

Content Based Rate Estimation using Lazy Membership Testing

Fang Hao* Murali Kodialam* T. V. Lakshman* Vivek Vishnumurthy† Hui Zhang‡

* Bell Laboratories

101 Crawfords Corner Road

Holmdel, NJ 07733

{fangh, muralik, lakshman}@lucent.com

† Dept. of Computer Science ‡ NEC Labs America

Cornell University

Ithaca, NY 14853

vivi@cs.cornell.edu

4 Independence Way

Princeton, NJ 08540

huizhang@nec-labs.com

ABSTRACT

Fast IP flow rate estimation has many potential applications in network management, monitoring, security, and traffic engineering. Recently, low cost and memory efficient techniques to accurately estimate flow-rates in real-time have been developed. These techniques rely on flow definitions being constrained to being subsets of the fields in the packet header making flow-membership tests relatively inexpensive. In this paper, we consider a more general flow-rate estimation problem where flow membership testing is non-trivial and may involve more complex processing such as packet-payload based tests. An example is to estimate the amount of traffic that contains a given set of patterns (e.g., virus or worm signatures). We design new flow estimation techniques to reduce the number of membership tests. These techniques track pairs of arrivals that have the given property of interest and use lazy membership testing to avoid complex property testing unless absolutely necessary. The efficiency of the new schemes is evaluated by both analysis and simulation.

I. INTRODUCTION

With networks facing increasing threats from intruders, worms, viruses, and denial of service attacks, there is increasing interest in developing and deploying mechanisms to monitor and detect anomalous events. A fundamental component of a monitoring system is the deployment of methodologies for measuring traffic based on filters set by the network manager. Traditional monitoring techniques have focused on measuring traffic based on IP packet headers, where a typical objective is to measure the number of packets belong to a flow, where flow is defined to be subsets of the header fields such as the five-tuple or the destination address. Several memory and processing efficient schemes have been developed to measure traffic based on packet headers [1], [2], [5]. Over the last couple of years there has also been an increasing interest in monitoring traffic based on the packet payload. Monitoring traffic based on payload is significantly more difficult than processing traffic based on the header. This is due to the fact that payload processing typically involve some variant of string matching which is more difficult

than processing the header. Typically payload based monitoring checks for known patterns in the payload. The methodology developed in this paper is one step in monitoring traffic based on processing payloads.

The general problem that we consider is the following: Given an arrival stream to a system we want to determine the proportion of the arrival stream that has some pre-defined property. We assume that this proportion is to be determined to some desired accuracy level. An example of the predefined property is whether the current payload contains one of many strings or if the destination IP address of the packet belongs to a given set of destination IP addresses. In general we assume that it is non-trivial to check whether an incoming packet has the given property. Therefore, a natural objective is to minimize the total number of tests needed to determine the proportion of traffic having this property to the desired accuracy level.

Though the schemes developed in this paper have been motivated by payload processing applications, there are several other networking and non-networking applications for which this methodology can be applied. For example, determining the fraction of machines in an enterprise that are infected with some virus or the fraction of the population of a country that has a given blood type or disease.

A simple approach to determining the proportion is to check each arriving packet for the property and find the fraction of arrivals that have the given property. We show the surprising result that we can reduce the total number of tests in several cases by tracking pairs of arrivals that have the given property and testing for the property only when absolutely necessary. We call this Lazy Membership Testing. We consider two different testing models in this paper. In the first model, we assume that there is a (non-trivial) test for the property of interest. In the second model, we assume that there is a relatively easy test to check for some necessary condition for the property of interest and the more difficult sufficiency test is performed only if an arrival passes the necessary condition.

We use both theoretical analysis and experimental studies to evaluate the performance of both models of the lazy membership testing compared with the conventional exhaustive membership testing. Our results show that when it is likely that the

proportion of arrivals that satisfy the property of interest is relatively small, i.e., less than about 0.35 (which is indeed true in many realistic scenarios), lazy membership testing, when combined with an effective necessary condition test filter, can reduce the amount of membership testing by up to several orders of magnitude. To our knowledge, this is the first study that focuses on reducing membership testing overhead for flow measurement problems.

II. MODEL ASSUMPTIONS AND NOTATION

We consider a system processing a sequence of arrivals. A typical example is a network link carrying IP packets. We use P to determine the property of interest and p to denote the fraction of packets traversing this link that has property P . We index the arrivals with j and we use $j \in P$ to denote the event that arrival j has property P . We use $j \notin P$ to denote the event that arrival j does not have property P . We use the term arrival j belongs to (does not belong to) P to mean $j \in P$ ($j \notin P$). We want an estimate \hat{p} for p such that $\hat{p} \in \left(p - \frac{\beta}{2}, p + \frac{\beta}{2}\right)$ with probability greater than α . In other words, we want to determine an estimate for p that is within $\pm \frac{\beta}{2}$ of p with probability greater than α . We assume that the probability that any given arrival has property P is p independent of other arrivals. We consider two different testing models in this paper. In the first model, we assume that there is a single test for property P . In the second model, we assume that there is a necessary condition for property P that is easier to check than the sufficient condition. We derive the number of tests (in the second model the number of necessary tests as well as the number of sufficiency tests) to determine the fraction p . Throughout this paper, we use $\mathcal{N}[a, b]$ to represent a normal distribution with mean a and variance b .

For ease of reference, we list a few terms and notations that we have used throughout this paper in Table I. Detailed explanations are in the next few sections.

Term	Meaning	Described in Section
EMT	Exhaustive Membership Testing	III
LMT	Lazy Membership Testing	V
N_E	Sample size with Exhaustive Testing	IV-A
N_L	Sample size with Lazy Testing	IV-B
T_E	#Membership Tests with EMT	IV-A
T_L	#Membership Tests with LMT	V
FEMT	Filtered Exhaustive Membership Testing	VI-A
FLMT	Filtered Lazy Membership Testing	VI-B
T_{FE}^1	# P_1 Membership Tests with FEMT	VI-A
T_{FE}^2	# P_2 Membership Tests with FEMT	VI-A
T_{FL}^1	# P_1 Membership Tests with FLMT	VI-B
T_{FL}^2	# P_2 Membership Tests with FLMT	VI-B

TABLE I
GLOSSARY OF TERMS AND NOTATIONS

III. OVERALL APPROACH

A straightforward approach to estimating p is to check if each arrival has property P and use the fraction of arrivals that belongs to P as an estimate for p . We call this approach *Exhaustive Membership Testing (EMT)*. If there is no a priori knowledge of p then immediate membership testing uses the minimum number of samples [12]. The number of samples can be derived using elementary statistics. Since each arrival is tested, the total number of membership tests in the case of exhaustive membership testing equals the sample size. We show the surprising result that the number of tests can be reduced for certain values of p using a different approach which does not involve testing all the arrivals. This new approach called *Lazy Membership Testing (LMT)*, is based on the idea of counting coincidences.

Definition 1: Arrival j sees a coincidence if both the arrival j and its predecessor belong to P , i.e., $j \in P$ and $j - 1 \in P$.

The idea of using coincidences for estimating rates was proposed in [5]. The feature of coincidences that they exploit is that small flows are less likely to have coincidences and this leads to a decrease in the memory needed to perform the estimation. In this paper, we are only concerned about the amount of testing required. (In fact we can assume that there is only one flow of interest, namely the set of objects that have the desired property). Here, we exploit a different aspect of the coincidence process. If we are counting the number of coincidences we do not have to perform a membership test for each arrival. For example, if we know that $j - 1 \notin P$ and $j + 1 \notin P$, then arrival j cannot be part of any coincidence. Therefore, we do not need to check if $j \in P$. The key idea is to defer the testing as much as possible in order to avoid testing the arrival if it is not part of a coincidence (and hence the name Lazy Membership Testing). As we show in Section IV, the number of samples needed to get the desired accuracy using coincidence counting is larger than that of EMT. However, as we show in Sections V and VI, LMT in fact results in decreased membership testing in spite of the increased sample size for a range of commonly occurring values of p . In the next section, we derive the sample size needed for both EMT and LMT. The total number of tests for EMT equals the sample size. In Section V, we derive the probability that an arrival is tested for membership in LMT. The product of the probability of testing and the sample size of LMT gives the expected number of membership tests using LMT.

IV. DETERMINING SAMPLE SIZES

In this section we derive the number of samples needed to determine p to the desired level of accuracy for both EMT and LMT. First we quote the following result from elementary statistics [12].

Lemma 1: Let X_1, X_2, \dots be independent, normally distributed random variables with common mean θ and variance

$\sigma^2(\theta)$. The number of samples needed to estimate θ with an error of $\pm \frac{\beta}{2}$ with probability greater than α is given by

$$N = \frac{4Z_\alpha^2 u}{\beta^2}$$

where Z_α is the α percentile for the unit normal distribution, and u is such that $\sigma^2(\theta) \leq u \quad \forall \theta$.

A. Sample Size for Exhaustive Membership Testing

Assume that we are given a sequence of arrivals $j = 1, 2, \dots$. As stated earlier $Pr[j \in P] = p$. In the case of EMT, we check if each arrival belongs to P and we count the number of arrivals that have property P . We denote N_E to denote the number of samples needed by EMT and S to denote the number of arrivals that belonged to P . The estimator for EMT is

$$\frac{S}{N_E}.$$

The next result gives the number of samples needed by EMT.

Theorem 2: Let N_E be the number of samples needed for Exhaustive Membership Testing. Then

$$N_E = \frac{Z_\alpha^2}{\beta^2}.$$

Proof: We first define an indicator random variable

$$\omega_j = \begin{cases} 1 & \text{if } j \in P \\ 0 & \text{Otherwise} \end{cases}$$

An estimator for p in the case of EMT is

$$\frac{S}{N} = \frac{\sum_{j=1}^N \omega_j}{N},$$

where N is the sample size (to be determined). From the independent arrivals assumptions, ω_j is a Bernoulli random variable, so we have

$$\begin{aligned} E[\omega_j] &= p \\ Var[\omega_j] &= p(1-p). \end{aligned}$$

Therefore the maximum variance is attained when $p = 0.5$ and the maximum variance value is 0.25. Setting $u = 0.25$ in Lemma 1, we get

$$N_E = \frac{Z_\alpha^2}{\beta^2}.$$

Since each arrival is tested, the number of tests performed by EMT is

$$T_E = \frac{Z_\alpha^2}{\beta^2}.$$

We now derive the sample size for Lazy Membership Testing.

B. Sample Size of Lazy Membership Testing

In the case of LMT, we count the number of coincidences C , that occur in the traffic stream after N arrivals. The exact method of counting these coincidences is given in the next section. We use an indicator random variable

$$\xi_j = \begin{cases} 1 & \text{if } j \in P \text{ and } j-1 \in P \\ 0 & \text{Otherwise} \end{cases}$$

to denote a coincidence at arrival i .

We now compute the mean and variance of the coincidence process in the next lemma.

Lemma 3: Let

$$C = \sum_{j=1}^N \xi_j$$

represent the number of coincidences after N arrivals. Then,

$$E[C] = Np^2 \quad (1)$$

$$Var[C] = Np^2(1 + 2p - 3p^2) \quad (2)$$

Proof: Due to the independence of j and $j-1$,

$$Pr[\xi_j = 1] = Pr[j \in P \text{ and } j-1 \in P] = p^2.$$

In addition, we can show that

$$E[\xi_i \xi_j] = \begin{cases} p^2 & \text{if } i = j \\ p^3 & \text{if } |i-j| = 1 \\ p^4 & \text{if } |i-j| > 1 \end{cases}$$

The variance of C is computed as follows:

$$\begin{aligned} Var[C] &= E[C^2] - (E[C])^2 \\ &= E[(\sum_{j=1}^N \xi_j)^2] - (E[\sum_{j=1}^N \xi_j])^2 \\ &= E[\sum_{j=1}^N \xi_j^2] + E[\sum_{j=1}^N \sum_{i=1, i \neq j}^N \xi_i \xi_j] - N^2 p^4 \\ &= N[(N-3)p^4 + 2p^3 + p^2] - N^2 p^4 \\ &= Np^2(1 + 2p - 3p^2) \end{aligned}$$

From Equation 1, note that an estimator for p is

$$\hat{p} = \sqrt{\frac{C}{N}}.$$

We now derive the sample size for this estimator.

Theorem 4: Let N_L be the number of samples needed for Lazy Membership Testing to achieve the desired accuracy. Then

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

Proof: From the normal distribution for weakly dependent random variables [13], it is easy to show that

$$\sqrt{N} \left[\frac{S}{N} - p^2 \right] \sim \mathcal{N} [0, p^2(1 + 2p - 3p^2)]$$

where $\mathcal{N}[a, b]$ represents a normal distribution with mean a and variance b . In order to translate this into the normal distribution for the estimator we use the following standard result in statistics [12] which states that if X_n is a sequence of statistics such that

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

and f be a differentiable function of one variable, then

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

Applying this result for our case, we get

$$\sqrt{T} \left[\sqrt{\frac{S}{N}} - p \right] \sim \mathcal{N} \left[0, \frac{1 + 2p - 3p^2}{4} \right].$$

Since

$$\frac{1 + 2p - 3p^2}{4} \leq \frac{1}{3} \quad 0 \leq p \leq 1$$

we set $u = \frac{1}{3}$ in Lemma 1 to get

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

Note that the sample size for LMT is greater than the sample size for EMT. In the next section, we estimate the probability that a given arrival is tested in LMT.

V. ESTIMATING THE PROBABILITY OF TESTING WITH LAZY MEMBERSHIP TESTING

In order to estimate the probability a given arrival is tested in LMT, we first have to give a precise description of the implementation of LMT. Figure 1 gives the pseudo-code for the algorithm used in LMT. We first define the state of the system Z_j in terms of the state of the current packet j after completing the processing for j at the current step j :

$$Z_j = \begin{cases} 0 & \text{if } j \text{ is tested and } j \notin P. \\ 1 & \text{if } j \text{ is tested and } j \in P. \\ 2 & \text{if } j \text{ is not tested so far.} \end{cases}$$

The algorithm, in essence, works in the following manner:

- If $Z_{j-1} = 0$ then do not test arrival j immediately.
- If $Z_{j-1} = 1$ then test arrival j .

```

Coincidence Counter  $C \leftarrow 0$ .

For each arrival  $i$ 
switch  $Z_{i-1}$ 
  case 0: //Defer testing  $i$ 
     $Z_i = 2$ ;
  case 1: //Test  $i$  now
    If  $(i \in P)$  then
       $C \leftarrow C + 1$ 
       $Z_i = 1$ 
    Else  $Z_i = 0$ ;
  case 2: //Test  $i$  now
    If  $(i \in P)$  then //Test  $i - 1$  now.
      If  $(i - 1 \in P)$  then  $C \leftarrow C + 1$ 
       $Z_i = 1$ 
    Else  $Z_i = 0$ ;

```

Fig. 1. Description of Lazy Membership Testing

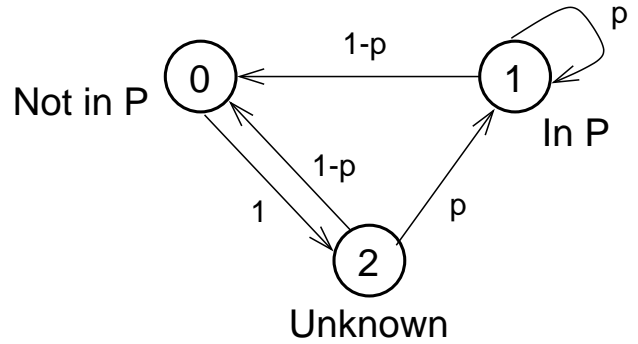


Fig. 2. System states in the coincidence counter

- If $Z_{j-1} = 2$ then test arrival j and if $Z_j = 1$, go back and test arrival $j - 1$.

Note that if $Z_{j-1} = 0$ and $Z_{j+1} = 0$ then arrival j will not be tested. This is the reason we defer the testing of the arrival until we discover the possibility that the arrival could be part of some coincidence.

Since the operation to be performed on each packet only depends on the state of the previous packet, the Z_i 's form a Markovian process. In order to compute the expected number of membership tests, we use a 3-state Markov chain (see Figure 2) to represent the states of the system, where the states correspond to $Z_j = 0, 1$, or 2 , respectively.

The transition probability matrix of this Markov chain is

$$M = \begin{bmatrix} 0 & 0 & 1 \\ 1-p & p & 0 \\ 1-p & p & 0 \end{bmatrix}$$

Let (π_0, π_1, π_2) denote the steady state probability of this Markov Chain where π_k is the steady state probability that the system is in state k for $k = 0, 1, 2$ respectively. Then

$$[\pi_0 \ \pi_1 \ \pi_2] M = [\pi_0 \ \pi_1 \ \pi_2]$$

Using this and the fact that these probabilities add to 1, we get

$$\pi_0 = \pi_2 = \frac{1-p}{2-p}, \text{ and } \pi_1 = \frac{p}{2-p}$$

If the system is in states 0 or 1 of Figure 2, the membership test would have already been performed for the current packet. The only way the membership test is never performed for a packet is if the system is in state 2 after the packet has arrived, and the next packet fails the membership test. The probability of this happening is $\pi_2(1-p) = \frac{(1-p)^2}{2-p}$. So the probability that a packet is tested for membership is $1 - \frac{(1-p)^2}{2-p} = \frac{1+p-p^2}{2-p}$.

Theorem 5: Let T_L denote the total number of packets tested with Lazy Membership Testing. Then

$$E[T_L] = \frac{4Z_\alpha^2}{3\beta^2} \cdot \frac{1+p-p^2}{2-p}.$$

Proof: The total expected number of tests is the product of the sample size and the probability that a given arrival is tested. From Lemma 1 for the sample size and the result above on the probability of testing a given arrival, the result follows. ■

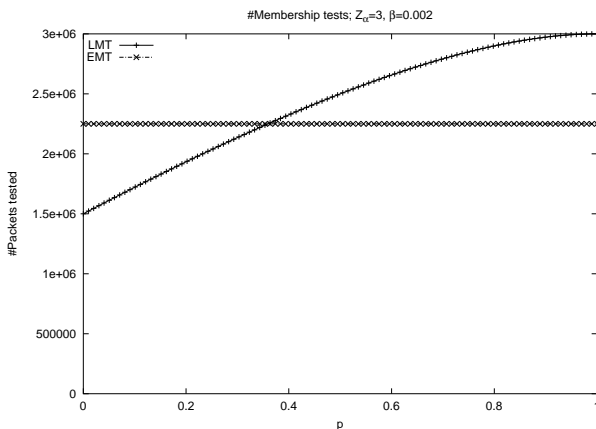


Fig. 3. Number of packets tested on Lazy Membership Testing and Exhaustive Membership Testing, for $Z_\alpha = 3$ and $\beta = 0.002$

Figure 3 plots the expected number of membership tests for both Lazy Membership Testing and Exhaustive Membership Testing, for $Z_\alpha = 3$ and $\beta = 0.002$. LMT achieves an improvement over EMT when p is less than about 0.36.

So far, we have assumed that we have a single test for checking for the property. In the next section, we explore the case where we have a (faster) test to check for some necessary condition for P .

VI. MEMBERSHIP TESTING WITH APPROXIMATE TESTS

So far we assumed that the only test we have is to determine if an arrival has property P . In this section, we see if we can use Lazy Membership Testing in cases where there is a relatively inexpensive test to check for some necessary condition for property P . For example, if the objective is to see if the packet's destination address belongs to a designated set of IP addresses, it is possible to use a hashing mechanism (like a Bloom Filter) to check if the destination address of the current packet is in the set of destination addresses. If this test comes out negative then we know that the current destination address is not in the designated set of addresses. If the test is positive, there is the possibility of a false positive and therefore a more detailed test has to be done to determine if the current destination address is on the list. In general, if we have a test for the necessary condition for property P , and if the current packet fails the test then we know that the current packet does not belong to P . If it passes the test then we have to perform a more expensive sufficiency test to determine if the current packet belongs to P . The question is to determine how much Lazy Membership Testing can exploit this test for the necessary condition to reduce the number of sufficiency tests. In order to clearly differentiate between the necessary test and the sufficiency test, we denote by P_1 the (sufficient) property for which we test incoming packets, and by P_2 the necessary property. Let p_1 denote the probability that a packet has property P_1 and let p_2 denote the probability that the packet has property P_2 . Note that $p_2 \geq p_1$ since P_2 is a necessary condition for P_1 and hence $P_1 \subseteq P_2$. We use the prefix "Filtered" to distinguish the tests in this section from the tests in the previous sections.

A. Filtered Exhaustive Membership Testing (FEMT)

In Filtered Exhaustive Membership Testing, we can use P_2 membership tests to decrease the number of P_1 membership tests performed. Each incoming packet is first tested for P_2 ; if it satisfies P_2 , it is then tested for P_1 . So the probability that a sampled packet is tested for P_1 is p_2 . Using T_{FE}^1 to denote the total number of P_1 membership tests for the Simple Counter, we see that the expected value of T_{FE}^1 is given by p_2 times the sample-size, i.e.,

$$E[T_{FE}^1] = p_2 \cdot \frac{Z_\alpha^2}{\beta^2}.$$

Also, since a P_2 membership test is performed on each packet belonging to the sample, the number of P_2 membership tests T_{FE}^2 with FEMT is just the sample size, i.e.,

$$E[T_{FE}^2] = \frac{Z_\alpha^2}{\beta^2}.$$

```

Coincidence Counter  $C \leftarrow 0$ 
For each arrival  $i$ 
switch  $Z_{i-1}$ 
  case 0:
     $Z_i = 3$ ;
  case 1:
    If ( $i \in P_2$ ) then
      If ( $i \in P_1$ ) then
         $C \leftarrow C + 1$ 
         $Z_i = 1$ 
      Else
         $Z_i = 0$ ;
    Else
       $Z_i = 0$ ;
  case 2:
    If ( $i \in P_2$ ) then
      If ( $i \in P_1$ ) then
         $Z_i = 1$ 
        If ( $i - 1 \in P_1$ ) then  $C \leftarrow C + 1$ 
      Else
         $Z_i = 0$ 
    Else
       $Z_i = 0$ 
  case 3:
    If ( $i \in P_2$ ) then
      If ( $i - 1 \in P_2$ ) then
        If ( $i \in P_1$ ) then
           $Z_i = 1$ ;
          If ( $i - 1 \in P_1$ ) then  $C \leftarrow C + 1$ 
        Else
           $Z_i = 0$ ;
      Else
         $Z_i = 2$ ;
    Else
       $Z_i = 0$ ;

```

Fig. 4. Filtered Lazy Membership Testing. Here a P_l membership test ($l \in \{0, 1\}$) is performed when control reaches the “If ($i \in P_l$) ...” statement.

B. Filtered Lazy Membership Testing (FLMT)

As we did in the case of FEMT, we always test for P_2 before testing for P_1 in Filtered Lazy Membership Testing. And, as was the case with LMT, we do not test a packet for P_1 (P_2) membership if we know that the packets immediately following it and preceding it both do not satisfy P_1 (P_2).

We again define the state of the system as the state of the current packet after processing the packet at the current step, and before the next packet arrives:

$$Z_j = \begin{cases} 0 & \text{if } j \text{ is tested (for } P_1 \text{ or } P_2) \text{ and } j \notin P_1. \\ 1 & \text{if } j \text{ is tested for } P_1 \text{ and } j \in P_1. \\ 2 & \text{if } j \text{ is tested for } P_2 \text{ and } j \in P_2. \\ 3 & \text{if } j \text{ is not tested so far.} \end{cases}$$

We describe the processing for each arrival in FLMT in Figure 4. Basically, we perform the P_1 test for a packet only if it satisfies the P_2 test and either the next or previous packet satisfies the P_1 test. Similarly, we perform the P_2 test only when the next or previous packet satisfies the P_2 test.

We again use a Markov chain to represent the states of the system. To compute the number of P_1 membership tests though, we need to differentiate between the cases where a packet $i \notin P_1$ is discarded after a P_1 test and the case where the packet is discarded after a P_2 test. Accordingly, we replace state 0 by two new states 0_1 and 0_2 , as follows:

$$Z_j = \begin{cases} 0_1 & \text{if } j \text{ is tested for } P_1 \text{ and } j \notin P_1. \\ 0_2 & \text{if } j \text{ is tested for } P_2 \text{ and } j \notin P_2. \end{cases}$$

Given this, we compute the transition probability matrix M to be

$$M = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 \\ p_2 - p_1 & 1 - p_2 & p_1 & 0 & 0 \\ p_2 - p_1 & 1 - p_2 & p_1 & 0 & 0 \\ p_2(p_2 - p_1) & 1 - p_2 & p_1 p_2 & p_2(1 - p_2) & 0 \end{bmatrix}$$

where the first two rows correspond to the outgoing probabilities from states 0_1 and 0_2 respectively, and rows 3,4,5 correspond to states 1,2,3 respectively.

Noting that rows 1 and 2 (corresponding to states 0_1 and 0_2) and rows 3 and 4 (corresponding to states 1 and 2) are identical, we can compress them to single rows to get a 3-by-3 matrix M' :

$$M' = \begin{bmatrix} 0 & 0 & 1 \\ 1 - p_2 + p_2^2 - p_1 p_2 & p_1 p_2 + p_2 - p_2^2 & 0 \\ 1 - p_2 + p_2^2 - p_1 p_2 & p_1 p_2 + p_2 - p_2^2 & 0 \end{bmatrix}$$

This lets us write

$$[\pi_0 \ \pi_{12} \ \pi_3] M = [\pi_0 \ \pi_{12} \ \pi_3]$$

where π_0 is the probability that the system is in states 0_1 or 0_2 , and π_{12} is the probability that the system is in states 1 or 2. And again, we have the fact that the probabilities add to 1. Solving for the unknowns, we get

$$\begin{aligned} \pi_0 = \pi_3 &= \frac{1 - p_1}{(2 - p_2)(1 + p_2 - p_1)}, \text{ and} \\ \pi_{12} &= \frac{p_2(1 - p_2 + p_1)}{(2 - p_2)(1 + p_2 - p_1)} \end{aligned}$$

Looking at M again, we see that

$$\begin{aligned}\pi_{0_1} &= \pi_{12}(p_2 - p_1) + \pi_3 p_2(p_2 - p_1), \\ \pi_{0_2} &= (\pi_{12} + \pi_3)(1 - p_2), \\ \pi_1 &= \pi_{12} p_1 + \pi_3 p_1 p_2, \text{ and} \\ \pi_2 &= \pi_3 p_2(1 - p_2)\end{aligned}$$

To compute the probability that a P_1 test is performed, observe that if the system is in states 0_1 or 1 , we will already have performed a P_1 test for the packet, and if the system is in states 2 or 3 , we might conditionally perform a P_1 test on this packet in the next step. So the probability L_{FL}^1 that a P_1 tests is performed is

$$\begin{aligned}L_{EL}^1 &= \pi_{0_1} + \pi_1 + \pi_3 \cdot p_1 p_2 + \pi_2 \cdot p_1 \\ &= \frac{p_2(p_1 + p_2 - p_1^2)}{1 + p_2 - p_1}\end{aligned}$$

Similarly, for P_2 tests, we see that the only case where this test is not performed for a packet is when the system is in state 3 after the packet has arrived, and the next packet happens to not belong to P_2 . Denoting by L_{FL}^2 the probability that a P_2 test is performed, we get

$$\begin{aligned}1 - L_{FL}^2 &= \pi_3 \cdot (1 - p_2) \\ L_{FL}^2 &= \frac{1 + 2p_2 - p_1 - p_2^2}{(2 - p_2)(1 + p_2 - p_1)}\end{aligned}$$

Using the expressions for L_{EL}^1 and L_{EL}^2 and the sample size derived in Section IV, we can show the following result.

Theorem 6: Let T_{FL}^1 denote the number of P_1 membership tests performed by FLMT and T_{FL}^2 denote the number of P_2 membership tests performed by FLMT then

$$E[T_{FL}^1] = \frac{4Z_\alpha^2}{3\beta^2} \cdot \frac{p_2(p_1 + p_2 - p_1^2)}{1 + p_2 - p_1}$$

and

$$E[T_{FL}^2] = \frac{4Z_\alpha^2}{3\beta^2} \cdot \frac{1 + 2p_2 - p_1 - p_2^2}{(2 - p_2)(1 + p_2 - p_1)}$$

The regions where Filtered Lazy Membership Testing achieves an improvement over Filtered Exhaustive Membership Testing is shown in Figure 5.

Figures 6 and 7 compare the expected number of membership tests over a range of values of p_1 and p_2 . The results show that Lazy testing leads to a significant reduction in the number of P_1 membership tests when the proportion of arrivals that satisfy the property of interest is not too large.

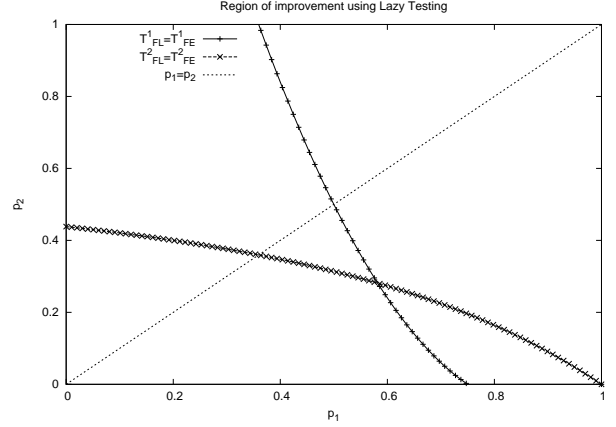


Fig. 5. Region where Lazy Testing achieves improvement over Exhaustive Testing: Only the region above the $p_1 = p_2$ curve is valid, and Lazy testing achieves an improvement in the number of P_1 tests when p_2 is below the $T_{FL}^1 = T_{FE}^1$ curve, and an improvement in the number of P_2 tests when p_2 is below the $T_{FL}^2 = T_{FE}^2$ curve.

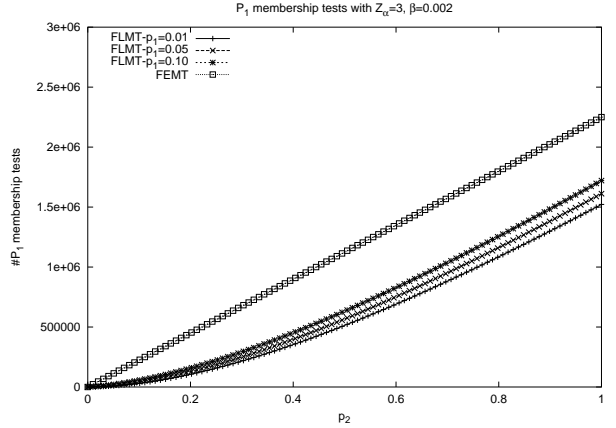


Fig. 6. Number of P_1 membership over a range of values of p_1 and p_2 . $Z_\alpha = 3$ and $\beta = 0.002$

VII. EXPERIMENTAL RESULTS

In this section, we present our experimental results from two case studies: destination address set estimation and pattern estimation. We intend to verify the estimation accuracy and savings on membership tests of our approach. Both experiments can be viewed as generalized flow estimation, where flows are defined based on properties that require non-trivial membership tests. For simplicity, flow rate is defined as proportion of the packets that belong to the flow. The actual flow rate in terms of packets per second can be derived based on link packet rate.

A. Experiment I: destination address set estimation

In the first case study, we try to estimate the amount of traffic that goes to a given set of IP destination addresses. Such techniques may have applications in areas of security or traffic monitoring. For example, one can monitor the amount of traffic

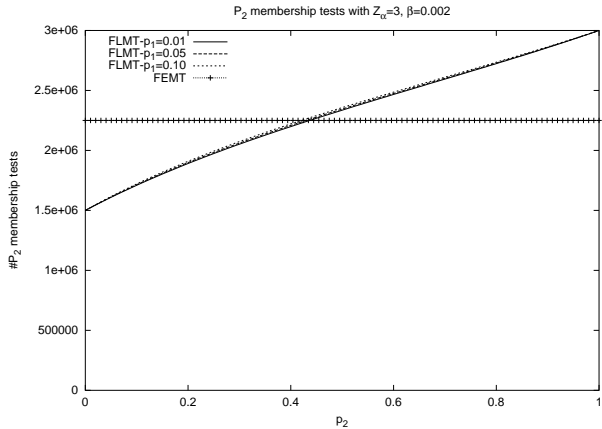


Fig. 7. Number of P_2 membership over a range of values of p_1 and p_2 . $Z_\alpha = 3$ and $\beta = 0.002$

that goes to a set of distributed database servers or server farms that contain critical information. Or one may simply want to estimate the total amount of traffic reaching a particular set of web servers that contain similar content (e.g., major news sites or music content sites) as to gain more insight into the network traffic composition or user behavior.

A naive exhaustive approach is to compare the destination address of each arriving packet with each element in the destination set, and count the number of matches. The complexity for processing each packet is $O(m)$, where m is the set size. When we have a few hundred or thousand destinations, the overhead may be too large for real-time measurement.

The naive approach can be improved by using a Bloom filter for the first stage necessary condition test. All destination addresses in the set are hashed to the same M bit map by using h hash functions. Each arriving packet is also hashed using the same h hash functions. If all h hashed positions in the bit map are 1, a *positive* is generated, and the packet moves on to the second stage (sufficient condition) test, in which its destination address is matched with the addresses in the set.

We can further improve the above approach by applying lazy membership testing, as described in previous sections. In this study, we compare all the four approaches: exhaustive membership testing (EMT), filtered exhaustive membership testing (FEMT), lazy membership testing (LMT), and filtered lazy membership testing (FLMT).

Experiment configuration

We use three sets of IP trace data that are available from NLANR, as listed in Table II. These traces that are collected between April 25, 2004 and May 5, 2004 from various OC-3 and OC-12 links. Each trace contains a sequence of IP headers along with their arrival time. Prior to running simulations for each trace, we scan the trace and randomly choose seven sets of destination addresses; each set contains 1000 destination addresses. We run the simulation for each set separately.

Trace name	Date	Link type	Length	# packets
COS	4/25/04	OC-3	90 sec.	2.11 M
MRA1	4/25/04	OC-12	90 sec.	6.28 M
MRA2	5/5/04	OC-12	90 sec.	6.23 M

TABLE II
NLANR IP TRACE DATA

Confidence interval β and error probability α for estimation are set to be 0.002 and 99.75%. This corresponds to 2.25 million samples under exhaustive testing and 3 million samples under lazy testing¹. Across this experiment, we use one hash function for the Bloom filter. Bloom filter size ranges from 1000 to 1 M bits.

Note that our analysis assumes that packet arrivals are independent, which may not generally hold for the real traffic trace. In order to cope with the potential correlation among the traffic, we use a buffer to hold the destination address of the last 1000 arrivals. When we count coincidences, we compare the current arrival with the packet that arrived 1000 packets before, instead of the immediately preceding one. The estimation accuracy and overhead are shown next.

Estimation accuracy

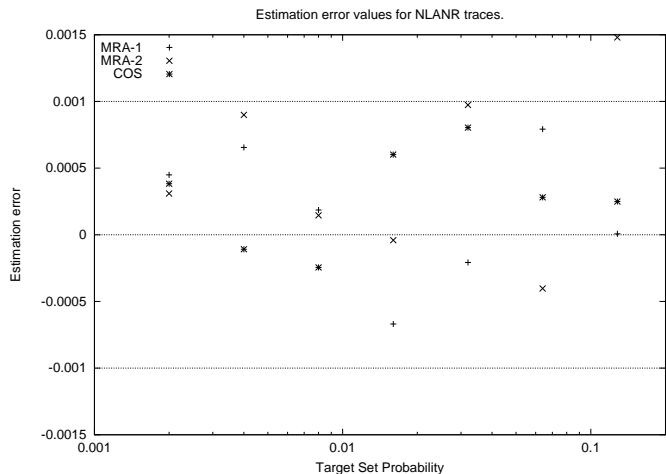


Fig. 8. Destination set estimation: estimation error

Figure 8 shows the estimation error for the three traces. Here we calculate the estimation error as the flow rate estimate given by lazy testing minus the flow rate estimate given by exhaustive testing. Note that the filter does not affect the estimation accuracy. The confidence intervals ($\pm \frac{\beta}{2}$) are shown in the figure; we observe that the accuracy of the estimation is within the required range.

Measurement overhead

¹Trace COS only has 2.11 million packets, and hence simulation for COS only has 2.11 million samples

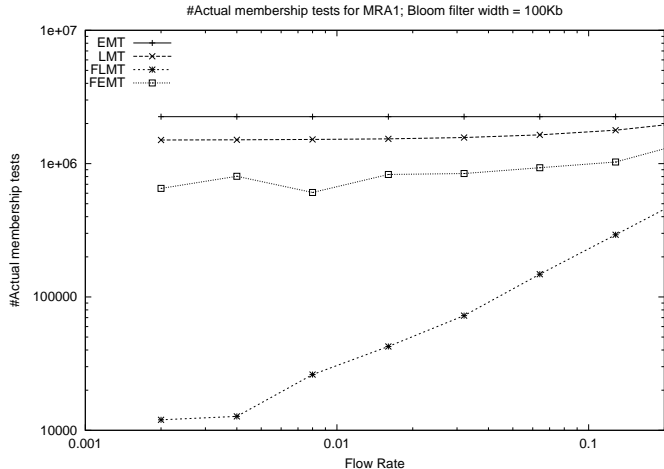


Fig. 9. Destination set estimation: sufficient condition testing (actual set testing) overhead for MRA1.

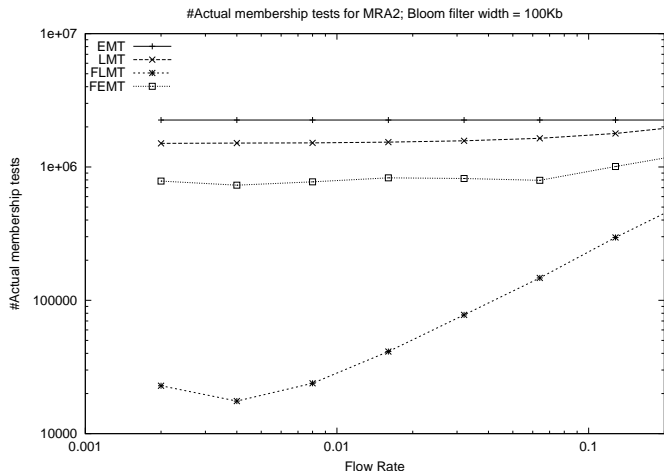


Fig. 10. Destination set estimation: sufficient condition testing (actual set testing overhead) for MRA2.

Figures 9 and 10 show the number of actual membership tests for MRA1 and MRA2. In both figures, we compare all the four approaches: EMT, FEMT, LMT, and ELMT. We find that across a wide range of relatively small flow rates, lazy testing has much less membership testing overhead than exhaustive testing². Filtered lazy membership testing (FLMT) has the best performance in this range: it reduces membership testing overhead by up to two orders of magnitude compared to exhaustive membership testing (EMT). The advantage of lazy testing is the most evident when flow rate is small, consistent with our analysis.

Impact of Bloom filter size on measurement overhead

Bloom filter size affects the false positive rate of the filter:

²On a backbone link, the chance for an individual flow to exceed 20% of the total link traffic is extremely small. Hence we believe the flow rate setting in our experiments is fairly representative of the typical backbone traffic.

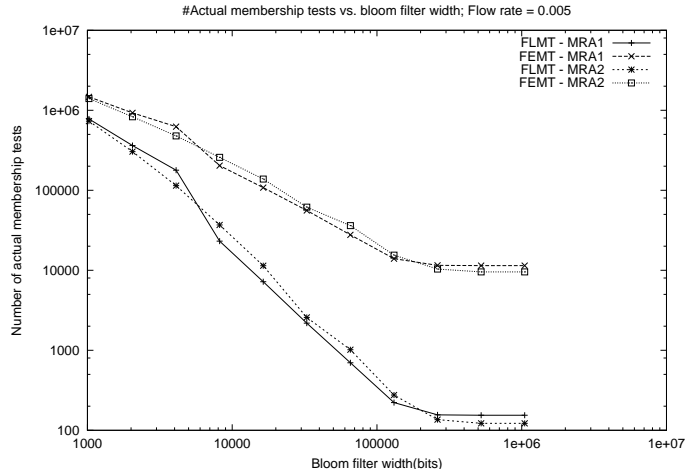


Fig. 11. Destination set estimation: impact of Bloom filter size on sufficient condition tests

larger size leads to fewer false positives and hence more efficient filtering. Figure 11 shows the change in number of set membership testing with change in Bloom filter size for traces MRA1 and MRA2. Here the flow rate is 0.005. We observe that lazy testing in this experiment always incurs fewer number of membership testing compared to exhaustive testing. Overhead of both methods decreases with increase in Bloom filter size, until Bloom filter size reaches around 100 K to 200 K. Furthermore, the advantage of lazy testing is more evident when Bloom filter size is larger. Hence one can trade off estimation overhead versus memory cost by adjusting the Bloom filter size.

B. Experiment II: pattern estimation

In the second case study, we address the following issue: given a set of existing patterns (bit strings), and a packet stream, estimate the number (or proportion) of packets that contain any of the patterns in the set. This can be viewed as a flow estimation problem where a packet belongs to a flow if its payload contains *any* of the patterns in the given set. In practice, such patterns could be virus or worm signatures, keywords, or sections of music or video files. This allows network operators to gain more insight into the content of packet payload.

A naive way to estimate the pattern rate is to match all patterns with all packets in the stream. However, since pattern matching is a fairly complex operation, we develop an alternate approach that aims to minimize the number of times pattern matching is attempted.

Our approach is to construct a first stage filter (the necessary condition test) based on combination of Rabin's fingerprint and Bloom filter.

Constructing the first stage filter

Let k be the minimum length of all patterns in the given set. For each pattern $P = [c_1 c_2 \dots c_l]$ where $l \geq k$, we compute

hash functions for the first k bytes in P . Assume that we have a prime q . The hash function is computed as

$$H = c_1 q^{k-1} + c_2 q^{k-2} + \dots + c_{k-1} q + c_k \pmod{M}$$

where M is the number of bits in the Bloom Filter. For each pattern, we apply h hash functions by using h different primes. The Bloom filter is derived by hashing all patterns to the same M bit space.

To test an arriving packet using the Bloom filter, the first k bytes of the payload are hashed using the same h hash functions that are used in the previous step. The test result is *positive* if all h corresponding bits in the Bloom filter are 1; the result is negative otherwise. If the result is negative, we then shift the payload by one byte, and test the second k byte string (i.e., the k byte string that starts from the second byte of the payload). Note that Rabin's fingerprint makes the computation for the second k length string very simple: $H_2 = qH_1 - c_1 q^k + c_{k+1} \pmod{M}$, where H_1 is the hash of first k byte string, and H_2 is the hash of the second k byte string. We continue this process until we either get a positive, or reach the end of the payload. Note that:

- The string matching algorithm for a single pattern has the worst-case complexity of $O(l \cdot L)$, where l and L are length of the pattern and packet payload, respectively.
- The complexity of the Bloom filter test for each packet is $O(h \cdot L)$, typically much faster than string matching. Nevertheless, the Bloom filter test complexity is also non-trivial, and should be minimized if possible.
- The filter constructed above does not have false negatives, but may have false positives. Several factors may contribute to false positives, including: (1) the hash functions (Rabin's fingerprint); (2) truncating patterns to k bytes when computing the hash; and (3) finite Bloom filter size M .

In this experiment, we compare the lazy testing approach with the exhaustive testing approach, both combined with the first stage filter that we described above.

Experiment configuration

In the experiment, we use 200 randomly generated patterns with length uniformly distributed between 20 and 100 bytes. Background traffic parameters such as packet size distributions are selected according to typical backbone link measurements available from [14]. The total rate of each pattern ranges between 0.005 and 0.3 of the entire packet stream. Within the packets that contain patterns, the rate is split uniformly among different patterns. A pattern may appear at any location within the packet payload; any payload bits other than the pattern are filled randomly. Confidence interval width β and error probability α for estimation are set to be 0.002 and 99.75%. This corresponds to 2.25 million samples under exhaustive testing and 3 million samples under lazy testing. We use 4 hash func-

tions for the Bloom filter. Bloom filter size is 10 K bits, unless mentioned otherwise.

In the following, we evaluate both the estimation accuracy and measurement overhead of lazy testing. Measurement overhead is reflected in two metrics: number of Bloom filter tests and number of actual string matching tests.

Estimation accuracy

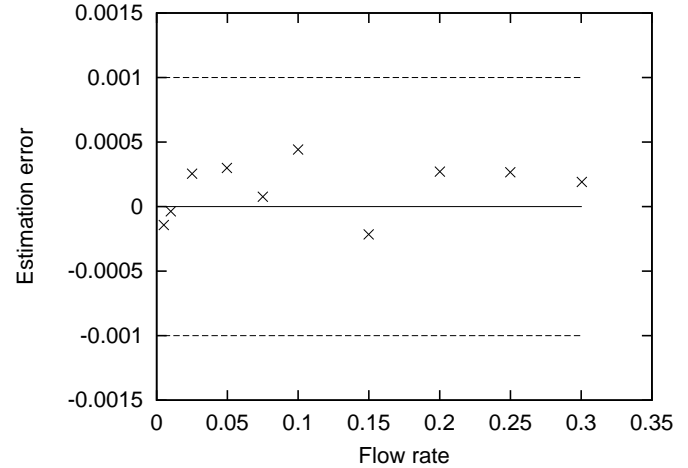


Fig. 12. Pattern estimation: estimation error

Figure 12 shows the estimation error of the lazy testing scheme under pattern flow rates ranging from 0.005 to 0.3. Note that each data point corresponds to a separate experiment. We define the estimation error as estimated flow rate minus actual flow rate. Confidence intervals ($\pm \frac{\beta}{2}$) are shown in the figure. Estimation error for exhaustive testing is similar and not shown. We observe that the estimation error is well within the confidence interval.

Measurement overhead

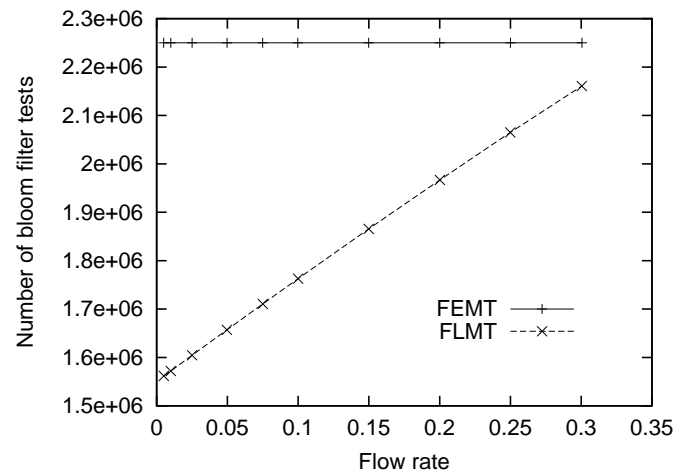


Fig. 13. Pattern estimation: necessary condition tests (Bloom filter)

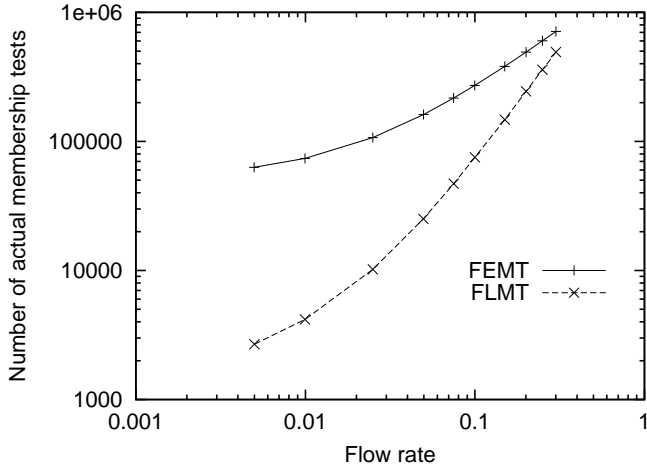


Fig. 14. Pattern estimation: sufficient condition tests (actual string matching)

Measurement overhead consists of both the Bloom filter test overhead and the actual string matching overhead. Figure 13 compares the number of Bloom filter tests with both exhaustive testing and lazy testing. Figure 14 compares the number of actual membership tests of the two schemes. We find that lazy testing outperforms exhaustive testing across a wide range of commonly occurring flow rates. Lazy testing reduces the number of actual membership tests by up to one to two orders of magnitude, and also significantly reduces the number of Bloom filter tests. The advantage of lazy testing is the most evident when flow rate is small, similar to the first experiment. Note that even in the unlikely case where the flow rate turns out to be relatively large (and thus outside the range tested above), the number of actual membership tests performed by lazy testing is at most $4/3$ times that of exhaustive testing (as can be seen from the expressions for T_{FL}^1 and T_{FE}^1). Thus lazy testing results in a saving of upto a few orders of magnitude when the flow rate is small, but only a small extra overhead in the rare scenario where the rate turns out to be large.

Impact of Bloom filter size on measurement overhead

Figures 15 and 16 show the number of Bloom filter tests and actual membership tests vs. Bloom filter size, respectively. In this experiment, we find that although lazy testing always has fewer actual membership tests than does exhaustive testing, it incurs more Bloom filter tests when the Bloom filter size is small (e.g., ≤ 1 K bits). This is because lazy testing has longer sampling time than exhaustive testing for the same level of accuracy. When Bloom filter size is small and false positive ratio is high, both schemes have to test almost all packets, and hence lazy testing loses its advantage in terms of filter testing overhead. This result suggests that an effective (low false positive) filter is important for the performance of lazy testing.

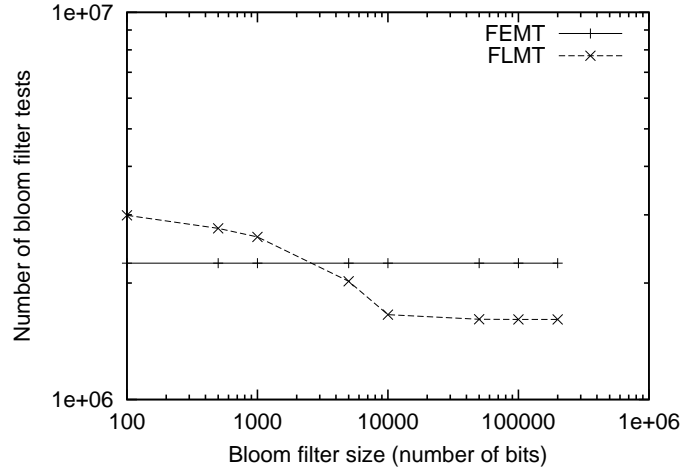


Fig. 15. Pattern estimation: impact of Bloom filter size on necessary condition tests

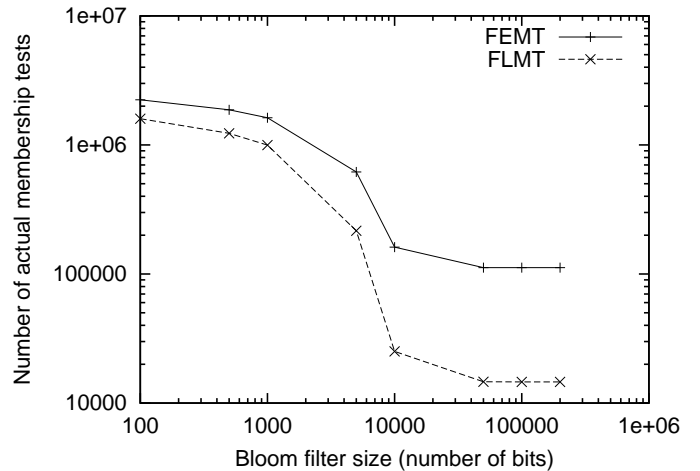


Fig. 16. Pattern estimation: impact of Bloom filter size on sufficient condition tests

VIII. CONCLUSIONS

In this paper, we have focused on reducing flow membership testing overhead. We have proposed the lazy membership testing approach that can reduce the total number of tests by tracking pairs of arrivals that have the given property. Our analysis and experiments have shown the approach can reduce membership testing overhead significantly in most commonly occurring cases. When combined with an effective necessary condition test filter, it can reduce the membership test overhead by several orders of magnitude.

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] Meyn, S.P. and Tweedie, R.L., *Markov Chains and Stochastic Stability*, Springer-Verlag, 1993.
- [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] Feldmann, A. et al., "Deriving Traffic Demands for Operational IP Networks: Methodology and Experience", *Proceedings of ACM SIGCOMM'2000*.
- [12] Cochran, W. G., *Sampling Techniques*, John Wiley, 1977.
- [13] Rao, C.R., *Linear Statistical Inference and its Applications*, John Wiley, 1973.
- [14] Sprint ATL, <http://ipmon.sprint.com>.