Scope-based Flow Monitoring to Improve Traffic Analysis in Programmable Networks

Christoph Hardegen
Department of Applied Computer Science, Fulda University of Applied Sciences, Germany
Email: christoph.hardegen@cs.hs-fulda.de

***Abstract*—Flow monitoring allows to obtain an aggregated network traffic view that can be leveraged for subsequent analysis. Since network management tasks like flow-based traffic classification or prediction benefit from broader data views, the flow tracking scope used to export required traffic metadata can be enlarged: First, coherent packet streams can not only be monitored in a unidirectional but also bidirectional context that combines interrelated forward and backward direction views. Second, time-based subflow management for both contexts separates observed packet streams into consecutive windows covering a particular fraction to gain higher data granularity. To support these diversified traffic views in combination with variable feature sets for demand-driven data export serving different traffic analysis tasks, flow tracking and export strategies are required to operate in a dynamic manner. This paper proposes a flow monitoring approach enabling to track the four aforementioned scopes while adapting timeout-based data export operating on programmable switches. A multi-level system architecture and an adaptive protocol ensure flexible sharing and analysis of data records. Evaluations show that exported data can be used to improve analysis outcomes, whereby the considered data scope affects achieved accuracy but also the monitoring overhead.**

***Index Terms*—Flow Monitoring, Flow Tracking, Flow Export, Traffic Analysis, Programmable Switches**

I. Introduction

Network traffic in modern environments is subject to complex patterns and conditions. This is due to ever growing network architectures comprised of various sub-networks and a constantly increasing number of interconnected systems. While these run different platforms and a variety of network services, experienced dynamics are high. For example, traffic volumes fluctuate because the number of individual packet streams having diversified properties and the amount of transmitted data vary over time [1]. As a consequence, management complexity w.r.t. traffic analysis is challenging.

Flow monitoring can be run to collect an aggregated data basis serving as decision input for flow-based traffic analysis. Since the available data view contained in each obtained metadata record affects analysis outcomes, the tracking scope for observed packet streams can be enlarged: First, streams belonging to the same network communication can be monitored either in a unidirectional or bidirectional scope that consider forward and backward direction contexts in an isolated or combined manner. Second, consecutive windows of smaller time scales can be maintained on uni- and bidirectional packet streams, thus enabling various granularities for tracked data features and their timely resolution. Due to enhanced data views, these diversified scopes ensure accurate traffic analysis.

To support sophisticated management decisions in an open or closed loop cycle, machine learning (ML) methods can assist in establishing traffic classification or prediction systems. While these can leverage the aforementioned flow scopes, for example, flow level network intrusion detection strategies are enabled to precisely differentiate between benign and malicious traffic in order to enforce proper flow control [2]. Because required data scopes and feature sets vary for different management tasks, monitoring approaches need to be adaptive.

Fostered by the paradigms Software-Defined Networking (SDN) and Programming Protocol-independent Packet Processors (P4), switch architectures are currently evolving. Devices equipped with a programmable data plane enable flexible provisioning of network functions and, e.g., due to Linux-based network operating systems (NOS), dynamic assistance through locally running agent processes. In addition, tasks that require a global network view or exceed local processing capabilities can be assisted by entities located in centralized SDN controllers. Since data plane, NOS and controller level hold individual capabilities w.r.t. their available data view, system resources and processing scalability, hybrid solutions being deployed on multiple levels offer large practicability.

Programmable switches allow for fine-grained feature collection while operating at packet level. As an example, timestamps can be determined with high precision to overcome challenges like inaccurate computation of flow duration and derived throughput due to improper timestamp resolution [3]. This ensures accurate data views that can, for example, be used for precise flow-based traffic classification [2] or traffic prediction estimating likely arising flow loads [1] [3].

This paper presents a flow monitoring approach entitled `FlowMoni` holding the following main contributions:

- A multi-level architecture for flow data tracking, export and collection allowing traffic analysis at a centralized SDN controller and distributed switch level is proposed.
- Accordingly, an adaptive data export protocol supporting four flow scopes with flexible feature sets is introduced.
- Differentiated data export w.r.t. multiple collectors combined with in-network data preprocessing is enabled.
- The accuracy benefit of leveraging diversified data views for flow-based traffic classification in the context of network intrusion detection is exemplarily measured.
- The monitoring overhead w.r.t. processing time, export data volume, latency impact and memory demand is estimated.
- The code base is publicly available to ensure reproducibility.

II. Related Work

[4] tracks network flows on programmable data planes with P4 to share maintained data records with a collector. Compared to [4], this paper extends flow tracking considering four scopes: First, besides monitoring unidirectional packet streams, interrelated bidirectional ones are tracked as combined data view. Second, window-based subflow data scopes of higher granularity are supported for both contexts.

[5] proposes a flow monitoring system that relies on programmable data planes with P4 and aims at providing insights on heavy hitter flows. As soon as a particular flow reaches a first packet threshold, detailed monitoring is run and in case the observed packet count exceeds another limit, data collection is initiated by a controller using crawler packets. In contrast, this paper considers timeout-based tracking and export operation to enable a general flow-based traffic view rather than focusing on heavy hitters. In addition, data tracking and export is entirely run in-network at switch level, whereby a NOS agent that runs monitoring in collaboration with a data plane module pushes preprocessed data records to collectors.

[6] presents an approach for accurate flow data record generation. At data plane level, a microflow generator tracks data for observed packet streams at short timescales. At CPU level, a microflow aggregator buffers and aggregates shared microflow records to flow records that are afterwards sent to a collector. Compared to [6], this paper runs data tracking and corresponding record construction entirely in the data plane, whereby a NOS agent supports data export management. Thus, the cooperative behavior between data plane and CPU level is the same as in [6] but with different motivation. In addition, the concept of subdividing a packet stream is adopted by including a subflow scope that respects a trade-off between increased monitoring granularity and associated export complexity.

[7] introduces a strategy for flow level monitoring that decouples data collection from feature statistics computation. A cache with grouped packet vectors is operated within the data plane. Entries contain a sequence of timestamp and per-packet feature value tuples that describe the observed trend for short timescales and are shared as data stream with collector instances on external servers that run subsequent feature computation and downstream analysis. In comparison to [7], this paper focuses on the coherent tracking of per-flow feature data for considered flow scopes instead of continuously streaming per-packet data. Although this is related to a loss of granularity, it reduces export complexity and volume to enable in-network traffic analysis at switch level in addition to those running at collectors located in SDN controller platforms.

Commonalities and Differences to NetFlow[8] & IPFIX[9]: First, considered export conditions, i.e., timeout-based operation, and the support for uni- and bidirectional flow tracking are similar, whereby the approach presented in this paper also allows to maintain subflow data views on streams from both contexts. Second, while newer protocol versions of NetFlow and IPFIX support more flexible data collection w.r.t. feature templates, considered properties have to be available as generally registered or vendor-specific data fields. In contrast to this, the introduced approach employs programmable switch platforms as operational basis, thus ensuring a highly adaptive feature tracking protocol due to configurable packet processing. Third, the possibility for differentiated monitoring w.r.t. multiple collectors receiving data in demand-driven formats is also common with NetFlow and IPFIX. However, besides data processing and analysis at centralized controllers running a collector instance, the proposed multi-level tracking and export architecture at switch level comprised of data plane module and NOS agent not only enables in-network data preprocessing but also distributed analysis, which is challenging for closed switch platforms and vendor-specific implementations.

III. Traffic Analysis Use-Case: Flow-based Network Intrusion Detection

In order to enable traffic analysis, flow metadata is continuously monitored based on observed packet sequences and shared as a stream of data records. Therefore, `FlowMoni` runs flow tracking and export at a network switch providing a flow-based traffic view that is collected and sent to downstream analysis either at distributed switch or centralized SDN controller level. This way, obtained data views can be leveraged to maintain traffic classification or prediction systems in order to derive sophisticated network management decisions.

For example, intrusion detection methods can be applied, whereby flow-based classifiers differentiate between benign and malicious traffic to enforce flow control based on obtained decisions, e.g., applying permit, drop or throttling policies to affected streams during packet forwarding. [2] proposes a hierarchichal architecture and strategy for intrusion detection that consumes flow metadata as input to run distributed, lightweight in-network flow classification analysis at switch level being backed by more advanced and compute-intensive classification methods at higher hierarchy tiers located in centralized controller platforms running in cloud infrastructures.

While considering flow-based intrusion detection as exemplary network management task, this paper mainly focuses on the data collection part required to obtain diversified data bases serving as decision input for downstream traffic analysis.

IV. `FlowMoni` Approach

A. Flow Scopes and Tracking

Multiple data tracking scopes are supported during flow monitoring to diversify and enlarge the flow-based data view that can be leveraged for subsequent traffic analysis: While the *flow* scope tracks unidirectional packet streams, the *biflow* view focuses on interrelated bidirectional packet sequences. Thus, whereas *flow* metadata describes forward and backward direction by separate data records, *biflow* metadata summarizes both contexts at once. In addition, to track timely trends with higher granularity, metadata can be maintained for successive time windows on uni- or bidirectional packet streams, i.e., leading to *subflow* and *bisubflow* scopes respectively.

Hash tables are used to hold flow data statistics, whereby n_f states their size and hence the number of supported slots. To map packets to coherent streams, flow 5-tuple data is utilized to compute a hash value leveraged as unified flow identifier.

While this strategy can be applied to unidirectional scopes directly, tracking bidirectional views requires additional steps: For example, first, IP addresses and port numbers have to be sorted before hashing to ensure consistent flow identification. Second, to account forward and backward data contexts, 5-tuple data of each packet has to be compared to the one observed in a flow's first packet to identify the direction.

To intentionally exclude particular flows from data tracking, header criteria- and hash-based filters can be applied.

B. Flow Features

Table I summarizes the initially supported flow feature set. Besides flow 5-tuple data, properties are categorized into raw and computed ones. Packet headers, metadata and timestamps provided by switch platforms serve as data collection sources.

TABLE I: FlowMoni Flow Features.

Type	Features	Sources
5-Tuple	Source/Destination IP & Port, Protocol	Packet Header (IP/TCP/UDP)
Raw	Start & End Timestamp	Switch Clock Times
	Number of Packets & Bytes	Switch Packet & Byte Counter
	TCP Flags Mask & Counts	Packet Header (TCP)
	Packet Lengths & Inter-Arrival Times (min, max, (weighted) moving average)	Packet Metadata & Switch Clock Times
Computed	Duration (Activity Time)	Start & End Timestamps
	Throughput (Packet & Bit Rate)	Packet & Byte Count, Duration

In case of bidirectional instead of unidirectional data views, raw and computed data features are maintained for forward and backward direction. Additionally, throughput rates in terms of transmitted packets and bytes from both contexts are put into relation to determine up-down- and down-up-ratios.

If window-based data views are managed, packet and byte counters as well as packet length and inter-arrival time features are tracked as raw data for each elapsed time window. In case of running window management for bidirectional views, this again applies to forward and backward direction contexts separately. Based on feature data obtained for individual windows, several statistics are computed as well, i.e., minimum, maximum, mean, median and standard deviation values.

C. High-Level System Overview

The system architecture consists of deployed FlowMoni Exporter and Collector instances (Figure 1).

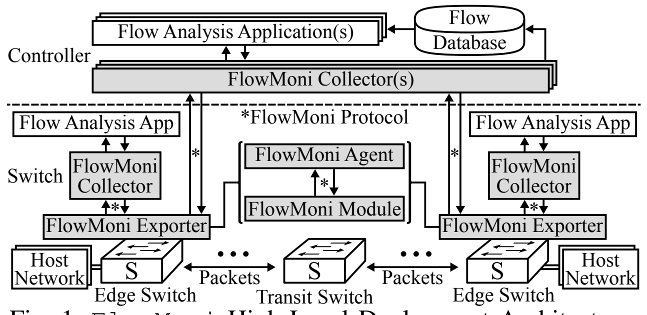

Fig. 1: FlowMoni High-Level Deployment Architecture.

The exporter is comprised of FlowMoni Module that is part of the packet processing pipeline on a programmable switch and FlowMoni Agent that runs as process within the NOS. Both perform data tracking and export collaboratively.

FlowMoni Collector either also runs locally in the switch's NOS or on a centralized SDN controller platform deployed on an external server. After receiving exported data, records are stored in a flow database for delayed analysis purposes (offline use-cases) or streamed for downstream evaluation in flow analysis applications (online scenarios). While ML-based methods like Random Forest (RF) or Deep Neural Network (DNN) instances are applied to streamed data for flow-based traffic analysis, the collector serves as streaming server and associated apps as clients. Each collector registers at an exporter to request data required by assigned apps.

A direct link, i.e., connected to a switch's CPU port, is used to exchange export packets between modules and associated agents. The latter also establish a management channel towards exposed runtime APIs to run module configuration and another one towards assigned collectors to push exported data records.

Reasons for this multi-level export architecture are as follows: Although the module runs flow tracking at data plane level, compute capabilities at this stage are limited. Therefore, the agent at CPU level in the NOS is required to assist in data export operation, i.e., retrieving maintained metadata from data plane, preparing record messages and sending those towards a collector. In addition, the NOS agent offers the potential to run in-network data preprocessing steps at switch level that need to be applied before record sharing. Examples are filtering or selecting particular feature subsets from exported data, to run address anonymization or to enrich further metadata properties. Afterwards, preprocessing continues at a collector, e.g., to derive or compute additional features, before feeding finally prepared data to downstream traffic analysis applications. The option to deploy the collector and associated apps at local switch or external server level allows to run lightweight and compute-intensive analysis tasks, whereby the available data view is either local or global. Likewise, combined multi-step analysis methods are possible. This way, scenarios that consider to run ML-based traffic analysis distributedly over the network on switches or in centralized controller platforms are enabled, e.g., flow classification and prediction strategies similar to [2] or [1] and [3] respectively.

D. Operational Strategy

Data export is triggered based on flow level active and inactive timeout hits as well as in case of detecting finished or reset TCP connections. The rationales for incorporating these conditions to provide a general flow-based traffic view are as follows: The first enforces metadata export for a particular flow after continuously observing packet activity during time interval w_A, which is of relevance for tracking long-lasting packet streams. The second initiates metadata export for a flow in case no packet activity is observed for the time frame w_I, which is of relevance for ended streams that are not handled by the third condition identifying traceably ended TCP connections for which immediate export is possible.

In case subflow or bisubflow data scopes are tracked, the active timeout is set as multiple of the used window size w_S to obtain consistent data, i.e., $w_A = n * w_S$.

If the inactive timeout is hit or a TCP connection traceably ends while maintaining subflow or bisubflow scopes, the last window may contain truncated and timely limited data that is handled by a weighting strategy to balance feature data for statistics computation w.r.t. observed window lengths.

Since an active timeout hit and a traceable termination of TCP connections for a particular flow can be detected at module level, packets including corresponding data records are pushed to the agent. Regarding inactive timeout hits, the agent is required to identify inactive streams in collaboration with the associated module and request related export packets asynchronously by sending respective trigger packets (inactivity management). Thus, each time the export timer w_E expires, the agent queries the first plus last packet timestamps for flows maintained in the assigned module and evaluates a hit w.r.t. w_I. While this is due to no further packet activity being able to trigger data export, there is one exception: In case the same flow hash is computed after flow-based time interval w_I has elapsed and the next inactivity management iteration w.r.t. w_E was not run by the agent yet, export can be initiated directly in the module before continuing data tracking.

As soon as the agent receives a raw record, preprocessing is run before pushing the prepared message towards the intended collector. Therefore, in case of subflow management, window-based data is extracted and used to compute subflow statistics that are appended to the initial base record. In addition, a set of data preparation steps ensures flexible data export for multiple registered collectors. This way, individual scope and feature requirements or constrained data sharing can be respected.

While *feature filtering* applies filter expressions for particular feature sets to reduce the number of exported data records, *feature selection* applies selector expressions to limit the number of exported features per record. Hence, a dynamic protocol that supports the export of records in an arbitrary format with regard to the available feature set is enabled. The initial set of preprocessing operations can be extended, for example, with *feature enrichment* or *anonymization* strategies.

Whereas aforementioned preprocessing steps are run on agent side, data preparation continues at collector side that performs *feature computation* to determine additional properties before data is shared with analysis apps. Further tasks like *data aggregation* and *normalization* are feasible as well.

V. Evaluation

A. Experimental Environment

Whereas the export module is implemented as P4$_{16}$ program built for the software switch behavioral model v2 (bmv2), the export agent and collector are run using Python scripts[1]. The CPU port of a switch instance is used for packet exchange between the implemented module and agent. Scapy library is utilized to build and send as well as to sniff and parse packets. The agent leverages a P4 runtime library for exposed thrift API and gRPC message interfaces to run module configuration and flow inactivity management over a TCP-based channel. UDP communication between the agent and collector is enabled using libraries socket and struct. Exported data is stored in files and subsequently loaded for analysis in separate scripts. While tcpreplay is used to inject packet traces into the switch for flow data tracking and export evaluation, iperf is used to intentionally generate flows required for evaluation measures.

[1]https://gitlab.cs.hs-fulda.de/flow-routing/cnsm2022/flow_monitoring

B. Experiments

1) Flow Export Datasets: Two network security datasets are considered for exemplary data export. First, a subset of CIC-IDS2017 dataset [10] is used (CIC-IDS). Besides benign packet data, malicious traffic corresponds to (Distributed) Denial of Service ((D)DoS), botnet and port scan attacks. Second, CIC-DOS2017 dataset [11] is employed (CIC-DOS). Packet data includes benign and DoS attack traffic.

For each dataset, data for supported scopes is exported from included sub-datasets having individual packet traces and afterwards merged into common sets. Hash table sizes $n_f = 2^{20}$ are used to ensure a low level of hash collisions. In case of collision, packet data is excluded from flow monitoring. The export timer w_E as well as the active and inactive timeouts w_A and w_I are set to 16 s while the subflow window size w_S is 4 s. These values are chosen to balance the analysis aspects of timely available data exports and decent flow lengths.

Table II lists obtained record counts for flow and biflow data views. Compared to tracking unidirectional packet streams, the combined view on interrelated bidirectional ones reduces the number of data records significantly. Since not all observed packet streams are bidirectional, the reduction ratio depends on the corresponding proportion. In addition, the number of flow and subflow as well as biflow and bisubflow data records are close due to similar tracking mechanisms.

TABLE II: Data Record Counts (in K samples).

CIC-IDS		**CIC-DOS**	
Flows	**Biflows**	**Flows**	**Biflows**
3 911	2 081	522	363

The ratio of occurred packet-based hash collisions remains low in all export iterations ($<0.1\,\%$). Because the majority of packet data is processed during flow monitoring, comprehensive data views are ensured for subsequent analysis.

2) Flow-based Traffic Classification Analysis: Data exports are used for flow-based intrusion detection. Therefore, traffic classification is treated as binary task differentiating between benign and malicious flows. To compare the classification performance associated with leveraging diversified data views, achieved accuracy scores are determined for each flow scope.

Exported data is labeled based on IP addresses and timestamps. After computed features are determined, flow 5-tuple data plus timestamps are excluded from classification to reduce the tendency for overfitting and allow for generalization.

RF and DNN classifiers are evaluated to consider two popular ML algorithms that may be of relevance for different traffic analysis tasks using flow metadata as decision input. scikit-learn and Keras libraries are used to build classifiers with exemplary hyper-parameters. While parameter sensitivity is abstracted, this paper does not aim at classifier optimization or comparison but rather on confirming the analysis benefits associated with diversified flow data scopes.

Tables III and IV summarize achieved accuracies. Result trends are similar for both classifiers and datasets but have varying dimensions. Although scores are already high for classifying flow data, except for one deviation, enlarged biflow and subflow views improve measured accuracies.

TABLE III: Classification Accuracy (RF, in %).

Dataset	Traffic Data Scope			
	Flows	Subflows	Biflows	Bisubflows
CIC-IDS	98.392	98.428	99.784	99.791
CIC-DOS	98.025	98.115	98.158	98.295

TABLE IV: Classification Accuracy (DNN, in %).

Dataset	Traffic Data Scope			
	Flows	Subflows	Biflows	Bisubflows
CIC-IDS	98.213	98.287	99.709	99.719
CIC-DOS	96.958	97.556	96.685	98.129

To further investigate the benefits of running subflow management during uni- and bidirectional data tracking, accuracy scores are determined while reducing the analysis basis to just include metadata for packet streams that saturate at least 1, 2 or 3 windows (Tables V to VII). Again, result trends apply to both datasets and classifiers with individual dimension.

TABLE V: Window-based Data Record Counts (in K samples).

Windows	CIC-IDS		CIC-DOS	
	Flows	Biflows	Flows	Biflows
1	1 238	661	127	78
2	1 002	525	85	54
3	182	98	43	30

First, the number of data samples included in the reduced datasets decreases for a larger window count. Second, except deviations, the higher the number of saturated windows, the higher the accuracy improvements for subflow management.

TABLE VI: Window-based Classification Accuracy (RF, %).

Dataset	Windows	Traffic Data Scope			
		Flows	Subflows	Biflows	Bisubflows
CIC-IDS	**1**	99.968	$+$ 0.018	99.959	$+$ 0.028
	2	99.968	$+$ 0.021	99.952	$+$ 0.037
	3	99.853	$+$ 0.104	99.757	$+$ 0.198
CIC-DOS	**1**	97.565	$+$ 0.430	97.380	$+$ 0.672
	2	97.317	$+$ 0.714	97.032	$+$ 1.045
	3	97.208	$+$ 1.331	96.837	$+$ 1.791

TABLE VII: Window-based Classification Accuracy (DNN, %).

Dataset	Windows	Traffic Data Scope			
		Flows	Subflows	Biflows	Bisubflows
CIC-IDS	**1**	99.848	$+$ 0.066	99.905	$+$ 0.011
	2	99.875	$+$ 0.053	99.895	$+$ 0.014
	3	99.285	$+$ 0.342	99.445	$+$ 0.095
CIC-DOS	**1**	95.398	$+$ 1.292	97.337	$+$ 0.085
	2	94.833	$+$ 1.498	96.993	$+$ 0.084
	3	93.814	$+$ 2.348	96.583	$+$ 0.182

Depending on increases or decreases for enlarging the uni- to a bidirectional view, i.e., using biflow instead of flow data scopes, subflow benefits for the bidirectional context may be higher or lower as for the unidirectional one.

As a consequence, leveraging wider data views enabled by running the proposed flow monitoring approach ensures improved attack detection rates, thus allowing to precisely differentiate between benign and malicious traffic. Although only slight accuracy improvements are achieved in some experiments, they are of high relevance, especially in the case of network intrusion detection. Even relatively low error rates may lead to many badly classified flows such that malicious packet streams remain undiscovered or benign ones are mistakenly affected by mitigation policy enforcement. Hence, seemingly high accuracy values close to or even above 99 % do not necessarily correspond to acceptable conditions.

3) Flow Export Overhead: The control path overhead between export module and agent w.r.t. processing times and volumes associated with supported flow scopes is estimated. Therefore, times to query data for flow inactivity management while using different hash table sizes (Table VIII) as well as to send export request packets and handle respective responses (Table IX) are measured. The lengths of these packets that include export metadata and data records are also given.

TABLE VIII: Query Times for Inactivity Management (in ms).

Hash Table Size	2^{12}	2^{14}	2^{16}	2^{18}	2^{20}
Average Time	43.76	133.81	482.45	1 716.29	6 781.69

The time to retrieve data from register arrays rises significantly for increasing hash table sizes. Each collection cycle involves gathering three register arrays, i.e., flow activity flags plus first and last seen packet timestamps, whereby the more slots are available, the more register objects have to be accessed in order to expose contained data via runtime APIs.

TABLE IX: Export Packet Processing Times (in ms).

Packet Type	Traffic Data Scope			
	Flows	Subflows	Biflows	Bisubflows
Export Request	0.47	0.48	0.47	0.48
Export Response	1.06	3.46	1.47	5.49

First, since the format and processing behavior for export requests is similar for all flow scopes, packets have a uniform length (19 bytes) and processing times are approximately constant. Second, due to different feature sets and record formats, the size of export response packets varies for diversified data views, i.e., 100, 252, 167 and 471 bytes for flow, subflow, biflow and bisubflow scopes, whereby 4 individual windows are exemplarily considered. Compared to flows, the export response sizes for subflows, biflows and bisubflows grow about 2.52×, 1.67× and 4.71× respectively.

In line with an increased data volume, processing times for export response packets rise w.r.t. different flow scopes. Compared to flows, on average, times for handling responses containing subflow, biflow or bisubflow data grow approximately 3.26×, 1.39× and 5.18×. As a consequence, while up to ≈2 100 export request packets can be sent per second, about 940, 290, 680 and 180 responses can be handled each second. Whereas these rates correspond to a single data processing instance, rates can be increased using parallelism at agent level running multiple workers concurrently. This is necessary to cope with export rates encountered in practical network environments. For example, ≈4 000 and ≈500 data records are exported per second at central switches in a university network at daytime peaks and during nighttime respectively [1].

4) Flow Latency Impact: The flow latency impact experienced due to flow monitoring-related packet processing overhead is determined. Measurements are run for all flow scopes while differentiating three tracking levels (feature groups): First, entirely skipping flow monitoring operation while running pure packet forwarding (none) serves as reference for comparison. Second, monitoring a flow's 5-tuple and first plus last seen timestamp data (basic) allows to estimate general tracking and export impact associated with, for example, flow-based hash computation, timer and timeout management or the evaluation of export conditions. Third, tracking all supported flow features (all) enables to measure the total overhead.

Because flow data tracking and most parts of export management, i.e., timers, condition checks and record construction, are run in the switch's ingress pipeline part, the difference between egress and ingress pipeline entry timestamps is computed and treated as an estimate to approximate packet processing times w.r.t. flow monitoring operation (Table X). Running iperf measurements to determine flow level latency impacts provides similar trends, although obtained values are slightly higher since the measurement scope is enlarged.

TABLE X: Flow Latency Impact (in ms).

Feature Group	Traffic Data Scope			
	Flows	Subflows	Biflows	Bisubflows
none	0.02	0.02	0.02	0.02
basic	0.84	0.89	0.90	0.95
all	0.92	1.06	0.94	1.28

As expected, the required time for packet forwarding without any flow monitoring operation is similar for all flow scopes (none). Using the flow data view as reference, enlarging the data scope is associated with latency overhead because biflow views require to track forward and backward direction in a combined manner and subflow as well as bisubflow ones to maintain windows on uni- and bidirectional packet streams. For each scope, the main increase is caused by general tracking and export management (basic) while maintaining pure feature data (all) adds slight delays as well, whereby the ones for subflow management are significantly higher compared to those for respective base views. Compared to flows, on average, overall latency rises about $1.15\times$, $1.02\times$ and $1.39\times$ for subflow, biflow and bisubflow scopes respectively.

5) Switch Memory Allocation: Table XI outlines memory requirements for maintaining a single data record for supported flow scopes. Because pure record data occupation is considered, additional overhead due to general export management like activity tracking and timeout evaluation is abstracted. Feature data is accounted as 8, 16, 32 or $48\,\mathrm{bit}$ values. While memory sizes stated for flows serve as reference, the values provided for subflows and biflows are relative to these whereas the ones for bisubflows are relative to biflow requirements. Overall, a subflow, biflow and bisubflow data record occupies about $2.90\times$, $1.84\times$ and $5.64\times$ the memory of a flow one.

TABLE XI: Data Plane Memory Requirements (in bit).

Flow Features	Traffic Data Scope			
	Flows	Subflows	Biflows	Bisubflows
flow 5-tuple data	2×32 (addresses) $+ 2 \times 16$ (ports) $+ 8$ (protocol)			
TCP flags mask	8	$\times 1$	$\times 2$	$\times 1$
TCP flag counts	8×16			
packet & byte count	2×32	$+\mathrm{n} \times 2 \times 32$		$+2 \times \mathrm{n} \times 2 \times 32$
packet lengths	3×32	$+3 \times \mathrm{n} \times 32$		$+3 \times 2 \times \mathrm{n} \times 32$
start & end timestamps	2×48	$\times 1$		$\times 1$
inter-arrival times	3×48	$+3 \times \mathrm{n} \times 48$		$+3 \times 2 \times \mathrm{n} \times 48$
overall (n = 4 windows)	640	$640 + \mathrm{n} \times 304$ (1 856)	1 176	$1176 + 2 \times \mathrm{n} \times 304$ (3 608)

For example, Intel Tofino 3 chip architectures support up to eight match-action pipelines each with $160\,\mathrm{Mbit}$ of SRAM, i.e., $1\,280\,\mathrm{Mbit}$ in total [12]. Since register and counter objects are used for data tracking, SRAM capacities are occupied. Using these capabilities as practical reference, about $250\,\mathrm{K}$, $86\,\mathrm{K}$, $136\,\mathrm{K}$ or $44\,\mathrm{K}$ flow, subflow, biflow or bisubflow data records can theoretically be maintained per pipe and approximately $2\,000\,\mathrm{K}$, $688\,\mathrm{K}$, $1\,088\,\mathrm{K}$ or $352\,\mathrm{K}$ entries on the entire switch platform. While reducing the feature set to packet and byte counters plus flow duration to describe flow activity at a basic level, tracking these features in conjunction with 5-tuple data causes flow, subflow, biflow and bisubflow record sizes of 264, 520, 424 and $936\,\mathrm{bit}$. Consequently, about $606\,\mathrm{K}$, $308\,\mathrm{K}$, $377\,\mathrm{K}$ or $171\,\mathrm{K}$ data records can be tracked per pipe and about $4\,848\,\mathrm{K}$, $2\,464\,\mathrm{K}$, $3\,016\,\mathrm{K}$ or $1\,368\,\mathrm{K}$ in the entire switch.

Running flow export on a core switch deployed in a data center of a university campus network, traffic analysis revealed that up to about $750\,\mathrm{K}$ data entries are maintained in caches during daytime while observed numbers drop below $100\,\mathrm{K}$ during nighttime and the weekend [1]. Using this environment as practical reference while taking entire switch and not pipeline level capabilities into account, tracking different flow scopes with the aforementioned reduced feature set is feasible. The same applies to flow and biflow data views if tracking all features but might be challenging for sub- and bisubflow ones.

VI. Discussion

`FlowMoni` protocol enables demand-driven data export: First, a collector registers at an exporter and requests the data scope and features required for downstream analysis apps. Second, data is preprocessed at exporter and collector levels, whereby steps running on a switch assist in collector-specific data preparation prior to sharing. Moreover, `FlowMoni` architecture allows to run traffic analysis within a switch's NOS and in an SDN controller. Whereas the first enables to distribute analysis load across the network, the latter enables more sophisticated decisions due to increased analysis capabilities.

Because tracking and export behavior is based on timers and timeouts to ensure data export, their specification defines the accepted trade-off between timely available data records and their granularity while also affecting the overhead with respect to, e.g., export rate and data volume. First, the higher the active timeout w_A, the higher the time until data records for long-lasting packet streams are exported and, due to aggregated or average properties, the lower the granularity. In addition, the number of exported data records decreases accordingly. Second, the higher the inactive timeout w_I, the longer the time until streams are assumed inactive and data export can be initiated while, except for recurring streams, the number of exported data records is the same. Third, the timer w_E defines the interval at which records for inactive streams are requested, whereby the higher the value, the higher the export delay and the more bursty the export rate w.r.t. single iterations.

Enlarging the data tracking scope increases the memory overhead and thus lowers the number of available monitoring slots. Though a single bidirectional data record occupies about double the memory required for a unidirectional one, the accumulated overhead is approximately equal. This holds true if most packet streams are bidirectional. Otherwise, the ratio of both contexts decides the amount of occupied but unused memory. Maintaining subflow views for uni- and bidirectional contexts occupies significantly more memory, whereby the used window length and resulting number of individual windows affects both memory demand and the resolution of tracked data trends. The smaller w_S, the higher the granularity although more windows have to be maintained and vice versa.

As hashing is used to identify flows, collisions occur and explicit handling is necessary to ensure consistent traffic views.

Due to extended packet processing, flow monitoring is associated with additional delay that has to be accepted in order to gain traffic insights. Since respective latency increases may not be a valid option for all application types, property- and hash-based filtering ensure differentiated flow monitoring. Therefore, particular tracking scopes can be mapped to specific applications or monitoring operation skipped entirely. This way, available data views are intentionally limited or packet streams fully excluded from traffic analysis.

For all supported flow scopes, main latency increases are caused by general data export management whereas each tracked feature adds slight delay as well but the estimated sum for pure tracking of all considered properties is considerably lower compared to the first. As expected, measured latencies depend on the tracked flow scope and grow with enlarged data views due to higher packet processing overhead, whereby times for subflow management are significantly higher compared to ones for respective flow and biflow scopes.

The control path overhead in terms of data volume and processing times with respect to exchanged export packets including data records rises for flow scopes with broader data views. While this is due to larger message sizes and more complex processing behavior, there is a high dependency on implementation quality and the employed switch architecture. First, since packets are shared between exporter module and agent at data plane and NOS level respectively, packet sniffing and manipulation are required to provide efficiency and scalability to cope with high export packet rates in dynamic network environments. Second, agent level query times for retrieving data plane register and counter objects required to run flow inactivity management as well as associated data volumes highly depend on the exposed runtime APIs, whereby the higher the number of supported monitoring slots, the higher the query overhead. One direction for optimization is to move query operation to lightweight raw packet exchange, hence decoupling inactivity management from any runtime API.

Latency and control path overhead measurements provide estimates. In practice, there is a high dependency on the used switch platform with significant differences between virtual, software-based deployments relying on CPU performance and real systems equipped with hardware chips. Nonetheless, relative result trends from the experiments are transferable though their dimension and related absolute measures vary.

Although a diversified data scope is associated with an increased overhead, more sophisticated traffic analysis decisions are enabled. Though the flow classification performance achieved in the network intrusion detection scenario is already quite high for leveraging flow data views, using biflow data scopes that combine forward and backward contexts raises accuracy scores. Likewise, more granular sub- and bisubflow data views allow for further accuracy improvements. This way, an enhanced differentiation between benign and malicious traffic is ensured, which helps to considerably reduce false positive and negative rates. Even slight accuracy increases associated with data views from enlarged flow scopes are of high relevance because otherwise occurring network attacks are not revealed and a significant proportion of benign and malicious packet streams is affected by improper flow control.

VII. Conclusion and Future Work

`FlowMoni` runs data tracking and export to collect flow-based traffic views serving as decision input for subsequent data analysis tasks like ML-supported flow classification or prediction. Whereas an export module and assisting agent are deployed on data plane and NOS level to perform flow monitoring in a collaborative manner, obtained data records are pushed to collectors with downstream analysis apps. Since these are run in a switch's NOS or at an SDN controller, distributed and lightweight as well as centralized and intensive tasks are enabled. This architecture offers a new way to analyze diversified traffic contexts and granularities at different network layers, either in an isolated or combined manner, i.e., multi-step analysis. Therefore, an adaptive protocol including various data preprocessing steps while respecting four different flow scopes with enlarged data views and flexible feature sets ensures demand-driven data sharing. The considered data scope affects both analysis benefits and monitoring overhead.

`FlowMoni` is planned to be ported on a hardware switch run in a university network to analyze practical suitability.

References

[1] C. Hardegen, B. Pfülb, S. Rieger, A. Gepperth, and S. Reißmann, "Flow-based Throughput Prediction using Deep Learning and Real-World Network Traffic", *IEEE International Conference on Network and Service Management*, 2019.

[2] C. Hardegen, M. Petersen, C. Ezelu, T. Geier, S. Rieger, and U. Bühler, "A Hierarchical Architecture and Probabilistic Strategy for Collaborative Intrusion Detection", *IEEE International Conference on Communications and Network Security*, 2021.

[3] C. Hardegen, B. Pfülb, S. Rieger, and A. Gepperth, "Predicting Network Flow Characteristics using Deep Learning and Real-World Network Traffic", *IEEE Transactions on Network and Service Management*, 2020.

[4] J. Hill, M. Aloserij, and P. Grosso, "Tracking Network Flows with P4", *IEEE/ACM Innovating the Network for Data-Intensive Science*, 2019.

[5] L. Castanheira, R. Parizotto, and A. E. Schaeffer-Filho, "FlowStalker: Comprehensive Traffic Flow Monitoring on the Data Plane using P4", *IEEE International Conference on Communications*, 2019.

[6] J. Sonchack, E. Keller, A. J. Aviv, and J. M. Smith, "TurboFlow: Information Rich Flow Record Generation on Commodity Switches", *ACM EuroSys Conference*, 2018.

[7] J. Sonchack, O. Michel, A. J. Aviv, E. Keller, and J. M. Smith, "Scaling Hardware Accelerated Network Monitoring to Concurrent and Dynamic Queries With *Flow", *USENIX Annual Technical Conference*, 2018.

[8] B. Claise, "Cisco Systems NetFlow Services Export Version 9", *Request for Comments 3954*, 2004.

[9] B. Claise, B. Trammell, and P. Aitken, "Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of Flow Information", *Request for Comments 7011*, 2013.

[10] I. Sharafaldin, A. H. Lashkari, and A. Ghorbani, "Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization", *International Conference on Information Systems Security and Privacy*, 2018.

[11] H. H. Jazi, H. Gonzalez, N. Stakhanova, and A. A. Ghorbani, "Detecting HTTP-based Application Layer DoS Attacks on Web Servers in the Presence of Sampling", *Computer Networks*, 2017.

[12] "Intel Tofino 3 Intelligent Fabric Processor Brief", intel.com/content/www/us/en/products/network-io/programmable-ethernet-switch/tofino-3-brief.html.