

Fast, Memory-Efficient Traffic Estimation by Coincidence Counting

Fang Hao Murali Kodialam T.V. Lakshman Hui Zhang

Bell Laboratories
Lucent Technologies
101 Crawfords Corner Road
Holmdel, NJ 07733, USA

{fangh, muralik, lakshman, hzhang2}@bell-labs.com

Abstract

We consider the problem of fast estimation of flow rates in backbone network links with possibly millions of flows. Accurate flow rate estimation is necessary for network traffic management, network planning, measuring compliance to service level agreements, and network security. Ideally, a rate estimation scheme should have short estimation times with provable bounds on estimation error, be low in memory usage, and be easily implementable in hardware for operation at high speeds. We develop such a scheme, and achieve upto two orders of magnitude speed-up in estimation time over the previously proposed two-runs-based RATE scheme [5]. The speedups are achieved without a significant increase in memory usage, by using coincidences instead of runs. Counting coincidences has a higher processing overhead than detecting two-runs, but this higher overhead is not significant for a hardware implementation. We show that the proposed scheme is faster and more accurate than other recently proposed schemes such as ACCEL-RATE [4] and Smart Sampling [1]. The faster estimation time of the new scheme has many benefits including quicker detection of incipient denial of service attacks. We prove bounds on the scheme's accuracy, memory needs, and also show that it performs well by simulations that use both synthetic and real traffic traces.

1 Introduction

Because of the connectionless nature of IP networks, backbone IP routers generally do not need to maintain per-flow state. While this is useful for scaling, this lack of per-flow state necessitates the development of new techniques for the gathering of flow-level statistics needed for various network management purposes. As an example, on a high-speed link it is generally difficult to find out how many packets are sent between pairs of hosts during a certain time period, or how the traffic pattern shifts over time. However, flow-rate statistics of this type are often needed for a variety of purposes such as monitoring for service level compliance,

detecting potential denial-of-service attacks, network traffic management and planning, etc. Hence, designing an efficient mechanism that can make fast and accurate flow rate estimations can greatly help network service providers.

Flows in IP networks can be defined using any information in the packet. The most frequently used definition is one based on five-tuples in the IP header, i.e., all packets that have the same (SourceIP, DestinationIP, SourcePort, DestinationPort, ProtocolID) are considered to belong to the same flow. We use this definition for explanatory purposes in the paper, but the scheme presented is applicable to other definitions as well.

A good flow estimation scheme should have the following features:

- It should rapidly estimate flow rates to the specified accuracy. This implies that, for a specified accuracy, the shorter the sampling time the better the scheme. Rapid estimation is critical for fast detection of anomalous events such as DoS attacks.
- The scheme should be suitable for real-time processing of traffic streams. This implies that the operations performed should be simple and preferably amenable to hardware implementation.
- The scheme should be memory efficient. A backbone link may have millions of flows, and it should not be necessary to maintain state for a large fraction of the flows in order to gather statistics on a few. For high speed links, state information may have to be kept in SRAMs and large amounts of SRAM will render the scheme too expensive to be practical.

The naive approach of directly counting the number of packets for each flow (called *naive counting* in the rest of the paper) satisfies the first two requirements but the memory requirement is quite large. This is due to the fact that the total number of different flows on an IP backbone link ranges from hundreds of thousands to millions; to keep counters for all flows and access the counters for each arrival is too expensive.

1.1 Prior work

There has been much recent research on flow measurement [5, 1, 2, 4] and packet sampling [8, 9, 11, 12, 13]. The work of Estan and Varghese [2] presented a sampling method that first selects the *heavy hitters*, i.e., flows with rate above a certain threshold (say, 1% of the entire traffic), and then counts all packets belonging to these heavy hitters. For this scheme, deriving the number of samples needed to achieve a specified estimation accuracy does not appear to be easy.

Duffield et al. [1] proposed the Smart Sampling mechanism that is based on random sampling of m packets out of n arrivals. Longer packets are given more weight during sampling in order to get unbiased estimation of flow bit-rates. Note that sampling accuracy is fixed for a given amount of available memory; the sampling result can be very inaccurate when m is small.

Kodialam et al. [5] proposed RATE, a flow estimation mechanism based on counting *two-runs* (flow id matches for two consecutive samples from the traffic stream) For a given accuracy level, RATE requires worst-case sampling time slightly longer than (1.38 times) the naive counting scheme. However, it uses significantly less memory (square root of number of samples). A drawback is that for the same sampling time, RATE has much worse accuracy than naive counting for flows with low rates.

ACCEL-RATE [4] was proposed as an enhancement to the original RATE scheme. In ACCEL-RATE, each arrival packet is hashed into multiple buckets based on its flow id, so that packets of the same flow are more “concentrated” within each bucket. As a result, a larger number of two-run samples can be generated for each flow. It is shown that under uniform hashing the sampling time, in comparison to RATE, can be reduced to $\frac{7.3}{k}$ where k is the number of buckets. This is achieved with about 2.7 times more memory needs than RATE. Note that, however, such sampling time reduction only holds when hashing is uniform, which implies the rate of the largest flow should be significantly less than $\frac{1}{k}$. Hence ACCEL-RATE is best suited for cases where the maximum flow rate is small, and a loose upper bound is known a priori.

In general, all the proposed mechanisms tend to trade reduced estimation accuracy and/or increased sampling time for lower memory cost. For a given sampling time, naive counting scheme can still produce the most accurate results regardless of flow rate but at the expense of much higher memory requirements. We use the width of the confidence interval of the naive counting scheme as the benchmark to determine the effectiveness of the sampling schemes developed in this paper.

1.2 Our contributions

In this paper, we propose a new traffic estimation mechanism called *Coincidence bAsed Traffic Estimator* (CATE).

The new scheme works by keeping registers for k previous arrivals and comparing the new arrival with each of them. Two-run is redefined as a coincidence of matching between the new arrival and one of the previous arrivals in registers. Note that each new arrival may generate as many as k two-runs in the new scheme. Similar to RATE scheme, recorded two-run count of each flow is used to calculate flow rate at the end of sampling.

Through thorough analysis and extensive simulations, we show that CATE is better than previous approaches in many ways:

- CATE approaches *naive counting* in terms of accuracy for *both* small and large flows when k is reasonably large (e.g., 10), while requires significantly less memory size than *naive counting*. This is a nice property that neither RATE nor ACCEL-RATE has.
- For a given k , our new scheme is guaranteed to be order of k times more accurate than RATE for small flows with rate lower than $\frac{1}{2k-1}$, and almost as accurate as naive counting for larger flows.
- The new scheme is more robust than ACCEL-RATE. In ACCEL-RATE *hashing* results in *uncertainty*: even with the same rate, the small flows that share hashing buckets with large flows will be much more difficult to be detected accurately than the small flows that do not. But the new scheme treats all flows equally.
- CATE uses multiple comparisons and this may be amenable to parallel or pipelined hardware implementation.

The rest of the paper is organized as follows: A formal problem definition is in Section 2. Section 3 outlines CATE scheme and describes the implementation details. Section 4 presents the analysis of CATE. The experimental results of CATE along with comparisons with RATE, ACCEL-RATE, and the smart sampling approach in [1] are in Section 5. Lastly, Section 6 concludes the paper.

2 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. In the analysis in this paper, we assume that the estimation of the rates is performed after N arrivals to the system. Given a fixed number of arrivals N , to the system, the objective of this paper is to estimate the proportion of flow belonging to the different flows as accurately as possible. We compare

different estimation schemes based on the width of the α confidence interval for the estimator for the proportion.

2.1 Notation

We assume that each arrival belongs to one of F flows. The rate of arrivals to flow $f \in F$ will be denoted by r_f and let $\lambda = \sum_{f \in F} r_i$ denote the total arrival rate (packets/second) to the node. Let $p_f = \frac{r_f}{\lambda}$ denote the proportion of traffic to the node that belongs to flow $f \in F$. The objective of this paper is to design an efficient scheme to estimate r_f for each $f \in F$. Since it is easy to measure λ , instead of directly estimating r_f , we solve the equivalent problem of getting an estimate \hat{p}_f of p_f for each $f \in F$. We then use $\lambda \hat{p}_f$ to estimate r_f . We can view p_f as the probability that an arriving packet belongs to flow f . We assume that p_f 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 controlled by some closed loop control like TCP then arrivals that are close to each other will be dependent. We can sample randomly in order to reduce this dependence. We now give the accuracy requirement for RATE [5] and the modified accuracy requirement for CATE.

2.2 Accuracy Requirement for RATE

The traffic estimation algorithm RATE satisfies the following accuracy requirement. For any given flow $f \in F$, RATE will determine an estimate \hat{p}_f for p_f such that $\hat{p}_f \in \left(p_f - \frac{\beta}{2}, p_f + \frac{\beta}{2}\right)$ with probability greater than α . In other words, RATE will tolerate an error of β 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 flow f determine p_f within an error of ± 0.005 with a probability greater than 99.75%. This requirement translates to $\beta = 0.01$ and $\alpha = 0.9975$. 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 $\alpha = 99.75\%$ then $Z_\alpha = 3.0$.

2.3 Accuracy Requirements for CATE

In order to relax the accuracy requirement, we assume that the proportion for most flows lie below some threshold proportion Δ and we want the estimation to be accurate in the range $[0, \Delta]$. In case there are flows with proportion greater than Δ , we still want the estimation to have a guaranteed performance but we are willing to sacrifice the quality of the guarantee somewhat. Formally, we are given a threshold proportion $0 \leq \Delta \leq 1$ and a parameter $\theta \geq 1$, and we

want to design a sampling scheme such that:

$$\hat{p}_f \in \begin{cases} \left(p_f - \frac{\beta}{2}, p_f + \frac{\beta}{2}\right) & \text{if } p_f \leq \Delta \\ \left(p_f - \frac{\theta\beta}{2}, p_f + \frac{\theta\beta}{2}\right) & \text{if } p_f > \Delta \end{cases}$$

with probability greater than α . In other words, we are willing to tolerate an error of β with probability less than α for all $p_f \leq \Delta$, and an error of $\theta\beta$ with probability less than α for all $p_f > \Delta$. For example, If we set $\Delta = 0.01$, $\beta = 0.0001$, $\alpha = 99.9\%$, and $\theta = 10$, then the modified accuracy requirement states that we want to estimate all flows below 0.01 with an accuracy of 0.0001 and for flows greater than 0.01 with an accuracy of 0.001 with probability greater than 99.9%. We now describe the coincidence based traffic estimator that decreases the estimation time significantly compared to RATE while meeting these modified accuracy requirement.

The two dimensions along which we measure the performance of the algorithm are:

- *Sample Size*: Sample size is defined to be the number of arrivals needed to perform the estimation. We use the terms *estimation time*, *sampling time* and sample size interchangeably.
- *Memory Size*: We use the number of flows that are tracked as a surrogate for the amount of memory required¹.

The main question that we address in this paper is the following: Is it possible to reduce the estimation time (sample size) with a modest increase in memory and a slightly relaxed accuracy requirement?

3 Coincidence based Traffic Estimator (CATE):

The estimation of the proportion of each flow in RATE is based on counting the number of two runs in the incoming stream. CATE speeds up the estimation time of RATE by making multiple comparisons for each arrivals and counting the number of coincidences. Since multiple comparisons are made for each arrival, the coincidence processes for different arrivals are correlated unlike the runs process in RATE. This makes the analysis of CATE more difficult than RATE. This decrease in sample size comes at the cost of increase in the amount of processing needed for each arrival. But such increased processing is easy to implement in hardware. Similar to RATE, the estimation process in CATE maintains two tables.

¹As we describe next, CATE uses multiple registers for past arrivals, which may incur additional memory cost. But this is typically small (e.g., 50 or 100) and hence insignificant compared to the number of tracked flows

Predecessor Table (PT)

The predecessor table maintains the last k arrivals to the system. Note that this is a generalization of RATE, where only the last arrival to the system is maintained in the two run register.

Coincidence Count Table (CCT)

Upon each arrival into the system, we compare the flow f of the current arrival, to the flow of the last k arrivals to the system that is stored in the PT. The number of coincidences for the current flow-id f is the number of times the flow-id f occurs in the predecessor table. Therefore, for any arrival there can be potentially k coincidences. The Coincidence Count Table (CCT) keeps count of the number of coincidences for each flow-id. This table is updated whenever any arriving flow has one or more coincidences. If an arriving flow f has $l \leq k$ coincidences and f is not in CCT then f is added to CCT with a count of l . If flow f is already in the CCT, then the count for f is incremented by l .

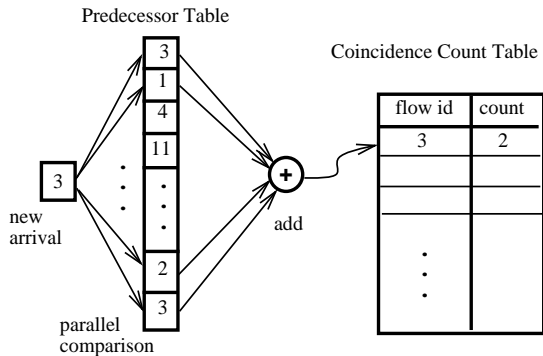


Figure 1: Illustration of CATE operation

Figure 1 illustrates how CATE works. The flow id of a new arrival packet is compared with each of last k arrivals in PT. Such comparison can be done in parallel using various hardware implementations such as FPGA or CAM. The result of each comparison is either 1 (when there is a match) or 0 (when there is no match). Such results are added together and used to update the CCT. The new arrival also replaces the oldest packet in PT after comparison is done.

3.1 Main Result

We have the following theorem to describe the main result of this paper:

Theorem 1 *Given the modified accuracy requirement, let N^R be the number of samples required by RATE and N^C be the number of samples required for CATE. If the accuracy requirement for large flows can be relaxed to*

$$\theta \geq \sqrt{\frac{\left(\frac{1}{\Delta} + 1\right)^2}{3\left(\frac{2}{\Delta} + 1\right)}}$$

then setting

$$k = \frac{1}{2} \left(\frac{1}{\Delta} + 1 \right)$$

gives

$$N^C = \frac{1}{0.23 \left(\frac{1}{\Delta} + 1 \right)} N^R.$$

Proof:

Refer to Appendix A for the proof. \diamond

CATE estimates small flows quickly and accurately, and still gives a good estimation accuracy for large flows. For example we want to estimate all $p_f < 0.01$ to an accuracy of 0.0001 with probability greater than 99.9 % and all $p_f > 0.01$ to an accuracy of 0.001 with probability greater than 99.9 %. Since $\Delta = 0.01, \theta = 10$, based on Theorem 1 RATE will need only $\frac{1}{23}$ of the samples that RATE needs by choosing $k = 50$.

In fact the performance of CATE for flows with $p_f > \Delta$ can be much better than the statement of the theorem, especially if p_f is close to Δ . We can show that if $p_f > \Delta$ then the actual accuracy amplification factor

$$\theta \leq \frac{p_f}{\Delta}.$$

This bound will be much tighter if p_f is close to Δ . As an example, suppose that we have enough memory to set $k = 500$ and want to estimate all $p_f < 0.001$ to an accuracy of 0.0001 with probability greater than 99.9 % and we do not have any explicit requirement for $p_f > 0.001$. In this case CATE needs only $\frac{1}{230}$ of the samples that RATE needs, and still gives all $p_f > 0.01$ to an accuracy of $\min\{0.0001 \frac{p_f}{\Delta}, 0.002\}$ with probability greater than 99.9%.

4 Analysis of CATE

The analysis of CATE is significantly more complicated than RATE. This is due to the fact that the different comparisons in CATE are not independent. Therefore, we need to account for the covariances between the different comparisons in order to accurately compute the variance of the estimators of the proportions. In the case of RATE, the runs process is a renewal process and therefore easier to analyze. In this section we focus on flow-id f and assume that the predecessor table maintains the last k arrivals to the system. Therefore when there are N arrivals to the system, we would have made kN comparisons. We make less than k comparisons for the first k arrivals. Since k is small compared to N , we ignore this in the rest of the analysis. (Accounting for this, makes the notation more complicated without affecting the results significantly). We assume that arrivals to the system are an *iid* process where the probability that an arrival belongs to flow f is given

by p_f . We label the arrivals 1 to N based on the arrival sequence. Let

$$C_{ij}(f) = \begin{cases} 1 & \text{arrivals } i \text{ and } j \text{ belong to flow } f \\ 0 & \text{otherwise} \end{cases}$$

Let $M(N, f)$ denote the number of coincidences for flow f after N arrivals. In CATE with k entries in the predecessor table, we measure coincidences between the current arrival and the last k arrivals. Therefore,

$$M(N, f) = \sum_{i \leq N} \sum_{j=i-k}^{i-1} C_{ij}(f).$$

4.1 Correlation among the comparisons

Before we study the correlation structure of the comparisons, we first state the following elementary result.

Lemma 2 *Let $C_{ij}(f)$ be as defined above. Then*

$$E[C_{ij}(f)] = p_f^2$$

and

$$\text{Var}[C_{ij}(f)] = p_f^2(1 - p_f^2).$$

Proof:

This result follows directly from the assumption that arrivals are independent and that the probability that an arrival belongs to flow f is p_f . \diamond

In CATE, the comparisons are not always independent of each other. To see this, let's use the comparisons $C_{ij}(f)$ and $C_{im}(f)$ ($i \neq j \neq m$) as an example. Note that $P(C_{ij}(f) = 1) = p_f^2$ due to the independence of arrivals. But $P(C_{im}(f) = 1 | C_{ij}(f) = 1) = p_f$ because the conditioning already implies that packet i belongs to flow f . In general, for any pair of comparisons $C_{ij}(f)$ and $C_{lm}(f)$, are independent if and only if all the four indices are distinct. If any two of the indices are identical, then the comparisons are dependent. For example, $C_{ij}(f)$ and $C_{im}(f)$ are dependent. The next result gives the correlation between the random variables $C_{ij}(f)$ and $C_{im}(f)$.

Lemma 3 *Consider $C_{ij}(f)$ and $C_{im}(f)$ for $i - k \leq j, m \leq i - 1$. Then*

$$\text{Cov}(C_{ij}(f), C_{im}(f)) = p_f^3(1 - p_f).$$

Proof:

Let $\tau = \text{Cov}(C_{ij}(f), C_{im}(f))$. Then,

$$\begin{aligned} \tau &= E[(C_{ij}(f) - E[C_{ij}(f)])(C_{im}(f) - E[C_{im}(f)])] \\ &= E[(C_{ij}(f) - p_f^2)(C_{im}(f) - p_f^2)] \\ &= p_f^4 - 2p_f^2 E[(C_{ij}(f) + E[C_{ij}(f)]C_{im}(f))] \\ &= E[C_{ij}(f)C_{im}(f)] - p_f^4 \\ &= p_f^3(1 - p_f) \end{aligned}$$

where the third equality follows from the fact that

$$E[C_{ij}(f)] = E[C_{im}(f)]$$

last step follows from the fact that C_{ij} and C_{im} are both one if and only if arrivals i, j and m all belong to the flow f , which happens with probability p_f^3 . \diamond

In fact, it is easy to show that the covariance is $p_f^3(1 - p_f)$ for any two comparisons that are correlated. We can now derive the mean and the variance for the number of coincidences for flow f .

4.2 Expectation and variance of $M(N, f)$

Lemma 4 *Let $M(N, f)$ denote the number of coincidences for flow f after N arrivals to the system. Then*

$$E[M(N, f)] = Nkp_f^2$$

and

$$\text{Var}[M(N, f)] = Nkp_f^2(1 - p_f^2) \left[1 + \frac{2(2k-1)p_f}{(1+p_f)} \right].$$

Proof:

Note that

$$\begin{aligned} E[M(N, f)] &= E\left[\sum_{i \leq N} \sum_{j=i-k}^{i-1} C_{ij}(f)\right] \\ &= Nkp_f^2. \end{aligned}$$

To simplify the notation, we assume that we index the comparisons using a single index m where $I_m(f)$ is set to one if comparison m results in a coincidence for flow f . The variance can be computed as follows:

$$\begin{aligned} \text{Var}[M(N, f)] &= E[M^2(N, f)] - (E[M(N, f)])^2 \\ &= E[M^2(N, f)] - (Nkp_f^2)^2 \end{aligned} \quad (1)$$

$$\begin{aligned} E[M^2(N, f)] &= E\left[\sum_{i=1}^{Nk} (I_i^2(f)) + \sum_{i=1}^{Nk} \sum_{j:1 \leq j \leq Nk, j \neq i} I_i(f)I_j(f)\right] \end{aligned} \quad (2)$$

$$\begin{aligned} E\left[\sum_{i=1}^{Nk} (I_i^2(f))\right] &= Nkp_f^2 \\ E\left[\sum_{i=1}^{Nk} \sum_{j:1 \leq j \leq Nk, j \neq i} I_i(f)I_j(f)\right] &= E\left[\sum_{i=1}^{Nk} \left(\sum_{j:1 \leq j \leq Nk, j \neq i} I_i(f)I_j(f)\right) + \sum_{l:1 \leq l \leq Nk, l \neq i} I_i(f)I_l(f)\right] \end{aligned} \quad (3)$$

$$\begin{aligned}
&= \sum_{i=1}^{Nk} (2(2k-1)p_f^3 + (Nk-1-2(2k-1))p_f^4) \\
&= Nk((Nk-1)p_f^4 + 2(2k-1)(p_f^3 - p_f^4))
\end{aligned}$$

Therefore,

$$\begin{aligned}
\text{Var}[M(N, f)] &= Nk[p_f^2 - p_f^4 + 2(2k-1)(p_f^3 - p_f^4)] \\
&= Nk(p_f^2 - p_f^4) \left(1 + \frac{2(2k-1)p_f}{(1+p_f)}\right).
\end{aligned}$$

◇

Notice that $Nkp_f^2(1-p_f^2)$ is the variance of $M(N, f)$ when all samples are independent from each other. Therefore, the correlation among samples in CATE increase the variance of $M(N, f)$ by a factor of $\frac{2(2k-1)p_f}{(1+p_f)}$. Since we know the mean and the variance for the number of coincidences, we now use the central limit theorem to obtain a normal approximation for the number of coincidences and then use the result to estimate the proportions. The next theorem gives the expression for the estimator of the proportion along with its variance.

Theorem 5 *Let $M(N, f)$ represent the number of coincidences for flow f after N arrivals for CATE with k comparisons for each arrival. Then,*

$$\sqrt{Nk} \left[\sqrt{\frac{M(N, f)}{Nk}} - p_f \right] \sim \mathcal{N} [0, \sigma_f^2]$$

where

$$\sigma_f^2 = \frac{(1-p_f^2)(1 + \frac{2(2k-1)p_f}{(1+p_f)})}{4} \quad (4)$$

Proof:

Though the comparisons are not independent, the comparisons are a stationary k^2 -dependent sequence with finite expectation and variance. Following the central limit theorem for dependent sequences [3], we can show that for large N ,

$$\sqrt{Nk} \left[\frac{M(N, f)}{Nk} - p_f^2 \right] \sim \mathcal{N} [0, \delta_f^2]$$

where

$$\delta_f^2 = p_f^2(1-p_f^2) \left[1 + \frac{2(2k-1)p_f}{(1+p_f)} \right]. \quad (5)$$

The above result can be shown as in Theorem 5 in [5]. ◇

Therefore, the point estimate for \hat{p}_f of p_f is

$$\hat{p}_f = \sqrt{\frac{M(N, f)}{Nk}},$$

and an estimation for the variance of the \hat{p}_f is

$$\hat{\sigma}_f^2 = \frac{(1-\hat{p}_f^2)(1 + \frac{2(2k-1)\hat{p}_f}{(1+\hat{p}_f)})}{4}.$$

We now consider this expression for the variance of the estimator and derive upper bounds on its value. This bound is derived in two regions. The first upper bound on the variance hold in the entire $[0, 1]$ range and is a function of k . The second bound on the variance is a constant independent of k and holds when the proportion is below a threshold.

Lemma 6 *Let*

$$\sigma_f^2 = \frac{(1-p_f^2)(1 + \frac{2(2k-1)p_f}{(1+p_f)})}{4}.$$

Then

$$\sigma_f^2 \leq \frac{k^2}{4k-1}.$$

$$\sigma_f^2 \leq 0.75 \quad \text{if } p_f < \frac{1}{2k-1} \quad \forall k$$

Proof:

Setting the derivative of the variance with respect to p_f to zero gives us the first upper bound. When $p_f < \frac{1}{2k-1}$,

$$\sigma_f^2 \leq \frac{(3-2p_f-p_f^2)}{4}.$$

This variance takes on a maximum value of $\frac{3}{4}$ when $p_f = 0$, which gives the result in (2). ◇

The above bounds on the variance can now be used to compute the sample size and the estimation accuracy of CATE. Please see Appendix A for details.

4.3 CATE with One Comparison

Since we will be comparing the performance of CATE with RATE in terms of the estimation time, we now consider a version of CATE that almost mimics RATE. This happens when CATE is used with $k = 1$. When $k = 1$, CATE is almost the same as the RATE scheme with a minor modification. In the case of RATE, as soon as we get a two run, the register is set to null. This is done in order to ensure that the two runs form a renewal process. In the case of CATE with $k = 1$, the register is not set to null when there is a coincidence. The performance of CATE with $k = 1$ is almost the same as RATE. In fact, the variance of the estimator of CATE with $k = 1$ is slightly lower than RATE. In the rest of the paper, we will use CATE with $k = 1$ and RATE interchangeably. Note that setting $k = 1$ in the variance of the estimator, gives

$$\hat{\sigma}_f^2 = \frac{(1+2p_f-3p_f^2)}{4} \leq \frac{1}{3}$$

This upper bound on the variance can now be used to compute the sample size in the case of CATE with one comparison (RATE). Given α and β , the number of samples needed by RATE is

$$N^R = \frac{4Z_\alpha^2}{3\beta^2}.$$

In the next section, we characterize the memory requirements for CATE.

4.4 Memory requirements for CATE

We now focus on the memory requirement for CATE. We show that the worst case expected memory requirement is a factor of \sqrt{k} greater than RATE. The memory requirement for CATE again is more difficult to compute than RATE due to dependencies of the comparison.

Theorem 7 *Let $\xi_f(N)$ represent the indicator random variable that is set equal to one if there is at least one coincidence for flow f after N arrivals. Then*

$$\Pr \{ \xi_f(N) = 1 \} \leq 1 - (1 - p_f^2)^{Nk}$$

Proof:

If there are N arrivals to the system then there are Nk comparisons. Let $e_i(f)$ represent the event that the i th comparison does not produce a coincidence for flow f , and $\bar{e}_i(f)$ the event it does. Then,

$$\begin{aligned} & P[\xi_f(N) = 1] \\ &= 1 - P[\wedge_{i=1}^{Nk} e_i(f)] \\ &= 1 - P[e_1(f)] \times P[\wedge_{i=2}^{Nk} e_i(f) | e_1(f)] \\ &= 1 - P[e_1(f)] \times P[e_2(f) | e_1(f)] \\ &\quad \times P[\wedge_{i=3}^{Nk} e_i(f) | \wedge_{i=1}^2 e_i(f)] \\ &= 1 - P[e_1(f)] \times \prod_{i=2}^{Nk} P[e_i | \wedge_{j=1}^{i-1} e_j(f)] \end{aligned}$$

Clearly,

$$\begin{aligned} P[e_1(f)] &= 1 - P[\overline{e_1(f)}] = (1 - p_f^2), \\ \text{and } P[e_i(f)] &= 1 - P[\overline{e_i(f)}] = (1 - p_f^2) \end{aligned}$$

But $P[e_i | \wedge_{j=1}^{i-1} e_j(f)]$ may not equal to $P[e_i(f)]$ due to the correlations among comparisons. We can show that (the proof is ignored due to the space limit)

$$P[e_i(f) | \wedge_{j=1}^{i-1} e_j(f)] \geq (1 - p_f^2) \quad \forall i \in [2, Nk].$$

The proof follows immediately. \diamond

For small values of p_f , we can effectively re-write the above expression as

$$\Pr \{ \xi_f(N) = 1 \} \leq 1 - \exp(-Nkp_f^2).$$

In fact we can show that for small p_f , coincidences for flow f roughly occur as a Poisson Process with rate p_f^2 .

Now we can compute the worst case traffic pattern as in [5].

Theorem 8 *Given the specified accuracy requirement described in Section 2.3, the maximum expected memory is*

$$E[L(N)] \leq \frac{1.11Z_\alpha}{\beta}. \quad (6)$$

Proof:

Let $L(T)$ represent the number of flows that have at least one comparison two-run up to N arrivals. Then,

$$E[L(N)] = E\left[\sum_{i=1}^n \xi_f(N)\right] \leq \sum_{i=1}^n 1 - \exp(-p_i^2 Nk) \quad (7)$$

and

$$E[L(N)] \leq 0.638\sqrt{kN} \quad (8)$$

Substituting the minimum sample size determined in Equation 9 at Appendix A gives the above result. \diamond

5 Performance Evaluation

We present our performance evaluation results for CATE in this section. We intend to find out how CATE performs compared to other existing approaches including RATE, ACCEL-RATE, and smart sampling in terms of estimation time, accuracy, and memory cost. For convenience, we define *flow rate* as the ratio of the number of packets contained in each flow over total number of traffic during the measurement period; and define *sampling time* as total number of traffic arrivals. As explained in Section 2.1, these can be easily converted to the actual flow rate and sampling time when scaled with the measured link rate.

We describe the details of experimental setup and results in the following.

5.1 Experimental setup

We conduct two sets of experiments. We first use synthetic trace to compare performance of all four schemes, and then use real IP data traffic traces from NLANR² to further validate our observations.

In order to find out how effective various approaches can capture and estimate relatively small flows, we generate the synthetic data as follows: we first produce a list of one million (10^6) flows, including 5 “large” flows, 1000 “medium-sized” flows, and $10^6 - 1005$ “small” flows. Rate of each large flow is uniformly selected between 0.1 and 0.2 of the entire traffic. Rate of medium-sized flow is uniformly selected between 0.0001 and 0.0002 of the entire traffic. Small flows evenly share the remaining 10% of the traffic, each with rate roughly about 10^{-7} . During simulation, each arrival packet is assigned a flow id, which is uniformly selected among one million flows according to their rates. Note that the large flows should be easy to capture; a good estimation scheme should capture those medium-sized flows while not wasting resource (e.g., memory) on flows that are too small, i.e., below the required accuracy level.

We also use seven sets of IP trace data that have been collected between April 1, 2004 and May 5, 2004 on various OC-3 and OC-12 links, available from NLANR.

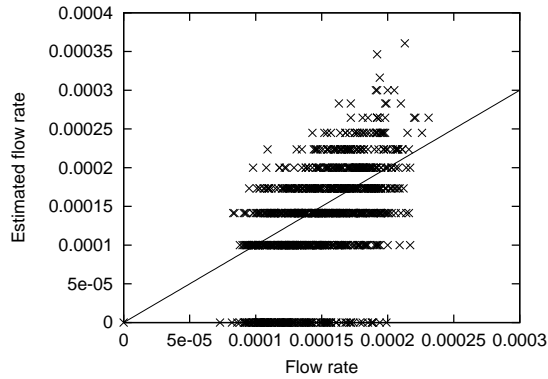


Figure 2: Accuracy of CATE for medium-sized flows, $k=100$

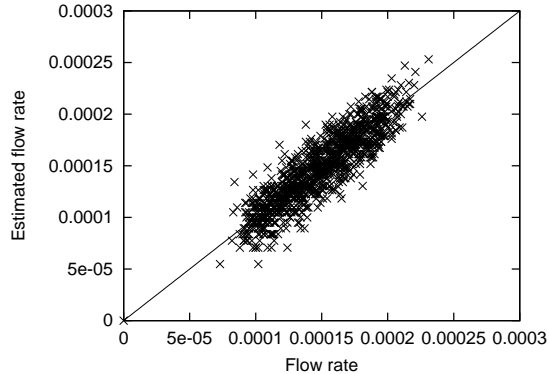


Figure 3: Accuracy of CATE for medium-sized flows, $k=1000$

Performance results are presented in the next three sections. Sections 5.2 and 5.3 use synthetic trace. Section 5.4 uses NLNR trace.

5.2 Evaluation of CATE

We evaluate CATE with k (number of comparisons for each arrival) ranging from 1 to 1000. Figures 2, and 3 show the estimated flow rate vs. actual flow rate when k is 100 and 1000. Sampling time is one million packet arrivals. Only medium-sized flows with rate $[0.0001, 0.0002]$

²<http://pma.nlanr.net>

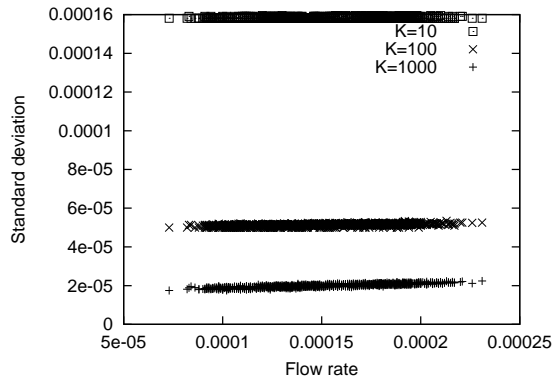


Figure 4: Standard deviation of CATE with $k=10, 100,$ and 1000

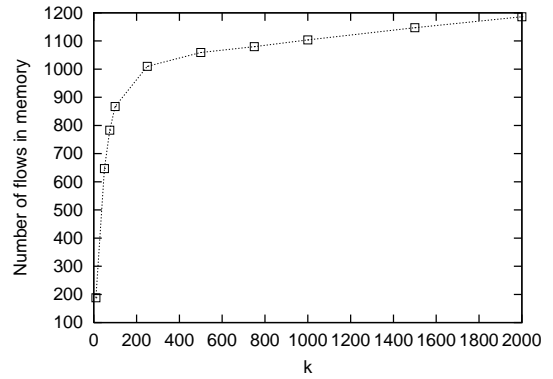


Figure 5: Memory cost under different k values

are shown here; the five large flows were estimated very accurately in all cases. The result for $k = 1$ (i.e., original RATE) and $k = 10$ are not shown since they fail to capture almost all medium-sized flows. We observe that estimation accuracy is greatly improved with increase in k . Here we have deliberately chosen a relatively short sampling time to illustrate the impact of k ; accuracy is much better when sampling time is longer.

Figure 4 shows the corresponding standard deviation for the above experiment. Note that confidence interval is proportional to standard deviation. Not surprisingly, standard deviation decreases with increase in k . Furthermore, we find that almost all flows have constant variance. This is because for k values of 10, 100, and 1000, the corresponding Δ are roughly 0.05, 0.005 or 0.0005; and hence all medium-sized flows have rates below Δ . We know from previous analysis that variance does not change much when $p_f < \Delta$.

Figure 5 shows the result for memory size. We observe that with increase in k , memory size first increases fast when $k < 1000$, and then increases slowly when $k > 1000$. This is because there are about 1000 medium to large size flows, for which CATE was able to capture very well. The smaller flows occasionally get on the count table, but number of such flows are insignificant.

5.3 Comparing CATE with RATE, ACCEL-RATE, and smart sampling

In this set of experiments, we change sampling time from 50K packets to 600M packets and compare the estimation accuracy of the four approaches for each given sampling time. We use number of significant flows (i.e., large and medium-sized flows) missed by each approach at the end of each given sampling time as an indicator for estimation accuracy.

Figure 6 shows the number of significant flows that each approach fails to capture under various sampling time. Here buffer size for smart sampling scheme is set to be 2000 flow entries. Both k in CATE and ACCEL-RATE are set to be 100.

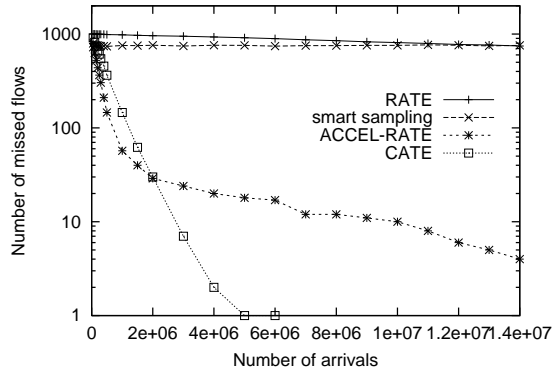


Figure 6: Accuracy comparison: number of significant flows missed

We find that all approaches can easily capture the five large flows with rate above 0.1. But their ability of capturing medium-sized flows varies significantly. The number of flows that smart sampling captures remains almost constant, determined by buffer size. Since most buffer were occupied by packets from large flows, the scheme missed about 75% of all the medium-sized flows.

CATE is the fastest: it can capture all flows after about 6 million packet arrivals. Note that it takes RATE and ACCEL-RATE 600 and 60 million packets to capture all flows, although sampling time longer than 14 million packets are not shown due to scale. This result indicates that CATE gets a speedup in sampling time over RATE by a factor of 100. ACCEL-RATE only gets a speedup of 10 over RATE because the largest flow rate is around 0.1, which implies the largest bucket size ACCEL-RATE is at least 10% of the total traffic regardless of value of k . The medium-sized flows that are hashed to such large buckets cannot get speedup more than 10 even for $k > 10$. That is why it takes ACCEL-RATE 10 times longer to capture all flows than CATE. However, existence of large buckets also implies that the other smaller buckets should have average size smaller than $1/k$; and hence the flows in such smaller buckets get more than k factor of speedup. This explains why ACCEL-RATE initially captures more flows than CATE when sampling time is small.

The difference between CATE and ACCEL-RATE can be further observed in Figures 7 and 8. They correspond to sampling time of one million packets. Figure 7 shows the estimation accuracy for the flows that are hashed to large buckets in ACCEL-RATE compared with the same flows in CATE. We observe that CATE is much more accurate than RATE in estimating such flows. Figure 8 compares the standard deviation for ACCEL-RATE and CATE. We observe that all flows in CATE have similar standard deviation. In ACCEL-RATE, although most flows have smaller standard deviation than CATE, flows that hashed to large buckets have much larger standard deviation, which limits the overall accuracy.

Figure 9 shows the change in memory size with increase in sampling time. From both Figures 6 and 9, we observe that

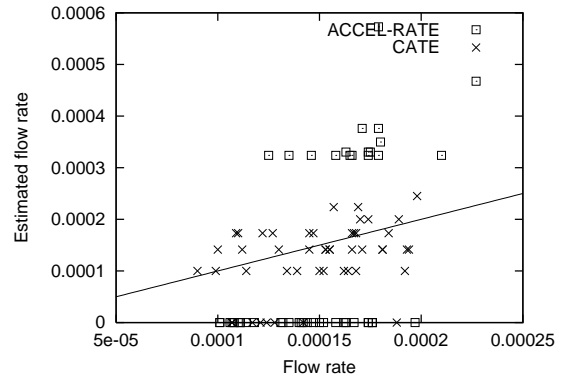


Figure 7: Accuracy of CATE and ACCEL-RATE for flows hashed to “large buckets”, $k=100$

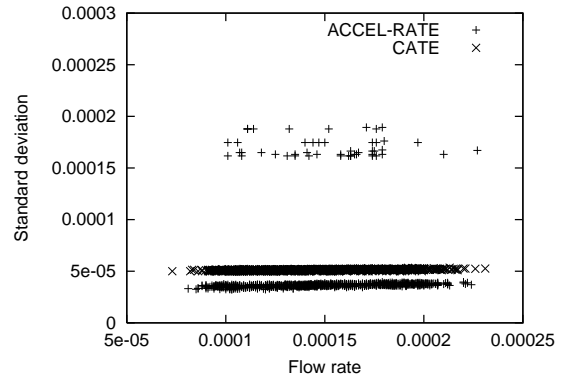


Figure 8: Standard deviation of CATE and ACCEL-RATE: $k=100$

smart sampling has the largest memory size but worst estimation accuracy. CATE is the most accurate and fastest, but uses more memory than RATE under the same sampling time. We also note that both CATE and RATE use about the same amount of memory by the time they capture all medium-sized flows (not shown in the figure), which indicates that they need similar memory size for the same estimation accuracy.

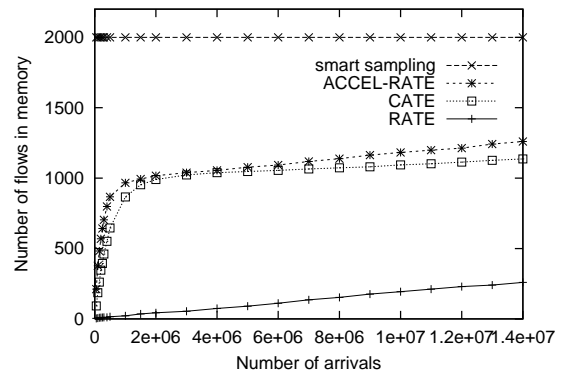


Figure 9: Memory cost comparison

5.4 Further evaluation with real IP trace

We further evaluate performance of CATE by using seven NLANR traces. Note that our analysis has assumed that the packet headers form an independent process. In practice the packet headers can be correlated, as we observed from NLANR traces. We overcome this problem by adding a buffer of size l between the Predecessor Table and new arrival in Figure 1, such that the new arrival is compared with the k arrivals that are not immediately before it, but at least l packets before. The buffer size l depends on the auto-correlation structure of the trace. Through simulation, we find that l on the order of a few hundred to a thousand usually suffice. We choose $l = 1000$ in simulation.

We show the results from one trace that was collected from an OC-3 link on April 1, 2004 from BWY. It contains 2.28 million packets. The results from other traces are similar and not shown due to space. Figures 10 and 11 show the accuracy of CATE when k is 1 (equivalent to original RATE) and 50, correspondingly. Similar to our observations from synthetic data, we find that CATE significantly improves sampling accuracy. The memory sizes are 547 and 1464 for $k = 1$ and $k = 50$, correspondingly. Hence we trade a modest increase in memory size for much improved accuracy. We also observe that both RATE and CATE tend to slightly over-estimate flow rates. This is due to the correlation within the packet arrivals in the trace. We are currently investigating better ways to accommodate the dependence in the trace so as to further improve estimation accuracy.

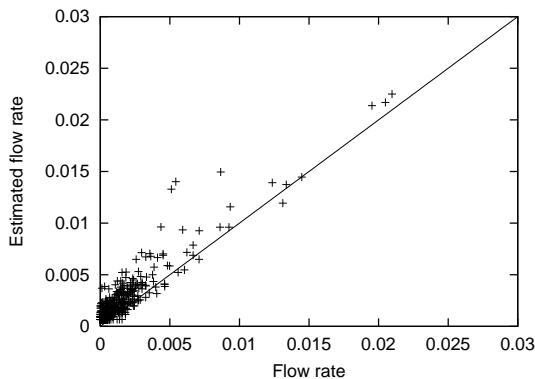


Figure 10: Estimation accuracy with $K=1$ (original RATE): NLANR trace BWY1

6 Conclusions

In this paper we have developed a traffic estimation scheme called *CATE* that uses coincidences between different arriving packets to estimate the traffic rate. We have shown through both theoretical analysis and extensive simulations that *CATE* is fast and memory-efficient. It estimates small flows quickly and accurately, and still gives a good estimation accuracy for large flows.

Dependence within packet arrivals may affect the estimation accuracy. Although the buffer-based approach can

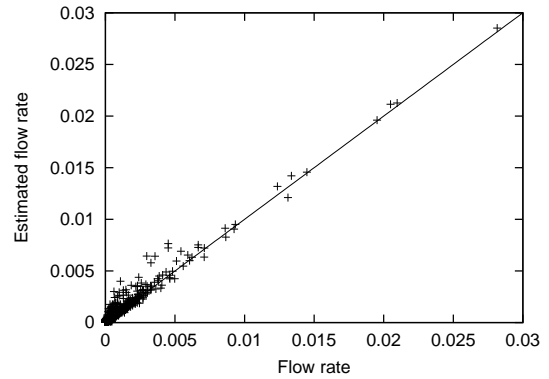


Figure 11: Estimation accuracy with $K=50$: NLANR trace BWY1

alleviate the problem, a better approach that can further minimize the impact of dependence without additional overhead may be more preferable. We attempt to address this issue as part of future work.

References

- [1] Duffield, N., Lund, C., and Thorup, M., "Flow Sampling Under Hard Resource Constraints", *Proceedings of ACM SIGMETRICS 2004*.
- [2] Estan, C., and Varghese, G., "New Directions in Traffic Measurement and Accounting", *Proceedings of ACM SIGCOMM 2002*.
- [3] Ferguson, T. S., "A Course in Large Sample Theory", *Chapman and Hall, 1996*.
- [4] Hao, F., Kodialam, M., and Lakshman, T. V., "ACCEL-RATE: a faster mechanism for memory efficient per-flow traffic estimation", *Proceedings of ACM SIGMETRICS 2004*.
- [5] Kodialam, M., Lakshman, T. V., and Mohanty, S., "Runs bAsed Traffic Estimator (RATE): A simple, Memory Efficient Scheme for Per-Flow Rate Estimation", *Proceedings of INFOCOM'2004*.
- [6] Estan, C., Savage, S., and Varghese, G., "Automatically Inferring Patterns of Resource Consumption in Network Traffic", *Proceedings of ACM SIGCOMM 2003*.
- [7] Aldous, D., *Probability Approximations via the Poisson Clumping Heuristic*, Springer-Verlag, 1987.
- [8] Duffield, N, Lund, C., and Thorup, M., "Charging from Sampled Network Usage", *SIGCOMM internet Workshop 2001*.
- [9] Duffield, N, and Grossglauser, M., "Trajectory Sampling for Direct Traffic Observation", *Proceedings of ACM SIGCOMM 2000*.

- [10] Fang, W, and Peterson, L., “Inter-AS Traffic Patterns and their Implications”, *Proceedings of IEEE GLOBECOM 1999*.
- [11] Feng, W. et al., “Stochastic Fair Blue: A Queue Management Algorithm for Enforcing Fairness”, *Proceedings of INFOCOM’2001*.
- [12] T. J. Ott, T. V. Lakshman, and L. H. Wong, “SRED: Stabilized RED”, *Proceedings of INFOCOM’99*, pp. 1346-1355, Mar. 1999.
- [13] R. Pan, B. Prabhakar, and K. Psounis, “CHOCe A Stateless Active Queue Management Scheme for Approximating Fair Bandwidth Allocation”, *Proceedings of INFOCOM’99*, pp. 2:942-951, Mar. 2000.

A Proof of Theorem 1

Before we prove Theorem 1, we need the following result on minimum sample size for CATE:

Minimum sample size when $(2k - 1)p_f \leq 1$ Let β be the desired estimation accuracy and Z_α the desired α percentile. For any flow f with $(2k - 1)p_f \leq 1$, since based on Lemma 6 its variance takes on a maximum value of $\frac{3}{4}$, the minimum sample size N in order to satisfy the accuracy requirement is given by

$$N = \frac{3Z_\alpha^2}{k\beta^2}. \quad (9)$$

Proof of Theorem 1: Let β be the desired estimation accuracy for all flows with rate $\leq \Delta$, and Z_α the desired α percentile.

For RATE, the α percentile confidence interval is given by

$$\hat{p}_f \pm Z_\alpha \sqrt{\frac{\hat{\sigma}_f^2}{N}},$$

and the minimum sample size N^R to satisfy the accuracy requirement is given by [5]

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

For CATE, the α percentile confidence interval is given by

$$\hat{p}_f \pm Z_\alpha \sqrt{\frac{\hat{\sigma}_f^2}{Nk}}.$$

When $k = \frac{1}{2}(\frac{1}{\Delta} + 1)$, $(2k - 1)p_f \leq 1$ for any flow f with $p_f \leq \Delta$. Based on Equation 9, for these flows the minimum sample size N^C in order to satisfy the accuracy requirement is given by

$$N^C = \frac{3Z_\alpha^2}{k\beta^2} = \frac{6Z_\alpha^2}{(\frac{1}{\Delta} + 1)\beta^2}.$$

Comparing N^C with N^R , we have

$$N^C = \frac{1}{0.23(\frac{1}{\Delta} + 1)}N^R.$$

Next we will show that in CATE when setting $k = \frac{1}{2}(\frac{1}{\Delta} + 1)$ and $N^C = \frac{3Z_\alpha^2}{k\beta^2}$,

$$Z_\alpha \sqrt{\frac{\hat{\sigma}_f^2}{N}} \leq \frac{\beta}{2} \sqrt{\frac{(\frac{1}{\Delta} + 1)^2}{3(\frac{2}{\Delta} + 1)}}.$$

Based on (1.) of Lemma 6, the maximum value of $\hat{\sigma}_f^2 = \frac{k^2}{4k-1}$ for any given k . Then,

$$\begin{aligned} Z_\alpha \sqrt{\frac{\hat{\sigma}_f^2}{N^C k}} &\leq Z_\alpha \sqrt{\frac{\frac{k^2}{4k-1}}{N^C k}} \\ &= \frac{\beta}{2} \sqrt{\frac{4k^2}{3(4k-1)}} \\ &= \frac{\beta}{2} \sqrt{\frac{(\frac{1}{\Delta} + 1)^2}{3(\frac{2}{\Delta} + 1)}} \end{aligned}$$

Therefore, the CATE scheme will meet the accuracy requirement for any flow f with $p_f > \Delta$ as long as $\theta \geq \sqrt{\frac{(\frac{1}{\Delta} + 1)^2}{3(\frac{2}{\Delta} + 1)}}$. \diamond