of their contents on stdout. See the manpage (*artsnexthops(l)*) for more information.

Example output:

```
% artsnexthops arts.19990819
router:  192.172.226.1
ifIndex: 1 (Ethernet0/0/0 192.172.226.1)
period:  08/19/1999 13:16:46 - 08/19/1999 13:23:10 EDT
        NextHop            Pkts       Pkts/sec          Bytes       Bits/sec
  -------------   -------------  -------------  -------------  -------------
        0.0.0.0            8135             21        7076762         147432
    198.17.46.10          37602             97        2052755          42765
  198.17.46.205             702              1         145110           3023
192.172.226.145             495              1          91916           1914
   198.17.46.43             507              1          40304            839
  198.17.46.110              67              0           5442            113
192.172.226.147              11              0           2625             54
  198.17.46.112              10              0            564             11
    198.17.46.1               4              0            483             10
   198.17.46.33               6              0            423              8
192.172.226.130               2              0            266              5

router:  192.172.226.1
ifIndex: 5 (Fddi0/1/0 198.17.46.56)
period:  08/19/1999 13:16:46 - 08/19/1999 13:23:10 EDT
        NextHop            Pkts       Pkts/sec          Bytes       Bits/sec
  -------------   -------------  -------------  -------------  -------------
192.172.226.145          123285            321       53931898        1123581
        0.0.0.0           38886            101       34222814         712975
192.172.226.146           23865             62        6359117         132481
 192.172.226.62             812              2        1168086          24335
192.172.226.110             918              2        1160614          24179
 192.172.226.24           18843             49        1109530          23115
 192.172.226.82             913              2         995412          20737
 192.172.226.35            1612              4         736684          15347
192.172.226.112            1780              4         500747          10432
 192.172.226.99             203              0         250952           5228
 192.172.226.30            1993              5         211179           4399
 192.172.226.89            2405              6         150416           3133
192.172.226.107             125              0          87562           1824
192.172.226.122             286              0          21316            444
192.172.226.113             174              0          20025            417
 192.172.226.16              68              0          19684            410
192.172.226.147              23              0           3558             74
192.172.226.104              21              0           2268             47
 192.172.226.44              16              0           1400             29
192.172.226.105               9              0           1384             28
```

4.2 Aggregation Utilities

It is often useful to reduce the time domain granularity of ARTS data, to produce summary reports of traffic over long time intervals. The `arts++` package contains a handful of utilities to aggregate some types of ARTS data in the time domain. While these are primarily examples of using features in the class library, they are often useful utilities in their own right.

4.2.1 artsagg

usage: `artsagg [-v] [-i ifIndexList] [-I ifIndexList] [-s portlist] [-T objectTypes] [-h hours] [-o] [-q] outfile infile(s)`

`artsagg` reads AS matrix, net matrix, port matrix, protocol table, TOS table, interface matrix and IP nexthop table objects from ARTS files (`infile(s)`) and aggregates them in the time domain, placing the resulting object(s) into an output file (`outfile`). See the manpage (*artsagg(l)*) for more information.

4.2.2 artsasagg

usage: `artsasagg [-v] [-i ifIndexList] [-I ifIndexList] [-h hours] [-o] [-q] outfile infile(s)`

`artsasagg` reads AS matrix objects from ARTS files (`infile(s)`) and aggregates them in the time domain, placing the resulting AS matrix object(s) into an output file (`outfile`). See the manpage (*artsasagg(l)*) for more information.

4.2.3 artsnetagg

usage: `artsnetagg [-v] [-i ifIndexList] [-I ifIndexList] [-h hours] [-o] [-q] outfile infile(s)`

`artsnetagg` reads net matrix objects from ARTS files (`infile(s)`) and aggregates them in the time domain, placing the resulting net matrix object(s) into an output file (`outfile`). See the manpage (*artsnetagg(l)*) for more information.

4.2.4 artsportmagg

usage: `artsportmagg [-v] [-i ifIndexList] [-I ifIndexList] [-s portlist] [-h hours] [-o] [-q] outfile infile(s)`

`artsportmagg` reads port matrix objects from ARTS files (`infile(s)`) and aggregates them in the time domain, placing the resulting port matrix object(s) or selected port table objects into an output file (`outfile`). See the manpage (*artsportmagg(l)*) for more information.

4.2.5 artsprotoagg

usage: `artsprotoagg [-v] [-i ifIndexList] [-I ifIndexList] [-h hours] [-o] [-q] outfile infile(s)`

`artsprotoagg` reads protocol table objects from ARTS files (`infile(s)`) and aggregates them in the time domain, placing the resulting protocol table object(s) into an output file (`outfile`). See the manpage (*artsprotoagg(l)*) for more information.

4.2.6 artsintfmagg

usage: `artsintfmagg [-v] [-i ifIndexList] [-I ifIndexList] [-h hours] [-o] [-q] outfile infile(s)`

`artsintfmagg` reads interface matrix objects from ARTS files (`infile(s)`) and aggregates them in the time domain, placing the resulting interface matrix object(s) into an output file (`outfile`). See the manpage (*artsintfmagg(l)*) for more information.

4.2.7 artsnexthopagg

usage: `artsnexthopagg [-v] [-i ifIndexList] [-I ifIndexList] [-q] [-o] [-h hours] outfile infile(s)`

`artsnexthopagg` reads nexthop table objects from ARTS files (`infile(s)`) and aggregates them in the time domain, placing the resulting nexthop table object(s) into an output file (`outfile`). See the manpage (*artsnexthopagg(l)*) for more information.