Deterministic Packet Marking for Time-Varying Congestion Price Estimation

R.W. Thommes and M.J. Coates
Department of Electrical and Computer Engineering
McGill University
3480 University St
Montreal, QC, Canada H3A 2A7
Email: rthomm@tsp.ece.mcgill.ca, coates@ece.mcgill.ca

***Abstract*— The addition of the two-bit Explicit Congestion Notification (ECN) field to the IP header provides routers with a mechanism for conveying link price information necessary for the successful operation of a number of congestion control schemes. Two recent proposals for probabilistic packet marking at the routers allow receivers to estimate path price from the fraction of marked packets. In this paper we introduce an alternative deterministic marking scheme for encoding path price. Each router quantizes the price of its outgoing link to a fixed number of bits. Every data packet sent along the path encodes a partial sum of the quantized link prices in its ECN field, allowing the receiver to estimate the path price. We evaluate the performance of our algorithm in terms of its error in representing prices, and compare it to probabilistic marking. We show that based on empirical Internet traffic characteristics, our algorithm performs better when estimating time-varying prices and static path price using small blocks of packets.**

Keywords: statistics, network measurements, packet marking, congestion control

I. Introduction

Recently, a number of optimization-based network congestion schemes have been proposed [1]–[7]. Most of these are *price*-based congestion control protocols: they require routers to maintain a congestion price which is a function of the arrival rate of incoming traffic and the capacity of the outgoing link. Each router must (indirectly) convey information about its calculated congestion price to the source so that its rate can be changed as required. The proposed two-bit Explicit Congestion Notification (ECN) field addition to the IP header facilitates a mechanism for conveying price information [9]. Several of the proposed congestion control protocols separate the tasks of calculating the link price and communicating it (through packet marking) [2], [6], [7]. Two probabilistic marking proposals have emerged to carry out the latter task [2], [10]. Both allow receivers to estimate the path price – the sum of the link prices – by examining the proportion of marked packets. In this paper we present and evaluate a novel deterministic marking scheme for encoding and estimating path prices.

A. The Marking Problem

The problem we consider in this paper is that of determining the sum of the prices of a set of links making up a path between a source and receiver. We now formalize the problem as in [10], slightly deviating occasionally. Consider a set of links $1, \ldots, n$ constituting an end-to-end path from a source to a receiver. Each link i has a non-negative price s_i. Define $z_n = \sum_{i=1}^{n} s_i$ as the sum of the prices along the path. The routers pass price information to the receiver by encoding it in data packets traversing the path. Routers may modify the two congestion notification (CN) bits present in the header of every packet. We denote the value of these bits after undergoing the marking action of link/router i by X_i. X_i can take on four possible values: $00, 01, 10, 11$. As per the the proposal of RFC 3168 [9], the (00) state is reserved to communicate that ECN is not being used. This leaves three codepoints for passing information.

After calculating z_n based on the receipt of marked data packets, the receiver must communicate its estimate to the sender. Our primary focus in this paper is on the problem of routers conveying congestion price informa-

This research was sponsored by the Natural Sciences and Engineering Research Council of Canada (NSERC) through the Discovery Grants Program.

This paper expands on material [8] we presented at IEEE Infocom 2004.

tion to the receiver. Our objective is to design a *marking algorithm* that routers apply to the IP ECN field of all packets traversing the path in order to provide congestion information to the receiver.

According to the authors of [10], a marking algorithm must obey some design constraints. It has to be fully distributed, meaning that a router may only use local information – the current link price and the value of the ECN field in the IP packet header – when making a marking decision. Furthermore, the marking algorithm should not keep any per-flow state information, or retain a history of how previous packets were marked. Adler et al. [10] make the claim that "there is no deterministic marking algorithm under these conditions". They present Random Additive Marking (RAM), a probabilistic algorithm, and compare it to Random Exponential Marking (REM). REM is another probabilistic algorithm, proposed by Athuraliya et al. [2].

A binomial distribution lurks beneath any probabilistic marking scheme, meaning one can at best achieve a linear decay in variance and mean-squared error with respect to the size of the block of data packets upon which the estimate is based. Thus, substantial variance in price estimates may persist even when estimates are based on a significant number of data packets. Due to the fact that path prices are dynamic, accurate estimates have to be obtained using a relatively small number of packets. Furthermore, the actual value of link prices has a significant impact on estimation error: in order to garner an accurate estimate with RAM, link prices must be close to 0.5 (assuming they are bound between zero and one). In the case of REM, performance is heavily affected by the choice of its parameters and how well they match the given path lengths and prices.

We believe that Adler's design constraints are unnecessarily restrictive. Since these constraints allow routers to read a packet's ECN field, it seems there is no increase in the fundamental difficulty of practically implementing a marking algorithm which allows routers to read additional fields in the IP header. Specifically, if we expand the list of locally available information to include the value of the time-to-live (TTL) and IP Identification (`IPid`) fields in each packet's IP header, it is possible to define a deterministic marking algorithm[1]. In this paper we outline a deterministic quantized marking (DQM) algorithm and evaluate its performance. In order to make deployment of our algorithm more attainable, we have endeavoured to introduce as few changes to the current protocols as possible. The scheme does not require any modification to the TCP or IP headers.

[1]Adler also assumes only one bit in the packet header is available for marking. While our algorithm makes use of both bits in the ECN field, it could be simply modified to function using only one bit.

The paper is organized as follows. In Section II, we review the current IP standard for packet marking and examine in more detail the probabilistic marking algorithms, REM and RAM. In Section III we propose a new deterministic marking algorithm and describe its structure. In Section IV, we analyse the performance of the deterministic algorithm in terms of mean-squared error and make comparison with RAM. In Section V, we examine the performance of our algorithm based on Internet trace-driven simulations. Finally, in Section VI we present what we consider to be the most significant results in our paper: we consider several scenarios with time-varying link prices and explore how well our algorithm and RAM track the path price.

II. Packet Marking and Probabilistic Techniques

A. IP Standard

ECN provides intermediate network routers with an alternative to dropping packets in order to signal congestion to end systems. RFC 3168 provides details of the proposed ECN addition to the IP protocol [9]. It specifies that the two-bit ECN field, comprising four *codepoints*, may be used to indicate the presence of congestion. In order to ensure backwards-compatibility, one codepoint (00) is used to indicate that a sent packet does not support ECN marking.

RFC 3168 indicates that when ECN is used, the sender initializes the ECN field to one of two codepoints (01,10). This indicates ECT (ECN-Capable Transport), implying the sender and receiver support the use of ECN. An intermediate node experiencing congestion will set the ECN field of a received packet with an ECT codepoint to to the Congestion Experienced (CE) codepoint (11) to indicate its congestion to the receiver. When an end system receives a single packet with the CE codepoint set, the RFC requires that the end system transport layer protocol respond in essentially the same manner as when it detects a dropped packet.

RFC 3168 does not specify how routers are to calculate congestion nor how they shall decide whether to mark a given packet.

B. REM

We now provide a brief summary of the Random Exponential Marking (REM) scheme proposed by Athuraliya and Low in [2]. Prices are unbounded, but must be non-negative. The ECN codepoint X_0 is initialized to 01 (10 could also be used). If $X_{i-1} = 01$, the i-th router sets the ECN codepoint to 11 with probability $1 - \phi^{-s_i}$. $\phi > 1$ is a parameter chosen by the designer. If X_{i-1} is already 11, the i-th router makes no change. The value of the codepoint at the receiver is X_n. $X_n = 01$ with probability $\phi^{-\sum_{i=1}^{n} s_i}$ and $X_n = 11$ otherwise. The indicator function $\mathcal{I}[X_n = 11]$ has an expected value of $1 - \phi^{z_n}$. After collecting N packets the receiver may estimate the total price by computing $\overline{X} = \sum_{i=1}^{n} \mathcal{I}[X_n = 11]$ and forming the estimate $\widehat{z_n} = -\log_\phi(1 - \overline{X})$. This estimate is biased, but does converge to z_n almost everywhere, almost surely [10].

The local computation requires no information aside from the link price, but the choice of ϕ is difficult. The performance of REM is highly dependent on the value of ϕ, as demonstrated by Adler et al. [10]. The optimal choice of ϕ requires knowing the path length and end-to-end price, which means that it generally cannot be deduced in a practical setting. A suboptimal choice of ϕ can result in poor estimation performance.

C. RAM

Random Additive Marking (RAM), proposed by Adler et al. [10], is an alternative probabilistic marking scheme. It requires that each link's price fall in the range bounded by 0 and 1, and that each router be aware of its position i within the path. Routers may estimate their position using a method based on the TTL field in the IP header [10]. The authors demonstrate that price estimation performance does not significantly decline when applying this method. Although the RAM algorithm is described in terms of a single bit, for consistency with our notation we provide an equivalent 2-bit description. Initially $X_0 = 01$, and the i-th router in the path leaves the ECN field X_{i-1} unchanged with probability $(i-1)/i$,sets it to 11 with probability s_i/i, and sets it to 01 otherwise.The expectation of the indicator function $\mathcal{I}[X_n = 11]$ is equal to the expectation of $\sum_{i=1}^{n} s_i/n$. Upon receiving N data packets, the receiver may produce an unbiased estimate of the path price by computing the average of the indicator function and multiplying it by the path length n.

Adler et al. compare RAM and REM under the assumption of independent, uniformly distributed link prices normalized to have a mean price of approximately 0.5. They show that RAM always exhibits a lower error probability than REM even when the optimal value of ϕ is used. In addition, the performance of RAM, unlike REM, is not affected by path length. However, it suffers severely when the average link price strays significantly from 0.5.

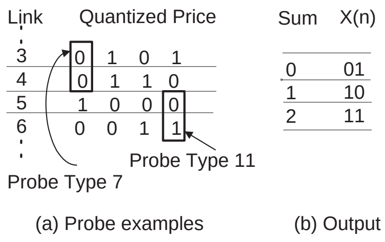


Fig. 1. Example of the operation of Deterministic Marking Algorithm I. (a) The nature of the probe types – each performs a sum of two bits, as indicated by the boxes. (b) The state output of the sum.

III. Deterministic Packet Marking

A. Preliminaries

We now specify a marking mechanism which allows the routers on a path between transmitter and receiver to convey the sum of their *quantized* prices to the receiver. Our proposed algorithm makes use of only the two existing ECN bits in the IP header, and retains the RFC allocation of the 00-codepoint [9]. However, we modify both the router marking algorithm and the manner in which the receiver interprets the marking information. Unlike REM and RAM, our marking scheme is deterministic in the sense that *every* packet is marked, and the marking is performed according to a deterministic function applied to quantized link prices.

As with RAM, our scheme requires that every link price s_i is bounded: $0 \leq s_i \leq 1$. Every router calculates a b-bit uniform quantization of its congestion price. We use $b = 4$ for our empirical analysis and examples in this paper, as this value provides a good trade-off between quantization error and the number of bits needed to describe the price estimates.

The key idea behind our scheme is that each data packet encodes the sum of a small subset of link price bits. The receiver can reconstruct the sum of the quantized prices (and hence form an estimate of the path price) by combining all the partial sums. All data packets are mapped into so-called *probe types*, and each probe type may be modified by two routers so that it carries the sum of two bits of equal significance of the quantized prices of the two links outgoing from the routers. Figure 1(a) provides an example: probe type 7 carries the sum of the most significant bits (MSBs) of the quantized price of links 3 and 4. Figure 1(b) indicates the output of the sum. In order to initialize the summation procedure, X_0 is set to 01. Subsequently, only designated routers can change the ECN field; in this example routers 3 and 4. These routers modify the ECN field if the price bit of their outgoing link to which the probe type corresponds (the MSB in our example) is equal to 1. In this case, the ECN field is updated according to table 1(b); i.e. state 01 is changed to 10, and 10 to 11. Thus, the data packet indicates the sum of the two bits when it arrives at the receiver.

It it important to restrict the number of probe types to a reasonable value. If we limit the maximum path length to a value $n_{\max}$, the number of probe types required is $b\lceil n_{\max}/2\rceil$. We use $n_{\max} = 30$ in this paper, to facilitate a concrete description of our technique. This implies a need for 60 probe types. We choose $n_{\max} = 30$ due to the results of Begtavesic et al. [11], which indicate that paths of lengths greater than 30 are exceptionally rare in the Internet. In the unlikely event that a path does contain $n > 30$ (but less than 60) links, $2(n - 30)$ of the probe types are susceptible to overflow error because their ECN field may be incremented by three routers. An overflow will only occur if all three routers attempt to encode a price bit of 1. Therefore, the resulting estimation error is unlikely to be catastrophic unless the path length is significantly longer than 30 links.

In order for our marking procedure to function correctly, we must define a mapping that labels each data packet as a certain probe type. This mapping makes use of the IP identification field. In addition, the routers must have a way to determine whether to modify a given probe type. They make this decision by comparing the time-to-live (TTL) field with the probe type label.

B. The IP identification field and Probe Types

The `IPid` field provides a mechanism for fragment reassembly. According to RFC 791 [12] it "is used to distinguish the fragments of one datagram from another" and for each datagram, the identification field must be set to "a value that must be unique for the source-destination pair and protocol for the time the datagram will be active in the internet system." There are no additional requirements placed on the actual value of the `IPid` field.

A technique for counting NATted hosts, introduced by Bellovin [13], exploits the manner in which many hosts implement the `IPid`. Bellovin makes the observation, also noted in [14], that many hosts implement the IP field using a simple counter. This means successive data packets emitted by a host carry sequential `IPid` values. However, there are other, less common, approaches. Some hosts use pseudo-random number generators, while others use byte-swapped counters [13]. Some versions of the Solaris operating system implement separate sequence number spaces for each (source, destination, protocol) triple [13].

Since the 16-bit `IPid` field can take on significantly more unique values than the required $b\lceil n_{\max}/2\rceil$ (60 in our example) probe types, we require a mapping function. The function we choose sets the probe type identifier to $\texttt{IPid} \bmod m$, where m is a prime number slightly larger than our required number of probe types. The majority of the resulting m probe types carry price information, while the remaining probe types are reserved for other forms of communication. One of the major goals of this mapping function is that each probe type appears as regularly as possible (ideally once every 60 packets). The counter implementation and the Solaris mechanism are of substantial benefit here.

In the case where the host implements the `IPid` field as a counter, the sequence of data packets sent between a given host and receiver cycles through the probe types, skipping some values whenever packets belonging to other streams intervene. Each probe type appears regularly with this implementation. In the ideal scenario – where a separate space is dedicated to each (source, destination, protocol) triple, the `IPid` field is generated by a counter, and every data packet belonging to that triple is used to calculate the path price – each probe type appears once every m data packets.

In the next section, we analyse the performance of our marking algorithm assuming random `IPid` generation. The empirical survival function of the spacing between probe types of the same kind using data collected from the Internet is shown in Figure 2(a) in Section V. It illustrates that in most cases the spacing is slightly larger than m, which suggests the prevalence of counting implementations in the Internet. A strictly random `IPid` generation results in a substantially heavier tail.

C. The router marking algorithm

When a data packet arrives at a router, the router calculates its *Link ID* as $\mathrm{LinkID} = \texttt{TTL} \bmod n_{\max}$. Secondly, the router calculates the probe type as $\mathrm{ProbeType} = \texttt{IPid} \bmod m$. The router then uses the pair (Link ID, Probe Type) to determine whether it should perform any action and, if so, what price bit it should encode. The router makes its decision by consulting a $n_{\max} \times m$ lookup-table in its memory. This table, which is static and identical in all routers, returns an entry of *"perform no action"*, or *"increment ECN field if bit j=1"*, where

$j \in (1, 2, ..b)$. Every column of the table contains two entries of the latter variety, corresponding to the two routers which may modify a given probe type. Each row of the table contains b such entries, one for each quantization bit. If the table informs a router to modify an incoming data packet, and the indicated price bit is 1, the router increments the state X_{i-1}, from 01 to 10 or from 10 to 11. In order for this marking algorithm to work properly, we must make two mild assumptions. First, to ensure that at most two routers modify the ECN field of a probe type, every router must decrement the TTL field by 1. Second, to ensure that disparate data packets corresponding to the same probe type are marked by the same routers, the TTL field for every data packet of a given stream must be initialized to a constant value by the sender.

We summarize the marking procedure with the the following pseudocode. The procedure $MarkingDecision$ takes, as its parameters, pointers to three fields in the header of the currently buffered IP packet. Each router is aware of the constants $nmax$, m, and $MarkingTable$, the lookup table described above. $MarkingTable$ has entries of zero to indicate no marking action is to be performed, or positive integers corresponding to the price bit on which to base the marking decision. Routers store their current price estimate in the bit-array $CurrentPrice$ of length b.

```
procedure    $MarkingDecision(\&ttl, \&ipid, \&ecn)$

 1: $LinkId \leftarrow *ttl \bmod nmax$
 2: $ProbeType \leftarrow *ipid \bmod m$
 3: $BitToMark =$
    $MarkingTable(LinkId, ProbeType)$
 4: if $BitToMark \neq 0$ then
 5:    $PriceBitValue =$
       $CurrentPrice(BitToMark)$
 6: end if
 7: if $PriceBitValue == 1$ then
 8:    if $*ecn == 01$ then
 9:       $*ecn = 10$
10:    else
11:       if $*ecn == 10$ then
12:          $*ecn = 11$
13:       end if
14:    end if
15: end if
```

D. *Path Price Estimation*

When every probe type has arrived at the receiver at least once, it can determine the sum of quantized prices exactly and the estimation error will be equal to the quantization error (as detailed in the following section). However, in any practical scenario, the path price can only be assumed to remain fixed for short durations. We specify two procedures for estimation in such a scenario:

1) *Block-based estimation*: After receiving a block of K data packets, form the path price estimate based on the available probe types. If one or more probe types are missing, insert values that minimize the expected error under the assumption of a uniform link price distribution.
2) *Time-varying estimation*: Upon the reception of a data packet, form a new estimate based on the values of the most recently received instances of each probe type.

In the following section, we provide bounds on the mean-squared error under the block-based estimation approach. In Section V we analyse performance empirically using probe type sequences derived from Internet traces.

IV. Error Analysis

In this section, we analyse the performance of our marking algorithm based on fixed block lengths, with probe types generated independently and uniformly over the possible m values. Under the assumption of uniformly distributed link prices over the range $[0, 1]$, we derive a bound on the mean-squared error. In Section V we perform an empirical analysis of mean-squared error for comparison.

There are two potential sources of error when a receiver calculates the end-to-end price using the deterministic marking algorithm. One is due to the quantization, and the other is due to the possibility of one or more unique probes types not being observed. We assume that the path is of length n links, each link price is quantized using b bits, a block size of K packets is examined by the receiver to determine the end-to-end price,and the number of unique probe types m is $\lceil \frac{nb}{2} \rceil$. We commence with some background discussion.

1) Quantization Error: Under the assumption that the price of each link is uniformly distributed and normalized to fall in the interval [0,1), an ideal b-bit quantizer is obviously also uniform. That is to say, its representation points will of the form

$$\left\{ \frac{1}{2^{b+1}}, \frac{3}{2^{b+1}}, \frac{5}{2^{b+1}}, ..., \frac{2^{b+1}-1}{2^{b+1}} \right\} \qquad (1)$$

The quantization error e_q is uniformly distributed between the two extremes: $f(e_q) \sim U\left(\frac{-1}{2^{b+1}}, \frac{1}{2^{b+1}}\right)$ and has a variance of $Var(e_q) = \frac{1}{12\cdot 2^{2b}}$.

2) Error due to Missing Bits: In order to examine the error in representing the price of a link due to missing one or more of the b bits, we consider the decimal representation q_d of a binary quantization value $(a_0, a_1, ..., a_{b-1})$ where a_0 is the MSB (most significant bit). It is given by:

$$q_d = \frac{1}{2^{b+1}} + \left(1 - \frac{1}{2^b}\right)\left(\frac{a_0 2^{b-1} + ... + a_{b-1}2^0}{2^b}\right) \quad (2)$$

For the purposes of estimating q_d we adapt the approach of replacing a missing bit a_i by its expected value under the assumption of a uniform distribution. Since any bit is equally likely to be 0 or 1, a missing bit will always be replaced by a value of $1/2$. Clearly this results in a lower error variance than if a missing bit is randomly chosen to be 0 or 1. The error due to a missing bit a_i is a discrete uniform random variable (denoted by UD) taking on one of two values with equal probability:

$$e_{a_i} \sim U_D\left\{-\left(1 - \frac{1}{2^b}\right)\frac{1}{2^{2+i}}, \left(1 - \frac{1}{2^b}\right)\frac{1}{2^{2+i}}\right\} \quad (3)$$

The expected value of the error e_{a_i} is 0, and its variance is given by: $Var(e_{a_i}) = \frac{(1-\frac{1}{2^b})^2}{12\cdot 2^{2i+2}}$

A. Distribution of Missing Probe Types

The distribution of the number of probes not observed in a block of size K is strongly dependent on how the IP identifier field increments between contiguously transmitted packets. Ideally, the identifier increases by one every time, in which case any block of $K > m$ is guaranteed to include at least one packet mapping to every probe type. However, it is known (and verified by empirical data) that some hosts on the Internet generate IP identifiers that change in a random manner. In this section, we consider the case where identifiers and hence probe types are randomly selected based on a uniform distribution. We note that there exist theoretical cases worse than randomly distributed identifiers. For instance, if all identifiers equivalent to a certain value $s \bmod m$ are skipped by the host, then the probe type corresponding to this value will never be observed for any sequence length. However, our observation of Internet traffic does not provide evidence that such pathological behaviour is likely to occur.

Due to our assumption of all probe types being equally likely, the analysis of the number of missing probes is an instance of the classical occupancy problem [15], [16]. Briefly, this problem considers the number of empty bins resulting from a random allocation of K balls into m different bins.

B. A Bound on the Expected Mean-Squared Error

We now derive a bound on the expected mean-squared error of the end-to-end price estimate assuming uniform generation of probe types. We make use of another result related to the Occupancy Problem – an upper bound $H(K, m, W)$ on the probability $p(W|m, K)$ of missing W probe types when K probes are generated and there are m unique types (from Theorem III in [17]):

$$H(K, m, W) =$$
$$\begin{cases} \exp - \left(W \ln\left(\frac{W}{E[W]}\right) + E[W] - W\right), W \geq E[W] \\ \exp - \left(\frac{W^2}{2E[W]} + \frac{E[W]}{2} - W\right) \qquad , W < E[W] \end{cases} \quad (4)$$

where $E[W] = m\left(1 - \frac{1}{m}\right)^K$.

Now, given that W probe types are not observed, the scenario that maximizes the contribution to the total error is if all missing probes are of the types measuring the MSB of the quantized price. Thus, we will assume that $2W$ MSBs are unobserved. If $2W$ exceeds n, some of the missing probes must correspond to non-MSB bits. However, we will model all missing probes as contributing an error equal to the maximum. This is in keeping with deriving an upper bound on total error variance.

Each unobserved probe results in two MSBs being replaced by their expected value of $1/2$ for purposes of calculating the quantized price for a given link. Given that an MSB a_0 contributes $\left(1 - \frac{1}{2^b}\right)\left(\frac{a_0 2^{b-1}}{2^b}\right)$ to the quantized price, a missing MSB results in a discrete error taking on the values $\pm\left(1 - \frac{1}{2^b}\right)\left(\frac{1}{2^2}\right)$ with equal probability. We can thus upper-bound the error due to a missing MSB (and hence any bit) at $\pm 1/4$ and the variance at $1/16$.

Finally, due to independence, the variance in the error of the path price estimate is just the sum of the individual variances, given by $\frac{n}{12\cdot 2^{2b}} + \frac{W}{8}$.

Multiplying this variance by (4) and summing over all possible values of W then gives the following result:

Theorem 1: For an estimation block of K probes uniformly drawn from m unique probes, with n links in the path and b bits used to quantize each link price, the expected mean-squared error in the path-price estimation is bounded as:

$$MSE(m, b, K, n) \leq \sum_{W=0}^{K-1} min(H(K, m, W), 1)\frac{W}{8} + \frac{n}{12 \cdot 2^{2b}} \quad (5)$$

where $H(K, m, W)$ is determined from (4).

V. Simulation Performance

In this section of the paper, we analyse the performance of the estimation algorithm using trace-driven simulations. During the last week of June, 2003, we collected 100 traces of 2000 packets with the program TCP-Dump by downloading files from 100 different servers. Of these servers, 50 were based in USA/Canada, 25 in Europe, 15 in Asia and Oceania and the remaining ten in other locations worldwide including South America and Africa. Using a simple script, we extracted the IP identifier field for each packet. Next, we mapped the field into a probe type using $m = 63$. We perform our analyses assuming a maximum path length of 30 (the TTL is taken modulo 30) and 4 bit quantization. These settings imply that the algorithm requires 60 probe types for estimation. The remaining three are reserved for protocol communication.

Since the performance of the algorithm is dependent on the properties of the sequence of IP identifiers in received packets, a closer examination of the traces is in order. 44 of the traces exhibited a consistent increase of 1 in the IP identifier of successive packets . This is the ideal case for our algorithm, as it ensures that any block of size m contains all probe types. 53 of the traces exhibited a general sequentially increasing trend, but on occasion the IP identifiers of contiguous packets did not increase by 1 (i.e. there were ”skips”). The behaviour of all servers with sequentially increasing IP identifiers is summarized in Table I.

The remaining three traces had randomly changing IP identifiers. We conducted a simple statistical autocorrelation test on each sequence that suggested the sequences were truly random.

For most servers, we were able to determine the operating system being used. The distribution is given in Table II.

Percentage of Received Packets with IP Identifier Increasing by 1	*Number of Servers Displaying this Behaviour*
100	44
99+	16
90-98	18
80-89	5
70-79	3
60-69	1
50-59	6
less than 50	4

TABLE I. The breakdown of observed server `IPid` behaviour

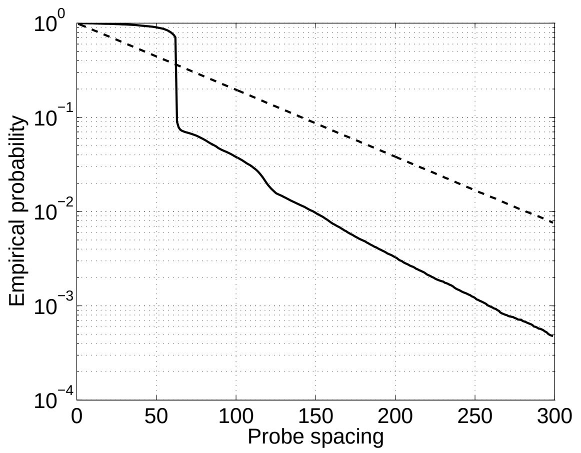
Operating System	*Number of Servers*
Linux	53
FreeBSD	23
Solaris	15
Windows	6
Unknown	3

TABLE II. The breakdown of observed Server Operating Systems.

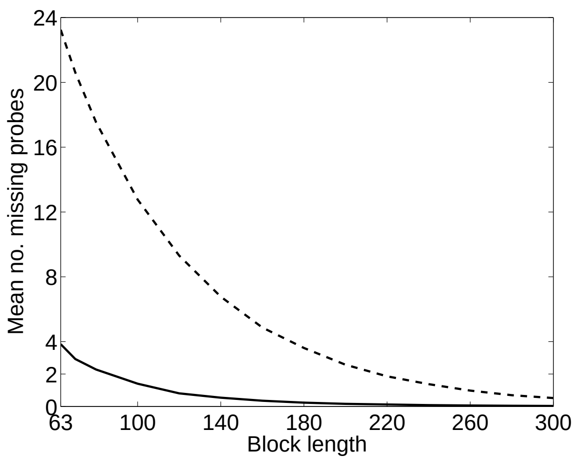
A. Missing Probe Types

First, we examined the nature of the distribution of probe types in the empirical data sequences. It is important that every probe type appears regularly. Figure 2(a) shows the survival function of the spacing between probes of the same type over the 100 traces. The function generated from a uniform allocation of probe types is shown for comparison. If the IP identifier were implemented as a strict counter, with no intervening packets from the host, probe types of the same kind would always be 63 packets apart. The empirical survival function indicates that the counter implementation is widespread; more than 90 percent of the time, the spacing is 63 packets. Sometimes the probe types are spaced much further apart; this occurs due to pseudorandom implementations and busy servers where the counter frequently skips. 99 percent of the time, the spacing between probes of the same type is less than 150 packets. Clearly, the uniform generation of probe types results in a substantially heavier tail in the spacing distribution (although the decay rate is approximately the same).

The second feature that interests us for the purposes of block-based path price estimation is the number of probe types missing from a block of size K. Figure 2(b) shows the mean number of missing probes as a function of block length for the empirical traces and for uniform random allocation. For a block length of 63, the mean number of missing probe types is slightly less than 4 in the empirical case (as compared to 24 for uniform allocation). The effects of the IP identifier counter implementation are again evident. The mean number of missing probe types is substantially less than 1 when the block size is 140.



(a) The survival function of the spacing between probes of the same type.



(b) The mean number of missing probes as a function of block length.

Fig. 2. Analysis of the distribution of probe types. In each plot, the solid curve corresponds to the Internet traces and the dashed curve to uniform random allocation of probe type.

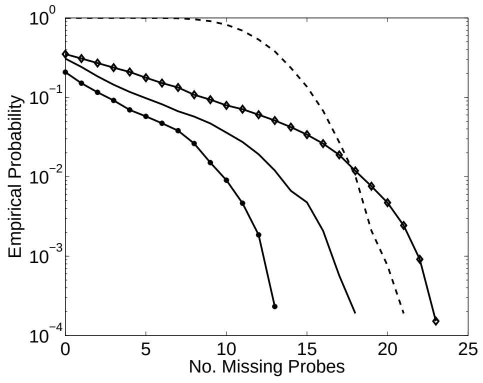


Fig. 3. The survival function for the number of missing probe types. Dashed line corresponds to uniform random probe type allocation for a block length of 100. Solid lines correspond to Internet traces; diamond marker is block length 80, no marker block length 100, * marker block length 120.

Figure 3 show the empirical survival functions for the number of missing probes in a block for the Internet traces. In the case of a block size of 100 packets, 80 percent of the time there are no missing types, and approximately 95 percent of the time there are less than 10 missing types.

B. Error Analysis

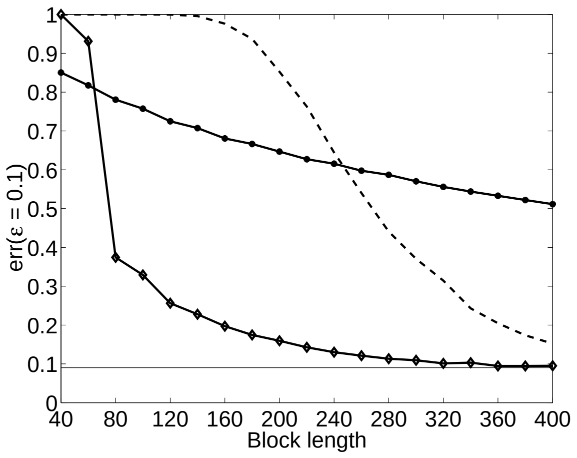
We performed an examination of the error probability of the deterministic estimation to enable comparison with RAM [10] probabilistic price estimation. The error probability $err(\epsilon)$ is defined as the probability that the path-price estimate falls outside of a range (defined by ϵ) about the true price:

$$err(\epsilon) = 1 - \Pr[(1-\epsilon)z \leq \widehat{z} \leq (1+\epsilon)z] \qquad (6)$$

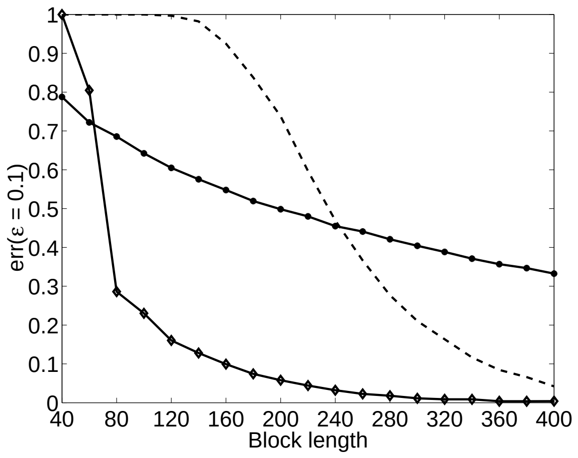
We examined two cases, each with a path consisting of 10 links. In case 1, all link prices were uniformly distributed over $[0, 0.2]$. In case 2, all link prices were uniformly distributed over $[0.4, 0.6]$. For both cases, one hundred realizations of link prices were generated, and the error probability was evaluated for each realization using the Internet traces to generate probe types. The results were averaged to determine an average error probability.

Figure 4 displays the resulting averages for $\epsilon = 0.1$ as a function of block length. The performance of RAM is shown for comparison (with results derived from expressions in [10]). Both algorithms show improved performance when the marking range is centered around 0.5. When the Internet traces are used to generate probe types for these two scenarios, the deterministic algorithm outperforms RAM for block lengths greater than approximately 60. However, RAM performs better for block sizes below 240 when probe types are generated from a uniform random distribution.

We note that as the level of congestion is lowered, RAM will eventually provide a lower error probability than the deterministic algorithm for any block length. Since the expected quantization error inherent to our algorithm is constant regardless of path price, the relative error grows as the path price decreases. Thus, if a network is expected to operate under congestion levels consistently near zero, RAM may be a better choice for

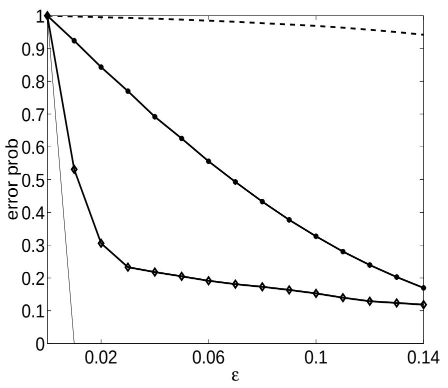


(a) Link prices uniformly distributed over the range 0 to 0.2

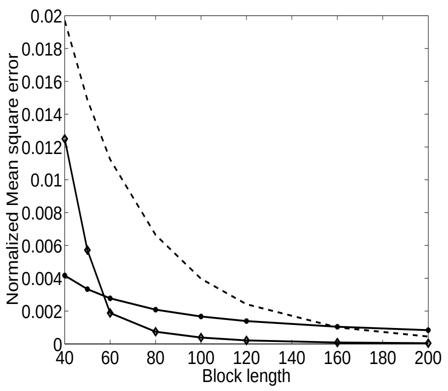


(b) Link prices uniformly distributed over the range 0.4 to 0.6

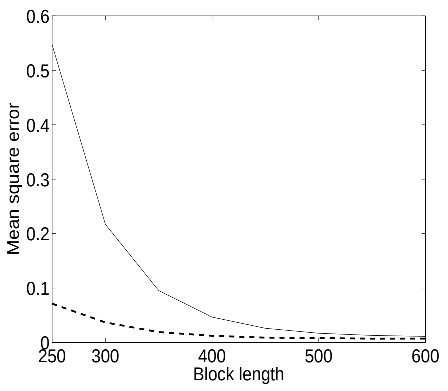
Fig. 4. Error probability as a function of block length for a path of 10 links and $\epsilon = 0.1$. The solid line with diamond markers is the Internet trace deterministic algorithm. The * markers are the RAM algorithm. The dashed line is the deterministic algorithm with uniform random probe type generation. The thin, solid line in figure (a) represents the lower bound on the error probability due to quantization. The lower bound is not shown in the other figure, as it is too close to zero be distinguishable.



(a) The error probability as a function of ϵ for a block length of 100.



(b) Normalized mean-squared error as a function of block length. Error is normalized by the number of links (20).



(c) Comparing the MSE bound to simulated MSE behaviour. The dashed line is the deterministic algorithm, and the solid line is the bound.

Fig. 5. Error Analysis – error probability and mean-squared error. The solid line with diamond markers is the deterministic algorithm with probe types generated from Internet traces. The * markers are the RAM algorithm. The path length is 20 links and prices are uniformly distributed between 0 and 1.The dashed line is the deterministic algorithm with uniform random probe type generation. The thin, solid line in figure (a) represents the lower bound on the error probability due to quantization.

estimating path prices. However, in such a situation, one may consider renormalizing the congestion metric so that prices are distributed over a wider range of values and the deterministic algorithm performs better.

C. Mean-Squared Error Analysis

We performed an analysis of the mean-squared error of the deterministic algorithm for the case of uniformly distributed link prices over $[0, 1]$. We generated 100 realizations of link prices for a path of length 20 links, and then evaluated the mean-squared error for each realization. The probe types were generated both from the Internet traces and from a uniform random allocation. Figure 5(a) shows the error probability as a function of ϵ. Figure 5(b) shows the normalized mean-squared error (the error is normalized by the number of links in the path). The mean-squared error performance of RAM is shown for comparison. As in error probability, the deterministic algorithm outperforms RAM for small block lengths when probe types are generated from the Internet traces. As the block length becomes large (> 1000), RAM begins to outperform the deterministic algorithm; the performance of the latter is bounded by quantization error.

We also examined the accuracy of the MSE bound derived in IV-B. Figure 5(c) shows the bound, along with the simulated MSE for a path length of 20 links whose price is estimated with random `IPid`s. The figure illustrates the range of block lengths over which our bound is somewhat tight. The bound offers little insight for block lengths of under 250.

Scenario	*MSE: Deterministic w/ random* `IPid`*s*	*MSE: Deterministic w/ trace-based* `IPid`*s*	*MSE: RAM*	*RAM Optimal Block Length*
1	0.630	0.270	1.078	450
2	0.786	0.223	1.314	380
3	1.296	0.904	1.325	500
4	0.845	0.224	1.782	370

TABLE III. Mean-squared error of estimates, and empirically determined optimal block length for RAM

VI. Time Varying Behaviour

We have, to this point, examined the performance of our algorithm and RAM in static price scenarios. A more realistic model of a network will include time-varying levels of congestion and hence time-varying link prices. In this section we consider how well the algorithms perform in a time-varying scenario.

The deterministic algorithm is naturally suited to estimating time-varying prices due to the periodic nature in which instances of a given probe type arrive. At any point in time, the receiver estimates the current path price by considering the most recent instance of every probe type received. When a packet arrives, the ECN value of the associated probe becomes the new current value stored by the receiver.

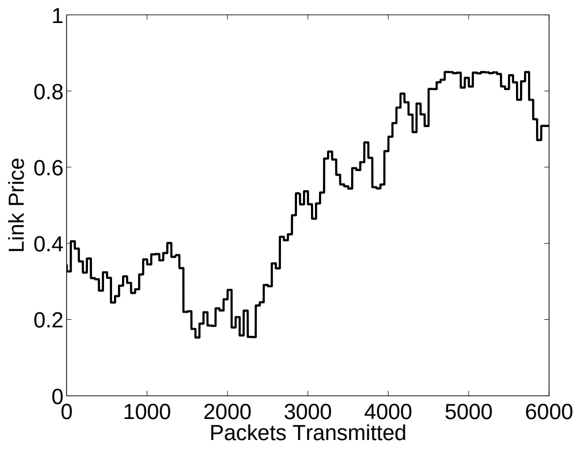
In order to adapt RAM to estimate a time-varying path price, one must choose an appropriate block length. The best choice is not apparent. On the one hand, the block length should be as short as possible, so that the algorithm can accurately detect price variations; if the block is so long so that the path price varies significantly while the packets making up the block are sent, the estimate will be an average path price rather than an instantaneous value. On the other hand, the expected estimation error of RAM decays linearly with block length, meaning a long block length is necessary to avoid significant estimation errors. The block length that garners the best result is dependent on the specific dynamics of the link prices. For each time-varying scenario, we experimentally chose a block length that minimized the sample mean-squared-error over all estimates. These block lengths are provided in Table III.

We consider 4 scenarios. In our first two scenarios link prices change after every block of 50 packets is transmitted. While the interval between price changes is the same for all links, the price transitions are not synchronized. This is accomplished by having the first price transition occur after a random number of probes - uniformly distributed between 1 and 50. Each link price is initialized to a random value uniformly distributed in the range [0.25, 0.75]. Every price change is normally distributed with mean 0 and standard deviation 0.1. In addition, there are reflective boundaries at 0.85 and 0.15 which define the range of values the prices may take on. The behaviour of each link price may be considered a random walk bounded in the range [0.15, 0.85]. The two scenarios model paths with 20 and 30 links.

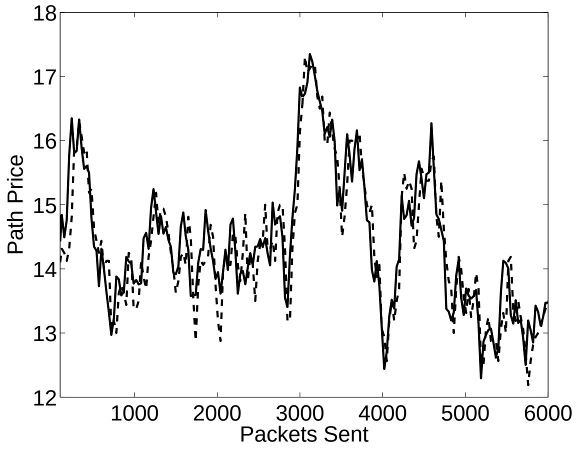
Figure 6(a) depicts the time-varying price behaviour of a sample link over the interval in which 6000 data packets are sent. Figure 6(b)-(d) shows a sample path price for the 30-link scenario, and the instantaneous estimates generated by RAM and the deterministic algorithm when exposed to data packets with random `IPid`s as well as Internet trace-based `IPid`s. The sample estimation plots suggest that RAM tends to experience longer runs of over/under estimation than the deterministic algorithm. In order to draw more meaningful conclusions, we consider $err(\epsilon)$ as defined in (6). We ran 100 iterations of each of the time-varying scenarios. Figures 7(a) and 7(b) depict the results. For every value of ϵ, both instances of the deterministic algorithm perform better than RAM. As with the simulations involving static prices, the deterministic algorithm performs better when subjected to Internet trace-based `IPid`s than to random ones.

Our final two scenarios involve link price that change less often - once every 100 packets. However, the standard deviation of each change is 0.2, and the reflective boundaries are at 0 and 1. The resulting $err(\epsilon)$ performance is shown in 7(c) and 7(d). The relative performance of the algorithms remains the same.

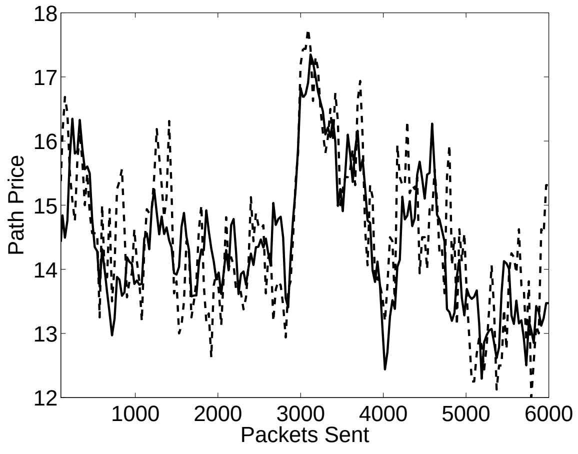
Another metric of interest in comparing the algorithms is the mean-squared error over all estimates. Table III summarizes this data. The deterministic algorithm exhibits the lowest MSE in all scenarios.



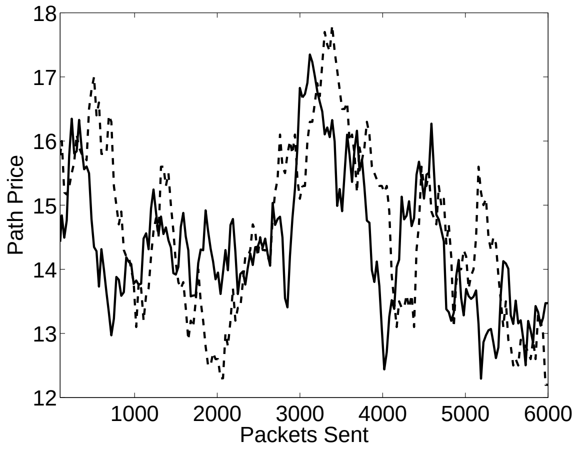
(a) Sample time-varying link price



(b) Path price and estimate generated by deterministic algorithm with Internet trace-based `IPid`s



(c) Path price and estimate generated by deterministic algorithm with random `IPid`s



(d) Path price and estimate generated by RAM

Fig. 6. Estimating a Time-Varying Price. Figure (a) is an example of how the price of one link varies with time. Figures (b)-(d) illustrate a sample path price as the heavy curve and the given algorithm's current estimate as the dashed line

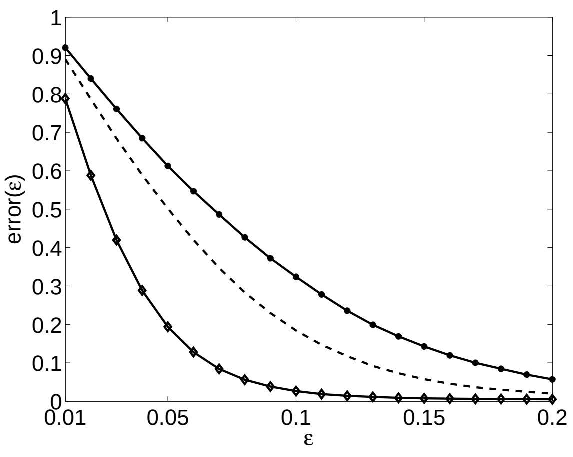
VII. Practical Issues

A. Receiver Feedback

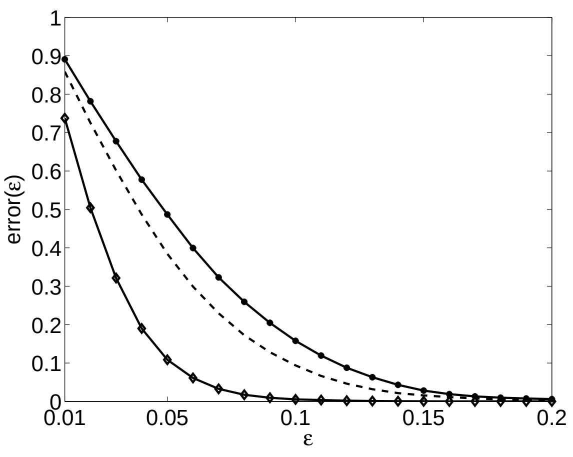
Any price-based congestion control protocol requires that the source be able to determine the price along its path to the receiver. As is the case for the REM and RAM proposals, we have not addressed the mechanism by which a receiver informs the sender of its path price estimate (the performance of REM in conjunction with RFC 3168-style feedback has been explored via simulation in [2]). RFC 3168 suggests the addition of a 1-bit ECN-echo (ECE) in the TCP header [9]. The RFC proposes that after a client receives a packet with the ECN field in the CE state, it set the ECE bit to 1 in every acknowledgment it sends until it receives a packet from the sender with the Congestion Window Reduced (CWR) flag set. The CWR flag, another new TCP flag suggested in [9], is set by the sender to acknowledge the receipt of an ECE packet. Unfortunately, this mechanism cannot be readily adapted to provide explicit feedback of the path price to the sender. The fundamental problem is that there is the potential for multiple acknowledgments to be sent with the ECE field set in response to a single received packet with the ECN field in the CE state. This means that the sender is unable to accurately determine what proportion of its sent packets were marked. Thus, a novel approach is necessary in order to provide more informative price feedback.

B. Security

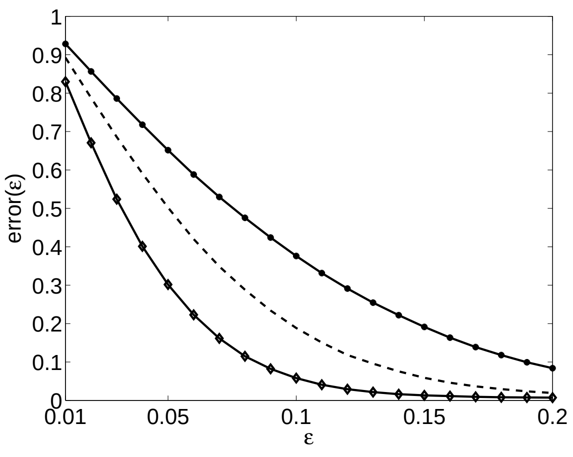
Although the specific method used by the receiver to convey its path price estimate back to the sender lies



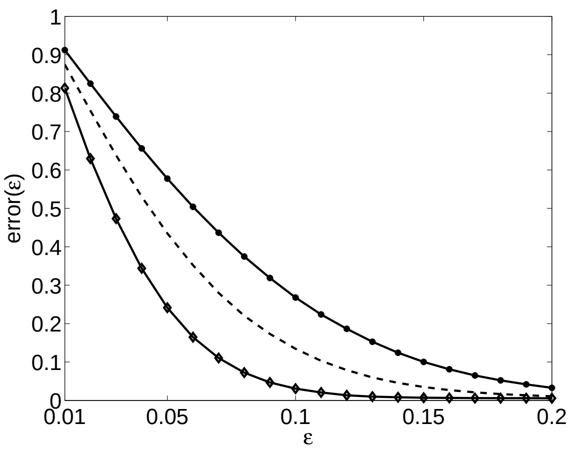
(a) Error probability for Scenario 1: 20 links.



(b) Error probability for Scenario 2: 30 links.



(c) Error probability for Scenario 3: 20 links.



(d) Error probability for Scenario 4: 30 links.

Fig. 7. Error probability $err(\epsilon)$. The solid line with diamond markers is the deterministic algorithm with probe types generated from Internet traces. The * markers are the RAM algorithm. The dashed line is the deterministic algorithm with uniform random probe type generation.

outside of the scope of our paper, we will examine the associated security concerns. The primary concern is that a malicious receiver stands to receive a disproportionate share of a congested link's capacity if it conceals the presence of congestion from the sender [18]. Under our proposed algorithm, the receiver forms an estimate of the path price based on the received probes. Using our example of 4-bit quantization and a maximum of 30 links, the estimated path price will fall between 0 and 30 and can be represented – to the full precision afforded by the chosen quantization – using only 9 bits. In the simplest possible model of a feedback mechanism, the receiver periodically sends the 9-bit estimate to the sender (possibly embedded in TCP acknowledgment packets). This approach generates the smallest amount of traffic along the feedback path, but provides no security. It is trivial for a receiver to send any arbitrary path price estimate it wishes, and the receiver has no way to detect any possible deceit.

We now present a modification to our algorithm which guards against a malicious receiver falsifying its path estimate. Rather than always initializing the ECN field to 01, the sender chooses, while setting up the connection, a random ECN initialization for all probes corresponding to each bit-position. For every data packet subsequently sent, the sender determines the probe type so that it knows what bit position the probe is encoding. Based on this information, it initializes the ECN field to the (now) fixed value it randomly chose during setup. Routers increment the field as before, but may now also "wrap-around" from 11 to 01. The receiver is not aware of the b initial ECN fields, rendering it incapable of calculating the estimated path price itself. For each of the b categories of probe types, the receiver keeps a tally of how many of the probe types have their ECN field set to each of the 3 possible codepoints. Once it has seen every probe type, the sender provides its tally as feedback to the sender. This is enough information for the sender to calculate the path price estimate. For example, if the sender set the initial ECN field of probe types encoding MSB sums to 11, and the receiver informs it that of the 15 MSB probe types it received five were 11, four were 01, and six were 10, the sender would determine that five probes encoded a bit-sum of zero, four encoded a one, and six encoded a two. Thus it would deduce that 16 of the 30 MSBs were 1. With this approach, the total amount of data the receiver has to transmit to the sender under our standard assumptions is 32 bits: for each of the four bit-positions of the quantized link price, a value of 0 to 15 for the number of probe types with an ECN value of 01 (requiring 4 bits to represent), and a value from 0 to 15 for the number of probe types with an ECN value of 10 (again, requiring 4 bits). We note that the number of probe types taking on the remaining possible value of 11 is uniquely determined by the other two values, and hence does not have to be explicitly encoded.

With this approach, we achieve a level of security because the receiver does not know the mapping between codepoints and bit-sums for any of the probes. However, due to the fact that the codepoint initialization remains constant for a subsets of the probes, the receiver could conceivably keep a long term count of how many of the three codepoints it observes for each of the sets and attempt to deduce the codepoint initialization. It would take considerable effort to configure a receiver to carry out this attack and the results achieved may not be reliable.

VIII. Conclusion

We have specified a novel deterministic packet marking algorithm that allows a host to deduce the sum of the prices over the links traversed to a client. By reading a packet's `IPid` field to uniquely identify the probe type, and the TTL field to identify the LinkID, every router is able to determine whether it should modify the ECN fields based on the quantized price of its outgoing link. Based on empirical data, the sequential manner in which the majority of Internet hosts increment the `IPid` of transmitted packets is conducive to observing all probe types in a relatively short block of packets with high probability. This is vital to the performance of our algorithm, because the chief source of error in estimating the path price based is failing to observe one or more probe types in an estimation block. Quantization, the other, generally less critical source of error is independent of the block length.

Our results indicate that our algorithm performs better than RAM up to certain block lengths in estimating static prices. Since the levels of congestion in networks tend to vary dynamically, the most significant feature of our deterministic algorithm is its performance in estimating time-varying prices. In all the scenarios considered, our algorithm exhibits a lower mean-squared error than RAM and has a greater proportion of estimates falling within any given error bound up to 30%. The improvement over RAM is especially pronounced when using Internet trace-based `IPid` behaviour.

References

[1] S. Athuraliya and S.H. Low, "Optimization flow control II: Implementation," Tech. Rep., Netlab, California Institute of technology, 2000.

[2] S. Athuraliya, V.H. Li, S.H. Low, and Q. Yin, "REM: Active queue management," *IEEE Network*, vol. 15, pp. 48–53, May 2001.

[3] R.J. Gibbens and F.P. Kelly, "Resource pricing and the evolution of congestion control," *Automatica*, vol. 35, pp. 1969–1985, 1999.

[4] D. Katabi, M. Handley, and C. Rohrs, "Internet congestion control for future high bandwidth-delay product environments," in *Proc. ACM Sigcomm 2002*, Pittsburgh, PA, Aug. 2002.

[5] S. Kunniyur and R. Srikant, "A time scale decomposition approach to adaptive ECN marking," in *Proc. IEEE INFOCOM*, Anchorage, AL, 2001, pp. 1330–1339.

[6] S.H. Low and D.E. Lapsley, "Optimization flow control I: Basic algorithm and convergence," *IEEE/ACM Trans. Networking*, vol. 7, pp. 861–875, Dec. 1999.

[7] F. Paganini, Z. Wang, S.H. Low, and J.C. Doyle, "A new TCP/AQM for stable operation in fast networks," in *Proc. IEEE INFOCOM*, San Francisco, CA, Apr. 2003.

[8] R.W. Thommes and M.J. Coates, "Determinstic packet marking for congestion price estimation," in *IEEE Infocom 2004*, March 2004.

[9] K. Ramakrishnan, S. Floyd, and D. Black, "The addition of explicit congestion notification (ECN) to IP," Sept. 2001, IETF RFC 3168.

[10] M. Adler, J-Y Cai, J.K. Shapiro, and D. Towsley, "Estimation of congestion price using probabilitic packet marking," in *Proc. IEEE INFOCOM*, San Francisco, CA, Apr. 2003.

[11] F. Begtasevic and P.V. Mieghen, "Measurements of the hop-count in the Internet," in *Proc. Passive and Active Measurement*, Amsterdam, The Netherlands, Apr. 2001.

[12] J. Postel, "Internet protocol," Sept. 1981, IETF RFC 791.

[13] S. Bellovin, "A technique for counting NATted hosts," in *Proc. Internet Measurement Workshop*, Marseille, France, Nov. 2002.

[14] R. Mahajan, N.T. Spring, and D. Wetherall, "Measuring ISP topologies with Rocketfuel," in *Proc. ACM Sigcomm 2002*, Pittsburgh, PA, Aug. 2002.

[15] N.L. Johnson and S. Kotz, *Urn Models and their applications*, John Wiley & Sons, 1977.

[16] V.F. Kolchin, B.A. Sevastyanov, and V.P. Chistyakov, *Random allocations*, John Wiley & Sons, 1978.

[17] A. Kamath, R. Motwani, K.V. Palem, and P.G. Spirakis, “Tail bounds for occupancy and the satisfiability threshold conjecture,” *Random Structures and Algorithms*, vol. 7, no. 1, pp. 59–80, 1995.

[18] D. Wetherall, D. Ely, N. Spring, S. Savage, and T. Anderson, “Robust congestion signaling,” in *IEEE Conference on Network Protocols*, November 2001, pp. 332–341.