# The Effects of DDoS Attacks on Flow Monitoring Applications

Ramin Sadre, Anna Sperotto, and Aiko Pras

University of Twente
Design and Analysis of Communication Systems
The Netherlands
{r.sadre, a.sperotto, a.pras}@utwente.nl

***Abstract*—Flow-based monitoring has become a popular approach in many areas of network management. However, flow monitoring is, by design, susceptible to anomalies that generate a large number of flows, such as Distributed Denial-Of-Service attacks. This paper aims at getting a better understanding on how a flow monitoring application reacts to the presence of massive attacks. We analyze the performance of a flow monitoring application from the perspective of the flow data it has to process. We first identify the changes in the flow data caused by a massive attack and propose a simple queueing model that describes the behavior of the flow monitoring application. Secondly, we present a case study based on a real attack trace collected at the University of Twente and we analyze the performance of the flow monitoring application by means of simulation experiments. We conclude that the observed changes in the flow data might cause unwanted effects in monitoring applications. Furthermore, our results show that our model can help to parametrize and dimension flow-based monitoring systems.**

## I. Introduction

In the last decade, flows have become quite popular in IP network monitoring, since they help to cope with scalability issues due to the increasing network speeds and loads. A flow is defined as "a set of IP packets passing an observation point in the network during a certain time interval and having a set of common properties", usually known as *flow keys* [1]. Nowadays all major vendors offer flow-enabled devices, such as, for example, Cisco routers with Netflow [2], and the IETF is currently working on an IP flow standard, IPFIX. In recent years network flows have become an interesting approach also for other disciplines [3], as for example intrusion detection [4], [5]. Research in this field builds upon the advantages offered by network flows, such as, for example, the data reduction and the possibility to monitor large networks at high-speed.

A problem that might arise is that flow generation platforms, or *flow exporters*, are by design susceptible to resource exhaustion when dealing with anomalous data. An example is a Distributed Denial-of-Service (DDoS) attack creating an anomalous number of flows. To guarantee the monitoring functionality in such situations, flow exporters employ certain rules to manage their internal memory. In particular, if an increased number of flows are observed, the flow exporter might export *flow records*, as the snapshots of active flows are called, at a higher rate. In extreme cases, this can consequently cause problems to the system in charge to analyze such flows, as for example an intrusion detection system, with all the security implications related to this issue.

978-1-4673-0269-2/12/$31.00 

Existing literature has identified the problem of flow monitoring in case of massive attacks (see Section II). Research focuses on how to make the flow monitoring more scalable to attack traffic. However, it should be noted that the proposed solutions, such as sampling and data aggregation, can have as side effect the degradation of the performance of a detection system [6], [7]. Moreover, most of the existing, "ordinary" flow monitoring applications do not follow such sophisticated approaches and simply rely on the collected flow data.

This paper investigates how a flow monitoring application reacts to the presence of massive attacks. To the best of our knowledge, no other publication tackles this problem. We analyze the performance of a flow monitoring application from the perspective of the flow data it has to process. Our approach develops in two steps. First, we identify, with simple theoretical reasoning, the possible changes in the flow data as caused by a massive attack. The reasoning is extended to a simple queueing model that describes the behavior of the flow monitoring application. The strength of this paper is that it makes use of real traffic traces captured at the University of Twente (UT). As second step of our approach, we present a case study based on a DDoS attack targeting an IRC server on the university network. We analyze the performance of the flow monitoring application under the conditions imposed by the observed attack by means of trace-driven simulation experiments.

The paper is organized as follows. Section II describes related work of interest. Section III and IV respectively describe the considered scenario and discuss the impact of a DDoS attack on the flow data. In Section V, we describe the model of the monitoring application. We introduce our case study and present our simulation experiments in Section VI. Finally, we draw our conclusions in Section VII.

## II. Related Work

In the literature, only a few contributions explicitly discuss the security of flow-based detection systems [8]. Such solu-

tions focus on how the detection engine can be made resilient to attacks, for example by preventing uncontrolled growth of its internal memory components. Our contribution studies, on the other hand, the situations in which the stream of flows to be analyzed is able to overload the monitoring application.

Several publications have highlighted scalability problems in flow probes. For example, the work in [9] discusses how to design a performant probe. In [10], the authors clearly describe the performance problems a flow exporter can incur due to changes in the traffic mix, for example in case of a DoS attack. The authors propose sampling as a possibility to partially overcome cache performance issues. Several other sampling strategies have been introduced with the aim of taming the resource consumption in the flow exporter [11], [12]. McGlone *et al.* discuss the introduction of sampling mechanisms with respect to the resilience of the probe to DoS attacks. Differently from our contribution, the aforementioned works focus on the flow exporter. In our case, as we will discuss in Section V, we assume the flow exporter to be able to handle the amount of traffic and we concentrate on the performance of a flow monitoring application.

In the broad context of communication networks, we are not the first researchers employing queueing models to study the effects of DoS attacks. Wang *et al.* construct a queueing model with an underlying two-dimensional Markov chain in order to analyze connection buffers containing half-open TCP connections [13]. In [14], Long *et al.* propose queueing models to quantitatively investigate how DoS attacks degrade the performance of network-based control systems (NBCS). The authors of [15] analyze a queueing-based resource model under the, in our case study not valid, assumption of exponentially distributed inter-arrival times.

## III. The considered scenario

In this section, we describe the considered scenario. We begin with a description of the flow monitoring system in Section III-A. The DDoS attack is then discussed in Section III-B.

### A. The flow monitoring system

We consider a typical flow monitoring setup consisting of three components (see Figure 1). First, the flow exporter monitors network traffic on packet level and exports flow records via UDP packets (called "flow packets" in the following) to the flow collector, which is located on a separate network device. No sampling is applied to the monitored packets or flows. Second, the flow collector receives the flow records and forwards them to a flow monitoring application for further processing. The latter, finally, processes the flow records. A wide range of applications are possible, from simple archiving systems up to complex and resource intensive intrusion detection or visualization systems. Often, the monitoring application and the collector are physically located on the same machine.

Since the focus of this work is on the flow monitoring application, we assume that the exporter is not a bottleneck in the system, in the sense that it is able to process all packets seen at the measurement point. This is not necessarily guaranteed by flow exporters that are located in network routers because routing operations have usually higher priority than the flow-metering process [16], but it is true for the dedicated flow probe that we have used for our experiments in Section VI. Packet losses at the exporter can be regarded as a form of uncontrolled sampling and are not the subject of this paper.

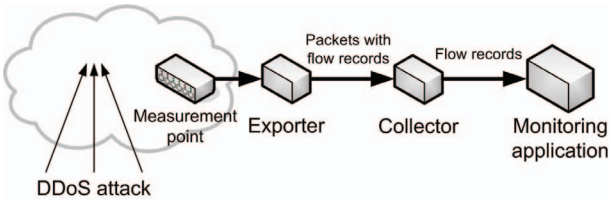



Fig. 1: The considered scenario

### B. The DDoS attack

In our scenario, we assume that a DDoS attack is observed by the probe. It should be emphasized that we do not require that the attack is aimed at the monitoring system itself.

In general, DoS attacks can be roughly classified into *semantic* and *brute-force* attacks. Semantic attacks exploit a particular weakness of the target system. In contrast, brute-force attacks rely on resource exhaustion by generating a large number of network connections, network traffic, or service requests. For the following study, not all kinds of DoS attacks are of equal interest. Semantic attacks generally do not cause a significant change to the traffic on flow level. Similarly, brute-force attacks that generate a few flows with very high volumes do not have any impact on the monitoring system, provided that the probe is able to handle the increased number of network packets, as assumed in the previous section.

Hence, in the following, we focus on an attack that generates a large amount of flows with unique flow keys: A DDoS attack where multiple attackers simultaneously send a high number of SYN packets to a victim host over a short period of time.

## IV. The impact of the attack on the flow exporter

This section investigates the impact of the attack on the flow exporter and on the created flow records. We start with a brief discussion of the behavior of the flow exporter in an attack-free situation in Section IV-A. Then, in Section IV-B, we discuss the effects of the attack on the exporter.

### A. The attack-free situation

In the following, we consider an attack-free situation as a situation where the flow probe operates with a light load. In this situation, flow records mostly expire because of a FIN or RST flag or, in the case of UDP flows, because of the inactivity timeout. Although fluctuations at very small time scales (due to traffic bursts) and at very large time scales (day-night patterns etc.) exist, it is reasonable to assume for our work that the number of exported flow packets per time unit is rather constant on minutes scale, as well as the number of flow records per packet.

### B. Effects of the attack on the flow exporter

The DDoS attack introduced in Section III-B is based on SYN-flooding and generates a large number of flows with unique flow keys. If the attack rate is high enough, the attack can quickly exhaust the internal memory, or *flow cache*, of the probe because every incoming SYN packet generates a new flow record. In order to insert the new flow record into the flow cache, the probe has to prematurely expire an existing flow record, directly export the newly created flow or directly prevent the flow from being accounted. The probe we have considered in our experiments adopts the first strategy, thus increasing the number of exported flow records during an attack. This effect does not only concern the flow records of the attack traffic (called "attack flow records" in the following) but also those of the normal, benign traffic (called "normal flow records").

As a result, the rate of exported flow records increases both for the attack traffic as well as for the normal traffic. The change in the lifetime of flow records also affects the other flow metrics. Since a flow record stays less time in the probe, the number of network packets aggregated by that flow record decreases. To summarize, from the viewpoint of the flow monitoring application, flows become (i) more frequent, (ii) smaller (in terms of number of packets and number of bytes), and (iii) shorter (in terms of time between first and last packet) during the attack.

It should be noted that normal flow records can still aggregate more than one packet during the attack, even if the flow cache is completely filled. Let be $\lambda_a$ the arrival rate of the malicious SYN packets (measured in packets per second) and $C$ the size of the flow cache. We assume that the number of normal flows is negligible. If the probe employs a simple LRU strategy to manage the flow cache, a flow record can stay $\theta = C/\lambda_a$ seconds in the cache before expiring, i.e., any flow record of a flow with a packet inter-arrival time of less than $\theta$ seconds can aggregate more than one packet.

Since the inactive timeout $\tau_{in}$ is usually much larger than $\theta$ (typically $\tau_{in} \geq 15$ s [17], while $\theta = 5$ s for $C = 5 \cdot 10^5$ and $\lambda_a = 10^5$ packets/s), we also see a decrease of the delay between flow expiration and flow export because flow records leave the flow cache before the inactive time becomes effective.

What is the minimum attack rate $\lambda_{min}$ to cause the above effects? The attacker has to fill the flow cache before the flow records time out, i.e., $\lambda_{min} = C/\tau_{in}$. For $\tau_{in} = 15$ s, we obtain $\lambda_{min} = 33\,333$ packets/s. That rate can be easily achieved by a DDoS attack.

## V. The impact of the attack on the application

The goal of this section is to study the behavior of the flow monitoring application in the presence of the attacks, by means of a simple queueing model of the monitoring application.

Our model, described in Section V-A, will be used in two ways in this paper. First, it allows us to identify several general effects of the attack on the flow monitoring application in Section V-B. The consequences of these effects, combined with those already identified in Section IV-B, are then discussed and summarized in Section V-C. Secondly, we will use the model in our trace-driven simulation experiments in Section VI-D.

### A. A model of a flow monitoring application

We propose a simple queueing model to study the behavior of flow monitoring applications in the presence of DDoS attacks. The model is based on the following assumptions:

- As already described in Section III-A, we assume that the exporter is not a bottleneck in the system.
- We assume that the network connection between the exporter and the collector and also the processing power of the collector are sufficiently dimensioned and, hence, are not relevant for our study.
- Finally, we assume that the flow application buffers incoming flow records if needed.

We model the monitoring application by a single queueing station with a single service station, buffer capacity $B$ and FCFS service strategy. The jobs arriving at the queue represent the flow packets as sent by the exporter. A job only carries the following information of a flow packet: the number of attack flow records and the number of normal flow records. Details of the flow records inside the packet, such as IP addresses, are not modeled. For the simulation experiments performed in Section VI-D, those numbers, as well as the inter-arrival times of the jobs, will be extracted from measurement data.

In order to specify the service process $S$ of the station, we consider the two, rather opposite, types of applications already hinted in Section III-A. The first type stands for a simple storage process that directly writes the flow records to disk for later usage. Typically, such a storage process has a rather constant service rate, hence we choose a deterministic service time distribution. For the second type, we consider a rather complex flow analysis system, such as an intrusion detection or visualization system, where the service times show a high variance. For example, an intrusion detection system could perform different processing steps depending on specific characteristics of the flow records. In order to cover a wide range of scenarios, we have chosen for our experiments a proven and flexible service process with flow packet service times independent-identically distributed according to a hyper-exponential phase-type distribution [18], [19].

Different values are possible for $B$. For example, the monitoring application could only rely on the small I/O buffers of the operating system. At the other extreme, the flow collector could write the collected flow records to disk from where they are read by the application. In that setup, the disk would act as a buffer of nearly infinite capacity. We will perform the experiments in Section VI-D with different buffer sizes.

### B. Effects of the finite queueing capacity

A powerful DDoS attack can easily overload a monitoring application if the latter is not explicitly designed for that. In such a situation, i.e., $\lambda \gg \mu$, the rate of processed flow records approaches $\mu$ and the losses, caused by the finite queueing capacity, are distributed over the normal flow records and the

attack flow records according to their relative proportion. Let $\lambda_n$ and $\lambda_a$ be the rate of incoming normal, respectively attack, flow records. The effective rate of processed (not lost) flow records is then $\lambda_{n,\text{eff}} = \mu \frac{\lambda_n}{\lambda_n+\lambda_a}$ for the normal flow records, respectively $\lambda_{a,\text{eff}} = \mu \frac{\lambda_a}{\lambda_n+\lambda_a}$ for the attack flow records.

The above equations show that changes in $\lambda_a$ can influence both $\lambda_{n,\text{eff}}$ and $\lambda_{a,\text{eff}}$. We illustrate this by a numerical example. Let $\lambda_n = 10^4$, $\lambda_a = 10^5$, and $\mu = 2 \cdot 10^4$ (we left out the units for better readability). We obtain $\lambda_{n,\text{eff}} = 1.82 \cdot 10^3$ and $\lambda_{a,\text{eff}} = 1.82 \cdot 10^4$. Now, if the attack rate increases by 10% to $\lambda'_a = 1.1 \cdot 10^5$, the effective rates become $\lambda'_{n,\text{eff}} = 1.67 \cdot 10^3$ and $\lambda'_{a,\text{eff}} = 1.83 \cdot 10^4$, i.e., the effective rate of attack flow records increases by 0.83%, while the effective rate of normal flow records decreases by 8.3%.

### C. Consequences of the attack

An important conclusion that we can draw from Section IV-B and V-B is that a DDoS attack causes "phantom" fluctuations in the characteristics of the observed flows from the perspective of the monitoring application. As discussed in Section IV-B, the attack not only creates a large number of attack flow records but also dramatically changes many characteristics of the normal flow records. Furthermore, we have seen in Section V-B that changes in the attack rate can cause large relative variations in the observed rate of normal flow records. This means that the monitoring application, respectively the human operator using that application, perceives changes in the traffic that are, in fact, artifacts of the measurement setup.

This can affect the performance of flow monitoring applications that rely on the flow metrics, such as traffic classification systems or security related applications. For example, a flow-based intrusion detection system could suspect malicious activities in the *normal* traffic, hence raising alerts pointing the security manager to the wrong part of the traffic.

## VI. Experiments and validation

The validation and the experiments are based on real flow measurement data of a DDoS attack monitored at the University of Twente. We provide information on the measurement platform used for the data capture in Section VI-A. The characteristics of the attack are discussed in Section VI-B. In Section VI-C, we analyze how the attack affected the collected flow records and compare the empirical data with our results from Section IV. Finally, in Section VI-D, we present simulation results on the performance of (fictive) monitoring applications. In particular, we focus our attention on the case where an application is overloaded due to the attack, as discussed in Section V.

### A. Data collection and post-processing

The trace we analyzed was collected at the UT network in February 2011, when an IRC server at the UT became target of a massive DDoS attack. We used a dedicated flow probe [20] equipped with a 10 GBit/s monitoring interface to generate NetFlow v9 records from the IP traffic at one of the main switches of the UT. This switch handles all traffic to and from the UT servers and the student (campus) network, with an average load of 7000 flows/s, equivalent to 615 000 packets/s and 4.5 Gbps. The probe exports flow records in UDP packets that we collect via tcpdump [21]. The data collection had a duration of 24 hours and a subset of the trace has been used for the following discussion.

Two important post-processing operations have been performed offline on the data. First, the DDoS attack put extra load on the collector, causing packet losses. Fortunately, missing flow packets can be easily detected thanks to the sequence counter field in the NetFlow v9 packet header. In total, around 5.8% of the 22 670 310 flow packets generated by the probe were lost. Since detailed knowledge of the packet contents is not needed for the following experiments, we have filled gaps in the trace with synthetic packets with the same characteristics as the packets surrounding the gap.

The second operation concerns the timestamps of the flow packets. The data trace offers two potential sources for the arrival times: (i) the export timestamps in the flow packet headers, and (ii) the timestamps recorded by tcpdump when collecting the packets. However, the export timestamps in NetFlow v9 only have second resolution[1] (see also [22]). On the other hand, the timestamps assigned by tcpdump on the collector have roughly 10 ms resolution, but can include delays caused by the network stack. We have observed, however, that packets exported in the same second get timestamps on the collector that are rather evenly distributed over one second, even if the collector is very weakly loaded, i.e., when queueing delays can be assumed to be close to 0. Consequently, we have employed the following scheme to reconstruct timestamps with millisecond precision: Given a series of $n$ flow packets with the same export time $x$ (in seconds), we compute the synthetic export time $x + (i-1)/n$ for the $i$th packet of the series.

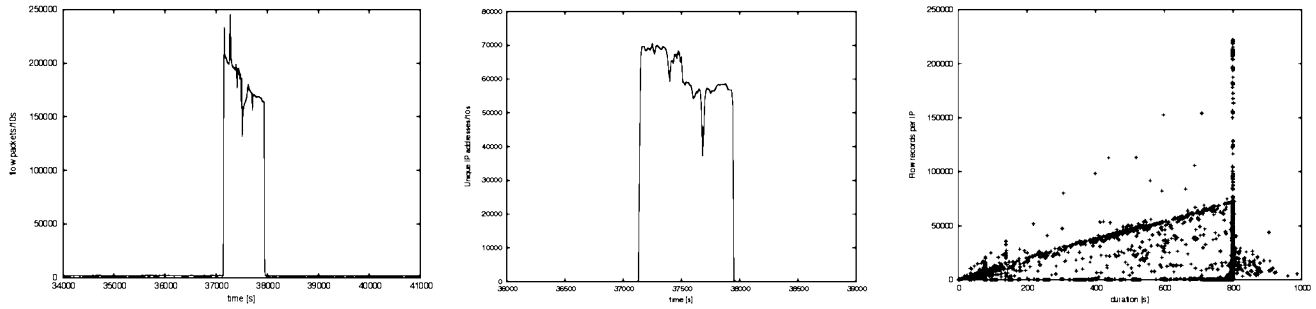
Figure 2(a) shows the timeseries of exported flow packets per 10 seconds, based on the synthetic export times. The time on the x-axis is expressed as the number of seconds elapsed since the beginning of the trace (around 10.00 AM). We note a very low base line of around 2000 exported packets per 10 seconds and a burst of activity beginning at timestamp 37140. This anomaly, which is in fact a DDoS attack, will be the topic of the following section.

### B. Description of the attack

The monitored DDoS attack lasted for 800 seconds (13.3 minutes) and flooded its target with around 376.2 million TCP packets on port 6667. The packets had an average size of 61.2 bytes and most of them (368.6 million) were SYN packets.

The attack can be easily identified in Figure 2(a) as the period from time 37140 to 37950 with very high flow packet export rate. Note that there is a direct relationship between the number of SYN packets sent by the attackers and the number of exported flow packets, because every monitored SYN packet creates a new flow record in the probe. Since the

[1]For the particular flow probe used by us, this limitation also applied to the `sysuptime` field, which would be otherwise another potential source.



(a) Exported flow packets (b) Unique IP addresses (c) Host activity

Fig. 2: Time series (10 sec) of the number of exported flow packets (Fig. (a)), the number of unique IP addresses (Fig. (b)), and attacker activity (Fig. (c)).

number of flow records per flow packet is nearly constant (99% of all flow packets contain 27 or 28 flow records), Figure 2(a) indeed provides an overview of the evolution of the attack intensity in terms of network flows.

More than 20 000 unique IP addresses participated to the attack, however with varying intensity. 127 attackers sent more than 100 000 packets each, and 3185 attackers sent between 50 000 and 100 000 packets. There are strong indications that the attack was coordinated: Most of the top 10 000 attackers joined in exactly the same second and then stayed active for the entire attack duration. Figure 2(b) shows the number of active unique IP addresses per 10 seconds. When the attack begins, we observe a sharp rise from a base line of around 10 unique IPs to almost 70 000 and the number fluctuates from 70 000 to 60 000 during the duration of the attack. The figure also shows sudden drops in the number of attacking hosts around the second 37600. This is due to packet loss occurred when the load on the collector was too high (see Section VI-A).

An additional proof that the attack has been coordinated is given by Figure 2(c). The figure shows, on the x-axis, the time in seconds during which an attacker has been active (and contacted the target at least 50 times); on the y-axis, the number of flow records generated by each attacker. We can see that a large portion of the attackers has been active for precisely 800 seconds, as indicated by the vertical line at the right side of the figure. Moreover, it also becomes evident that a second group of attackers has been active for an interval of time varying from few seconds to 800 seconds, but with a constant rate of flows per second (corresponding to a rate of 100 SYN packets per second). In addition, a third group of attackers have sent a relatively low number of SYN packets per second, and they generate the uneven baseline in the figure. The major characteristic of such hosts is that they are clustered in groups of attackers sending the same number of SYN packets. Finally, from the figure we can also infer that several other hosts have contacted the target with varying activity durations and intensity (the dots in the plot that do not follow any of the three behaviors previously indicated).

### C. Impact of the attack on the flow exporter

We now concentrate our attention to the impact of the attack on the flow exporter. As described in Section IV-B,

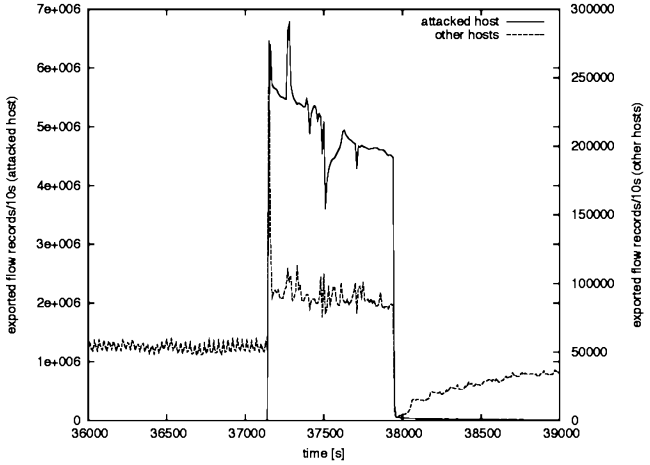



Fig. 3: Number of exported flow records per 10 seconds, for flow records of the attacked host and of the other hosts.

a SYN flood attack will force the monitoring probe to deal with an anomalous number of flow records. In order to better understand how the attack affects the flow records, we have split up them into two sets: (i) flow records of flows from/to the attacked host and (ii) flow records of flows from/to the other hosts. Figure 3 shows the resulting two timeseries of the number of exported flow records per 10 seconds. Note the different scales on the two y-axes. We can see that the attacked host is not very active before the attack. In average, only 10 to 15 records per second contain the attacked host as source or destination. As expected, the flow record export rate for the attacked host sharply increases when the attack starts because every SYN packet creates a new flow record.

However, we can also observe that the export rate for the *other* hosts increases as well during the attack. This behavior has been predicted in Section IV-B. As described there, if a very large number of flows with unique flow keys is created, as happens in the DDoS attack, the internal memory of the probe is quickly exhausted and new flow records displace existing records. This mechanism is also responsible for the extreme peak in the export rate for the other hosts at the begin of the attack (timestamp 37140): The new flow records for the malicious traffic "push" most of the existing flow records out

TABLE I: Impact of the attack on the exported flow records

| | before attack (all hosts) | during attack | |
|---|---|---|---|
| | | other hosts | attacked host |
| avg. #flow packets/s | 192 | 18 055 | |
| avg. #flow records/s | 5301 | 9294 | 495 909 |
| avg. flow duration [s] | 8.0 | 2.3 | 0.08 |
| avg. #packets/flow | 87.3 | 52.5 | 1.0 |
| avg. #bytes/flow | 75 671 | 46 824 | 61.2 |

of the probe. Therefore, the observed increase in the number of non-malicious flow records is an artifact of the flow creation process and does not correspond, on the network, to a real increase or change in traffic.

In Table I we compare the flow behavior in a normal situation and during the attack. The table presents, both for the time frame of the attack and the one preceding it, average values for the number of exported flow packets per second, the number of flow records per second, the flow duration, the packets per flow, and the bytes per flow. The last four measures are reported both for the flows not related to the attacked host and for the traffic from/to the attacked host. Our measurements show that the average flow duration before the attack is 8 seconds. During the attack, flows from/to the attacked host have a duration almost close to 0, since they are mainly constituted by a single SYN packet. However, the average duration of the other flows also is sensibly shorter than before the attack. It indeed decreases to 2.3 seconds. Correspondingly, the average numbers of packets and bytes per flow also decrease. Again, these results are consistent with the effects identified in Section IV-B.

Finally, it should be also noted that the concrete numbers depend on the characteristics of the attack and the implementation details of the flow probe. Nevertheless, the described underlying mechanisms are present in a wide range of setups.

### D. Impact of the attack on the flow monitoring application

We study the impact of the attack on the flow monitoring application by means of the queueing model presented in Section V-A. We perform a discrete-event simulation of the model with different buffer sizes $B$ and different service time distributions, using the FiFiQueues tool [19]. The arrival process of the queue is trace-driven: the jobs, representing the collected flow packets, are generated according to the timestamps in our data trace. Each job specifies the number of normal flow records and the number of attack flow records. In order to simplify the following discussions, we consider all flows from or to the attacked host as attack traffic, hence ignoring that a tiny fraction of those flows (see Section VI-C) is in fact legitimate. In the following, we simulate different behaviors by varying the service rate, the service time variation and the buffer size of the queueing model. Our aim is to cover typical application setups by our experiments.

*1) A high-performance monitoring application:* We consider a powerful monitoring application with a high service rate of $\mu = 600\,000$ flow records/s, deterministic service times, and a small buffer with space for 100 flow packets (corresponding to approximately 2800 flow records). We show the

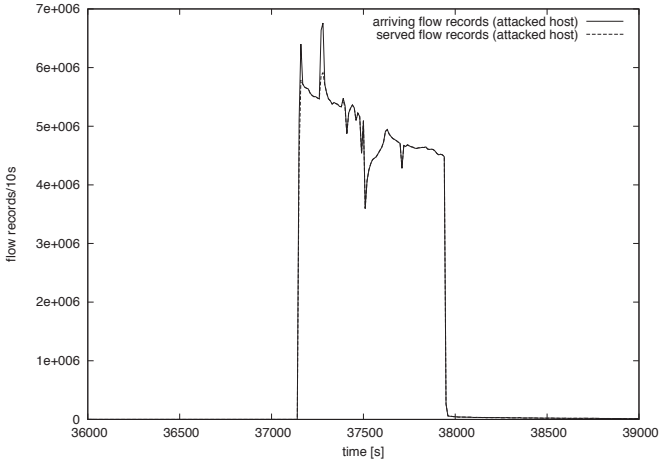



Fig. 4: Number of arriving and served flow records per 10 seconds, for flow records of the the attacked host ($\mu = 6 \cdot 10^5$ flow records/s, $B = 100$, deterministic service process)

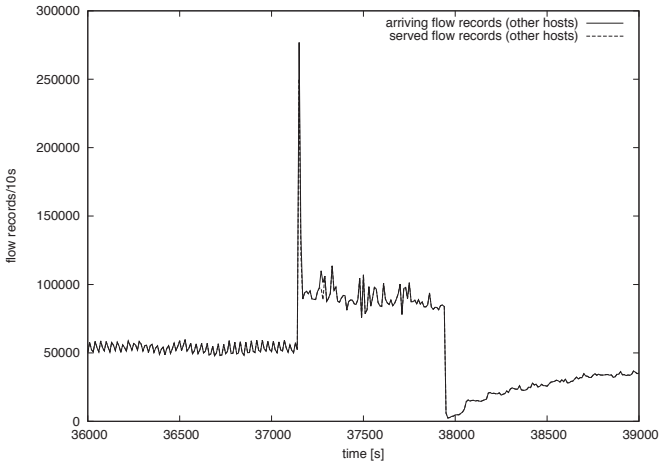



Fig. 5: Number of arriving and served flow records per 10 seconds, for flow records of the other hosts ($\mu = 6 \cdot 10^5$ flow records/s, $B = 100$, deterministic service process)

rate of flow records arriving to the application and the resulting rate of flow records effectively served by the application as function of time for the flow records of the attacked host in Figure 4, respectively for the flow records of the other hosts in Figure 5. The time resolution is 10 seconds.

As expected, the difference between the rate of arriving records and the rate of served records, i.e., the throughput, is very small. Most of the losses occur during the two peaks in the first two minutes of the attack. In order to illustrate the influence of the buffer size on these losses, we repeat the simulation with a buffer sizes of $B = 10\,000$. In Figure 6, we compare the resulting rates of served flow records for $B = 100$ and $B = 10\,000$ for the period of interest. The time resolution is one second. As can be seen, the throughput of the application is close to the theoretical maximum $\mu$ during the peaks for both buffer sizes, but the larger buffer can slightly reduce the total number of lost flow records from around 2 970 000 to 2 260 000. Note that the queueing station needs

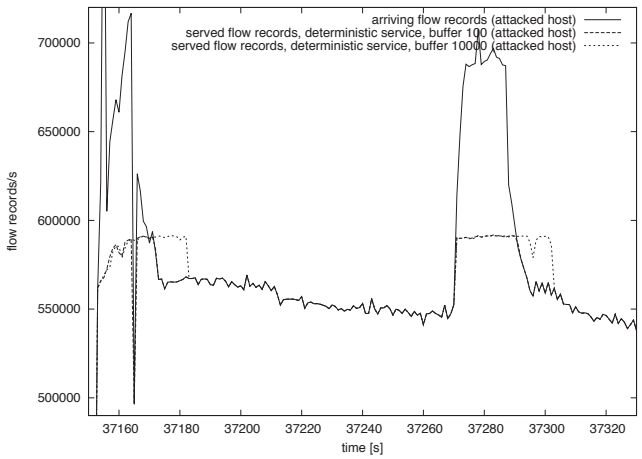



Fig. 6: Number of arriving and served flow records per second for different buffer sizes for flow records of the attacked host ($\mu = 6{\cdot}10^5$ flow records/s, deterministic service process)

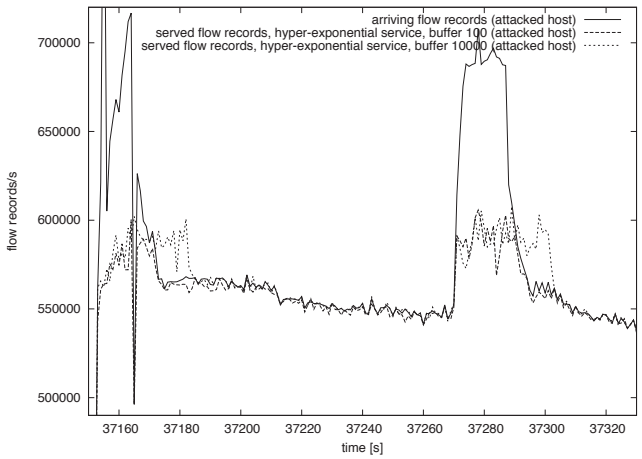



Fig. 7: Number of arriving and served flow records per second for different buffer sizes, for flow records of the attacked host ($\mu = 6{\cdot}10^5$ flow records/s, hyper-exponential service process)

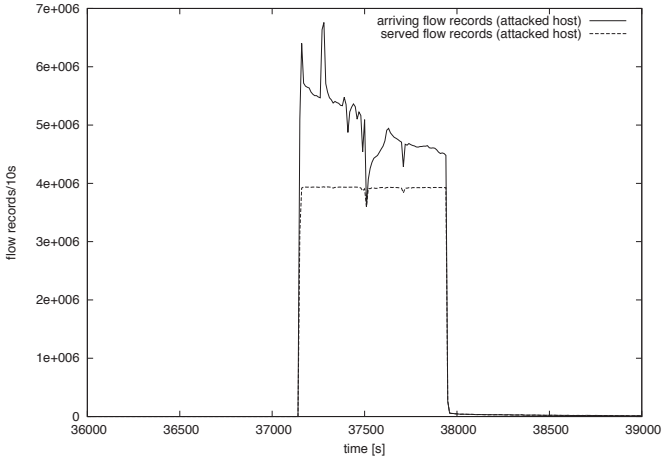



Fig. 8: Number of arriving and served flow records per 10 seconds, for flow records from/to the attacked host ($\mu = 4{\cdot}10^5$ flow records/s, $B = 100$, deterministic service process)

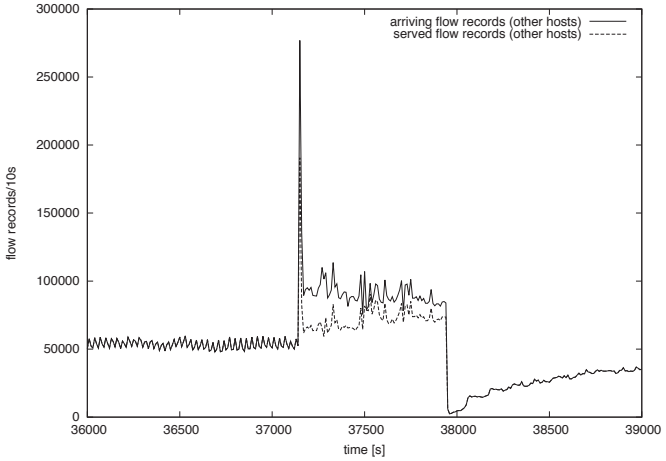



Fig. 9: Number of arriving and served flow records per 10 seconds, for flow records from/to other hosts ($\mu = 4 \cdot 10^5$ flow records/s, $B = 100$, deterministic service process)

several seconds to empty the queue after the attack peaks because the attack rate remains at a high level.

In the next experiment, we replace the deterministic service times by a hyper-exponential distribution with a high squared coefficient of variation of 4 (see Section V-A). The overall behavior of the model does not significantly change compared to the previous experiments. The impact of the higher service time variation can be best seen during the two attack peaks that we have already studied above. In Figure 7, we show the evolution of the throughput over time as obtained from two simulation runs with $B = 100$ and $B = 10\,000$. We observe that the throughput varies much more during the attack peaks than in the setup with constant service times. Consequently, the number of losses is larger with around $3\,480\,000$ records for $B = 100$, respectively $2\,300\,000$ records for $B = 10\,000$.

*2) A medium-performance monitoring application:* For the next experiments we lower the service rate to $\mu = 400\,000$ flow records/s, i.e., to a service rate smaller but still close to the rate of attack flow records. Again, we consider a deterministic service time distribution and $B = 100$. Figure 8 and Figure 9 show the arrival rate and the resulting throughput as function of time for the flow records of the attacked host, respectively for the flow records of the other hosts. The time resolution is 10 seconds. Obviously, the loss rate is very high (around 20%) because the station is clearly overloaded. The results for other buffer sizes and service time distributions look very similar in such a situation, hence we have refrained to show them here.

In Figure 9, we can see that the rate of served normal flow records increases although their arrival rate decreases. This effect has been described in Section V-B. The equations given in that section predict that the effective rate of normal flow records increases when the attack intensity decreases, provided that the system is overloaded. In order to validate this analytical result, we calculate the rates of the served flow records at two points in time: at the begin of the attack (shortly after the first attack peak) and at the end of the attack (shortly

TABLE II: Influence of the attack rate on the application ($\mu = 4 \cdot 10^5$ flow records/s, $B = 100$, deterministic service process)

| #flow records/s | simulation | analytical | RE |
|---|---|---|---|
| other hosts (begin) | 6513 | 6598 | 1.3% |
| attacked host (begin) | $3.940 \cdot 10^5$ | $3.934 \cdot 10^5$ | -0.15% |
| other hosts (end) | 7463 | 7400 | -0.84% |
| attacked host (end) | $3.929 \cdot 10^5$ | $3.926 \cdot 10^5$ | -0.08% |

before the attack rate drops to nearly 0). In Table II, we compare the results obtained by simulation (first column) with those from our equations (second column). The third column gives the relative error (RE) between the analytical results and the simulation results. As can be seen, the error is very small.

*3) A low-performance monitoring application:* For our last experiment, we choose a, relative to the attack intensity, low service rate of $\mu = 100\,000$ flow records/s. Of course, this results in a high loss rate during the attack. For $B = 100$ and deterministic service times, we lose 73% of the flow records. This can have a severe impact on the effectiveness of the monitoring application, since the application only sees a small fraction of the ongoing network traffic.

For certain applications, such as intrusion detection systems, it might help to choose a higher buffer capacity. A buffer of $B = 10\,000$ would not significantly improve the loss ratio (it stays at 73%), but it would allow an intrusion detection system to, at least, monitor and (hopefully) identify the beginning of the attack. Once the attack has been identified, counter measures, like a reconfiguration of the firewall, could be initiated. Such counter measures may take several seconds to minutes to become effective. During that time, the network is vulnerable: A secondary, slower attack hidden in the normal traffic has a high probability to stay undetected.

If we further increase the buffer size, our model approaches the behavior of a semi-real time analysis system. A (probably disk based) buffer with size $B = 1\,000\,000$, which we can consider as "nearly infinite", certainly lowers the number of missed flow packets, however with the price of a much higher latency. Once the queue is filled, incoming flow records experience a waiting time of 280 seconds. This opens another way for a successful secondary attack: the detection system is first "blinded" by a massive DDoS attack, which gives a subsequent attack enough time to act before detection.

## VII. Conclusions

In this paper, we have studied the effects of Distributed Denial-Of-Service attacks (DDoS) on flow monitoring applications. Based on an analysis of the functioning of flow probes, we have shown that a DDoS attack can significantly alter the major characteristics of the data exported by a probe. Thereafter, we have provided a simple queueing model of a flow monitoring application. This model has allowed us to describe the influence of the attack on the application, in this way showing that variations in the attack density can, from the perspective of the monitoring application, also impact the normal traffic data. Finally, we have validated our findings by means of real flow measurement data. We have used the data in trace-driven simulations of the application model and studied the behavior for different parameter sets.

We draw the following conclusions from the obtained results. First and foremost, we have successfully identified, on flow record level, the impact of DDoS attacks. The observed changes in the flow metrics are relevant and might cause unwanted effects in monitoring applications. We suggest that the developers of such applications carefully evaluate the response of their designs to network anomalies with real data traces. Furthermore, our simulation experiments have shown that our model, although currently very simple, can help to parameterize and dimension flow-based monitoring systems and reveal potential vulnerabilities, such as the risk to miss stealth attacks hidden in DDoS attacks.

*Acknowledgment*: This research has been supported by the EU FP7-257513 UniverSelf Collaborative Project and the SURFnet's GigaPort3 project for Next-Generation Networks.

## References

[1] J. Quittek, T. Zseby, B. Claise, and S. Zander, "Requirements for IP Flow Information Export (IPFIX)," RFC 3917 (Informational).
[2] Cisco.com, "Cisco IOS NetFlow Configuration Guide, Release 12.4," http://www.cisco.com, Sept. 2010.
[3] A. Pras, R. Sadre, A. Sperotto, T. Fioreze, D. Hausheer, and J. Schoenwaelder, "Using NetFlow/IPFIX for Network Management," *Journal of Network and Systems Management*, vol. 17, no. 4, p. 6, 2009.
[4] A. Sperotto, G. Schaffrath, R. Sadre, C. Morariu, A. Pras, and B. Stiller, "An Overview of IP Flow-Based Intrusion Detection," *IEEE Communications Surveys & Tutorials*, vol. 12, no. 3, pp. 343–356, 2010.
[5] A. Sperotto, "Flow-based intrusion detection," Ph.D. dissertation, University of Twente, October 2010.
[6] D. Brauckhoff, B. Tellenbach, A. Wagner, M. May, and A. Lakhina, "Impact of packet sampling on anomaly detection metrics," in *Proceedings of the 6th ACM SIGCOMM conference on Internet measurement*, ser. IMC '06, 2006, pp. 159–164.
[7] J. Mai, C.-N. Chuah, A. Sridharan, T. Ye, and H. Zang, "Is sampled data sufficient for anomaly detection?" in *Proceedings of the 6th ACM SIGCOMM conference on Internet measurement*, ser. IMC '06, 2006, pp. 165–176.
[8] Z. Li, Y. Gao, and Y. Chen, "HiFIND: A high-speed flow-level intrusion detection approach with DoS resiliency," *Computer Networks*, vol. 54, no. 8, pp. 1282 – 1299, 2010.
[9] M. Molina, A. Chiosi, S. D'Antonio, and G. Ventre, "Design principles and algorithms for effective high-speed IP flow monitoring," *Computer Communications*, vol. 29, no. 10, pp. 1653 – 1664, 2006.
[10] N. Duffield and C. Lund, "Predicting resource usage and estimation accuracy in an IP flow measurement collection infrastructure," in *Proceedings of the 3rd ACM SIGCOMM conference on Internet measurement (IMC '03)*. ACM, 2003, pp. 179–191.
[11] C. Estan, K. Keys, D. Moore, and G. Varghese, "Building a better NetFlow," *SIGCOMM Comput. Commun. Rev.*, vol. 34, pp. 245–256, 2004.
[12] Y. Hu, D. M. Chiu, and J. C. Lui, "Adaptive Flow Aggregation - A New Solution for Robust Flow Monitoring under Security Attacks," in *In IEEE/IFIP Network Operations and Management Symposium (NOMS '06)*, 2006, pp. 424–435.
[13] Y. Wang, C. Lin, Q.-L. Li, and Y. Fang, "A queueing analysis for the denial of service (DoS) attacks in computer networks," *Computer Networks*, vol. 51, pp. 3564–3573, August 2007.
[14] M. Long, C.-H. Wu, and J. Hung, "Denial of service attacks on network-based control systems: impact and mitigation," *IEEE Transactions on Industrial Informatics*, vol. 1, no. 2, pp. 85–96, 2005.
[15] S. Khan and I. Traore, "Queue-based analysis of DoS attacks," in *Proceedings from the Sixth Annual IEEE SMC Information Assurance Workshop (IAW '05)*, 2005, pp. 266–273.
[16] Cisco.com, "Cisco White Paper "NetFlow Performance Analysis"," May 2007.

[17] T. Fioreze, "Self-management of hybrid optical and packet switching networks," Ph.D. dissertation, University of Twente, February 2010.
[18] B. Haverkort, "QNAUT: Approximately analyzing networks of PH|PH|1|K queues," *Proceedings of the 1996 International Computer Performance and Dependability Symposium*, p. 57, 1996.
[19] R. Sadre, B. Haverkort, and A. Ost, "An efficient and accurate decomposition method for open finite- and infinite-buffer queueing networks," in *Proc. 3rd Int. Workshop on Numerical Solution of Markov Chains*, 1999, pp. 1–20.
[20] INVEA-TECH.com, "INVEA-TECH - High-Speed Networking and FPGA Solutions," http://www.invea-tech.com, May 2011.
[21] Tcpdump.org, "TCPDUMP & LIBPCAP," http://www.tcpdump.org/, May 2011.
[22] B. Trammell, B. Tellenbach, D. Schatzmann, and M. Burkhart, "Peeling Away Timing Error in NetFlow Data," in *Passive and Active Measurement conference (PAM 2011)*, March 2011.