PNNL-SA-120800
PNNL-18428, Rev. 4

Pacific Northwest
NATIONAL LABORATORY

Proudly Operated by **Battelle** *Since 1965*

FLOWER Version 06 (flr06)

FLOWER Build Guide

July 2017

DS Curtis
BK Olsen

U.S. DEPARTMENT OF
ENERGY

Prepared for the U.S. Department of Energy
under Contract DE-AC05-76RL01830

Contents

Introduction

The network and software engineers developed software intended for deployment on the FLOWER appliance sensors. This Data Guide contains information about the data generated by that sensor.

This guide is intended to help the user understand the data generated by FLOWER on the FLOWER appliance sensors. If you have any questions about the data, please contact FLOWER support at flower-support@pnnl.gov.

Acronyms and Abbreviations

FC	FlowCache
GRE	Generic Routing Encapsulation
ICMP	Internet Control Message Protocol
IP	Internet Protocol
PAWS	Protect Against Wrapped Sequence (numbers)
PB	PacketBuilder
PP	PacketParser
PR	PacketRinger
RHEL	Red Hat Enterprise Linux
SFO	Session Force Out
SIT	Session Inactivity Timeout
TCP	Transport Protocol Port
UDP	User Datagram Protocol
UTC	Coordinated Universal Time
VLAN	virtual local area network
NTP	Network Time Protocol

Contents

Figures

Tables

1.0 Data Commonalities

Certain data elements are common across Network Flow appliance products such as a 5-tuple including source IP, source port, destination IP, destination port, and IP protocol.

1.1 Time Synchronized to Coordinated Universal Time

The FLOWER appliance represents time in the Coordinated Universal Time (UTC) time zone, regardless of the physical location of each FLOWER appliance. Analysis of site behavior needs to consider the local time zone and any daylight-saving time changes.[1,2]

1.2 Internet Protocol Address Representation

FLOWER represents all Internet Protocol (IP)v4 addresses in Octet-Coded Decimal (OCD) notation and IPv6 addresses in hexadecimal IPv6 notation.

1.2.1 IPv4 Addresses

An IPv4 address will be represented in quad-dotted notation.[3]

1.2.2 IPv6 Addresses

All IPv6 address representations will only be in full notation, that is, eight groups where each group contains hexadecimal digits. The short notation of IPv6 addresses, which replaces the longest sequences of zeros with '::', is not currently supported.

An example of an IPv6 address is `2001:DB8:0:0:0:0:1428:57AB`.

1 UTC Reference: http://aa.usno.navy.mil/faq/docs/UT.html.
2 FLOWER appliance clocks are synchronized to UTC using the NTP. Per RFC 1305 section E.8, "The NTP timescale is based on the UTC timescale, but not necessarily always coincident with it." "When a leap second is inserted in UTC and subsequently in NTP, knowledge of all previous leap seconds is lost."
3 https://en.wikipedia.org/wiki/IPv4

2.0 FLOWER Data

2.1 Overview

FLOWER is a network sensor designed and written by network and software engineers. FLOWER can either read tcpdump/libpcap format files, through a PCAP library, or from the Linux kernel ring buffer for an Ethernet interface on the appliance. Network and software engineers developed and tested FLOWER on various platforms including RHEL Linux, Windows XP, MacOS X, and the Bivio network appliance.

2.2 Terminology

Term(s)	Definition
flow, IP flow	A sequence of packets between two network addresses, closely related in time with common source and destination IP addresses and protocol, as well as source and destination port numbers if the protocol is UDP or TCP.
Bi-directional flow record	A record of traffic between two communicating network addresses. The address and port value of each endpoint are associated with the converse values for packets traveling in the reverse direction. The "first seen" packet on the wire establishes the flow connection in sensor memory, and follow-on packet values are added to that connection. Fields in each record include the first seen source IP address – packet counts in each direction, etc.
FLOWER	Flow Analyzer – Network sensor program name.

2.3 Restrictions

The FLOWER software has been constructed to handle standard Ethernet and 802.1Q (VLAN) traffic. No other layer 2 protocols are supported. The payload portions of the packet are not examined or analyzed in any way.

2.4 Processing

The code is organized around a simple loop. A packet is read from either a file or a live Ethernet interface. The packet header is decoded and information about the packet at the IP layer is saved in a flow cache. Protocol specific information for UDP and TCP is also saved. The packet is assigned to a new network flow or merged with an existing network flow.

Flow duration and ending times are measured to the limit of the accuracy of the time stamps recorded by the operating system's network device driver. The resolutions of the time stamps are microseconds, though the accuracy of the values may be much less. Observation of two different systems recording the same data stream and having their clocks synchronized with Network Time Protocol (NTP) can show that the time stamps may differ by at least 0.2 seconds.

The two parameters that control the assembly of packets into flows are

- Session Inactivity Timeout (SIT)
- Session Force Out (SFO).

The default value of SIT is nominally set to 2 minutes (120 seconds). Sessions that have been idle for this length of time are written to the capture file with the FRAGMENT_TYPE field set to "C", indicating that the record is complete, and there are no subsequent records related to this flow.

Long duration flows are defined as those whose packet arrival times exceed the span of time specified by the value of SFO, nominally set to 15 minutes (900 seconds). Long running sessions will have more than one record relating to that session. At the time a new packet for a flow is processed, the duration of the flow is compared with the SFO value. If the duration exceeds this value, the session is written to the data store, and the packet data accumulators are cleared. These sessions will be marked as being fragmented by using the `FRAGMENT_TYPE` field.

The first such record in a long running flow will have the `FRAGMENT_TYPE` field set to `F` to indicate that it is the first record, and there may be more records related to this record. Subsequent records will have the `FRAGMENT_TYPE` flag set to indicate they are not the first records in the flow. The `FRAGMENT_TYPE` in the record will be set to `N` if it is the next (but not the last) record that continues to exceed the SFO. The `FRAGMENT_TYPE` field will be set to `L` if this is the last record for the session, and its idle time has exceeded the SIT value. In this way, a long running flow will consist of a series of records. The summation of the data from the individual records will reproduce the totals for the flow.

If the packet capture (pcap) data is read from a file, and the last record has been processed, any flow records remaining in the cache will be written out to the data store and have the `FRAGMENT_TYPE` field set to `A` to indicate that the record was aborted. This process also occurs when reading from an Ethernet interface, and the FLOWER process is terminated.

2.5 File Naming Convention

All FLOWER output data files will have the format of <`YYYYMMDDhhmmss`>-<`SITE`>`-flr`<`GV`>`.`<`EXT`> where

- <`YYYYMMDDhhmmss`> is the human readable version of the UTC time value of the last packet from the first flow record written to file. **Note**: If the configuration file option, `suppress-ipv4-output`, is turned on, then the filename and the timestamp of the first record are not guaranteed to be the same.
- <`SITE`> is a unique identifier of the data source
- <`GV`> is the version of the data guide to which the output corresponds (not the version of the software)
- <`EXT`> is the file extension (default value is `dat`).

An example of an output filename for the pnnldev site at 5:24:30 pm on Sep. 24, 2008 is `20080924172430-pnnldev-flr06.dat`.

2.6 Record Types

The FLOWER executable produces five record types. Each record in the output file contains a record type number indicating what the record contains. The record types are:

Table 2.1. Record Type Details

#	Record Type	Primary Purpose	Explanation
0	Session	Data	Data was read from an active network interface.
1	Session	Data	Data was read from a PCAP file.
2	Heartbeat	Operations	Indicates there was no data to read for SIT seconds.
3	Metric	Data Quality	Contains counters, statistics, and performance metrics.
4	Version	Data Quality	Contains the same version information that is displayed when typing `FLOWER -v` on the command line.
5	Error	Troubleshooting	Shows a summary of the errors encountered while processing the records.

2.6.1 Session Record Type

The following examples show flow records for IPv4 and IPv6 protocols. Note that the key difference between IPv4 and IPv6 is the length of the populated address field. Higher level protocol fields are identical.

IPv6 ICMP flow:

```
1,pnl_dev,1227171519.699847,0.000000,,58,,,0:0:0:0:0:0:0:250,0:0:0:0:0:0:0:2,15
7,,211,,1,,,,,,0104,,+0104,,,,,,,,,,,,,,,,,,A,,
```

IPv4 TCP flow:

```
1,pnl_dev,1226731486.835646,0.000000,,6,122.169.107.137,140.221.166.35,,,,,62,,
1,,,,2971,445,,,,1,,02,,+02,1088212673,,1088212673,,,,,,,,,,A,,
```

IPv4 UDP flow:

```
1,pnl_dev,1226731486.917566,0.001797,,17,218.247.41.254,140.221.227.213,,,33,42
7,75,469,1,1,,,56672,2649,,,,,,,,,,,,,,,,,,,,,A,,
```

IPv4 ICMP flow:

```
1,pnl_dev,1226731488.986121,0.000000,,1,140.221.251.90,200.150.188.147,,,28,,70
,,1,,,,,,0301,0000,+0301,,,,,,,,,,,,,,,,,,A,,
```

IPv6 ICMPv6 flow Tunneled in IPv4 flow:

```
1,pnnl_dev,1226731132.190268,41.450664,,58,,,FE80:0:0:0:200:5EFE:8CDD:EEFD,FE80
:0:0:0:0:5EFE:D043:DB84,,,82,,1,,,,,,8500,,+8500,,,,,,,,,,1,140.221.238.253,208
.67.219.132,,,41,,,A,,
```

The following diagram is a simplistic flow chart that shows how each packet is parsed.

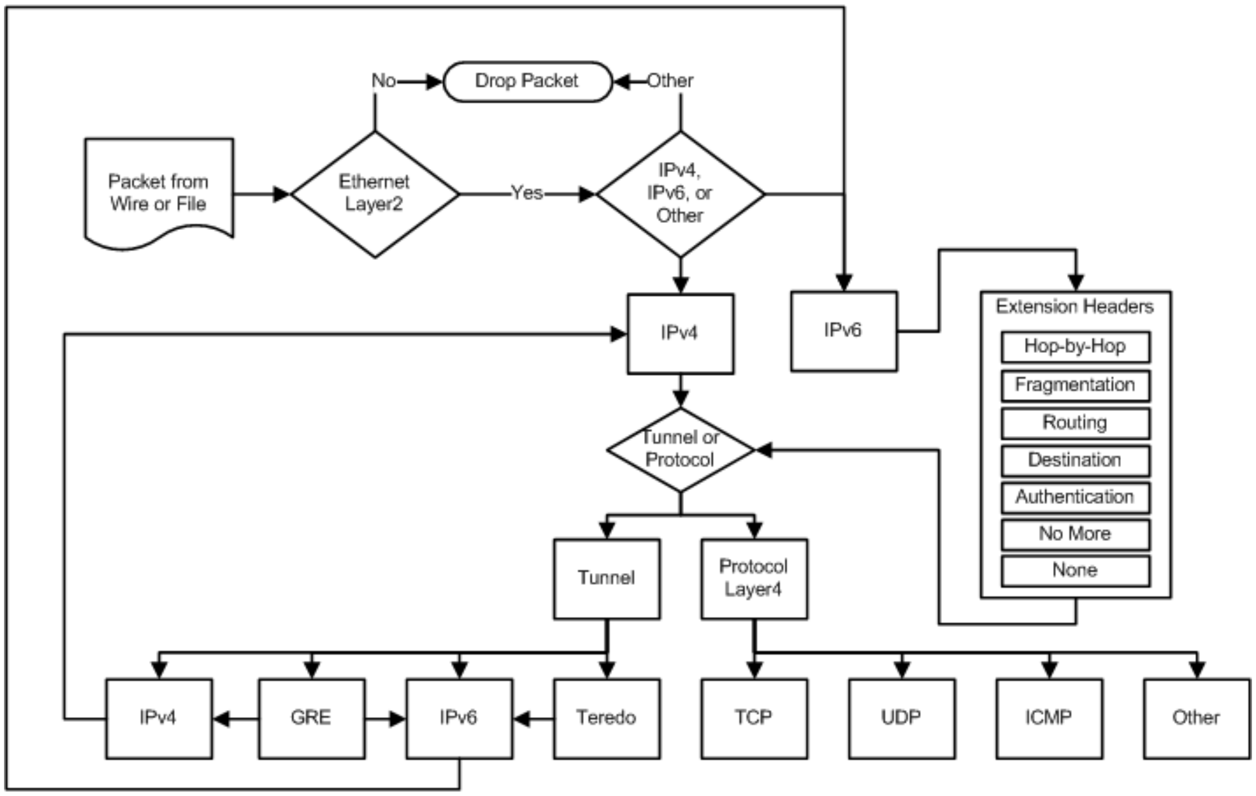

Figure 2.1. Simple Flow Chart of How a Packet is Parsed

Table 2.2 lists the fields in the order they will appear in the output from FLOWER along with a brief explanation.

The `Name` column holds a symbolic name for the field and the attributes of the data for that field. The field attributes include:

- `ValueType` attribute adds more context to the `DataType`. `ValueType` will be:
  - `CONSTANT` if the data never changes.
  - `TIME` for time values, `COUNTER` if the field represents a count.
  - `ENUM` if the field has an enumeration representing a range of numbers or letters.
  - `ADDRESS` for IPv4 or IPv6 addresses.
  - `BITSUM` for bitwise sums.
  - `FLAGSEQ` for protocol flag sequences.

  The `ValueType` is useful because all fields with the same `ValueType` are treated the same. For example, fields that have a `ValueType` of `COUNTER` will be null unless the value is greater than zero. Some fields with an attribute of `ENUM` will only have a few possible values, while others will have a large range of values.
- `Default` attribute will be used unless data from a flow overwrites the value.

- `Min` attribute will be the minimum non-null value for the field.
- `Max` attribute will be the maximum value for the field.
- `Null` attribute will indicate the circumstances when the field can be null.
- `Width` attribute indicates the range of characters needed to represent the field value.

The `Layer` column will have values of `2`, `3`, `4`, `C`, or `D`, which indicates origin of the field value. The value `2` represents the Ethernet layer. The value `3` represents the IP layer. The value `4` represents the Protocol layer (e.g., ICMP, TCP, UDP, etc). If the value is `2`, `3`, or `4`, the field value comes from or is derived from that networking layer. If the value is `C`, the field value comes directly from the FLOWER configuration file or command line. If the value is `D`, the field value is derived by some other means (e.g., pcap struct, formula, or algorithm).

Table 2.2. Field Order and Field Details

#	Name	Description	Data Type	Layer	Explanation
1	SOURCE ValueType: ENUM Default: N/A Min: 0 Max: 5 Null: Never Width: 1	Source Flag	int	`D`	This field indicates if the flows were created by reading packets from a network interface or from a file. The value will be a `0` if it is read from a network interface and `1` if it was read from a file. Any value greater than `1` is used for meta data and should not be processed as data. A value of `2` indicates that no packets were seen for SIT seconds. A value of `3` indicates the record is statistics. A value greater than `1` indicates that this record is used for Health and Status metadata for operations staff and should not be used for data quality purposes.
2	SITE ValueType: CONSTANT Default: N/A Min: 2 characters Max: 20 characters Null: Never Width Range: 2-20	Site ID	string	`C`	A text string that uniquely identifies the source of the data stream. The value must start with a letter and be alphanumeric (`a-zA-Z_0-9`)

#	Name	Description	Data Type	Layer	Explanation
3	TIMET ValueType: TIME Default: N/A Min: N/A Max: N/A Null: Never Width: 17-27	UNIX epoch format in seconds + microseconds	float	D	Standard UNIX time value (seconds elapsed since 00:00:00 UTC on January 1st, 1970) but including microseconds. The time specified is the time of the **last packet** to enter this flow. The start time of the flow can be derived by subtracting the DURATION from TIMET.
4	DURATION ValueType: TIME Default: N/A Min: 0.000000 Max: SFO + SIT[1].000000 Null: Never Width: SFO[1]+7	Duration	float	D	Session duration specified in seconds and microseconds. The value specified indicates the time span from the first seen packet to the last packet included in the flow. The format is <s.uuuuuu>, where s is seconds and uuuuuu is microseconds.
5	VLAN_ID ValueType: ENUM Default: Null Min: 0 Max: 4095 Null: When no VLAN ID Width: 0-4	VLAN ID	int	2	The VLAN Identifier. The value in the MAC Frame if it is tagged with an IEEE 802.1Q VLAN Identifier.[2] The value will be the last VLAN Identifier if there are nested VLANs.
6	PROTO ValueType: ENUM Default: Null Min: 0 Max: 255 Null: Never Width: 0-3	IP Layer Protocol ID	int	3	The IP number (1= ICMPv4, 6=TCP, 17=UDP, 58=ICMPv6, etc.).[3] Valid protocol values include 0 – 255.
7	FSSADDR_V4 ValueType: ADDRESS Default: Null Min: see section 1.2.1 Max: see section 1.2.1 Null: When flow is IPv6 Width: 0 or 7-15	First Seen Source IPv4 address	IPv4	3	IPv4 address of the source address of the first packet seen in the flow. The range of values follows the IPv4 convention.

1 See SFO and SIT in Section 2.4, Processing.
2 Reference: http://en.wikipedia.org/wiki/IEEE_802.1Q.
3 Reference: http://www.iana.org/assignments/protocol-numbers.

#	Name	Description	Data Type	Layer	Explanation
8	FSDADDR_V4 ValueType: ADDRESS Default: Null Min: see section 1.2.1 Max: see section 1.2.1 Null: When flow is IPv6 Width: 0 or 7-15	First Seen Destination IPv4 address	IP	3	IPv4 address of the destination address of the first packet seen in the flow. The range of values follows the IPv4 convention.
9	FSSADDR_V6 ValueType: ADDRESS Default: Null Min: see Section 1.2.2 Max: see Section 1.2.2 Null: When flow is IPv4 Width: 0 or 15-39	First Seen Source IPv6 address	IPv6	3	IPv6 full hexadecimal notation of the source address of the first packet seen in the flow. The range of values follows the IPv6 convention.
10	FSDADDR_V6 ValueType: ADDRESS Default: Null Min: see Section 1.2.2 Max: see Section 1.2.2 Null: When flow is IPv4 Width: 0 or 15-39	First Seen Destination IPv6 address	IPv6	3	IPv6 full hexadecimal notation of the destination address of the first packet seen in the flow. The range of values follows the IPv6 convention.
11	SRC_PAYLOAD ValueType: COUNTER Default: Null Min: 0 Max: 2^{64} -1 Null: When no data Width: 0-20	First Seen Source payload bytes	int	3	The sum of the payload bytes from packets with an address equal to the first seen source address of this flow. The payload size is the number of bytes in the packet not including the IP header length or the protocol header length for TCP, UDP, or ICMP packets. All other protocol headers are included in the payload size.
12	DST_PAYLOAD ValueType: COUNTER Default: Null Min: 0 Max: 2^{64} -1 Null: When no data Width: 0-20	First Seen Destination payload bytes	int	3	The sum of the payload bytes from packets with an address equal to the first seen destination address of this flow. The payload size is the number of bytes in the packet not including the IP header length or the protocol header length for TCP, UDP, or ICMP packets. All other protocol headers are included in the payload size.

#	Name	Description	Data Type	Layer	Explanation
13	SRC_BYTES ValueType: COUNTER Default: Null Min: 0 Max: 2^{64} -1 Null: When no data Width: 0-20	First Seen Source total bytes	int	3	The sum of the bytes reported by the IP header **plus** the number of bytes that comprise the Ethernet header from packets with an address equal to the first seen source address of this flow.
14	DST_BYTES ValueType: COUNTER Default: Null Min: 0 Max: 2^{64} -1 Null: When no data Width: 0-20	First Seen Destination total bytes	int	3	The sum of the bytes reported by the IP header **plus** the number of bytes that comprise the Ethernet header from packets with an address equal to the first seen destination address of this flow.
15	SRC_PACKETS ValueType: COUNTER Default: Null Min: 0 Max: 2^{64} -1 Null: When no data Width: 0-20	First Seen Source packet count	int	3	Number of packets with an address equal to the first seen source address of this flow.
16	DST_PACKETS ValueType: COUNTER Default: Null Min: 0 Max: 2^{64} -1 Null: When no data Width: 0-20	First Seen Destination packet count	int	3	Number of packets with an address equal to the first seen destination address of this flow.
17	SRC_IP_OPTS ValueType: COUNTER Default: Null Min: 0 Max: 2^{64} -1 Null: When no data Width: 0-20	First Seen Source IP layer option packet count	int	3	Number of packets that have IP layer options included in the packet with an address equal to the first seen source address of this flow.
18	DST_IP_OPTS ValueType: COUNTER Default: Null Min: 0 Max: 2^{64} -1 Null: When no data Width: 0-20	First Seen Destination IP layer option packet count	int	3	Number of packets that have IP layer options included in the packet with an address equal to the first seen destination address of this flow.

#	Name	Description	Data Type	Layer	Explanation
19	SPORT ValueType: ENUM Default: Null Min: 0 Max: 65535 Null: See Explanation Width: 0-5	First Seen Source port	int	4	Source ports respectively for UDP and TCP packets. A `null` value indicates one of the following: the flow is composed of layer 3 packets; or layer 4 packets exist, but IP layer protocol is not TCP or UDP.
20	DPORT ValueType: ENUM Default: Null Min: 0 Max: 65535 Null: See Explanation Width: 0-5	First Seen Destination port	int	4	Destination ports respectively for UDP and TCP packets. A `null` value indicates one of the following: the flow is composed of layer 3 packets; or layer 4 packets exist, but IP layer protocol is not TCP or UDP.
21	SRC_ICMP_FLAGS ValueType: BITSUM Default: Null Min: 0000 Max: FFFF Null: See Explanation Width: 0-4	ICMP source flags sum	hex	4	For ICMP Flows (IPv4 `PROTO=1, IPv6 PROTO=58`): The bitwise sum of the `Type` and `Code` fields that were set in the packets having their source address equal to the first seen source address of this flow. A `null` value indicates one of the following: the flow is composed of layer 3 packets; or layer 4 packets exist but the protocol is not ICMP.
22	DST_ICMP_FLAGS ValueType: BITSUM Default: Null Min: 0000 Max: FFFF Null: See Explanation Width: 0-4	ICMP destination flags sum	hex	4	For ICMP Flows (IPv4 `PROTO=1, IPv6 PROTO=58`): The bitwise sum of the `Type` and `Code` fields that were set in the packets having their source address equal to the first seen destination address of this flow. A `null` value indicates one of the following: the flow is composed of layer 3 packets; or layer 4 packets exist, but the protocol is not ICMP.

#	Name	Description	Data Type	Layer	Explanation
23	ICMP_EARLY_LATE_FLG ValueType: FLAGSEQ Default: Null Min: See Explanation Max: See Explanation Null: See Explanation Width: 0-160	ICMP Flags from early and late packets	string	4	For ICMP Flows (IPv4 `PROTO=1, IPv6 PROTO=58`): Up to 32 hex encoded ICMP `Type` and `Code` values (4 hex digits). Each value is preceded by either a "+" (source) or a "-" (destination) to indicate that flag is from a packet whose source address matches the first seen source address or first seen destination address of the flow. If there are more than 32 packets in the ICMP flow, the last five entries will be flags from the final five packets in the flow. If this is the case, they will be marked with a "<" (source) or a ">" (destination).
24	SRC_TCP_OPTS ValueType: COUNTER Default: Null Min: 0 Max: 2^{64} -1 Null: See Explanation Width: 0-20	First Seen Source TCP layer option packet count	int	4	Number of packets that have TCP layer options included in the packet with an address equal to the first seen source address of this flow. A `null` value indicates one of the following: the flow is composed of layer 3 packets; or layer 4 packets exist, but IP layer protocol is not TCP.
25	DST_TCP_OPTS ValueType: COUNTER Default: Null Min: 0 Max: 2^{64} -1 Null: See Explanation Width: 0-20	First Seen Destination TCP layer option packet count	int	4	Number of packets that have TCP layer options included in the packet with an address equal to the first seen destination address of this flow. A `null` value indicates one of the following: the flow is composed of layer 3 packets; or layer 4 packets exist, but IP layer protocol is not TCP.
26	SRC_TCP_FLAGS ValueType: BITSUM Default: Null Min: 00 Max: FF Null: See Explanation Width: 0-2	TCP Source flags sum	hex	4	For TCP Flows (`PROTO=6`): The bitwise sum of the TCP flags (8 bits) that were set in the packets having an address equal to the first seen source address of this flow. A `null` value indicates one of the following: the flow is composed of layer 3 packets; or layer 4 packets exist, but the protocol is not TCP.

#	Name	Description	Data Type	Layer	Explanation
27	DST_TCP_FLAGS ValueType: BITSUM Default: Null Min: 00 Max: FF Null: See Explanation Width: 0-2	TCP Destination flags sum	hex	4	For TCP Flows (`PROTO=6`): The bitwise sum of the TCP flags (8 bits) that were set in the packets having an address equal to the first seen destination address of this flow. A `null` value indicates one of the following: the flow is composed of layer 3 packets; or layer 4 packets exist, but the protocol is not TCP.
28	TCP_EARLY_LATE_FLG ValueType: FLAGSEQ Default: Null Min: See Explanation Max: See Explanation Null: See Explanation Width: 0-96	TCP Flags from early and late packets	string	4	For TCP Flows (`PROTO=6`): Up to 32 hex encoded TCP flag values (2 hex digits). Each value is preceded by either a "+" (source) or a "-" (destination) to indicate that flag is from a packet whose address matches the first seen source address or first seen destination address of the flow. If there are more than 32 packets in the TCP flow, the last five entries will be flags from the final five packets in the flow. If this is the case, they will be marked with a "<" (source) or a ">" (destination).
29	SRC_FIRST_TCP_TS ValueType: ENUM Default: Null Min: 0 Max: 2^{32} -1 Null: See Explanation Width: 0-10	Source TCP initial timestamp	int	4	The TCP timestamp from the first packet whose address matches the first seen source address of the flow. A value of `0` indicates that the first seen source packet had a value of `0`. A `null` value indicates one of the following: the flow is composed of layer 3 packets; or there was no timestamp option set in the TCP header or layer 4 packets exist, but the protocol is not TCP.

#	Name	Description	Data Type	Layer	Explanation
30	SRC_FIRST_TCP_SEQ ValueType: ENUM Default: Null Min: 0 Max: 2^{32} -1 Null: See Explanation Width: 0-10	Source TCP initial sequence number	int	4	The TCP sequence number from the first packet whose address matches the first seen source address of the flow. A value of `0` indicates that the first seen source packet had a value of `0`. A `null` value indicates one of the following: the flow is composed of layer 3 packets; or layer 4 packets exist, but the protocol is not TCP.
31	DST_FIRST_TCP_TS ValueType: ENUM Default: Null Min: 0 Max: 2^{32} -1 Null: See Explanation Width: 0-10	Destination TCP initial timestamp	int	4	The TCP timestamp from the first packet whose address matches the first seen destination address of the flow. A value of `0` indicates that the first seen destination packet had a value of `0`. A `null` value indicates one of the following: the flow is composed of layer 3 packets; or there was no timestamp option set in the TCP header, or layer 4 packets exist, but the protocol is not TCP.
32	DST_FIRST_TCP_SEQ ValueType: ENUM Default: Null Min: 0 Max: 2^{32} -1 Null: See Explanation Width: 0-10	Destination TCP initial sequence number	int	4	The TCP sequence number from the first packet whose address matches the first seen destination address of the flow. A value of `0` indicates that the first seen destination packet had a value of `0`. A `null` value indicates one of the following: the flow is composed of layer 3 packets; or layer 4 packets exist, but the protocol is not TCP; or there is one-way traffic; (e.g., ssh session to a host that is not responding).

#	Name	Description	Data Type	Layer	Explanation
33	SRC_LAST_TCP_TS ValueType: ENUM Default: Null Min: 0 Max: 2^{32} -1 Null: See Explanation Width: 0-10	Source TCP last timestamp	int	4	The TCP timestamp from the last packet whose address matches the first seen source address of the flow. A value of `0` indicates that the last seen source packet had a value of `0`. A `null` value indicates one of the following: the flow is composed of layer 3 packets; or there was no timestamp option set in the TCP header or layer 4 packets exist, but the protocol is not TCP.
34	SRC_LAST_TCP_SEQ ValueType: ENUM Default: Null Min: 0 Max: 2^{32} -1 Null: See Explanation Width: 0-10	Source TCP last sequence number	int	4	The TCP sequence number from the last packet whose address matches the first seen source address of the flow. **Note**: If there is only one source packet, this value will be the same as the `SRC_FIRST_TCP_SEQ` number. A `null` value indicates one of the following: the flow is composed of layer 3 packets; or layer 4 packets exist, but the protocol is not TCP.
35	DST_LAST_TCP_TS ValueType: ENUM Default: Null Min: 0 Max: 2^{32} -1 Null: See Explanation Width: 0-10	Destination TCP last timestamp	int	4	The TCP timestamp from the last packet whose address matches the first seen destination address of the flow. A value of `0` indicates that the last seen destination packet had a value of `0`. A `null` value indicates one of the following: the flow is composed of layer 3 packets; or there was no timestamp option set in the TCP header or layer 4 packets exist, but the protocol is not TCP.

#	Name	Description	Data Type	Layer	Explanation
36	DST_LAST_TCP_SEQ ValueType: ENUM Default: Null Min: 0 Max: 2^{32} -1 Null: See Explanation Width: 0-10	Destination TCP last sequence number	int	4	The TCP sequence number from the last packet whose address matches the first seen destination address of the flow. **Note**: If there is only one destination packet, this value will be the same as the `DST_FIRST_TCP_SEQ` number. A `null` value indicates one of the following: the flow is composed of layer 3 packets; or layer 4 packets exist, but the protocol is not TCP; or there is one-way traffic; (e.g., ssh session to a host that is not responding).
37	SRC_TCP_RETRANS ValueType: COUNTER Default: Null Min: 0 Max: 2^{64} -1 Null: See Explanation Width: 0-20	Source TCP retransmissions	int	4	Number of TCP packets that have retransmitted, whose address matches the first seen source address of the flow.
38	DST_TCP_RETRANS ValueType: COUNTER Default: Null Min: 0 Max: 2^{64} -1 Null: See Explanation Width: 0-20	Destination TCP retransmissions	int	4	Number of TCP packets that have retransmitted, whose address matches the first seen destination address of the flow.
39	TUNNEL_DEPTH ValueType: ENUM Default: Null Min: 0 Max: 255 Null: If no tunnel exists Width: 0-3	The depth or number of tunnels in the current flow.	int	4	A simple tunnel (IP over IP) will have a depth of 1. A tunnel in another tunnel will have a value of 2.
40	TUNNEL_SADDR_V4 ValueType: ADDRESS Default: Null Min: see section 1.2.1 Max: see section 1.2.1 Null: When flow is IPv6 Width: 0 or 7-15	Tunnel source IPv4 address	IPv4	4	IPv4 address of the source address of the first packet seen in the flow. The range of values follows the IPv4 convention. The address of the outer most layer of the tunnel.

#	Name	Description	Data Type	Layer	Explanation
41	TUNNEL_DADDR_V4 ValueType: ADDRESS Default: Null Min: see section 1.2.1 Max: see section 1.2.1 Null: When flow is IPv6 Width: 0 or 7-15	Tunnel destination IPv4 address	IPv4	4	IPv4 address of the destination address of the first packet seen in the flow. The range of values follows the IPv4 convention. The address of the outer most layer of the tunnel.
42	TUNNEL_SADDR_V6 ValueType: ADDRESS Default: Null Min: see section 1.2.2 Max: see section 1.2.2 Null: When flow is IPv4 Width: 0 or 15-39	Tunnel source IPv6 address	hex	4	IPv6 full hexadecimal notation of the source address of the first packet seen in the flow. The range of values follows the IPv6 convention. The address of the outer most layer of the tunnel.
43	TUNNEL_DADDR_V6 ValueType: ADDRESS Default: Null Min: see section 1.2.2 Max: see section 1.2.2 Null: When flow is IPv4 Width: 0 or 15-39	Tunnel destination IPv6 address	hex	4	IPv6 full hexadecimal notation of the destination address of the first packet seen in the flow. The range of values follows the IPv6 convention. The address of the outer most layer of the tunnel.
44	TUNNEL_PROTO ValueType: ENUM Default: Null Min: 0 Max: 255 Null: If no tunnel exists Width: 0-3	Tunnel IP layer protocol	int	4	The IP protocol number (`1= ICMPv4, 6=TCP, 17=UDP, 58=ICMPv6`, etc.).[1] This is the protocol used to establish the outer most layer of the tunnel. Valid protocol values include `0 – 255`.
45	TUNNEL_SPORT ValueType: ENUM Default: Null Min: 0 Max: 65535 Null: See Explanation Width: 0-5	Tunnel source port	int	4	Source port number used to establish the outer most layer of the tunnel.
46	TUNNEL_DPORT ValueType: ENUM Default: Null Min: 0 Max: 65535 Null: See Explanation Width: 0-5	Tunnel destination port	int	4	Destination port number used to establish the outer most layer of the tunnel.

1 Reference: http://www.iana.org/assignments/protocol-numbers.

#	Name	Description	Data Type	Layer	Explanation
47	FRAGMENT_TYPE ValueType: ENUM Default: N/A Min: See Explanation Max: See Explanation Null: Never Width: 1	Fragment type	char	D	Session fragmentation indicator. A value of C indicates a completed flow; a value of F indicates this record is the first of a long running data stream; a value of N indicates this record is part of a previously long running data stream but is not the last; a value of L indicates this is the last record of a previously long running data stream; a value of A indicates this record was forced to the data file before the timeout parameters were exceeded (usually indicates records were flushed at program shutdown or no more data when reading from an input file).
48	IP_FRAG ValueType: ENUM Default: Null Min: N/A Max: 1 Null: See Explanation Width: 0-1	IP Packet Fragmentation	int	D	If IP packets are fragmented and have to be reassembled into a packet before being added to a flow session, this field will be set to 1.
49	ANOMALY ValueType: BITSUM Default: Null Min: 0 Max: 2^{64} -1 Null: See Explanation Width: 0-16	Flow Anomaly	int	D	If any part of an IP packet in a Flow does not conform to an RFC, the Flow has an anomaly and will have bit set. See discussion below for more information on anomalies.

2.6.1.1 Time Fields

The TIMET and DURATION fields represent the time span between the first seen packet and the last seen packet. If the packets were received out of order (e.g., when read from a file causing the timestamps to be out of order), the times will be set using the first seen packet and the last seen packet.

2.6.1.2 IP Fragment Processing

FLOWER will reassemble IP packets that are fragmented prior to adding them to the flow cache. The process matches fragmented packets on IP addresses and the IP identification field in the IP header. Once a packet is reassembled, the packet will be added to the flow cache as a new flow or merge with an existing flow.

2.6.1.3 Length Fields

The `SRC_PAYLOAD` and `SRC_BYTES` fields (as shown in the Table 2.2) refer to byte counts associated with the traffic originating with the "first seen" source address. Similarly, `DST_PAYLOAD` and `DST_BYTES` fields refer to byte counts associated with the traffic originating with the "first seen" destination address. For clarity, this section refers only to the former set of fields. The same principles apply to the latter set of fields.

In all cases, the relationships `SRC_PAYLOAD < SRC_BYTES` and `DST_PAYLOAD < DST_BYTES` should apply. IP fragmentation is handled by reassembling the packets as they come in using the IP identifier and then applying them to the appropriate flow. The process for calculating payload is shown in Table 2.3.

Table 2.3. Process for Calculating Payload

Packet Type	Payload Calculation
First Packet Fragment	Subtract IP header and any ICMP, UDP, or TCP header
Middle and Last Packet Fragments	Subtract IP header

Malformed packets can also cause failures of the above relationships. Of course, these malformations may be intentionally malicious or unintentional host/network problems.

2.6.1.4 ICMP and TCP Flag Fields

The intent of the `ICMP_EARLY_LATE_FLG` and `TCP_EARLY_LATE_FLG` is to capture and preserve the value of the flag and the direction relative to the first seen source packet. Flow records will hold the first 27 flags and the last 5 flags. If the number of flags exceeds 32, the first 27 flags will be preserved and indicate the direction of the packets using "+" and "-" and the last 5 will indicate direction using "<" `and` ">" respectively. For example, if the number of packets was 34, then packets 28 and 29 would not be preserved, and packets 30-34 would indicate their direction using "<" and ">". If the number of flags is less than or equal to 32, all flags will be preserved and use "+" and "-" to indicate direction.

2.6.1.5 TCP Session Flags Description

This section describes the TCP flags and how the sessions, incorporating the flags, are represented in the data. It is intended to assist in understanding the session information relative to the TCP flags. The first two bits, indicated by `XX`, are reserved in the TCP stack and can be a `0` or `1`. They are functionally ignored for our purposes. Thus, the flags `+02` and `+C2` are functionally equivalent.

`FIN XX00 0001 x01` – Finish session

Request to close the virtual circuit in an orderly fashion. This process is achieved by one system sending a packet with the finish flag set. If the other end is ready to close the virtual circuit, it will return a packet with the finish flag set as well. If both ends agree, then the connection is terminated.

`SYN XX00 0010 x02` – Synchronize (typically to start session handshake)
Synack response should follow. During the handshake, each of the endpoints sends a packet with the synchronize bit set and the initial sequence numbers that will be used for the connection. This provides the information that both systems need to proceed orderly.

`RST XX00 0100 x04` – Reset connection (connection refused)
The reset should only be seen when the connection cannot be torn down in an orderly fashion (due to errors), or when an incoming connection request is for an invalid socket (e.g., no service listening on the destination port) and must be rejected. Once a system issues a reset, the other end should not send any more packets, however, there is nothing to stop it; therefore, multiple resets may be common. Some systems use the reset – without first sending packets with the finish bit set – to expediently end connections; this is contrary to design.

`PSH XX00 1000 x08` – Push data (do not buffer)
The push flag is set to indicate that all of the data for the packet has been loaded by the sending application. The push flag is used to indicate that all of the data has been provided and indicates that the data should not be buffered.

`ACK XX01 0000 x10` – Acknowledge
When the acknowledge flag is set, the packet contains an acknowledgement of a previous packet. All packets should have the acknowledge flag set, except for the first packet in the stream (the syn packet) and the reset packets. The acknowledgement number identifies the next byte of data that a recipient expects to receive.

`URG XX10 0000 x20`
If the urgent flag is set, the packet contains urgent data up through the value in the urgent pointer field. The urgent pointer field is included in the TCP header and identifies the sequence of the last byte of urgent data contained in the packet.

Example combinations of typical flow sessions:

3-way handshake	`+02-12+10`	`+SYN-SYN/ACK+ACK ….`
Connection refused	`+02-04`	`+SYN-RST`
Connection close	`…-11+10 or …-11+10+04 or … <11>10>04`	

Example values of TCP individual flags and the typical meaning

`SYN x02`	Open connection request
`SYN/ACK x12`	Acknowledge syn packet
`PSH/ACK x18`	Acknowledge your packet, here is data
`PSH/FIN x09`	Last data packet
`FIN/ACK x11`	Acknowledge packet and finish; often followed by an `x11` and `x10` response

`RST/ACK x14`	Acknowledge your packet and connection refused
`SYN/FIN x03`	Invalid flag combination (SINFIN scan)
`SYN/RST x06`	Invalid flag combination
`SYN/FIN/PSH x0B`	Invalid flag combination
`SYN/FIN/RST x07`	Invalid flag combination
`URG/ACK/PSH/FIN x39`	Invalid flags (indicative of the XMAS scan)

Bits 7 and 6 are reserved, and their use is not precluded. Thus, a `SYN` could be `C2` (or `42` or `82`) and, a `RST` could be `C4` (or `84` or `44`); this means the possible – yet not necessarily valid – values for the first nibble are `x0-F`.

2.6.1.6 TCP Timestamp and Sequence Number Fields

The TCP timestamp values are extracted from TCP header options if the option is present. If there are no TCP header options the value of the TCP timestamp is 0. The timestamp and sequence numbers are handled in accordance to RFC 1321 (Protect Against Wrapped Sequence numbers – PAWS). The timestamp option is used to logically extend the 32-bit sequence number into a 64-bit number. Most TCP implementations do not actually use a system clock timestamp and typically use an arbitrary counter, which makes the timestamp field slightly misleading. In addition, the timestamp is only incremented by the sender, when that sender deems necessary to increment the value. In other words, there is no guarantee that the timestamp fields will increment.

The intent of the `SRC_FIRST_TCP_TS/SRC_FIRST_TCP_SEQ`, `SRC_LAST_TCP_TS/SRC_LAST_TCP_SEQ`, `DST_FIRST_TCP_TS/DST_FIRST_TCP_SEQ`, and `DST_LAST_TCP_TS/DST_LAST_TCP_SEQ` is to capture the first and last TCP timestamp and sequence numbers in a flow. The first observed TCP timestamp and sequence number establishes the value of the `SRC_FIRST_TCP_TS/SRC_FIRST_TCP_SEQ` (or `DST_FIRST_TCP_TS/DST_FIRST_TCP_SEQ`) and `SRC_LAST_TCP_TS/SRC_LAST_TCP_SEQ` (or `DST_LAST_TCP_TS/DST_LAST_TCP_SEQ`) fields, even if that value is `0`. Subsequent TCP sequence number observations will update the `SRC_LAST_TCP_TS/SRC_LAST_TCP_SEQ` (or `DST_LAST_TCP_TS/DST_LAST_TCP_SEQ`) field value, but not the `SRC_FIRST_TCP_TS/SRC_FIRST_TCP_SEQ` (or `DST_FIRST_TCP_TS/DST_FIRST_TCP_SEQ`) field value.

2.6.1.7 TCP Retransmissions

TCP retransmissions are a normal occurrence in network communications. The SRC_TCP_RETRANS and DST_TCP_RETRANS fields contain the count of packets in the flow that meet all the following criteria:

- The packets in the flow are TCP
- The current packet contains a non-zero payload

- There was a previous packet in the same direction with a non-zero payload (aka NZP)
- The current sequence number is lower than the sequence number of the NZP

2.6.1.8 Tunnel Fields

All tunnel fields will be the same data types and follow the same guidelines as the non-tunnel fields. For example, the `TUNNEL_PROTO` field will have the data type and follow the same rules as the `PROTO` field. The following tunnels are supported:

- IPv4 in IPv4, IPv4 in IPv6 (IP protocols 4)
- IPv6 in IPv4, IPv6 in IPv6 (IP protocol 41)
- GRE (IP protocol 47)
- Teredo (IPv6 in UDP)

The following diagram shows a tunnel where the tunnel depth is 2. The Session Record will consist of the actual Flow data and have the outermost tunnel information in the tunnel fields of the Session Record. The innermost tunnel information is not captured but it is added in the calculation of SRC_BYTES and DST_BYTES.

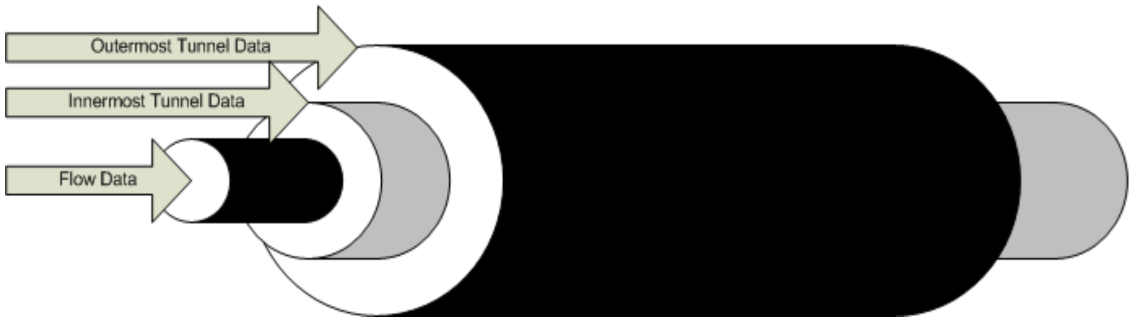

Figure 2.2. A Tunnel with a Depth of 2. The Session Record will contain the Flow Data and the outermost tunnel information in the tunnel fields.

2.6.1.9 Anomaly Field

Anomalies are flags that indicate that one or more of the packets in the Flow had a structure or data value that was not consistent with an RFC but enough information was retrieved to create a valid Flow record. For example, if a TCP header has a data offset value of 3 then we cannot trust the TCP header information. The reason is that the data offset value must have a minimum of value of 5 and a maximum value of 15. The example packet has a malformed TCP header that does not conform to RFCs for TCP. The anomaly field is a bitwise sum of all the anomalies encountered in the Flow record. The following is a list of anomalies and how they should be interpreted:

- 00001 – A malformed TCP header.
- 00002 – A malformed UDP header.

- 00004 – A Teredo packet is missing an IPv6 header. This implies that a packet was using a source or destination UDP port of 3544 for something other than tunneling IPv6 packets using Teredo as the tunnel mechanism.
- 00008 – A malformed Teredo header.
- 00010 – A malformed Teredo Authentication header.
- 00020 – A malformed ICMP header.
- 00040 – An IP packet with the Generic Routing Encapsulation tunneling protocol (47) does not have a GRE header.
- 00080 – A malformed GRE header.
- 00100 – A malformed IPv4 header.
- 00200 – A malformed IPv6 header. NOTE: This is currently not used but reserved for future use.
- 00400 – The value specified in the IPv6 header for the payload does not match what was seen on the wire.
- 00800 – An IPv6 header has a Hop-by-Hop extension header but it is not the first extension header.
- 01000 – An IPv6 header has multiple Fragmentation extension headers.
- 02000 – The total bytes calculated by using values in the headers exceed the number of bytes seen on the wire. The total bytes for that packet will be modified to the number of bytes seen on the wire.
- 04000 – The payload calculated by using values in the headers exceeds the number of bytes seen on the wire or the number of total bytes calculated. The payload for that packet will be modified to the number of total bytes.
- 08000 – An IPv6 header has multiple Hop-by-Hop extension headers.
- 10000 – An IPv6 header has more than two Destination extension headers.
- 20000 – An IPv6 header has multiple Routing extension headers.
- 40000 – An IPv6 header has multiple Authentication extension headers.

For example, if the anomaly field has a value of 08800 it means that at least one packet in the flow had an IPv6 header with multiple Hop-by-Hop extension headers (anomaly 08000) and that the second Hop-by-Hop is out of order (anomaly 00800). The RFCs for IPv6 specifies that there can only be one Hop-by-Hop extension header and that it must be the first extension header. Another case might be a single packet in a tunnel that has an invalid IPv4 packet header. For example, a flow with a single packet could have valid IPv6 tunnel addresses and have a malformed IPv4 header. The resulting flow would have IPv6 tunnel addresses, no IPv4 or IPv6 source or destination addresses, and an anomaly of 00100.

2.6.2 Heartbeat Record Type

The Heartbeat record can be used by data quality and operations staff to help answer runtime anomalies. The record indicates there was no data to read for SIT seconds. The purpose of this record is to indicate that FLOWER is running correctly but there may be a problem with a network interface or it may indicate an abnormal lack of network activity. The Heartbeat record follows the convention of a normal Session

Record but the only useful data in the Heartbeat record is the SITE name and the timestamp. All other data MUST be ignored. An example Heartbeat record looks like:

```
2,pnl_dev,1226731486.000000,0.000000,,0,0.0.0.0,0.0.0.0,,,,,,,,,,,,,,,,,,,,,,
,,,,,,,,,C,,
```

2.6.3 Metric Record Type

The Metric record contains counters, statistics, and performance data about the Session records in the file. An example record looks like:

```
3,PB:pc#2114676:pt#00:00:04.058238:pps#521082.30,PR,PP:tpr#2114676:ebbc#943109:
ebpc#13875:v4bbc#0:v4bpc#0:v6bbc#0:v6bpc#0:v4gbc#1558846428:v4bpc#2017281:v6gbc
#6490018:v6gpc#83520,FC:mff#13:mnf#80850:mfc#2019948:sfc#80850
```

The record contains the same data that is displayed when running interactively. The interactive output looks like:

```
Packets captured:   2114676
Processing time:    00:00:04.058238
Packets per second: 521082.30
Packets received:   2114676
Bad Eth Bytes:      943109
Bad Eth Pkts:       13875
Bad Ipv4 Bytes:     0
Bad Ipv4 Pkts:      0
Bad Ipv6 Bytes:     0
Bad Ipv6 Pkts:      0
Good Ipv4 Bytes:    1558846428
Good Ipv4 Pkts:     2017281
Good Ipv6 Bytes:    6490018
Good Ipv6 Pkts:     83520
Max Frag Flows:     13
Max Norm Flows:     80850
Merged Flow Count:  2019948
Summary Flow Count: 80850
```

The format of the record is encoded to save space and is intended to be parsed by data quality software. The first section is PB (for PacketBuilder) which is populated when the Session records are read from a file. The second section is PR (for PacketRinger) which is populated when the Session records are read from a network interface and the "use-ring" option is used from the command line or the configuration file. Only one of the PB or PR fields will be populated. The values for PB and PR are the same and consist of the following data:

- `pc – packets captured from the wire or file`
- `pt – processing time in seconds and microseconds`
- `pps – packets per second`

The third section is PP (for PacketParser)

- `tpr – total packets received from PacketRinger or PacketBuilder`
- `ebbc – ethernet bad byte count`
- `ebpc – ethernet bad packet count`

- `v4bbc – Ipv4 bad byte count`
- `v4bpc – Ipv4 bad packet count`
- `v6bbc – Ipv6 bad byte count`
- `v6bpc – Ipv6 bad packet count`
- `v4gbc – Ipv4 good byte count`
- `v4gpc – Ipv4 good packet count`
- `v6gbc – Ipv6 good byte count`
- `v6gpc – Ipv6 good packet count`

The fourth section is FC (for FlowCache)

- `mff – max number of fragmented flows`
- `mnf – max number of normal flows`
- `mfc – merge flow count (number of times flows were merged)`
- `sfc – summary flow count (number of flows)`

2.6.4 Version Record Type

The Version record contains the versions of the software used to create the output file. An example record looks like:

```
4,Ver:UNRELEASED (d654),Compiler:GNU g++
4.1.2,OptLevel:3,Debug:OFF,BoostLibVer:1-38,PcapLibVer:0.9.4,Compiled:Nov  3
2009  17:40:10,DataGuideVer:flr06
```

The Version record is the same information that is displayed when a user types "`FLOWER -v`" on the command line. See the Operations Guide for more detailed information.

2.6.5 Error Record Type

The Error record only exists if an error was encountered during the processing of the Session records in the file. An example record looks like:

```
5,121:1,255:86
```

The Error record is a comma separated list of error numbers and the number of times each error occurred. For example, in the record above the error 121 occurred 1 time and the error 255 occurred 86 times. These records can be used by data quality software, operations staff, and software developers to improve the software if errors occur too frequently. This data is an indication that something may be incorrect or that tuning parameters need to be updated. For example, if the flow cache is always full at a given site, the size of the flow cache at that site may need to be increased. The non-fatal errors that can be in the Error record are:

- `121 – FlowCache cannot merge fragmented flow`
- `122 – FlowCache cannot merge normal flow`
- `123 – FlowCache cannot add fragmented flow to the cache`
- `124 – FlowCache cannot add normal flow to the cache`
- `125 – FlowCache has a full fragment cache`
- `126 – FlowCache has a full normal cache`
- `251 – PacketParser has an ethernet packet that cannot be parsed`

- `252 - PacketParser has an IP packet that cannot be parsed`
- `253 - PacketParser has an Ipv4 packet that cannot be parsed`
- `254 - PacketParser has an Ipv6 packet that cannot be parsed`
- `255 - PacketParser cannot get a Flow from the Flow Pool`
- `256 - PacketParser has a VLAN packet but not enough data to pars`