cflowd configuration

Daniel W. McRobb Original: Oct 1998 Last Modified: August 30, 1999

This is the `cflowd` system configuration guide for cflowd-2-1-a2.

Contents

List of Figures

1 Overview

1.1 Data flow

It is useful to understand the flow of data in the `cflowd` system before configuration. **Figure 1** is a diagram showing a high-level view of the flow of data in the cflowd system.

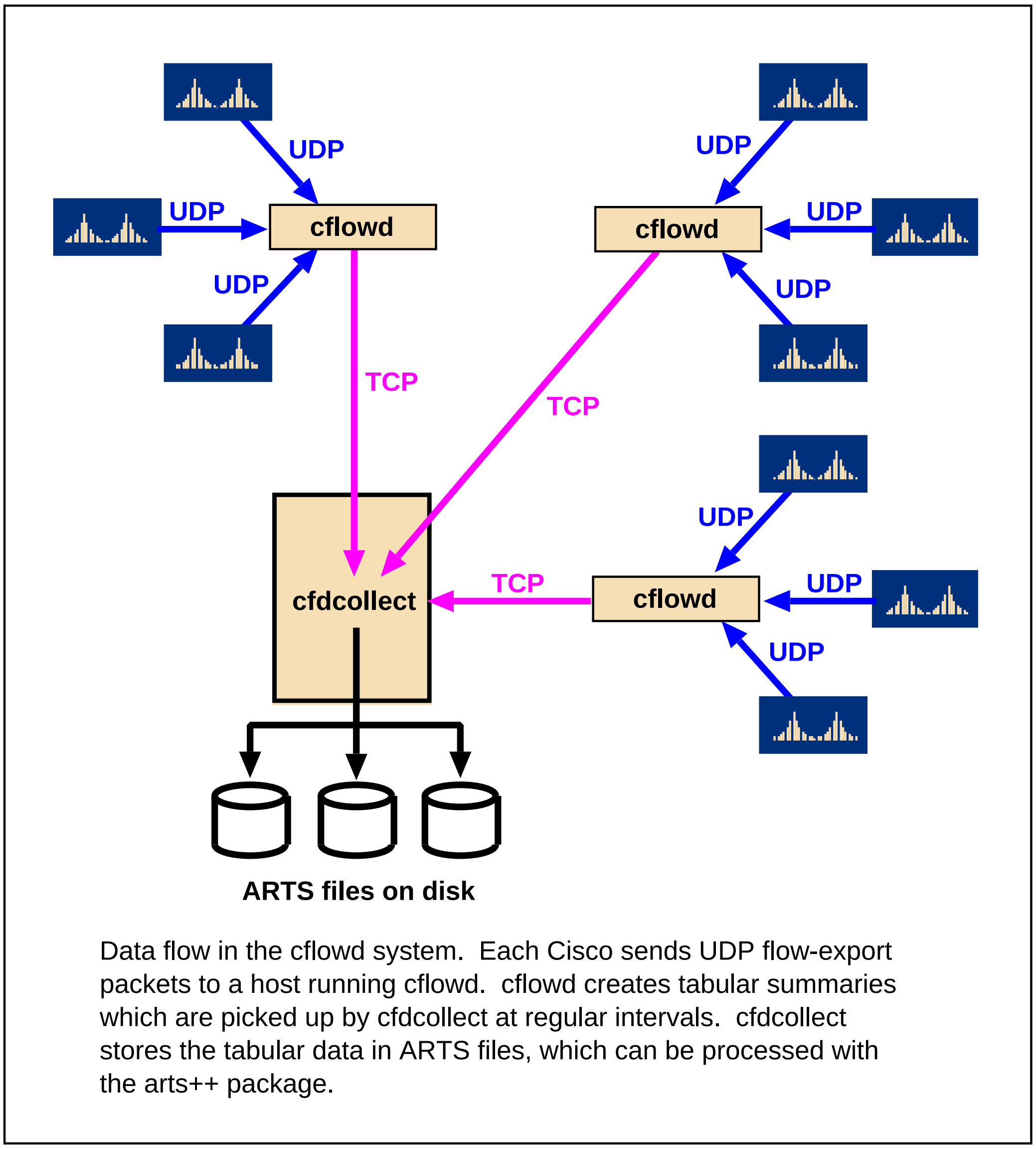

Figure 1: cflowd data flow

Each Cisco router sends flow-export packets (version 1, version 5 or version 8) to a host running `cflowdmux` and `cflowd`. `cflowd` creates tabular data from the data in the flow-export packets. `cflowd` also serves the tabular data to `cfdcollect`. `cfdcollect` will contact each configured instance of `cflowd` at regular intervals (configurable) to retrieve tabular data, and will store the data in ARTS files.

A typical configuration inside a provider network would have several workstations in the network running `cflowd`, each located in close proximity to the routers from which they're receiving flow-export data. A single instance of `cfdcollect` would be run on a centrally located server with plenty of disk space.

It should be noted that `cflowd` does not receive flow-export packets directly. A program called `cflowdmux` is responsible for handling UDP packets from the Cisco routers, and will put the packets in shared memory buffers which can be read by `cflowd`. In addition, `cflowd` acts a server to local table clients like `cfdases`, as well as acting as the server for `cfdcollect`. A more detailed diagram showing the data flow in the system is seen in **Figure 2**.

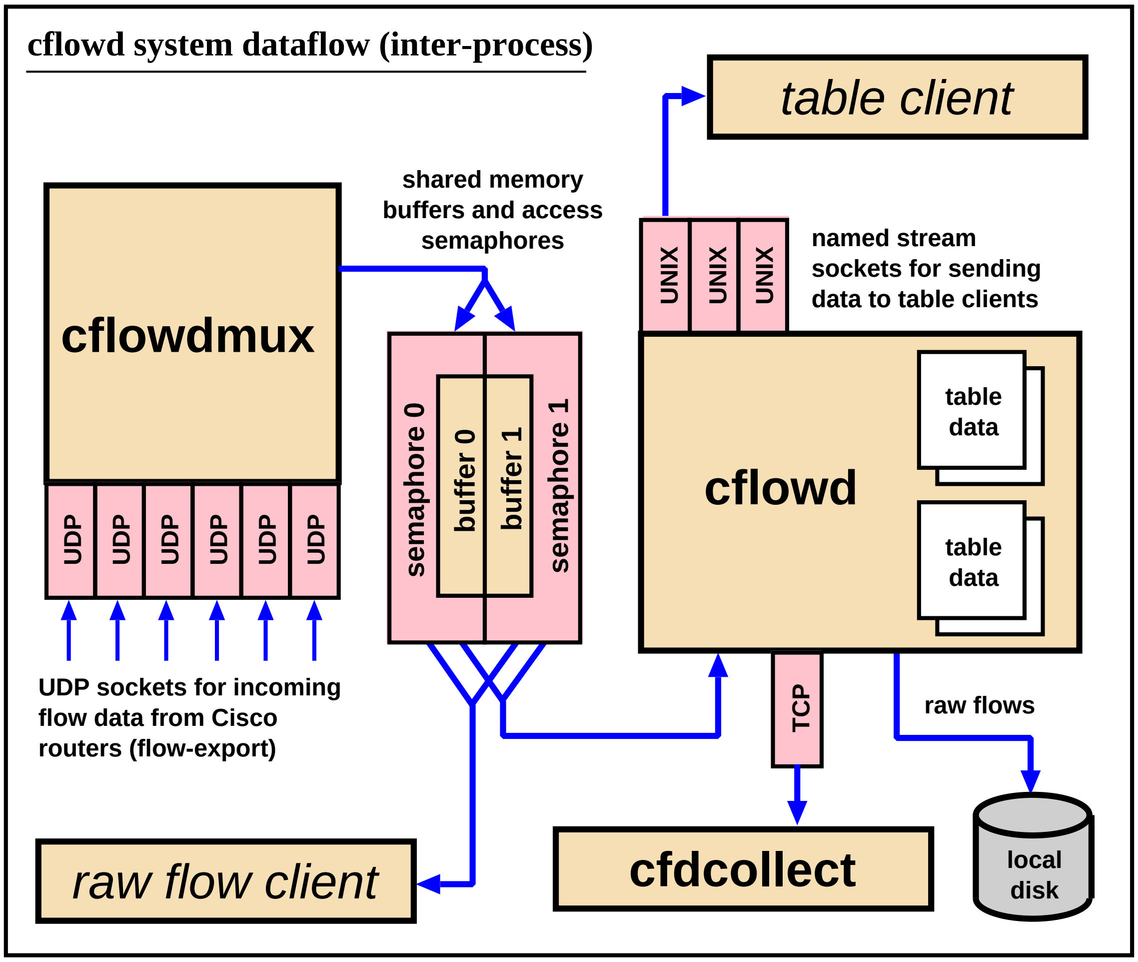

Figure 2: cflowd data flow detail

2 cflowd configuration

`cflowd`, `cflowdmux` and the local utilities (`cfdases`, `cfdnets`, et. al.) all read configuration information from `cflowd.conf`. In a standard installation, `cflowd.conf` will be located in the `/usr/local/arts/etc/` directory.

`cflowd.conf` contains three types of stanzas: an OPTIONS stanza specifying system-wide configuration values, CISCOEXPORTER stanzas specifying configuration values for each Cisco from which we're collecting data, and COLLECTOR stanzas specifying hosts from which we permit `cfdcollect` connections.

2.1 OPTIONS stanza

The OPTIONS stanza in `cflowd.conf` is used to set system-wide configuration values for `cflowd`, `cflowdmux` and local clients. There should be a single OPTIONS stanza in `cflowd.conf`, and it should be the first stanza. Following are descriptions of each of the settings in an OPTIONS stanza.

2.1.1 LOGFACILITY *(optional)*

The LOGFACILITY setting is used to set the syslog facility that will be used by `cflowd` and `cflowdmux` for logging. If unspecified, `local6` will be used.

2.1.2 TCPCOLLECTPORT *(optional)*

The TCPCOLLECTPORT setting is used to set the TCP port on which cflowd will listen for connections from `cfdcollect`. Connections to this port are used for downloading tabular data from `cflowd` and cause the tabular data in `cflowd` to be cleared after transmission. Only hosts with a COLLECTOR stanza are permitted to connect to this port and retrieve data. Normally, a single `cfdcollect` will be running on a COLLECTOR host and be the only program to connect to the TCPCOLLECTPORT.

If TCPCOLLECTPORT is unspecified, a default value of `2056` will be used.

2.1.3 PKTBUFSIZE *(optional)*

`cflowdmux` uses a 1 megabyte packet buffer by default, split into 2 toggle buffers in a single shared memory segment. Using the PKTBUFSIZE setting, you may change the default size of the packet buffer shared memory segment. This option is typically used to increase the size of the packet buffer.

2.1.4 TABLESOCKFILE

The TABLESOCKFILE setting specifies the path to the named stream socket on which `cflowd` will listen for local table client connections. `cflowd` will accept connections from table clients on this socket. Typical table clients are `cfdases`, `cfdnets`, et. al.

2.1.5 FLOWDIR

Specifies the directory in which raw flow files should be stored. This is used when `flows` is specified in a CISCOEXPORTER stanza.

2.1.6 FLOWFILELEN *(optional)*

Specifies the length of raw flow files. `cflowd` will roll over a raw flow file when it reaches this length. `cflowd` will not truncate a raw flow in a flow file, so it typically writes to some length just below the FLOWFILELEN. If unspecified, a default value of `1048576` (1 megabyte) will be used.

2.1.7 NUMFLOWFILES *(optional)*

Specifies the number of raw flow files to be user per router. This determines how many raw flow files will be kept by `cflowd` per router. If unspecified, a default value of `10` will be used.

2.1.8 MINLOGMISSED *(optional)*

Specifies the threshold at which `cflowd` will syslog a message about missed flows. `cflowd` only does this when contacted by `cfdcollect`. If this value is unspecified, a default value of 300 will be used.

2.1.9 Example

Below is an example OPTIONS stanza. We've specified `local6` as the LOGFACILITY, so `cflowd` and `cflowdmux` will syslog using this facility. We've specified a TCPCOLLECTPORT of 2056, so `cflowd` will listen for connections from `cfdcollect` on TCP port 2056. `cflowdmux` will listen for raw flow client connections on the named stream socket `/usr/local/arts/etc/cflowdmux.socket` as specified with RAWFLOWSOCKFILE. `cflowd` will listen for connections from table clients on the named stream socket `/usr/local/arts/etc/cflowdtable.socket` as specified with TABLESOCKFILE. `cflowd` will store raw flow files in the `/usr/local/arts/data/cflowd` directory if `flows` are specified in the COLLECT portion of a CISCOEXPORTER stanza. As specified by FLOWFILELEN, `cflowd` will roll over a raw flow file when it reaches `1000000` bytes. `10` flow files will be kept per router, as specified with NUMFLOWFILES. As specified by MINLOGMISSED, `cflowd` will syslog a message about missed flows when there are more than `300` flows missed between queries from `cfdcollect`.

```
OPTIONS {
  LOGFACILITY:          local6
  TCPCOLLECTPORT:       2056
  TABLESOCKFILE:        /usr/local/arts/etc/cflowdtable.socket
  FLOWDIR:              /usr/local/arts/data/cflowd
  FLOWFILELEN:          1000000
  NUMFLOWFILES:         10
  MINLOGMISSED:         300
}
```

2.2 CISCOEXPORTER stanza

The CISCOEXPORTER stanza is used to specify configuration values for a single Cisco router. There may be more than one CISCOEXPORTER stanza in `cflowd.conf`, with each corresponding to a Cisco router from which we would like to collect data.

2.2.1 HOST

The HOST setting is used to specify the IP address of the Cisco.

2.2.2 ADDRESSES

The ADDRESSES setting is used to specify the IP addresses of interfaces on the Cisco router. It's possible for flow-export packets to originate from more than one interface on a Cisco router; the ADDRESSES setting lets us specify multiple source addresses from which we'll accept flow-export data for a single Cisco router.

2.2.3 CFDATAPORT

The CFDATAPORT setting is used to specify the UDP port on which `cflowdmux` should listen for flow-export packets from the Cisco router. This should match whatever you've configured as the flow-export destination port on the Cisco router.

2.2.4 LOCALAS *(optional)*

The LOCALAS setting is used to specify the local AS of the Cisco router. This value is used when trying to fix 0 values in the source and destination AS fields in flows from version 5 flow-export and in the prefix aggregation flows in version 8 flow-export. If LOCALAS is unspecified, `cflowd` will not try to fix 0 values in the source and destination AS fields.

2.2.5 SNMPCOMM *(optional but recommended)*

SNMPCOMM is used to specify the SNMP (v1) community name to be used when retrieving interface descriptions and IP addresses from the router (ifDescr and ipAdEntIfIndex). It should be enclosed in single quotes. Currently we can't handle a community name that contains a single quote, but this will be fixed soon.

2.2.6 COLLECT

The COLLECT setting is used to specify the types of data that should be collected from the flow-export data for the Cisco router. There are several types of data which can be collected:

- `asmatrix` - AS matrix (packets and bytes from source ASes to destination ASes)
- `netmatrix` - net matrix (packets and bytes from source networks to destination networks)
- `portmatrix` - port matrix (packets and bytes from source ports to destination ports)
- `ifmatrix` - interface matrix (packets and bytes from input interfaces to output interfaces, by ifIndex)
- `protocol` - protocol table (packets and bytes per IP protocol)
- `nexthop` - IP nexthop table (packets and bytes per IP nexthop)
- `tos` - TOS table (packets and bytes per IP TOS)
- `flows` - raw flow data

2.2.7 Example

An example CISCOEXPORTER is shown below. It is for a router with an IP address of 204.212.46.1 (the HOST) that is configured to flow-export to port 2055 (the CFDATAPORT) on the host running `cflowd`. We specified two addresses for the Cisco router: 204.212.46.1 and 204.212.45.14. These correspond to the IP addresses of different interfaces on the Cisco router. We specified a LOCALAS of 195, corresponding to the local AS of the Cisco router. We used COLLECT to list the types of data which `cflowd` will collect for the Cisco router using the flow-export data from the router.

```
CISCOEXPORTER {
  HOST:         204.212.46.1            #  IP address of Cisco sending data.
  ADDRESSES:    { 204.212.46.1,         #  Addresses of interfaces on Cisco
                  204.212.45.14 }       #    sending data.
  CFDATAPORT:   2055                    #  Port on which to listen for data.
  SNMPCOMM:     'public'
  LOCALAS:      195                     #  Local AS of Cisco sending data.
  COLLECT:      { protocol, ifmatrix, portmatrix, netmatrix,
```

```
                  nexthop, asmatrix, tos, flows }
}
```

2.3 COLLECTOR stanza

The COLLECTOR stanza is used to hold configuration values for a host running `cfdcollect`. In a standard configuration, there will only be one or two of these, since a single `cfdcollect` host is normally used to collect data from all instances of `cflowd`. However, there may be more than one entry (for example, you may have a hot backup host on which you will run `cfdcollect` when the primary `cfdcollect` host is down).

2.3.1 HOST

The HOST setting is used to specify the IP address of the host running `cfdcollect`.

2.3.2 ADDRESSES

The ADDRESSES setting is used to specify the IP addresses of the host running `cfdcollect`. `cflowd` will permit connections from `cfdcollect` originating from any of the IP addresses in the list.

2.3.3 AUTH

Currently unused.

2.3.4 Example

Below is an example COLLECTOR stanza. This says that we will allow connections from `cfdcollect` that come from 195.83.243.2 or 195.83.241.9, and we will assume that connections from either of these addresses are from the same host.

```
COLLECTOR {
  HOST:          195.83.243.2    # IP address of host running cfdcollect
  ADDRESSES:     { 195.83.243.2, 195.83.241.9 }  # other addresses of host
  AUTH:          none
}
```

3 starting cflowd

3.1 starting cflowdmux

`cflowdmux` should be started before `cflowd`. It can be started with no arguments, in which case it will use a compiled-in default as the configuration file name (typically `/usr/local/arts/etc/cflowd.conf`). It will also accept an explicit configuration file name as the first argument. Examples:

```
% cflowdmux
```

would start `cflowdmux` using the compiled-in default configuration file.

```
% cflowdmux /etc/cflowd.conf
```

would start `cflowdmux` using `/etc/cflowd.conf` as the configuration file.

3.2 starting cflowd

After starting `cflowdmux`, you should start `cflowd`. Like `cflowdmux`, `cflowd` will use a compiled-in default configuration file (typically `/usr/local/arts/etc/cflowd.conf`) if given no arguments. It will also accept an explicit configuration file name as the first argument. Examples:

```
% cflowd
```

would start `cflowd` using the compiled-in default configuration file.

```
% cflowd /etc/cflowd.conf
```

would start `cflowd` using `/etc/cflowd.conf` as the configuration file.

4 cflowd logging

`cflowdmux` and `cflowd` use *syslog(3)* to record errors and runtime status information. As a convention, each syslog message starts with a capitalized letter in brackets indicating the priority of the message:

code	*priority*	*description*
`[A]`	`LOG_ALERT`	fatal errors
`[C]`	`LOG_CRIT`	errors that need attention
`[E]`	`LOG_ERR`	error conditions
`[I]`	`LOG_INFO`	informational messages

5 cflowd debugging

There are a number of things to examine when trying to find a problem with `cflowdmux` or `cflowd`.

The first is the syslog messages. In general, error messages will indicate a problem (and should always be included in requests for assistance, whether directed at the author or the mailing list).

The second is the socket status. On most UNIX systems, the output of `netstat -an` will show you open sockets. You should look for open sockets on UDP ports for each unique CFDATAPORT in `cflowd.conf`. If you do not find an open socket for one of the CFDATAPORT ports, there is a serious problem (and you should report it to the author). If you see a large receive queue value (on those systems that report the queue length) for prolonged periods on any of the CFDATAPORT ports, there is a serious problem (which should be reported to the author). Again using `netstat -an`, you should look for an open TCP port in the LISTEN state for the TCPCOLLECTPORT specified in the OPTIONS stanza of `cflowd.conf`. The lack of a socket in this state will prevent `cfdcollect` from working, and is a serious problem which should be reported to the author.

A third source of information is the status of the shared memory segment and the semaphore set. These can be seen with `ipcs -a` on most UNIX systems. You should see a shared memory segment owned by the user running `cflowdmux`, and at least 2 processes should be attached (`cflowdmux` and `cflowd`). The shared memory segment should be approximately 2 megabytes in length. You should also see a semaphore set owned by the user running `cflowdmux`. The size of the semaphore set should be 2.

A final handy source of information is the output of *lsof(8)* if you have it on your system. In particular, the output of `lsof -p PID` where PID is the process ID of `cflowdmux` or `cflowd`.

6 cfdcollect configuration

`cfdcollect` uses a simple configuration file, typically named `cfdcollect.conf` and located in the `/usr/local/arts/etc/` directory. `cfdcollect.conf` should contain two types of stanzas: a single 'system' stanza specifying system-wide values for `cfdcollect`, and one or more 'cflowd' stanzas (one for each instance of `cflowd`).

6.1 system stanza

The system stanza is used to set system-wide values for `cfdcollect`. There should be only one of these in `cfdcollect.conf`, and it should be the first stanza. Following are descriptions of each of the variables in the system stanza.

6.1.1 logFacility

Sets the syslog facility to be used for logging by `cfdcollect`.

6.1.2 dataDirectory

Sets the top-level directory in which to store ARTS data for each router. cfdcollect actually writes into subdirectories under dataDirectory for each router, with the subdirectories named by the IP address of each router (corresponding to the HOST setting for routers in `cflowd.conf`). For example, if you set `dataDirectory` to `/usr/local/arts/cflowd/data`, the data for router 204.212.46.1 would wind up in the `/usr/local/arts/cflowd/data/204.212.46.1` directory.

6.1.3 filePrefix

Sets the file prefix to be used when writing ARTS data to files. `cfdcollect` uses filenames of the form *filePrefix.YYYYMMDD* to store ARTS data. Ther normal `filePrefix` setting is 'arts'. Continuing with the example from the last section, data for router 204.212.46.1 for September 23, 1998 would wind up in `/usr/local/arts/cflowd/data/204.212.46.1/arts.19980923`

6.1.4 pidFile

Specifies the full path to the file in which `cfdcollect` should store its process ID.

6.1.5 Example

An example system stanza is shown below. It tells `cfdcollect` to syslog with the local6 facility, place data in the `/usr/local/arts/data/cflowd` directory, use 'arts' as the file prefix when creating data files, and to store its process ID in `/usr/local/arts/etc/cfdcollect.pid`.

```
system {
  logFacility:          local6         #  Syslog to local6 facility.
  dataDirectory:        /usr/local/arts/data/cflowd
  filePrefix:           arts
  pidFile:              /usr/local/arts/etc/cfdcollect.pid
}
```

6.2 cflowd stanza

The cflowd stanza is used to set configuration variables for an instance of `cflowd` from which we want `cfdcollect` to retrieve data. Following are the descriptions of each of the variables in the cflowd stanza.

6.2.1 host

The host running `cflowd`. This may be a hostname or an IP address.

6.2.2 tcpCollectPort

Sets the port to which to connect to on the host running `cflowd`. This is the TCP port on which `cflowd` is listening, corresponding to the TCPCOLLECTPORT setting in the OPTIONS stanza of `cflowd.conf`.

6.2.3 minPollInterval

Sets the minimum interval between connections to `cflowd`. This is not an exact interval timer, since `cfdcollect` processes all `cflowd` instances with a single thread (serially).

6.2.4 Example

Following is an example cflowd stanza. It tells `cfdcollect` to contact `cflowd` on foo.mydomain.net port 2056 every 300 seconds (5 minutes) to retrieve the tabular data from `cflowd`.

```
cflowd {
  host:                 foo.mydomain.net
  tcpCollectPort:       2056
  minPollInterval:      300
}
```

7 Network Service Provider NetFlow Configuration

In a network service provider environment, there are a number of considerations when configuring flow-switching and flow-export. Focusing only on those relevant to data collection using `cflowd`, these are the critical issues:

- The quantity of data. In a large network, it's important to reduce (or eliminate) data duplication. An optimal configuration generates only the data you need.
- The use of the data. The granularity necessary for the typical use (capacity planning and traffic management) must be available.

Data duplication has more than just performance and disk space penalties; if the possiblity exists that you've recorded the same traffic more than once in two different places, you often can't aggregate the traffic data from those two different places in a meaningful way without resorting to very granular duplication detection.

A typical network service provider wants source to destination traffic information at the AS and network prefix level (the AS matrix and the net matrix). To obtain this information, they **must** use version 5 flow-export or use version 8 flow-export and configure the prefix aggregation cache on the routers.

A provider may optionally want the port matrix and the protocol table. The necessary information is available in version 1, version 5 and version 8 flow-export. In the case of version 8, you need to configure the protocol/port aggregation cache on the router(s). In all cases, a provider will want the information per input interface (not aggregated across interfaces on a router) where available.

NetFlow data is input-based. Flows are instantiated as traffic enters the router, not as it exits. When you configure flow-switching for an interface on a Cisco router, flow data will be recorded for packets received from the network by the interface (and not for packets received from other interfaces on the router). In other words, flow data is recorded only in the receive direction per flow-switching interface.

Recording traffic data when the traffic first enters your network is critical to security related activities (attack backtracking). It's also important for determining the offered load to your network; it increases your data integrity (you get data on incoming traffic, possibly above and beyond what your transit facilities will handle), and it provides greater topological information (the source of your data is closer to the source of the traffic for which the data was collected).

Another reason to collect data as it enters your network: in IOS images using the prefix cache (not running any form of CEF), source netmask and source AS lookups frequently return a cache miss and IOS will not resort to a routing table lookup. The result is frequent zero values in the source AS and source netmask length fields in version 5 flow-export. The prefix cache is populated by destinations, not sources; if traffic to the source network is not seen frequently by the router, a prefix cache entry will not exist for the source network. Hence, asymmetric paths aggravate the zero value problem.

Taking these constraints into account, an optimal configuration for a network service provider usually looks something like **Figure 3**. To avoid recording data more than once, enable flow-switching only on interfaces at the edge of your network (external interfaces). Since you'll be collecting data as it enters the network, you'll also meet the other constraints.

8 More Information

Example configuration files are included in the cflowd package in the `etc/` subsdirectory. They're named `cflowd.conf.example` and `cfdcollect.conf.example`.

9 Appendix A

9.1 Version 1 flow-export

Version 1 flow-export packets contain a flow header followed by a number of flow entries. The number of flow entries in the packet is in the `count` field in the flow header.

Unlike version 5 flow-export, version 1 does not have sequence number information, AS numbers of netmask lengths. It is hence largely irrelevant in a network service provider environment.

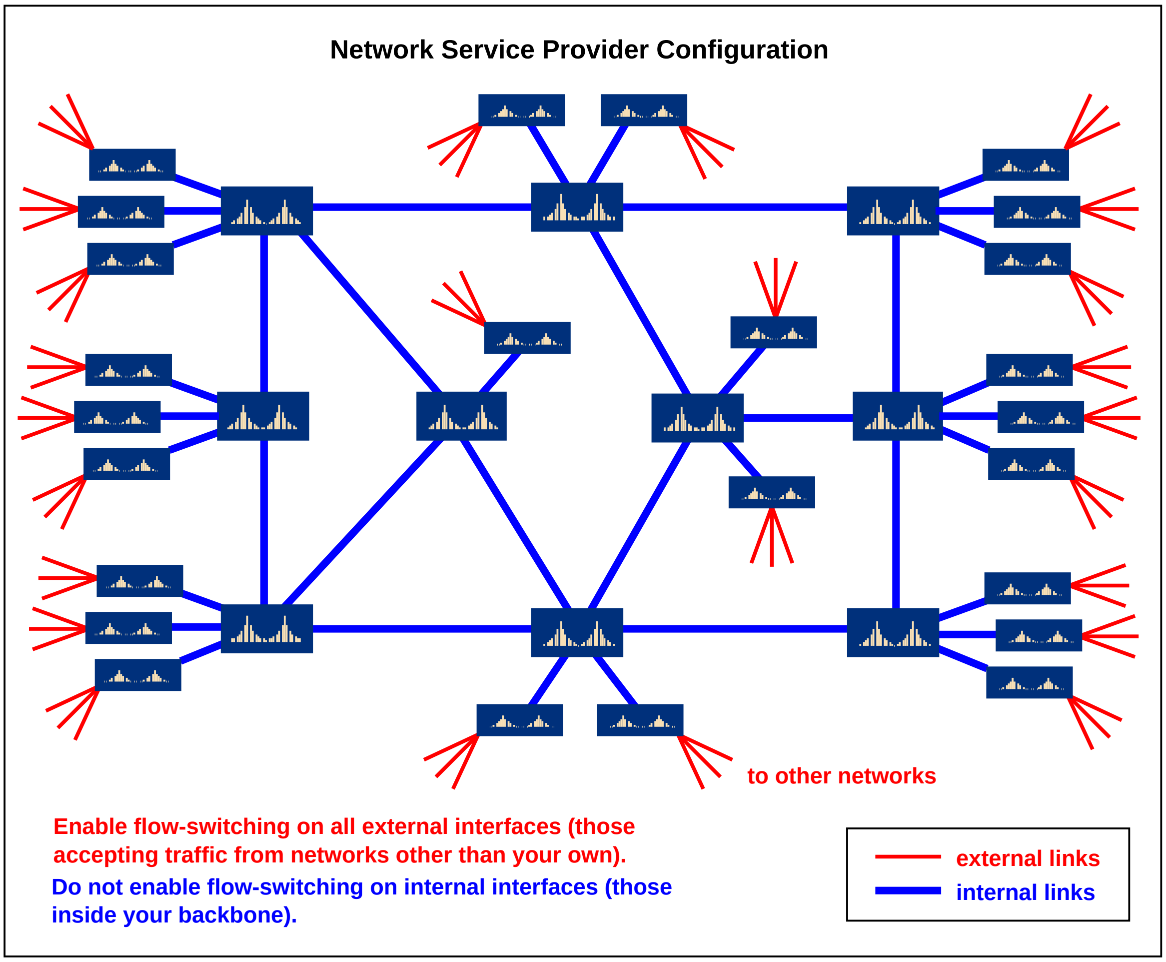

Figure 3: network service provider configuration

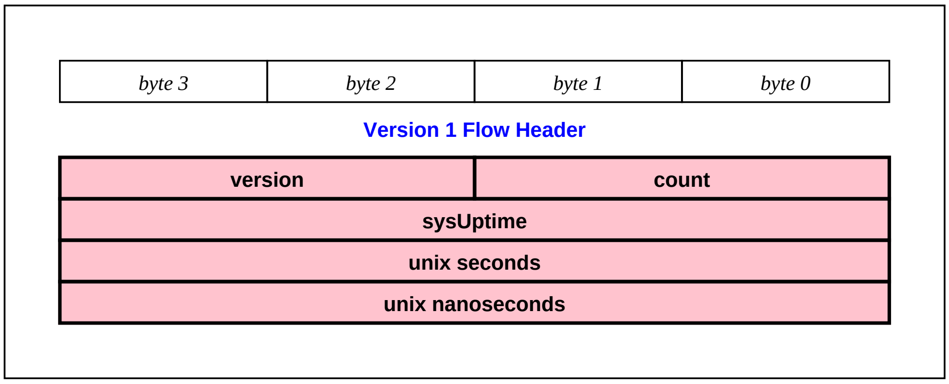

Figure 4: version 1 flow-export flow header

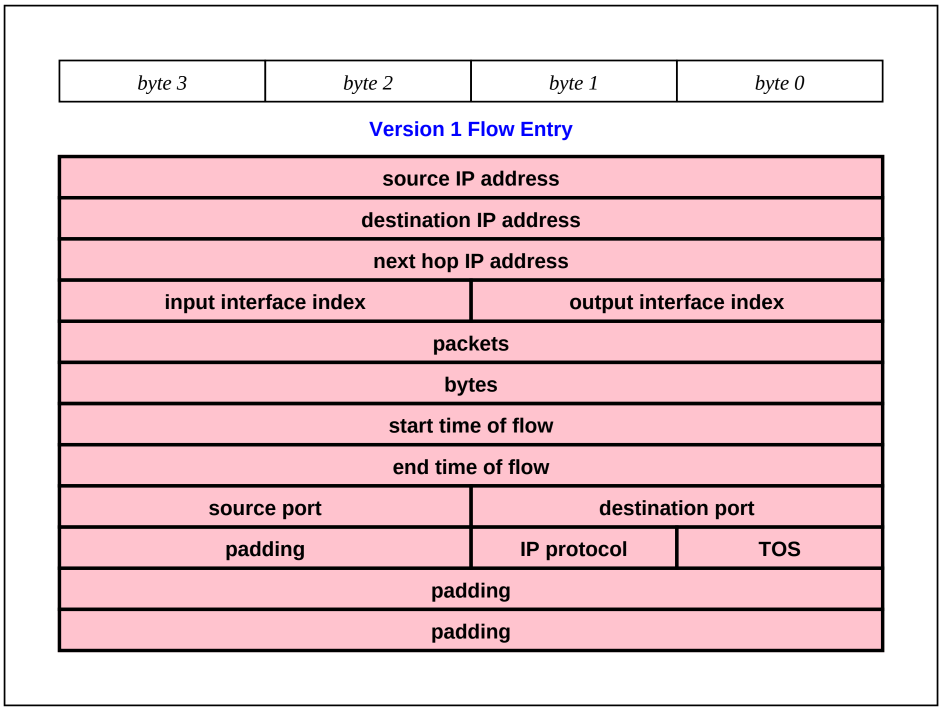

Figure 5: version 1 flow-export flow entry

9.2 Version 5 flow-export

Version 5 flow-export packets contain a flow header followed by a number of flow entries. The number of flow entries in the packet is in the `count` field in the flow header.

Unlike version 1 flow-export, version 5 flow-export has AS numbers and netmask lengths for the source and destination.

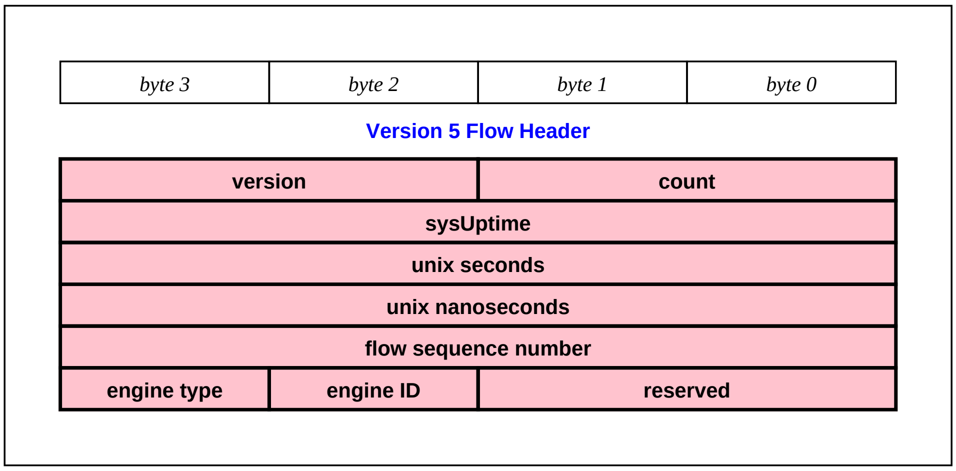

Figure 6: version 5 flow-export flow header

9.3 Version 8 flow-export

NOTE: version 8 flow-export is only available in IOS 12.0(2)S and 12.0(3)T images.

Version 8 flow-export packets contain aggregate information. These packets are significantly different in content than packets from other versions of flow-export; they only contain particular information, and are missing the granularity of other versions of flow-export. The intended benefit is for high-bandwidth situations in a provider environment where the most interesting information is to be used for capacity planning and highly granular information is not desired. Version 8 flow-export is more amenable to use in high-speed infrastructure where other versions of flow-export may be too process and bandwidth intensive to enable.

When using version 8 flow-export, you must configure aggregation caches on the router. A reference document is available at:

`ftp://ftp-eng.cisco.com/ftp/drowell/flow_agg.pdf`

Each version 8 flow-export packet contains data from a single aggregation cache on the router. There is a field in the version 8 flow-export header (`agg_method`) that indicates the aggregation cache from which the data was sent. In combination with the `agg_version`), this determines the layout of the data entries in the packet. Currently cflowd can make effective use of the protocol/port aggregation cache and the prefix cache, since they contain data needed to build the protocol table, port matrix, net matrix and AS matrix. cflowd can also use the AS aggregation cache, but this is generally not recommended because it makes it difficult to resolve 0 entries in the source and destination AS fields. You should also not configure export for both the AS cache and the prefix cache, since cflowd will use both types of data to populate the same tables, hence you'll wind up with data whose counters will be roughly twice as high as the actual traffic. I may add some heuristics for this in the future, but none are implemented in the current cflowd release. Hnece my recommendation is to configure flow-export for the prefix cache and the protocol/port cache and don't configure flow-export for any of the other caches (AS, source prefix or destination prefix).

NOTE: since there is no interface information present in the protocol/port data, cflowd will place all protocol/port flow entries under interface 0. In MIB-II, ifIndex can't have a value of 0, so this entry is easy for programs to recognize as not belonging to a particular interface.

NOTE: currently cflowd can recognize the source prefix flow data but has no tables in which to store it. Hence the data is not used by cflowd in the current release.

NOTE: currently cflowd can recognize the destination prefix flow data but has no tables in which to store it. Hence the data is not used by cflowd in the current release.

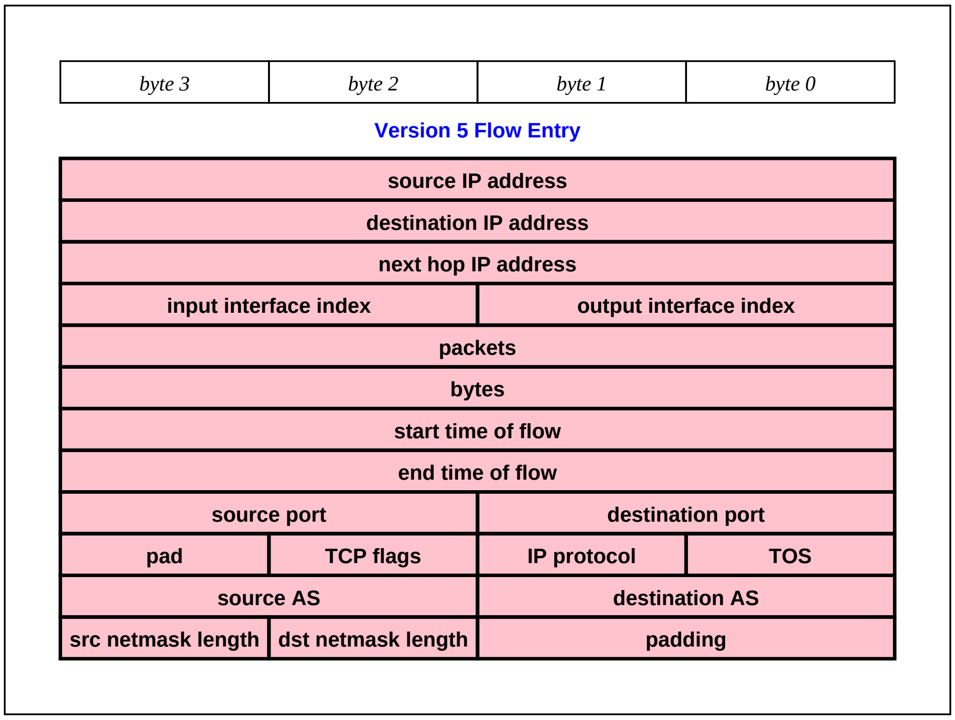

Figure 7: version 5 flow-export flow entry

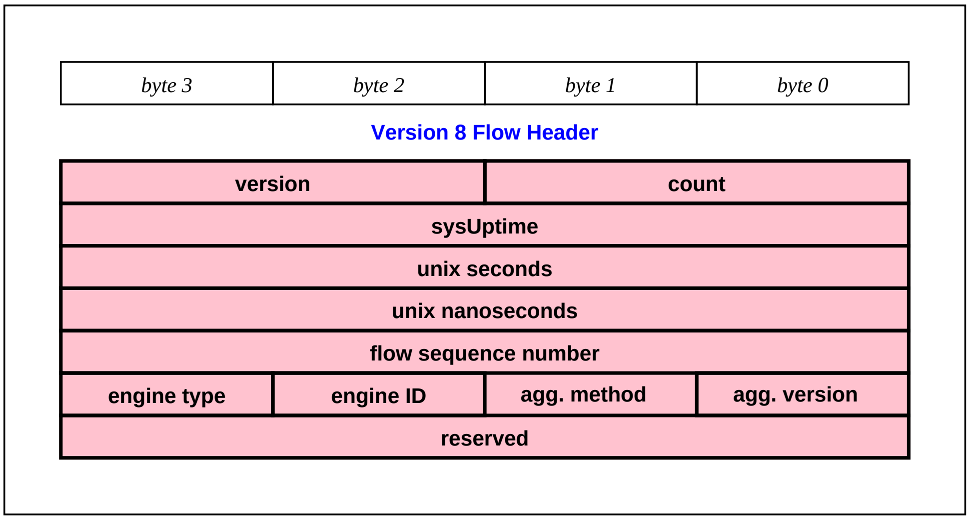

Figure 8: version 8 flow-export flow header

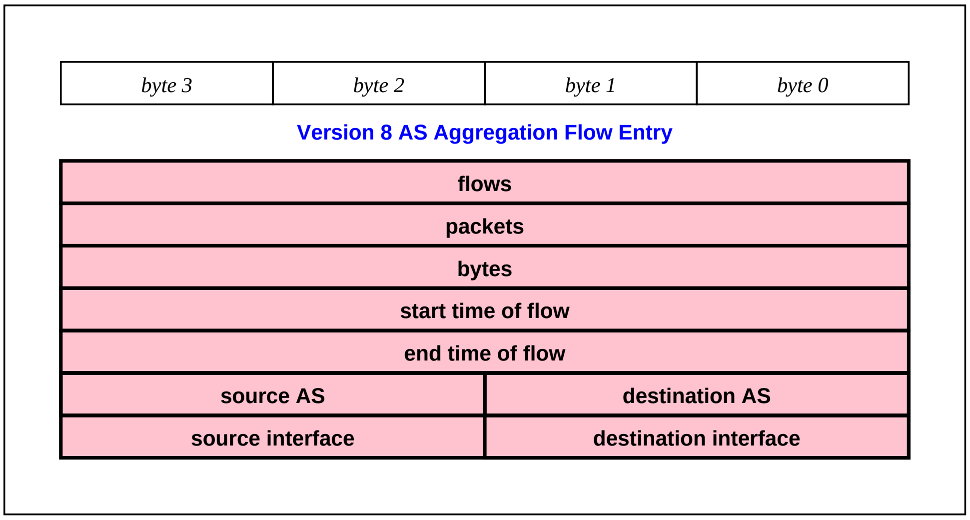

Figure 9: version 8 flow-export AS aggregation flow entry

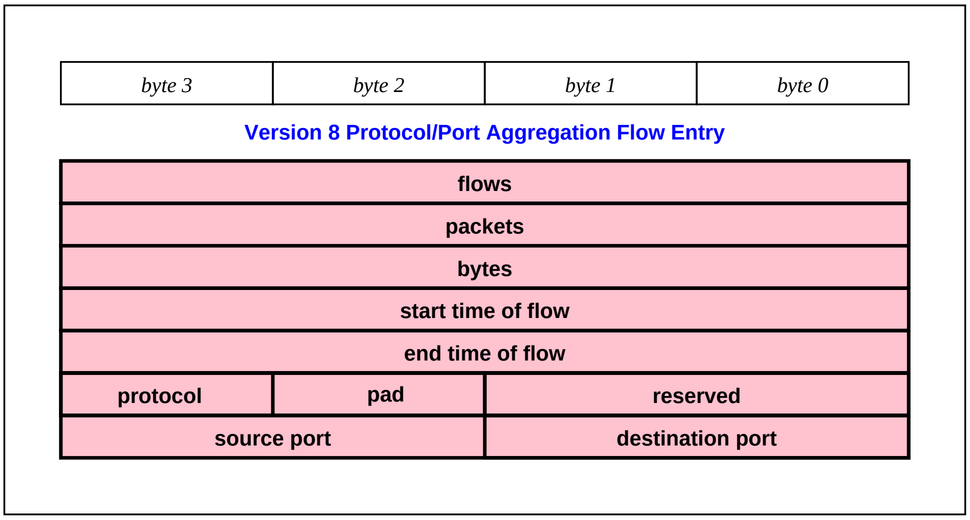

Figure 10: version 8 flow-export protocol/port aggregation flow entry

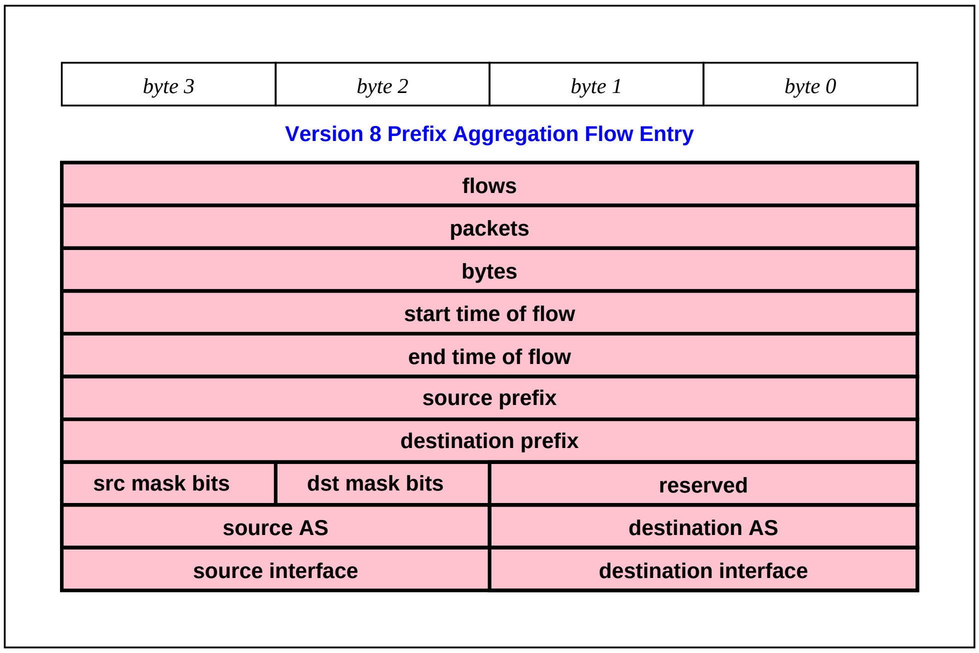

Figure 11: version 8 flow-export prefix aggregation flow entry

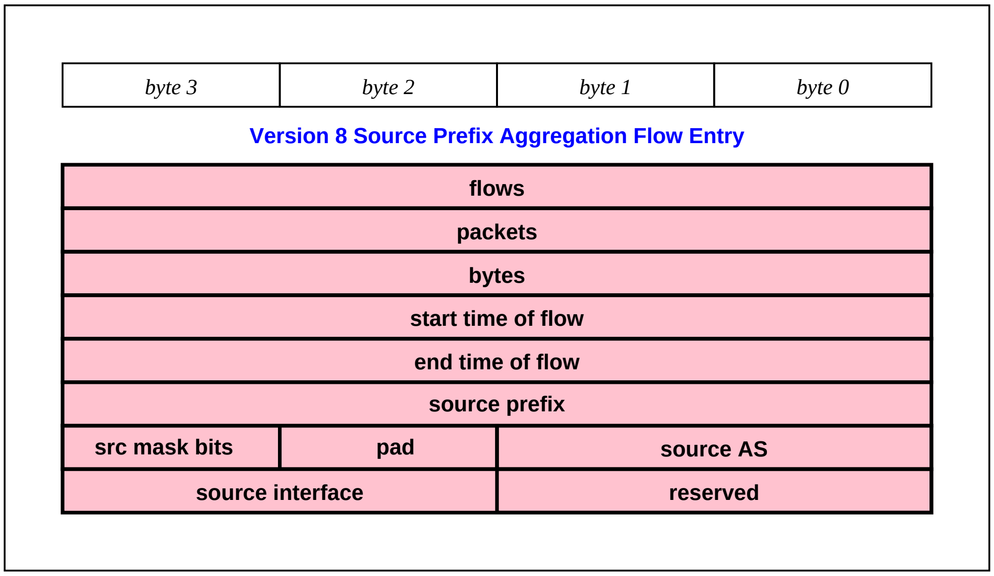

Figure 12: version 8 flow-export source prefix aggregation flow entry

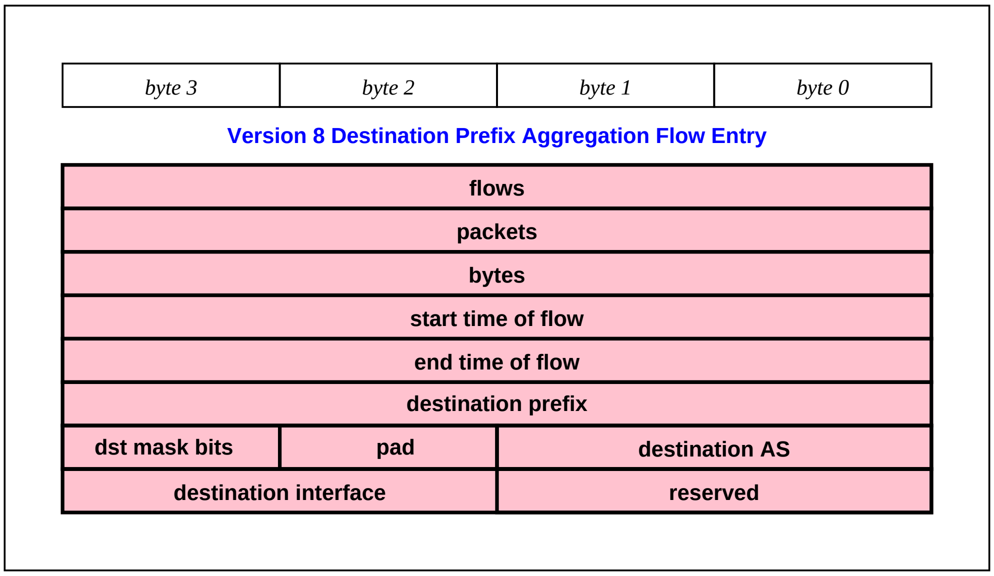

Figure 13: version 8 flow-export destination prefix aggregation flow entry