

RICE UNIVERSITY

**Gleaning Network Wide Congestion Information from
Packet Markings**

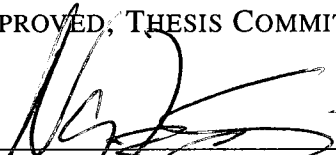
by

Florin Dinu

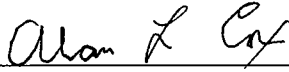
A THESIS SUBMITTED
IN PARTIAL FULFILLMENT OF THE
REQUIREMENTS FOR THE DEGREE

Master of Science

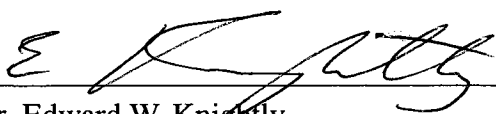
APPROVED, THESIS COMMITTEE:



Dr. T. S. Eugene Ng (Chair),
Assistant Professor,
Computer Science



Dr. Alan L. Cox,
Associate Professor,
Computer Science



Dr. Edward W. Knightly
Professor,
Electrical and Computer Engineering

HOUSTON, TEXAS

NOVEMBER, 2009

UMI Number: 1486051

All rights reserved

INFORMATION TO ALL USERS

The quality of this reproduction is dependent upon the quality of the copy submitted.

In the unlikely event that the author did not send a complete manuscript and there are missing pages, these will be noted. Also, if material had to be removed, a note will indicate the deletion.



UMI 1486051

Copyright 2010 by ProQuest LLC.

All rights reserved. This edition of the work is protected against unauthorized copying under Title 17, United States Code.



ProQuest LLC
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106-1346

Gleaning Network Wide Congestion Information from Packet Markings

Florin Dinu

Abstract

Congestion information can greatly benefit network level decisions. For example, fast-reroute algorithms should leverage congestion information when computing backup paths. They could also use the information to monitor if the re-routing decision itself causes congestion in the network. Today, most solutions for inferring congestion work at the end-host level and relay end-to-end congestion information to transport protocols. Network level decisions, on the other hand, may need link level congestion information. Unfortunately, the mechanisms that routers can use to infer link level congestion information are insufficient. Such information could potentially be obtained by periodically sharing estimates between routers. However, this solution increases the traffic load on the network and has difficulty in reliably delivering the estimates during periods of congestion.

In this thesis we show that routers inside an autonomous system can easily and accurately infer congestion information about each other. Routers first measure path level congestion information only from the congestion markings in the traffic that they forward. Next, we propose that routers combine routing information with the path level congestion information to obtain a more detailed description of the congestion in the network. Link level congestion information can be computed using this approach. Our techniques never add supplementary traffic into the network and use little router resources. They can be deployed incrementally or in heterogeneous environments. We show that the accuracy of the inference is good using experiments with multiple traffic patterns and various congestion levels.

Contents

Abstract	ii
List of Illustrations	v
1 Introduction	1
2 Overview	5
2.1 Prerequisites	5
2.2 AQM Algorithms and Congestion Marking	5
2.3 The ECN Standard	6
2.4 Overview of the Solution	7
2.5 From Marking Probability to Congestion Estimates	8
2.6 Discussion About the Assumptions	9
3 Inferring Path Level Congestion Estimates	11
3.1 Inference Using Data Packets	11
3.1.1 Implementation in Routers	13
3.2 Inference Using ACK Packets	14
3.2.1 Clean Slate Solution	14
3.2.2 Backward Compatible Solution	15
3.2.3 General Discussion About ACK Packet Analysis	25
4 From Path Level to Link Level Marking Probabilities	28
4.1 Computing Link Level Estimates	28
4.2 Applicability	30

5	Evaluation	33
5.1	Methodology	33
5.2	Sensitivity to Parameter Values	36
5.2.1	Backward Compatible: Sensitivity to Number of Monitored Flows .	36
5.2.2	Sensitivity to Ratio of the Estimation and Sampling Interval	37
5.2.3	Sensitivity to the Value of the Sampling Interval	38
5.3	Sensitivity to Network Environment Conditions	40
5.3.1	Congested Link Followed by Uncongested Links	40
5.3.2	Sensitivity to Different AQM Algorithms	41
5.3.3	Performance in Higher Bandwidth Environments	41
5.3.4	Sensitivity to Sudden Changes in the Congestion Pattern	42
5.4	Limitations	44
5.4.1	The Visibility Problem	44
6	Discussion	47
6.1	Limitations	47
6.2	Deployment Environment	47
6.3	Applications	48
6.4	Re-routing	49
6.5	Equal-Cost Multipath	49
6.6	Network Heterogeneity	50
6.7	The TCP Delayed ACK Algorithm	50
7	Related Work	51
8	Future Work and Conclusion	53
	Bibliography	54

Illustrations

2.1	The ECN protocol and message markings a) A congested router can set the CE bit in the ECN capable packets (i.e. ECT bit set) b) The TCP destination echoes the marking received into an ACK packet c) A TCP source set the CWR bit in the data packet as a result of receiving a marked ACK. d) A TCP destination receives 4 data packet out of which only the first is marked. However, the markings are echoed into all the ACKs until the data packet with the CWR bit set is received.	7
3.1	Computing path level estimates. Two packets arrive at R_i . The marking history already contains the bits 1,0,1,1,0. The markings from the packets are separately entered into the marking history. The history is shifted with every incoming packet and the percentage of 1s is computed. Finally, all percentages computed during a sampling interval are averaged to obtain a sample.	13
3.2	Inferred Marking Probability vs. Real Marking Probability and Window Size. EXP stands for experimental results. TH denotes results derived from the theoretical model.	19
3.3	Degree of Overestimation vs. Marking Probability	20
3.4	Computing average size of groups of unmarked packets	21
3.5	Groups of unmarked packets spanning multiple sampling intervals	23
4.1	A sample network	29

5.1	Chain topology and traffic pattern	34
5.2	Error vs number of flows monitored	37
5.3	Error vs ratio of estimation and sampling interval	38
5.4	Error vs hop nr for different values of the sampling interval	39
5.5	Error on uncongested links vs hop count	40
5.6	Absolute error for RED,PI,REM for different network hops	42
5.7	Aggregate path level marking probabilities	43
5.8	Performance in higher bandwidth environments	43
5.9	Network environment	45
5.10	The visibility problem	46

Chapter 1

Introduction

The success of the Internet is arguably based in a large part on the simplicity of the design of the networks. Networks were designed to simply carry the packets from a source to a destination. With time, as a result of the diversification of the services and applications that rely on the Internet, ideas were proposed that slowly increased the complexity of the network. Multicast, various queueing algorithms, label distribution protocols or complex filtering policies are only a few examples of the tasks that the routers perform today. We believe that this trend will continue, and, as more and more functionality is required from the network, routers will be faced with an increasing number of decisions that need to be taken automatically and efficiently. Such decisions are usually based on information about the state of the network. Gathering this information is a vital first step that routers must accomplish in order to make informed decisions.

The focus of this thesis is to provide a solution for routers to gather congestion information about other routers in the network. Network level decisions can benefit from congestion information. Fast-reroute (FRR) algorithms [1, 2, 3] are an example of a network level decision. They ensure connectivity even during the convergence periods caused by network failures. FRR algorithms compute and make use of backup paths. However, if a backup path proves to be congested then traffic can get dropped and the whole advantage of performing fast-reroute is lost. Conversely, a link can become congested as a result of the action taken by an FRR algorithm. If congestion information were available, then congestion-free backup paths could be computed and the congestion induced by re-routing could be discovered and addressed. Unfortunately, FRR algorithms are not currently concerned with congestion. We believe that one of the reasons is the absence of a reliable

method for providing accurate congestion information to them.

As another example, consider router data plane misbehavior detection mechanisms [4, 5, 6]. Such mechanisms have been proposed to monitor the trajectory of the packets in the network in order to detect forwarding anomalies caused by bugs or misbehavior. Usually, a group of monitoring routers collaborate to sample and analyze the incoming and outgoing traffic of one or more monitored routers. Packets could get dropped by the monitored routers and this could be a sign of misbehavior. However, a method is needed for the monitoring routers to distinguish between packet loss caused by legitimate network conditions such as congestion and packets loss caused by bugs.

Our solution provides a method for routers to estimate the congestion severity at other routers in the network. Both FRR algorithms and router misbehavior detection mechanism can leverage it to augment their decision making process with congestion information.

A potential method for sharing congestion information between routers today, is to have each router disseminate congestion information about its own links to other routers. Routers can do this periodically, or more efficiently, only when congestion is present on their links. Another option is to use a polling approach. Congestion information can then be sent only when another router requests it. Possible methods for delivering this information today include management protocol messages (e.g. SNMP) or traffic engineering extensions from link-state routing protocols [7, 8]. However, disseminating congestion information has limitations that greatly reduce the effectiveness. First, supplementary traffic is added into the network, and this can aggravate any existing congestion. If frequent updates are desired, then the overhead becomes significant. Second, packets carrying congestion information could get dropped because of existing congestion in the network. This impacts the timeliness of the delivery, as routers will have to wait for successful retransmissions.

In this work, we show that network-wide, link level congestion information can be accurately inferred and distributed among routers. Our solution is designed for networks that use congestion marking. The routers in these networks mark TCP data packets according

to some definition of congestion. TCP receivers then echo these markings back in the ACK packets, so that TCP sources can adapt their sending rates. ECN is an example of congestion marking that has been standardized for use in IP networks [9]. In our solution routers passively monitor the congestion markings in the ACK or data packets received from other routers and use this information to obtain path level congestion estimates. Using both link state routing information and the path level estimates routers then proceed to compute congestion information for increasingly shorter paths. We analyze the extent to which routers can infer estimates for shorter paths and show that link level estimates can be computed for a large number of networks today (e.g. backbone networks).

Today, the process of congestion marking is designed with the TCP end-hosts in mind. The goal is to let TCP sources react to network congestion fast and without unnecessary drops in performance. The focus on end-hosts imposes certain limits on the practicability of using the congestion markings for network wide congestion inference. For example, with ECN, a TCP receiver does not keep the sequence and number of markings it receives in data packets intact when transferring the markings to the ACK packets. It can actually mark a sequence of consecutive ACK packets as a result of a marking in a single data packet. While this process ensures reliable delivery of the markings to the TCP source it also deteriorates the useful congestion information. We present two solutions to deal with this problem. The first solution is a clean slate solution. It proposes an alternative echoing algorithm that marks an ACK packet if and only if the corresponding data packet was marked. With this modification, the congestion information carried by ACK packets exactly mirrors the information carried in the corresponding data packets. We present mechanisms to accurately infer congestion estimates by analyzing the number of congestion markings in both ACK and data packets. We discuss two methods of implementing this clean slate solution and argue that the influence on TCP behavior is minimal. We then return to the standard ECN marking algorithm and propose modifications to the clean slate solution that allows the congestion information to be recovered despite the deterioration caused by TCP receivers. This backward compatible solution results from a theoretical

model that we developed to quantify the degree of deterioration when echoing markings from data packets to ACK packets. We validate the theoretical model with experimental results. The solution relies on measuring the sequences of unmarked ACK packets, to infer congestion estimates.

Our solutions are lightweight and never add additional traffic into the network. The first solution uses only simple packet counters. The second solution necessitates more state in routers but uses only a constant amount irrespective of the increase in the number of flows, packets or the bandwidth in the network. We believe our solutions can be readily implemented in routers today. They can be incrementally deployed and are also suitable for heterogeneous network environments. Our evaluation on significantly congested network environments shows that the approaches can infer the congestion level of a link multiple hops away with good accuracy. Moreover, the accuracy of the solution significantly improves with an increase in bandwidth.

The contributions of this thesis are as follows:

- We show that routers can passively compute congestion information about paths in the network using the congestion markings from existing traffic. Our solution leverages both data and ACK packets. We present a method that uses the markings in the ACK packets even though the standard ECN protocol does not faithfully echo congestion markings from data packets into ACK packets.
- The solution uses few resources and therefore it is suitable for routers.
- We propose that routers combine congestion information with link state routing information to allow additional congestion estimates to be computed. We argue that link level congestion estimates can be computed using this approach.

Chapter 2

Overview

2.1 Prerequisites

In this thesis we focus on inferring network-wide congestion information for a single autonomous system (AS) using the traffic local to that AS. We assume that routers use AQM algorithms and that congestion marking is at least partially deployed in both routers and end-hosts. We further assume that routing in the AS is performed using a link-state protocol [10, 11] and that shortest-path routing is employed. We consider that the TCP delayed ACK algorithm [12] is not used, so TCP receivers separately acknowledge every data packet received. We use the terms data or forward path to refer to the path taken by data packets. Similarly, we use ACK or reverse path to refer to the path taken by ACK packets.

2.2 AQM Algorithms and Congestion Marking

An Active Queue Management (AQM) enabled router can be augmented with the ability to mark packets instead of dropping them. A TCP source then reacts to the packet markings the same way it would react to a packet drop. The overall result is the same, the TCP source reduces its sending rate, however, it does not have to pay the penalty of retransmitting data. The AQM algorithm [13, 14, 15] at a router computes a congestion estimate for the router's outgoing links. This estimate can be as simple as a function of the size of the router queue in RED's case, or more a complex expression that takes into consideration the incoming traffic rate and the available bandwidth for an algorithm like REM. The router then marks outgoing data packets probabilistically, based on the congestion estimate for the link they are sent on. The TCP receiver echoes back the markings to the TCP source in the ACK

packets. Note that, the TCP sender only needs to be informed about the congestion on the data path. Therefore, by construction, packet markings on both the data and ACK packets only carry information about the congestion on the data path. While a TCP connection is typically bidirectional, the markings on the two halves of the connection are independent because each TCP source can potentially send data through different paths with different congestion levels. Throughout this thesis we also consider a TCP connection as being composed of two standalone halves.

2.3 The ECN Standard

The Explicit Congestion Mechanism (ECN) standard [9] describes a packet marking scheme and the bits reserved in the packet headers for congestion marking. ECN is the standard for congestion marking in IP networks, today. ECN makes use of four bits in the packet header: two bits in the TCP header, and two in the IP header (bits 6 and 7 of the ToS byte). The TCP header bits are named ECN-Echo (ECE) and Congestion Window Reduced (CWR), while the IP header bits are the ECT (ECN-Capable Transport) and the CE (Congestion Experienced) bits. The two TCP header bits are used for communication between the TCP end-points while the IP header bits allow for communication between end-points and routers. If the ECT bit is set to 0 the data packet is considered non ECN capable and will be treated like any regular TCP packet. If ECT is set to 1 a router may decide to mark the packet. It does this by setting the CE bit. Therefore, a data packet with both the ECT and CE bits set signifies that congestion has been encountered somewhere on the path the packet took. When a TCP destination receives a packet with the CE bit set it sets the ECE bit in the subsequent ACKs. It stops marking ACKs only when it receives a data packet with the CWR bit set. The CWR bit is set by a TCP source to signal a reduction in the size of the congestion window. This can happen as a result of receiving an ACK with the ECE bit set or for other reasons. We call a packet with the ECE bit set a *marked ACK packet* and a packet with the CE bit set a *marked data packet*. ECN was initially proposed only for data packets, however extensions like ECN+ [16] have been designed for the SYN/ACK pack-

ets exchanged during the initial TCP handshake. For our solution, it makes little difference which variant is used.

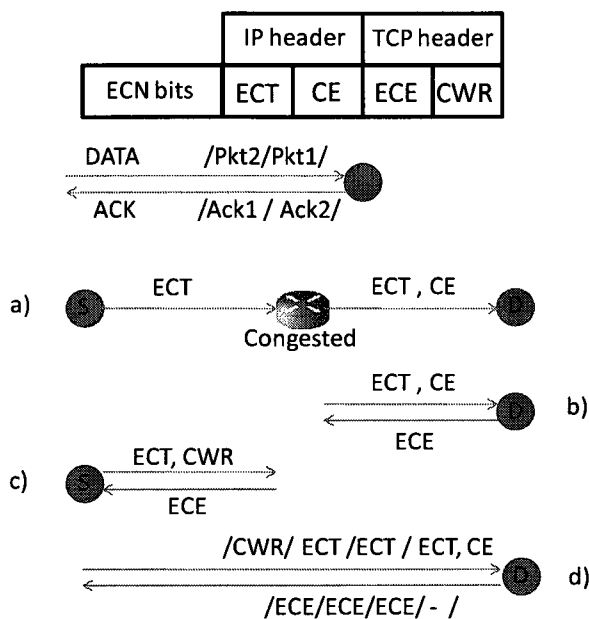


Figure 2.1 : The ECN protocol and message markings a) A congested router can set the CE bit in the ECN capable packets (i.e. ECT bit set) b) The TCP destination echoes the marking received into an ACK packet c) A TCP source sets the CWR bit in the data packet as a result of receiving a marked ACK. d) A TCP destination receives 4 data packets out of which only the first is marked. However, the markings are echoed into all the ACKs until the data packet with the CWR bit set is received.

2.4 Overview of the Solution

The solution described in this thesis allows a router to infer the congestion level of other routers in the network. First, routers analyze the packet markings in the data and the ACK packets that they receive. A group of packets traversing the same network path carries information about the aggregate marking probability on that path. By separately analyzing packets that traverse the same network path, routers can compute aggregate path level marking probabilities. We show how this computation can be directly performed by leveraging a new congestion marking algorithm. We also propose a more complex approach

that works with the standard ECN marking algorithm. After computing path level marking probabilities a router can leverage routing information to take advantage of common segments between paths. Using this computation, routers can compute aggregate marking probabilities for paths that are increasingly shorter. Marking probabilities for one link paths can be computed with this approach. These are exactly the link level marking probabilities. Since congestion marking is performed based on the congestion estimate, a router that has calculated a link level marking probability can obtain the corresponding congestion estimate. In this thesis we only briefly describe how the conversion between marking probabilities and congestion estimates can be performed. Our focus is on providing ways for routers to obtain accurate path level marking probabilities and then compute link level marking probabilities.

2.5 From Marking Probability to Congestion Estimates

Once a router has inferred the marking probability on some link it can convert this information into congestion estimates. To do this it needs to know the type of AQM used by the router that sends traffic on that link. Also, it needs to know the values of the AQM parameters. This is because each AQM algorithm translates congestion estimates into marking probability using a different function. Since the inference is performed within the boundaries of a single AS such information can be easily and safely distributed to routers. The type of the conversion between the marking probability and the congestion estimates can differ based on the use of the congestion estimates. Some applications may want precise congestion estimates while others may require just a coarse description of the levels of congestion (e.g. high, low, medium). The accuracy of the conversion also differs from one AQM algorithm to another. A linear marking function like RED's is easier to inverse, while an exponential marking function like REM's may lead to more pronounced errors as small differences in the congestion level are mapped to larger differences in the marking probability. While this effect may be detrimental when a precise conversion is desired, it will be far less pronounced when a conversion to coarse levels of congestion is desired. Some

AQM algorithm also use additional information when computing the marking probability as a function of the congestion severity. For example, RED uses a variable named *count* which numbers the packets since the last marked packet. This variable is very difficult for other routers to infer since it is based only on local information.

2.6 Discussion About the Assumptions

Our algorithm is designed to infer congestion for a single AS. However, networks today rarely function in isolation from each other. Most networks also receive traffic from other AS-es. Part of this traffic may already be marked, before entering the AS which is running the congestion inference algorithm we described. Also, if the AS is not the destination of the traffic then the markings received on the reverse path may contain additional markings applied after the data packets left the AS but before they reached the destination. In the cases described, the percentage of marked packets contains information about more than the congestion in the current AS. Routers cannot separate the information related to the current AS by observing the markings in the packets. Also, it is difficult to determine, only by analyzing ACK packets if the corresponding data packets were actually routed through the same AS. Since each AS is a standalone network, routers do not have enough information to apply the algorithm to routers in other ASes. Therefore, this inter-domain traffic should not be used for the inference algorithm. Fortunately, separating the inter-domain from intra-domain traffic can be easily done by checking if the source and destination addresses in a packet belong to the local AS.

Universal deployment of AQM and congestion marking algorithms is not necessary for our approach to be effective. However, if more hosts and routers supporting this scheme are deployed into the network, the amount of traffic that can convey congestion information will increase. This translates into more traffic that can be used by the inference algorithm. Our solution can be incrementally deployed. The only condition is that the position of non-compliant routers is known by all other routers in the network. Their presence can be factored out by the algorithm because their marking rate is effectively zero. Non-compliant

routers may also choose to drop packets as a signal of congestion. We will describe in a later section the effect that dropped packets have on our approach.

Chapter 3

Inferring Path Level Congestion Estimates

Routers can obtain path level marking probabilities from the analysis of packets traversing a common network path. Retrieving accurate path level probabilities however, is not as simple as computing the percentage of marked packets observed. Today, the main difficulty lies with the marking algorithm used by ECN. As discussed, once a marked data packet is received, the TCP destination marks the corresponding ACK packet and all subsequent ACK packets until an explicit message announcing a reduction in the congestion window is sent by the TCP source. The rationale behind this design choice is to ensure the reliable delivery of the echoed markings to the TCP source even if some ACK packets are dropped. However, this marking scheme makes the analysis of ACK packets for congestion inference difficult since the congestion information contained in the data packets is being altered at the TCP receiver when echoed to the ACK packets. In this chapter, we first propose a solution for inferring accurate congestion information from the data packets. We then address the challenge imposed by the analysis of the ACK packets and present two methods for inferring congestion information using ACK markings.

3.1 Inference Using Data Packets

To be useful, estimates of the marking probability need to be computed for a past period of time. The length of this period depends of the final use of the estimates. For example, a re-routing algorithm would be typically interested in a description of congestion on the scale of minutes since very short congestion episodes can be transitory. We call the interval of time over which a marking probability estimate is measured an *estimation interval*. After an estimation interval elapses, a single value is computed. This is the estimate for the

marking probability for the entire estimation interval and is simply called *the estimate*.

However, the number of packets used for analysis can vary between estimation intervals. Moreover, even during one estimation interval, there could appear significant variations in the number of packets that a router receives. As the length of the estimation interval increases, the chances of encountering such variations also increases. One cause for these variations is the burstiness inherent in the use of TCP. However, we desire our estimation interval to be an unbiased estimate of the marking probability on a path. Simply counting all the markings received in an estimation interval creates a bias towards stretches of time when bursts of packets are received. For this reason we introduce a parameter called the *sampling interval*. A sampling interval is an exact subdivision of an estimation interval. Only one value of the marking probability is computed for a sampling interval irrespective of the number of packets received. We call this value *a sample*. This technique can remove any bias because every estimate will only be computed by averaging over a fixed number of values, equal to the ratio between the estimation interval and the sampling interval. There is no universally good value for the sampling interval. As a guideline, a value on the order of the round-trip time (RTT) should provide enough packets for analysis. It is still possible that, no packets are received during one sampling interval. In this case, the value of the previous sampling interval is used.

During one sampling interval the percentage of markings received is computed using a bit vector of congestion markings called the *marking history*. Each router maintains a marking history for each different path it measures the marking probability on. The marking history is a bit vector of the last markings encountered in the packets traversing the same network path. The length of the marking history is configurable. With every incoming packet, the marking history is shifted by one bit, and a value of the percentage of bits set to 1 in the bit vector is computed. During a sampling interval a large number of packets are processed and for each packet a new value is obtained as described above. The final value computed for the sampling interval is the average of all the values obtained during it. Figure 3.1 shows a graphical representation of the entire computation.

The reason why a marking history is used instead of just computing the percentage of marked packets received during a sampling interval is to allow meaningful values to be computed when only a very small number of packets are received. Imagine receiving only two packets during a sampling interval. This small number of markings is definitely not enough to help compute a meaningful sample. However, taking into account the history of markings seen by a router in the previous packets allows for a much better assessment of the latest marking probability. To capture the markings from previous packets the marking history can span multiple sampling intervals. In a favorable case, where plenty of samples are received every sampling interval, using the marking history or directly computing the percentage of marked packets are expected to perform similarly.

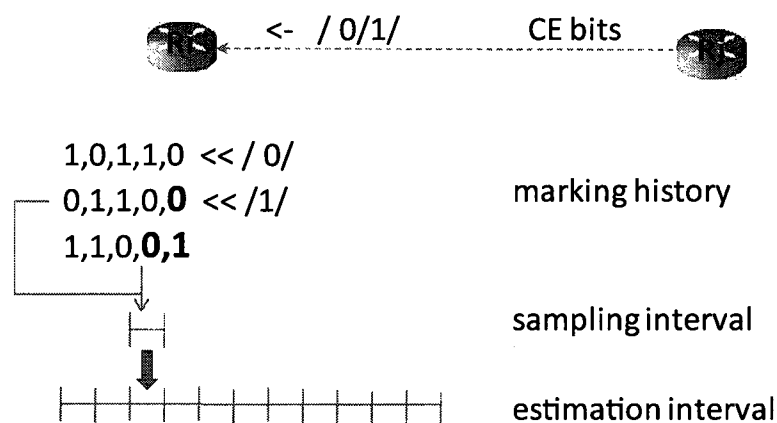


Figure 3.1 : **Computing path level estimates.** Two packets arrive at R_i . The marking history already contains the bits 1,0,1,1,0. The markings from the packets are separately entered into the marking history. The history is shifted with every incoming packet and the percentage of 1s is computed. Finally, all percentages computed during a sampling interval are averaged to obtain a sample.

3.1.1 Implementation in Routers

For current routers to support our solutions a number of changes need to be implemented. Routing today is mostly destination based. A router forwards a packet to the destination without information about where the packet came from and how it arrived there. Infor-

mation about the source of the packet is needed for our approach in order to know where a path starts. A data structure similar to a forwarding table can be used here. Instead of the destination network and a next hop it can contain the source network and the first hop taken by packets sent from that network to the router. All this information can be derived from the link state advertisements disseminated by the link state routing protocol. A second change involves the computation and storage of routes between sources and destinations. Normally a router only needs to compute the routes between itself and other routers. For our approach routers also need to compute and store the routes between all routers in the network.

For the computation of the path level estimates the implementation in routers is straightforward. For each path it monitors a router needs to maintain a bit vector with the markings in the previous packets, and two counters for computing the average. With each incoming packet a shift of the bit vector needs to be performed along with the computation of the percentage of bits that are set.

3.2 Inference Using ACK Packets

3.2.1 Clean Slate Solution

For the analysis of ACK packets, we first address the loss of congestion information when echoing markings from data packets to ACK packets by removing the cause of the problem. We propose to replace the ECN marking algorithm with an alternative echoing scheme that has TCP receivers mark an ACK packet if and only if the corresponding data packet was marked. With this modification a direct method for computing the marking probability can be developed.

We envision two methods for deploying this clean slate echoing scheme. A first method requires an additional bit in the packet headers. This ensures compatibility with existing uses of the ECN bits. The net result is that the TCP behavior remains exactly the same. This is the approach that we use for implementing and evaluating our approach. A second

method directly modifies the way ECN bits are set. This method has the benefit of not needing any extra header space. However, it cannot ensure the reliable delivery of the markings in the situation where ACK packets get dropped in the network.

Since this solution requires some changes to the protocols we call it the clean slate solution. With this solution, the method proposed for the analysis of data packets can be directly applied to ACK packets because the markings in the ACK packets exactly mirror the markings in the data packets.

3.2.2 Backward Compatible Solution

The main concern regarding the clean slate solution is that it requires changes to the ECN protocol. Recall that the changes proposed were necessary because the congestion information from the data packets was altered when echoed to the ACK packets. More than one ACK packet could be marked as a result of just one marked data packet. As a result, computing the percentage of marked ACK packets would overestimate the percentage of marked data packets. This overestimation would produce inaccurate path level marking probability estimates. In this subsection we focus on providing a backward compatible solution for inferring path level marking probabilities from ACK markings. This solution works with the standard ECN marking algorithm. This solution differs from the clean slate solution in the method used to collect samples. We present a method to retrieve marking probability samples despite the overestimation inherent in the marking scheme.

3.2.2.1 Using Only Small Flows

In this subsection we present a first attempt at a backward compatible solution. For this, note that the degree of overestimation when analyzing ACK packets marked by the standard ECN algorithm can depend on the value of the congestion window at the TCP source. If this value is 1, then the TCP source sends a new packet only after it receives the ACK for the previous data packet. If this ACK was marked, then the TCP source is bound by the ECN algorithm to mark the CWR bit in the next data packet it sends. As a result, the TCP

destination only marks a single ACK as a result of receiving a marked data packet, because the very next data packet it receives will request a stop to the marking by having the CWR bit set. This situation is not limited to flows having a congestion window of 1. Any flow that needs to acknowledge every packet before sending a new one will have an ACK packet marked if and only if the corresponding data packet was marked. A related category are very small flows, in which just one exchange of packets is performed over the duration of the flow. In all these scenarios the percentage of marked ACK packets will be a good estimate of the percentage of marked data packets.

The challenge is in selecting the flows with the properties described above. The routers will have to select only such flows for monitoring and they have to infer the needed properties just by analyzing the protocol headers. Studies have shown that today a very large number of flows are actually small flows. Traces of traffic sent from web servers to clients on the UNC network are analyzed in [17]. Even though persistent HTTP connections are used, more than 50% of the HTTP responses are smaller than a 1500 bytes. Even more striking is the distribution of HTTP request data sizes. More than 90% of HTTP requests are smaller than 1000 bytes. The number of exchanges performed in each HTTP flow is 1 for more than 60% of the servers studied. While this study analyzed only HTTP traffic, other studies [18, 19] show that the percentage of HTTP traffic on ISP backbone links can easily exceed 40%. It is therefore likely that there are enough small flows to be used for congestion inference.

While using small flows to drive the congestion inference process and then simply counting the percentage of marked ACK packets seems promising, this is not the avenue we pursue because the precise selection of such flows is difficult. Moreover, small flows may not be present in the same quantity on every network path. Instead, our focus is a general solution that works with any size of TCP flows. The following subsections address this concern and propose a more general, backward compatible solution.

3.2.2.2 Characterization of the Overestimation

Understanding our backward compatible solution requires a characterization of the overestimation. We first present a theoretical model that quantitatively describes the overestimation and then support our findings with experimental results. The solution will result directly from the theoretical model. Recall that the severity of the overestimation depends on the congestion window of the flows. The larger the window, the bigger the number of ACK packets marked as a result of a data packet being marked. The overestimation also depends on the real marking probability since more data packets in a window are marked when the marking probability is higher. In this case, the overestimation is lower compared to when only one data packet is marked in a window. The focus of our theoretical model is to compute a function that maps the inferred marking probability when analyzing ACKs to the real marking probability and the window size of the flows.

To start, let w be the window size for all flows (e.g. w packets are sent every RTT), let p be the real marking probability and I be the inferred marking probability. Suppose that w and p are fixed and that a large number of randomly started flows are monitored. Since the window size is fixed, there will always be w consecutively marked ACK packets. In our model we consider a large number of groups of w marked packets along with the following group of unmarked packets. Simply computing I as the percentage of marked ACKs will result in the formula:

$$I = \frac{w}{w + q} \quad (3.1)$$

where q is the average size of the groups of unmarked packets that come between groups of marked ACK packets. For computing q we use the following identity:

$$q = \sum_{n=0}^{\infty} n * (1 - p)^n * p \quad (3.2)$$

This identity computes the probability of groups of n unmarked packets to appear in the flow, for any value of n . Solving this identity gives a value of

$$q = \frac{1 - p}{p} \quad (3.3)$$

. Performing a simple substitution in 3.1 we obtain the final formula that characterizes the overestimation.

$$I = \frac{w}{w + \frac{1-p}{p}} \quad (3.4)$$

We designed experiments to test the validity of our findings. We modified the TCP sources in the ns-2 simulator and disabled the congestion control algorithm. Every time these sources send packets they send a window's worth of packets. We use a simple network of just one link and vary the real marking probability on that link by changing the number of TCP flows started. We then attempt to infer the marking probability by simply applying our clean slate solution. Because the default ECN marking algorithm was used, applying the averaging method of the clean slate solution yields the overestimate we want to characterize. We use a sampling interval of 0.2 seconds and an estimation interval of 30 seconds. We especially designed our experiment without congestion control because in this scenario the real marking probability on the link will remain roughly constant from one estimation interval to another. Since all the flows send with the same window size, and they are started at random intervals, even during one estimation interval the real marking probability remains roughly constant. Since in the experiment the window size and the real marking probability are kept roughly constant a comparison with the theoretical model is possible.

Figure 3.2 plots both the theoretical and experimental results for different values of the congestion window and the real marking probability. Notice that the experimental results support our theoretical findings.

Figure 3.3 plots the number of times the real marking probability is overestimated for different values of the real marking probability and the window size. For a window value of 8 packets and a real marking probability of 15% the inferred marking probability overestimates the correct result by almost a factor of 4. This result shows that simply computing the percentage of marked ACK packets is a very inaccurate solution.

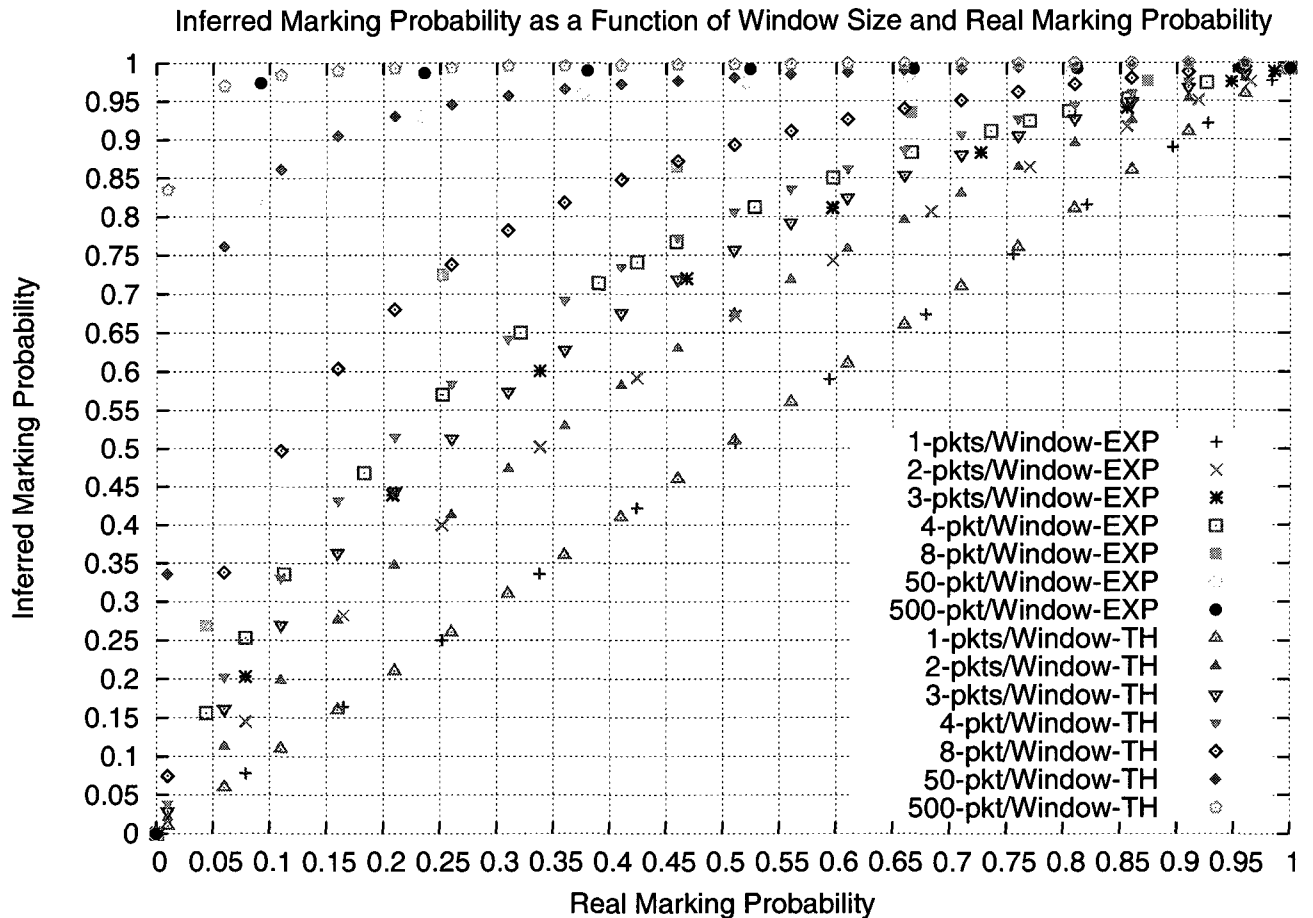


Figure 3.2 : Inferred Marking Probability vs. Real Marking Probability and Window Size. EXP stands for experimental results. TH denotes results derived from the theoretical model.

3.2.2.3 The Backward Compatible Solution for Inference Using ACK Markings

Equations 3.3 and 3.4 provide us with two methods for obtaining the real marking probability p .

In equation 3.4 since the inferred marking probability I can be measured, all it remains is to obtain the value of w . However, the value of the congestion window is not present in the TCP header. It is also not constant, since TCP continuously adapts the value of the congestion window to the network load. One solution could be to measure the RTT between routers and count the number of packets received for the flows during one sampling interval.

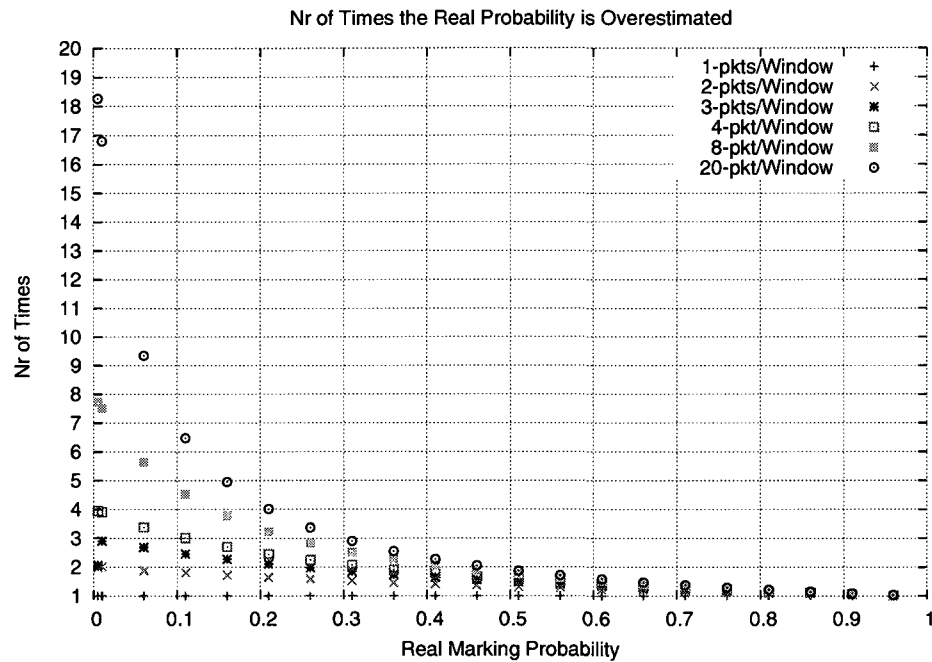


Figure 3.3 : Degree of Overestimation vs. Marking Probability

This can yield an estimate of the average congestion window value for a flow during one sampling interval. However, this approach is very sensitive to measurement error. The RTT between routers cannot be perfectly computed because of queueing delays. Even more, the RTT contains a component dependent on the end-host that cannot be inferred by routers.

In equation 3.3 recall that q is the average size of the groups of unmarked ACKs. The value of q does not depend on the flow window size but rather it is only a function of the real marking probability. Given enough flows, an average of the sizes of the groups of unmarked ACKs can be computed and then p can trivially be derived. To use this idea routers select a number of flows to monitor during each sampling interval. Different flows can be used for different sampling intervals, however, benefits can be gained from continuously monitoring the same set of flows. Routers then compute the number of unmarked ACKs and the number of groups of unmarked ACKs over all the flows. The process is depicted in Figure3.4.

A group of marked ACKs begins with the first marked ACK and ends with the ACK corresponding to a packet that had the CWR bit set. A group of unmarked ACKs appears

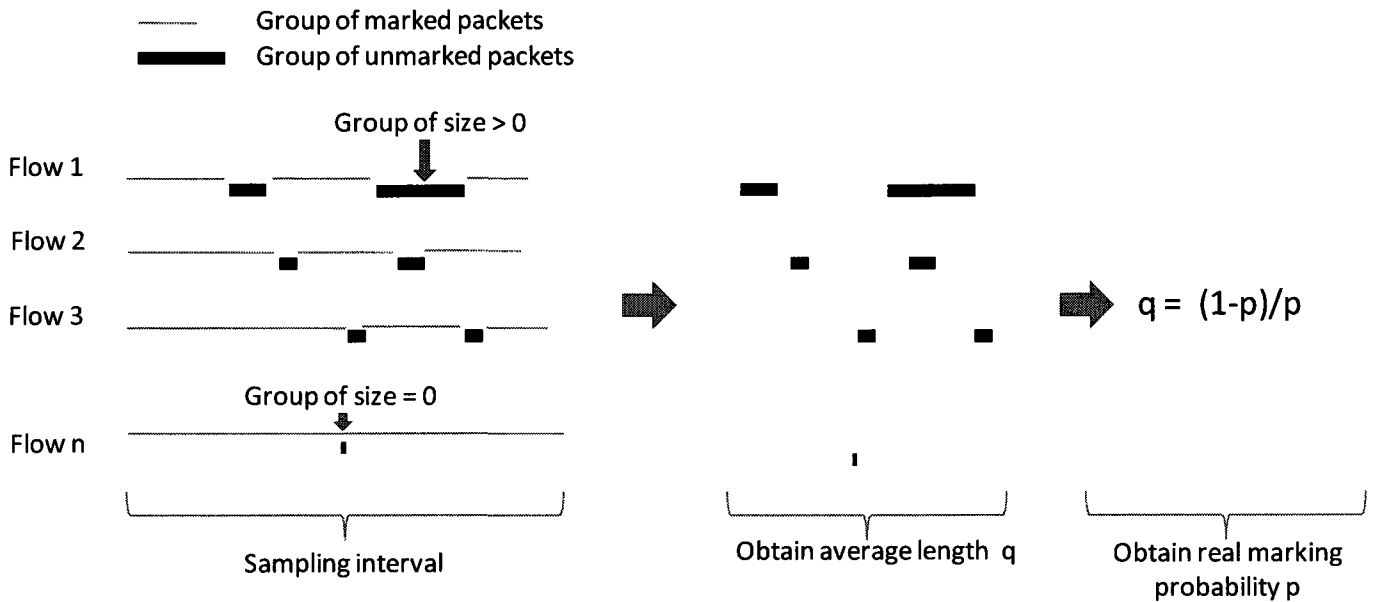


Figure 3.4 : Computing average size of groups of unmarked packets

between two consecutive groups of marked ACKs. The length of a group of unmarked ACKs can be either 0 (a group of marked ACKs starts immediately after another) or any value larger than 0. Note that a router needs to infer which is the ACK packets corresponding to a marked data packet. To make this correlation a router needs to observe both the data and the ACK packets for a flow. It needs to remember the sequence number of the data packets with the CWR bit set and then infer which is the corresponding ACK packet. This can be done using the sequence numbers. Let seq be the sequence number of a marked data packet. The ACK packet that acknowledges it is the one that has in its TCP header a value of the sequence number larger than seq .

3.2.2.4 Discussion About the Backward Compatible Solution

Note that our explanation does not take into account dropped ACK or data packets. For our backward compatible solution to correctly compute an estimate, the sequences of marked and unmarked ACKs that a router observes should be preserved the same as when leaving

the TCP destination. Dropped packets can alter this sequencing. We assume that in an AQM/ECN network the loss of ECN capable packet is a rare event. Since TCP sources should react to packet markings by reducing their sending rate, packet loss of ECN capable packets should mostly happen as a result of misbehaving TCP flows or high rate UDP flows. Both scenarios are undesirable for networks and are usually signs of attacks. However, not all packets in a flow are ECN capable. While ACK packets piggybacked on data packets are ECN capable, pure ACK packets are not. Instead of marking a pure ACK, a standard AQM/ECN router will drop the ACK during congestion. Since the backward compatible solution uses the sequences of ACK packets for computation, ACK losses can have an impact of the accuracy of the solution.

Reordered packets are another factor that can alter the sequences of marked and un-marked ACK packets. The undesirable effects of packet reordering on TCP performance are well known [20]. Therefore, packet reordering is not desired in practice and measures are taken to minimize the probability of appearance (e.g. ECMP algorithms use a single path to route one flow).

To use the backward compatible solution routers need to monitor both the ACK packets and the data packets for a flow. This does not imply routing in the network should be symmetric. Assuming that the first hop router for an end-hosts is unique, it can always perform this function.

Routers need to remember the sequence number of data packet that have the CWR bit set in order to match them with their corresponding ACKs. Once the matching is done the sequence number remembered can be removed. CWR bits are sent not only as an answer to marked ACK packets. They are sent every time the congestion window is reduced. Other possible reasons for the TCP source to send CWR marked data packets are restarting the transmission after an idle period or timeouts. Therefore it is entirely possible that multiple CWR marked data packets arrive at a router before any of the corresponding ACKs. Our simulations confirm this behavior. A router needs to be able to remember all such CWR marked packets. However, the number of CWR marked data packets received by a router

without any corresponding ACKs being received is limited since each CWR packet describes a reduction of the congestion window. When dealing with multiple CWR marked packets care must be taken when computing the sizes of the groups of unmarked ACKs. In this case, not all CWR marked packets signal the end of a group of marked ACKs.

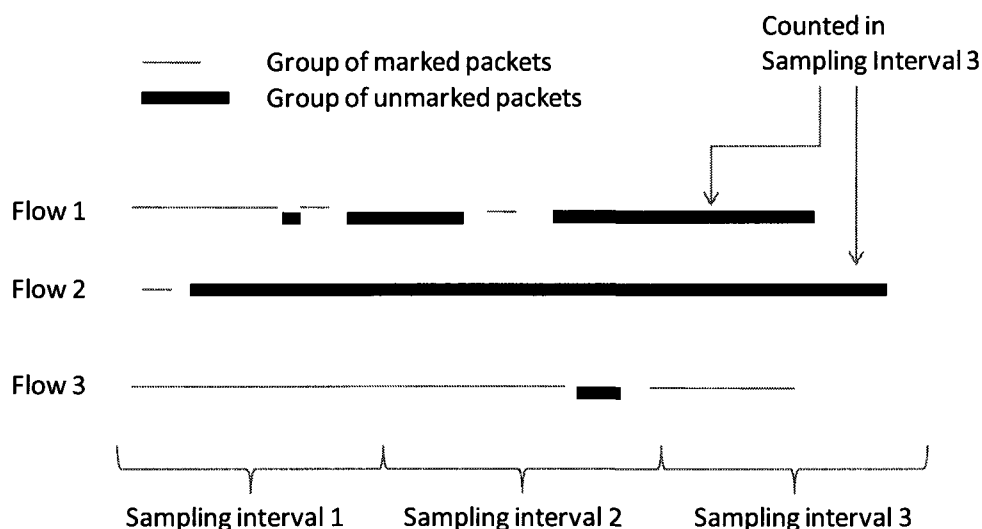


Figure 3.5 : Groups of unmarked packets spanning multiple sampling intervals

Our solution can use flows of any size. However, flows that send more packets are more useful for the inference because they provide more data points and have an increased chance of producing groups of unmarked packets of various sizes. Monitoring large flows is easier than monitoring flows that perform exactly only one exchange of packets. One reason is that there is no requirement that all monitored flows should be large flows. If a significant percentage of the monitored flows are large this should suffice even though some small flows might also be used for the inference. In recent years, peer-to-peer (P2P) applications have substantially grown in popularity. Studies estimate that the percentage of P2P traffic on network links can easily exceed 40% [19]. P2P traffic is mostly used for large data transfers such as video files. These transfers are long TCP flows. Therefore, in real networks, there is a good probability that enough long flows are available for the inference process.

Groups of unmarked ACKs can span sampling intervals. A graphic representation of this case is presented in Figure 3.5. We propose counting the groups of unmarked packets in the sampling interval where the groups end. However, for this, a router needs to monitor a specific flow for consecutive sampling intervals. For very low marking probabilities groups of unmarked ACKs can span multiple sampling intervals. Measuring those large groups presents an interesting trade-off. If they are counted, then a sampling interval can use potentially stale congestion information from previous sampling intervals. If they are not accounted for, then the value of q from 3.1 is underestimated and this translates into an overestimation of the marking probability. Therefore, our approach uses as a parameter the number of sampling intervals that a group of unmarked ACKs is allowed to span in order to be counted. As explained routers need to monitor a number of flows. This should not be interpreted as per-flow state since only a small, constant number of flows suffices for the purpose of the inference.

If a router wants to monitor some flows for consecutive sampling intervals it needs to remember information about them across intervals. The capacity of a router to continuously monitor flows depends on the number of flows it wants to monitor to compute samples and also the maximum number of flows it can monitor. For example, suppose a router can only monitor 100 flows (for performance reasons) and it wishes to monitor them for consecutive intervals. The router cannot be certain the flows will send packet every sampling interval. If it assumes this, it could be left with less flows to analyze than the desired 100. To be certain it has 100 flows for analysis and also try to continuously monitor flows, a router needs to have the capacity to monitor 200 flows. Monitoring a larger number of flows than required would also prove useful in selecting only the large flows. If only a fixed number of flows can be monitored then the router has to accept a trade-off: either try consecutive monitoring of flows and risk having less flows than desired at the end of the sampling interval or also use flows not sampled before.

In our theoretical model we made the assumption that p remains fixed. This does not necessarily occur in practice. The marking probability can change during a sampling in-

terval, especially when the value of the sampling interval is larger. This behavior can also differ from one AQM algorithm to another. RED and REM update the marking probability with every packet, whereas PI's updates are dictated by the flow with the smallest round trip times. For such an algorithm, the sampling interval can be a good approximation of an interval of time when p is constant. Even in RED's case [15] argues that updating the marking probability with every packet provides no benefit, and an updating method similar to PI is possible.

3.2.2.5 Implementation in Routers

The path computation that routers need to do is the same as for the data packet analysis and the clean slate solution for ACK analysis. The difference lies in the method of computing a sample. For this, the backward compatible solution requires routers to monitor a number of flows during a sampling interval. Routers need to be able to monitor both the data and ACK packets for a flow. However, per flow state is not needed. As our evaluation will show, monitoring a small, constant number of flows provides good accuracy. For each monitored flow only a few counters are required to keep track of the groups of unmarked packets. For each monitored flow, the sequence numbers of the CWR marked data packets need to also be remembered. They can be discarded when the corresponding ACK is encountered. For each flow only a limited number of such sequence numbers need to be stored as each of them signifies a reduction in the congestion window of the flow.

3.2.3 General Discussion About ACK Packet Analysis

For every inference mechanism an important question concerns the guarantees regarding overestimation and underestimation. Because our algorithm is based on gathering samples, in the general case no such guarantees can be made. Note however that the estimation errors are usually small, and as a consequence if the marking probability on links is described with discrete quantifiers (e.g. small, medium, high) the overestimation or underestimation will most likely still place a path or link in the same category.

Dropped and reordered packets are factors that can alter the number or sequencing of marked packets. The backward compatible solution for analyzing ACK packets is more sensitive to such events since it requires the length of intervals of unmarked packets. The clean slate solution is less sensitive because packet drops in an AQM/ECN network happen probabilistically. As a result, the percentages of marked ACK packets should remain roughly constant as long as enough ACKs are analyzed. Similarly, data packet analysis should not be sensitive to packet loss.

Since our approach uses the congestion markings in the packets, it is also sensitive to the number of packets received during a sampling interval. If a very large number of packets are received, this can only improve the accuracy of the estimation since there are more values to describe the marking probabilities. Conversely, a very small number of packets received can be detrimental to the estimation. The effects may be more or less visible depending on the variation of the marking probability. If the marking probability remains fairly constant, then the estimation will not suffer if too few are packets received. However, if the marking probability is changing, then, receiving too few packets will impact the sample's ability to reflect an increasing or decreasing trend in the marking probability.

A similar issue can arise because there is a limit to the amount of information that the packet markings can encode. For example, consider the case when a router R_i computes a value of the aggregate marking probability P_{ij} equal to 100% for the path to some other router R_j . Also suppose packets to the subsequent hop R_{j+1} take the same route to R_j . Because all the packets are already marked at R_j there is no way to encode additional congestion information for the link (R_j, R_{j+1}) . The case described is the extreme case. Effects will still be observed when instead of reaching 100% the value of P_{ij} is smaller, but very close to 100%. Few packets will escape unmarked from R_j and therefore, there may be too few packets to correctly encode the marking probability on (R_j, R_{j+1}) . The exact value of P_{ij} where this effect begins to appear is a function of both the bandwidth and the amount of traffic sent. We call this the "visibility problem". Note, however that this only appears in scenarios that are not common for networks, since it requires significant levels

of congestion at multiple, consecutive links or one link operating constantly at a very high level of congestion.

The choice of values for the sampling and estimation intervals can also affect the accuracy of the results. If the two intervals are chosen to be equal then the result will present the bias for periods of time with more packets received. If the sampling interval is very small (enough to contain one packet), the averaging performed for computing the value of the estimation interval will present the same bias. Also note that it takes some time (at most RTT) for a router to receive a marking once it has been set. This means that the markings present a picture of congestion that is slightly older. Because of this, choosing estimation intervals on the order of the RTT can yield estimate with little practical importance since the congestion severity can change in an interval of time equal to the RTT. Larger values of the estimation interval hide this latency and allow for meaningful estimates to be computed.

Chapter 4

From Path Level to Link Level Marking Probabilities

In this chapter, we will describe how the path level marking probabilities already measured using the methods from chapter 3 can be used in the computation of other path level as well as link level marking probabilities.

4.1 Computing Link Level Estimates

One of the first assumptions we made in this thesis was that the AS where our solutions are deployed employs link state routing. With link state routing, every router has a complete picture of the paths between the routers in the network. Using this information and the source and destination fields in the packet headers a router also has a complete picture of the paths it is monitoring.

In this section we analyze the extent to which a router can infer the marking probabilities on different paths and links for the clean slate and the backward compatible solution. We describe our approach using an idealized example, where, for simplicity, we assume that the marking probability on each link stays constant. We use P_{ij} to denote the estimate of the marking probability over the entire path that takes traffic from some router R_i to some router R_j . This is the value computed during an estimation interval. If R_i and R_j are neighbors then $P_{ij} = L_{ij}$, where L_{ij} is the link level estimate. The quantity P_{ii} is undefined for any value of i .

A router R_i can compute the link or path level estimate P_{jk} if it knows P_{tj} and P_{tk} ($t = i$ usually, but not necessarily), and P_{tj} is a strict subset of P_{tk} by using the formula:

$$P_{jk} = \frac{P_{tk} - P_{tj}}{1 - P_{tj}} \quad (4.1)$$

The same quantity can also be computed if P_{jt} and P_{kt} are known, and P_{kt} is a strict subset of P_{jt} . The formula is similar:

$$P_{jk} = \frac{P_{jt} - P_{kt}}{1 - P_{kt}} \quad (4.2)$$

In equation 4.1 we assumed that if P_{tj} is a strict subset of P_{tk} then $P_{tj} \leq P_{tk}$. However, this inequality may not always hold in practice. For example, if P_{jk} is uncongested then the values of P_{tj} and P_{tk} should be equal. In practice, small differences will appear. One possible reason is that routers may choose a different set of flows to monitor when using the backward compatible solution. Another reason is that a marking that is considered by a router as part of a sampling interval can be considered by another as part of the next sampling interval. In all these cases, if the quantity $P_{tk} - P_{tj}$ is negative then zero should be used instead. The same argument applies also to equation 4.2.

To exemplify the use of the formulas, consider the network in Figure 4.1. Bidirectional links are used to connect routers. Routing is shortest-path and each link is labeled with its corresponding weight. Unless otherwise stated, the route through R_5 is not used.

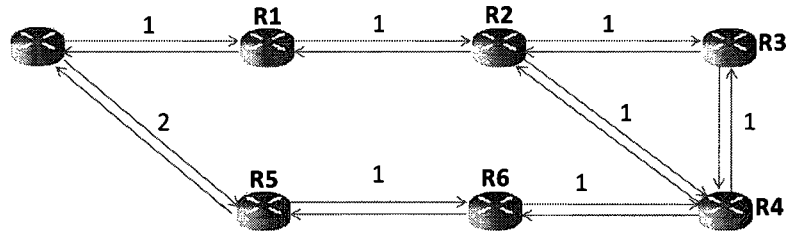


Figure 4.1 : A sample network

Lemma 1

A router R_i that observes the only data packets sent by some source R_s can compute P_{si} . In other words, R_i will be able to infer the marking probability on all the paths that carry data traffic to it. In Figure 4.1, if R_0 and R_1 send TCP traffic to R_4 , then, R_2 will be able to infer P_{02} and $P_{12} = L_{12}$.

Lemma 2

A router R_i that observes only the ACK packets sent by some receiver R_d back to some source R_s and uses the clean slate solution can compute P_{sd} . In other words R_i will be able to infer the marking probability over the entire forward path. In Figure 4.1, consider that R_0 sends traffic to R_4 using R_1 as a next hop but the reverse ACK traffic goes through R_5 . In this case, from the analysis of the ACK packets, R_5 will infer P_{04} .

Lemma 3

A router R_i that observes only the ACK packets sent by some receiver R_d back to some source R_s and uses the backward compatible solution cannot compute useful estimates.

Lemma 4

A router R_i that observes both the ACK and data packets sent between some sender R_s and some receiver R_d can calculate P_{si} , P_{id} and P_{sd} no matter which solution it is using. R_i can compute P_{sd} using Lemma 2 or by using the backward compatible solution. It can always infer P_{si} using Lemma 1. Using equation (4.1) it can then proceed to compute P_{id} . In other words, R_i can infer the marking probability on both the paths carrying traffic to it and away from it. In our example suppose there is traffic from R_0 to R_4 and it is taking the route through R_1 . R_1 can infer P_{01} by analyzing the data packets and the entire P_{04} with the help of the ACK packets. P_{14} can then be computed using equation (4.1).

Lemma 5

A router R_i might not be able to infer the P_{jk} quantity for all routers P_j and P_k in the network. This is the case when the path P_{jk} does not carry data packets to R_i nor does R_i receive the corresponding ACK packets. For example, the link $R_3 - R_4$ is never used to carry packets that eventually reach R_0 , since it is not on the shortest path from R_0 to any other routers. Therefore, R_0 will not be able to infer $P_{34} = L_{34}$.

4.2 Applicability

In real networks all the scenarios described are possible. Let Fwd_{sd} and Rev_{sd} be the set of routers on the forward path and, respectively the reverse path for traffic generated by R_s for

R_d . If R_i is either the source or the destination then the result obtained is P_{sd} . If routing is asymmetric, then, a router can possibly find itself on either a data path or an ACK path for which it is neither source nor destination. If $R_i \not\subseteq Fwd_{sd} \cap Rev_{sd}$ and $R_i \subset Fwd_{sd}$ then Lemma 1 applies. If $R_i \not\subseteq Fwd_{sd} \cap Rev_{sd}$ and $R_i \subset Rev_{sd}$ then Lemma 2 or Lemma 3 apply. If $R_i \subseteq Fwd_{sd} \cap Rev_{sd}$ then the case is described by Lemma 4. Note that Lemma 4 does not require routing to be symmetric. If the first-hop and last-hop router for end-hosts are unique, then those routers will always be able to analyze both data and ACK packets, irrespective of the degree of asymmetry of the entire end-to-end path.

Lemma 6

In a situation where there is traffic flowing between every pair of routers in the network (all-to-all traffic pattern), then, a router R_i can compute link level estimates for all links that carry data traffic to and away from it. Since R_i is the first hop for the traffic it generates it will receive both the data and ACK packets from every flow it generates. Therefore it can use Lemma 4 to compute an estimate for the paths to all other routers in the network. Equation 4.1 can then be used to calculate the link level estimates. Such a traffic pattern is very common today. For example, a large number of backbone networks deploy routers to aggregate traffic from entire cities, and there is usually traffic flowing between one city to another.

While networks with an all-to-all traffic pattern benefit the most from our approach the solution can be applied to any traffic pattern. However, for arbitrary traffic patterns, not all link level marking probabilities may be inferred by some router R_i , even though the links carry traffic to or away from R_i . In our example, suppose R_0 only sends traffic to R_5 and R_4 . It only uses the path through R_5 because link $R_1 - R_2$ failed. While the link $R_6 - R_4$ does carry traffic to and from R_0 , R_0 will only be able to infer P_{54} and not P_{64} . This is because R_0 does not receive any packets from R_6 . However, path level estimates can be equally useful for network level decisions. For example, in the case of an FRR algorithm, if re-routing causes a path to be congested, the exact link level location of the congestion might not be relevant.

To conclude, using the path level marking probabilities measured by analyzing data and ACK packets and routing information available from the link state advertisements a router can compute the marking probability of more paths in the network. For many networks today, link level marking probabilities can be computed this way.

Chapter 5

Evaluation

5.1 Methodology

We conduct ns-2 simulations to evaluate our algorithm. We now proceed to describe our default experiment setup. If certain experiments require changes in the parameters of our setup we discuss the changes and the reasoning behind them in the corresponding sections.

Network environment

Unless otherwise stated we use RED as the AQM algorithm because it is standardized [21] and present in many routers [22, 23, 24]. For RED we disable the waiting between marked packets (the ns-2 `wait_` parameter) in order to be compliant with the RFC design. We also set the final marking probability to linearly increase to 1 (the ns-2 `max_p_` parameter) as the average queue size grows from `min_thresh` to `max_thresh`. For every link in the network we use a router buffer size equal to the product between the link capacity and the average round-trip time in the network. This value is commonly used in routers today [25]. We then set `min_thresh` to 25% of the buffer size and `max_thresh` to 75% of the buffer size. For congestion notification we use the ECN+ variant because, the extra protection conferred to the SYN/ACK packets of flows gives us more control over the starting times of the flows and therefore more control over the design of our experiments. However, we do not expect this choice to influence the results and conclusions of our experiments. While the standard ns-2 simulator does not implement ECN+, the modifications needed to convert ECN into ECN+ are minimal.

Topologies

We use synthetic topologies for our experiments because they allow us to create more challenging scenarios to test our approach. Link bandwidth is limited to 100 Mbps in

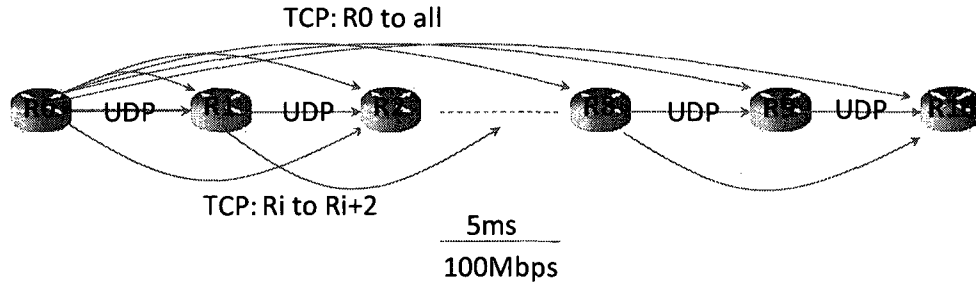


Figure 5.1 : Chain topology and traffic pattern

order to keep the simulation times tractable. However, we also discuss the effects of an increase in bandwidth. The setup for the synthetic topology is described in Figure 5.1. It is a chain topology with 10 links. The length was chosen to be greater than the typical number of hops in most ASes. We deliberately chose this type of topology because the computation performed by our inference algorithm on a chain topology is a generalization of the computation for any arbitrary topology. The reason is that, for any topology the inference algorithm is based on path level estimates and computations. The propagation delay on each link in the topology is 5 ms. Nodes are numbered consecutively from 0 to 10. We use node R_0 as a vantage point. Node R_0 performs all computations and observes the markings on the packets. All the inferences presented are from the point of view of node R_0 .

Traffic

We use both TCP and UDP sources to generate traffic. The TCP flows used are FTP flows that once started last of the entire duration of the simulation. We use TCP Reno for the TCP traffic. In certain experiments we use TCP source with the congestion avoidance algorithm disabled. These sources send packets using a fixed window size. Packet sizes are 500 bytes for both TCP and UDP traffic. There is TCP traffic flowing from node R_0 to each of the other nodes, and the number of such flows is computed using the expression:

$$Nr. \text{ Flows from } R_0 \text{ to } R_i = NRF * i^2 \quad (5.1)$$

The parameter NRF can be tuned and the default value is 50. NRF is also the number of flows started on the first hop in the network. Note that the number of flows started from R_0 to the other routers increases with the distance from R_0 in order to receive comparable number of packets from each router. This ensures node R_0 has the ability to infer all link level marking probabilities in the network. Background traffic is simulated by TCP traffic from any node R_i to node R_{i+2} . A fixed number of 50 flows are started between such pair of routers. We use UDP sources as method of inducing variation in the congestion level in the network. We devised a custom UDP source that changes its sending rate by 1Mbps every second, while continuously cycling between 0Mbps and the maximum link bandwidth. Such sources are started between every consecutive pair of routers. All the TCP and UDP flows described are set up on the forward path.

Metric and default parameter values

Unless otherwise stated a sampling interval of 0.5 seconds and averaging interval of 30 seconds. Each simulation runs for 600 seconds. To quantify the accuracy we use the 50th and the 90th percentile of the absolute differences between the inferred link level marking probabilities and the real ones.

$$AccuracyMetric = |InferredMark.Prob. - RealMark.Prob.| \quad (5.2)$$

The real marking probability at each router queue is averaged using sampling and estimation intervals. The values for these intervals are equal to the ones used for inference. We chose to perform numerical comparisons of the inferred and real marking probabilities instead of discretely quantifying congestion based on the severity (e.g. low, medium). In practice, most applications should be content with just a discrete representation of congestion. However, performing direct numerical comparisons allowed us to better understand the limits and limitations of our approach.

For the clean slate solution the packet history is fixed at 10 packets. For the backward compatible solution, groups of unmarked ACKs are allowed to span at most two sampling intervals. All the flows are monitored and finally at most 100 of them are used for the inference. Preference is given to flows that can be continuously monitored. The rest of the

flows are chosen randomly.

Description of experiments

In this evaluation section we focus on the analysis of the ACK path. One reason is that there are more factors influencing the congestion information available on the reverse path and this makes the inference more challenging and error-prone. Also, as explained, this allows us to present results for links downstream from a vantage point. Such links are usually very important for network level decisions. Moreover, the accuracy of the ACK path estimation is what differentiates our two solutions, because the data path computation is the same.

The clean slate solution is a direct method of computing the congestion estimates. The backward compatible solution tries to recover the congestion estimates using the size of the groups of unmarked packets. For this reason, the clean slate solution is expected to be the lower bound on the performance of the backward compatible solution.

The experiments that we perform can be grouped into three different categories. The first group of experiments analyze the behavior of the solutions under different values of their parameters. Most parameters are shared by the two solutions. When an experiment is relevant to just one solution this is described in the title of the section. The second group of experiments present the behavior of the solutions under various network environments. The last experiment presents a challenging scenario for the solutions and describes their behavior with respect to the limitations imposed by the scenario.

5.2 Sensitivity to Parameter Values

5.2.1 Backward Compatible: Sensitivity to Number of Monitored Flows

In this experiment we analyze the sensitivity of the backward compatible solution to the number of monitored flows. The value of the NRF parameter is 130. We chose to plot the estimation accuracy starting with the 4th hop since this is the first hop that receives more than 2000 flows from R_0 . This allows us to test scenarios where a wide range of flows are monitored. The results are shown in figure 5.2. When less than 10 flows are monitored the

results are bad. This is expected because a small number of flows will not provide enough data samples. This is especially true in scenarios when congestion is present because flows may have responded to congestion and decreased their sending rate. Monitoring 100 flows kept the estimation error under 0.05. More importantly, monitoring over 2000 flows did not present significant improvements. This suggests that monitoring a small, constant number of flows is enough to obtain good accuracy.

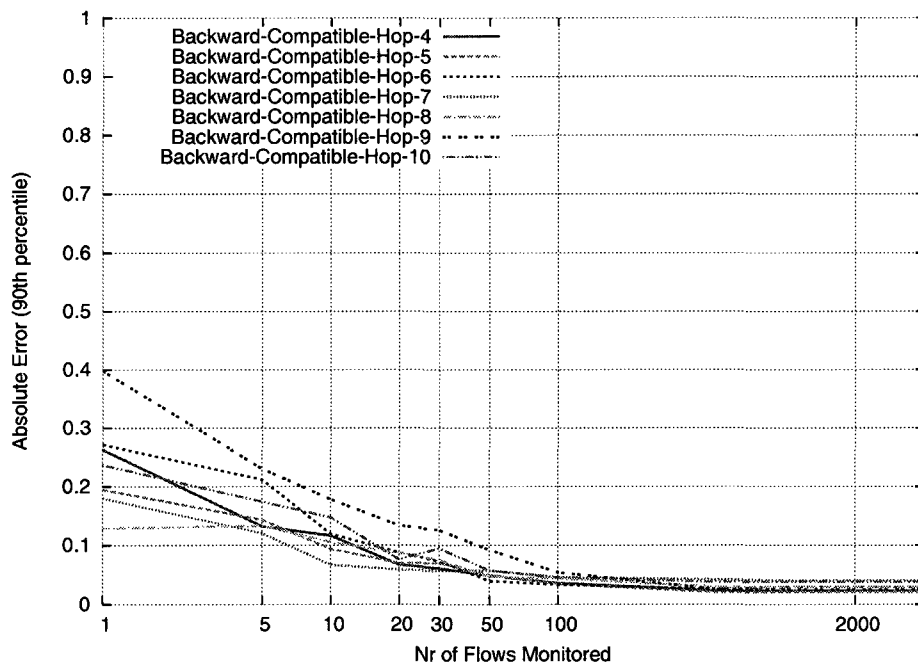


Figure 5.2 : Error vs number of flows monitored

5.2.2 Sensitivity to Ratio of the Estimation and Sampling Interval

In this experiment we analyze how the variation in the ratio of the estimation interval and the sampling interval affects the accuracy of the results. We fix the sampling interval at 0.5 seconds (which is in the range of the average network RTT) and vary the size of the estimation interval. The results are shown in figure 5.3. The x-axis is plotted at logarithmic scale.

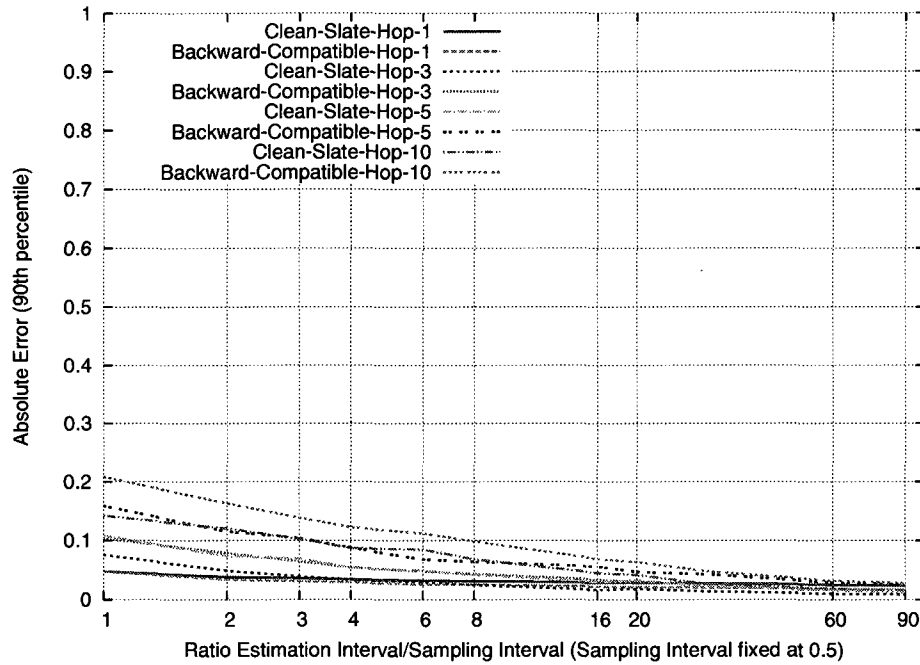


Figure 5.3 : Error vs ratio of estimation and sampling interval

The inference for small ratios is visibly more error-prone than for larger ratios. The reason is that, for small ratios, the value of the estimation interval approaches the range of the RTT values (ratio value of 1). The congestion information contained in the packets has an inherent delay in reaching a destination because of queuing and the network transmission time. This delay becomes more important as the ratio decreases to values close to 1. As a result, the estimates are less and less comparable to the real marking probabilities. On the other hand, once the estimation interval is well outside the RTT range, these factors exhibit negligible influence, and the accuracy of the inference remains high. The larger estimation intervals are precisely the intervals that network level decisions need to be based on.

5.2.3 Sensitivity to the Value of the Sampling Interval

We now analyze the effect the variation of the length of the sampling interval has on the accuracy of the inference. We use values of the sampling interval ranging from 0.05s to 1.5s and compute the 90th percentile of the accuracy error. For the backward compatible solu-

tion a group of unmarked packets is allowed to span at most two sampling intervals. The results are shown in 5.4. The clean slate solution is not affected by the variation because, as long as enough packets are received during a sampling interval, the correct percentage of marked packets can be computed. On the other hand the backward compatible solution is much more sensitive to the variation. The reason is that the chance for a group of unmarked packets to span multiple sampling intervals increases when the length of the sampling interval decreases. As a result, the chance that large groups of unmarked packets will be missed by the inference also increases. This results in overestimations of the marking probability. The effects of the overestimation are visible in the large errors for small values of the sampling interval. Effects are more pronounced for hops more distant from R_0 because of the larger RTT of the flows. For these flows it takes more sampling intervals to produce the same number of packets as the flows with shorter RTT. Therefore, the groups of unmarked packets span more sampling intervals and as a result errors are higher.

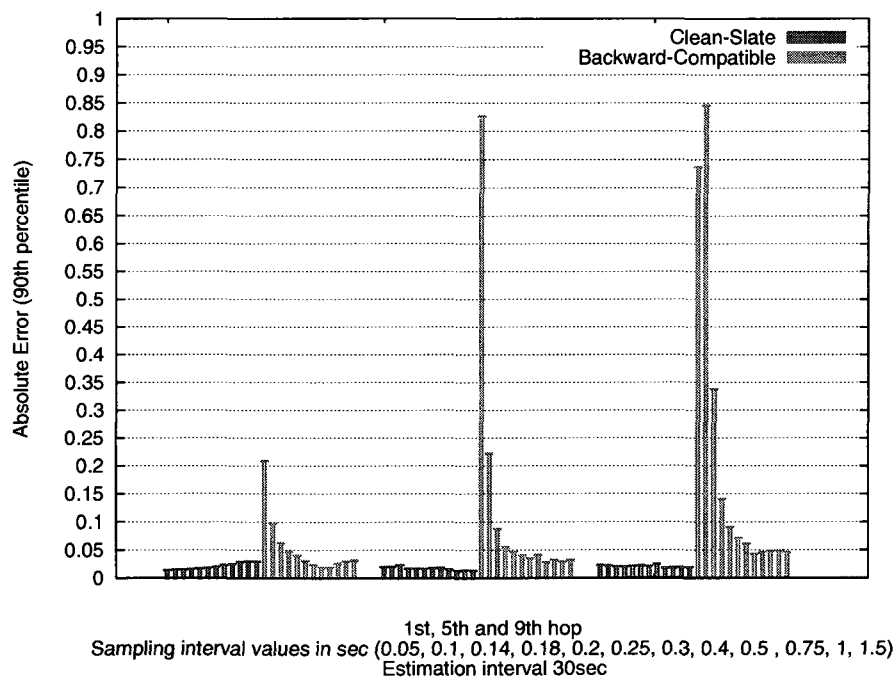


Figure 5.4 : Error vs hop nr for different values of the sampling interval

5.3 Sensitivity to Network Environment Conditions

5.3.1 Congested Link Followed by Uncongested Links

In this experiment we congest the first link in the network. To do this, we start a variable number of TCP flows between routers R_0 and R_1 . The rest of the links are kept uncongested. We show the inference errors for the uncongested links in figure 5.6. The figures plot the 90th percentile and the 50th percentile of the error. The marking probability on the uncongested links is sometimes overestimated for both solutions. The reason is that while the congested links marks all traffic to different destinations with the same probability, the resulting percentage of marked packets may differ from one destination router to another. This is inherent in the design of the marking algorithm, since it is a probabilistic approach. The overestimates are small. Our solutions can present an uncongested link as congested but the inferred value of the congestion is always low. Therefore, if congestion intervals are defined the true severity of the congestion on the link will correctly be understood (e.g. very low congestion).

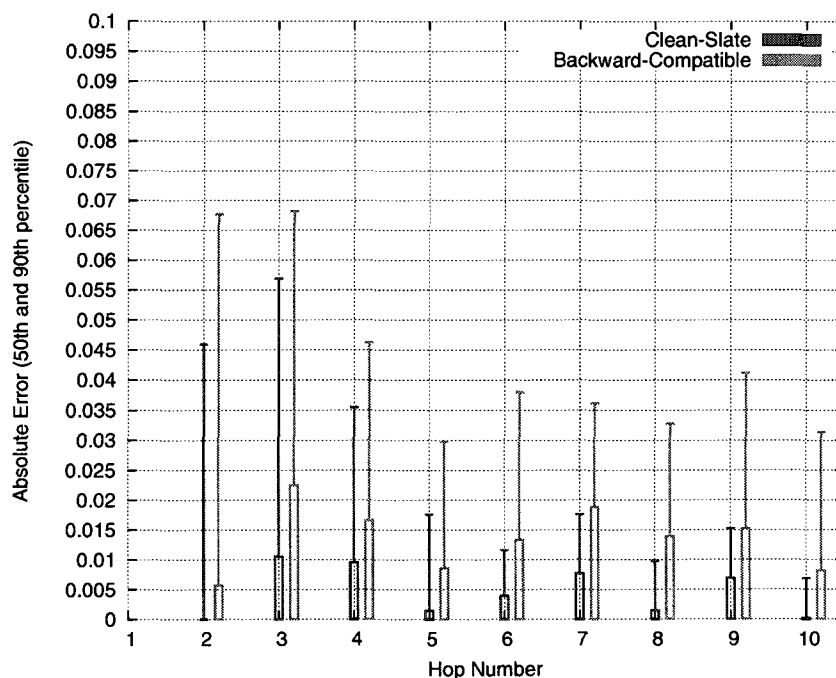


Figure 5.5 : Error on uncongested links vs hop count

5.3.2 Sensitivity to Different AQM Algorithms

We use different AQM algorithms to test our solutions. Alongside RED, we also chose REM and PI for evaluation. The parameters used for REM and PI are the default ns-2 values. The results are shown in Figure 5.6. The performance of REM is not as good especially when considering the 90th percentile of the error. There are two reasons for this. First, with REM the visibility problem appears as soon as the third hop. This is described in Figure 5.7 where the real, aggregate path level marking probability for all the paths in the network is plotted. The results in Figure 5.6 follow the pattern of the visibility problem. Note the increase in error starting with the third and sudden degradation of the estimation starting with the fifth hop. However, the inference for the first hop is still not comparable to the results for PI and RED, although the visibility problem does not affect the first hop. To understand this we need to look at the function that REM uses to map congestion to marking probability. This is an exponential function. This function is far more likely to produce a visibility problem compared to the linear function of RED and PI. Moreover, it produces high variations in the marking probability. Routers need more samples to be able to track such variations.

Note that the performance of PI is similar to that of RED. One reason is that the two algorithm use the same linear function to map congestion to marking probability.

5.3.3 Performance in Higher Bandwidth Environments

This experiment analyzes the sensitivity of the inference to an increase in network bandwidth. We start with the base case of 50Mbps and go up to 1Gbps. Intuitively, an increase in bandwidth provides more packets for the inference. To capture the importance of this effect we use a different traffic load pattern, where a constant number of flows (instead of increasing with the distance from R_0) equal to the value of NRF is started between the same pairs of routers used throughout all experiments. We set NRF to 125 flows for a bandwidth of 50Mbps and then increase the number of flows proportionally to the increase in bandwidth. At 50Mbps this setup allows only few packets to be exchanged between R_0

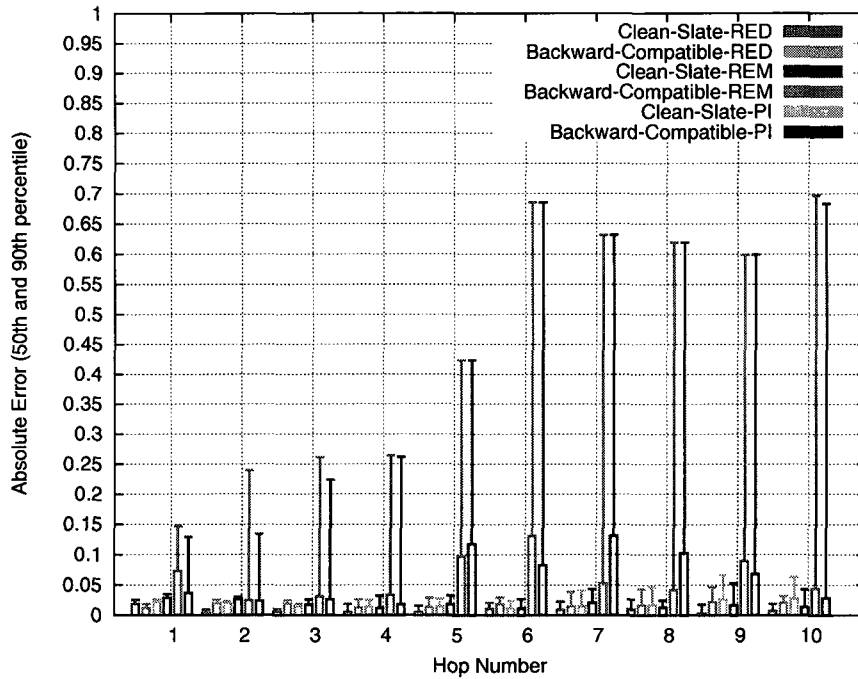


Figure 5.6 : Absolute error for RED,PI,REM for different network hops

and the last hops in the network since the flows with shorter RTTs use most of the network bandwidth.

Results are shown in Figure 5.8. With an increase in bandwidth there are more and more packets exchanged between R_0 and routers starting with R_7 . Since more packets are used for inference the accuracy of the results also increases. Note that the improvement diminishes when bandwidth is scaled up to 1Gbps because the inference process is already provided a large number of packets at 500Mbps.

5.3.4 Sensitivity to Sudden Changes in the Congestion Pattern

This experiment analyzes the impact of traffic dynamics on the behavior of the solutions. We start with the default network configuration and traffic pattern. At two points during the simulation, for one of the links (link (R_4, R_5) is used but others would have been equally good) a large number of TCP flows are started to mimic a considerable disruption in the traffic pattern. The two disruptions differ in the duration of the TCP flows. The first dis-

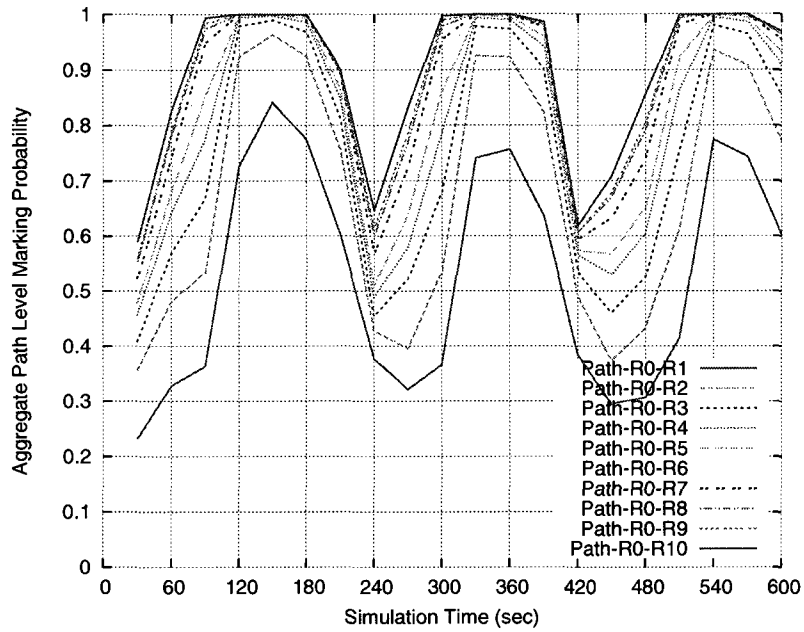


Figure 5.7 : Aggregate path level marking probabilities

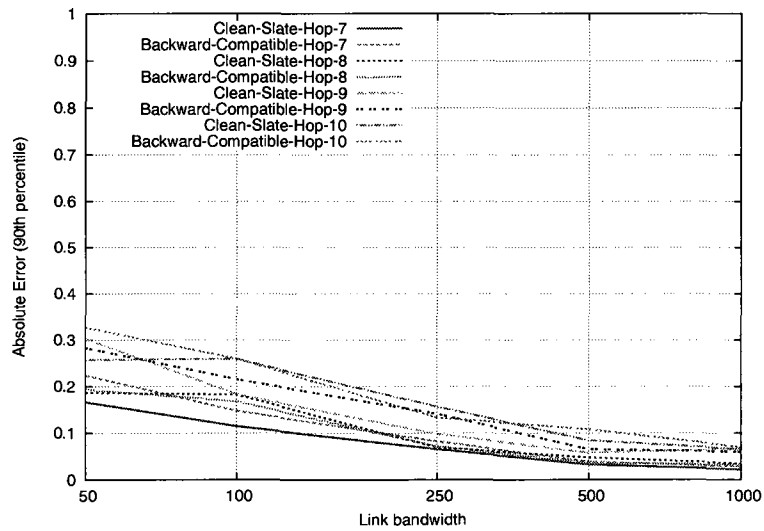


Figure 5.8 : Performance in higher bandwidth environments

ruption is confined within one estimation interval while the second one, which is more pronounced, spans three estimation intervals.

Nr Flows Started	Resulting Marking Probability	Maximum Error
250	0.28450	0.0427
500	0.35011	0.0328
1000	0.41810	0.0497
2500	0.50041	0.0446
5000	0.55416	0.0521
10000	0.58013	0.0552

Table 5.1 : Effects of a sudden change in the traffic pattern

Table 5.1 describes the more pronounced disruption . It contains values for the number of flows that are suddenly started on the link (R_4, R_5) , the resulting marking probability and the maximum value of the inference error for the link (R_4, R_5) for the whole simulation. Note that in the absence of the new TCP flows link (R_4, R_5) would have a marking probability under 0.05. The data shows that the disruptions have a minimal effect on the accuracy of the inference. Irrespective of the magnitude of the disruption the maximum error over the entire simulation is at most 5%.

5.4 Limitations

5.4.1 The Visibility Problem

In this experiment we analyze the visibility problem. Recall that this problem appears when a path from some router R_i to R_j is already marking close to 100% of the packets at some intermediate hop R_k . As a result there are only few unmarked packets that can carry useful information about the path $(R_k \dots R_j)$. In this experiment we use the modified TCP sources that send traffic using a fixed congestion window of 2. Any value would have sufficed, but a small value allows us to better control the increments in the marking probability. The choice of a fixed window TCP decouples the visibility problem from the effects of TCP's congestion control algorithm. This is meant to show that the visibility issue is a problem in itself and does not appear only as a result of TCP sources decreasing their sending rate and providing fewer samples for inference. Moreover, the fixed window TCP sources allow the

marking probabilities created to remain nearly constant and this allows us to better track the visibility problem as a function of the marking probability.

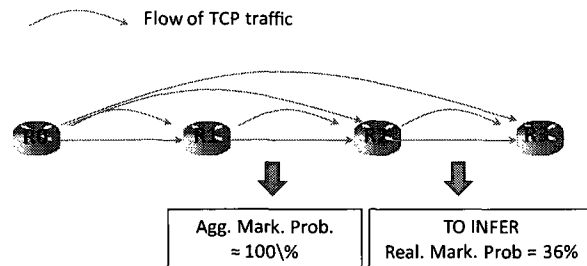


Figure 5.9 : Network environment

Figure 5.9 depicts the evaluated scenario. It consists only of the first three links of our chain topology as this is enough to describe the visibility problem. Flows are started from R_0 to all other routers, from R_1 to R_2 and from R_2 to R_3 . We fix the marking probability on the link (R_2, R_3) at 36% (any value works) by starting a fixed number of TCP flows to R_3 throughout all the simulations in this experiment. We vary the aggregate marking probability on the link (R_1, R_2) by starting different numbers of TCP flows between R_1 and R_2 . The purpose of the experiment is to see the degradation in the inference for the link (R_2, R_3) when the marking probability on (R_1, R_2) increases to 100%.

The results are shown in Figure 5.10. We plot the 50th and 90th percentiles of the accuracy metric for both our solutions. Throughout all the experiments the number of packets that R_0 receives from R_3 stays above 2000 for every sampling interval. Even with a large number of packet available for analysis, we see that the accuracy of the estimation drops significantly as the marking probability on link (R_1, R_2) increases to 100%. As fewer and fewer packet escaped unmarked from router R_1 , it is increasingly more difficult to encode the marking probability on (R_2, R_3) . In the extreme case (100% marking probability) there are no more packets to carry useful information about (R_2, R_3) and therefore, R_0 considers the marking probability to be 0. The resulting error is exactly 36%, the value of the marking probability on (R_2, R_3) . An exact point at which the visibility effect begins to appear

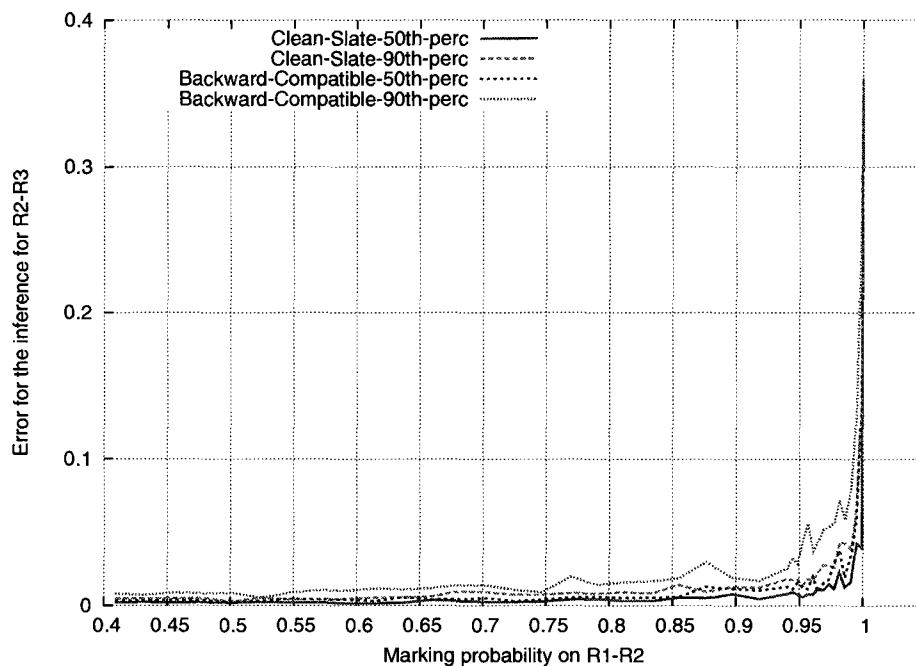


Figure 5.10 : The visibility problem

cannot be precisely determined as it depends on the number of packets received, the aggregate marking probability on the paths and the variation of the marking probability on the path we want to measure ((R_2, R_3) in our example).

In conclusion, the visibility problem can appear irrespective of the number of packets used for analysis. Note however, that the scenario used for the experiment is highly unlikely to occur in a RED/ECN network in practice. It requires either multiple links on a path to be significantly congested in order to drive the aggregate marking probability close to 100%, or one link to have a very high marking probability. Both cases are exceptional cases that, if occurring in an RED/ECN enabled network, signal significant design, underprovisioning, or attack problems for a network. Even so, our approach can detect the problem by signaling the unusually high marking probabilities on the links.

In the experiments presented so far we attempted to use marking probabilities that do not allow the visibility problem to appear.

Chapter 6

Discussion

6.1 Limitations

Our approaches for inferring congestion are not free of limitations. A limitation common to both data and ACK analysis is the dependency on a steady stream of traffic. If there is not enough traffic on a network path then the congestion estimation for that path will be inaccurate.

The clean slate solution for ACK packet inference requires some changes to the protocols.

The backward compatible solution also has its limitations. It requires routers to monitor a set of flows for each network path on which congestion estimates are desired. However, a small, constant number of flows is enough to provide good accuracy. Groups of unmarked packets can span multiple sampling intervals. Limiting this number of intervals causes large groups to be missed by the inference. This can cause the real marking probability to be overestimated when the limit is low and the marking probability is low (or the sampling interval is short) since in this scenario groups of unmarked packets have a higher probability of spanning multiple sampling intervals. On the other hand, allowing a group of unmarked packets to span multiple sampling intervals can bring stale congestion information into the computation.

6.2 Deployment Environment

We now analyze which network environment our congestion inference approach is best suited to. We have explained that our solution can allow the computation of link level

congestion estimates when the network presents an all-to-all traffic pattern. Moreover, a continuous flow of packet is desired for good accuracy. Backbone networks have both of these characteristics since routers are typically used to aggregate traffic coming from entire cities. With millions of users serviced by such a router, the chances of traffic being sent to all other routers in the network is very high. However, ISPs typically overprovision their networks and, as a result, congestion today is not frequently present in the backbone networks. Congestion can still happen though because of flash crowd events, denial of service attacks, mis-configurations or equipment failures. If FRR algorithms are deployed then this can also be a cause for congestion. For other types of networks such as data center networks our solution will not be able to infer link level estimates because the tree-like topologies used do not allow for an all-to-all traffic pattern to be realized. However, even the path level congestion estimates that can be computed can be useful in improving network performance. Moreover, congestion appears far more often in data center networks. As new designs are proposed for data centers we envision that our solution could also be deployed in such an environment with significant benefits.

6.3 Applications

In the introduction section we described to uses routers can have for congestion information to: fast-reroute algorithms and router verification mechanisms. Other routing schemes can also take advantage of our solution. Virtually any routing technique can be designed to take advantage of congestion information. For example, solutions that use source-routing can compute different source routes if congestion appears in the network. Congestion is not the only network information routing decisions need to be based on. Packet loss or available bandwidth are two other options. However, we argue that congestion information should also be taken into consideration because it offers important information about the network. Moreover, we have described an approach that infers the congestion in the network, without using additional traffic while at the same time requiring limited router resources.

6.4 Re-routing

Failures are events that can happen at any moment in today's networks. When a failure occurs and routers route around it takes some time for the link-state information to propagate throughout the network. During this time, routers that are not yet aware of the changes could still attribute old forward paths to flows. Consequently, some routers will have counters that contain incorrect information. The correct solution is for the routers to reset all the counters that have been affected by the change once the new link-state information becomes available. Since link-state information is reliably disseminated this is guaranteed to happen.

6.5 Equal-Cost Multipath

One assumption we states was the fact that a router must not calculate an incorrect forward path for data packets that correspond to the ACKs it receives. This possibility can arise when Equal-Cost Multipath (ECMP) is used in the network. Typically, with ECMP [26], the decision to assign a flow to a particular link of equal cost is deterministic but local to a router. Moreover, the reverse path need not be the same as the forward path. Therefore, ECMP decisions might not be discernible outside the router that takes them. In our approach, assuming steady state in the network, every router when receiving some ACK packet takes one of the following two decisions. If the corresponding data packets are not forwarded using ECMP then a router can simply identify the forward path using link state routing information. If from the routing information or any other source a router decides that data packets are routed using ECMP then the router should not use the corresponding ACK packets for the inference since it cannot reliably identify the forward path. If ECMP decisions are based on a set of rules known by all the routers, then congestion on equal cost links can also be inferred and the packets traveling over equal cost paths can be used by our inference algorithm.

6.6 Network Heterogeneity

Networks contain different kinds of equipment. There are practical and economical reasons for this. It is not trivial for an ISP or a company to upgrade all the routers in the network at once. Moreover, different routers serve different purposes, and therefore it is logical for them to differ in processing power, design or algorithms supported. For example, a core router needs to be more powerful than an edge router. Our approach does not require routers to use the same AQM algorithm nor the same parameters for the algorithm. All that we require is that congestion marking is used alongside an AQM algorithm.

6.7 The TCP Delayed ACK Algorithm

The TCP delayed ACK algorithm, first specified in [12] allows a TCP receiver to momentarily delay the acknowledgement of a received data packet. Multiple data packets can therefore be acknowledged in one ACK packet. This approach can reduce both the protocol processing overhead in end-hosts and the amount of traffic sent by the TCP receiver in the network. The TCP congestion control RFC [27] recommends that end-hosts should use this algorithm. Our congestion inference solution is based on analyzing the congestion markings in the packets. When one ACK packet is used to acknowledge multiple data packets the congestion information is altered. In this thesis we considered a TCP variant that separately acknowledges every data packet received. However, note that [27] recommends that at least every second TCP segment be acknowledged irrespective of size. Moreover, the ECN standard [9] requires that an ACK be marked if any of the data packets it acknowledges was marked. In this case, there are only two possible combination of markings in two data packets that would create a marking in an ACK packet. Therefore, even when using delayed ACKs, the congestion information in the ACK packets is not irreversibly damaged. As future work we plan to pursue this direction and attempt to extend our inference algorithm to be used in the presence of the TCP delayed ACK algorithm.

Chapter 7

Related Work

Counting a percentage of marked packets is not a new idea. The Pre-Congestion Notification (PCN) Architecture [28] uses packet markings to encode information about flows that exceed their allotted rate. These markings allow a node in the network to decide whether to admit or terminate a flow.

The Re-ECN protocol [29, 30] is perhaps the closest related work to this thesis. It is a method for holding flows accountable for the congestion they create in the network. In this protocol TCP receivers use an extra bit in the headers to convey the path level marking probability back to the TCP sources. The sources then mark the data packets they send in a such a way to encode the marking probability received from the TCP destinations. Routers on the paths then can use this information along with the percentage of marked packets with ECN to infer marking probabilities for both the upstream and the downstream path. In contrast, this thesis also explores the congestion inference problem without requiring any changes to protocols. Moreover, we also analyze the extent to which routing information can be combined with congestion information to allow routers to infer link level congestion information.

Most of the methods developed for inferring the congestion level are designed for TCP end-hosts and help them change their data sending rate according to current network conditions. TCP end-hosts are concerned only with the overall quality of the end-to-end paths, so they are not interested in the state of any particular link. As a result, most of today's algorithms convey path-level congestion information [31, 32]. Other approaches like [33] describe only the most congested link on a particular path. On the other hand, link level information is exactly the type of information that network level decisions are based upon.

Our solution goes beyond these methods and uses the path level measurements to obtain more fine grained link level information.

Another group of methods deal with inferring shared congestion between two flows [34, 35]. These methods are useful for performing cooperative congestion control. These solutions could also be potentially used in conjunction with routing information to infer congestion on network links. While this possibility has not been researched, this group of solutions also has its limitations. They works well when only one link on a path is congested, however they have difficulties when faced with arbitrary congestion levels on more links. Most solutions in this category also make us of additional traffic. Our solution, however, does not require additional traffic and can be used even in challenging scenarios when multiple links on a path are congested.

Chapter 8

Future Work and Conclusion

The approach we presented has its limitations. In the future we plan to address some of those limitations. We plan to study our solution in an environment where delayed ACKs are used. Dropped packets are also an importance concern. Dropped ACKs can influence the accuracy of our backward compatible solution. Moreover, dropped packets could be a concern in any networks that are not fully AQM/ECN enabled since some routers will still use packet drops as a signal of congestion. The backward compatible solution also requires a router to analyze both the data and ACK packets of a flow. We plan to consider changes to this solution that would allow a router to infer marking probabilities by analyzing only the ACK packets of a flow.

In this thesis we presented and analyzed methods that routers can use to passively infer path level marking probabilities from existing traffic. We showed how data packets can be leveraged in order to compute probabilities. Using ACK packets for inference is not straightforward because the congestion information is altered by the TCP receiver when echoing congestion markings from data to ACK packets. We also presented two solutions for inferring marking probabilities from ACK packets that deal with this inaccurate congestion information. We also argue that benefits can be obtained by combining routing information with the inferred path level marking probabilities. This allows link level marking probabilities to be computed. Our solutions use little router resources. The evaluation shows that the accuracy of the estimation is good over a variety of multiple traffic conditions and AQM algorithms. Moreover the accuracy significantly improves with an increase in bandwidth. As routers perform increasingly complex tasks we envision that the congestion information derived using our approach can be used to drive network level decisions.

Bibliography

- [1] K. Lakshminarayanan, M. Caesar, M. Rangan, T. Anderson, S. Shenker, and I. Stoica, “Achieving convergence-free routing using failure-carrying packets,” in *SIGCOMM*, August 2007.
- [2] A. Kvalbein, A. F. Hansen, T. Cicic, S. Gjessing, and O. Lysne., “Fast ip network recovery using multiple routing configurations,” in *INFOCOM*, April 2006.
- [3] M. Shand and S. Bryant, “IP fast reroute framework.” Internet Draft draft-ietf-rtgwg-ipfrr-framework-13.txt, October 2009.
- [4] A. Mizrak, Y. Cheng, K. Marzullo, and S. Savage, “Fatih: Detecting and Isolating Malicious Routers,” in *DSN*, 2005.
- [5] J. Hughes, T. Aura, and M. Bishop, “Using conservation of flow as a security mechanism in network protocols,” in *IEEE Symposium on Security and Privacy*, May 2000.
- [6] B. Zhang, *Efficient Traffic Trajectory Error Detection*. PhD dissertation, Rice University, Houston, TX, 2010.
- [7] D. Katz, K. Kompella, and D. Yeung, *RFC 3630 - Traffic Engineering (TE) Extensions to OSPF Version 2*, 2003.
- [8] H. Smit, *RFC 3784 - IS-IS Extensions for Traffic Engineering (TE)*, 2004.
- [9] K. Ramakrishnan, S. Floyd, and D. Black, *RFC 3168 - The Addition of Explicit Congestion Notification(ECN) to IP*, 2001.
- [10] J. Moy, *RFC 2178 - OSPF Version 2*, 1998.

- [11] D. Oran, *RFC 1142 - OSI IS-IS Intra-domain Routing Protocol*, 1990.
- [12] R. Braden, *RFC 1122 - Requirements for Internet Hosts – Communication Layers*, 1989.
- [13] S. Floyd and V. Jacobson, “Random early detection gateways for congestion avoidance,” *IEEE/ACM Transactions on Networking*, vol. 1, no. 4, pp. 397–413, 1993.
- [14] S. Athuraliya, V. H. Li, S. Low, and Q. Yin, “Rem: Active queue management,” *IEEE Network*, May 2001.
- [15] C.V.Hollot, V. Misra, D. Towsley, and W.-B. Gong, “On designing improved controllers for aqm routers supporting tcp flows,” in *INFOCOM*, April 2001.
- [16] A. Kuzmanovic, A. Mondal, S. Floyd, and K. Ramakrishnan, *RFC 5562 - Adding Explicit Congestion Notification (ECN) Capability to TCP’s SYN/ACK Packets*, 2009.
- [17] F. Hernandez-Campos, K. Jeffay, and F. Smith, “Tracking the evolution of web traffic: 1995-2003,” in *MASCOTS*, 2003.
- [18] C. Fraleigh, S. Moon, B. Lyles, C. Cotton, M. Khan, D. Moll, R. Rockell, T. Seely, and C. Diot, “Packet-level traffic measurements from the sprint ip backbone,” *IEEE Network*, November 2003.
- [19] H. Kim, K. Claffy, M. Fomenkov, D. Barman, M. Faloustos, and K. Lee, “Internet Traffic Classification Demystified: Myths, Caveats, and the Best Practices,” in *ACM CoNEXT*, December 2008.
- [20] E. Blanton and M. Allman, “On making tcp more robust to packet reordering,” *ACM Computer Communication Review*, vol. 32, no. 1, 2002.
- [21] B. Braden, D. Clark, J. Crowcroft, B. Davie, S. Deering, D. Estrine, S. Floyd, V. Jacobson, G. Minshall, C. Partridge, L. Peterson, K. Ramakrishnan, S. Shenker, J. Wroclawski, and L. Zhang, *RFC 2309 - Recommendations on Queue Management and Congestion Avoidance in the Internet*, 1998.

- [22] S. Floyd, “RED (Random Early Detection) Queue Management.”
<http://www.icir.org/floyd/red.html>.
- [23] Cisco, “Cisco IOS Software Releases 11.1 Distributed WRED.”
http://www.cisco.com/en/US/docs/ios/11_1/feature/guide/WRED.html.
- [24] Juniper, “JUNOS 9.1.x Quality of Service Configuration Guide: RED and WRED Overview.”
<http://www.juniper.net/techpubs/software/erx/junose91/swconfig-qos/html/drop-profiles-config3.html#1515246>.
- [25] G. Appenzeller, I. Keslassy, and N. McKeown, “Sizing router buffers,” in *SIGCOMM*, August 2004.
- [26] D. Thaler and C. Hopps, *RFC 2991 - Multipath Issues in Unicast and Multicast Next-Hop Selection*, 2000.
- [27] M. Allman, W. Paxson, and W. Stevens, *RFC 2581 - TCP Congestion Control*, 1999.
- [28] P. Eardley, *RFC 5559 - Pre-Congestion Notification (PCN) Architecture*, 2009.
- [29] B. Briscoe, A. Jacquet, C. Gilfedder, A. Salvatori, A. Soppera, and M. Koyabe, “Policing congestion response in an internetwork using re-feedback,” in *SIGCOMM*, August 2005.
- [30] B. Briscoe, A. Jacquet, T. Moncaster, and A. Smith, “Re-ecn : Adding accountability for causing congestion to tcp/ip.” Internet Draft [draft-briscoe-tsvwg-re-ecn-tcp-08.txt](#), work in progress, April 2010.
- [31] R. Thommes and M. Coates, “Deterministic packet marking for congestion price estimation,” in *INFOCOM*, April 2004.

- [32] M. Adler, J.-Y. Cai, J. Shapiro, and D. Towsley, “Estimation of congestion price using probabilistic packet marking,” in *INFOCOM*, April 2003.
- [33] L. L. H. Andrew, S. V. Hanly, S. Chan, and T. Cui, “Adaptive deterministic packet marking,” *IEEE Communications Letters*, vol. 10, no. 11, pp. 790–792, 2006.
- [34] M. S. Kim, T. Kim, Y. Shin, S. S. Lam, and E. J. Powers, “A wavelet-based approach to detect shared congestion,” in *SIGCOMM*, August 2004.
- [35] J. K. D. Rubenstein and D. Towsley, “Detecting shared congestion of flows via end-to-end measurement,” *IEEE/ACM Transactions on Networking*, pp. 381–395, 2000.