One-way Delay Measurement based on Flow Data: Quantification and Compensation of Errors by Exporter Profiling

Jochen Kögel
Institute of Communication Networks and Computer Engineering (IKR)
University of Stuttgart
Stuttgart, Germany
Email: jochen.koegel@ikr.uni-stuttgart.de

***Abstract*—One-way delay (OWD) is an important measurand for network management. It can indicate routing problems, network congestion, and is useful for tracking down application problems to network effects. While active measurements deliver high accuracy, measuring OWD actively on all paths of large networks results in high effort. Thus, passive measurements are attractive. Our approach for measuring OWD is based on flow data, which is often exported from flow capturing enabled routers (exporters) for traffic accounting and reporting. Thus, this approach does not require additional network components and often comes at almost no additional cost.**

It is well-known that flow data is often inaccurate and incomplete due to record loss. However, only little information is available on timestamp accuracy of flow data. Therefore, we investigated timestamp errors and developed a method for quantifying them in order to improve the accuracy of flow capturing based OWD measurements.

Our contribution is threefold: First, we analyze clock resolution effects based on a reference model for flow record creation. Second, we develop methods for exporter profiling, i.e., extracting errors introduced by exporters from flow data. Third, we present results obtained from data collected in a global enterprise network.

We conclude that precision of flow data based OWD calculation heavily depends on the capturing devices, especially on timestamp resolution. Standard deviations observed range from 2.07 ms to 34.02 ms.

***Keywords*-network monitoring; flow capturing; IPFIX; NetFlow; one-way delay**

I. Introduction

More and more business-critical services are based on the Internet or on global enterprise networks. Thus, network monitoring and early problem detection become critical tasks. An important characteristic of networks is the one-way delay (OWD) measured between any two points in the network, e.g. between endpoints or between network elements.

OWD is composed of propagation delay on the link (d_l), processing delay (d_p), and queuing delay (d_q). While d_l and d_p stay mostly constant for the same network path, d_q strongly depends on traffic load and queue levels. Therefore, changes of the OWD indicate effects and problems on the network layer, such as routing problems or network congestion.

Our work focuses on OWD measurement in global enterprise networks that connect a high number of branch locations via edge routers to a core network. In most cases, the core network belongs to an MPLS provider and its structure and properties are unknown. Therefore, taking into account topology for inferring OWDs is not possible, but all paths between edge nodes have to be considered for measurement. OWD in such networks can be measured in an active or passive manner. Active measurements send probe packets between different points in the network, e.g. from edge router to edge router. In contrast, passive measurements do not inject additional traffic.

An overview of active OWD measurements and especially clock synchronization issues is given in [1], while a probe-based measurement architecture is presented in [2]. For getting OWD values between any points of the network with a suitable amount of samples, probe traffic has to be sent between any two locations in both directions, which leads to a high effort. Thus, passive measurement approaches, which rely on performing measurements on existing traffic, are attractive . A passive approach on packet level is presented in [3]. Here, timestamps of the same packet from two different locations are compared, which requires, however, additional probe equipment for traffic capturing and packet ID generation.

Our approach does not work on packet level but on flow data, which is monitoring data exported using protocols like NetFlow [4] or the IPFIX protocol [5]. The advantage of this approach is that flow capturing and flow data export is implemented in most routers. Thus, no additional monitoring equipment is required. Additionally, in most well managed networks flow data is already collected from many routers for reporting, accounting, and anomaly detection purposes. Flow data based OWD measurement therefore comes at almost no additional cost.

Obviously, in order to know which accuracy can be achieved by flow data based OWD measurement, knowing errors of flow data timestamps is essential. Flow data export protocols specify timestamps for flow start and end times with millisecond or even higher resolution. However, it has not been investigated so far which timestamp accuracy the flow capturing devices eventually provide. There is related work on achievable accuracy for flow data based packet loss [6] and also the idea of trajectory sampling [7] can provide input data for flow-based OWD measurements. [8] describes effects of timeout mechanisms, however timestamp accuracy is not considered. Passive OWD measurement on packet level using the IPFIX protocol is described in [9] and [10]. However, this approach is not based on flow capturing, but the IPFIX

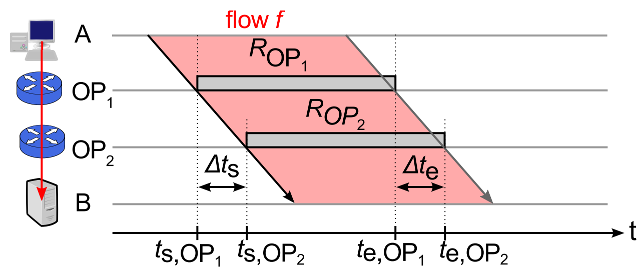

Figure 1. Terminology

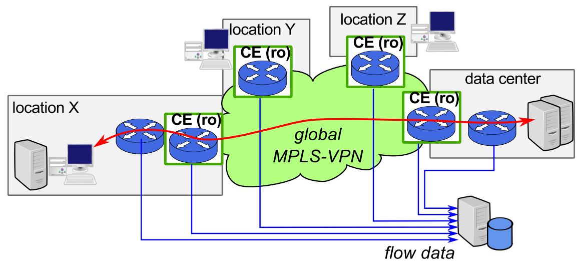

Figure 2. Flow capturing in a global enterprise scenario

protocol is used to transfer packet IDs.

Our work is not only the first that investigates timestamp errors of flow data, but it also provides a systematic and universal approach to determine the accuracy of flow capturing devices deployed in a global network. Our goal is to use as little additional knowledge and measurements as possible, i.e. we do not take knowledge on router models, configuration or topology into account. The main idea is to generate exporter profiles offline based on flow data from periods when the network is lightly loaded, i.e., d_q is negligible. These profiles specify accuracy parameters in terms of systematic and random errors. A system that extracts OWD samples from flow data shortly after it is exported (online) can then use these profiles for compensating the systematic errors. Additionally, with profile support the impact on accuracy from random errors can be quantified, e.g. for determining confidence bounds when averaging over several samples.

This paper is structured as follows. Sec. II introduces flow capturing terminology and traffic observation points in an enterprise network scenario, while Sec. III focuses on timestamp effects and their impact. Sec. IV presents the exporter profile creation. Sec. V shows results in terms of errors for OWD measurement between routers and Sec. VI concludes the paper.

II. Flow Capturing in Global Networks

A. Terminology

We use a concise flow definition, which is sufficient for timing effects of our studies. This definition is close to NetFlow v5 [4] and not as general as IPFIX terminology [11]. In our definition, a flow is a unidirectional unicast data transmission between a source endpoint A and a destination endpoint B on the transport layer (see Fig. 1). A and B are defined by their IP addresses, their port numbers and the transport protocol of both, i.e. this five tuple defines endpoints and direction. The *flow timestamps* t^*_s and t^*_e denote the flow's start and end at an observation point *OP*. A flow f can consist of multiple subflows f_s, which share the same A and B, but which can be distinguished further, e.g. by the Type-of-Service (ToS) byte.

We consider flow data that is captured at OPs of a flow capturing enabled router (exporter E). Therefore, we define an OP as the tuple formed by exporter E and input interface I. A *flow record* R is a data record that describes a subflow f_s as seen by the flow capturing device at an OP. t_s and t_e are the *record timestamps*. Further attributes that are relevant to us are the byte and packet counter b and p along with the previously mentioned flow attributes, i.e. $R := (f, OP, t_s, t_e, p, b, ToS)$. R can describe a complete flow or only a fraction of a flow measured in a certain time interval. The latter happens, e.g. if for a flow there are no packets observed for a certain time and timeout mechanisms trigger record export. If there is only one record for a flow, we call it a one-record-flow f_or. In this case $(t^*_\mathrm{s} = t_\mathrm{s}) \wedge (t^*_\mathrm{e} = t_\mathrm{e})$ holds. Often, there is bidirectional communication (e.g. TCP connections) where a reverse flow f_rv to a forward flow f_fw exists. The endpoint tuple is switched $(A_\mathrm{rv} = B_\mathrm{fw} \wedge B_\mathrm{rv} = A_\mathrm{fw})$ and the timing must be accordingly: $t_\mathrm{s,fw} < t_\mathrm{s,rv}$.

If the flow traverses an Observation Point Pair (*OPP*) of two OPs OP_1 and OP_2 as in Fig. 1, we can calculate the packet and byte differences with $\Delta b = b_{OP_2} - b_{OP_1}$, $\Delta p = p_{OP_2} - p_{OP_1}$. Accordingly, the start and end time differences are $\Delta t_s = t_{s,OP_2} - t_{s,OP_1}$ and $\Delta t_e = t_{e,OP_2} - t_{e,OP_1}$. If there is no packet loss and clocks are correct, Δt_s and Δt_e of an f_or are OWD samples of the *OPP*.

B. Scenario and observation points

Fig. 2 shows a typical deployment of flow capturing in a global enterprise scenario. Data centers and branch locations are connected via a global core network (MPLS VPN). The core network belongs to a carrier, which also operates the customer edge (CE) routers. Enterprise administrators have read-only (ro) access on CE routers. The CE routers as well as the routers operated by the enterprise (e.g. in data centers) send flow data to a flow data collector, from where data is taken for processing. There are two important facts in this scenario: First, flows typically cross more than one exporter. Second, there are different domains involved, which can also lead to different unsynchronized time domains.

Depending on the routing scheme employed and router/network configuration, several special properties have to be considered. First, load balancing schemes (ECMP) or asymmetric routing can happen, which leads to situations where a flow takes different paths or the reverse flow does not cross the same exporter as the forward flow. Furthermore, there are also different implementations of flow capturing mechanisms that often exist in parallel on the same exporter. Here, we can distinguish between software and hardware based implementations that can reside on line cards or on central units. Thus, it is possible that there are several OP of different kind within the same router. In most scenarios it depends on the ingress interface and routing at which OP flows are captured. Since characteristics of the implementations can differ, we have to build profiles for every OP and OPP separately.

In most cases flow capturing happens in processing stages where also forwarding decisions are taken. This is close to the ingress interface, i.e., only after passing the OP packets enter queues, where they can be subject to contention and experience d_q. If flow capturing is enabled on the egress path, the respective OP can be located after some queues. However, this egress capturing can be detected based on a direction field in the records.

III. TIMESTAMP EFFECTS

A. Timestamps in flow data

Fig. 3 shows our reference model for flow capturing with the blocks metering and exporting. The metering block takes the packets and updates flow entries in the flow cache based on the key (in NetFlow v5 the five tuple, ToS and input interface). Based on timer configurations, flow cache entries expire at t_r and are forwarded to the exporting block, where several flow records are put into a UDP packet and exported at t_x to a collector.

As our focus is on timestamps, their creation is illustrated in more detail in Fig. 3. There is a system clock $s(t)$ that counts the system uptime in milliseconds and a real time clock (RTC) $u(t)$ that holds the UNIX time in two 32 bit values for seconds and nanoseconds. The two clocks are used to create the *raw timestamps* (r_x) that are sent in flow data packets and from which t_s and t_e are calculated. The metering process uses the uptime counter only and writes r_first on record creation and updates r_last for every packet. When the records are finally exported, the values r_su , r_sec and r_nsec are written into the packet header. The record timestamps are calculated based on the boot time t_b, as illustrated on the bottom of Fig. 3:
$t_\mathrm{b} = r_\mathrm{sec} \cdot 1000 + \frac{r_\mathrm{nsec}}{1e6} - r_\mathrm{su}$; $t_\mathrm{s} = t_\mathrm{b} + r_\mathrm{first}$; $t_\mathrm{e} = t_\mathrm{b} + r_\mathrm{last}$.
$s(t)$ and $u(t)$ are typically not synchronized ($d(t)$ in Fig. 3), which can lead to errors of more than one millisecond for records with long durations. Due to space limitations, this effect is not detailed further.

B. Raw timestamp resolution

In Fig. 3 asterisks indicate points where a limited timestamp resolution (i.e. limited granularity of values) can be caused, e.g. if lower bits of a 32-bit value are dropped and/or internal timestamp values are converted in certain ways. This leads to a limited resolutions ρ_x of raw timestamps r_x and record timestamps t_x.

Characteristic values for raw timestamp resolutions are shown in Tab. I a) for three exemplary exporter types. For all exporter types, ρ_secs has a resolution of one. ρ_nsecs is 15,258 on most exporters. We assume this is the resolution RTC, since clock components often divide a second into 2^{16} ticks (15,258 is $\frac{1e9}{2^{16}}$). However, this resolution is below milliseconds and thus does not impact ρ_s or ρ_e.

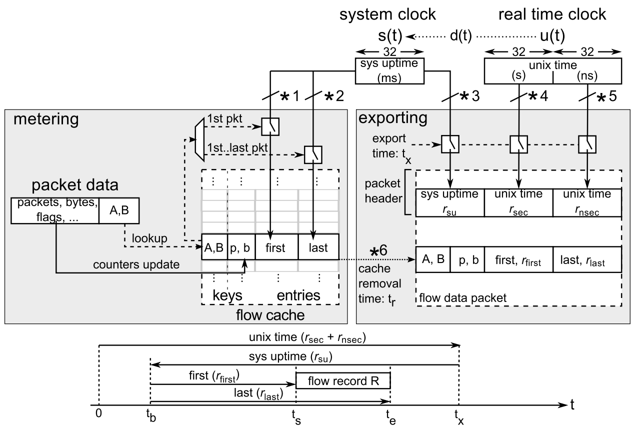

Figure 3. Reference model for record timestamp creation

Table I
EXEMPLARY TIMESTAMP RESOLUTIONS

a) Observed raw timestamp (r_x) resolutions (ρ_x)

OP type	ρ_secs	ρ_nsecs	ρ_su	ρ_first	ρ_last
1	1	15,258	4	4	4
2	1	15,258	2	1	1
3	1	15,258	1	1	1

b) Observed record timestamp (t_x) resolutions (ρ_x)

OP type	ρ_s	$\rho_\mathrm{s,IAT}$	ρ_e	$\rho_\mathrm{e,IAT}$	ρ_d
1	4	4	4	4	4
2	1	64	1	64	64
3	1	1	1	1	1

On exporters of type 1, ρ_su, ρ_first, and ρ_last are four (lower two bits always zero), while for type 2 only ρ_su has a limited resolution. Exporters of type 1 are e.g. Cisco 7200 series and type 2 are e.g. Cisco 6500 series performing hardware-based (MLS) NetFlow. Exporters of type 3 do not have a limited resolutions, except in ρ_nsecs.

C. Record timestamp resolution

A first approach for calculating the resolutions of start/end timestamp (ρ_s, ρ_e), and record duration (ρ_d) would be: $\rho_s = max(\rho_\mathrm{su}, \rho_\mathrm{first}, \rho_{t_\mathrm{x}})$,
$\rho_e = max(\rho_\mathrm{su}, \rho_\mathrm{last}, \rho_{t_\mathrm{x}})$, and $\rho_d = max(\rho_s, \rho_e)$, with $\rho_{t_\mathrm{x}} = max(\rho_\mathrm{uS} \cdot 1000, \frac{\rho_\mathrm{uN}}{1e6})$.

Tab. I b) shows observed resolution values of record timestamps. Here, we distinguish between resolution values we calculated by checking each record timestamp value separately and by checking the time between two values. The latter can be seen as the record IAT, thus it is denoted with this suffix.

Exporters of type 1 and type 3 allow calculating the resolution values as given by the previous equations. For Type 2 exporters, however, the equations cannot explain ρ_d. Additionally, ρ_s and ρ_e differ from $\rho_\mathrm{s,IAT}$ and $\rho_\mathrm{e,IAT}$. This lead us to the assumption that type 2 exporters have an internal resolution of 64 ms, but the the lower 6 bits of `first` and `last` are simply not set to zero at point *6 of Fig. 3. Indeed, we found that the lower 6 bits of type 2 exporters plotted against the export time roughly describe a saw tooth that seems to depend on $d(t)$, while the value of these bits with other exporter types are randomly distributed. Thus, profiling OP timestamp resolution cannot stop at evaluating raw timestamp resolution, but we have to calculate ρ_s, ρ_e, and especially $\rho_{\mathrm{s,IAT}}$, and $\rho_{\mathrm{e,IAT}}$.

D. Impact on OWD calculation and compensation

Limited timestamp resolution leads to timestamp errors with certain bias and thus impacts the OWD calculated. The error from limited timestamp resolution is a uniform PDF $s(x)$ of width ρ that has the standard deviation $\sigma = \sqrt{\frac{\rho^2}{12}}$. From timestamp creation, it is obvious that the timestamp resolution is limited in a way that the last bits are dropped. This additionally results in a bias since the reported timestamp will always be equal to or less than the real time. Thus, the bias will be $\beta = -\frac{\rho}{2}$, while in general β could be in $[-\frac{\rho}{2}; \frac{\rho}{2}]$ if timestamp values are created differently. In general, the PDF of resolution-caused errors is

$$s(x) = \begin{cases} \frac{1}{\rho} & -\rho/2 - \beta < x < \rho/2 - \beta \\ 0 & otherwise \end{cases} \quad (1)$$

If we calculate Δt_{s} or Δt_{e} between two OPs with resolutions ρ_1 and ρ_2, we get a PDF of the total error as convolution of two uniform PDFs:

$$s_{OPP}(x) = s_{OP1}(x) * s_{OP2}(x) \quad (2)$$

If both OPs have the same ρ, a triangular distribution of width $2 \cdot \rho$ results. With different ρ, we will get a trapezoidal distribution with the width $\rho_1 + \rho_2$. The standard deviation is calculated by adding the variances of s_{OP1} and s_{OP2}:

$$\sigma_{OPP}(x) = \sqrt{\frac{\rho_1^2 + \rho_2^2}{12}}. \quad (3)$$

Δt_s will be impacted on average by the bias as

$$\Delta t_s = t_{OP2} - t_{OP1} + (\beta_{\mathrm{OP2}} - \beta_{\mathrm{OP1}}) \quad (4)$$

This means we have to compensate the bias in order to get correct OWD mean values.

IV. Exporter profiling

A. Profiling Overview

For measuring OWD based on flow data, we determine accuracy profiles of exporters that describe systematic and random errors concerning timestamps. There are several possibilities for obtaining the exporter profile, e.g. knowledge about router implementations and configuration, lab measurements or comparing flow data with active measurements. However, relying on such knowledge requires reliable updates when changes happen (e.g. router software updates) and it is hard to design automated methods. Thus, we focus on automated methods for obtaining the exporter profile with as little additional knowledge as necessary, i.e. relying on flow data only.

We perform the accuracy profiling on data sets from time intervals when the network is lightly loaded (e.g. at night, on weekends). Profiling itself is not time critical and can be done offline and iteratively. Here, also resource consuming algorithms are feasible. In contrast, the OWD calculation that uses the profiles should deliver the OWD samples from the network as quickly as possible for timely performance and failure management and thus has to process the flow data online.

Table II
Exporter profile

field	content
`clk_offset`	clock offset to ω_{ref} in ms
`clk_skew`	skew in ms/h
`ref_exp`	OP_{ref} for this exporter
Per OP of this exporter	
ρ_{s}	resolution of start timestamp
ρ_{e}	resolution of end timestamp
β_{s}	bias of start timestamp
β_{e}	bias of end timestamp
Per OPP where an OP of this exporter is involved	
`sdistr`	accuracy: distribution of Δt_s error
`edistr`	accuracy: distribution of Δt_e error

The exporter profile format is given in Table II. The first part covers clock offset and clock skew detected compared to a reference exporter. This is necessary in order to compensate for clock inaccuracies, since the exporters might be configured to different time domains or not be synchronized at all. The second part of the profile deals with the limited time resolution. This part exists for every observation point OP of this exporter. In the third part of the profile, the accuracy for every OPP, in which this exporter is involved, given as distribution of the Δt_s and Δt_e error. The latter is important for validating the bias calculated, capturing errors not caused from resolution effects, and for detecting non-plausible error distributions.

Our approach is detailed in the next subsection and is currently limited to creating exporter profiles of networks where no flow independent load balancing is employed, only unsampled flow capturing is performed and with predominantly symmetric routing.

B. Profiling Steps

The exporter profile described in the previous section is created using a multi step process illustrated in Fig 4. All steps take the same flow data set as input and determine certain fields of the exporter profiles, which they write into the exporter profile data base. Each step is performed after the previous step is finished, as steps partly rely on information in the exporter profile written by previous steps. Some steps require preprocessing steps that are indicated in the left part of Fig. 4.

The first step is the simplest one. It identifies every exporter and OP in the flow data and instantiates the required exporter profiles. In the second step, the timestamp resolution for each OP is detected in terms of the ρ_{x} presented before. The third step infers all OPPs where an OP of the considered exporter is involved. The fourth and fifth steps require mainly the same preprocessing blocks, since they consider differences of timestamps and packet/byte counters of the OPP. While the fourth step takes Δt_{s} between the exporter considered and a reference

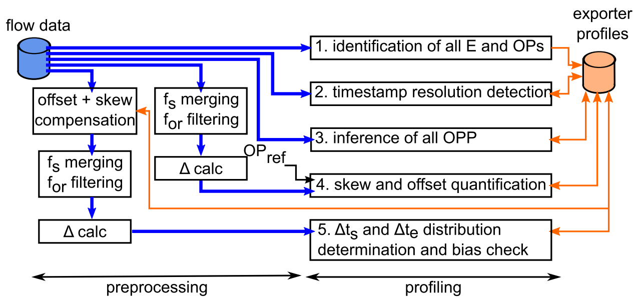
flow data
exporter profiles
1. identification of all E and OPs
2. timestamp resolution detection
3. inference of all OPP
4. skew and offset quantification
5. Δt_s and Δt_e distribution determination and bias check
offset + skew compensation
f_s merging f_{or} filtering
f_s merging f_{or} filtering
Δ calc
Δ calc
OP_{ref}
preprocessing
profiling

Figure 4. Profiling steps

exporter OP_{ref} to determine clock offset and skew, the fifth step takes Δt_{s} of values where clock offset and skew are already compensated. Both steps check for packet loss ($\Delta p = 0$) before considering the corresponding Δt_{s} and Δt_{e} values. If there is packet loss detected, we cannot use the flow records for OWD calculation, since Δt_{s} or Δt_{e} could be affected if the first or last packet is lost. Checking for byte consistency ($\Delta b = 0$) could improve reliability further. However, we found that there are systematic byte count differences between exporters: some exporters report the Ethernet payload length, while others report the IP packet size. This leads to byte count errors with an OPP for every packet smaller than 46 Bytes. Therefore, checking for $\Delta b = 0$ is not feasible on OPPs where OPs handle the byte count differently. Hence, we tolerate a certain deviation of Δb in such cases. As indicated, steps four and five require Δt_{s} and Δp, which are both calculated in the preprocessing block Δ calc. This block keeps flow records for a certain time window in memory and matches records from different OPs based on the five tuple. In order for this block to work reliably, only one record flows (f_{or}) can be considered. Also the five-tuple based matching does not work for subflows f_{s}, which might even have different packet count between two OP due to ToS Byte change from DiffServ Policing or ECN. To get around these two problems, we employ a preprocessing block that merges records of several f_{s} into a single R and allows only f_{or}.

For OWD calculation and clock offset detection, it is important to know the pairs of OPs between two exporters. Especially, it is important to know the OP ordering, i.e. in which order they are traversed by packets. Else negative OWD values for an OPP result and clock offsets cannot be determined. For determining which OP of an OPP is first and which is second, we observe the timing of forward and reverse flows between two exporters exploiting the nature of request-response protocols and TCP handshakes. Therefore, our algorithm works if at least for some flows there is symmetric routing (forward flow f_{fw} and reverse flow f_{rv} both seen on the same Exporter).

In order to determine the OP ordering, we search for reverse flows for a given flow. Then we calculate the RTT for every exporter where forward and reverse flow are seen. The exporters closest to the initiator see the largest RTT, while the exporters closest to the responder see the lowest RTT. Due to limited timestamp resolution, a single RTT difference sample does not always indicate the correct ordering. Thus, we collect for each OPP several RTT difference samples and decide at the end of the profiling step, which order the OPs of this OPP have.

Table III
PERCENTILES AND STANDARD DEVIATION OF DISTRIBUTIONS [ms]

case	ρ_a	ρ_b	2.5%	50 %	97.5%	σ	s
a	64	4	55	87	120	18.51	22.82
b	4	64	-10	24	57	18.51	22.99
c	64	64	51	105	158	26.13	34.02
d	4	4	-3	1	5	1.63	2.07

Finally, the last profiling step determines the error distributions and thus the precision of OWD measurement for each OPP. This step takes Δt_s and Δt_e where $\Delta p = 0$, Δb is tolerable and calculates histograms of for each OPPs by using 601 buckets for Δt_s and Δt_e values between -300 and +300 ms. The resulting sample counters for each bucket as well as overflow and underflow counters are stored as part of the exporter profile.

Based on the distribution obtained, the system can validate the bias calculated and eventually determine the precision of OWD measurement for an OPP. Additionally, non-plausible distributions can be detected that cannot be handled using the exporter profile.

V. EVALUATION

A. *Processing framework and scenario*

We implemented the exporter profiling based on a Java framework of modular processing blocks [12]. This allows us to chain processing blocks for each profiling step as needed. Exporter profiles are stored at the end of a processing step and loaded before the next step.

Input for our evaluations is data that was collected in an enterprise network where several hundred routers export NetFlow v5. We collected the flow data and wrote them to files using the flow-capture daemon of the flowtools package. Our prototype creates exporter profiles for several hundred exporters from flow data of one day (33 GB) within less than two hours. Currently, we do not use data from exporters where we detected clock offset or skew during exporter profiling.

B. *Results*

Fig. 5 shows the distribution of Δt_{s} values for four different OPPs. We selected four typical OPPs with different ρ_s in order to show the distributions of errors. Bias compensation has not yet been performed for these charts. Table III gives the percentiles that are also indicated by red and green lines in the charts of Fig. 5. Additionally, the table presents the standard deviations (σ) calculated from ρ as well as the standard deviation calculated from the distribution (s).

First, we can see that the distributions follow a trapezoid or triangle shape, which is a result from convolution of two uniform distributions of the resolution-based error (c.f. Sec. III-D). Additionally, there is some random error, since the slopes are not sharp and $s > \sigma$. We can also observe that the error distribution is almost symmetric and

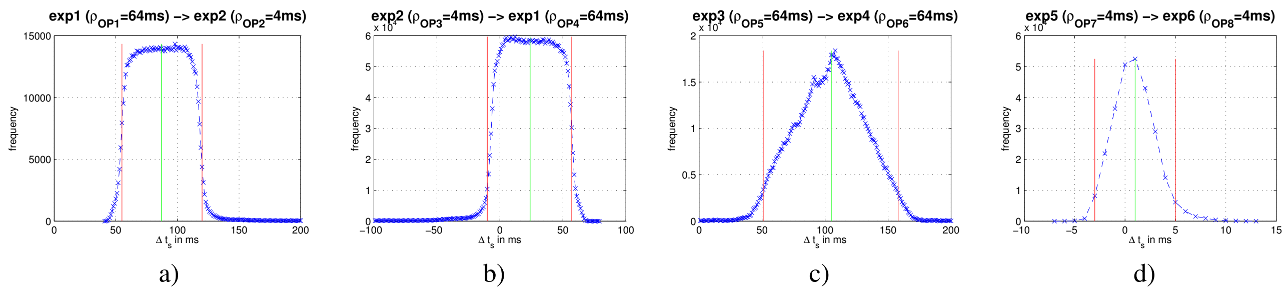

Figure 5. Distribution of Δt_{s} values (OPP precision) for different cases of ρ combinations

the distributions could be approximated as convolutions of resolution-caused errors and a normal distribution. The symmetric shape also indicates that we did the profiling in a lightly loaded network. Else, d_q would deform the distribution towards higher Δt_{s}.

Fig. 5 a) and Fig. 5 b) show the error distribution of two OPPs of a symmetric path between two exporters with different ρ_s. Here we can clearly see the impact of the bias denoted in Equation 4. In this case, we can conclude that the bias calculation of +/- 30 ms for this OPP is correct. This does not perfectly match the differences of the percentiles for this OPP, but is very close. A reason for this deviation might also be the high number of outliers of one OPP, which can be observed in Fig. 5 a) above 130 ms and in Fig. 5 b) below -20 ms.

For an OPP where $\rho_a = \rho_b = 64$ ms we get the expected triangular distribution of Fig. 5 c). Such an OPP has a high error, but it can still be used for OWD calculation if a sufficient number of samples is collected. In contrast, the distribution of Fig. 5 d) shows that there are also OPPs with a very low error (standard deviation of 2.07 ms). This is a very good accuracy for OWD measurements in global networks, where OWDs d_l of more than 100 ms and d_q in the order of some 10 ms may occur.

VI. Conclusion

We presented a method to quantify the systematic and random errors for flow data based OWD calculation. This exporter profiling approach uses flow data as single source of information and generates exporter profiles offline using data from periods when the network is lightly loaded. Based on profile contents, such as clock resolution, offset, skew, bias, and error distribution, we are able to compensate systematic errors and quantify random errors introduced in timestamp creation. These compensation and accuracy quantification steps can then be performed with low large effort in a system that calculates OWD from flow data online.

The evaluation showed that timestamp errors in flow data are dominated by errors resulting from limited timestamp resolution. We conclude that using flow data for OWD measurement is feasible, if average OWD values of a certain time interval are considered, where a high amount of samples can compensate the measurement errors. Nevertheless, on one path we observed error distributions with a standard deviation of 2.07 ms, which makes flow based OWD measurements feasible for determining OWD on a per-sample basis.

References

[1] L. De Vito, S. Rapuano, and L. Tomaciello, "One-way delay measurement: State of the art," *Instrumentation and Measurement, IEEE Transactions on*, vol. 57, 2008.

[2] J. Corral, G. Texier, and L. Toutain, "End-to-end active measurement architecture in IP networks (SATURNE)," in *Proceedings of passive and active measurement network PAM'03, La Jolla, CA*, 2003.

[3] S. Niccolini, M. Molina, F. Raspall, and S. Tartarelli, "Design and implementation of a one way delay passive measurement system," in *Network Operations and Management Symposium, 2004. NOMS 2004. IEEE/IFIP*, vol. 1, 2004.

[4] "Netflow services and applications," White Paper, Cisco Systems, 1999.

[5] B. Claise, "Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of IP Traffic Flow Information," RFC 5101 (Proposed Standard), Internet Engineering Task Force, Jan. 2008.

[6] Y. Gu, L. Breslau, N. Duffield, and S. Sen, "On passive one-way loss measurements using sampled flow statistics," 2009.

[7] N. G. Duffield and M. Grossglauser, "Trajectory sampling for direct traffic observation," *IEEE/ACM Trans. Netw.*, vol. 9, 2001.

[8] I. Cunha, F. Silveira, R. Oliveira, R. Teixeira, and C. Diot, "Uncovering artifacts of flow measurement tools," in *PAM '09: Proceedings of the 10th International Conference on Passive and Active Network Measurement*, 2009.

[9] T. Zseby, E. Boschi, N. Brownlee, and B. Claise, "IP Flow Information Export (IPFIX) Applicability," RFC 5472 (Informational), Internet Engineering Task Force, Mar. 2009.

[10] F. Fatemipour and M. Yaghmaee, "Design and implementation of a monitoring system based on IPFIX protocol," 2007.

[11] J. Quittek, T. Zseby, B. Claise, and S. Zander, "Requirements for IP Flow Information Export (IPFIX)," RFC 3917 (Informational), Internet Engineering Task Force, Oct. 2004.

[12] J. Kögel and S. Scholz, "Processing of Flow Accounting Data in Java: Framework Design and Performance Evaluation," in *Proceedings of the 16th EUNICE/IFIP WG 6.6 Workshop*, 2010.