arts++

Daniel McRobb, CAIDA Aug 1998

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)
- interface matrix (version 0)
- nexthop table (version 0)
- forward IP path and RTT (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 (an `ArtsAttribute` with an `Identifier()` equal to `artsC_ATTR_PERIOD`) which represents the time interval in which the traffic was seen. It is also stored with a host attribute (an `ArtsAttribute` with an `Identifier()` equal to `artsC_ATTR_HOST`) indicating the router from which the data was collected. Finally, it may also have in interface index attribute (an `ArtsAttribute` with an `Identifier()` equal to `artsC_ATTR_IFINDEX`) indicating the input interface on which the data was seen.

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 (an `ArtsAttribute` with an `Identifier()` equal to `artsC_ATTR_PERIOD`) which represents the time interval in which the traffic was seen. It is also stored with a host attribute (an `ArtsAttribute` with an `Identifier()` equal to `artsC_ATTR_HOST`) indicating the router from which the data was collected. Finally, it may also have in interface index attribute (an `ArtsAttribute` with an `Identifier()` equal to `artsC_ATTR_IFINDEX`) indicating the input interface on which the data was seen.

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 (an `ArtsAttribute` with an `Identifier()` equal to `artsC_ATTR_PERIOD`) which represents the time interval in which the traffic was seen. It is also stored with a host attribute (an `ArtsAttribute` with an `Identifier()` equal to `artsC_ATTR_HOST`) indicating the router from which the data was collected. Finally, it may also have in interface index attribute (an `ArtsAttribute` with an `Identifier()` equal to `artsC_ATTR_IFINDEX`) 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 (an `ArtsAttribute` with an `Identifier()` equal to `artsC_ATTR_PERIOD`) which represents the time interval in which the traffic was seen. It is also stored with a host attribute (an `ArtsAttribute` with an `Identifier()` equal to `artsC_ATTR_HOST`) indicating the router from which the data was collected. Finally, it may also have in interface index attribute (an `ArtsAttribute` with an `Identifier()` equal to `artsC_ATTR_IFINDEX`) indicating the input interface on which the data was seen.

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 (an `ArtsAttribute` with an `Identifier()` equal to `artsC_ATTR_PERIOD`) which represents the time interval in which the traffic was seen. It is also stored with a host attribute (an `ArtsAttribute` with an `Identifier()` equal to `artsC_ATTR_HOST`) indicating the router from which the data was collected. Finally, it may also have in interface index attribute (an `ArtsAttribute` with an `Identifier()` equal to `artsC_ATTR_IFINDEX`) indicating the input interface on which the data was seen.

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 (an `ArtsAttribute` with an `Identifier()` equal to `artsC_ATTR_PERIOD`) which represents the time interval in which the traffic was seen. It is also stored with a host attribute (an `ArtsAttribute` with an `Identifier()` equal to `artsC_ATTR_HOST`) indicating the router from which the data was collected. Finally, it may also have in interface index attribute (an `ArtsAttribute` with an `Identifier()` equal to `artsC_ATTR_IFINDEX`) indicating the input interface on which the data was seen.

2.7 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 (an `ArtsAttribute` with an `Identifier()` equal to `artsC_ATTR_PERIOD`) which represents the time interval in which the traffic was seen. It is also stored with a host attribute (an `ArtsAttribute` with an `Identifier()` equal to `artsC_ATTR_HOST`) indicating the router from which the data was collected. Finally, it may also have in interface index attribute (an `ArtsAttribute` with an `Identifier()` equal to `artsC_ATTR_IFINDEX`) indicating the input interface on which the data was seen (this is redundant but matches the current indentification scheme used by other classes used for `cflowd` data).

2.8 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 (an `ArtsAttribute` with an `Identifier()` equal to `artsC_ATTR_PERIOD`) which represents the time interval in which the traffic was seen. It is also stored with a host attribute (an `ArtsAttribute` with an `Identifier()` equal to `artsC_ATTR_HOST`) indicating the router from which the data was collected. Finally, it may also have in interface index attribute (an `ArtsAttribute` with an `Identifier()` equal to `artsC_ATTR_IFINDEX`) indicating the input interface on which the data was seen.

2.9 forward IP path and RTT (version 0)

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.

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_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.

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 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.

3.1.10 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.

3.1.11 The `ArtsIpPathData` Class

Not yet documented, used by 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 ArtsInterfaceMatrix

3.2.8 ArtsNextHopTable

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 `ArtsInterfaceMatrixAggregator` Class

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

3.3.7 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 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 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 [-p] [-s srcAS] [-d dstAS] infile(s)`

`artsases` displays a summary of each AS matrix object in ARTS files. Example output:

```
% artsases arts.19980817
router: 192.172.226.1
period: 08/17/1998 05:30:44 - 08/17/1998 05:37:42 EDT
  Src AS  Dst AS           Pkts       Pkts/sec          Bytes       Bits/sec
  ------  ------  -------------  -------------  -------------  -------------
     195     195          70784         169.34       18301728         350272
      87     195          55445        132.644       12702754         243115
       3     195          53667         128.39        5130268        98186.9
    1224     195           8764        20.9665        4746527        90842.6
    3967     195           4514         10.799        3230531        61828.3
    7220     195           2073        4.95933        3066911        58696.9
```

```
 195    234         2151       5.14593      3056251      58492.8
7050    195         3546       8.48325      2382911        45606
  33    195         3616       8.65072      2315453      44314.9
 297    195         1613       3.85885      2197017      42048.2
 194    195        10483       25.0789      2124627      40662.7
7224    195         1674       4.00478      1979142      37878.3
```

4.1.4 artsnets

usage: `artsnets [-p] [-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.19980817
router: 192.172.226.1
period: 08/17/1998 05:38:00 - 08/17/1998 05:48:03 EDT
         Src Network        Dst Network            Pkts          Bytes
  ------------------ ------------------ ------------- -------------
    205.189.33.75/32     224.2.202.41/32          77168       62334623
      130.149.0.0/16     224.2.231.31/32          30179       24064955
       164.58.0.0/16    224.2.131.215/32          18146        4852665
   128.223.83.204/32     224.2.246.13/32          12265        4397400
   131.188.34.134/32    224.2.172.238/32           2568        2047808
       164.58.0.0/16    224.2.131.215/32          17987        1913904
   133.82.241.137/32    224.2.172.238/32           6485        1683392
      204.123.0.0/16  192.172.226.146/32            959        1427774
        209.1.0.0/16  192.172.226.145/32           1231        1362956
    205.216.162.0/23  192.172.226.145/32            828        1162127
    129.89.143.30/32    224.2.172.238/32           3633        1019726
      206.204.0.0/16  192.172.226.145/32            687        1016236
   134.102.218.45/32         227.6.5.7/32            771         818813
    192.76.157.71/32    224.2.204.172/32            781         810713
    192.76.157.71/32    224.2.204.172/32            676         698131
    192.76.157.71/32    224.2.204.172/32            641         662238
        209.1.0.0/16  192.172.226.128/25            459         483147
      203.8.105.0/24  192.172.226.128/25           4450         482959
    192.172.226.0/25     204.212.44.0/22            580         460780
    192.76.157.71/32    224.2.204.172/32            440         447540
   193.174.75.166/32         227.6.5.7/32            374         401422
     192.150.12.0/24  192.172.226.145/32            168         367235
    204.152.184.0/21    192.172.226.0/25            223         327102
```

4.1.5 artsportms

usage: `artsportms [-p] [-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.19981005
router:  192.172.226.1
ifIndex: 1
period:  10/04/1998 23:55:06 - 10/05/1998 23:56:09 CUT
  srcPort  dstPort           Pkts       Pkts/sec          Bytes       Bits/sec
  -------  -------  -------------  -------------  -------------  -------------
       22    47164        1564950        17.3762     2133171600         189483
       22    47762         897360        9.96369     1343973900         119381
     1029    10000        1041240        11.5612     1070394720        95079.6
       22    47591         378600        4.20372      567690210        50426.1
       22    47365          31536       0.350155       47102464        4183.96
     6000    22041         516041        5.72978       43601192        3872.95
      979       22         461500        5.12419       39132700        3476.03
       80    21777          28500       0.316445       37517700        3332.57
       80     1193          28080       0.311782       37169280        3301.62
       80     1799          29040       0.322441       36595200        3250.63
       22    47169          26340       0.292462       36525240        3244.42
       80     1137          25020       0.277806       34630200        3076.09
    55678       53         483652        5.37015       30862702        2741.43
       80     2006          22320       0.247827       27806760        2469.98
```

4.1.6 artsports

usage: `artsports [-p] [-t] 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.19981005
router:  192.172.226.1
ifIndex: 1
period:  10/04/1998 19:55:06 - 10/05/1998 19:56:09 EDT
selected ports: 1-2048,6000,7070
   Port         InPkts        InBytes        OutPkts       OutBytes
  -----  -------------  -------------  -------------  -------------
     22        1200549       88829504        3200619     4163947490
   1029              0              0        1041240     1070394720
     80        1260232      106367970         848827      765735423
     53         501944       32169952         147542       30033536
   6000              0              0         516041       43601192
      0         527376       43382817              0              0
     20         874012       34961888              0              0
    513         177475        7224754              0              0
     25          25514        3993236          41671        2807293
    139         156654        6690843              0              0
    515           2457        3002610              0              0
    137              0              0          20257        2386046
```

```
     110        39438      1778969            0            0
    1017            0            0        20558      1502200
     443        14730      1452375            0            0
      23        21810       887940            0            0
     123            0            0        11428       868528
     113         9224       394818         2103        84120
     138            0            0         1089       254810
     750            0            0         1238       224052
    1772         2400       167040            0            0
    1025            0            0         1680       113040
    1033            0            0          600        82320
    1822          900        62640            0            0
```

4.1.7 artsprotos

usage: `artsprotos [-p] 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.19981005
router:  192.172.226.1
ifIndex: 1
period:  10/04/1998 23:55:06 - 10/05/1998 23:56:09 CUT
Protocol          Pkts       Pkts/sec          Bytes       Bits/sec
--------  -------------  -------------  -------------  -------------
       6        8466492             94     5237854239         465261
      17        2207792             24     1175888232         104450
       1        5785338             64      324733568          28845

router:  192.172.226.1
ifIndex: 8
period:  10/04/1998 23:55:06 - 10/05/1998 23:56:09 CUT
Protocol          Pkts       Pkts/sec          Bytes       Bits/sec
--------  -------------  -------------  -------------  -------------
       6      154258552           1712   120841663280       10733967
      17       19487284            216     1921468060         170677
       1        3662818             40      222174954          19735
```

4.1.8 artsintfms

`artsintfms` displays a summary of each interface matrix object in ARTS files. Example output:

```
% artsintfms arts.19981207
router:  204.212.46.1
ifIndex: 4
period:  12/06/1998 18:58:37 - 12/06/1998 19:03:37 EST
  SrcIntf  DstIntf           Pkts       Pkts/sec          Bytes       Bits/sec
  -------  -------  -------------  -------------  -------------  -------------
```

```
        4        5         108272        360.907        5694529         151854
        4        0            157       0.523333          11932        318.187

router:  204.212.46.1
ifIndex: 5
period:  12/06/1998 18:58:37 - 12/06/1998 19:03:37 EST
  SrcIntf  DstIntf           Pkts       Pkts/sec          Bytes       Bits/sec
  -------  -------  -------------  -------------  -------------  -------------
        5        4          36845        122.817        2191172        58431.3
        5        0             11      0.0366667            616        16.4267
```

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

4.1.9 artsnexthops

`artsnexthops` is a utility that reads nexthop table objects from ARTS files and displays a summary of their contents on stdout. Example ouptut:

```
% artsnexthops arts.19981207
router:  204.212.46.1
ifIndex: 4
period:  12/06/1998 18:58:37 - 12/06/1998 19:03:37 EST
        NextHop           Pkts       Pkts/sec          Bytes       Bits/sec
  -------------  -------------  -------------  -------------  -------------
        0.0.0.0         108429            361        5706461         152172

router:  204.212.46.1
ifIndex: 5
period:  12/06/1998 18:58:37 - 12/06/1998 19:03:37 EST
        NextHop           Pkts       Pkts/sec          Bytes       Bits/sec
  -------------  -------------  -------------  -------------  -------------
   204.212.46.2          35918            119        2008940          53571
   204.212.46.6            617              2         154368           4116
   204.212.46.3            310              1          27864            743
        0.0.0.0             11              0            616             16
```

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

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 artsasagg

usage: `artsasagg [-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.

4.2.2 artsnetagg

usage: `artsnetagg [-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.

4.2.3 artsportmagg

usage: `artsportmagg [-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.

4.2.4 artsprotoagg

usage: `artsprotoagg [-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.

4.2.5 artsintfmagg

usage: `artsintfmagg [-v] [-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.

4.2.6 artsnexthopagg

usage: `artsnexthopagg [-v] [-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.