Runs bAsed Traffic Estimator (RATE): A Simple, Memory Efficient Scheme for Per-Flow Rate Estimation

Murali Kodialam * T.V. Lakshman * Shantidev Mohanty†

* Bell Labs
101 Crawfords Corner Road
Holmdel, NJ 07733
{muralik,lakshman }@lucent.com

† Department of Electrical Engineering
Georgia Tech.
Atlanta, GA
shanti@ece.gatech.edu

***Abstract*— Per-flow network traffic measurements are needed for effective network traffic management, network performance assessment, and detection of anomalous network events such as incipient DoS attacks. Explicit measurement of per-flow traffic statistics is difficult in backbone networks because tracking the possibly hundreds of thousands of flows needs correspondingly large high-speed memories. To reduce the measurement overhead, many previous papers have proposed the use of random sampling and this is also used in commercial routers (Cisco's NetFlow). Our goal is to develop a new scheme that has very low memory requirements and has quick convergence to within a pre-specified accuracy. We achieve this by use of a novel approach based on sampling two-runs to estimate per-flow traffic. (A flow has a two-run when two consecutive samples belong to the same flow). Sampling two-runs automatically biases the samples towards the larger flows thereby making the estimation of these sources more accurate. This biased sampling leads to significantly smaller memory requirement compared to random sampling schemes. The scheme is very simple to implement and performs extremely well.**

I. Introduction

In this paper, we address the problem of accurate measurement of traffic in a packet network. Measurement of traffic is an important component for traffic management, accounting, detecting DoS attacks, and in traffic engineering [2], [3], [6]. The traffic in the network can typically be classified into flows, and measurements are required on a per-flow basis. The definition of flows can be very flexible. Examples are specific application to application traffic characterized by 5-tuples in the IP packet header, all traffic destined toward a a destination network (characterized by a destination address prefix), etc.

The standard approach used for measuring traffic is to sample the traffic arriving at the router, keep counts of the traffic arrivals on a per-flow basis and then use this counter to estimate the traffic. This approach has been suggested by the Real-Time Flow Measurement Working Group of the IETF. The main problem with this approach is scalability. If the number of flows are large, then keeping per-flow counts consumes considerable memory as well as processing power. In fact, measurements [5] have shown that there might be as many as a 0.5-1.0 million flows in the backbone. These traffic studies [2] [5] have also shown that even though there are a large number of flows in the network, a significant fraction of the traffic is carried by only a small number of flows. These *heavy hitters*, which may only be a few hundreds of flows, can constitute as much as 80-90 % of the traffic at a router [5]. Therefore, detecting and measuring these heavy hitters is an important aspect of traffic measurement. Further, measuring sudden increase in activity towards a given destination can be a sign of a denial of service attack. Therefore, monitoring traffic is a very important component in network security system. Another application area where per-flow measurements are needed is for active queue management [8] for providing fairness in networks. The main idea is to isolate large flows to reduce their impact on the rest of the flows in the network. This is especially important if the large flows are misbehaving open loop UDP sources or TCP sources with short round-trip delays. However, to track these small number of misbehaving sources, we have to wade through tens of thousands (if not hundreds of thousands) of small sources. As pointed out by Van Jacobson [10], we should not have to track millions of ants to track a few elephants. There has been recent work in this direction of keeping track of heavy hitters without consuming too much memory or processing time [4]. The work that clearly delineates this problem and provides a solution is the paper by Estan and Varghese [4]. They also provide an excellent summary of previous work in this area. They exploit the fact that there are a few heavy hitters to reduce the amount of memory required to measure these heavy sources. The basic idea is to sample packets with some probability and if the flow to which the packet belongs is not already in memory, then the flow is added to the memory, and from that point on, all packets arriving to this flow are counted. Since every packet is counted, the sampled flows are kept

in a hash table and at every packet arrival, its flow id has to be hashed into this hash table in order to increment the appropriate counter if it is already in the table. Therefore there is increased processing at each packet arrival compared to random sampling. However, since the size of the memory is reduced, they show that this scheme is easier to implement. They also give a more processing intensive multistage filter scheme to track large flows. Our approach is not to do random sampling as in all the earlier schemes but rather measure the number of two-runs that each flow generates. A flow has a run if there are consecutive arrivals from that source. As we will show in this paper, this is a very powerful technique for estimating traffic rates especially when the number of flows is large. In [4] the number of bits is used as a measure of the amount of traffic sent by a flow and the number of packets is used as a measure of the size of the flow. We can extend our approach to include this definition of traffic and flow sizes. Though there is a large body of literature in statistics dealing with the probability of runs, to our knowledge the inverse problem of determining probabilities based on observing runs has been addressed for the first time in this paper. The reason this approach works well is that

- It is very easy to detect runs.
- Measuring runs, automatically eliminates detecting and tracking small sources (ants) and automatically picks the large sources (elephants).
- The amount of memory required is reduced drastically compared to the straightforward sampling scheme.
  - For example if we want to detect the proportion of the traffic sent by a source to within an accuracy of 0.001, then the memory requirement for our approach is a factor of 1000 lower.

There are some previous papers that use coincidences to recognize large flows. For example, in CHoke [12] as well as in SRED [11], the flow id of an incoming packet is compared with the flow id of the packets in the buffer in order to bias the drop probabilities towards large flows. However neither of these explicitly estimate the traffic rate. We believe that it is possible to improve the performance of packet dropping algorithms by explicitly estimating per-flow rates.

II. Problem Definition

We consider a node in a network processing arrivals from multiple flows. An example is a router processing arrivals for multiple destination IP-addresses. Any field or combination of fields in the packet header can be defined to be a flow. We assume that the node is processing a large number of flows at any point in time. The objective of this paper is to design a *traffic rate estimator* to estimate the number of packets processed for each flow to a desired level of accuracy. There are many different schemes that can be used to estimate the traffic rates for the different flows. These schemes can be compared based on several metrics. The two metrics that we consider in this paper are:

- *Sample Size*
  Sample size is defined to be the number of samples needed to achieve a desired level of accuracy. The larger the sample size, the longer is the time needed to estimate the traffic rates. If the traffic characteristics changes over time, we would like the time scale needed to estimate the rates to be smaller than the time scale in which the traffic varies. Therefore, we would prefer a scheme with a low sampling size.
- *Memory Requirement*
  During the process of rate estimation, the estimation scheme keeps in memory traffic counts per-flow or a subset of the flows that is processed by the nodes. The memory requirement for the estimation scheme is proportional to the size of this subset of flows for which counts are maintained. We therefore use the *number of flows for which counts are maintained* as surrogate for the memory requirement. Keeping memory requirements low also leads to improved running time and ease of implementation. Keeping track of the arrivals for each flow, often referred to as per flow information, is clearly far more expensive and we argue that our scheme can be an effective substitute for accurate traffic estimation.

A. Notation

We assume that each arrival belongs to one of F flows. The rate of arrivals to flow $i \in F$ will be denoted by r_i and let $\lambda = \sum_{i \in F} r_i$ denote the total arrival rate (packets/second) to the node. Let $p_i = \frac{r_i}{\lambda}$ denote the proportion of traffic to the node that belongs to flow $i \in F$. The objective of this paper is to design an efficient scheme to estimate r_i for each $i \in F$. Since it easy to measure λ, instead of directly estimating r_i, we solve the equivalent problem of getting an estimate $\hat{p_i}$ of p_i for each $i \in F$. We then use $\lambda \hat{p_i}$ to estimate r_i. We can view p_i as the probability that an arriving packet belongs to flow i. We assume that p_i is stationary over the time in which the estimation is done. We also assume that the probability that an arriving packet belongs to a given flow is independent of all other packets. If the arrivals to a given node are dependent, i.e., if the flow id of the next packet arriving at a node is dependent on the flow id of the current packet, then we can sample the arriving stream randomly in order to mitigate this

effect. The analysis in this paper *does not assume that we look at all packets in order to count the number of runs.* The analysis carries over, if packets are sampled at random from the arriving stream. This of course, will increase the time needed to accumulate the needed number of samples.

Accuracy Requirement

We want to design a sampling scheme that for any given flow $i \in F$, will determine an estimate $\hat{p_i}$ for p_i such that $\hat{p_i} \in \left(p_i - \frac{\beta}{2}, p_i + \frac{\beta}{2}\right)$ with probability greater than α. In other words, we are willing to tolerate an error of $\pm\frac{\beta}{2}$ with probability less than α. For example, the requirement on the sampling scheme can be the following: At the end of the sampling period, given any user i determine p_i within an error of ± 0.0001 with a probability greater than 99.99%. This requirement translates to $\beta = 0.0001$ and $\alpha = 0.9999$. Throughout this paper, we use $\mathcal{N}[a, b]$ to represent a normal distribution with mean a and variance b. We use Z_α to denote the α percentile for the unit normal distribution. If $\alpha = 99.99\%$ then $Z_\alpha = 4.0$. In the next section, we outline a straightforward sampling scheme that maintains a count for each class that is seen by the node. The proportion of traffic sent by each class is estimated from these counts. We will refer to this scheme as the naive sampling scheme. We then outline RATE (Runs bAsed Traffic Estimator) which is based on sampling two-runs, in Section 4 and analyze the sample size for this scheme. We then analyze the memory requirements for the two schemes in Section 5 and show that for the same level of accuracy, the memory requirement for RATE is significantly lower. We give some applications of RATE and some numerical results in Section 6.

III. Naive Sampling Scheme

We now outline a straightforward sampling scheme to estimate the arrival rate for all the flows. The node maintains a table of all the flows that it has seen so far, along with the number of packets for each of these flows. When a packet is processed at a node, the sampling scheme first determines the flow to which the packet belongs. It then determines whether the flow is already in the table of flows that it maintains . If this flow is already in the list, then the count for that flow is incremented by one. If the flow is not on the list, then the flow is added to the list and the count is initialized to one. Once a sufficient amount of sampling has been done, the node determines the proportion of traffic for each flow in the list based on the count for that flow. For any flow not in the list, the node estimates its rate to be zero. The estimation of the proportion of traffic for each flow is based on the following result.

Theorem 1: Let $M(i, T)$ represent the number of arrivals for flow $i \in F$ after T samples. For large T,

$$\sqrt{T}\left[\frac{M(i,T)}{T} - p_i\right] \sim \mathcal{N}\left[0, p_i(1-p_i)\right]$$

where $\mathcal{N}[a, b]$ represents a normal distribution with mean a and variance b.

Proof: This follows directly from the facts that $M(i, T)$ is a binomial random variable with probability of success p_i and using a normal approximation for a binomial random variable with a large number of trails. ■

This result can now be used to determine the sample size that is needed to ensure that the estimates have the desired level of accuracy.

A. Estimating the Sample Size

Let $M(i, T)$ denote the total number of arrivals to flow i after T samples. It is easy to show that

$$\hat{p_i} = \frac{M(i,T)}{T}$$

is the maximum likelihood estimator for p_i. The α percentile confidence interval is given by

$$\hat{p_i} \pm Z_\alpha \sqrt{\frac{\hat{p_i}\,(1-\hat{p_i})}{T}}.$$

Since we do not know the value of p_i apriori, we need an upper bound on the variance in order to determine the sample size. Note that the maximum value of $p(1-p) = 0.25$ is attained when $p = \frac{1}{2}$. The minimum sample size T' needed for the naive sampling scheme in order to satisfy the accuracy requirement is given by

$$T' = \frac{4\,Z_\alpha^2\,0.25}{\beta^2} = \frac{Z_\alpha^2}{\beta^2}.$$

IV. Description and Analysis of RATE

In this section, we outline the Runs bAsed Traffic Estimator (RATE) for determining the proportion of traffic sent by each flow. The scheme is based on sampling only a subset of the arriving traffic at the node but it picks this subset carefully so that flows that send a larger proportion of the traffic are sampled more frequently. This is achieved by sampling two runs. A flow $i \in F$ is defined to have a two-run if two consecutive samples belong to flow i. Since small sources are sampled with very low probability, the list of sources that are detected will be quite small. This leads to memory efficient implementation of the rate detector. RATE detects and measures two-runs by maintaining the following information:

- *Two-Run Detecting Register R:* This register needs to hold only one flow id. The register typically holds the flow id of the last sample. If the flow id of the current flow is the same as the content of the register, then a two-run is detected, the Two-run Count Table (described below) is updated and the run detecting register R is set to null. If the flow id of the current sample is different from the content of R, then R is reset to the flow id of the current sample.
- *Two-Run Count Table TCT:* The Two-run Count Table maintains counts for the number of two-runs for each flow that has had a two-run. When a two-run is detected for a particular flow and if the flow is already in TCT then the two-run count for the table is incremented by one. If the flow for which a two-run has been detected is not in TCT, then this flow id is added to TCT and its count is initialized to one.

A. Overall Approach

The implication of setting the register to null when a two-run is detected is the following: Consider the case where there are 3 flows at the node. Assume the following arrival sequence .....21113..... When a two-run is detected for flow id 1, the register will be reset. Therefore, there is only one two-run. If the register is not reset after a two-run, then three consecutive arrivals will be counted as two two-runs. The register is reset in order to ensure that the two-runs forms a regenerative process that is easy to analyze. The relationship between the proportion and the number of runs has been studied extensively in the statistics literature but the main concentration is on determining the characteristics of the runs based on the probability. We consider the inverse problem, i.e., one of determining the probability by measuring the number of runs. The overall approach that we use is the following:

- Since the two-runs form a renewal process, we first characterize the first two moments of the inter-arrival time between two-runs. (Theorem 2).
- Since we measure the number of two-runs (and not the inter-arrival time), We use a standard result in renewal theory to relate the first two moments of the number of renewals to the first two moments of the inter-arrival time between renewals. (Theorem 3)
- We use the result of Theorem 2 and Theorem 3, to derive the first two moments for the number of two-runs in any given time. (Theorem 4).
- The number of renewals (two-runs) for flow i is a non-linear function of the proportion of traffic p_i that is sent by flow i. We can solve this non-linear (quadratic) equation in closed form to determine and estimator for p_i. In order to determine the variance of the estimator, we use a standard result in statistics (Theorem 5). This variance of the estimator is given in Theorem 6.

We now characterize the first two moments of the inter-arrival time between two-runs. It is easy to use a generating function approach to get the generating function for the inter-arrival time. However, we only need the first two moments for our paper.

Theorem 2: Let Z_i represent the inter-arrival time between two-runs for flow i. Then

$$E[Z_i] = \frac{1+p_i}{p_i^2}.$$

$$Var[Z_i] = \frac{(1-p_i)(p_i^2+3\,p_i+1)}{p_i^4}.$$

Proof: The inter-arrival time Z_i between two-runs for flow i is a renewal process. We will characterize the first two moments of Z_i. Note that we get a two-run, if there is an arrival for flow i followed immediately by another arrival from the same flow i. Let X_1 represent the random variable that represents the time to get the first arrival for flow i. If the next arrival is from flow i, then there is a two-run. Else the process starts all over again and let X_2 represent the time until we get the next arrival from flow i, and so on. Note that X_k is a geometric random variable and

$$E[X_k] = \frac{1}{p_i} \qquad \text{and } Var[X_k] = \frac{1-p}{p^2}.$$

Let $W_k = X_k + 1$. Note that

$$Z_i = W_1 + W_2 + \ldots + W_N$$

where N is a random variable that is the time until we get a two-run. Note that N itself is geometrically distributed with parameter p_i. We now compute the first two moments of Z. By Wald's result [9], we get,

$$E[Z_i] = E[W_1]E[N]$$

$$Var[Z_i] = Var[W_1]E[N] + E^2[Y_1].Var[N].$$

Note that $E[W_1] = E[X_1] + 1$ and $Var[W_1] = Var[X_1]$. Substituting these values in expressions for the moments of Z, we get

$$E[Z_i] = \frac{1+p_i}{p_i^2} \text{ and}$$

$$Var[Z_i] = \frac{(1-p_i)(p_i^2+3\,p_i+1)}{p_i^4}.$$

■

Instead of measuring the time between two-runs for a given source, it is easier to measure the number of two-runs. We now state an asymptotic result on renewal process that relates the first two moments of the number of arrivals in a given time to the first two moments of the inter-arrival time for the renewal process.

Theorem 3: Let E be a renewal event where the time between renewals has a finite mean m and variance ν^2. Then the number of renewals in n trials is asymptotically normal with mean

$$\frac{n}{m}$$

and variance

$$\frac{n\nu^2}{\mu^3}.$$

Proof: The proof of this theorem is outlined in Feller [7]. ■

In the next theorem we give the mean and variance of the number of two-runs in time T for flow $i \in F$.

Theorem 4: Let p_i represent the probability that an arriving packet is from flow $i \in F$. Let $N_2(i,T)$ denote the number of two-runs in T samples for flow $i \in F$. For large T,

$$\sqrt{T}\left[\frac{N_2(i,T)}{T} - \mu_2(p_i)\right] \sim \mathcal{N}\left[0, \sigma_2^2(p_i)\right]$$

where $\mathcal{N}[a,b]$ represents a normal distribution with mean a and variance b and

$$\mu_2(p_i) = \frac{p_i^2}{1+p_i}$$

and

$$\sigma_2^2(p_i) = \frac{(1-p_i)\ p_i^2\ \left(1+3\,p_i+p_i^2\right)}{(1+p_i)^3}.$$

Proof: Substituting the values of $m = E[Z_i]$ and $\nu^2 = Var[Z_i]$ from Theorem 2 to the result in Theorem 3, gives the mean and the variance for the number of renewals. ■

We plot the $\mu_2(p_i)$ and $\sigma_2^2(p_i)$ versus p_i in Figure 1. Note that for small values of p_i the mean and the variance are equal. In fact, for small values of p_i, the two-runs come as Poisson process with rate p_i^2. We use this fact in Section V, to characterize the memory requirements of the system.

Ultimately, we want to measure $N_2(i,T)$ and use that to get an estimate on p_i. Note that the mean of $N_2(i,T)$ is a non-linear function of p_i. We use the following standard inversion result from statistics to get error bounds on the estimate of p_i.

Theorem 5: Let X_n be a sequence of statistics such that

$$\sqrt{n}\left[\frac{X_n}{n} - \theta\right] \to X \sim \mathcal{N}[0, \sigma^2(\theta)].$$

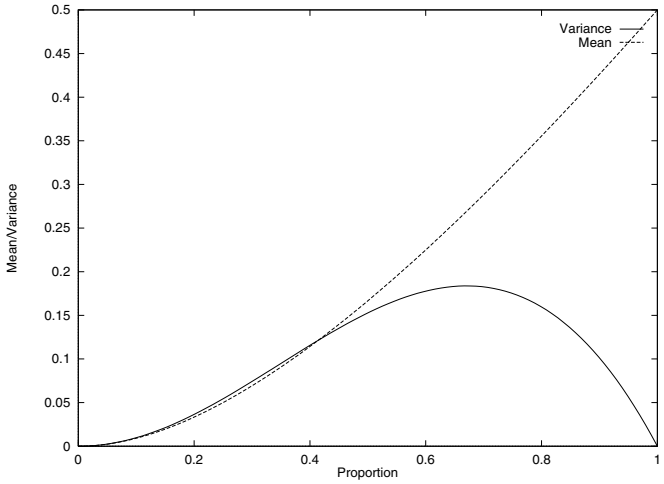

Fig. 1. Mean and Variance for the Number of two-runs

Let f be a differentiable function of one variable. Then

$$\sqrt{n}\left[f\left(\frac{X_n}{n}\right) - f(\theta)\right] \to f(X) \sim \mathcal{N}[0, \sigma^2(\theta)(f'(\theta))^2].$$

Proof: See Rao [13] for the details of the proof. ■

Since

$$\mu_2(p_i) = \frac{p_i^2}{1+p_i}$$

is monotone and increasing for $p_i \in [0,1]$, it has an inverse. We represent the inverse by $g()$ where $g(\mu_2(p_i)) = p_i$. We now apply Theorem 5, to our case.

Theorem 6: Let $N_2(i,t)$ be the number of two-runs for flow i in T samples and let $g()$ be the inverse function of $\mu_2(p_i)$. Then

$$\sqrt{T}\left[g\left(\frac{N_2(i,T)}{T}\right) - g\left(\mu_2(p_i)\right)\right] \sim \mathcal{N}\left[0, \delta_i\right].$$

where

$$\delta_i = \frac{(1-p_i)\ (1+p_i)\ \left(1+3\,p_i+p_i^2\right)}{(2+p_i)^2}. \qquad (1)$$

Proof: From Theorem 5,

$$\sqrt{T}\left[g\left(\frac{N_2(i,T)}{T}\right) - g\left(\mu_2(p_i)\right)\right] \sim \mathcal{N}\left[0, \sigma_2^2(p_i)\left[g'(\mu_2(p_i))\right]^2\right].$$

From the definition of $g()$, note that

$$g(\mu_2(p_i)) = p_i.$$

Differentiating this equation with respect to p_i we get,

$$g'(\mu_2(p_i))\,\mu_2'(p_i) = 1.$$

Therefore,

$$g'(\mu_2(p_i)) = \frac{1}{\mu_2'(p_i)}.$$

Therefore, the variance of the estimator of p_i,

$$\sigma_2^2(p_i)\left[g'(\mu_2(p_i))\right]^2 = \frac{\sigma_2^2(p_1)}{\left[\mu_2'(p_i)\right]^2}.$$

Performing the algebra, gives the result. ■

In the case of naive sampling scheme the variance of the estimator for p_i is $p_i(1-p_i)$ and for the two-run sampling scheme the variance of the estimator for p_i is δ_i in Theorem 6. In Figure 2, we plot the theoretical variance of the estimators as well as the variance of the estimator from an experimental run. We considered a fixed source whose proportion was varied from 0 to 1 in steps of 0.01. The experiment was run for a sample size of $10,000$. In other words, the experiment was run until there were 10000 arrivals into the system and the number of two-runs was measured for this source. This experiment was run 300 times and in each case, the proportion of the traffic from this distinguished source was estimated (This is outlined in the next section). The plot shows the variance of the estimate over these 300 trials. As the sample size increases the experimental results gets closer to the theoretical value. Note that the variance of the estimator from the two-runs goes to 0.25 as the proportion goes to zero. In the case of the naive sampling scheme the variance attains its maximum value of 0.25 when $p_i = 0.5$. For the two-run sampling scheme, differentiating δ with respect to p_i and equating to zero, it is easy to see that the variance attains a maximum value of 0.345 when $p = 0.362$. This fact is used to determine the sample size for the two-run sampling scheme.

B. Estimating the Proportion and Sample Size

The point estimate for $\hat{p_i}$ of p_i can be obtained by solving the following equation:

$$\frac{N_2(i,T)}{T} = \frac{\hat{p_i}^2}{1+\hat{p_i}}.$$

We get

$$\hat{p_i} = \frac{1}{2}\left(\frac{N_2(i,T)}{T} + \sqrt{4\frac{N_2(i,T)}{T} + \left(\frac{N_2(i,T)}{T}\right)^2}\right).$$

Since we measure the $N_2(i,T)$, we can estimate the value for $\hat{p_i}$. An estimate for the variance of the $\hat{p_i}$ is given by

$$\frac{(1-\hat{p_i})\,(1+\hat{p_i})\,\left(1+3\,\hat{p_i}+{\hat{p_i}}^2\right)}{T(2+\hat{p_i})^2}.$$

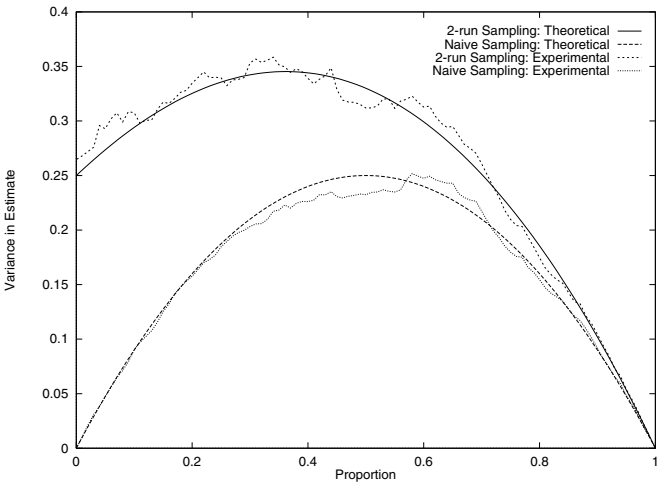

Fig. 2. Comparison of Variance

The larger the number of samples, the less is the variance of the estimator and the more accurate the estimate of p_i. The α percentile confidence interval for the point estimate is computed as follows: Let Z_α denote the α percentile of a standard normal distribution. The α percentile confidence interval for p_i is

$$\hat{p_i} \pm Z_\alpha \hat{\sigma_i}. \tag{2}$$

Let us assume that it is desired that the α percentile confidence interval should not be wider that β. As stated in Theorem 6, δ_i represents the estimate for the variance and this takes on a maximum value of 0.345. Therefore the confidence interval by Equation will not be greater than

$$2\sqrt{\frac{Z_\alpha 0.345}{T}}.$$

We set this quantity to be less than β and solve for T to determine the length of time we have to sample to reach the objective that the α percentile confidence interval is less than β. The minimum sampling time T is given by

$$T = \frac{4\,Z_\alpha^2\,0.345}{\beta^2} = \frac{1.38 Z_\alpha^2}{\beta^2}.$$

The overall algorithm for RATE is shown below:

INPUT
Confidence Interval Width β and
error probability α

INITIALIZATION

- Compute the sample size
$$T = \frac{1.38 Z_\alpha^2}{\beta^2}.$$
- Two-run Count Table $TCT \leftarrow \emptyset$.
- Two-run Register $R \leftarrow \emptyset$

PROCESSING AT EACH ARRIVAL

Flow id of current arrival is i.
If $i = R$ then
  Set $R \leftarrow \emptyset$
  If $i \notin TCT$ then
    Add i to TCT and set the count for i to one.
  If $i \in TCT$ then
    Increment the count for i.
If $i \neq R$ then
  Set $R \leftarrow i$.

PROPORTION ESTIMATION

- Number of two-runs for user i is $N_2(i)$.
- Compute $\hat{p_i}$ which is an estimate for p_i.
$$\hat{p_i} = \frac{1}{2}\left(\frac{N_2(i)}{T} + \sqrt{4\frac{N_2(i)}{T} + \left(\frac{N_2(i)}{T}\right)^2}\right).$$
- Compute $\hat{\delta_i}$, the estimate for the variance δ_i as
$$\hat{\delta_i} = \frac{(1-\hat{p_i})\ (1+\hat{p_i})\ \left(1+3\,\hat{p_i}+\hat{p_i}^2\right)}{T(2+\hat{p_i})^2}.$$
- Compute the confidence interval of the user
$$\left(\hat{p_i} - \frac{1}{2}Z_\alpha\sqrt{\hat{\delta_i}}\ ,\ \hat{p_i} + \frac{1}{2}Z_\alpha\sqrt{\hat{\delta_i}}\right).$$

Complete Description of RATE

V. Memory Requirements for the Sampling Schemes

We now compare the size of the list in the case of naive sampling sampling versus 2-run sampling. As we stated at the end of Theorem 1, the mean and the variance of the inter-arrival time are the equal when p_i is not too large. This is true in most practical situations where the proportion of traffic from any given source is expected to be not too large (less that about 20 %). In fact, it is known that the number of two-runs forms a Poisson process. This result is formally stated in the next theorem. The heuristic proof of this result is given in Aldous [1]. The proof can be made rigorous as outlined in [1].

Theorem 7: Let p_i be small and let $X(i)$ denote the time between successive two-runs. Then

$$\Pr[X(i) \geq t] \approx \exp(-p_i^2 t)$$

Proof: See Aldous [1] for a proof of this theorem. ■

The implication of the Poisson process that we are exploit is the fact that the inter-arrival time between two-runs is exponentially distributed. It is possible to derive this result directly from the approach followed in Theorem 2. Let $\xi_i(T)$ represent the indicator random variable that is set equal to one if there is at least one two-run for flow i by sample T. We want to estimate the expected memory requirement for the two-run sampling scheme. The amount of memory needed to store the results of T samples is equal to the number of different flows that have at least one two-run upto t samples. Let $L(T)$ represent the number of flows that have had at least one two-run upto T samples. Then,

$$E[L(T)] = E[\sum_{i=1}^{n} \xi_i(T)] = \sum_{i=1}^{n} 1 - \exp(-p_i^2 T). \quad (3)$$

In the case of naive sampling, let $L'(t)$ denote the number of sources that have had at least one arrival upto sample T. Note that

$$E[L'(T)] = \sum_{i=1}^{n} 1 - (1-p_i)^T. \quad (4)$$

If p_i is small then we can write this as

$$E[L'(T)] = \sum_{i=1}^{n} 1 - exp(-p_i T).$$

A. Worst Case Memory Requirements

Given a sample size T, we now want to determine the expected amount of memory required in the worst case. In other words, we want to find an allocation of p_i that maximizes the expected number of flows in TCT. We solve this problem in two steps.

- First, we fix the number of sources n and then determine the distribution of the probabilities across these n sources in order to maximize the expected memory requirement.

- We then determine the number of sources n that maximizes the memory requirement.

We first consider the memory requirement for two-run sampling case. Fix the number of sources n.

Maximizing the memory requirement can now be formulated as the following maximization problem.

$$\max\ E[L(T)] = \sum_{i=1}^{n} 1 - exp(-p_i^2 T).$$

$$\sum_{f \in F} p_i = 1.$$

$$p_i \geq 0\ \ \forall i.$$

Let p_i^* denote the probability allocation for flow i that maximizes the the list length. We assume that there are atmost n flows present.

Theorem 8: Let $S_1 = \{i : p_i^* < \frac{1}{2T}\}$ and let $S_2 = \{i : p_i^* \geq \frac{1}{2T}\}$. Then

- $p_i^* = p_j^*$ for any $i, j \in S_2$.
- $|S_1| \leq 1$.

Proof: We use $OBJ(x)$ to denote $1 - exp(-x^2 T)$. The objective function is convex in the interval $[0, \frac{1}{2T})$ and is concave in the interval $[\frac{1}{2T}, \infty)$. The objective function is shown in Figure *.

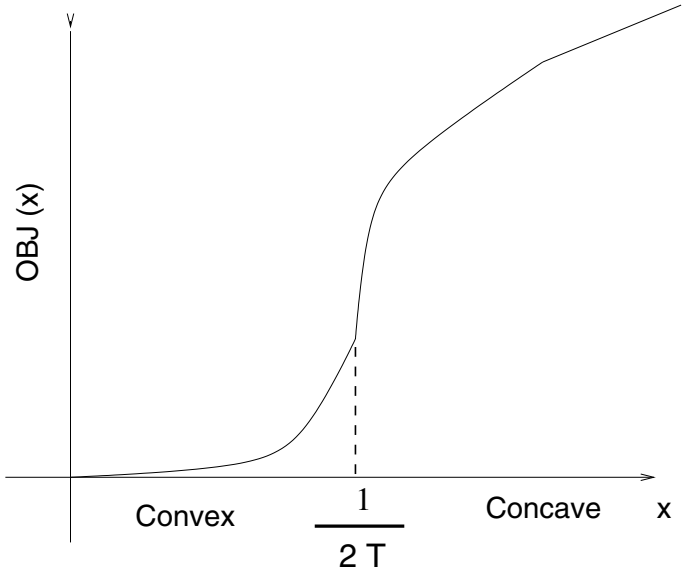

Fig. 3. Objective Function

Let $p_i^* > p_j^*$ for some $i, j \in S_2$. Let $\bar{p} = 0.5\left(p_i^* + p_j^*\right)$. Since the function is concave in this interval,

$$2OBJ(\bar{p}) \geq OBJ(p_i^*) + OBJ(p_j^*).$$

This proves the first part of the result. To see the second fact, note that if $|S_1| > 1$ then there exist $i, j \in S_1$. Assume that $p_i^* \geq p_j^*$. Since the function is convex in the interval,

$$OBJ(p_i^* + \delta) \geq OBJ(p_i^*) + \delta OBJ'(p_i^*)$$

$$OBJ(p_j^* - \delta) \geq OBJ(p_j^*) - \delta OBJ'(p_j^*)$$

Adding these two inequalities and using the fact that $OBJ'(p_i^*) \geq OBJ'(p_j^*)$, gives

$$OBJ(p_i^* + \delta) + OBJ(p_j^* - \delta) \geq OBJ(p_j^*) + OBJ(p_j^*).$$

This process can be repeated until one of the variables goes to zero or the other goes to $\frac{1}{2T}$. This elimination can be done repeatedly until there is only one variable in S_1. ■

Ignoring the (at-most) one flow that belongs in $|S_1|$, we now only consider the variables in S_2 which are all equal. It is easy to make the analysis exact at the expense of more notation.

Theorem 9: Let $L(T)$ represent the number of flows in TCT after T samples. Then

$$E[L(T)] \leq 0.638\sqrt{T}$$

in the worst case.

Proof: From Theorem 9 , the objective function is maximized when

$$p_i = \frac{1}{n}\ \forall i.$$

Therefore for a fixed value of n, the maximum expected list length is

$$n\left(1 - \exp(-\frac{T}{n^2})\right).$$

We now want to determine the value of n for which this function is maximized. Assuming that n is continuous, and differentiating this expression with respect to n we get

$$1 - \exp(-\frac{T}{n^2})\left(\frac{2T}{n^2} + 1\right) = 0.$$

If we set $\frac{T}{n^2} = w$ we can rewrite the expression as

$$1 - e^{-w}\left(2w + 1\right) = 0.$$

Solving for w we get $w = 1.255$. Therefore,

$$n = \sqrt{\frac{T}{1.255}} = 0.89\sqrt{T}.$$

Setting this value of n in the objective function, the maximum expected memory requirement is given by

$$E[L(T)] \leq 0.638\sqrt{T}.$$

■

Corollary 10: Given the width of the confidence interval β and the error probability α, the maximum expected memory

$$[L(T)] = \frac{0.74 Z_\alpha}{\beta}.$$

Proof: The sample size in the case of two-run sampling is

$$\frac{1.38 Z_\alpha^2}{\beta^2}.$$

Substituting this result in the above theorem gives the required result. ■

In fact, using standard inequalities on the tail of the binomial distribution it is easy to show that in the case of equal $0.89\sqrt{t}$ sources, that with very high probability

$$L(T) \leq 3\sqrt{T}.$$

Note that this expression is independent of the number of sources or their characteristics and just depends on the number of samples. The number of samples, in turn depends on the amount of accuracy needed in the estimation. Therefore once the desired accuracy is known, then it is easy to estimate the amount of memory needed as about

$$\frac{\gamma Z_\alpha}{\beta}$$

where $\gamma \approx 3$. This ensures that with a very small probability there will be an overflow of the two-run counting table.

B. Discussion

Consider the case where $\beta = 0.002$ with the error probability $\alpha = 99.75$ with a corresponding $Z_\alpha = 3.0$ The sample size for the naive sampling is 2.25×10^6 and for RATE is 3.1×10^6. Assume that there are 100000 flows. These memory requirement for RATE compared to naive sampling can be interpreted in one of two ways:

- For a given value of β, say $\beta = 0.001$, the memory requirement in the case of RATE is about a factor of 1000 less than the naive sampling scheme.
- If the amount of memory M is fixed, then the accuracy of RATE is proportional to $\frac{1}{M}$ whereas the the accuracy of the naive sampling scheme is proportional to $\frac{1}{\sqrt{M}}$. Note the similarity of this interpretation to the results in [4].

VI. Experimental Results

In this section, we assess the performance of RATE by simulations. We compare the performance of RATE to the naive sampling scheme. We considered a system with 100000 flows. Most of the flows send small amounts of traffic and a few sources, varying from a few tens to a few hundred sources are larger sources. In general, the larger the source, the easier it is for RATE to detect and estimate the rate for this source. The general conclusions are the following:

- RATE performs extremely well in detecting and estimating the traffic sent by the different flows.
- The number of samples needed to get the desired level of estimation accuracy is typically far less than the theoretical sample size.
- The memory requirement in practice is *less than* the theoretical values.
- RATE detects the large flows very quickly and the longer time is needed to estimate the rates.
- RATE is robust with dependent arrivals. This seems to be the case because the intermingling of a large number of sources reduces the dependencies between different arrivals. In the case where there are a small number of sources, the dependency can be reduced by randomly sampling the incoming stream to detect and count runs. (We only monitor runs and use random sampling only to reduce dependencies).

We now give some results to show the performance of the algorithm. In the example that follows there were 100000 flows. The accuracy level $\beta = 0.0002$ with $\alpha = 99.99\%$. In other words, we want to estimate the proportion ± 0.0001 with error probability less than 0.0001. This corresponds to a $Z_\alpha \approx 4.0$. The sample size for two-runs is 550×10^6. We assumed that except for about 100 sources that are relatively large (see the Figure 3 for the proportions) the remaining traffic is about evenly distributed across the remaining sources.

- In the Figure 3 we show the actual and estimated proportions after 100000 samples. The $+$ in the plot is the actual proportion and $\times$ is the estimated value. The estimated value will be in the same vertical line as the proportion. The closer these two are, the closer is the estimate. We only plot the proportions for sources that send more than 0.0001 proportion of the traffic. All source that have not had two-runs are estimated to be zero. Note that even at this stage RATE has identified the large sources pretty effectively. The memory size (number of different flows in memory) is **92**.

- Figure 4 shows the same quantities after 10×10^6. samples. Note that in this case the estimated values are closer to the actual values than the first plot. The memory size at this stage is **101**.
- The x-axis in Figure 5 is the actual proportion and the y-axis is the estimated proportion after all the sampling is done. The fact that the points lie on the 45 degree indicates that the quantities are estimated accurately. (We also show the 0.001 band around the values, the actual accuracy is 0.0001 but showing this on the plot obfuscates the points).The memory size is **108**. Note that counts are maintained for only 108 out of the 100000 flows.
- In the case of naive sampling the sample size is $\approx 400 \times 10^6$. The memory size (the number of elements in the list) at the end of the sampling interval is **82453**.
- Note that the two-run sampling scheme needs three orders of magnitude less memory space.

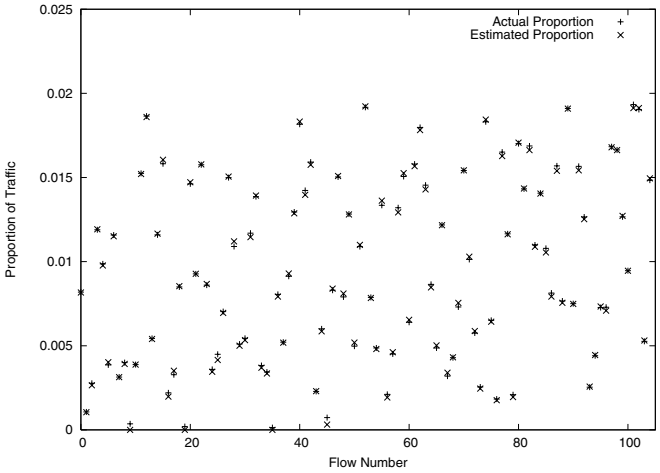

Fig. 4. Performance of RATE After 100000 Samples

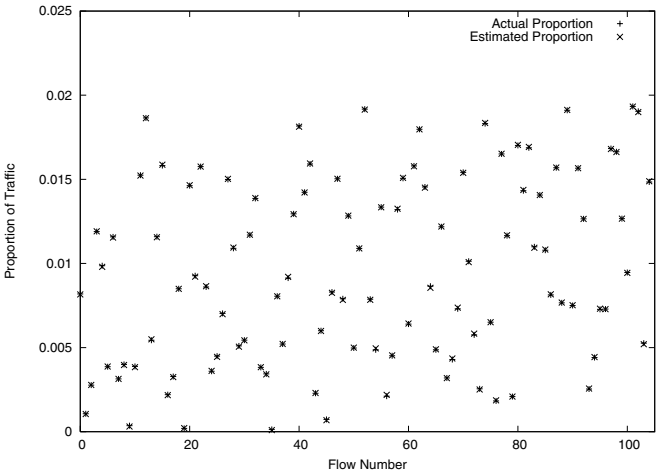

Fig. 5. Performance of RATE After 10×10^6 Samples

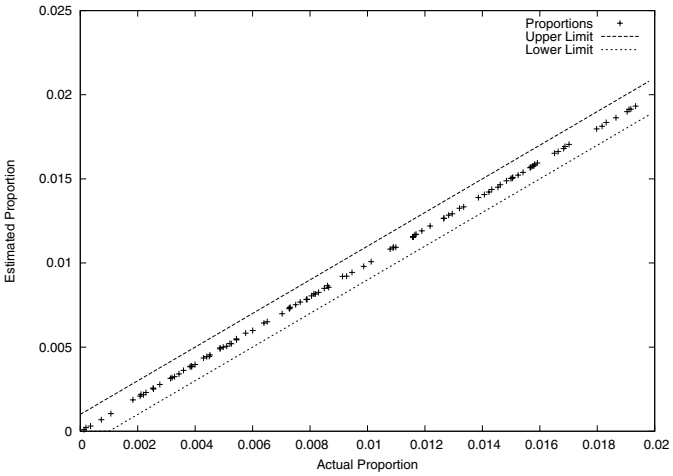

Fig. 6. Performance of RATE at the end of the Sampling Process

VII. Some Comments

One natural question that arises from the analysis in the paper is the following: Will there be any additional memory savings if we use higher order runs, for example 3-runs. It turns out that the variance of the estimate of the proportion in the case of k-runs for $k > 2$ goes to infinity as p_i goes to zero. If we derive results analogous to Theorem 4 and 7, we can show that, unlike two-runs where the number of samples is 1.38 times the number of samples in the naive sampling case, the number of samples in the case of 3-runs or more increases the number of samples exponentially while not having a significant impact on the memory size. We do not describe this effect in more detail here. We want to emphasize again that if the arrivals are correlated then it is easy to get rid of this by sampling for two-runs randomly. This will increase the sampling time but all the results still hold. We presented a new sampling mechanism for estimating per-flow traffic using the idea of counting runs. This leads to a significant savings both in per packet processing as well as in the amount of memory required to perform the estimation. We theoretically bounded the amount of sampling as well as the memory requirement for our system. We are currently testing a practical implementation of the system as well as comparing

the performance of the system to other sampling schemes.

REFERENCES

[1] Aldous, D., *Probability Approximations via the Poisson Clumping Heuristic*, Springer-Verlag, 1987.

[2] Duffield, N, Lund, C., and Thorup, M., "Charging from Sampled Network Usage", *S*IGCOMM internet Workshop 2001.

[3] Duffield, N, and Grossglauser, M., "Trajectory Sampling for Direct Traffic Observation", *P*roceedings of ACM SIGCOMM 2000.

[4] Estan, C, and Varghese, G., "New Directions in Traffic Measurement and Accounting", *P*roceedings of ACM SIGCOMM 2002.

[5] Fang, W, and Peterson, L., "Inter-as Traffic Patterns and their Implications", *P*roceedings of IEEE GLOBECOM 1999.

[6] Feldmann, A. et al., "Deriving Traffic Demands for Operational IP Networks: Methodology and Experience",*P*roceedings of ACM SIGCOMM'2000.

[7] Feller, W., *An Introduction to Probability Theory and Its Applications*, Vol 1, John Wiley, 1968.

[8] Feng, W. et al., "The Blue Queue Management Algorithms", *IEEE/ACM Transactions on Networking, Vol.10, Number 4, 2002*.

[9] Heyman, D.P., and Sobel, M.J., *Stochastic Models in Operations Research, Vol 1.*, McGraw Hill, 1982.

[10] V. Jacobson, *E*nd to End Research Meeting, June 2000.

[11] T. J. Ott, T. V. Lakshman, and L. H. Wong, "SRED: Stabilized RED", *P*roceedings of INFOCOM'99, pp. 1346-1355, Mar. 1999.

[12] R. Pan, B. Prabhakar, and K. Psounis, "CHOKe A Stateless Active Queue Management Scheme for Approximating Fair Bandwidth Allocation", *P*roceedings of INFOCOM'99, pp. 2:942-951, Mar. 2000.

[13] Rao, C.R., *Linear Statistical Inference and its Applications*, John Wiley, 1973.