Peeling Away Timing Error in NetFlow Data

Brian Trammell, Bernhard Tellenbach, Dominik Schatzmann, and Martin Burkhart

ETH Zurich, Switzerland

Abstract. In this paper, we characterize, quantify, and correct timing errors introduced into network flow data by collection and export via Cisco NetFlow version 9. We find that while some of these sources of error (clock skew, export delay) are generally implementation-dependent and known in the literature, there is an additional cyclic error of up to one second that is inherent to the design of the export protocol. We present a method for correcting this cyclic error in the presence of clock skew and export delay. In an evaluation using traffic with known timing collected from a national-scale network, we show that this method can successfully correct the cyclic error. However, there can also be other implementation-specific errors for which insufficient information remains for correction. On the routers we have deployed in our network, this limits the accuracy to about 70ms, reinforcing the point that implementation matters when conducting research on network measurement data.

1 Introduction

In the practice of network measurement, packet data is collected at one or more observation points within a network. Some combination of transformations may then be applied to the packets, such as sampling, or assembly into flows. This transformed data then undergoes some combination of export, collection, aggregation, filtering, storage, and analysis, in order to produce successively refined information from which knowledge about the network is derived, whether for research or operational purposes. Each of these stages may be seen as a function applied to the result of the previous stage. Ideally, each of these functions should lead to further refinement of the information of interest without introduction of error or loss of fidelity. Some of these stages, especially observation, export, and collection, should have no impact on the information content at all.

However, this is not the case. Each stage in the measurement process may introduce error. Some of these sources of error are well-known, such as failing to properly provision measurement devices leading to packet loss, or failing to synchronize clocks among distributed observation points. Other errors have more obscure causes. In this work, we examine a cyclic source of timing error in flow data exported via Cisco Netflow version 9 (v9) [1] which, instead of having a deployment- or implementation-time cause, is a consequence of the design of the protocol itself. Together with load-dependent export delay and long-term drift of the clocks from which timestamps are generated, we find that the accuracy of

timestamps in flow data exported using v9 is degraded by about three orders of magnitude, to about two seconds in the worst case, instead of millisecond-level precision implied by the protocol.

After discovering this error in a flow data set collected from a national-scale network and stored as a sequence of raw NetFlow v9 export packets in received order, we set about "peeling" these layers of error away, devising an algorithm for correcting the cyclic error while compensating for delay and drift. We do this only with reference to timing information on the NetFlow v9 export packets; that is, the correction is independent of the individual flows exported. This is important both for the scalability of the approach, and for its independence on the actual content of the traffic. We find that our approach can completely remove the protocol-induced cyclic error, in the general case allowing millisecond timing resolution with NetFlow v9, even for flows exported in different export packets. This is sufficient to sequence flows occurring between one millisecond and one second apart, e.g. to determine the direction of a bidirectional flow as in [2] when the connection establishment time is more than 1ms, or to enable flow-based round-trip-time measurement for quality of service applications.

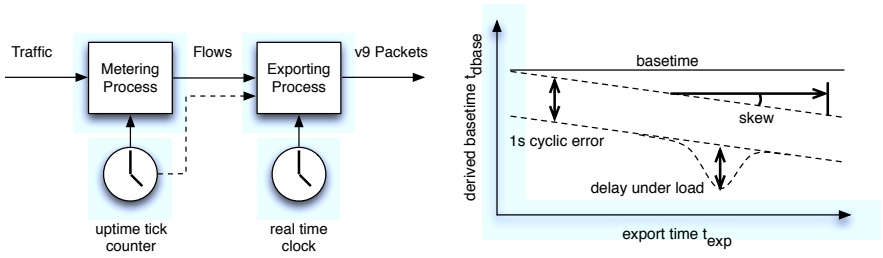
However, in practice we can only peel so far: on the Cisco 6500 and 7600 series routers that collect the data in the network we measure, additional flow-level inaccuracy of about 70ms remains, which we do not have sufficient information to correct. We thereby confirm that deployment, implementation, and design-time choices made in the systems collecting and processing the traffic data under study do not have the neutral effect one could assume on the data. We further note that this work quantitatively supports the common wisdom that router-based flow measurement is generally insufficient for applications requiring precision timing.

Section 2 characterizes the sources of timing errors we see in the examined data set and section 3 quantifies them and presents concrete examples of artifacts in the data caused by these sources. We then present and evaluate a method for correcting cyclic error based solely on the export packet headers in section 4. In section 5, we review related work in data fidelity for network measurement, and we present our conclusions in section 6.

2 Characterizing Timing Error in NetFlow version 9

NetFlow v9 [1] exports flow data in records described by templates, allowing the flexible inline definition of record formats. However, flow start and end timestamps are expressed as with older NetFlow versions, in terms of uptime, or the time that has passed since the device started. This approach has the advantage of not requiring a real-time clock at the *metering process*, which generates flows from an observed traffic stream. We call these per-flow timestamps f_{start} and f_{end}.

Flows are exported by an *exporting process* in protocol data units called *export packets* by NetFlow v9. The exporting process stamps each outgoing packet with an export timestamp p_{export} expressed in UNIX epoch time (i.e., seconds since



(a) Timing sources in NetFlow v9 (b) Sources of basetime error

Fig. 1. Illustrating basetime error

midnight UTC, 1 January 1970). It also exports the time since the metering process started p_{uptime}. This arrangement is illustrated in Figure 1(a)[1].

From these two timestamps, the time at which the device started (which we call the *true basetime* or t_{base}) is given by $t_{base} = p_{export} - p_{uptime}$. The start and end time in UNIX epoch seconds for each flow in the packet can then be determined by adding t_{base} to each of the per-flow timestamps f_{start} and f_{end} for each flow in the packet.

This would be the ideal situation. However, while the uptime is expressed in milliseconds, the export time is truncated to second-level precision before export due to the design of the NetFlow v9 packet header, implicitly flooring it. Therefore, the *derived basetime* from each packet is in effect given by $t_{dbase} = \lfloor p_{export} \rfloor - p_{uptime}$. The implicit floor operation causes the milliseconds part of the export timestamp not to be accounted in the basetime derivation, injecting a *cyclic error* of up to one second subtracted from the derived basetime, which can consequently cause errors in the flow timestamps leading to incorrect sequencing of flows exported in different export packets.

Further complicating this situation are two effects of the architecture and its use of separate clocks. First, the two clocks are not necessarily synchronized; that is, one second does not necessarily pass on the real time clock for each second on the uptime counter. This *clock skew* can be a result of inaccuracy in either of the two clocks. The magnitude of the skew observed in our data set is on the order of seconds per day, and appears to be stable over time. Second, the timestamp from the uptime counter and the timestamp from the real time clock are not necessarily applied simultaneously to the export packet. Export packets may be held at the Exporting Process due either to resource exhaustion or explicit export rate limiting. This *delay* can inject a further subtractive error into the derived basetime. Delays observed in our data set are uncommon, intermittent, correlated with periods of heavy load, and on the order of less than one second. These three sources of error are illustrated in Figure 1(b); here, we show the

[1] Here we use terminology and arrangement from the IPFIX architecture [3], since the IPFIX architecture was based on that from NetFlow v9.

true basetime, and the dotted lines define the area within which the derived basetimes fall.

To see how this would affect flow measurement, consider the following example: a flow f_1 starts at 1.000s after router start (i.e., true basetime), and a flow f_2 which starts at 1.100s. The export packet containing f_1 is exported at 11.000s, and that containing f_2 at 11.950s. Assuming no drift or other delay on this time scale, we then have:

$t_{dbase1} = \lfloor 11.000 \rfloor - 11.000 = 0,\ t_{dbase2} = \lfloor 11.950 \rfloor - 11.950 = -0.950$

$f_1 = t_{dbase1} + 1.000 = 1.000, \qquad f_2 = t_{dbase2} + 1.100 = 0.150$

Even though f_1 started before f_2, the apparent sequence is reversed.

We observe a further peculiarity of export in the data from our Cisco routers: that of derived basetime *quantization*. The derived basetimes in our data set are all divisible by 4ms. Whether this is a source of error or not is uncertain without examining the implementation: the 4ms quantization could be caused either by export driven by a 4ms interrupt, or by timestamps being stored internally in 4ms units.

Due to the magnitude of these errors, applications which perform time-series aggregation with intervals greater than one second (e.g., most billing applications) are largely unaffected. However, we show in section 3 that any assumption that devices exporting NetFlow v9 are capable of millisecond-level accuracy and/or strict ordering of flows does not hold. We set out to see what could be done to improve this situation.

The most troublesome source of error on a per-flow basis is the cyclic error. The timestamps of the flows skew at the same rate as that of the basetime, so skew, while visible in the basetime series, is cancelled out for each flow. Therefore, in section 4, we will focus on correcting cyclic error, treating skew, delay, and quantization as complications to correction.

3 Quantifying Timing Errors in NetFlow v9

Our data set includes data collected from SWITCH[2], the Swiss research and education network. This network contains about 2.3 million IPv4 addresses, and the typical total traffic volume ranges from 500 megabytes to one gigabyte per second. We receive NetFlow v9 from six Cisco routers (6500 or 7600 series) deployed around the SWITCH border; we designate these routers A-F. Each router also exports flows from multiple Source IDs; these correspond to line cards. Here we examine one week of data, 26 June to 3 July, 2010.

Figure 2 shows the density of exported derived basetimes for a single source. The upper part of the figure is a density map of exported basetimes by offset from the maximum observed basetime. The lower part shows the number of export packets per second for the same time period.

The vast majority of basetimes fall within the skewed one-second cyclic error band. Note the daily seasonality in the density of basetimes. There is a maximum number of flows which can be exported in an export packet (ep), so higher

[2] `http://www.switch.ch`

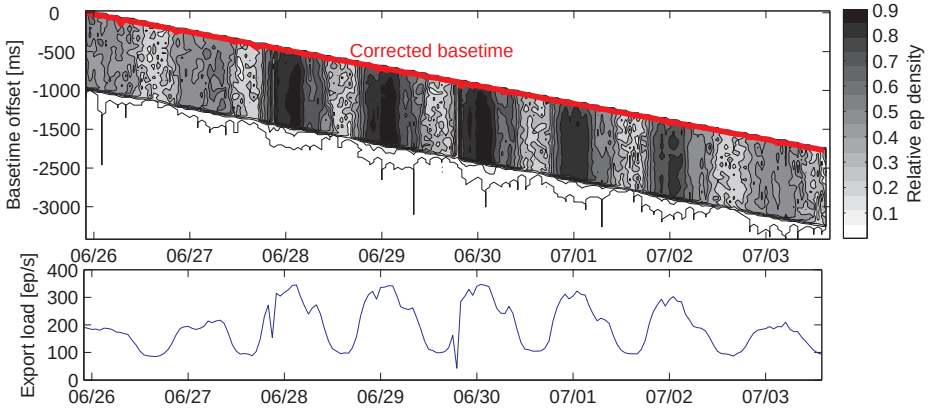


Fig. 2. Derived basetime density distribution, and basetime correction, for a single source on a single router (A/513).

traffic load leads to larger flow counts leads to larger export packet counts. This increases the number of unique derived basetimes seen per second. However, this density is not uniformly distributed among the possible values within the cyclic error band, a fact which further complicates correction. Delay also increases with traffic load, due to resource exhaustion and/or rate limiting in the exporting process. However, the number of delayed export packets is relatively low even under load.

We examine the errors on each source on each router in Table 1. Here, errors are reported in relation to the presumed real basetime as determined by the correction mechanism detailed in section 4; therefore we report the rate of unique observed derived basetimes per second and the correction interval parameter used. These will be explained further below.

Mean drift is a per-source, not a per-router parameter; we hypothesize that this is related to some physical property of the clocks on each of these line cards. Drift ranges from about -2s/day to +1s/day.

The minimum and maximum measured error show that the range due to cyclic error and delays ranges from 1050ms to 2280ms; lower values demonstrating predominantly cyclic error, with higher values as evidence of more delay. The width of the band between the 5th and 95th percentile ranges between about 950ms and 1200ms, demonstrating that the vast majority of this error is cyclic. Note that all error measurements for the routers are values divisible by four milliseconds; this is an artifact of the 4ms quantization mentioned above.

We also examined the output of the `softflowd`[3] NetFlow v9 metering and exporting process, which was developed independently from the Cisco codebase. `softflowd` was run on a small set of flows generated on an experimental local-area network, running on a Mac OS X host. We observed the same cyclic error, but negligible drift, negligible delay, and no quantization of derived basetimes.

[3] `http://www.mindrot.org/projects/softflowd/`

Table 1. Overview of timing errors for each source on each router for the examined week.

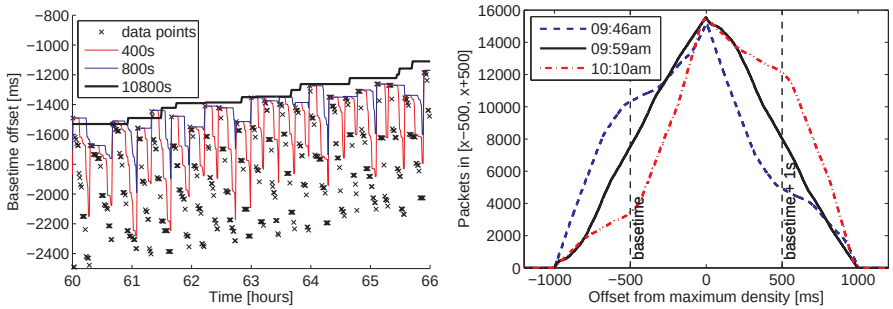
Router	Source	Drift [ms/day]		Error [ms]				Rate [s^{-1}]		Correction
		mean	std.	max	95th	5th	min	mean	std.	interval [s]
A	0	-423	14.5	+344	-72	-972	-1052	0.24	0.03	10800
A	513	-228	17.0	+136	-56	-960	-1960	193.9	79.5	400
A	518	-423	7.9	+208	-56	-976	-1876	157.4	39.2	400
B	0	-2039	355.4	+280	-124	-1068	-1465	0.005	0.0008	10800
B	513	+560	41.8	+28	-48	-988	-1020	314.7	64.5	400
B	517	+81	35.4	+244	-52	-992	-1592	201.2	41.7	400
C	0	+1543	49.9	+256	+52	-848	-972	0.09	0.008	10800
C	518	+1577	70	+428	-48	-992	-1824	74.2	29.7	400
D	0	+1053	37.9	+112	+12	-924	-984	0.12	0.02	10800
D	517	+1055	21.5	+316	-52	-946	-1824	302.9	100.7	400
E	0	+239	38.3	+60	-28	-928	-1012	0.11	0.02	10800
E	513	+453	8.9	+500	-48	-952	-1600	204.6	60.8	400
E	515	-17	21.7	+280	-52	-968	-2000	333.5	101.9	400
F	0	+47	14.9	+176	-40	-936	-1044	0.07	0.005	10800
F	513	+46	20.5	+88	-48	-948	-1328	15.6	7.0	10800
`softflowd`		+5.5	12.0	+16	-43	-940	-1001	0.61	0.02	10800

4 Correcting Cyclic Timing Error

Having observed and quantified this error, we set out to devise a method for correcting it. Since the basetime is related to the time at which the router started, and router restarts in production networks are relatively rare events, correct basetime information could be determined out-of-band via the router's management interface (e.g., SNMP or the command line). However, this method would have two disadvantages. First, it requires the management interface of the router to be accessible to the measurement infrastructure, which is not always desirable. Second, static out-of-band basetime determination ignores the drift of the realtime clock, which is included in each of the flow timestamps; this error would then need to be corrected in any event.

Therefore, we focused on generating a *corrected basetime* estimating the true basetime from the derived basetime information. Our first attempt at this consisted of a simple robust maximum detector. The primary problem with this method is it requires a rather high packet density; otherwise it has a tendency to "follow" downward-cycling derived basetimes into the cyclic error band. The problem also initially appeared to be suited to simple linear regression, but the widely variable density of derived basetimes within the cyclic error band ruled this method out.

We therefore settled on a correction mechanism based on sliding density windows. Recalling the density diagram in Figure 2, we first take the set of derived basetimes for a specified "horizontal" (export time) interval, called the *correction interval*. We then slide a one-second "vertical" (derived basetime) interval over the correction interval, and select the position for this interval which maximizes the derived basetime density within the rectangular correction window. The top (maximum derived basetime) of this window is then taken to be the corrected basetime. For source 513 on router A, the corrected basetime is shown as the top line in Figure 2.



(a) Stability of corrected basetime varies with correction interval size

(b) Density distributions within the correction window vary over time

Fig. 3. Illustrating density-window basetime correction

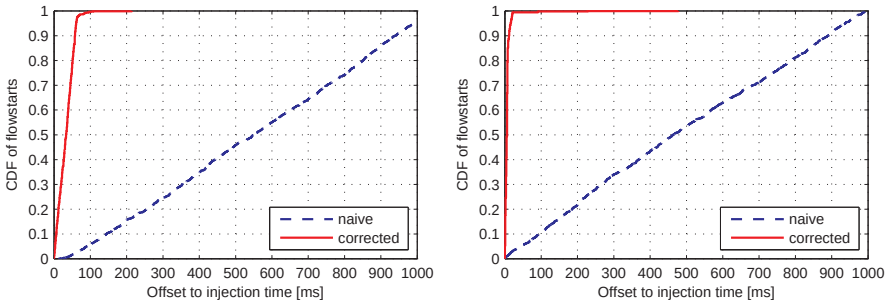
The correction interval is selected based upon the density of derived basetimes for a given source, which is itself dependent on the traffic volume. In principle, it should be chosen to observe at least several wraparounds of the cyclic error. Figure 3(a) shows the effect that different correction intervals have on the corrected basetime series, observing the effects of three different windows on correcting the relatively low-density source 0 of router C. In general, longer correction intervals provide more stable and therefore more accurate corrections, but require more processing as they must consider more data points, and would lead to longer delays in reporting corrected times during stream processing. For this study, we selected a 400 second window for higher density sources, and a 10800 second (three hour) window for lower density sources.

Figure 3(b) illustrates how this correction method works in the presence of variable density of derived basetimes. Here we show the derived basetime density as a function of the position of the vertical interval for three different correction windows. Even though the density distributions differ greatly, the method leads to the same basetime correction.

4.1 Evaluation

To measure the effective accuracy of the cyclic error correction method, we compared exported flow timestamps to known flow timing. We placed a traffic-generating host on the measured network to send single-packet UDP flows to known hosts outside the network, chosen such that these flows would be routed across a known source on a known router in our collection infrastructure. We saved the injection time for each flow key, and compared this to the timestamps on the flows exported via NetFlow, both with uncorrected derived basetimes as well as basetimes corrected using the method described above.

The CDF of the deviation from known timing of per-flow timestamps over 30 hours of data over 3-4 September 2010 for source 513 on router A are shown in Figure 4(a).



(a) Cisco (b) softflowd

Fig. 4. Corrected and uncorrected flow start times

Here, we see that the flow timestamps calculated from uncorrected basetimes exhibit a uniform deviation to about 1s, caused mainly by the cyclic error in the basetime. If the cyclic error were the only remaining error in the data, we would expect the flow timestamps calculated using the corrected basetimes to exhibit no more than a few milliseconds of error to account for the one-way delay between the source and the router at which the flows were measured.

We see this in the `softflowd` data, as shown in 4(b). This demonstrates that our cyclic error correction method can completely eliminate the timing error introduced by the design of the NetFlow v9 protocol. However, even after applying corrected basetimes, there is an additional source of about 70ms of uniformly distributed error on Cisco routers. Further investigation shows this to be constant across sources and routers, and independent of time, load, or other factors measured in this work. Therefore, we conclude that this error is implementation-specific and an unavoidable property of the packet or flow handling of this specific implementation, either within the metering process or between the metering and exporting processes.

5 Related Work

The question of the fidelity of data used in Internet measurement studies is well-addressed in the literature. The effects of sampling of packets (e.g., as in [4]) as well as flows (e.g., as in [5]) have been widely studied. However, these works tend to be theoretical, focusing more on the mathematical properties of the techniques used and the essential tradeoff between overhead and utility than on the effects of specific implementations or protocols used in the collection of the data used.

Sommer and Feldmann [6] examined the information loss associated with flow measurement as opposed to directly operating on packet data, and find that for one particular application, TCP connection summary generation, flow data suffices "[using] large enough time intervals": an acknowledgement of the impact that flow timeouts and timing in general have on later analysis.

Paxson [7], in establishing a set of best practices for Internet measurement studies, classifies sources of error into precision, accuracy, and misconception. It discusses timing precision and accuracy and discusses the need to consider and calibrate for measurement infrastructure induced error in source data sets. It advocates the export of metadata along with source data for measurement studies, a call we reiterate in our conclusions.

A closer antecedent for the present work is Cunha et. al. [8], which provides a similar study of largely implementation-related artifacts in flow data produced by the Juniper routers which generated commonly-used datasets from Abilene and GEANT. These artifacts were related to timeout and flow cache expiry, and as such have a destructive impact on the distribution of flow duration.

The IETF addressed various design issues with NetFlow v9 in the specification of the IPFIX protocol [9]. Crucially, IPFIX supports flexible timestamps from second to nanosecond resolution, and allows the association of an absolute timestamp with every flow. As it does not mandate the export of potentially inconsistent timestamps in each message, it does not suffer from the cyclic error we present in this work. However, it does not necessarily address other sources of inaccuracy within the implementation of the metering or exporting process.

6 Conclusions

In seeking to maximize the timing precision available from data exported via Cisco NetFlow v9, we discovered and quantified a cyclic source of up to one second of error in flow timestamps, inherent in the design of the protocol. Correcting this cyclic error can improve the accuracy of NetFlow v9 data to millisecond-level. However, inaccuracy remains within the examined Cisco implementation which we do not have sufficient information to correct, limiting our effective correction for our production dataset to one order of magnitude, for about 70ms accuracy.

The set of routers from which we receive NetFlow v9 data represents an admittedly small sample of deployed implementations. However, the cyclic error is a protocol issue. It is therefore implementation-independent, and should affect NetFlow v9 export from any vendor. We note that an implementation built with an awareness of the cyclic error could avoid it, by faking the system uptime and/or export timestamps in order to export real basetimes, but we did not observe this behavior in any examined NetFlow implementation.

In addition, we presume that similarities in NetFlow v9 metering and export process implementations could lead to implementation-specific sources of error similar to those we observed on Cisco devices on implementations from other vendors. These measurements are an area for future work. The guidance to take from our work in any case is this: researchers using NetFlow v9 data sets should not assume better than second-level accuracy unless employing a method for correcting cyclic basetime error such as the one we present here, and should measure the residual error specific to their metering and exporting processes.

In this work, we were able to observe and correct timing error from the Netflow v9 export packet headers which is not apparent from an examination of flow data alone. This leads us to reiterate the call in [7] to export and maintain implementation-specific metadata alongside flow data used for research. Our experience in this work additionally indicates the wisdom of keeping measurement data in as "raw" a form as possible. While all flow data is theoretically the same, and should be freely convertible among formats, this is not the case in practice: as we have shown, implementation matters.

7 Acknowledgments

The authors would like to acknowledge SWITCH for providing the data used in this study. Thanks as well to the European Commission for its material support of this work through the DEMONS project (FP7-257315).

References

1. Claise, B., Sadasivan, G., Valluri, V., Djernaes, M.: Cisco Systems NetFlow Services Export Version 9. RFC 3954 (Informational) (October 2004)
2. Trammell, B., Boschi, E.: Bidirectional Flow Export Using IP Flow Information Export (IPFIX). RFC 5103 (Proposed Standard) (January 2008)
3. Sadasivan, G., Brownlee, N., Claise, B., Quittek, J.: Architecture for IP Flow Information Export. RFC 5470 (Informational) (March 2009)
4. Brauckhoff, D., Salamatian, K., May, M.: A signal processing view on packet sampling and anomaly detection. In: Proceedings of INFOCOM 2010, San Diego, California, USA, IEEE (2010) 713–721
5. Choi, B.Y., Bhattacharyya, S.: Observations on Cisco sampled NetFlow. SIGMETRICS Perform. Eval. Rev. **33**(3) (2005) 18–23
6. Sommer, R., Feldmann, A.: Netflow: information loss or win? In: Proceedings of IMW 2002, Marseille, France, ACM (2002) 173–174
7. Paxson, V.: Strategies for sound internet measurement. In: Proceedings of IMC 2004, Taormina, Sicily, Italy, ACM (2004) 263–271
8. Cunha, I., Silveira, F., Oliveira, R., Teixeira, R., Diot, C.: Uncovering artifacts of flow measurement tools. In: Passive And Active Measurement Conference 2009, Seoul, South Korea (March 2009)
9. Claise, B.: Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of IP Traffic Flow Information. RFC 5101 (Proposed Standard) (January 2008)