Balancing Information Preservation and Data Volume Reduction: Adaptive Flow Aggregation in Flow Metering Systems

Adrian Pekar, Laszlo A. Makara, Winston K. G. Seah, and Oscar Mauricio Caicedo Rendon

***Abstract*—The critical role of network traffic measurement and analysis extends across a range of network operations, ensuring quality of service, security, and efficient resource management. Despite the ubiquity of flow-level measurement, the escalating size of flow entries presents significant scalability issues. This study explores the implications of adaptive gradual flow aggre- gation, a solution devised to mitigate these challenges, on flow information distortion. The investigation maintains flow records in buffers of varying aggregation levels, iteratively adjusted based on the changing traffic load mirrored in CPU and memory utilization. Findings underscore the efficiency of adaptive gradual flow aggregation, particularly when applied to a specific buffer, yielding an optimal balance between information preservation and memory utilization. The paper highlights the particular significance of this approach in Internet of Things (IoT) and contrasted environments, characterized by stringent resource constraints. Consequently, it casts light on the imperative for adaptability in flow aggregation methods, the impact of these techniques on information distortion, and their influence on network operations. This research offers a foundation for future studies targeting the development of more adaptive and effective flow measurement techniques in diverse and resource-limited network environments.**

***Index Terms*—adaptive computing, gradual flow aggregation, flow measurement, data reduction, performance optimization**

I. INTRODUCTION

As the digital age progresses, networks become increasingly complex, connecting myriad devices and applications. Each of these applications and services possesses unique requirements, making the task of managing and understanding network traffic of paramount importance. This management hinges on effective *traffic measurement* and *analysis*. The significance of this task goes beyond mere data monitoring. It provides essential data for operations such as SLA compliance evaluation, QoS provisioning, intrusion detection, and traffic management [1].

The predominant approach to understanding network traffic is through *flow level* analysis [2]. Conceptually, a "flow" represents a series of packets, moving from a source to a destination, that share some common attributes. It is a foundational concept that serves as the basis for the more complex methodologies discussed in this paper. However, as networks expand in complexity and size, grappling with the *expanding magnitude of flow records* becomes an uphill task. Several strategies aim to manage this via *adaptive aggregation of flow records*, wherein flow records are dynamically consolidated based on certain parameters to streamline the data without losing significant detail.

However, as traffic volumes surge, particularly under heavy loads, the very technique of adaptive flow aggregation can alter the richness and reliability of traffic information. This distorted information can directly impede the efficiency of network operations, making it crucial to understand flow aggregation's true implications. Particularly, with the growing emphasis on the Internet of Things (IoT) and similar network environments where resources are often constrained, understanding this balance becomes critical. As edge computing starts to overshadow cloud-based solutions, especially in the IoT landscape, ensuring minimum resource consumption while maintaining data richness becomes indispensable.

Given this backdrop, our research delves into the realm of adaptive flow aggregation, probing its effects on flow size distortion—a critical metric determining the reliability of traffic management. Specifically, our exploration involves analyzing flow records organized across buffers with varying degrees of aggregation, which are dynamically generated based on traffic loads reflected through CPU and memory utilization.

One of our pivotal observations reveals that employing adaptive flow aggregation can achieve a desirable trade-off. Specifically, a certain degree of aggregation resulted in a marginal information loss of just 2.42%, a compromise that significantly optimizes resource utilization. This insight is particularly vital in settings like IoT where there is a pressing need to maximize data integrity while operating within constrained resources. The distinguishing contributions of this paper encompass:

(*i*) A deep dive into how differential aggregation levels, particularly in light of fluctuating traffic volumes, influence the fidelity of flow information.

(*ii*) The introduction of the SEE indicator, a novel metric to quantify information biases induced by flow aggregation. SEE provides a rigorous gauge of estimation errors in flow size—a cornerstone for gauging the repercussions of adaptive aggregation on flow data.

A. Pekar and L. A. Makara are with the Department of Networked Systems and Services, Faculty of Electrical Engineering and Informatics, Budapest University of Technology and Economics, Budapest, Hungary.

A. Pekar is also with the HUN-REN-BME Information Systems Research Group, Budapest, Hungary.

W. K. G. Seah is with the School of Engineering and Computer Science, Victoria University of Wellington, Wellington, New Zealand

O. M. Caicedo Rendon is with the Department of Telematics Engineering, Engineering Telematics Group, Universidad del Cauca, Popayán, Colombia

Corresponding author: A. Pekar (e-mail: apekar@hit.bme.hu)

DOI: 10.36244/ICJ.2023.3.9

(*iii*) The exploration into the aftereffects of adaptive flow aggregation on flow size distortion, enriched by SEE's integration.

Our research accentuates the pressing need for truly adaptive flow aggregation methodologies, underscoring their implications on data fidelity and overall network operations. We aspire that our insights catalyze the development of refined, adaptive flow measurement techniques tailored for diverse and resource-challenged network settings.

The rest of this paper is structured as follows: Section II provides a succinct background on flow measurement, aggregation, and its adaptive variants. Section III reviews relevant literature in the domain. In Section IV, we outline our research design, highlighting our novel gradual aggregation scheme, adjustment strategies, preliminaries, and measurement techniques. Section V presents the findings of our study, analyzed to enhance comprehension of our approach. Discussions on these results are in Section VI, while Sections VII and VIII delve into their broader significance. Finally, Section IX offers reflections on our research.

II. BACKGROUND

This section provides a brief yet comprehensive overview of network traffic flow aggregation in a context relevant to the scope of this paper. From exploring the dynamics of network traffic flow, through presenting an overview of the process of flow aggregation, to considering the principle of adaptive flow aggregation, this section sets the stage for the subsequent methodology of our study.

A. The Dynamics of Network Traffic Flow

Presently, the most prevalent method for network measurement is the collection of traffic data at the flow level, commonly known as flow export [3]. The term *flow* denotes a group of packets that possess a shared *key* and pass through a specific observation point within a determined period [4]. This shared *flow key* is typically characterized by a five-element tuple, including the source and destination IP addresses, source and destination ports, and the protocol.

Traffic data, such as *flow features*—for instance, all packets related to a specific flow quantified in bytes—and the corresponding flow keys, often designated as *flow properties*, are contained in the *flow records*. Network management tasks rely on the analysis of these flow records, where the most crucial data is procured from the packet headers encapsulated in the flow features. Analyzing these records includes descriptive calculations like determining the minimum, mean, standard deviation, and maximum of the flow sizes along with the packet inter-arrival time statistics.

In a conventional flow-based measurement scenario, the flow records traverse through several platform components positioned at different layers, as illustrated in Figure 1. The procedure typically involves organizing the captured packets into flows in a *flow cache*, after which they are intermittently exported to a data collector. The collector then either stores the data in a data store or forwards them directly for further analysis and visualization [2].

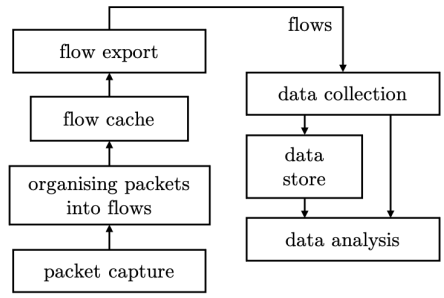


Fig. 1. General architecture of a flow measurement platform.

However, the enormous quantity of measurement data presents several challenges [5] as they traverse through the platform's components, right from packet capture to data analysis. Making sense of the intricate, ever-expanding data for operational purposes is a daunting task. The processing essential for both online and offline analysis poses considerable challenges, often due to a lack of sufficient resources to manage flow-related tasks efficiently, accurately, and sustainably [6].

B. The Impact of Tuple Variations on Potential Flow Entries

In the realm of IP flow measurement, the flow cache plays a pivotal role by aggregating packets into flows using various combinations of a five-tuple. This aggregation results in flow entries, which provide a comprehensive view of network traffic. The nature and number of these entries can vary based on the specifics of the tuple in use and the practicalities of network operations.

The five-tuple used for flow measurement in IP networking typically consists of the following:

1) *Source IP Address*: IPv4 addresses have a 32-bit length, leading to 2^{32} possible addresses. IPv6 addresses, on the other hand, have a 128-bit length, leading to 2^{128} possible addresses.
2) *Destination IP Address*: Same as above.
3) *Source Port*: There are 65,536 possible port numbers (ranging from 0 to 65,535).
4) *Destination Port*: Same as the source port.
5) *Protocol*: The IP protocol field is 8 bits, so there are 2^8 or 256 possible values. However, not all 256 values are used in practice. Common protocols include TCP, UDP, and ICMP. For simplicity, we use the full range of 256 possible values.

In Table I, the potential combinations of these elements showcase the vast number of unique flow entries one could encounter in IP networking. Evidently, the magnitude of possible combinations is colossal, especially when factoring in the expansive address space of IPv6. However, as more fields from the five-tuple are excluded, the number of combinations decreases substantially.

It is important to note that these figures represent a theoretical maximum. In practice, IP flow measurement on a specific network segment does not capture all these combinations because of the inherent nature of network design and purpose,

TABLE I
THE TOTAL NUMBER OF POSSIBLE TUPLES FOR DIFFERENT COMBINATIONS

Tuple Combination	IPv4	IPv6
Src IP, Dst IP, Src Port, Dst Port, Proto	1.1×10^{28}	1.2×10^{79}
Dst IP, Src Port, Dst Port, Proto	1.1×10^{21}	1.5×10^{43}
Src Port, Dst Port, Proto	1.1×10^{13}	1.1×10^{13}
Dst Port, Proto	1.7×10^{7}	1.7×10^{7}
Proto	256	256

hence observing only a subset of potential traffic patterns. This highlights the nuanced and comprehensive scope of flow measurements in IP networks, drawing a distinction between theoretical possibilities and real-world observations.

C. The Process of Flow Aggregation

Flow aggregation serves as a popular method to streamline the handling of measurement data. It primarily aims to combine several underlying measurement data points into a single, unified record. The merging process is governed by properties whose values may remain static or fluctuate over time. It should be noted that flow records are themselves representations of aggregated traffic, with flow features compiled from information sourced from packet headers.

However, flow aggregation stands as a higher abstraction level where the properties of multiple flow records are synthesized according to specific criteria, yielding a more concise representation of the original data. This aggregation results in consolidated flow records that maintain the overall traffic properties and characteristics, albeit with a broader granularity and diminished detail.

While aggregation invariably results in flow records with lesser information richness about network traffic, the information loss incurred is substantially less than that observed in sampling-based methods [7]. In certain traffic dynamics, these methods can overlook entire flows, typically those composed of a solitary packet or very few packets. This limitation has led to the development of existing flow aggregation techniques [8], [9] which employ partial aggregation of flow records deemed non-essential. The process is facilitated by *gradual flow key reduction*—that is, a step-by-step reduction in the count of flow properties that function as flow keys—in order to safeguard crucial information and preserve accuracy.

However, without the careful calibration of aggregation intensity, achieving the appropriate granularity of flow-level information becomes challenging. Ensuring optimal levels of aggregation becomes essential in maintaining the delicate balance between data manageability and the preservation of pertinent information.

D. The Principle of Adaptive Flow Aggregation

Adaptive systems leverage iterative adjustments to their parameters based on predefined criteria to ensure that the system operates optimally or as close to optimal conditions as possible. Adaptive aggregation adheres to a similar principle—modifying aggregation according to one or more criteria to enhance system operation. Such criteria could include the flow size (measured in bytes), CPU or memory utilization by the flow measurement process, link bandwidth utilization of the capture device, or a combination thereof.

For instance, if flow aggregation is adjusted in response to CPU utilization, the number of processed packets can be perceived as directly proportional to the CPU utilization by the flow metering tool. When the traffic volume increases, the count of captured packets also escalates, subsequently intensifying the resource utilization of flow measurement. In contrast, a decline in traffic leads to a decreased packet count, thereby reducing the associated resource utilization. This reciprocal relationship allows the definition of different network traffic load levels—such as low, moderate, and high. These levels serve as markers for estimating how the traffic load is registered by flow measurement, enabling the adjustment of flow aggregation accordingly.

Adaptive flow aggregation proves beneficial as it applies a relatively less destructive degree of aggregation contingent upon the release of additional resource capacity. As a result, more informational value is preserved when fluctuations in resource capacity permit. This advantage is particularly notable in the context of constrained IoT operations, which often face limitations in processing capability and memory. Furthermore, as recent advancements in IoT solutions demonstrate a trend towards transitioning computational logic from the cloud to the edge [10], the need for low resource consumption while preserving information value is increasingly crucial.

Contrarily, traditional flow aggregation with static operation introduces a consistent level of distortion to flow-level information, irrespective of resource availability. This can result in unavoidable information loss during memory overflow events. Notably, although adaptive flow aggregation employs more aggressive merging under high loads, it helps to maintain information integrity even when the flow cache overflows (i.e., when the memory is entirely filled). Despite the fact that this approach might result in lower information granularity in the aggregated flow records, it ensures no information is lost in overflow situations. Furthermore, the process of gradual flow key reduction is specifically designed to mitigate the erosion of relevant information.

III. RELATED WORK

The domain of network management has witnessed extensive research, but only a few studies have specifically honed in on flow aggregation. An earlier work, Aguri, offers a distinct approach to aggregation-centric traffic profiling suitable for real-time, long-term, and wide-area traffic monitoring [9], [11]. Unlike conventional methods that rely on predefined filter rules to classify traffic types, Aguri aggregates low-volume flows until they become distinctly identifiable. This ensures even minor traffic types are not overlooked. With its capacity to generate concise profiles spanning source and destination addresses and protocols, Aguri adeptly monitors traffic, spots anomalies, and counters threats like DDoS attacks.

Diving deeper into the nuances of flow aggregation, Cheng *et al.* [12] proposed an Aggregation Flow Measurement

(AFM). This scheme reconfigures traffic clusters leveraging the quintessential five fields of a fine-grained flow. Central to this method is dynamic sampling that adeptly recalibrates in response to traffic shifts. Instead of direct packet value recordings, it banks on estimates, optimizing CPU resource allocation. Adding finesse to this approach is a secondary process that prioritizes heavy-tailed flows during flow information updates. This dual-pronged strategy ensures rich information capture and impeccable estimation accuracy while judiciously pruning smaller flows to streamline flow cache.

In assessing these methodologies, it is clear that adaptation has been progressively factored into flow aggregation techniques. While Aguri offers a unique perspective on traffic profiling without considering real-time adaptation, AFM brings resource utilization, specifically CPU dynamics, into the fold. However, packet sampling entails a loss of granularity, potential inaccuracies, challenges in rare event detection, inconsistent accuracy across flows, and a lapse in bursty traffic information capture. The added layer of flow sampling in AFM aggravates these issues as flows, once organized, could be counterproductively discarded.

Gradual Flow Key Reduction by Irino *et al.* [8] presents a more refined approach. It orchestrates flows based on predetermined criteria—like the volume of transported octets in descending order. Only flows surpassing a user-defined significance threshold are retained; the rest undergo iterative aggregation. This proactive strategy curtails network congestion and safeguards upper-tier platform components, as visualized in Figure 1. However, despite its valuable insights, this method does not fully embrace 'adaptation.' It misses reacting to pivotal aspects such as resource utilization by measurement tools or the fluidity of network traffic. Undoubtedly, there is still a broader spectrum of criteria, like the full range of network traffic dynamics, that might further refine the granularity and efficiency of flow-level data processing.

In a notable shift, recent studies increasingly integrate flow aggregation with Heavy Hitter (HH) flow detection [13], [14]. Central to this detection is a threshold demarcating HHs from their non-HH counterparts. Recognizing the challenges posed by anomalous traffic patterns on measurement tools, Hu *et al.* [15], [16] introduced a dynamic strategy utilizing adaptive flow aggregation. At the heart of the methodology is the insight that a significant proportion of network attacks manifest as non-HH patterns, predominantly generating a plethora of short-lived flows. The primary application of this method finds relevance in scenarios like Denial of Service (DoS) attacks that target the same destination IP address, and worm attacks with identical source IP addresses. In response to these situations, their approach combines a two-dimensional hash table structure with a tiered clustering system. Once sorted, the traffic within these clusters is aggregated into metaflows, mitigating issues related to memory and export bandwidth, while preserving the accuracy of legitimate flows.

Pekar *et al.* [17] introduced a technique inspired by Gradual Flow Key Reduction but extended its application to the domain of HH detection. This method tailors the aggregation of flow records to match the characteristics of the traffic and the specific objectives of the monitoring process. For instance, this could include anomaly detection or flow-based accounting of transferred data volumes. In this scheme, HHs are maintained in the flow cache, while non-HHs are aggregated in dedicated buffers, following a hierarchy defined by flow key precedence. However, the work examines the adaptive aggregation of flow records exclusively through the prism of reduction efficacy in relation to HH detection, thereby also neglecting to explore the performance efficacy, particularly in relation to information distortion and resource consumption.

Building upon the theme of HH detection in software-defined data center networks, Bi *et al.* [18] sculpted a dynamic threshold, examining the nexus between elephant thresholds and network traffic dynamics in data centers. Drawing parallels with the optimal receive system of baseband signal transmission, they unearthed an equilibrium between positive and negative false rate detections by studying the overlap of two flow probability distribution curves.

In a similar vein, Wang *et al.* [19], [20] differentiated flows into short and long categories, applying a fluid threshold. The unique twist in their approach is the distinct management of these flows: short flows are shepherded by distributed algorithms, while long ones are entrusted to centralized solutions. This method is further enhanced by capitalizing on end-hosts for precise flow tagging and curtailing overhead through centralized algorithms.

Lastly, Liu *et al.* [21] leveraged the Dynamical Traffic Learning algorithm. This tool facilitates real-time dynamic configuration of threshold values, ensuring swift and efficient identification of HH flows with minimal latency and overhead.

Pivoting from flow record data to table rule management, Saha *et al.* [22] hones in on flow table rule aggregation. Focusing on QoS, flow table entry aggregation, and IoT, their adaptive scheme reduces flow-rules volume without sacrificing IoT traffic's QoS. A key-based mechanism empowers user choice over OpenFlow match-fields. Balancing QoS path selection and switch flow-table use, the "Best-fit" heuristic adaptively selects a QoS path to minimize network flow-rules.

Navigating the intricate orchestration of the small flows emblematic of modern mobile core networks and IoT, Minh *et al.* [23] introduces the 'flow tree', a controller-based binary search tree-like structure mirroring the OpenFlow switch table. By exploiting the wildcard of a flow's dstIP, the method manages the tree responsively to network shifts, resulting in a flow table that is more efficient and trimmer than standard OpenFlow tables.

Taking a step further, Phan *et al.* [24] developed a mechanism to optimize traffic flow monitoring in SDN-based networks. It adjusts flow table entries in SDN switches based on the detailed traffic information required by systems such as intrusion detection or traffic engineering. Instead of a fixed threshold, the method uses a machine-learning algorithm to determine optimal entry limits. This continual assessment ensures SDN switch performance remains optimal.

Lastly, in an endeavor to streamline table occupancy in SDNs, Jia *et al.* [25] showcases a flow-table aggregation

strategy. Through dynamic address and port rewriting, the method aggregates multiple same-destination flows from varied sources into a singular flow entry. This approach drastically reduces core-layer SDN switch table occupancy, proving effective even in environments with dispersed IP address allocations. The method can operate in both software-defined IPv4 and IPv6 networks, though it does not provide explicit adaptability features.

In reflection, while foundational groundwork in flow measurement has been laid by prior research, there is a distinct evolution from static criteria to more dynamic, HH detection-driven approaches. Nevertheless, predominant focus areas have been network security and measurement data reduction. The overarching challenge has been to comprehensively blend adaptability, precision in aggregation, and nuanced flow handling. The landscape, enriched by these advancements, still beckons for solutions that holistically address the multifaceted challenges of modern network environments.

Our study steps into this gap with an innovative approach. Through the adaptive multi-buffer flow measurement strategy, we aim to navigate beyond the traditional confines of prior research. Our methodology fuses adaptability with structured aggregation, ensuring that flow data is managed judiciously and resourcefully. By doing so, we intend to shed light on the impact of adaptive flow aggregation on information bias and flow measurement instrumentation, providing a roadmap for improved network traffic management. In essence, our approach endeavors to set a new course in extracting meaningful insights from flow data, even in environments marked by resource variability. To our knowledge, this paper is the first effort in bridging this particular domain gap.

IV. METHODOLOGY

This section delves into our methodology designed to study adaptive gradual flow aggregation. We pay special attention to the aggregation and adaptation techniques, foundational assumptions, primary propositions, and our evaluative metrics.

A. Gradual Flow Aggregation Scheme

Anchoring our strategy is the *Gradual Flow Key Reduction* mechanism [8]. This technique systematically and progressively aggregates flow data, trimming flow key elements as data navigate a sequence of buffers. It pivots on two fundamental components: the *flow key precedence* and the *multi-buffer structure*.

1) Flow Key Precedence: The linchpin of our strategy is the preordained *Flow Key Precedence*. This order determines the hierarchy of flow key element reduction. As we traverse each level of reduction, we witness flow records of differing granularity—starting from the most detailed, progressively becoming more aggregated.

2) Multi-buffer Structure: Our multi-buffer structure embodies a delicate equilibrium between data granularity and resource constraints, setting forth a well-considered data reduction route. Under the assumption of a standard five-tuple flow key precedence as 'protocol' $>$ 'src_port' $>$ 'dst_port' $>$ 'src_ip' $>$ 'dst_ip', we detail the flow granularity across buffers as follows:

- *Main Flow Cache (Buffer B0)*: Retains the entire flow key set, guaranteeing maximum information fidelity.
- *Buffer B1*: Initiates aggregation by removing 'dst_ip', leaving behind 'protocol', 'src_port', 'dst_port', and 'src_ip'.
- *Buffer B2*: Proceeds further by excluding 'src_ip', encompassing only 'protocol', 'src_port', and 'dst_port'.
- *Buffer B3*: Excludes 'dst_port' next, safeguarding 'protocol' and 'src_port'.
- *Buffer B4*: At its apex of aggregation, only the 'protocol' remains.

Even though rooted in the predominant five-tuple, our methodology displays inherent adaptability. We ensure that the buffer configuration aligns with the diversity of flow key elements, thereby providing versatility for various flow definitions.

3) Gradual Flow Aggregation Operation: The aggregation cycle commences by targeting the lowest-ranking flow key. Resultant flow records, housed in buffer B_1, present the least coarse granularity. Subsequent aggregation then zeroes in on the next flow key, now the lowest-ranked of the survivors. The records emanating from this second aggregation tier are stationed in buffer B_2, presenting a marginally coarser granularity view.

This iterative process persists through subsequent flow keys, ushering in an incremental level of flow granularity with every pass. The culmination is the aggregation of the highest-ranking flow key, where the consequent flow records—residing in buffer B_4—capture the coarsest granularity snapshot.

In summation, the Gradual Flow Aggregation strategy adeptly manages data granularity, striking a harmonious balance between storage efficiency and insightful flow data richness.

B. Enhanced Gradual Flow Aggregation: A Refined Approach

Our proposed methodology optimizes the original Gradual Flow Aggregation scheme, introducing structure, adaptability, and improved data management. Central to this refinement is a multi-buffer structure reminiscent of the original method. However, what sets our method apart is its disciplined, sequential flow aggregation. Contrary to the original method, which allowed flow records the potential freedom to transition directly from the primary buffer to the last, our approach ensures flow records aggregate over only one flow key element when transitioning between consecutive buffers. This mandates a smoother, less abrupt data reduction, leading to enhanced information preservation.

Our methodology also integrates an improved threshold mechanism. Each buffer possesses its specific threshold, which can be consistent or varied across buffers. This threshold serves as a deciding factor, determining the flows to retain or those designated for aggregation. Moreover, by adjusting this mechanism to be responsive to resource utilization, we achieve a delicate balance between preserving pertinent flow details and efficiently managing resources.

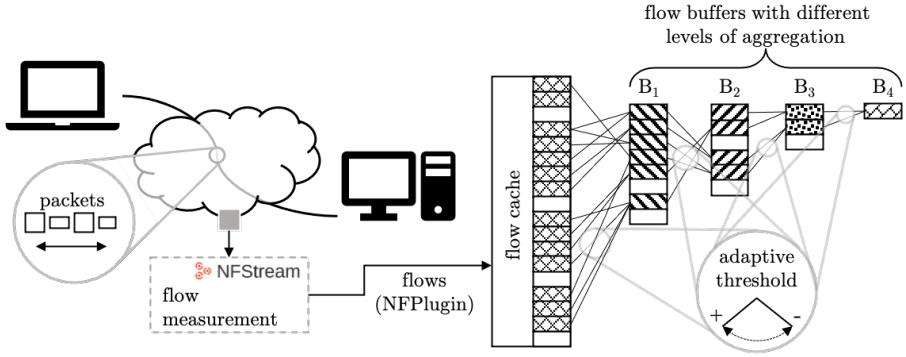


Fig. 2. Workflow from flow measurement achieved by NFStream to adaptive gradual flow aggregation implemented as an NFPlugin.

A notable innovation is the provision to redirect overflowing flows to subsequent buffers when a buffer reaches its capacity. This ensures that the aggregation process remains structured, even under significant network traffic demands. Furthermore, the last buffer, B4, is designed to capture all possible network flows, thus offering a holistic aggregation framework.

In summary, our enhanced methodology, characterized by its tiered flow key element reduction and dynamic threshold mechanism, not only minimizes the risk of sudden data reductions but also provides a more resource-efficient, organized, and information-rich flow aggregation process compared to its predecessor.

C. Proof-of-Concept Implementation

Our methodology for enhanced gradual flow aggregation was built upon the NFStream framework [26], a Python-based tool engineered for rapid, flexible, and expressive data handling.

NFStream's robustness stems from its hybrid design, blending Python's accessibility with the speed of C. At the heart of this design is the NFlow structure, defined in C. This structure represents a network flow, encapsulating core attributes such as the five-tuple flow key, and is further enriched with additional flow statistics and metadata. These metadata are organized into multiple categories, including core features, L7 visibility, post-mortem statistics, and SPLT details. Users can dynamically toggle these feature sets on or off during flow measurement, offering a bespoke network analysis environment. Each of these features is stored in a specific variable type, tailored to accommodate its potential maximum value size.

NFStream not only simplifies the transition from raw network measurements to refined data science analytics due to its core functions in flow measurement and feature computation but also shines with its extendable architecture. This extensibility, particularly evident in the integration of custom network functionalities via the NFPlugin component [26], allowed us to embed our adaptive gradual flow aggregation directly as an NFPlugin. This integration was facilitated by the use of four distinct buffers (B_1-B_4), which collaboratively operate alongside NFStream's primary flow cache, as illustrated in Figure 2. These buffers are designed to retain flow records with varying granularities.

D. Resource Utilization-driven Adaptability

Our approach to flow aggregation is adaptive, with the degree of aggregation varying in accordance with changes in resource utilization by the flow meter. The number of processed flows has been found to be directly proportional to resource utilization [27]. Essentially, an increase in network traffic, leading to a growing number of flows, results in a concurrent rise in CPU and memory load. Conversely, a reduction in traffic, and thus the number of flows, leads to a decrease in resource utilization. Assuming that the temporal aspect of the flow meter's resource utilization aligns with its flow cache load, we designed our aggregation process to reflect the changes of its CPU and memory usage.

With the above in mind, our methodology involves ongoing monitoring of CPU and memory utilization. This yields a set of n observations denoted as $\mathcal{O} = o_t, o_{t+1}, \ldots, o_{t+n}$ $(t = 0, 1, \ldots, k)$. Each observation is the average CPU and memory usage at time t, or $o_t = (cpu_t + memory_t)/2$. CPU time and memory usage are combined in our methodology to provide a more comprehensive measure. Analyzing these two metrics separately might overlook scenarios where one resource is heavily utilized while the other is not, resulting in a misleading representation of the overall system load. By combining these metrics, a more holistic and accurate assessment of the current state of system resource utilization is obtained. Furthermore, each observation, o_t, is expressed as a percentage, representing the proportion of the total available CPU and memory resources that are currently being used. Therefore, the value of o_t in our approach can range from 0 to 100%.

The average resource utilization, represented as σ, is then calculated using the following formula:

$$\sigma = \begin{cases} \dfrac{\sum_{i=k-w}^{k} o_{t+i}}{w}, & \text{if} \quad k \geq w. \\ \dfrac{\sum_{i=0}^{k} o_{t+i}}{k}, & \text{otherwise.} \end{cases} \tag{1}$$

The sliding window of size w is employed to ensure that our assessment of average resource utilization is up-to-date and sensitive to recent changes, by considering only the last

w observations. However, at the beginning of our monitoring process, it may take some time to accumulate w observations. The second case in the equation accounts for this scenario, where the average resource utilization is calculated using all available observations, instead of just the last w. Once we have collected at least w observations, the first case will be used to calculate the average resource utilization, effectively implementing the sliding window approach.

Achieving adaptability involves comparing average resource utilization (σ) between consecutive times, $t-1$ and t. The aggregation threshold is adjusted based on the difference between σ_{t-1} and σ_t. This can be formally expressed using the following formula:

$$T_{t+1} = \begin{cases} T_t + (|\sigma_{t-1} - \sigma_t|)\% \text{ of } T_{t-1}, & \text{if } \sigma_{t-1} < \sigma_t. \\ T_t - (|\sigma_{t-1} - \sigma_t|)\% \text{ of } T_{t-1}, & \text{if } \sigma_{t-1} > \sigma_t. \\ \emptyset, & \text{otherwise.} \end{cases} \quad (2)$$

where

T_{t+1} = signifies the aggregation threshold for time $t+1$;
T_t = represents the currently employed aggregation threshold at time t;
T_{t-1} = refers to the aggregation threshold used at time $t-1$;
σ_{t-1} = designates the previous average resource utilization as measured by CPU and memory load at time $t-1$;
σ_t = indicates the current average resource utilization as measured by CPU and memory load at time t.

Equation (2) ensures that the aggregation threshold is regularly fine-tuned in accordance with variations in traffic load as manifested in CPU and memory utilization. As such, resource utilization near its maximum will trigger more aggressive flow aggregation, while lower resource utilization will result in a lesser degree of flow aggregation.

The degree of aggregation is based on a threshold T. Flows that transfer bytes ($flow_{bytes}$) equal to or exceeding T are preserved in their original state. Conversely, flows that transfer fewer bytes than what T prescribes are gradually aggregated within the buffers.

Aggregation commences once a buffer reaches its maximum capacity. Prior to adding a flow entry into the flow cache, the current buffer's capacity is assessed. Should it be at full capacity, the aggregation procedure is invoked. Flows adhering to the adaptively adjusted threshold remain intact within their existing buffer. Conversely, flows falling short of the threshold undergo aggregation. This entails discarding the flow key of the underlying buffer before relocating the flows to said buffer. This process is consistent across all buffers: each time a buffer reaches its limit, aggregation ensues, relegating flows to subsequent buffers characterized by diminished information retention capacities.

E. Assumptions

Our experimental evaluation is anchored on several foundational assumptions.

1) *Hierarchy of Flow Features*: Building upon the guidance of [8] and [17], we delineated a hierarchy for flow features, sequenced from highest to lowest precedence as: *protocol*, *source port*, *destination port*, *source IP*, and *destination IP*. This structure aims to replicate a genuine scenario, ensuring that flows in the concluding buffer (B_4) can be differentiated based purely on their protocol identifiers.
2) *Aggregation and Flow Movement*: We postulate that aggregation happens solely within each discrete buffer, with the flow cache being an exception. Flow records transition from the primary flow cache towards the buffer B_4, as illustrated in Figure 2. This directional progression of records guarantees that transporting a record from the flow cache to B_1 is more cost-efficient than shifting records between buffers B_3 and B_4. This movement approach is optimized to curtail information loss, especially for flow records with significant byte transfer. To further impede information loss during synchronization, a flow record's relocation is restricted to one move per aggregation cycle.
3) *Adaptive Thresholding*: The adaptation mechanism hinges on the threshold T, dictated by the overall data volume in the flow, denoted as $flow_{bytes}$. Our prototype harnesses a universal threshold influencing all buffers, encompassing the flow cache. Future studies should probe into the advantages of individual thresholds for each buffer.
4) *Sliding Window Size*: For our prototype evaluation, we fixed the sliding window size at 10 elements. This choice is underpinned by thorough examination of diverse settings and the resultant outcomes. Yet, more investigation is warranted to gain deeper insights into the ramifications of varied window sizes.
5) *Memory Consumption and Flow Record Size*: Our results are simplified with the assumption that each flow record consumes 21 bytes, given by the size of the flow key. The memory consumption of flow keys in NFStream are:
   - Source & Destination IPv4 Address: 8 bytes each
   - Source & Destination IPv6 Address: 16 bytes each
   - Source & Destination Port: 2 bytes each
   - Protocol Identifier: 1 byte

   This culminates in a memory consumption of 21 bytes for IPv4 flow records and 37 bytes for IPv6 flow records. For instance, storing 1 million IPv4 flow records would require approximately 20.96 MB. Nonetheless, for actual memory allocations, the results should be multiplied by the size of the complete feature set, computed as the sum of all feature sizes. Factoring in the 86 flow features NFStream can measure amplifies the memory requirements, emphasizing the need for effective memory management in IP flow measurement.
6) *TCP Flow Record Aggregation*: During the aggregation phase for TCP flow records, TCP flags are omitted. This is because these flags, being intrinsically flow-specific, lose their informational significance upon aggregation, rendering them redundant in the aggregated context.

F. Information Bias in Adaptive Flow Aggregation

Through the process of aggregation, the information value of flow records diminishes, and this effect intensifies with increasingly aggressive flow aggregation, a consequence of surging resource utilization. Nevertheless, the impact on traffic management-related activities is direct and substantial. Information bias, in this context, refers to the distortion of flow size data resulting from the aggregation process. As the flow records are aggregated and the granularity of the information reduces, the interpretation of these records can become less precise, or "biased".

Consider the varied sensitivities of different applications to this information bias. For example, application type classification and DoS attack detection are highly sensitive to information bias. In application type classification, a detailed breakdown of the flow is essential to accurately identifying the type of application from its network patterns. Similarly, DoS attack detection relies heavily on identifying anomalies or spikes in individual flows, which can be obfuscated by aggregated data.

On the other hand, certain applications might be less affected by this bias. Accounting applications, for instance, often focus on the overall data transfer volume, which can be accurately measured even with aggregated flow data. Load balancing, too, is typically concerned with overall network utilization across multiple routes or servers, rather than the details of individual flows, and thus may remain largely unaffected by the aggregation process.

Therefore, understanding the effects of adaptive gradual flow aggregation on the value of flow size information, particularly in relation to the saturation of flow cache size, is critical. It is a balancing act between maintaining network performance and retaining the precision necessary for certain network tasks. Quantifying how flow size-based aggregation across the buffers $B_1 - B_4$ contributes to information bias can provide key insights that improve network traffic management. It is worth noting that while 'information bias' may not be a universally recognized term in this context, it serves effectively to describe the phenomenon discussed here.

G. Dataset Preparation and Ground Truth Establishment

Our study utilized the UNIV1 dataset [28], [29], a publicly accessible traffic trace collected from a university campus data center. The dataset encompasses a wide array of services, including system backups, distributed file system hosting, e-mail servers, web services, and multicast video streams.

Organization of packets into forward, backward, and bidirectional flows was carried out using NFStream, with the passive and active expiration of flows set to NFStream's default of 120 and 1800 seconds, respectively. The resultant dataset comprised a total of 468,905 IP flows distributed across 14 distinct protocols. Specifically, UDP, TCP, and other protocols (such as ICMP, IGMP, EGP, IGP, ESP, and AH) accounted for 270,028, 196,305, and 2,572 flows respectively.

The CDF of flow sizes is displayed in Figure 3. Detailed analysis revealed that around 85% of flows were smaller than 10 kB, with the majority of the data (20 kB and larger)

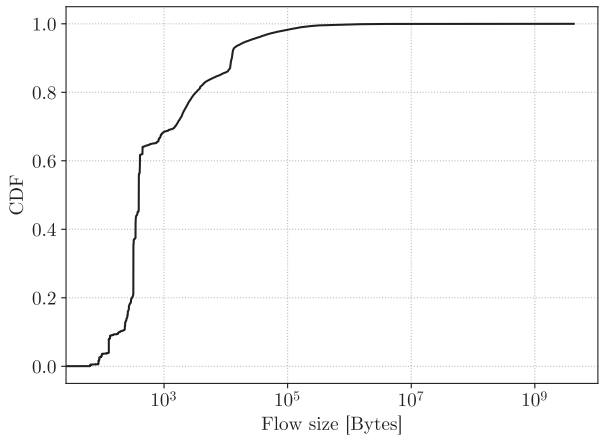


Fig. 3. Flow Size Distribution.

concentrated in the top 5% of flows. Interestingly, less than 10 packets generated roughly 70% of the flows, while a mere 2% of all flows originated from more than 100 packets. We also discovered that about 14% of all flows in the UNIV1 dataset transferred data larger than 10 kB in size. This observation is consistent with findings in [29], which reported that 10% of the flows transport the majority of the traffic. The slight difference between our measurements and these findings can be attributed to variations in flow metering methodologies and the flow expiration timeout used.

Given our observations and the significant influence of flow size on our adaptive aggregation implementation, we selected the transmitted number of bytes in the flow ($flow_{bytes}$) as the primary metric for our evaluation. Consequently, we established the flow size threshold at 13.4 kB, which has been recognized as the optimal threshold for the UNIV1 dataset concerning flow size [30]. This threshold served as the benchmark for our evaluation.

H. Quantifying Information Bias with Size Estimation Error

To study the impact of adaptive gradual flow aggregation on information bias, we introduce Size Estimation Error (SEE). The SEE is a measure of the error in the estimated flow size for flows within the buffers (B_1-B_4) at the conclusion of each measurement interval, relative to the memory size m_s.

1) Definition of SEE: SEE serves as an indicator of the average number of flows merged within the buffers and the relative change in their sizes compared to their original dimensions. For any given buffer i (B_i), we calculate SEE as:

$$S\hat{E}E_i = |B_i| \times (\frac{1}{4} \times i). \tag{3}$$

In this equation, $S\hat{E}E_i$ is a preliminary estimation of the Size Estimation Error for buffer i. We employ the hat (ˆ) notation to signify a preliminary value or estimate within the statistical framework.

2) Normalization of SEE and Definition of SEE Vector: To acquire a normalized SEE (SEE_i) that provides a relative estimation error for each buffer, we normalize each $\hat{SEE}_i$ as follows:

$$SEE_i = \frac{\hat{SEE}_i}{\sum_{j=1}^{4} \hat{SEE}_j}. \tag{4}$$

Each SEE_i now represents the normalized size estimation error for buffer i, providing a proportionate share of the total estimation error.

We then compile these normalized size estimation errors into a single vector, referred to as the SEE vector:

$$SEE = (SEE_1, SEE_2, SEE_3, SEE_4). \tag{5}$$

We operate under the premise that the sum of all SEE_i values is 1, and each SEE_i is equal to or greater than 0, i.e., $\sum_{i=1}^{4} SEE_i = 1, \quad SEE_i \geq 0$.

3) Adaptive Gradual Flow Aggregation and SEE Implementation: Our implementation of adaptive gradual flow aggregation encompasses two potential scenarios for transferring flows from a lower-ranked buffer B_n to a higher-ranked buffer B_{n+1}. In striving for a realistic environment, we set the memory limit of each buffer to 10,000 bytes, aligning with the parameter settings proposed by [14].

The first scenario comes into play when the memory reaches saturation, necessitating memory clearance. Consequently, the adaptive gradual flow aggregation relocates flows into lower-ranked buffers, with a particular focus on moving the smallest-sized flows between buffers.

In the second scenario, the system iteratively examines the flows in buffers $B_1 - B_4$ based on the size of the $flow_{bytes}$ parameter (*cf.* Section IV-D). The flows are then tagged and moved in accordance with the actual threshold T. As the system nears the memory limit, it begins to reposition the flow records relative to the current system utilization, adhering to the adaptive gradual flow aggregation scheme discussed in Section IV-A.

4) Evaluation of Information Bias: After distributing all the flows across buffers $B_1 - B_4$, we contrast the results to quantify the information bias induced by adaptive gradual flow aggregation on the flow size feature. Importantly, our method retains all necessary metadata for reverting aggregated flows over the buffers back to their original form, enabling an accurate efficacy evaluation.

V. RESULTS

This section delineates the quantitative ramifications of adaptive gradual flow aggregation. Initially, it assesses the parameters of a system operating under constraints. Subsequently, it explicates the influence of adaptive flow aggregation in a multi-buffer system arrangement.

A. Bounds of a Constrained System Operation

Gauging the necessities and constraints for full-fledged, dependable flow metering presents a formidable challenge. In spite of only a fraction of the traffic being measured, the network invariably houses more devices than the metering system. This culminates in a vast discrepancy between the resources available for traffic generation and those for traffic measurement, leading to a pronounced resource asymmetry. To quantify this asymmetry, our initial step involved examining the bounds for constrained operation of our prototype implementation devoid of active adaptive gradual flow aggregation.

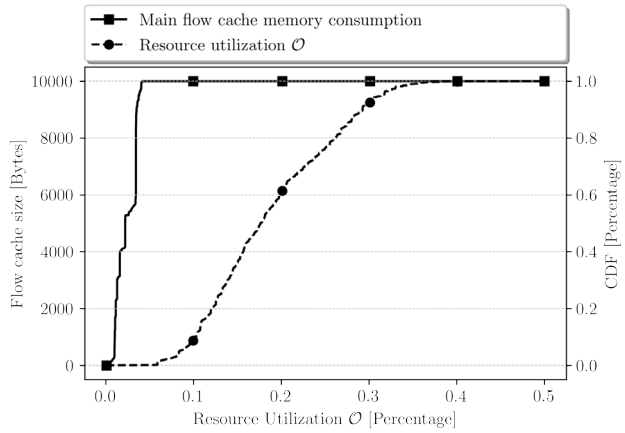


Fig. 4. Correlation between flow occupancy and resource utilization $\mathcal{O}$.

Figure 4 visualizes the correlation between per-buffer flow occupancy size and resource utilization, denoted as $\mathcal{O}$. The left y-axis signifies buffer memory usage in bytes, the right y-axis represents the distribution of flow sizes via CDF, and the x-axis denotes resource utilization $\mathcal{O}$, which comprises memory and CPU loads. The system's behaviour is depicted in Figure 4 through two curves: (*i*) the distribution of flow occupancies relative to $\mathcal{O}$, marked using rectangular markers, and (*ii*) the CDF relative to $\mathcal{O}$, indicated using dot markers.

Figure 4 reveals that a surge in the number of flow records in the flow cache directly leads to a significant increase in resource utilization $\mathcal{O}$, which is composed of memory and CPU loads. Consequently, the maximum memory cap of 10,000 bytes is attained at a resource utilization as low as 4.81%. As the observation of additional flows sustains the memory load at 100%, the combined resource utilization $\mathcal{O}$ promptly escalates beyond 40%. This translates to both a subpar operation of the system and the dropping or discarding of packets and flows. We also noted that there is a 9.88% chance of observing 10% $\mathcal{O}$ at a memory load of 1,420 bytes.

This implies that memory saturation, regardless of flow creation and maintenance, substantially elevates the $\mathcal{O}$ parameter of a given device. Therefore, flow meters ought to be calibrated with respect to memory limits to prevent reaching the saturation level. In situations where such conditions are unattainable, adaptive gradual flow aggregation can aid in guaranteeing a reliable system operation.

B. Implications of Adaptive Gradual Flow Aggregation

Adaptive gradual flow aggregation is conducted relative to the actual memory load, capped at 10,000 bytes, and CPU utilization. This leads to the distribution of flows among indi-

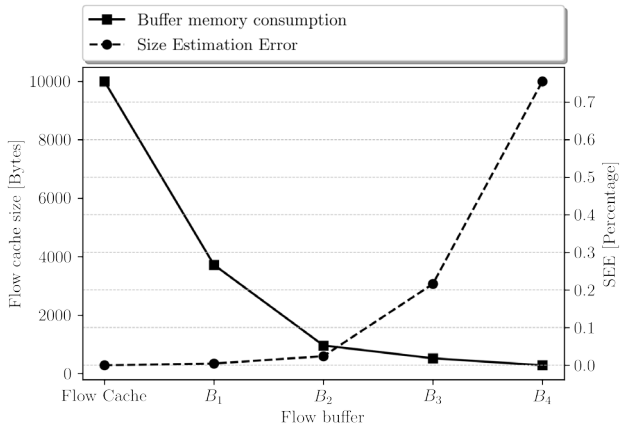


Fig. 5. Correlation between the size of per-buffer flow occupancy and SEE.

vidual buffers. With this in mind, we explored SEE resulting from this distribution. Accordingly, packets in the UNIV1 were arranged using adaptive gradual flow aggregation, and the buffer status was examined post-operation. Note that, for the evaluation of this approach's operational conditions, flow expiration causing flows to be flushed from the cache(s) was disregarded.

Figure 5 illustrates the correlation between the size of per-buffer flow occupancy and SEE. The left y-axis signifies buffer memory usage in bytes, the right y-axis represents SEE, and the x-axis denotes the flow buffers. Figure 5 portrays the system behavior via two curves: (*i*) the distribution of flow occupancies between the buffers, marked using rectangular markers, and (*ii*) the SEE for each buffer, indicated using dot markers.

Figure 5 reveals that the buffers' occupancy by flows decreased logarithmically due to gradual flow reduction. The most substantial reduction occurred in buffer B_4, which aggregates flows over the flow key with the highest rank—the protocol identifier (*cf.* Section IV-A). Quantitatively, the SEE amounted to 75.49%, signifying a considerable information loss of all the UNIV1 flows. Nevertheless, this occurred at the cost of a significant reduction factor, implying all flows moved into this buffer were aggregated to yield 14 entries, each 21 bytes in size (UNIV1 contains 14 unique protocols, as discussed in Section IV-G).

Shifting towards a configuration devoid of adaptive gradual aggregation, the ratio discernibly shifts in favor of preserving information value, albeit at the expense of deteriorated memory utilization. Empirically, buffer B_3, which aggregates UNIV1 flows over the flow keys with the second-highest rank, yielded an SEE of 21.63%, consuming 540 bytes of buffer memory. Buffer B_2 presented a further improvement, with an SEE of 2.42% at a memory utilization of 960 bytes. Finally, aggregation over buffer B_1 resulted in an SEE of 0.46%, occupying $3{,}840$ bytes of memory.

In light of the above, adaptive gradual flow aggregation over B_2 appears to offer the most optimal balance, providing an excellent compromise between preserving maximum information value and minimal memory utilization, as demonstrated in Figure 5. Buffer B_2 aggregates flow records over the flow key with the second-lowest rank among all flow keys, thereby the information contained in the aggregated flow records has a less fine level of granularity compared to complete flow records, inevitably leading to information loss. Quantitatively, this approach aggregated $20{,}434$ flows in B_2 from a total of $468{,}905$ flows in UNIV1, translating to a loss of 2.42%. This could be considered a reasonable trade-off for enhanced resource utilization. Furthermore, the extent of loss can be mitigated by bolstering the physical hardware, specifically the allocated memory. Nevertheless, adaptive gradual flow aggregation significantly improved the system operation under limited constraints. It can aid in achieving optimal resource utilization while retaining the maximum information value embedded in the flow records.

VI. DISCUSSION

The adaptive gradual flow aggregation mechanism presented in this research study aims to optimize the balance between memory utilization and information value preservation in the context of network flow metering. The results shed light on the implications of adopting this mechanism, particularly in scenarios where the system is resource-constrained.

In a context where there exists a significant asymmetry between traffic generation and traffic measurement resources, our adaptive flow aggregation technique comes into play. It addresses the problem by efficiently distributing flows across multiple buffers, a fact underpinned by our results that highlight the performance of the mechanism under constrained operation.

Equally important is the role of SEE, a measure introduced to quantify the error in flow size estimation. Through the use of SEE, we can understand and assess the trade-off between memory utilization and information preservation. Indeed, our findings show that, as the level of flow aggregation increases, SEE increases too, which translates to greater information loss. However, it is noteworthy that the amount of information loss varies significantly across buffers, with some buffers offering an optimal trade-off.

Our results revealed that saturation, irrespective of the occurrence of flow creation and maintenance, causes the resource utilization parameter $\mathcal{O}$ of a given device to rise significantly. This underlines the importance of appropriately configuring flow meters concerning memory bounds to prevent reaching the saturation level. In scenarios where this is not feasible, adaptive gradual flow aggregation can ensure reliable system operation.

It was also observed that the application of adaptive flow aggregation resulted in a reduction in the total number of flow records in the cache, with a corresponding decrease in the SEE. The interplay between memory utilization and information preservation became apparent, illustrating the trade-offs involved. The buffer occupancy, and consequently the SEE, decreased logarithmically due to gradual flow reduction. While this led to a loss in information, it also enhanced system performance by significantly reducing memory usage.

An intriguing finding was the optimal balance achieved with adaptive gradual flow aggregation over buffer B_2. This offered an excellent trade-off between maximum information value and minimal memory utilization. Buffer B_2 aggregates flow records over the flow key with the second-lowest ranking among all flow keys, implying the information contained in the aggregated flow records has a coarser level of granularity compared to complete flow records. This led to an inevitable, yet acceptable, information loss. Nevertheless, with enhanced physical hardware or increased allocated memory, the degree of this loss can be minimized, hinting at potential avenues for improvement.

Overall, the results demonstrate the effectiveness and potential of our proposed adaptive gradual flow aggregation mechanism. Despite the inherent information loss, the mechanism significantly improves system operation under limited constraints and resource asymmetry. It achieves optimal resource utilization while striving to retain the maximum possible information value in the flow records. As such, it presents a promising solution for environments constrained by memory and processing resources.

VII. Implications

The absence of a systematic data reduction strategy, while ensuring full preservation of information value, incurs higher operational costs and risks a potential crash of the measurement instrument as resource capacity reaches its limits. On the other hand, the application of gradual flow aggregation reduces the size of flow measurement data, thus lowering operational costs, but inevitably distorts flow-level information. Our study reveals that the operation of such a system can be further optimized for better resource utilization through adaptability.

From a traditional viewpoint, achieving optimal operation without adaptive flow aggregation seems impracticable as the dual objectives of preserving information value and reducing data volume are essentially mutually exclusive. Moreover, optimizing for either of these objectives leads to increased operational costs. In contrast, adaptive flow aggregation enables system operation optimization by balancing between information preservation and data volume reduction. The goal of optimization, therefore, is to determine the aggregation threshold for a given buffer n, given factors such as flow record creation rate, buffer capacity, memory utilization, and CPU utilization. The aim is to minimize the volume of flow records while maximizing the preservation of information in the flow records.

Our results show that optimizing system operation through preliminary observation-based configurations can already yield a more compact data volume while preserving considerable information value, all without incurring additional operational costs. This operation can be enhanced by solving the optimization problem to determine the optimal threshold per buffer n, including the flow cache. However, further research is necessary to establish the efficacy of algorithms designed to solve this optimization problem. These investigations could provide additional insight into the interplay between information preservation, data volume reduction, and resource utilization in flow metering systems.

VIII. Broader Applications and Future Directions

Given the versatility and profound implications of network monitoring, our study extends beyond merely reducing data volume to encompass a broad spectrum of applications:

1) Quality of Service: Effective flow aggregation is instrumental in accurately monitoring service levels. By ensuring network services consistently meet their designated quality parameters, we can enhance SLA adherence and elevate the overall network experience for users.

2) IoT and Edge Computing: The inherent constraints of IoT devices and the trend towards edge computing highlight the indispensability of adaptive flow aggregation. In these settings, striking a balance between data richness and resource efficiency is of paramount importance.

3) Software-Defined Networking: In the realm of SDNs, it is essential to fine-tune flow table rules and guarantee efficient load distribution. Our adaptive aggregation approach offers a promising avenue for honing rule sets and optimizing traffic management.

4) Attack Detection: Leveraging adaptive flow aggregation allows for the efficient clustering of disparate anomalous patterns. This capability can significantly bolster intrusion detection systems and fortify network security frameworks.

Peering further into areas like attack detection and QoS, we discern the potential for synergizing our adaptive flow aggregation techniques with advancements in AI and machine learning. Such a fusion could usher in predictive flow management, seamlessly navigating the dichotomy between preserving information and reducing data volume. Moreover, we envision subsequent research endeavors delving into various contexts and proposing algorithmic solutions to the optimization challenge we have underscored. Crafting these solutions would enhance the potency of adaptive flow aggregation, optimizing flow metering systems, especially in resource-limited environments.

IX. Conclusion

In this study, we examined adaptive gradual flow aggregation as a methodology for coping with resource asymmetry in flow metering systems. Our results illuminated the practical implications of adaptive flow aggregation and highlighted the inherent trade-off between memory utilization, CPU load, and information preservation. By exploiting this trade-off, we found that adaptive gradual flow aggregation could help achieve a desirable balance between resource utilization and the maintenance of maximum information value within the flow records. In particular, we identified that buffer B_2 offered the most optimal balance, yielding a compromise loss of 2.42%, which we consider acceptable in the context of improved resource utilization.

Overall, our study underscores the potential of adaptive gradual flow aggregation to improve resource utilization in flow metering systems while preserving vital flow-level information. Adhering to open science principles, we have made the scripts used in our experiments publicly accessible[1]. We

[1] https://github.com/FlowFrontiers/AGFA

believe this will facilitate a thorough comprehension of our methodologies and encourage additional investigation in this domain.

ACKNOWLEDGEMENT

This work was supported by the János Bolyai Research Scholarship of the Hungarian Academy of Sciences. Supported by the ÚNKP-23-5-BME-461 New National Excellence Program of the Ministry for Culture and Innovation from the source of the National Research, Development and Innovation Fund. The work presented in this paper was supported by project no. TKP2021-NVA-02. Project no. TKP2021-NVA-02 has been implemented with the support provided by the Ministry of Culture and Innovation of Hungary from the National Research, Development and Innovation Fund, financed under the TKP2021-NVA funding scheme.

REFERENCES

[1] B. Li et al., "A survey of network flow applications," *Journal of Network and Computer Applications*, vol. 36, no. 2, pp. 567–581, 2013. DOI: 10.1016/j.jnca.2012.12.020.

[2] R. Hofstede et al., "Flow monitoring explained: From packet capture to data analysis with netflow and ipfix," *IEEE Communications Surveys Tutorials*, vol. 16, no. 4, pp. 2037–2064, 2014. DOI: 10.1109/COMST.2014.2321898.

[3] S. Bauer et al., "On the evolution of internet flow characteristics," in *Proceedings of the Applied Networking Research Workshop, ser.* ANRW '21, Proceedings of the Applied Networking Research Workshop, 2021, pp. 29–35. DOI: 10.1145/3472305.3472321.

[4] P. Velan, "Improving network flow definition: Formalization and applicability," in *NOMS 2018 – 2018 IEEE/IFIP Network Operations and Management Symposium*, NOMS 2018 – 2018 IEEE/IFIP Network Operations and Management Symposium, 2018, pp. 1–5. DOI: 10.1109/NOMS.2018.8406203.

[5] A. Dainotti, A. Pescape, and K. C. Claffy, "Issues and future directions in traffic classification," *IEEE Network*, vol. 26, no. 1, pp. 35–40, 2012. DOI: 10.1109/MNET.2012.6135854.

[6] S. Lee, K. Levanti, and H. S. Kim, "Network monitoring: Present and future," *Computer Networks*, vol. 65, pp. 84–98, 2014. DOI: 10.1016/j.comnet.2014.03.007.

[7] S. Dong and Y. Xia, "Network traffic identification in packet sampling environment," *Digital Communications and Networks*, 2022. DOI: 10.1016/j.dcan.2022.02.003.

[8] H. Irino, M. Katayama, and S. Chaki, "Study of adaptive aggregation on ipfix," in *Proceedings of the 7th Asia-Pacific Symposium on Information and Telecommunication Technologies*, Proceedings of the 7th Asia-Pacific Symposium on Information and Telecommunication Technologies, 2008, pp. 86–91. DOI: 10.1109/APSITT.2008.4653545.

[9] K. Cho, R. Kaizaki, and A. Kato, "Aguri: An aggregation-based traffic profiler," in *Quality of Future Internet Services*, M. I. Smirnov et al., Eds., Quality of Future Internet Services, 2001, pp. 222–242. DOI: 10.1007/3-540-45412-8_16.

[10] L. Kong et al., "Edge-computing-driven internet of things: A survey," *ACM Comput. Surv.*, 2022, Just Accepted. DOI: 10.1145/3555308.

[11] K. Cho, R. Kaizaki, and A. Kato, "An aggregation technique for traffic monitoring," in *Proceedings 2002 Symposium on Applications and the Internet (SAINT) Workshops*, 2002, pp. 74–81. DOI: 10.1109/SAINTW.2002.994556.

[12] G. Cheng and J. Gong, "Adaptive aggregation flow measurement on high speed links," in *Proceedings of the 11th IEEE Singapore International Conference on Communication Systems (ICCS)*, 2008, pp. 559–563. DOI: 10.1109/ICCS.2008.4737246.

[13] K.-c. Lan and J. Heidemann, "A measurement study of correlations of internet flow characteristics," *Computer Networks*, vol. 50, no. 1, pp. 46–62, 2006. DOI: 10.1016/j.comnet.2005.02.008.

[14] V. Sivaraman et al., "Heavy-hitter detection entirely in the data plane," in *Proceedings of the Symposium on SDN Research*, ser. SOSR '17, Proceedings of the Symposium on SDN Research, 2017, pp. 164–176. DOI: 10.1145/3050220.3063772.

[15] Y. Hu, D.-M. Chiu, and J.-S. Lui, "Adaptive flow aggregation – a new solution for robust flow monitoring under security attacks," in *10th IEEE/IFIP Conference on Network Operations and Management Symposium*, ser. NOMS '06, 10th IEEE/IFIP Conference on Network Operations and Management Symposium, 2006, pp. 424–435. DOI: 10.1109/NOMS.2006.1687572.

[16] Y. Hu, D. M. Chiu, and J. C. S. Lui, "Entropy based adaptive flow aggregation," *IEEE/ACM Transactions on Networking*, vol. 17, no. 3, pp. 698–711, 2009. DOI: 10.1109/TNET.2008.2002560.

[17] A. Pekar et al., "Adaptive aggregation of flow records," *Computing and Informatics*, vol. 37, no. 1, pp. 142–164, 2018. DOI: 10.4149/cai_2018_1_142.

[18] C. Bi et al., "On precision and scalability of elephant flow detection in data center with SDN," in *Proc. 32nd IEEE Global Communications Conf. Workshops*, ser. GLOBECOM'13, 2013, pp. 1227–1232. DOI: 10.1109/GLOCOMW.2013.6825161.

[19] S. Wang et al., "Fdalb: Flow distribution aware load balancing for datacenter networks," in *2016 IEEE/ACM 24th International Symposium on Quality of Service (IWQoS)*, 2016, pp. 1–2. DOI: 10.1109/IWQoS.2016.7590409.

[20] S. Wang et al., "Flow distribution-aware load balancing for the datacenter," *Computer Communications*, vol. 106, pp. 136–146, 2017. DOI: 10.1016/j.comcom.2017.03.005.

[21] Z. Liu et al., "An adaptive approach for elephant flow detection with the rapidly changing traffic in data center network," *Int. J. of Network Management*, vol. 27, no. 6, e1987, 2017, e1987 nem.1987. DOI: 10.1002/nem.1987.

[22] N. Saha, S. Misra, and S. Bera, "Qos-aware adaptive flow-rule aggregation in software-defined iot," in *2018 IEEE Global Communications Conference (GLOBECOM)*, 2018, pp. 206–212. DOI: 10.1109/GLOCOM.2018.8647471.

[23] Q. T. Minh et al., "Flow aggregation for sdn-based delay-insensitive traffic control in mobile core networks," *IET Communications*, vol. 13, no. 8, pp. 1051–1060, 2019. DOI: 10.1049/iet-com.2018.5194.

[24] T. V. Phan et al., "Destination-aware adaptive traffic flow rule aggregation in software-defined networks," in *2019 International Conference on Networked Systems (NetSys)*, 2019, pp. 1–6. DOI: 10.1109/NetSys.2019.8854510.

[25] W.-K. Jia and X. Wang, "Flow aggregation for large-scale sdns with scattered address space allocation," *Journal of Network and Computer Applications*, vol. 169, p. 102 787, 2020. DOI: 10.1016/j.jnca.2020.102787.

[26] Z. Aouini and A. Pekar, "Nfstream: A flexible network data analysis framework," *Computer Networks*, vol. 204, p. 108 719, 2022. DOI: 10.1016/j.comnet.2021.108719.

[27] Y. Fu et al., "Jellyfish: Locality-sensitive subflow sketching," in I*EEE INFOCOM 2021 - IEEE Conference on Computer Communications*, IEEE, 2021, pp. 1–10. DOI: 10.1109/INFOCOM42981.2021.9488847.

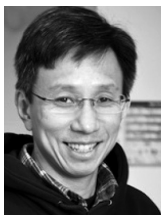
[28] T. Benson, *Data set for IMC 2010 data center measurement*, University of Wisconsin-Madison, 2010. https://pages.cs.wisc.edu/~tbenson/IMC10_Data.html.

[29] T. Benson, A. Akella, and D. A. Maltz, "Network traffic characteristics of data centers in the wild," in *Proc. 10th Internet Measurement Conf.*, ser. IMC '10, Proc. 10th Internet Measurement Conf., 2010, pp. 267–280. DOI: 10.1145/1879141.1879175.

[30] A. Pekar et al., "Knowledge discovery: Can it shed new light on threshold definition for heavy-hitter detection?" *Journal of Network and Systems Management*, vol. 29, no. 3, p. 24, 2021. DOI: 10.1007/s10922-021-09593-w.

Adrian Pekar received the Ph.D. degree in computer science from the Technical University of Košice, Slovakia, in 2014. Currently, he is a Senior Researcher with the Department of Networked Systems and Services, Budapest University of Technology and Economics, Hungary. Prior to this, he held research, teaching, and engineering positions in Slovakia and New Zealand. His research interests include network and services management, software-defined networking, network function virtualization, and cloud computing.

Laszlo A. Makara earned his MSc degree from the Department of Networked Systems and Services at the Budapest University of Technology and Economics, Hungary, in 2023. Presently, he is pursuing his PhD at the same department, delving deeper into the world of computer engineering and research. His research is primarily focused on network and services management, software-defined networking, and network programmability.



Winston K. G. Seah received the Dr. Eng. degree from Kyoto University, Kyoto, Japan, in 1997. He is currently a Professor of network engineering with the School of Engineering and Computer Science, Victoria University of Wellington, New Zealand. Prior to this, he has worked for more than 16 years in mission-oriented industrial research, taking ideas from theory to prototypes, most recently, as a Senior Scientist with the Institute for Infocomm Research, Singapore. He has been actively involved in research in the areas of mobile ad hoc and sensor networks and co-developed one of the first QoS models for mobile ad hoc networks. His latest research interests include IoT, mobile edge computing, SDN, network anomaly detection, and 5G ultra reliable low latency and machine- type communications.

Oscar M. Caicedo Rendon (GS'11-M'15-SM'20) is a full professor at the University of Cauca, Colombia, where he is a member of the Telematics Engineering Group. He received his Ph.D. degree in computer science (2015) from the Federal University of Rio Grande do Sul, Brazil, and his M.Sc. in telematics engineering (2006) and his degree in electronics and telecommunications engineering (2001) from the University of Cauca. His research interests include network and service management, NFV, SDN, and Machine Learning for Networking.