ReFeR: Resource Critical Flow Monitoring in Software-Defined Networks

Yihui Qian*, Yutong Liu*, Linghe Kong*, Minyou Wu*, and Shahid Mumtaz†
*Shanghai Jiao Tong University, Shanghai, China
†Instituto de Telecomunicações, Losboa, Portugal
Email: {yh_tsien, isabelleliu, linghe.kong, mwu}@sjtu.edu.cn, smumtaz@av.it.pt

***Abstract*—Flow monitoring is widely applied in software-defined networks for monitoring network performance. Especially, the detection on heavy hitters can prevent the Distributed Denial of Service attack. However, many existing approaches fall in one of two undesirable extremes: (i) inefficient collection where only accuracy is concerned in the method; (ii) low accuracy caused by the sacrifice with fast detection. As a result, we aim to find a balance between the accuracy and efficiency of flow monitoring, where the network resources can be saved and the error rate can also be confined simultaneously.**

In this paper, we present ReFeR, a novel "Report-Feedback-Report" scheme to improve the detection efficiency of heavy item detecting while ensuring low error rate of the measurement. ReFeR leverages the binary order of magnitude of item measurement to replace the long statistical information shared between switches and controller; after the items are analyzed with the magnitude, only fewer uncertain items are involved in further detection, where their information (i.e., significant digits) is provided for final judgment.

Theoretical analysis and simulated evaluation have proved the effectiveness of our solution. ReFeR keeps the error rate under 1% and the saving rate larger than 20% in most cases as the selecting fraction $\alpha > \frac{5}{24}$, which guarantees both high efficiency and low error rate compared with existing methods.

I. Introduction

With the proliferation of the network threats in big data era, network flow monitoring becomes indispensable to prevent attacks. Distributed Denial of Service (DDoS) attack [1] is one of the typical attacks in this class. It floods the targets with superfluous requests to overload the system and block other legitimate requests. Flow monitoring can effectively prevent this attack by detecting heavy hitters in network topology. The detected heavy items are defined as items with a larger flow than a given threshold, including source or destination addresses, OD pairs, etc. In software-defined network, the control plane performs detection by collecting measurements from source and destination nodes, involving the transmitted number of data packets.

In the single-path setting, the measurement at each switch simply belongs to one certain item. However, in multi-path routing, the measurement for an item is collected from various routes and summed up by controllers. Unfortunately, most existing approaches ignore the efficiency of the whole collection process. They get the long lists of measurements even when there are tremendous routes in the network. When applied in a network where resources are much more critical, their approaches cause too much waste.

To realize the full utilization of network resources, we intend to shorten the length of the messages shared between the switches and controller using the mathematical features of measurement values. However, there are still some challenges needed to be dealt with. On the one hand, the measurement accuracy will be impacted by these approximate messages. On the other hand, in a network where an item is highly possible to be split between different sketches, whether it is heavy or not will often be misjudged [2].

To address the above challenges, in this work, we present a novel flow monitoring method, named ReFeR: a "**Re**port-**Fe**edback-**R**eport" three-step detecting scheme, which is especially efficient for heavy items detection. In this detecting scheme, switches firstly report the binary order of magnitude of measurements. After the calculation of controllers, reported items will be asserted to three degrees: exactly heavy, not heavy and uncertain. The controller then sends a feedback containing the list of uncertain items to the switches, and the switches will respond the significant digits of these uncertain items in return, which leads to the final judgment on all items.

We run a simulator to evaluate the accuracy and the efficiency of ReFeR. When practical parameters are set, the error rate is kept under 1% in most cases. And the total length of messages can usually be shortened by 20%–60%. In summary, this paper makes the following contributions:

- We present ReFeR, a detecting scheme which allows the controller to detect heavy items in a network with shorter messages shared with switches.
- We leverage the order of magnitude and most significant digits on measurements to shortened the messages, ensuring the required detecting accuracy and higher efficiency.
- We evaluate ReFeR by extensive simulations with different characteristics and different monitoring tasks, analyzing the appropriate parameters to set in practice.

The remaining part of the paper is organized as follows. Section II illustrates the background and the related work. Section III gives a statement of the problem that our approach ReFeR solves. Section IV presents the design of ReFeR. Section V demonstrates the theoretical analysis on the accuracy of the result and the efficiency of data transmission. Section VI presents the results of our stimulated evaluation. Finally, Section VII concludes the paper.

II. Related work

The methods of flow measurements have been proposed for measuring the traffic in software-defined networks in recent years. As there are three parts in flow monitoring system: monitor, target switches and shared measurement information, previous efforts can be classified into the three categories respectively: monitor optimization, collection cost from switches optimization and sketches leverage.

First of all, some researchers focus on the optimization of monitors, including the monitoring coverage and sampling granularity. Zang et al. [3] apply greedy heuristics for flow monitoring, while optimizing coverage and cost in an IP network. To maximize the coverage of the entire universe of potential communication pairs, there is another way proposed by Jackson et al. [4] that to minimize the number of monitors considering the autonomous systems. For another perspective, Suh et al. [5] optimize the sampling rates of flow monitoring while sampling only a portion of the network. While the network can be always changing, Zhang et al. [6] develop a prediction-based algorithm, dynamically changing the granularity of measurement. And recently, LEISURE [7] achieves a load-balanced performance among monitors dealing with monitoring tasks globally. All of these efforts can reduce the cost of the monitors to some extent, but some of the monitors only have one monitoring function where extra overhead is unavoidable.

The second category tackles with the bandwidth cost caused by collection of statistics from target switches. The most popular approach is OpenFlow [8]. It firstly splits the data and the control planes by standardizing an open interface which can be dictated by the remote software controllers. The push-based mechanism designed for flow statistics collection has some limits in practical applications. The dynamic traffic will frequently trigger switches to send massive number of measurement reports to the controller, causing large cost of control channel bandwidth. To deal with this, Su et al. [9] and Chowdhury et al. [10] reduce the communication cost while making a balance between cost and flexibility. Moreover, Yu et al. [11] propose a memory-efficient solution which balances measurement load. Recently, Xu et al. [12] use wildcard-based requests and also extend the problem to the general case, reducing the amount of floats needed to be collected. And a OverWatch detection algorithm [13] is proposed in the same year. This lightweight flow monitoring algorithm can capture the key features of DDoS attack traffics on the data plane by polling the values of counters in OpenFlow switches, where the method is specifically designed for DDoS attack prevention. Although the collection quantity is reduced by the above methods, the accuracy sacrifice is still noneligible.

The third category is about researches on the measurement messages shared between switches and control plane. Previous work focuses on using and optimizing sketches for messages. OpenSketch [14] separates the measurement data plane from the control plane and stores a library of predefined functions in routers, where controllers can easily reprogram them depending on different task requirements. SCREAM [15] allocates resources in an efficient way on hardware switches with constrained resources. Although both of these two methods can offer convenient library or efficient resource allocation, they cannot operate new custom sketches for each task. According to this, UnivMon [2] is designed to achieve both generality and high accuracy, supporting multiple applications with high fidelity. Its data plane nodes play the monitoring role and report sketch summaries to the control plane which calculates application-specific metric estimates. However, as the network resources become more and more critical, this sketches summary information is still as long as an exchanging information.

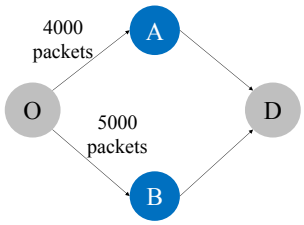


Fig. 1. Example topology of multi-path routing

In this control-switch transmission scheme, what we focus on is to efficiently collect the flow measurement information with low cost and monitor network performance accurately. To realize it, we propose a novel method especially for this type of tasks in multi-path routing, where each switch needs to report less data, combining by the binary order of magnitude and significant digits for further analysis. A correct result can be gotten after the controller's calculation on data collected from all switches.

III. Problem statement

In a network containing m switches and one controller, $G(V, E)$ is the topology of flow network, where the set of verticals V denotes the set of switches, and the set of edges E denotes the set of paths among switches, sources and destinations. The controller receives enquiries of detecting heavy items, whose values are larger than a given threshold g, which we define as the *heavy threshold*. Here, the items are the objects in the network, including source or destination addresses, OD pairs, etc. The values include the amount of data, or number of packets sent from or to the address. The value of an item can be split over several switches, as in the multi-path routing, which means a switch itself is only able to measure a part of the value, and the value of an item is known only after the controller sums up from all switches. An example is shown in Figure 1. The packets sent from O to D split across two paths, and the number of packets sent from O or to D should be measured as the sum $4000 + 5000 = 9000$.

We need to find an approach, allowing the messages between the switches and controller to be much shorter than just sending every exact value, while ensuring a high accuracy on heavy item detection. For convenience of describing the problem, we define the *splitting degree* s_P of an item, as the number of switches that its paths are split across. Therefore, the value of the item is $P = \sum_1^{s_P} p_i$, where p_i is the value of the item at the ith passed switch. Again as shown in Figure 1, the split degree of source O and destination D should be

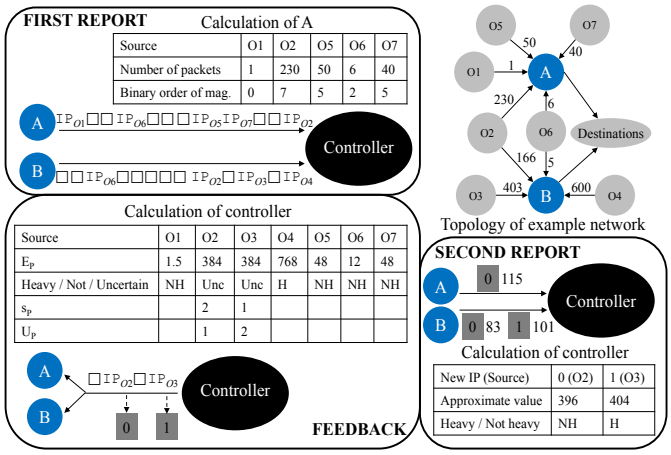


Fig. 2. An example network and our approach

measured as 2. We also define the *average splitting degree* s_{av} as the average value of splitting degrees of all items.

IV. Design of ReFeR

We propose a novel measurement method named ReFeR, standing for "Report-Feedback-Report". A measurement report in ReFeR is created by three steps: *first report*, *feedback*, and *second report*.

The upper right part of Figure 2 shows an example of a network. We suppose that the task is to detect heavy sources, so the target items are source addresses. The heavy threshold is set as $g = 400$. We explain the three steps of ReFeR in the example network as following.

A. First report

We define the *binary order of magnitude* of P as the integer k which satisfies $2^k \leq P < 2^{k+1}$. To make a first report, each switch categorizes the items linked to it into different sets, according to the binary order of magnitude of their values. The message sent from that switch to the controller is created by grouping up the IPs of items within the same set, and putting a pre-defined separator (hereinafter referred to as □) between two groups where the difference of their binary order of magnitude is 1.

For switch A in the example network, 5 items send packets with a path passing the switch. As shown in the upper left part of Figure 2, the switch first calculates their binary order of magnitude, such as 7 for $O2$, because its value 230 satisfies $2^7 \leq 230 < 2^{7+1}$. Then the switch categorizes them into 4 sets and sends $\texttt{IP}_{O1}\square\square\texttt{IP}_{O6}\square\square\square\texttt{IP}_{O5}\texttt{IP}_{O7}\square\square\texttt{IP}_{O2}$ to the controller. Note that no item at this switch has a value whose binary order of magnitude is 1, so 2 separators are between the IP of $O1$ (binary order of magnitude 0) and IP of $O6$ (binary order of magnitude 2). Similarly, 3 separators are between $\texttt{IP}_{O6}$ and $\texttt{IP}_{O5}$ and 2 are between $\texttt{IP}_{O7}$ and $\texttt{IP}_{O2}$. In other words, the binary order of magnitude for an item value at a switch is equal to the number of separators before its IP in the message sent by that switch, as shown in Figure 3. In the same

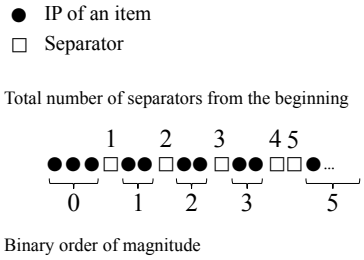


Fig. 3. Number of separators and binary order of magnitude

way, switch B sends $\square\square\texttt{IP}_{O6}\square\square\square\square\texttt{IP}_{O2}\square\texttt{IP}_{O3}\square\texttt{IP}_{O4}$ to the controller.

B. Feedback

In this step, the controller sums up the reported values and calculates the approximate value of each item. It picks out the items whose values are close enough to g, which need further verification before making an assertion of being heavy or not. Then it sends the list of selected items to all switches.

From the first report, the controller sums up the approximate value of each item. For an item whose value is P and split degree is s_P, if each of the s_P switches reports the binary order of magnitude k_1, k_2, ..., k_{s_P}, we have $\sum_1^{s_P} 2^{k_i} \leq P < 2\sum_1^{s_P} 2^{k_i}$. Let $L_P = \sum_1^{s_P} 2^{k_i}$, and then $L_P \leq P < 2L_P$. Since the controller has not yet got any extra information, we consider P at complete random and estimate it as its expectation $E_P = \frac{3}{2}L_P$. The controller processes these steps for each distinguished item, and estimates the corresponding value.

Next, the controller compares E_P of each item with g, to determine whether IP_P is a heavy item or not. According to the inequality above, we know that IP_P cannot be heavy if g is larger than $2L_P = \frac{4}{3}E_P$, and similarly IP_P must be heavy if g is smaller than $L_P = \frac{2}{3}E_P$. But we are uncertain with the remaining items. The list of items that the controller makes a feedback is selected from them, about which the switches need to make one more report for verification. For the efficiency of data transmission, we set a *selecting fraction* α. If $(1-\alpha)E_P < g < (1+\alpha)E_P$, then the item is put into the list. Otherwise the controller simply determines whether it is heavy or not by whether E_P is larger than g. As shown in Figure 4, α is selected between 0 and $\frac{1}{3}$. $\alpha = 0$ means the list of feedback is empty, and $\alpha = \frac{1}{3}$ means all uncertain items are put into the list. The value of α will have an effect on the accuracy of the result and efficiency of data transmission. We determine its adequate value in Section VI. In the example network, the controller calculates E_P of each item as shown in the bottom left part of Figure 2. Supposing that we set $\alpha = \frac{1}{3}$, then an item is uncertain when $300 < E_P < 600$. The controller detects IP_{O4} as heavy, IP_{O1}, IP_{O5}, IP_{O6} and IP_{O7} as not heavy, and the rest 2 items as uncertain.

For the items put into the list, we do not expect the switches to report every digit of their values. We define the n most significant digits of a number as the n digits who have the greatest values, such as the digits 110_2 are the 3 most significant digits of the integer 1100_2, and similarly for the

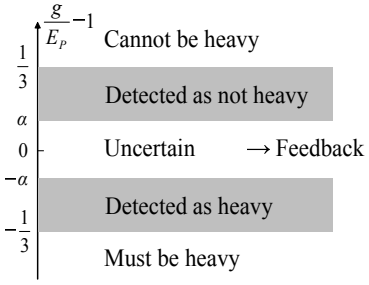


Fig. 4. Selecting fraction and feedback

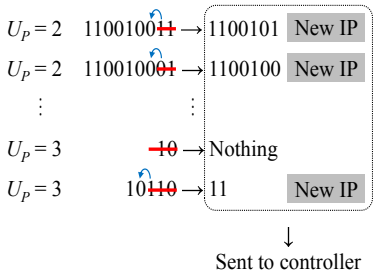


Fig. 5. Some items do not need to be sent at all, including its new IP

least significant digits. We can discard several least significant digits to shorten the messages. Suppose that we are estimating the value $P = \sum_1^{s_P} p_i$, each value of p_i is in binary. We already know that its splitting order is s_P, and the required absolute error is at most ϵ. The number of digits that can be discarded depends on s_P. Fewer digits can be discarded when s_P is larger, since P is calculated as the sum of s_P addends. In the example network, the splitting degree of $O2$ is 2, while that of $O3$ is 1. If we discard U least significant digits of the value of $O3$, the absolute error is at most 2^U. While for $O2$, discarding U digits makes the absolute error at most $2 \cdot 2^U$ because its value at switch A and B are summed up. We define the number of *unnecessary digits* of P as U_P, where U_P is the integer satisfying $2^{U_P} \leq \frac{\epsilon}{s_P} < 2^{U_P+1}$. Then U_P least significant digits can be discarded for P.

The controller gets s_P by counting the number of switches from which this item has appeared in the message. Aiming at keeping the error rate in the best cases under 1%, we set $\epsilon = 0.01g$. If the values are integers, like the number of packets, there are no negative unnecessary digits. It means that if the integer U_P satisfying $2^{U_P} \leq \frac{\epsilon}{s_P} < 2^{U_P+1}$ is negative, we set $U_P = 0$. In the example network, the controller calculates that $O2$ has 1 unnecessary digit, while $O3$ has 2.

What the controller sends to all switches is similar to the first report, but the binary order of magnitude is replaced by U_P. In the example network, it sends $\square\texttt{IP}_{O2}\square\texttt{IP}_{O3}$. Also, U_P is equal to the number of separators before the IP of the item in the message sent by the controller.

C. Second report

After receiving the feedback from the controller, each switch makes a second report to allow the controller to verify the values of the items within the list of feedback. The message sent from each switch involves only the items in the list that has a path at it. To shorten the message, we do not need to send the original IP of an item, but instead its sequence number within the list. Therefore, we only need $\lceil \log_2 F \rceil$ bits to identify an item when there are F items in the list. We define the sequence number of an item as its *new IP*. In the example network, the new IP of $O2$ and $O3$ are 0 and 1 respectively.

The unnecessary digits of the values are discarded, but since the values will be summed up by the controller, we use the rounded value when discarding them. In the example network, switch B needs to send the value of $O2$, $166_{10} = 10100110_2$ with 1 digit discarded, so $1010011_2 = 83_{10}$ is sent. It also needs to send the value of $O3$, $403_{10} = 110010011_2$, and 2 digits are unnecessary, so $1100101_2 = 101_{10}$ is sent instead of $1100100_2 = 100_{10}$, since 110010100_2 is closer to 110010011_2 than 110010000_2 is to 110010011_2. This can be seen as the integer closest to $\frac{p}{2^{U_P}}$ is sent, where p is the value of the item at that switch. If $\frac{p}{2^{U_P}} < 0.5$, the item does not need to be sent, since the integer supposed to be sent is 0. Therefore, it is possible that a switch does not have to send the value of an item even when there is a path at it, and thus the switch does not have to send its new IP either, contributing to improve the efficiency, as shown in Figure 5.

When the controller receives the messages of the second report, it sums up the values of every distinguished item. If an item has U_P unnecessary digits, and v_1, v_2, ..., v_{s_P} are the integers sent from s_P switches ($v_i = 0$ if the item does not appear in the message from the ith switch), then the controller estimates the value of the item as $2^{U_P} \sum v_i$, and determines whether it is heavy or not according to it. As shown in the bottom right part of Figure 2, the controller in the example network estimates the value of $O2$ and $O3$ as 396 and 404 respectively, so it detects IP_{O3} as heavy.

V. Theoretical Analysis

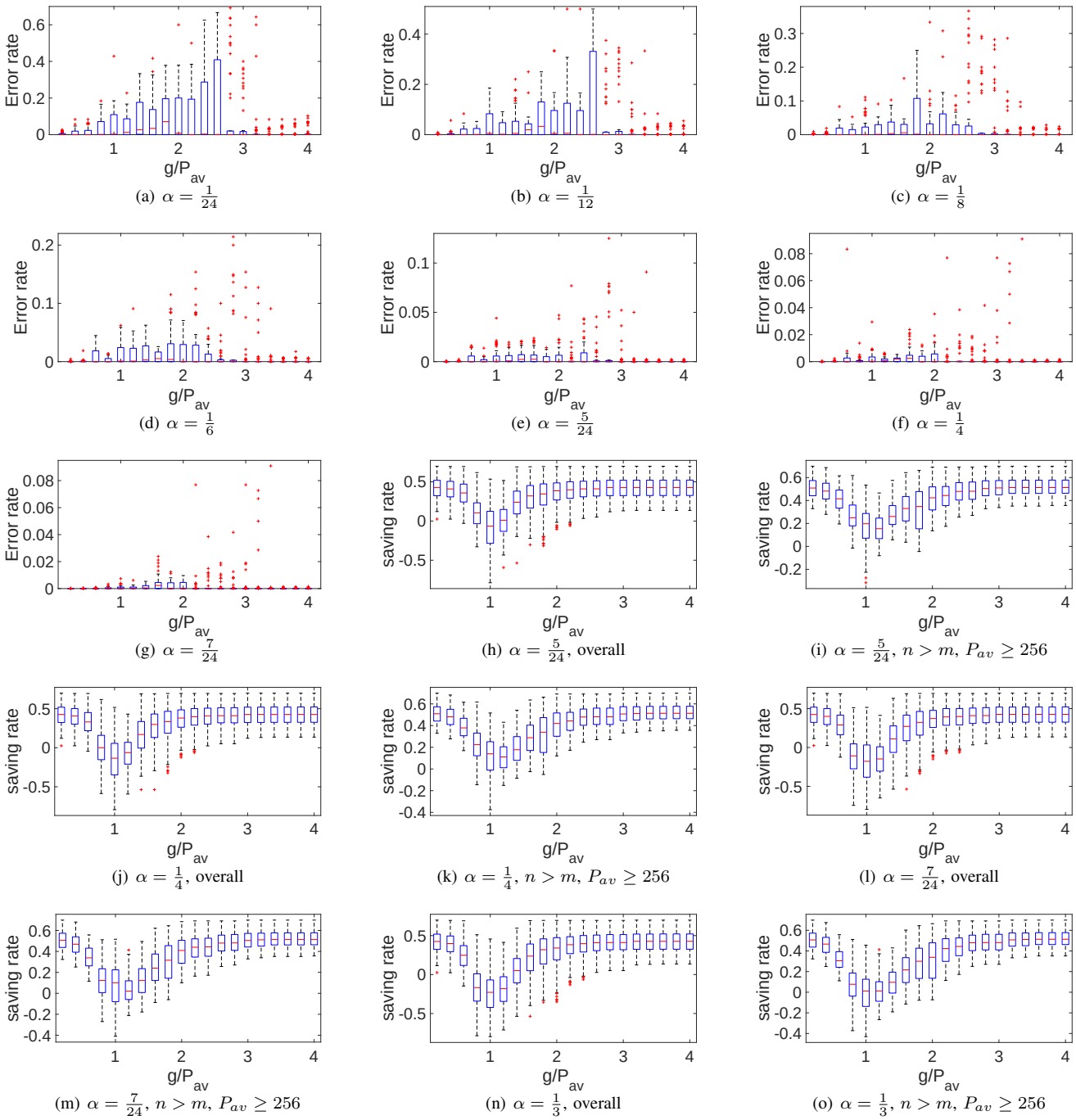
In this section, we analyze the accuracy of the result and efficiency of data transmission of our approach theoretically, in order to find out which parameters have an effect on the final result.

A. Accuracy analysis

The accuracy of the result is affected by the selecting fraction α, as mentioned in Section IV-B. We consider an item uncertain when $(1-\alpha)E_P < g < (1+\alpha)E_P$, where α is between 0 and $\frac{1}{3}$. We then define a false positive as an item detected as heavy while actually not heavy, and similarly, a false negative as an item detected not heavy while actually heavy. Thus we define the error rate as $\frac{n_{FP}+n_{FN}}{n_H}$, where n_H, n_{FP} and n_{FN} are the number of heavy items, false positives and false negatives respectively. If we define $N(x, y)$ as the number of items whose values are between x and y, then $n_H = N(g, +\infty)$, $n_{FP} \leq N(\frac{2}{3}g, (1-\alpha)g)$, and $n_{FN} \leq N((1+\alpha)g, \frac{4}{3}g)$.

B. Efficiency analysis

To evaluate the efficiency of data transmission, we define the saving rate as $\frac{L_0 - L}{L_0}$, where L_0 and L are the minimum length of messages between the switches and controller to detect all heavy items, without and with ReFeR respectively.



(a) $\alpha = \frac{1}{24}$ (b) $\alpha = \frac{1}{12}$ (c) $\alpha = \frac{1}{8}$

(d) $\alpha = \frac{1}{6}$ (e) $\alpha = \frac{5}{24}$ (f) $\alpha = \frac{1}{4}$

(g) $\alpha = \frac{7}{24}$ (h) $\alpha = \frac{5}{24}$, overall (i) $\alpha = \frac{5}{24}$, $n > m$, $P_{av} \geq 256$

(j) $\alpha = \frac{1}{4}$, overall (k) $\alpha = \frac{1}{4}$, $n > m$, $P_{av} \geq 256$ (l) $\alpha = \frac{7}{24}$, overall

(m) $\alpha = \frac{7}{24}$, $n > m$, $P_{av} \geq 256$ (n) $\alpha = \frac{1}{3}$, overall (o) $\alpha = \frac{1}{3}$, $n > m$, $P_{av} \geq 256$

Fig. 6. (a)–(g) are error rates for 7 different values of α; (h)–(o) are saving rates for 4 different values of α

Without ReFeR, each switch will send the IP and the value of all items with a path through it. We consider the length of each IP as 16 bits, and the length of each value is at least the number of its binary digits, while assuming they are all integers. It needs at least $\text{int}(\log_2 t)+1$ bits to send the exact value of an integer t. If there are n items within the whole network, the splitting degree of the ith item is s_i, and the value of each one of the s_i paths is p_j, then $L_0 = \sum_{i=1}^{n} \sum_{j=1}^{s_i} (16 + \text{int}(\log_2 p_j) + 1)$.

In ReFeR, the length here is made up of three parts:

First report. The message that each switch sends is combined by the IPs of items at it, and separators □. Suppose that there are m switches, the number of items with a path at the ith switch is a_i, and the value of the jth item at that switch is p_j. Since the binary order of magnitude of the value of an item at a switch is equal to the number of separators before its IP in the message sent by that switch, as mentioned in Section IV-A, the number of separators sent by a switch is equal to the maximum binary order of magnitude of the items at it. Thus $\max(\text{int}(\log_2 p_j))(1 \leq j < a_i)$ separators are needed at the ith switch. Again considering the length of IP as 16 bits, and a separator costs at least 1 bit, then the minimum length

for the first report is:

$$L_{first} = \sum_{i=1}^{m}(16a_i + \max(\text{int}(\log_2 p_j)))(1 \leq j < a_i).$$

Feedback. The length of the feedback message from the controller depends on the number of items in the feedback list and separators. Suppose that there are F items, the ith item has U_i unnecessary digits. Similar to how many separators are needed at each switch, we can conclude that $\max(U_i)$ separators S are needed by the controller. The controller puts the IPs of these items and the separators in the message. Still considering the length of IP as 16 bits, and a separator costs at least 1 bit, then the minimum length for the feedback is:

$$L_{feedback} = 16F + \max(U_i).$$

Second report. The length of the message that each switch sends depends on the total length of the new IPs of items at it, and their values without the unnecessary digits. As mentioned in Section IV-C, the length of the new IP of an item is $\lceil \log_2 F \rceil$ when there are F items in the list of feedback. If the number of items in the list of feedback with a path at the ith switch is b_i, and the value of the jth item at that switch is p_j, with U_j unnecessary digits, then the minimum length for the second report is:

$$L_{second} = \sum_{i=1}^{n}\sum_{j=1}^{b_i} f(j),$$

where

$$f(j) = \begin{cases} \lceil \log_2 F \rceil + \text{int}(\log_2 \left\lfloor \frac{p_j}{2^{U_j}} + \frac{1}{2} \right\rfloor) + 1 & , \frac{p_j}{2^{U_j}} \geq \frac{1}{2} \\ 0 & , \frac{p_j}{2^{U_j}} < \frac{1}{2} \end{cases}.$$

Note that $\left\lfloor \frac{p_j}{2^{U_j}} + \frac{1}{2} \right\rfloor$ is the rounded value of p_j discarding U_j unnecessary digits. When it is 0, which means all the digits of the item are unnecessary, both the value and the new IP of the item do not need to be sent, as mentioned in Section IV-C.

Finally, the whole length is the summary of these three parts: $L = L_{first} + L_{feedback} + L_{second}$.

Whether L is smaller than L_0 determines whether the saving rate is positive. Various parameters of network topology have an effect on the saving rate, including n, m, the splitting degree of each item, the value of each item (which determines the binary order of magnitude), and the heavy threshold g (which determines F). In the next section, we show how the results change when these parameters change, and analyze in which type of networks is better to apply ReFeR.

VI. Performance Evaluation

Evaluations are divided into two parts, dealing with the accuracy of the result and the efficiency of our approach respectively. We run simulators with different parameters of network topology. Packet flows are generated at random. A Python program is used for calculating the error rate and saving rate for each case.

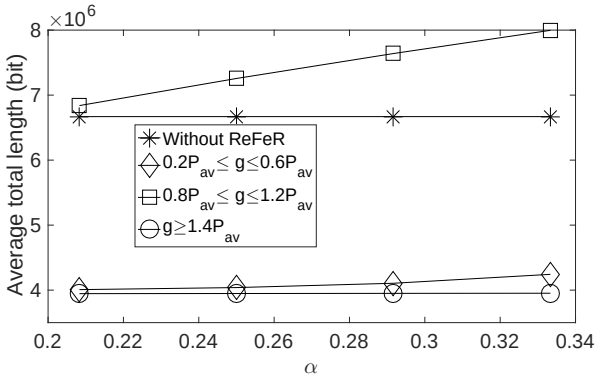


Fig. 7. Average total length of messages to detect all heavy items

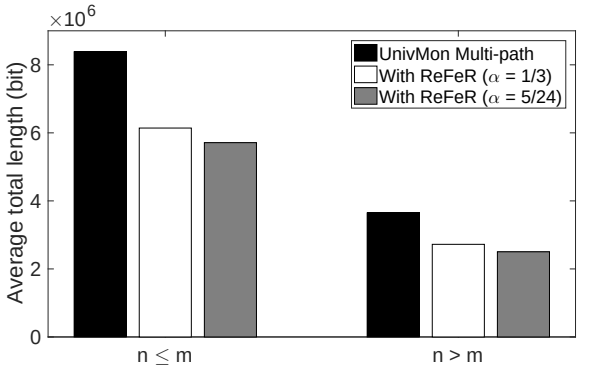


Fig. 8. Comparison of average total length of messages

A. Accuracy evaluation

We run a simulator to show how the error rate changes when the number of items n, average splitting degree s_{av}, and average value of all items P_{av} change. We set α at 7 different values where $0 < \alpha < \frac{1}{3}$, like $\frac{1}{24}$, $\frac{1}{12}$, $\frac{1}{8}$, ..., $\frac{7}{24}$, and repeat the simulation. We run it in 96 different cases for each α, with n from 2 to 65536, s_{av} from 4 to 256, and P_{av} from 256 to 65536. The heavy threshold g is another parameter that has an effect on the error rate. For each case, we set g at 20 different values, from $0.2P_{av}$ to $4P_{av}$.

As shown in Figure 6 (a)–(g), when α is set as the smallest of the 7 values, $\frac{1}{24}$, the error rate grows up to over 20% in some cases, and sometimes even larger than 50%, because most items which should be in the list of feedback are not put into the list. Smaller error rates appear more often when α gets larger. And when $\alpha > \frac{5}{24}$, the error rate is mostly kept under 1%, and rarely over 5%.

Therefore, we recommend that α be set between $\frac{5}{24}$ and $\frac{1}{3}$ in practical use considering the accuracy. While a larger α allows the result to be more accurate, the efficiency of data transmission drops down when α grows, which is mentioned in the next subsection. The selection of α should depend on the tasks to accomplish. A larger α is appropriate for tasks requiring higher accuracy, while a smaller α for those requiring better efficiency.

B. Efficiency evaluation

We run another simulator to show how the saving rate changes when the number of switches m, number of items n, average splitting degree s_{av}, and average value of all items P_{av} change. According to the result of accuracy evaluation, we set α at 4 different values selected in Section IV which keeps the error rate low, $\frac{5}{24}$, $\frac{1}{4}$, $\frac{7}{24}$, and $\frac{1}{3}$, and repeat the simulation. We run it in 348 different cases for each α, with m from 4 to 65536, n from 2 to 65536, s_{av} from 4 to 256, and P_{av} from 256 to 65536. The total length of messages will get longer if we apply ReFeR while there is less than one item on average at each switch. Therefore, we keep $ns_{av} > m$. The heavy threshold g also has an effect on the saving rate, similar to its effect on the accuracy. For each case, we again set g at 20 different values, from $0.2P_{av}$ to $4P_{av}$.

We first set $\alpha = \frac{5}{24}$. As shown in Figure 6(h), the saving rate is mostly larger than 20% when $g < 0.8P_{av}$ or $g > 1.2P_{av}$. When $0.8P_{av} \leq g \leq 1.2P_{av}$, the saving rate gets negative in half of the cases, and reaches below -70% in extreme cases. When the saving rate is negative, the total length of messages with ReFeR is larger than that without ReFeR. We observe that most negative saving rates appear when $n \leq m$ or $P_{av} = 256$. The saving rates when $n > m$ and $P_{av} \geq 256$ are shown in Figure 6(i). In these cases, negative saving rates appear in only few cases, and the saving rate is usually between 20% and 60%. When α gets larger, as shown in Figure 6(j)–6(o), the result is similar to that when $\alpha = \frac{5}{24}$, but negative saving rates appear a bit more often, because more items are put into the list of feedback, making the messages longer.

When we consider g, as shown in Figure 6, for all values of α, the smallest saving rate appears mostly when $g = P_{av}$, because there are usually more items in the list of feedback when g is closer to the average value of all items. We get larger saving rates more often when $g \leq 0.6P_{av}$ or $g \geq 1.4P_{av}$. We divide all monitoring tasks into 3 groups according to g, and compare the average total length of messages of each group with and without ReFeR. As shown in Figure 7, we run the simulator with 4 values of α. When $0.8P_{av} \leq g \leq 1.2P_{av}$, the average total length with ReFeR gets larger than that without ReFeR, because g is close to P_{av}, as mentioned above. Otherwise, the average total length is shortened.

UnivMon uses sketches for flow monitoring [2]. When extended to multi-path routing, it requires all switches with sketches to report the counters to the controller. We compare the total length of messages between the switches and controller in tasks of heavy item detection in networks. As shown in Figure 8, in both categories of tasks where there are relatively fewer items per switch ($n \leq m$) and more items per switch ($n > m$), ReFeR shortens the total length of messages to detect all heavy items, either when we set a larger or smaller selecting fraction ($\alpha = \frac{1}{3}$ and $\alpha = \frac{5}{24}$ respectively).

From the results of the evaluation, we recommend that ReFeR be applied to networks where the average number of items per switch and average value of each item are relatively large. If g can be estimated in advance, it is better to apply ReFeR if g usually satisfies $g \leq 0.8P_{av}$ or $g \geq 1.2P_{av}$, like where it is often required to monitor quite heavy items with g much larger than $1.2P_{av}$. Also, whether to adopt a smaller or larger value of α should be decided according to the degree of importance on accuracy and efficiency.

VII. Conclusion

We present the novel method ReFeR for reporting measurements from switches to the controller in flow monitoring tasks in software-defined networks, especially sufficient for detecting heavy items. When the items within a network, like source addresses, are mostly in a multi-path setting, we can shorten the total length of the messages between the switches and controller. Adopting different values of the selecting fraction α results in different error rate and saving rate. While in most cases an agreeable tradeoff can be reached. We plan to adjust the chronological order of the report from each switch to optimize the time cost of our approach in future work.

References

[1] J. Mirkovic and P. Reiher, "A taxonomy of DDoS attack and DDoS defense mechanisms," *ACM SIGCOMM Comput.commun.rev*, vol. 34, no. 2, pp. 39–53, 2004.

[2] Z. Liu, A. Manousis, G. Vorsanger, V. Sekar, and V. Braverman, "One sketch to rule them all:rethinking network flow monitoring with UnivMon," in *ACM SIGCOMM Conference*, 2016.

[3] H. Zang and A. Nucci, "Traffic monitor deployment in IP networks," *Computer Networks*, vol. 53, no. 14, pp. 2491–2501, 2009.

[4] A. W. Jackson, W. Milliken, C. A. Santivez, M. Condell, and W. T. Strayer, "A topological analysis of monitor placement," in *IEEE International Symposium on Network Computing and Applications*, 2007.

[5] K. Suh, Y. Guo, J. Kurose, and D. Towsley, "Locating network monitors: complexity, heuristics, and coverage," *Computer Communications*, vol. 29, no. 10, pp. 1564–1577, 2006.

[6] Y. Zhang, "An adaptive flow counting method for anomaly detection in sdn," in *ACM Conference on Emerging NETWORKING Experiments and Technologies*, 2013.

[7] C. W. Chang, G. Huang, B. Lin, and C. N. Chuah, "LEISURE: load-balanced network-wide traffic measurement and monitor placement," *IEEE Transactions on Parallel & Distributed Systems*, vol. 26, no. 4, pp. 1059–1070, 2015.

[8] O. E. Ferkouss, R. B. Ali, Y. Lemieux, and C. Omar, "Performance model for mapping processing tasks to OpenFlow switch resources," in *IEEE International Conference on Communications*, 2012.

[9] Z. Su, T. Wang, Y. Xia, and M. Hamdi, "CeMon: a cost-effective flow monitoring system in software defined networks," *Computer Networks*, vol. 92, pp. 101–115, 2015.

[10] S. R. Chowdhury, M. F. Bari, R. Ahmed, and R. Boutaba, "PayLess: a low cost network monitoring framework for software defined networks," in *Network Operations and Management Symposium*, 2014.

[11] Y. Yu, C. Qian, and X. Li, "Distributed and collaborative traffic monitoring in software defined networks," in *The Workshop on Hot Topics in Software Defined NETWORKING*, 2014.

[12] H. Xu, Z. Yu, C. Qian, X. Y. Li, and Z. Liu, "Minimizing flow statistics collection cost of SDN using wildcard requests," in *INFOCOM 2017 - IEEE Conference on Computer Communications, IEEE*, 2017.

[13] X. Yang, B. Han, Z. Sun, and J. Huang, "SDN-based DDoS attack detection with cross-plane collaboration and lightweight flow monitoring," in *GLOBECOM*, 2017.

[14] "Software defined traffic measurement with OpenSketch, author=Yu, Minlan and Jose, Lavanya and Miao, Rui, booktitle=NSDI, year=2013,."

[15] M. Moshref, M. Yu, A. Vahdat, and A. Vahdat, "Scream: sketch resource allocation for software-defined measurement," in *ACM Conference on Emerging NETWORKING Experiments and Technologies*, 2015.