cflowd design

Daniel W. McRobb Sept 1998

This document contains a description of the design and architecture of `cflowd`.

Contents

1 System Overview **1**

2 cflowdmux **1**

3 cflowd **2**

4 cfdcollect **3**

1 System Overview

`cflowd` is divided into three distinct high-level modules, which map directly to three distinct programs.

The first program is responsible for handling raw flow data from Cisco routers and making it available to clients on the local host. The name of this program is `cflowdmux`.

The second program is responsible for maintaining per-input-interface tabular data for each Cisco, and passing it back to a central collector. This program is called `cflowd`.

The third program is responsible for retrieving tabular data from instances of `cflowd`. This program is called `cfdcollect`.

A diagram of the relationships of different programs in the system is shown in **Figure 1**.

2 cflowdmux

`cflowdmux` accepts Cisco flow-export packets arriving on UDP sockets and saves the packets in shared memory buffers. `cflowdmux` toggles between two packet buffers when writing packets, and clients may read the buffer that is not currently being written. A semaphore is used as the read/write mutex for each buffer, so that clients can avoid reading a buffer while it's being written by `cflowdmux`. **Figure 2** is a diagram of the basic functions of `cflowdmux`.

As UDP flow-export packets arrive, `cflowdmux` writes them into one of the shared memory packet buffers. Before writing into a share memory buffer, `cflowdmux` gets the semaphore for the buffer to prevent writing a buffer that's being read by a client like `cflowd`. When `cflowdmux` reaches the end of the buffer or 2 seconds transpires without coming to the end of the buffer, it releases the semaphore for the buffer and toggles to the other buffer.

The motivation for the use of shared memory for inter-process communication: performance. A bottleneck at this interface in the system is important to avoid because it will affect all downstream processing.

Currently `cflowdmux` is a standalone program, and can be run alone without clients. There's no particularly good reason to do so, since `cflowdmux` does not analyze the contents of the data. However, it does mean

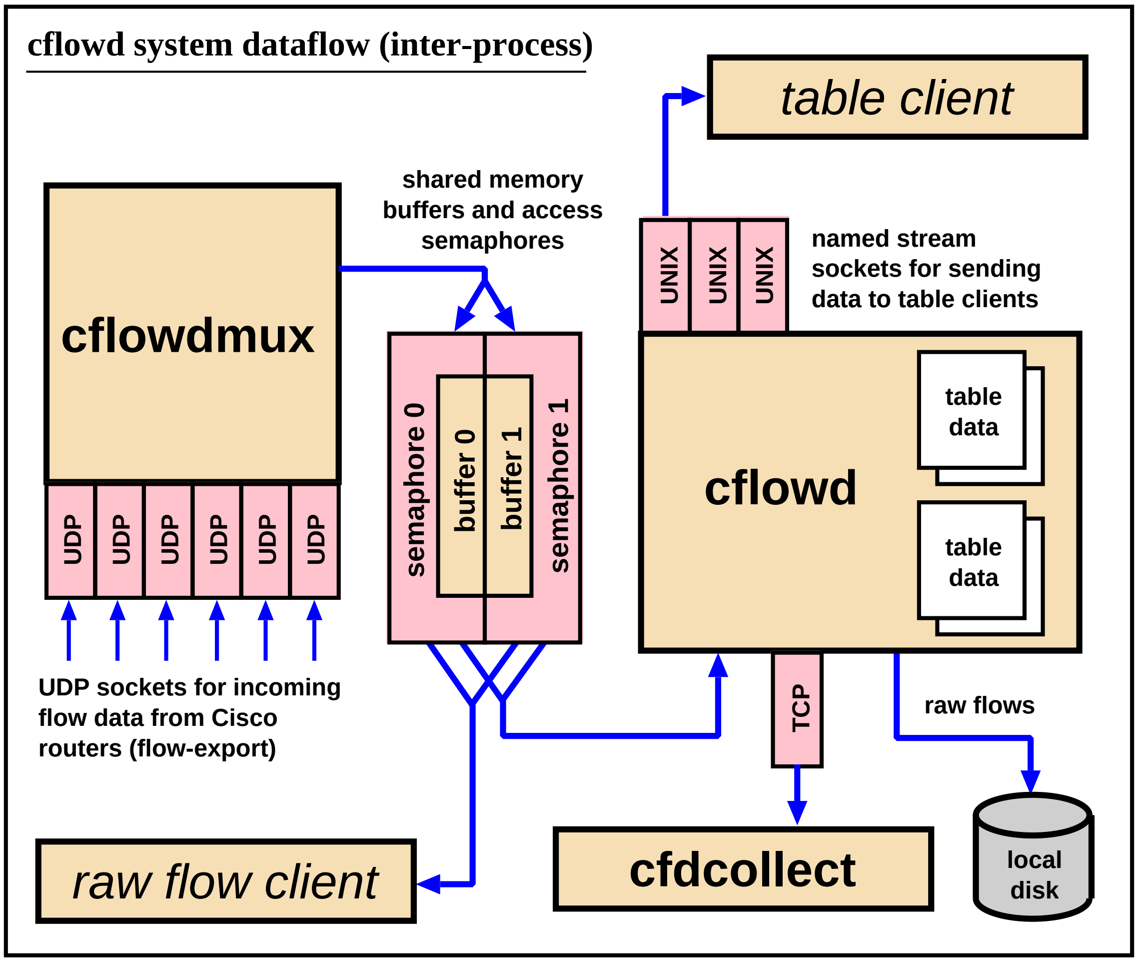

Figure 1: cflowd data flow detail

that it's fairly safe to start `cflowdmux` at boot time even if you will not be running clients all of the time.

3 cflowd

`cflowd` watches the shared memory packet buffer semaphores and reads a packet buffer when it becomes available. `cflowd` uses the `CflowdRawFlow` class to convert the flow-export packets into `CflowdRawFlow` objects, and uses the `CflowdRawFlow` objects to populate tables for which it is configured (the AS matrix, net matrix, etc.). `cflowd` does not yet write raw flow data to disk. This facility will be provided soon (but obviously be a bottleneck in most cases since disk I/O is generally slow).

`cflowd` will `listen()` for local client connections on a UNIX domain socket. If a client connects, `cflowd` will accept() and fork(), read the client request, then send the client the data they requested. This IPC channel is used for clients like `cfdases`, `cfdnets`, et. al. The data sent is current data, and `cflowd` does not clear any tables.

`cflowd` will also listen for remote client connections on a TCP socket. When a client connects, `cflowd` checks to see if the client is allowed to connect. If not, it closes the socket with prejudice. If allowed, `cflowd` will `fork()` and the child will send all of the tabular data to the client. The child simply exits when done (or if it gets a SIGPIPE while writing to the client), while the parent clears its tables, updates some timestamp information, then continues collecting data.

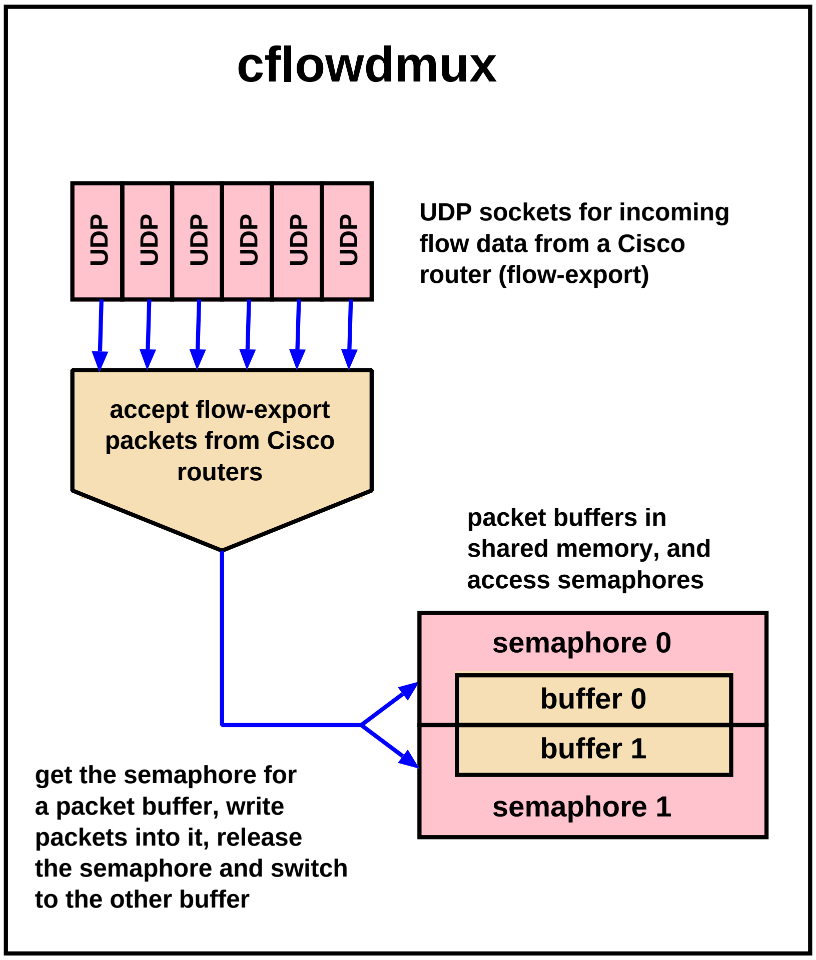

Figure 2: cflowdmux

4 cfdcollect

`cfdcollect` is used to retrieve and archive tabular data from instances of `cflowd` at regular intervals. This is how we generate time series data for the tabular information (AS matrix, net matrix, protocol table and port matrix). `cfdcollect` uses a `CflowdServer` class as an interface to instances of `cflowd`. `cfdcollect` writes data in ARTS files, using the ARTS C++ class library (from the arts++ package). A simple picture of tabular data collection via `cfdcollect` is shown in **Figure 3**.

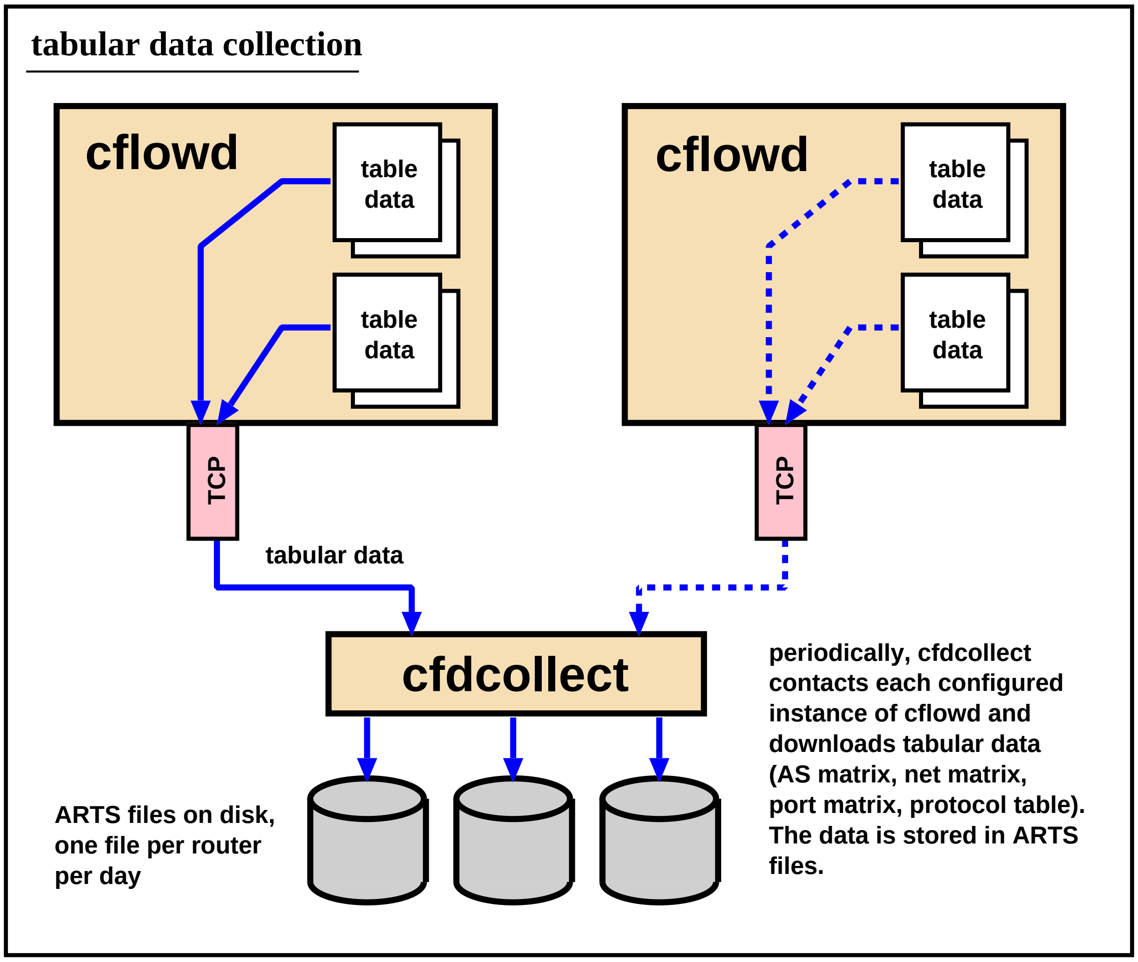

Figure 3: tabular data collection via cfdcollect