Two-stage adaptive Bloom Filters for per-flow monitoring in Software Defined Networks

Yan Du
University of Electronic
Science and Technology of China
Email:duyan@std.uestc.edu.cn

Sheng Wang
University of Electronic
Science and Technology of China
Email:wsh_keylab@uestc.edu.cn

***Abstract*—Per-flow monitoring attracts great attention for its importance in networks. Thus a two-stage Bloom Filter is proposed for it. However, without considering the scalability of the two-stage Bloom Filter to different probability distributions of target flows' monitoring action types with high probability deviations, it leads to some serious problems, such as high false positive rate or low resource utilization. Therefore we propose the two-stage adaptive Bloom Filters that are operated collaboratively in network-wide range in software defined networks. The Bloom Filter supports dynamic mappings from the pre-undetermined action types of its constituent Bloom Filters to specific ones and adjustable number of the constituent Bloom Filters. This proposal has two major advantages: 1) false positive rate can be kept under a threshold in the different probability distributions; 2) resource utilization and rejection probability can be significantly improved at low cost of increasing the threshold. We analyze and discuss the two-stage adaptive Bloom Filters from false positive rate, resource utilization and rejection probability. The results from our simulation based on real-life network topology agree with our analysis and discussion.**

Keywords—Bloom Filter; Network management; SDN; Per-flow monitoring; Distributed and collabirative network monitoring

I. INTRODUCTION

Network traffic measurement plays an important role in many network applications, such as abnormal detection and traffic engineering [1,3]. Generic traffic monitoring, typically based on packet sampling (e.g. sFlow [4] and Netflow [5]), is beneficial for coarse-grained visibility. But it cannot satisfy some application requirements of per-flow monitoring, i.e., different monitoring actions performed on different flows or different accuracy requirements to different flows. Per-flow monitoring attracts a great attention in recent studies [3,6,7,15-18]. The key challenge is to satisfy the requirements of different applications to accuracy while achieving the goal of low overhead and memory cost. Therefore Bloom Filters have been applied to per-flow monitoring in networks for their space efficiency and simple management [7,8].

Based on Bloom Filters, a two-stage Bloom Filter (TSBF) is proposed to store and classify target flows for per-flow monitoring [7]. The structure of TSBF (Fig.1) can be divided into two parts, admission Bloom Filter (admBF) having one Bloom Filter for flow filtration and action Bloom Filter (actBF) having an array of Bloom Filters for flow classification. In order to design an array of efficient Bloom Filters to reduce false positive rate while best utilizing scare memory, it is necessary to customize the memory size and number of hash functions of each constituent Bloom Filter according to the number of elements in it. Considering different distributions of the elements, it is quite difficult [12]. Thus an array of Bloom Filters is appropriate to represent static sets of which the memory sizes, hash function numbers and element numbers can be determined with a known distribution before design.

Since actual distribution is different from the one utilized for the design, the actual number of elements inserted in each Bloom Filter is possibly different from the determined one. With increasing difference between the distributions, it causes high false positive rate due to extra insertions or low resource utilization and high rejection probability for rejecting the extra elements to keep the false positive rate [9,10,11]. Therefore, it is necessary to consider the scalability of TSBFs to different probability distributions of target flows' action types in order to keep false positive rate under a boundary while achieving high resource utilization and low rejection probability.

Inspired by software defined networking (SDN) [13] and distributed monitoring in collaboration [1,2,7], our proposal to solve the issue of scalability is the two-stage adaptive Bloom Filters (TSABFs) operated collaboratively in network-wide range with help of controller in SDN. We utilize the separation of traffic measurement control from date plane functions (Fig.2). The controller takes advantage of its global view of network and flows to select an appropriate switch for a target flow. It also adaptively decides un-predetermined action types and adjusts the number of Bloom Filers in actBFs of TSABFs (Fig.3). Owing to our proposal, the false positive rate can be kept under a threshold while maximizing resource utilization subjecting to the minimal rejection probability at low cost of increasing the threshold.

In data plane, a TSABF can be triggered on each switch. The structure of TSABF is composed of two parts (Fig.3). The two key points are that the specific action type of each Bloom Filter in actBF is not predetermined and the number of the Bloom Filters is adjustable. For further discussion on TSABF, we provide a TSABF configuration that identifies the size of memory allocated to each Bloom Filter and the boundary number of the elements in it. Then the memory efficiency of

978-1-5386-3180-5/18/$31.00 ©2018 IEEE

this configuration is optimized while implementation and management are simplified. Each Bloom Filter of TSABF is implemented on TCAM [14] to support fast lookup, although it is limited, expensive, and power hungry [20].

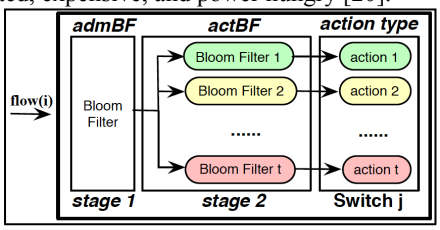

Fig. 1. Structure of two-stage Bloom Filter

In control plane, controller performs two functions: 1) target flow assignment, controller selects a right switch for a target flow on its path; 2) Bloom Filter management, controller can decide specific action types of the Bloom Filters in actBFs and adjust the number of the Bloom Filters. With global view of a network and all flows, controller can utilize its intelligence to combine the two functions and make TSABFs "adaptive" to response to different probability distributions, which keeps false positive rate under a boundary while achieving maximal resource utilization and the minimal rejection probability.

The main contributions of our work are listed as follows:

1) We propose TSABFs and provide an architecture to operate them collaboratively in network-wide range in SDN;

2) We provide an optimal configuration to TSABF and a flow assignment algorithm based on heuristic approach;

3) We give analysis and discussion on false positive rate, resource utilization and rejection probability. We demonstrate the simulation and evaluate the results.

The rest of the paper is organized as follows. Section Ⅱ provides an introduction to the architecture, structure and configuration of TSABF. In Section Ⅲ, we discuss the false positive rate. Resource utilization and rejection probability are discussed in Section Ⅳ and Ⅴ respectively. We demonstrate the simulation and evaluate the results in Section Ⅵ. The related work is introduced in Section Ⅶ. The final section concludes this paper.

II. Architecture, Structure and Configuration

A. Architecture and structure

Control plane is separated from date plane (Fig.2). It makes TSABFs feasible to implement and simple to configure in commodity switches while making controller flexible and easy to program in control plane [6,7,17].

In data plane, a TSABF is triggered on the switch selected by controller. The structure of TSABF consists of two parts, admBF and actBF (Fig.3). The admBF has one Bloom Filter and the actBF is comprised of an array of Bloom Filters. After receiving a configuration message from controller, the identification of each target flow, such as the 5-tuple [7], is hashed in admBF and the corresponding Bloom Filter in actBF in the switch selected on the flow path. Each flow passing the switch must be filtered out successfully by admBF and then classified by actBF to perform the corresponding monitoring action. TSABFs have two important features: 1) the specific action type of each Bloom Filter in actBF is not predetermined but decided by controller instantaneously; 2) the number of Bloom Filters in actBF is not restricted to that of total action types and can be adjusted by controller.

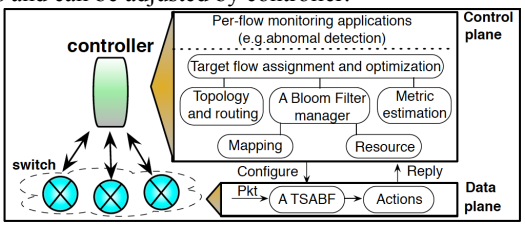

Fig. 2. Architecture for TSABFs

The control plane consists of several necessary components (Fig.2). According to network topology, routing strategy and state of Bloom Filter manager, the component of target flow assignment and optimization performs two functions to keep false positive rate under a boundary and maximize resource utilization while minimizing rejection probability. The first is that it selects an appropriate switch for a target flow. The second is that it decides the specific action type of each Bloom Filter in actBFs of the switch selected and the number of the Bloom Filters. The component, Bloom Filter manager, records the mapping relations that indicate the determined action types of the Bloom Filters. It also records the residual resource of the Bloom Filters. Besides, It updates the records according to the decisions from the component of target flow assignment and optimization. The component of metric estimation is used to make necessary estimations. For example, after receiving an estimated value over or below a threshold from it, controller decides to adjust the number of Bloom Filters in actBFs of new TSABFs and keep the number until a new determinant value arrives. Note that we don't include the discussion on this component and leave it to our future work. In following parts, we suppose all TSABFs have the same configuration.

A simple example illustrates the basic principle (Fig.4). A target flow with type A, $f_{1,5}(A)$, is admitted by controller. Switch S_1, after receiving an request from controller, triggers the TSABF in it because $f_{1,5}(A)$ is the first target flow assigned to it. Then flow $f_{1,5}(A)$ is inserted in Bloom Filter of admBF and actBF's first Bloom Filter of which the action type is mapped to type A according to the request. For conciseness, we assume that the boundary number of elements in each Bloom Filter of actBF is set to one. Therefore flow, $f_{2,6}(A)$, has to be inserted in admBF and actBF's second Bloom Filter mapped to type A in switch S_1. Since each actBF has two Bloom Filters, a new TSABF needs to be triggered to hold coming target flows. For $f_{4,8}(B)$, the action type is mapped to type B. Owing to controller, the least number of TSABFs are triggered to cover all target flows.

B. Configuration

In this section, a configuration to TSABF is provided. It is used to determine the size of memory allocated to each Bloom Filter and the boundary number of elements in it based on following conditions. Then we optimize the configuration to maximize memory efficiency or the number of elements in a

TSABF while reducing the number of free parameters to simplify the implementation and management of TSABFs.

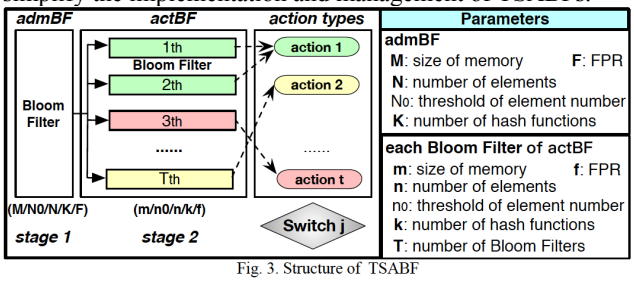

Fig. 3. Structure of TSABF

Firstly, we list the conditions (Fig.3): 1) each triggered TSABF is assigned to a fixed same size of memory; 2) each actBF has the same number of Bloom Filters and the false positive rate thresholds of the Bloom filters are same; 3) the Bloom Filter of admBF also has the same threshold of false positive rate. The conditions restrict the flexibility of TSABFs to some extent and bring some negative effects. For instance, the fixed size of memory is not beneficial for the asymmetric monitoring load on different paths. However, there are some reasons for the conditions. The fixed uniform configuration simplifies the implementation and management of TSABFs. The network-wide collaboration of TSABFs can balance the load. The path based flow assignment and dynamic mapping can also be exploited to reduce the negative effects.

For each Bloom Filter in actBF, false positive rate f [8] satisfies (1).

$$f=(1-e^{-\frac{k.n}{m}})^k \tag{1}$$

Since the Bloom Filter has a threshold of false positive rate θ and satisfies $f \le \theta$, we get the representation of m in (2).

$$m=-k.n_0/\ln(1-\theta^{1/k}) \tag{2}$$

It means that at least m size of memory is allocated to the Boom Filter in order to keep the false positive rate f within the threshold θ when there are k hash functions and no more than n_0 elements in it.

For the Bloom Filter of admBF, we follow the same process above. False positive rate F [8] satisfies (3).

$$F=(1-e^{-\frac{K.N}{M}})^K \tag{3}$$

Because $F \le \tau$ where τ is the threshold of F, we have (4).

$$M=-K.N_0/\ln(1-\tau^{1/k}) \tag{4}$$

Similarly, at least M size of memory is allocated to the Bloom Filter in order to keep the false positive rate F under the threshold τ when there are K hash functions and no more than N_0 elements in it.

Supposing that total memory μ is assigned to each triggered TSABF having one Bloom Filer in admBF and T Bloom Filters in actBF, we get (5).

$$M+T.m=\mu \tag{5}$$

We substitute M and m in (5) with (2) and (4) respectively. And let $N_0=n_0.T$. Then we have the following relation (6).

$$[-Kn_0T/\ln(1-\tau^{1/K})]+T[-kn_0/\ln(1-\theta^{1/k})]=\mu \tag{6}$$

We get the boundary n_0 (7) from (6).

$$n_0=\left\lfloor -\mu/\{T[\frac{K}{\ln(1-\tau^{1/K})}+\frac{k}{\ln(1-\theta^{1/k})}]\}\right\rfloor \tag{7}$$

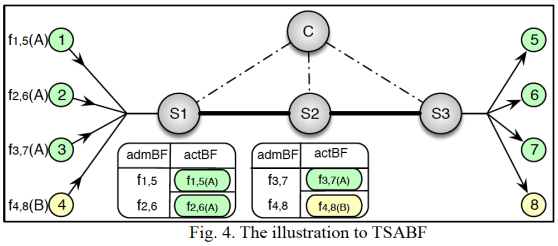

Fig. 4. The illustration to TSABF

Substituting n_0 in (2) and (4) with (7), the expressions of m and M are acquired. At this point, given the values of K, k, T, μ, τ and θ, all necessary parameters of TSABF can be determined. In the derivation above, we assume $N_0=n_0.T$ in (6). In fact, $N<=N_0$ and we obtain the equality when each flow has just one action type. Therefore, N_0 is an upper bound of N. It ensures that the Bloom Filter of actBF is not overflowed. Thus the assumption is reasonable.

Next the configuration is improved to reach two goals: 1) acquiring the optimal n_0 to increase the memory efficiency defined as $(N_0+Tn_0)/\mu=(2Tn_0/\mu)$; 2) reducing the number of free parameters to simplify the configuration and management.

We have equations, $\tau=(1/2)^{Kopt}$, $K_{opt}=ln2(M/N_0)$, $\theta=(1/2)^{kopt}$ and $k_{opt}=ln2(m/n_0)$ [8]. Let M, N_0, m and n_0 fixed and substitute K, k, τ and θ in (7) with the equations. After a derivation, we attain the expression of n_0 in (8).

$$n_0=-\frac{\mu\ln 2}{T(\log_2{}^{\tau}+2\log_2{}^{\theta})} \tag{8}$$

Let $\tau=a\theta$ and substitute τ in (8) with it to get (9). We define $a\in(0,1)$ since admBF acts as a flow filter at the first stage [7].

$$n_0=\frac{\mu\ln 2}{T(\log_2{}^{1/a}+2\log_2{}^{1/\theta})} \tag{9}$$

Given the constants, T, μ and θ, n_0 is a monotonous increase function of a. In order to get the optimal n_0, n_{0_max}, to increase memory efficiency, we set a=1 and get n_{0_max} in (10). The optimal N_0, m, and M are represented in (11), (12) and (13) respectively. We also get the relations, $\tau=\theta$ and $K_{opt}=k_{opt}$.

$$n_{0\_max}=\left\lceil \mu\ln 2/2T\log_2{}^{1/\theta}\right\rceil \tag{10}$$

$$N_{0\_max}=Tn_{0\_max} \tag{11}$$

$$m_{min}=\left\lceil -n_{0\_max}\log_2{}^{\theta}/\ln 2\right\rceil=\left\lceil \mu/2T\right\rceil \tag{12}$$

$$M_{min}=\left\lceil -Tn_{0\_max}\log_2{}^{\theta}/\ln 2\right\rceil=\left\lceil \mu/2\right\rceil \tag{13}$$

Considering the relations above, when T and θ are decided, the configuration of TSABF can be determined while reaching the goals mentioned above (for rest of the paper, we use n_0, N_0, m, M, k and K to denote the optimal values for conciseness).

III. False Positive Rate

Although controller can detect false positives, false positive rate should be controlled below a threshold in order to maintain the performance. For example, the triggered switches deal with false positives at cost of extra resources, such as computing resource. A huge amount of false positives causes an overhead of controller and a waste of bandwidth. Next we analyze and discuss the threshold of false positive rate in terms of TSABF.

Let $f_{adm(j)}$ and $f_{act(j)}$ to denote the false positive rates of admBF and actBF in switch S_j respectively. $n_{i,j}$ denotes the number of flows with the i_{th} action type in actBF of switch S_j. N_j represents the number of flows in admBF of switch S_j.

We use $f^{BF}_{(M, K, N0, Nj)}$ to denote false positive rate of a Bloom Filter at a state (M, K, N_0, N_j) representing that there are N_j elements in the Bloom Filter with M size of memory, K hash functions and a boundary N_0 of N_j in switch S_j. Since admBF consists of one Bloom Filter, $f_{adm}(j)$ can be expressed in (14).

$$f_{adm}(j) = f^{BF}_{(M,K,N0,Nj)} = (1 - e^{-\frac{K \cdot Nj}{M}})^K \tag{14}$$

In switch S_j, actBF has an array of Bloom Filters and false positive rate of each Bloom Filter is represented by $f^{BF}_{(m,k,n0,nij)}$. Therefore, $f_{act}(j)$ is expressed in (15).

$$\begin{aligned} f_{act}(j) &= 1 - \prod_{i=1}^{t}(1 - f^{BF}_{(m,k,n0,nij)}) = 1 - \prod_{i=1}^{t}(1 - f^{BF}_{(m,k,n0,n0)})^{\left\lfloor \frac{nij}{n0} \right\rfloor}(1 - f^{BF}_{(m,k,n0,rij)}) \\ &= 1 - \prod_{i=1}^{t}[1 - (1 - e^{-\frac{k \cdot n0}{m}})^k]^{\left\lfloor \frac{nij}{n0} \right\rfloor}[1 - (1 - e^{-\frac{k \cdot rij}{m}})^k] \end{aligned} \tag{15}$$

Where: $$r_{ij} = n_{ij} - \lfloor n_{ij} / n_0 \rfloor . n_0$$

Overall false positive rate of switch S_j is $f(j)$ in (16).

$$\begin{aligned} f(j) &= 1 - [1 - f_{adm}(j)][1 - f_{act}(j)] \\ &= 1 - (1 - f^{BF}_{(M,K,N0,Nj)})\{1 - [1 - \prod_{i=1}^{t}(1 - f^{BF}_{(m,k,n0,nij)})]\} \end{aligned} \tag{16}$$

Then we derive the threshold of overall false positive rate of switch S_j, $f_{Th}(j)$.

$$\begin{aligned} f(j) &\le 1 - (1 - f^{BF}_{(M,K,N0,N0)})(1 - f^{BF}_{(m,k,n0,n0)})^T \\ &= 1 - (1 - \theta)^{T+1} \\ &= f_{Th}(j) \end{aligned} \tag{17}$$

In (17), the $f_{Th}(j)$ is related to $f^{BF}_{(M,K,N0,N0)}$, $f^{BF}_{(m,k,n0,n0)}$ and T. In section Ⅲ, we attain the relation $\tau = \theta$. Thus the threshold is determined by θ and T. Next TSABF is discussed in terms of the threshold of overall false positive rate.

1) Although the distributions of target flows' action types with high probability deviations (23) are different, the false positive rates of the distributions can be kept under a threshold of overall false positive rate since $f_{Th}(j)$ is not relevant to the distributions according to (17).

2) θ is supposed to be constant in section Ⅲ. The threshold of overall false positive rate is a monotonous increase function of T in (17).

Average overall FPR f_{avg} and average threshold of overall FPR ft_{avg} are defined in (18) and (19) and used as evaluation indexes in the part of our simulation. Z is the number of triggered switches.

$$f_{avg} = [\sum_{j=1}^{z} f(j)] / Z \tag{18}$$

$$ft_{avg} = [\sum_{j=1}^{z} f_{Th}(j)] / Z = 1 - (1 - \theta)^{T+1} \tag{19}$$

IV. Resource Utilization

TSABFs are operated collaboratively in network-wide range to optimize resource utilization. Resource utilization (RU) is defined as a ratio of the number of flows accepted to the number of flows that total triggered TSABFs hold in (20).

$$RU = (\sum_{j=1}^{z}\sum_{i=1}^{t} n_{ij} + N_j) / [Z(n_0 T + N_0)] \tag{20}$$

A. Network-wide optimization to resource utilization

We consider the objective that maximizes the resource utilization across all SD(Source Destination)-pairs subjecting to the minimal rejection probability per SD-pair and the false positive rate boundaries. For this problem formulation, we make some assumptions: 1) Network topology and routing information are available and do not change frequently; 2) Monitoring load matrix $L=[L_{i,k}]$ can be estimated and does not change frequently; 3) Each SD-pair has a single path; 4) Each flow has one action type, which supports following relations.

$$N_j = \sum_{i=1}^{t} n_{ij} \text{ and } N_0 = n_0 . T$$

Then (20) is transformed to (21).

$$RU = \sum_{j=1}^{z}\sum_{i=1}^{t} n_{ij} / (Z n_0 T) \tag{21}$$

Each SD-pair SD_k (k=1,..,F) is featured by the number $L_{i,k}$ of target flows with the i_{th} action type on path P_k. $C_{i,k}$ denotes the rejection probability of target flows with the i_{th} type on path P_k. TSABF in switch S_j (j=1,.., E) has T Bloom Filters in actBF. $Y_{i,j,k}$ is the number of target flows with the i_{th} action type in TSABF of switch S_j on path P_k. Z_j is a binary variable to identify whether switch S_j is triggered or not. The binary variable $U_{j,k}$ is set to 1 if switch S_j is on path P_k. The problem formulation is list as follow.

Given factor(λ)

$$obj \quad \max \quad RU = \frac{\sum_i\sum_j\sum_k Y_{i,j,k}}{\sum_j Z_j n_0 T}$$

$$\begin{aligned}
s.t \quad & 1)\ \sum_j Y_{i,j,k} \le U_{j,k} L_{i,k}(1 - C_{i,k}) && \forall i,k \\
& 2)\ \sum_i \left\lceil \sum_k Y_{i,j,k} / n_0 \right\rceil \le U_{j,k} T && \forall j \\
& 3)\ Z_j = \begin{cases} 0, & when \ \sum_k\sum_i Y_{i,j,k} = 0 \\ 1, & others \end{cases} && \forall j \\
& 4)\ U_{j,k} = \begin{cases} 0, & S_j \notin P_k \\ 1, & S_j \in P_k \end{cases} && \forall j,k \\
& 5)\ Y_{i,j,k} \ge 0 && \forall i,j,k \\
& 6)\ (C_k = \sum_i C_{i,k}) \ge 0 && \forall k \\
& 7)\ (C_k = \sum_i C_{i,k}) \le \lambda && \forall k
\end{aligned}$$

The formulation is a non-linear programming that includes a parameter λ, the expected minimal rejection probability per-SD pair. After determining λ, the programming maximizes resource utilization while ensuring that each SD-pair has a rejection probability at most λ and each Bloom Filter operates within its resource constraint since the threshold of false positive rate. In order to attain the solution, we firstly obtain the smallest possible maximal rejection probability by viewing the problem formulation as a programming with the objective of minimizing max{C_k} subjecting to constraints 1)-6). Then the value acquired from the programming is used as the value of λ and the solution is computed.

We give a brief explanation to each of the constraints. For the flows with the i_{th} action type on path P_k, 1) indicates that

the number of flows assigned to all triggered switches is not over the boundary load. 2) ensures that the number of flows accepted in a triggered switch does not exceed its resource constraint. 3) identities whether a switch is triggered. 4) shows whether switch S_j is on path P_k. 5) confines the values of $Y_{i,j,k}$ to positive integers. 6) and 7) denote the boundaries of C_k.

The programming is non-linear and NP-hard. The direct computation is very time consuming. In addition, it is based on some assumptions that are not in line with dynamic scenario. We need more time efficient approach and leave the problem in future work. In the following section, we provide a heuristic algorithm based on greedy approach for target flow assignment.

B. Heuristics

A heuristic algorithm based on greedy approach is provided in Algorithm for target flow assignment. Note that resource utilization is related to the Bloom Filters in actBFs since (21). Thus we just consider these parts. The algorithm includes two key points: (1) TSABFs are triggered as few as possible to increase resource utilization; (2) the asymmetric monitoring load and different number of switches among SD-pairs are considered to reduce rejection probability.

We briefly illustrate the algorithm. For a target flow $f_k(x)$ with the x_{th} type on path P_k, controller checks whether the x_{th} type Bloom Filters in the switches triggered on the path have an idle position. If only one switch satisfies it, the flow is assigned to the Bloom Filter of the switch selected. If the number of the switches is more than one, it is assigned to the switch whose the x_{th} type Bloom Filter has the maximal number of idle positions.

If none of the x_{th} action type Bloom Filters in triggered switches has an idle position, controller checks whether one of the switches has a Bloom Filter undetermined. If there is only one, it will be selected directly. For more than one the switch satisfying the condition, the switch in which the actBF has the maximal number of the Bloom Filters undetermined will be chosen. After the switch is decided, the flow is inserted into the Bloom Filter that is identified as the x_{th} type.

Supposing that no switch triggered on path P_k satisfies the conditions above, controller checks whether a switch is un-triggered. If only one exists, it will be chosen. For more than one such switch, the switch with the maximal node pressure is selected. The node pressure of switch S_j is defined as the ratio of the positions occupied to total positions in all switches on the paths across the switch S_j in (22). $P(j)$ and S_p represent the sets of the paths across switch S_j and the switches on each of the paths respectively. $|S_p|$ denotes the number of elements in set S_p.

$$NP(j) = \sum_{p \in P(j)} \sum_{s \in Sp} \sum_{i} n_{i,s,p} / \left(\sum_{p \in P(j)} |S_p| . n_0 T \right)$$
$$P(j) = \{ P_k \mid S_j \in P_k, \forall k \} \quad (22)$$
$$S_p = \{ S_j \mid S_j \in p, \forall j \}$$

After triggering the selected switch, the first Bloom Filter is labeled as the x_{th} type and the flow is inserted in it. Without satisfying one of the previous conditions, the flow is rejected for no idle position. Note that we reach the two points above by balancing the load of triggered switches according to their residual resource and triggering switches by considering node pressure.

Algorithm:Flow Assignment Algorithm

```
While (assign a flow with the xth action, fk(x), on path Pk)
    if (SN>0)                                   // SN is number of triggered switches on Pk
      index=Select (TS, x, Pk)                  // TS is the set of indexes of triggered switches on Pk
      if (index≠0)
        Insertion (index, x, fk)
        Update ( )
      else if (NS>0)                            // NS is number of untriggered switches on Pk
        index=Choose (US, Pk)                   // US is the set of indexes of untriggered switches on Pk
        Insertion (index, x, fk)
        Update ( )
      else return false
    else if (SN=0)
      index=Choose (US, Pk)
      Insertion (index, x, fk)
      Update ( )
    else return false
  Select (ts, y, k)
    for ( n∈ts )                                // Rs(n,y) is residual resource of type y on switch n
      if (Rs(n, y)>0)      sy=syun
      else if (R(n)>0)     sb=sbun              // R(n) is number of undetermined Bloom Filters on switch n
    if (sy≠Φ)
      I=rs_max(sy)                              // rs_max (sy) returns index of switch with the maximal type y residual resource in sy
    else if (sb≠Φ)
      I=rn_max (sb)                             // rn_max (sb) returns index of switch with maximal number of undetermined Bloom Filters
    else I=0
    return I
  Choose (us, k)
    q=Max_NP(us, k)                             // Max_PathNum returns index of switch having maximal node pressure
    return q
```

C. Analysis

The TSABFs are operated collaboratively in network-wide range to reduce the impact on resource utilization from the different distributions with high probability deviations. The effect is related to two factors: probability deviation and the number of Bloom Filters in actBF.

1) Probability deviation

Probability deviation is denoted as σ_p and defined in (23). It quantifies the degree that a distribution deviates from a reference one. With increasing σ_p, the difference between the distributions becomes larger.

$$\sigma_p = \sqrt{\sum_{i=1}^{t} (p(i) - p_{rf}(i))^2} = \sqrt{\sum_{i=1}^{t} (p(i) - \frac{1}{t})^2} \quad (23)$$

In (23), $p(i)$ is the probability of the i_{th} action type in a distribution. $p_{rf}(i)$ is the probability of the i_{th} action type in a reference distribution. Here we take uniform distribution as a reference one. When σ_p is increased, resource utilization of the corresponding distribution gets worse.

2) Number of the Bloom Filters in actBF

The flow assignment problem is a combinatorial problem to optimize resource utilization. Resource utilization is improved by increasing T. And larger the value of T is, more obvious the improvement is.

V. Rejection Probability

The threshold of false positive rate results in rejection to target flows. Therefore we study how the trend of rejection probability varies with depletion of space in Bloom Filters and show the impact from different distributions on rejection probability. The definition of rejection probability (RP) is:

$$RP(N_l) = NR_l / (N_l - N_{l-1}) = NR_l / \Delta \quad (24)$$

In (24), NR_l is the number of rejected flows in the l_{th} section. N_l and N_{l-1} are the numbers of total flows at the end of the l_{th} section and the $(l\text{-}1)_{\text{th}}$ section. Rejection probability becomes more obvious and growth of rejection probability gets faster when σ_p is increased. Rejection probability is decreased as T is increased.

Table. 1. Probability distribution

Probability distribution		
Distribution 1	P(i=1)=3/4; P(i=2)=1/8; P(i=x)=1/48, x=2,...,8	σ_{p1}=0.67508
Distribution 2	P(i=1)=1/2; P(i=2)=1/4; P(i=x)=1/24, x=2,...,8	σ_{p2}=0.44488
Distribution 3	P(i=x)=1/8, x=1,2,...,8	σ_{p3}=0

VI. Simulation and Evaluation

Our simulation is carried out on the platform of C++. The parameter setting of TSABF subjects to the configuration in section Ⅲ. The parameters θ=τ=0.001 are referenced in [10]. We restrict T primarily to 8 and16. μ is set to 0.1Mb [14]. n_0 and m are acquired by (10) and (12). K and k can be acquired and are equal to 10. The length of △ is set to 100.

For topology, we use the real-life Abilene [21] network (V=12, E=15). The dynamic traffic at 5min granularity over 6 month horizon in this network is utilized [21]. The volume of the traffic demand between each SD-pair is out of our current consideration and link capacity is assumed to be unlimited. Dijkstra shortest path algorithm takes the link distance as the cost and produces the shortest path between each SD-pair.

A simulated generator generates the action types of target flows according to the probability distributions in Table.1. The target flows are synthesized by combining the real traffic flows with the action types. The heuristic algorithm in Section Ⅴ is applied to flow assignment. The probability deviation σ_p of these distributions is attained by (23) in Table. 1. Although the distributions are not from real traffic, it does not impact the results and related evaluation.

We compare TSABF with TSBF [7]. TSBFs have the same configuration and parameter setting as TSABFs. They are also simulated in the same scenario, such as the network topology and target flows. The flow assignment algorithm for TSBFs [7] is still applied here.

A. False positive rate

For TSABF with T=8 in Fig.5, the average threshold of overall FPR (log(ft_{avg})=-2.05) is acquired by (19). The average overall FPRs of the different distributions do not exceed it even though some of the distributions have high probability deviations. Thus it indicates the scalability of our proposal to different distributions in terms of the threshold of overall FPR. For TSBF, with continuous insertions of flows, the average overall FPRs of the different distributions increase rapidly until they reach the point of 100%. In addition, the larger the value of σ_p is, the faster the average overall FPR of the distribution increases. Supposing that a required threshold of average overall FPR is (-2.05), the curve of TSBF distribution 1 shows that average overall FPR gets the threshold at about 7000. However the resource utilization is less than 20% at this point in Fig.7.

In Fig.6, We show the average thresholds of overall FPR in different values of T by (19) for comparing the differences in the thresholds.

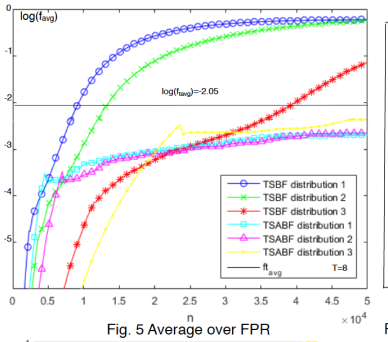

Fig. 5 Average over FPR

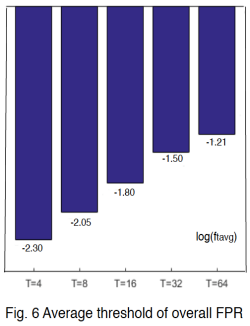

Fig. 6 Average threshold of overall FPR

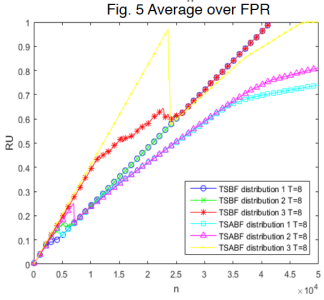

Fig. 7 Resource utilization T=8

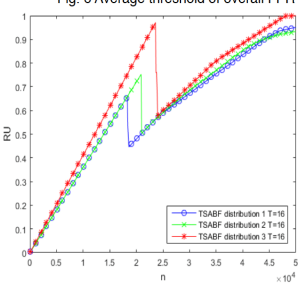

Fig. 8 Resource utilization T=16

B. Resource utilization

In Fig.7, for TSABF with T=8, the resource utilization gets worse with increasing probability deviation. When comparing the curves of TSABF with corresponding ones of TSBF, the resource utilizations of TSABF are not better than those of TSBF except for a sudden increasing part of the curve of TSABF distribution 3. There are two main reasons for the results. The first is that TSBF does not reject any target flow. Although it increases resource utilization, it deteriorates the average overall FPR in Fig.5. The second is that TSABF with T=8 is not "flexible" enough to offset the impact from large probability deviation. However scalability of TSABF can be improved by increasing the number of Bloom Filters in actBF. In Fig.8, resource utilization curves of distribution 1 and 2 are obviously improved and moved closely to that of distribution 3. When compared with TSBF, results of TSABF are better before about 30000.

Although the scalability of TSABF can be improved by increasing T, it leads to deteriorating average threshold of overall FPR. But, in Fig. 6, we can see that the cost is low when considering the improvement to the resource utilization in Fig.8. When T is increased from 8 to16, the threshold (-2.05) is increased to (-1.80) but the curves of the non-uniform distributions with high probability deviations almost overlap with the best one of the uniform distribution in Fig.8.

For the sudden increase and following huge drop in Fig. 7 and Fig. 8, several factors can cause it, such as the fixed memory size of TSABF and flow assignment algorithm. But it is out of our consideration now.

C. Rejection probability

In Fig.9, TSABF with distribution 3 has the best curve of rejection probability. The larger the value of σ_p is, the worse the curve becomes. In Fig.10, the curves of the non-uniform distributions can be significantly improved by increasing T. It also shows the scalability of TSABFs to different distributions with high probability deviations. The rejection probability of TSABF distribution 1 is almost 0 at the point of around 25000, where the average overall FPR is under the average threshold of overall FPR and resource utilization is also better than those of corresponding TSBF. However, at the point, the average FPR of TSBF distribution 1 is larger than 0.1, which brings serious impact on system performance.

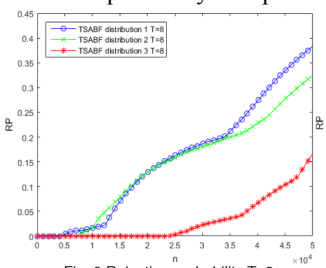

Fig. 9 Rejection probability T=8

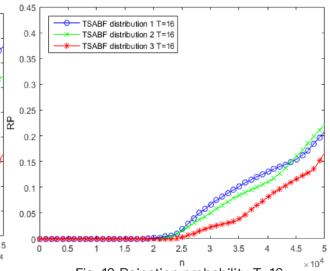

Fig. 10 Rejection probability T=16

VII. RELATED WORK

Traffic monitoring is essential for traffic engineering and network management. The generic traffic monitoring is based on packet sampling, such as sFlow [4] and Network [5]. In cSamp [1], author distributes the monitoring load of tasks to routers to achieve network-wide monitoring subjecting to the resource constraint. For SDN-based traffic monitoring, one of hot research topics focuses on the tradeoff between resource allocation and accuracy [3,6,7,15-18].

Bloom filters are applied widely to solve network problems. In [8], author gives an introduction to Bloom filters and their applications. The scalability problem of Bloom Filter has been discussed in [9,10,11,12]. Our work is most related to [7,9]. In [9], Dynamic Bloom Filter can reduce false positive rate by adjusting the number of Bloom Filters according to the actual size of dynamic set. A two-stage Bloom Filter is proposed for monitoring in SDN [7].

VIII. CONCLUSION

Per-flow monitoring is attractive for its important role in networks. We propose TSABFs operated network-widely in collaborative mode with help of controller in SDN. In date plane, un-predetermined action types and adjustable number of the Bloom Filers in TSABFs are decided by controller according the global view of network and flows. They make TSABFs adaptive to keep the overall FPRs under a threshold while significantly improving resource utilization and rejection probability in different distributions with high probability deviations at low cost of increasing the threshold.

There have many efforts to design the APIs for network control, such as OpenFlow [13] in SDN. But the simple interface of OpenFlow is not enough for the implementation of our proposal. We can utilize P4 [23] working in conjunction with SDN control protocol like OpenFlow to achieve it. In data plane, we can map the functional components, such as hashing and Bloom Filters, to P4 [6]. In control plane, we can implement the components by C++ as sets of modules. In addition, the P4 frame allows us to define the API for control-data plane communication.

Although the number of the Bloom Filters can be adjusted to make TSABFs scalable to different distributions with high probability deviations, an adaptive mechanism is needed to determine the optimal T for dynamic distributions. They will be also discussed in our future work.

REFERENCES

[1] V. Sekar, M. K. Reiter, W. Willinger, H.Zhang, R. R. Kompella, and D. G. Andersen. csamp: A system for network-wide flow monitoring. In *Proc. of USENIX NSDI*, 2008.

[2] V. Sekar, A. Gupta, M. K. Reiter, and H. Zhang. Coordinated sampling sans origin-destination identifiers: algorithms and analysis. In *Proc. of IEEE COMSNETS*, 2010.

[3] Y. Zhang. An adaptive flow counting method for anomaly detection in SDN. In *Proc. of ACM CoNEXT*, 2013.

[4] P. Phaal and M. Lavine. sflow version 5, 2004.

[5] B. Claise. Cisco systems netflow services export version 9, 2004.

[6] Zaoxing Liu and Antonis Manousis. One Sketch to Rule Them All: Rethinking Network Flow Monitoring with UnivMon. In *Proc. of ACM SIGCOMM*, 2016.

[7] Y. Yu and C. Qian. Distributed and collaborative traffic monitoring in Software Defined Networks. In *Proc. of ACM HotSDN*, 2014.

[8] A. Broder and M. Mitzenmacher. Network applications of Bloom filters: A survey. *Internet Mathematics*, vol. 1, no. 4, pp. 485-509, 2005.

[9] D. Guo, J. Wu and H. Chen. Theory and Network Applications of Dynamic Bloom Filters. In *Proc. of IEEE INFOCOM*, 2006.

[10] S. Dharmapurikar, P. Krishnamruthy, and D. E. Taylor. Longest Prefix Matching Using Bloom Filters. In *Proc. of ACM SIGCOMM*, 2003.

[11] K. Xie, Y. Min and D. Zhang. A Scalable Bloom Filter for Membership Queries. In *Proc. of IEEE GLOBECOM*, 2007.

[12] H. Song, F. Hao and M. Kodialam. IPv6 Lookups using Distributed and Load Balanced Bloom Filters for 100Gbps Core Router Line Cards. In *Proc. of IEEE INFOCOM*, 2009.

[13] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner. *Openflow: enabling innovation in campus networks, SIGCOMM Comput. Commun. Rev.*, vol. 38, pp. 69-74, March 2008.

[14] M. Yu and A. Fabrikant. BUFFALO: Bloom filter forwarding architecture for large organizations. In *Proc. of ACM CoNEXT,* 2009.

[15] L. Jose and M. Yu. Online measurement of large traffic aggregates on commodity switches. In *Proc. of USENIX HotCloud*, 2011.

[16] M. Moshref, M. Yu, and R. Govindan. Resource/accuracy tradeoffs in software-defined measurement. In *Proc. of ACM HotSDN,* 2013.

[17] M. Yu, L. Jose, and R. Miao. Software defined traffic measurement with OpenSketch. In *Proc. of USENIX NSDI*, 2013.

[18] M. Moshref, M. Yu, R. Govindan and A. Vahdat. DREAM: Dynamic resource allocation for software-defined measurement. In *Proc. of ACM SIGCOMM*, 2014.

[19] A. Goel, and P. Gupta. Small subset queries and bloom filters using ternary associative memories, with applications. *Proc. of ACM SIGMETRICS,* 2010.

[20] K. Kannan and S. Banerjee, CompactTCAM: Flow entry compaction in TCAM for power aware SDN, *Springer*, 2013.

[21] Abilene, http://http://sdnlib.zib.de.

[22] P4 specification. http://goo.gl/5ttjpA.