Estimating Shared Congestion Among Internet Paths

Weidong Cui, Sridhar Machiraju, Randy H. Katz, Ion Stoica
EECS Department, University of California, Berkeley
Email: {wdc,machi,randy,istoica}@eecs.berkeley.edu

***Abstract*— Recent work on media streaming has proposed to exploit *path diversity*, i.e., the use of multiple end-to-end paths, as a means to obtain better performance. The best performance is achieved when the various paths are independent in the sense that the two paths do not share a *Point of Congestion (PoC)*. However, topologies used in media streaming applications do not meet the assumption of Inverted-Y or Y topologies made by prior work on detecting shared PoC. In this paper, we propose a new technique called *CD-DJ (Correlating Drops and Delay Jitter)* which solves this problem. CD-DJ is better than earlier solutions for three main reasons. First, CD-DJ overcomes the clock synchronization problem and can work with most topologies relevant to applications. Second, it provides applications with an estimate of the fraction of packet drops caused by shared PoCs. This information is more useful than a "yes/no" decision for media streaming applications because they can use it to choose a path based on the level of shared congestion. Third, CD-DJ makes the estimation by correlating bursts of packet drops in conjunction with the correlation of delay jitter in a novel way. A key contribution of our work is our evaluation methodology. We use a novel overlay-based method to evaluate our technique extensively using about 800 hours of experimental traces from Planetlab, a global overlay network. Our results indicate that CD-DJ calculates estimates which are at least within a factor of 0.8 of the actual fraction of shared drops for $80-90\%$ of the flows. We also illustrate the advantage of using CD-DJ with a simple streaming video application.**

I. Introduction

The traditional approach to congestion control in the Internet has been for end hosts to use AIMD (Additive Increase Multiplicative Decrease) schemes similar to those used by TCP. By using end host based measurements, overlay networks such as RON [1] introduce the concept of rerouting around IP links that cause packet drops. In this paper, we refer to such links as *Points of Congestion (PoCs)*. Another recent approach to limit the effects of congestion has been to use path diversity. In [2], [3], [4], the authors propose to send suitably encoded video streams on multiple paths to increase their tolerance to network congestion. The reasoning is that such multiple paths may not share a PoC. For applications such as CDNs, and enterprise backups that run on large scale distributed systems and inject large amounts of data into the network, there are no known methods to coordinate hosts distributed over the wide-area for cooperative congestion management. Proposals for congestion sharing such as Congestion Manager [5] can be used only at a single host.

All the aforementioned approaches to adapt to congestion in ways other than simple AIMD either assume that multiple paths not sharing a PoC are known [2], or achieve this goal using rapid probing to determine loss rates etc. [1], [6]. In case of large scale applications such as CDNs, application-level congestion management is unknown. In fact, it is an open issue if such congestion management would actually help these applications. We believe that this is indeed the case. In this paper, we propose *CD-DJ (Correlated Drops and Delay Jitter)*, a technique to estimate the fraction of packet drops caused by shared PoCs of two paths on the Internet. We also use multipath media streaming as an application to illustrate various aspects of our technique. Media streams are susceptible to bursty errors which result in degradation of playback [2] [3] [4]. By using multiple overlay paths the size of such bursty errors can be reduced which results in better playback quality.

Prior work related to detecting shared congestion [7], [8], [9] use the fact that most Internet routers employ droptail queues which cause bursty drops [10]. Hence, packets that traverse the PoC around the same time are likely to be dropped or not dropped together. They are also likely to have correlated delays and delay jitter. One of the main limitations of these techniques is that they cannot be applied to applications such as multipath media streaming which violate the topology assumptions of these techniques.

To estimate shared congestion between two paths, CD-DJ obtains information about packet drops and delay jitter using probe flows along the two paths. It calculates the estimate by correlating bursts of packet drops in conjunction with the correlation of delay jitter in a novel manner. The main challenges in designing such a technique are:

- Packets that traverse a PoC at around the same time are likely to both get dropped or not. Similarly, packets traversing a PoC at around the same time are likely to be correlated in delay. Determining if two packets traversed a PoC at the same time based on their sending and receiving times is not easy since the senders and receivers of the two flows may not have synchronized clocks. Even if they are synchronized, different one-way delays from the senders to a PoC cannot be measured. We refer to the sum of the clock skew and the difference in one-way delays as *synchronization lag (synclag)*.
- Links other than the PoC could be responsible for a significant fraction of end-to-end delay. Drops may occur at multiple PoCs some of which may not be shared. Both these create a lot of noise which adversely affects the correlation among two flows sharing a PoC. Such noise could cause false positives and false negatives.
- Evaluating any technique to determine shared PoCs is not easy since information about PoCs is not readily available from the network.

The following are the main innovations of our work:

- We provide a solution to overcome problem due to synchronization lag. Our solution is to infer the synchronization lag as the value that would maximize the correlation among packet drops of the two flows.
- Multiple PoCs may exist along the two paths and only some of them might be shared. Given two flows along these paths, CD-DJ produces an estimate for each flow that represents the fraction of drops of each flow that occur on a shared PoC. Such an estimate is a natural rank of a potential path and is more useful than a "yes/no" decision.
- Flows sharing a PoC may also be expected to experience similar delays and delay jitter. However, the amount of noise in delay jitter is very large especially when the high latency links follow the PoC. We therefore propose to use delay jitter conservatively.
- A key contribution of our work is our evaluation methodology. We use a novel overlay-based method using PlanetLab [11], a global overlay network, to construct paths with shared PoCs. This method provides non-trivial (not equal to zero and one) bounds on the fraction of losses that occured on shared PoCs. Our results indicate that CD-DJ calculates estimates which are at least within a factor of 0.8 of the actual fraction of shared drops for $80-90\%$ of the flows.

This paper is organized as follows. We provide an overview of the related work in Section II. In Section III, we explain our goals and assumptions, and describe our complete solution. In Section IV, we describe our evaluation strategy and our implementation experiences. We present the results of our experiments in Section V and discuss several open issues in Section VI. We present our conclusions in Section VII.

II. Related Work

Our work was motivated by previous studies on Internet path characteristics [12] [13] [14] [15], proposed approaches to detecting PoCs [7] [8] [9] [16] [17], and various applications which exploit path diversity [2] [3] [4] [18] [19].

It is well known [10] that the droptail queues used in most Internet routers lead to periods of bursty loss. Jiang *et al.* [12], Sanneck *et al.* [13], and Yajnik *et al.* [15] use discrete-time Markov chain models, particularly the 2-state Markov chain model (also known as the Gilbert model), to study the temporal dependence in packet loss. In the Gilbert model, a droptail queue has two states: *lossless* and *loss*. The loss state represents the periods of bursty losses. They show that the Gilbert model captures temporal dependence with a good trade-off between accuracy and complexity. Yajnik *et al.* found that the correlation timescale of packet loss is $1000ms$ or less. In [14], Zhang *et al.* show that the fine time scale correlation is caused by trains of consecutive losses. They also find that the duration of loss runs is very short, $220ms$ at the 95 percentile.

There are a few solutions that have been proposed for detecting if a pair of flows share a PoC. In [7], Rubenstein *et al.* use cross and auto correlation of delay and loss of Poisson probes to detect a shared PoC. They prove that the cross correlation of delay or loss is greater than the autocorrelation when two flows share a PoC. An implicit assumption made by them is that arrival times of Poisson probes in the queue at a shared PoC still follow a Poisson distribution. However, this need not be true especially if there is significant jitter before the shared PoC. Harfoush *et al.* [9] use loss probabilities of single-packet and packet-pair probes (so-called conditional Bayesian probing) to identify shared losses. They use only simulations to evaluate the Bayesian probing technique. The applicability of their scheme in real wide-area networks is not known. More recently, Katabi and Blake [8] measure Renyi entropy of packet inter-arrival times to infer shared PoCs and then clustere flows which share the same PoC. This technique does not perform well under heavy cross traffic, which limits its applicability. In [16], Padmanabhan *et al.* study the use of various sampling methods to characterize lossy links to a server from clients. They use passively observed client-server traffic to identify lossy link(s) from the client to the server. In [17], Younis *et al.* use in-band delay measurements to cluster clients sharing the same congested route to a server.

Recent work [2], [3], [4] propose the use of multiple paths to improve the performance of media streaming. Besides the IP path, they assume the use of an overlay node to construct an alternative path. This corresponds to the triangle topology shown in Figure 1. Prior solutions to detecting shared PoCs cannot be applied in this case since they are limited to the Y and Inverted-Y topologies in Figure 1.

Our work differs from prior work in three significant ways:

- Other than making a binary decision on the existence of a shared PoC, we can provide an estimate of the fraction of drops of each flow caused by shared PoCs. When no completely independent paths exist, such information can be useful for applications to determine which alternate paths to be used.
- All prior solutions can work only with the so-called Inverted Y and Y topologies. This is because the information on the order and/or times of packets traversing the shared PoC required by earlier solutions is not available in general topologies. Our solution overcomes the clock synchronization problem and can work with most topologies relevant to applications.
- Most prior work do not verify their techniques in a variety of wide-area scenarios. In contrast, we extensively evaluate our CD-DJ technique based on 800 hours of experimental traces collected from 45 sites in PlanetLab, a global overlay network.

III. Design

We first describe our goal and assumptions. We then discuss technical challenges to achieve our goal. We finally present our approaches in detail.

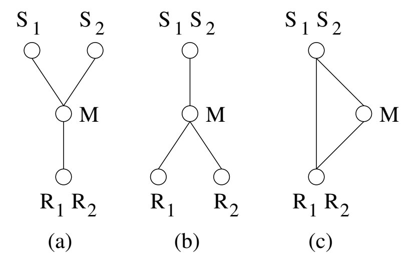


Fig. 1. (a) Y-topology (b) Inverted-Y topology (c) Triangle Topology

A. Goals and Assumptions

Given two paths in the Internet, our goal is to estimate the fraction of shared congestion using probe flows along the paths. Our technique outputs the fraction of packet drops of each flow caused by shared PoCs. We do not want to merely detect shared PoCs for two reasons: (1) Such estimates of shared congestion are natural ranks for potential paths and are much more useful than a "yes/no" decision. (2) There may be multiple PoCs along a path.

In this paper, we use two types of probing mechanisms: (1) active probing: probe the network by sending periodic UDP packets; (2) passive monitoring: monitor continuously backlogged FTP-like TCP traffic. For the TCP flows, we assume that we can observe packets at both the senders and the receivers using a tcpdump[20]-like program.

We make two assumptions about the Internet: (1) Most routers use droptail queuing disciplines. (2) Most traffic is TCP-based. Prior work [12] [13] [14] [15] on measuring and monitoring Internet path characteristics show that Internet paths experience bursty packet drops which is observed in [10]. Data analysis from the Sprint ATL IP Monitoring Project [21] shows that about 95% traffic is TCP-based.

We assume that the the node running the CD-DJ technique can be requested by the application to estimate shared congestion between two paths. Upon such a request, this node initiates two probe flows on each of the paths. In case of passive probing, it initiates monitoring processes to passively monitor the probe flows. At the end of some pre-specified probing interval it retrieves the sender and receiver logs and runs the CD-DJ technique on these logs. In the multipath media streaming application, the node running CD-DJ may be the sender of the media stream. It will initiate probing on various candidate paths. The sender runs the CD-DJ technique on the logs of these probing flows. It chooses the path with the least estimate of shared congestion as the alternative path.

B. Challenges

To estimate the fraction of drops at shared PoCs in a general topology, we take the approach of correlating packet drops of two flows. The intuition behind it is that packets of two flows traversing a PoC at roughly the same time are either both dropped or not dropped. We face several technical challenges when we try to correlate packet drops of two flows.

- *Synchronization*
  When the senders of two flows are not synchronized and the delay from each of them to a shared PoC is not known, it is hard to obtain information regarding the order in which packets of the two flows traverse the PoC. However, this kind of information is generally required for correlating two flows to detect if they share a PoC.
- *False Positive*
  Though the period of bursty packet drops is usually $200ms$ or less, some bursty packet drops may last longer than that. When one flow is experiencing a long period of bursty drops caused by a PoC, another flow is likely to have some packet drops at a different PoC during this period even if they are independent. This will give us false positive information of the correlation of the two flows.
- *False Negative*
  The packet drop probability is not necessarily 1 during a bursty loss period. This is because the queue size of a droptail queue may drain or build up in that period. This may cause two flows sharing a PoC not to see simultaneous packet drops at the shared PoC.

In the rest of this section, we will present our technique and explain how it tackles these challenges.

C. The CD-DJ Technique

We propose *CD-DJ (Correlated Drops and Delay Jitter)*, a technique that correlates packet drops and delay jitter along the two paths to find the fraction of packet drops at a shared PoC. The CD-DJ technique consists of three stages:

- Determining the synchronization lag.
- Calculating μ_1, μ_2 (see Table I), the fraction of correlated packet drops, i.e., drops that occurred at the same time after synchronizing the sending times.
- Inflating μ_1, μ_2 using the delay jitter correlation of the uncorrelated drops.

TABLE I

NOTATIONS USED IN THIS PAPER

Notation	Comments
μ	estimated fraction of packet drops at the shared PoC of any flow
$\hat{\mu}$	real fraction of packet drops at the shared PoC of any flow
μ_i	estimated fraction of packet drops at the shared PoC of flow i
$\hat{\mu_i}$	real fraction of packet drops at the shared PoC of flow i
μ_{min}	lower bound of the fraction of packet drops at the shared PoC of any flow
μ_{max}	upper bound of the fraction of packet drops at the shared PoC of any flow
f	overlap fraction
b	burst interval
synclag	synchronization lag
CCC	cross correlation coefficient

1) **Determining Synchronization Lag:** Before we can correlate packet drops and delay jitter, we need to synchronize the two flows such that we can determine which packets of the two flows traversed the shared PoC at roughly the same time. In a general topology the senders and receivers are not synchronized. The one-way delay to a shared PoC may also be different for each flow. These two factors lead to a non-zero *synchronization lag (synclag)* We illustrate an example of synchronization lag in Figure 2. The synclag of a flow F_2 with respect to flow F_1 is the sum of the clock skew between its sender and the sender of flow F_1 (Δ) and the difference in delay from the senders to a shared PoC ($d_2 - d_1$). Subtracting this value from the sending times of packets of F_2 gives us an estimate of the sending times of F_2 in terms of the clock used at F_1's sender. Since the clock skew is bounded by the RTT between the two senders and the delay to a shared PoC is bounded by the RTT of each flow, the synclag itself is bounded by $2 \cdot RTT_{max}$ which is not more than a second or two in today's Internet.

Synclag makes it hard to determine the times at which packets of the two flows traversed a shared PoC. Our solution is to try various values of synclag and correlate the drops of the two flows. We take the value that maximizes the cross-correlation coefficient[1] (CCC) as the synclag for the two flows in question. With this estimate of the synclag, we can determine the fraction of drops that occurred simultaneously in both flows.

To calculate the synclag, we generate *loss indicator*

[1] $C(X,Y) = \frac{E[(X-E[X])(Y-E[Y])]}{\sqrt{E[X^2-E^2[X]]E[Y^2-E^2[Y]]}}$

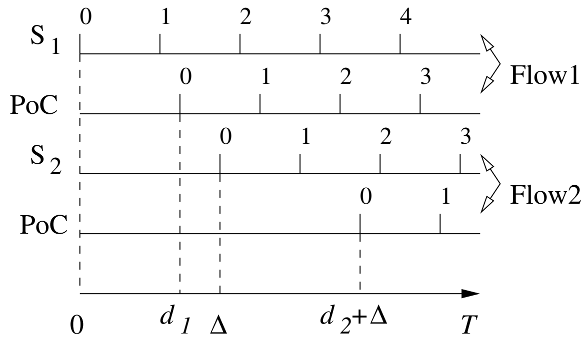


Fig. 2. An illustration of synchronization lag $\Delta + d_2 - d_1$ when the two senders S_1 and S_2 start within Δ of each other and have a delay of d_1 and d_2 to the shared PoC

sequences for the two flows. Each sequence is a string of 0's and 1's each of which corresponds to a time interval. A 1 indicates the loss of at least one packet sent in that time interval. We assume various values for the synclag and generate these loss indicator sequences with the sending times "synchronized" using our assumed synclag. We then calculate the CCC of these two sequences. Since synclag is bounded by $2 \cdot RTT_{max}$, we generate the CCC for all possible values in this range. The value producing the maximum CCC is taken to be the synclag of the two flows. Figure 3 shows a typical case where the maximum cross correlation (and the results of our technique) reached a maximum for the correct value of synclag which is 0.5. For the remainder of our technique we use synchronized sending times only.

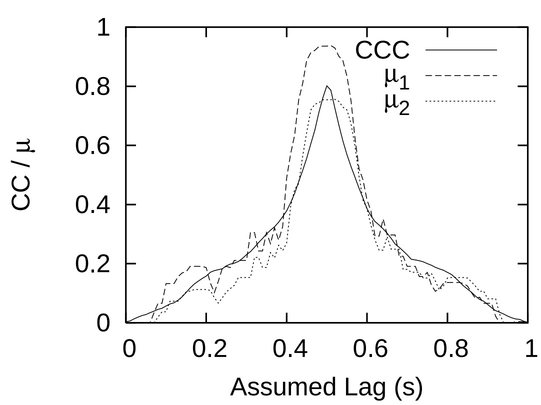


Fig. 3. An Illustration of the synclag where cross correlation and the output of CD-DJ achieve maximum

2) **Calculating Fraction of Correlated Drops:** As mentioned earlier in Section III-B, packet drop information contains noise which may cause both false positives and false negatives when packet drops are correlated. We tackle this problem by correlating bursts of packet drops rather than single packet drops.

Previous work [12] [13] [15] on losses in the Internet conclude that most losses in the Internet occur in bursts lasting up to $200ms$. They also conclude that the time between consecutive bursts is much greater than $200ms$. While bursty losses are a result of the droptail queueing discipline used in most routers, TCP-related congestion control is responsible for the comparatively large time difference between consecutive bursts. Our traces collected on PlanetLab also confirm the occurrence of bursty drops. In Figure 4, we show the CDF of the number of consecutive packet drops of UDP flows sending packets at a rate of 100Hz. It shows that more than 20% of consecutive packet losses consisted of more than one packet. The occurrence of bursty drops implies that packets of two flows traversing a PoC at roughly the same time are either both dropped or not dropped.

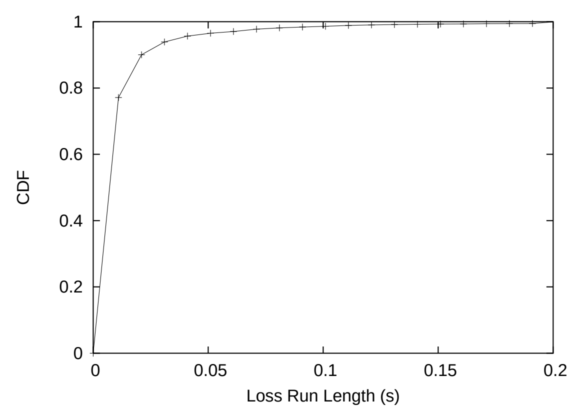


Fig. 4. Loss Run length

We consider two dropped packets to be in the same burst if they were sent within a time *b (burst interval)* of each other. Using this, we construct bursty loss periods for each flow. We define two bursts to be correlated only if the overlap of the two bursts consists of more than an *overlap fraction* f of the drops of *each* burst. Our estimate of $\mu_1(\mu_2)$ is the fraction of drops of $F_1(F_2)$ that belong to such correlated bursts. Note that μ_1 and μ_2 need not have the same value since each flow could traverse other PoCs, too.

Figure 5 shows an example of how false positives and false negatives can be avoided. In this example, we assume all bursts except $Burst_1$ of flow 2 are caused by the same PoC and overlap fraction f is 0.5 (we will explain parameter selection in Section V). Since the overlap of $Burst_1$ of flow 1 only contains 2 packets

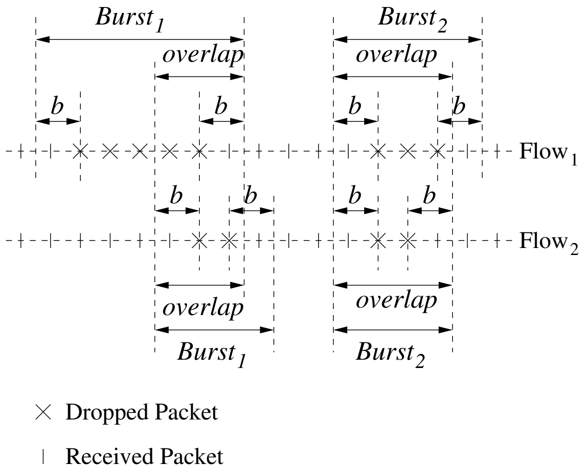


Fig. 5. Example of overlaps of bursty drops

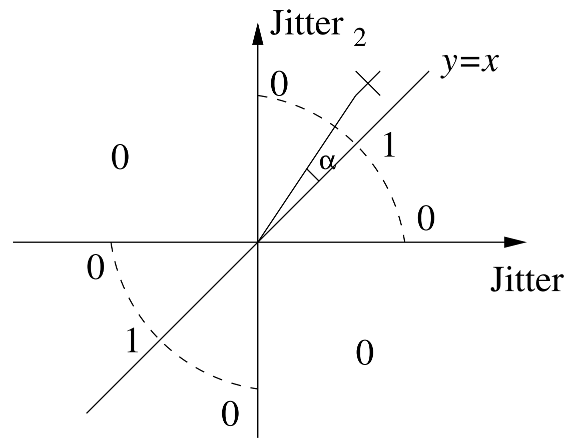


Fig. 6. Example of overlaps of bursty drops

which is less than $5*0.5 = 2.5$, we decide that $Burst_1$'s are not correlated. As for $Burst_2$, we decide they are correlated because the overlap consists of more than 0.5 fraction of the packet drops of each burst though those two bursts do not match exactly.

3) **Correlating Jitter of Uncorrelated Drops:** The third challenge in Section III-B shows that two flows sharing a PoC may not have all drops at that PoC correlated because the packet drop probability during a bursty loss period may not be one. This will cause underestimation of μ. However, during a bursty loss period of a PoC, packets traversing it at roughly the same time may experience similar delay jitter when the droptail queue at the PoC is draining (or building up). This motivates us to inflate the estimate of μ by correlating delay jitter. Given the fact that end-to-end delay in today's Internet has large noise, We do not attempt to correlate jitter for the whole life of the flows. We consider the correlation of delay jitter of uncorrelated drops only. This use of jitter in conjunction with packet drops as a means to estimate shared congestion is a novel aspect of our solution.

For each uncorrelated packet drop, we calculate the jitter observed around this dropped packet as the difference in delays of the closest previous and next received packets. We calculate the jitter experienced around the packet of the other flow that was sent closest to this dropped packet. The basic idea is to decide the probability that an uncorrelated drop is actually caused by a shared PoC based on how similar these two delay jitters are. We add this probability to the estimated number of packet drops on shared PoCs when we calculate μ. We use $cos(2\alpha)$ as the probability that this drop is on a shared PoC where α is the angle between the $y = x$ line and the line joining the origin and the point whose coordinates are the two delay jitter samples (see Figure 6). This function is chosen so that we add a large value when the two jitter values are equal and a small value if they differ by a lot. Note that we consider drops with a positive $cos(2\alpha)$ only to ignore negative values returned by uncorrelated jitter. Since two random jitter values themselves could contribute a positive value on average, we need to compensate for this overestimation. By considering all these factors, we inflate the the estimate μ by $\max(\frac{\Sigma \max(cos(2\alpha),0)-0.3\cdot n}{0.5\cdot N}, 0)$. Here, n and N are the number of uncorrelated and total drops of that flow. The complete derivation of this is provided in the longer version of the paper[22].

IV. Evaluation Methodology and Implementation

In this section, we present our evaluation methodology and implementation experiences.

A. **Evaluation Methodology**

To evaluate our technique thoroughly, we needed to observe two flows along paths that have drops caused by shared PoC(s). To determine the accuracy of the estimate μ for each flow, we also needed to be able to compare it with $\hat{\mu}$ (see Table I), the actual fraction of drops on a shared PoC. We needed to be able to do this for various values of $\hat{\mu}$ and over a diverse set of nodes that spanned the Internet.

To determine $\hat{\mu}$ for two IP flows, we need access to detailed information about drops at various PoCs. Since this is not possible, we used an overlay-based approach. We explain this using an example. Consider the abstract

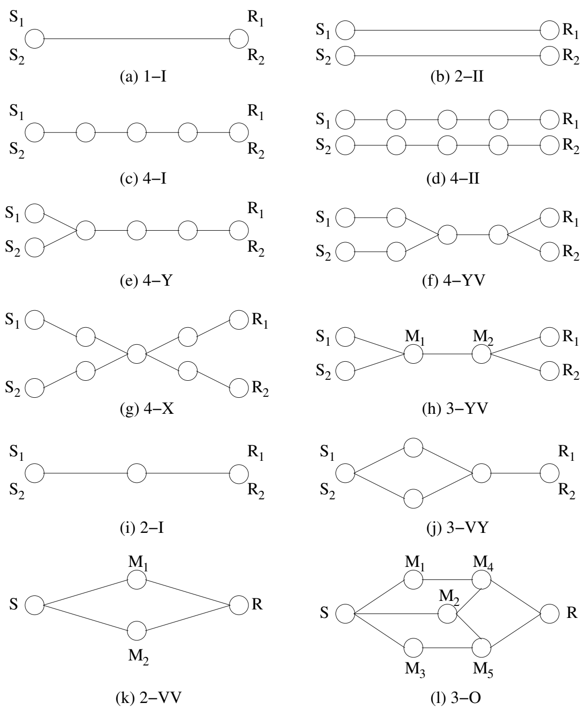


Fig. 7. The Various Topologies Used in Our Experiments

$3-YV$ topology in Figure 7. This topology shows the case of two flows with different senders S_1, S_2 and receivers R_1, R_2. The path taken by the two flows has a shared part from M_1 to M_2. We can create this topology at an overlay level by using overlay routers at M_1 and M_2. The two flows routed on this overlay thus share the overlay link from M_1 to M_2 and all PoCs on this overlay link. Using information about packets received at M_1 and not received at M_2 we can determine all drops at shared PoCs between M_1 and M_2.

The drops on the overlay link from M_1 to M_2 may not be the only drops on shared PoCs. Paths from M_1 to each of S_1, S_2 and M_2 are likely to have some IP links in common. A PoC among these links would be a shared PoC whose drops are not caused on the overlay link from M_1 to M_2. Hence, counting the drops only on the overlay link from M_1 to M_2 can ignore some drops on shared PoCs and would give us μ_{min}, a lower bound on $\hat{\mu}$. Counting the drops on all overlay links from M_1 and M_2 would include drops caused by all possible shared PoCs. Thus, we can also obtain μ_{max}, an upper bound on $\hat{\mu}$. For the $3-YV$ topology, μ_{max} turns out to be 1. The $4-YV$ topology (see Figure 7) is an example of a topology that need not have a μ_{max} of 1.

In summary, we use the overlay approach to evaluate the CD-DJ technique. The evaluation consisted of comparing the estimate μ of CD-DJ with a range $[\mu_{min}, \mu_{max}]$ of possible values. A disadvantage of the overlay approach is that this range can be very large. The use of overlays is purely for evaluation purposes and does not limit the applicability of our technique.

Figure 7 shows the various topologies that we used in our experiments. Four hops were used in most topologies since this provided instances where μ_{min} and μ_{max} were both not trivial (0 and 1 respectively). In order to evaluate CD-DJ with a diverse set of nodes, we used PlanetLab [11], a global overlay network consisting of 45 sites around the world.

B. Flow Generation

We used two kinds of flows: UDP and TCP. We used system timers to implement the Constant Bit Rate UDP flows. The TCP flows were generated using a simple TCP server that would sit in a loop calling blocking "send" system calls for a pre-specified period of time. The overlay routers forwarded packets from the previous hop to the next and collect information on which packets were forwarded. We used such information to deduce the overlay-links on which each packet drop occurred. Compared to two flows using one end-to-end IP path, these overlay-based flows differed in two aspects: (1) The one-way delay (RTT) and jitter seen by these flows is much larger in the multi-hop topologies. (2) Application overhead is encountered at each overlay router twice.

Generating TCP flows that could be routed on the overlay was not easy. With UDP flows, overlay paths could be specified by using the first-hop router as the destination for the sender. Doing so with TCP led to the creation of an end-to-end connection from the server to the first hop router. What we needed was a way to circumvent TCP dynamics between the server and the first hop router. Our inability to use IP tunneling on PlanetLab meant that we would have to implement such tunneling at the user-level.

We implemented a *TCP proxy* that worked as follows (see Figure 8). The TCP server would be given an inactive port on the local machine as the client address. All packets sent to this port would be sniffed using pcap library [20] and tunneled using UDP to the first-hop router. We used truncated TCP packets as payloads for the UDP packets in order to prevent IP fragmentation. The encapsulated TCP packets received by a proxy at the receiver's side would send the TCP packet to the local client by spoofing the destination port. Most machines return a Reset(RST) message if a TCP host attempts to open a connection to an inactive port. In order to

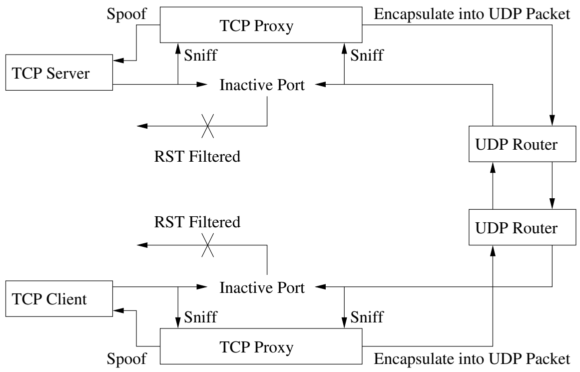


Fig. 8. An Illustration Explaining the TCP Proxy Implementation

prevent this, we filtered all incoming RST messages. The proxy program had about 250 lines of code. In periods of high bursts, it sometimes did not catch all relevant packets. This caused unnecessary drops and reduced the throughput of the TCP flow.

While the above algorithm worked, it required "root" privileges that we did not possess on the PlanetLab machines. To solve this problem, we used our personal machines for the hop running the proxy, the TCP server and the TCP client. The PlanetLab machines ran overlay routers that forwarded UDP-encapsulated TCP packets. Since all machines on which we had "root" privileges were on the same LAN, all our TCP experiments had the same machine as the first and last hop. This is the reason why we used the $3-VY$ topology (Figure 7) for TCP experiments. Since all TCP experiments shared a common hop (our personal machine), more than one TCP experiment could not be run simultaneously. This limited our ability to collect TCP traces.

C. MPEG Streaming

To better motivate the use of our technique with an application, we conducted simple experiments involving MPEG streams using multiple paths. Our experimental setup was partly motivated by the architecture in [2]. We used an MPEG server that would packetize an MPEG file using the recommendations in [23]. It would then send out even-numbered packets on an overlay path P_0 and odd-numbered packets on two overlay paths P_1 *and* P_2. All the three paths started at a host S and terminated at a host R. Paths P_2 and P_0 shared at least one overlay link other than S and R. P_1 did not share any overlay link with P_0 and P_2. This was done so as to obtain an independent pair of paths (P_0 and P_1) and a dependent pair (P_0 and P_2). For simplicity, we packetized only the MPEG frames and assumed that all the MPEG headers were received. We reconstructed MPEG files using the data received from the dependent and independent pair of paths. The reconstructed MPEG packets were compared to the original. The actual metric we used was the mean square error (MSE) of the uncompressed YUV [24] frames of the received and original video files. We obtained 3 MSE values - one for each of Y, U and V frames for each reconstructed file.

Using the MPEG server described above, we wanted to motivate why independent paths need to be chosen and how our technique would help in choosing such paths. We chose two overlay links that had been observed to be lossy. One lossy link would be in P_0 and P_2 while the other would be in P_1. The results of such an experiment are highly dependent on the loss characteristics along each lossy link which could be caused by the flows themselves (the lossy link on the dependent paths carried more traffic). Running this multiple times would not solve this problem. Instead, we ran two MPEG servers with the lossy link on the dependent paths of the first server being on the independent path of the other and vice versa.

The topologies we used are the $2-VV$ and $3-O$ topologies shown in Figure 7. Table II shows the paths used in the two topologies we used for the MPEG experiments. With the $2-VV$ topology, the dependent paths were identical. One of the two simultaneous MPEG servers used the path through M_1 as the dependent path and the other used the path through M_2. In case of the $3-O$ topology, the lossy links were the links to R from M_4 and M_5.

TABLE II
MPEG TOPOLOGY EXPLAINED

Topology	Server Number	Dependent Path 1	Dependent Path 2	Independent Path
$2-VV$	1	SM_1R	SM_1R	SM_2R
$2-VV$	2	SM_2R	SM_2R	SM_1R
$3-O$	1	SM_1M_4R	SM_2M_4R	SM_3M_5R
$3-O$	2	SM_3M_5R	SM_2M_5R	SM_1M_4R

V. PERFORMANCE ANALYSIS

In this section, we present the results of extensive experiments conducted on PlanetLab. We collected about 800 hours of traces from overlay-based experiments over a period of two weeks. Each experiment consisted of running two flows for a period of $600s$. Unless specified otherwise, the flows used sent UDP packets of size 40 bytes at a constant frequency of 100Hz. We first

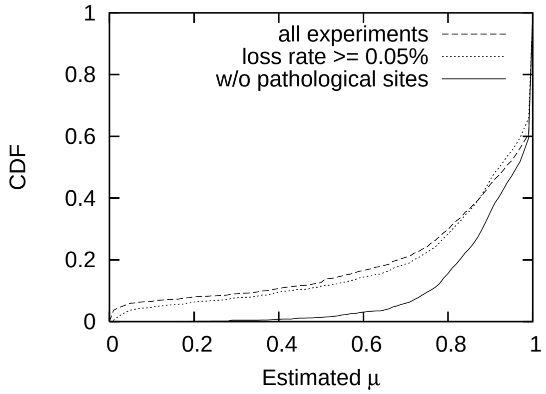


Fig. 9. CDF of μ with and without pathological sites

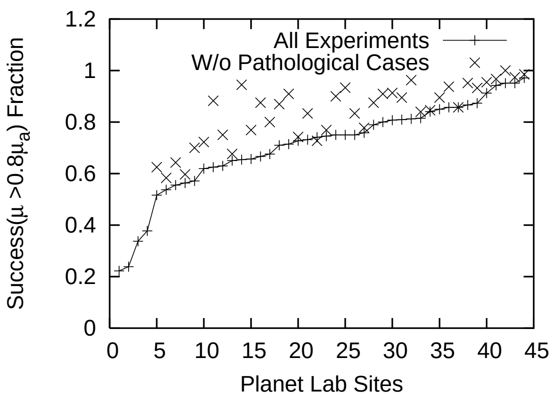


Fig. 10. The success fraction of individual sites with and w/o including pathological cases

explore the reasons behind certain choices we made in designing our CD-DJ technique. These include parameter values and design decisions. We then look at results of UDP experiments with various topologies. We present the results of running CD-DJ with TCP flows next. We end our evaluation section with results that illustrate how CD-DJ may be used with an video streaming application.

A. Evaluation and Design Choices

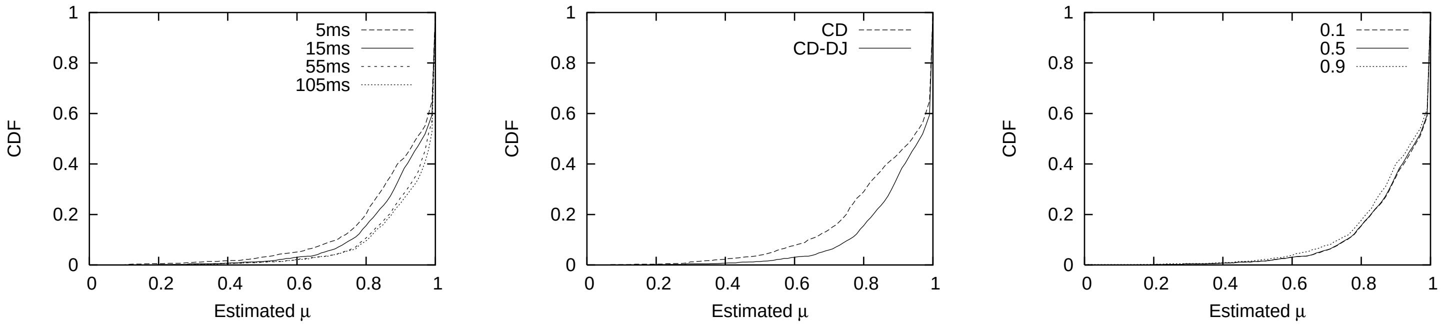
In this subsection, we support the various choices that we made in designing the CD-DJ technique. We do this by plotting the results of experiments of the $1-I$ topology (see Figure 7). Though this is a trivial case in which we expect μ to be 1, this helps us benchmark our technique. By determining if drops (and delay jitter) of two flows using the same path are correlated, we can validate the assumptions we made in deriving our technique (such as bursty losses etc.). We also show the results for the $1-II$ topology in which the two flows use independent paths i.e., do not share any PoC. The reason for analyzing independent paths is to measure the false positive rate of the technique. The actual value of μ is 1 when both flows use the same path and 0 when they use independent paths.

We do not consider experiments in which at least one of the flows had less than 0.05% of its packets dropped. The rationale behind this is that at such low loss rates, losses occur only sporadically and applications may use such paths without hesitation. The 0.05% fraction was chosen rather arbitrarily but our results were not sensitive to this fraction. Also, we did not expect all drops caused by a shared PoC to be classified as shared drops by our technique for reasons discussed in Section VI. For the purposes of this discussion, we define CD-DJ to be *successful* with a flow if the estimate μ is at least $0.8\cdot\hat{\mu}$. We chose this definition as a concise way to state our results.

1) **Pathological Sites:** In analyzing our experiments, we determined that four (out of forty four) sites consistently underestimated the value of μ (i.e., it was less than the $\hat{\mu}$). Figure 10 shows the fraction of successful experiments involving each site on PlanetLab. We see 4 sites that have less than 0.5 success rate whereas others have significantly better success rate. We therefore do not consider these sites in any of our results. The figure also shows that removing these pathological sites increases the success fraction of all other sites barring one. Such removal of pathological sites is justified because we believe that even for real applications, such sites may be determined by running two flows to this site and determining if the technique produces consistently good values for μ. The CDF of μ output by the CD-DJ technique is shown in Figure 9 with and without the pathological sites. For the rest of our analysis, we do not consider these pathological sites.

The results shown in Figure 9 indicate the limit on the performance achievable since this is the case where all drops are on a shared PoC. As we can see from this figure, the μ of more than 80% of our experiments was at least $0.8\hat{\mu}$. Another encouraging aspect is that most unsuccessful experiments estimated μ to be at least $0.5\cdot\mu$. The results for the $1-II$ topology (not shown here) indicated that there were few false positives and all had estimates of μ less than 0.2.

2) **Burst Interval:** We first motivate the need to consider bursts of drops. Figures 11(a), 12(a) show the performance of our scheme assuming that drops are classified to be in the same burst if they were sent within b of each other, for different values of b. A value of $5ms$

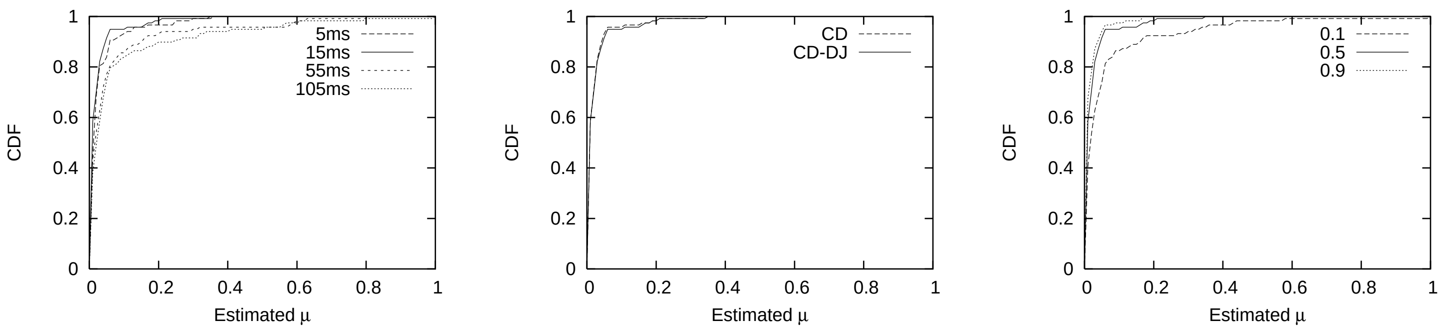


(a) CDF of μ with different burst interval

(b) CDF of μ with and without jitter correlation

(c) CDF of μ with different overlap ratios

Fig. 11. Results with 2 Flows Using the Same Path



(a) CDF of μ with different burst interval sizes

(b) CDF of μ with and without jitter correlation

(c) CDF of μ with different overlap ratios

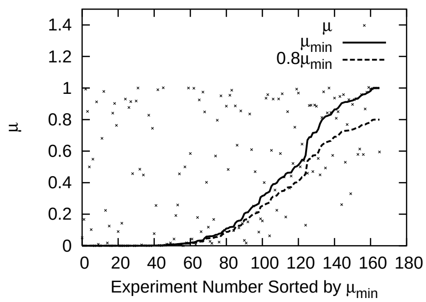
Fig. 12. Results with 2 Flows Using Independent Paths

for b implies that every drop is considered as a stand-alone burst whereas a value of $15ms$ for b implies, in the case of our 100 Hz UDP flows, that all consecutive drops are considered as belonging to a single burst. The graphs plot the cumulative distribution function (CDF) of the estimate μ of the CD-DJ technique. The graphs show that there is not much difference between $55ms$ and $105ms$. The estimates μ are about 5% larger using $55ms$ as compared to using $15ms$. However, large burst intervals also lead to increased false positives as is shown in Figure 12(a). Notice that, with $55ms$, CD-DJ estimates μ to be as high as 0.6 in some cases. In contrast, μ does not exceed 0.2 for any case with $15ms$. We conclude that using a burst value b of $15ms$ performs best.

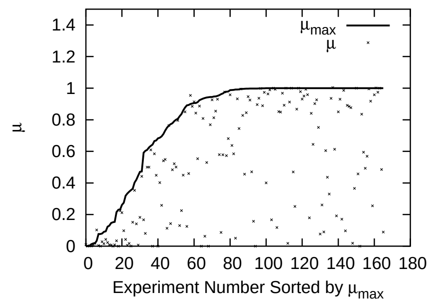
3) **Parameter Selection:** In Section III, we explained that we do not consider two bursts of drops to have occurred at the same PoC unless more than a fraction f of the drops occurred during their burst intervals. In Figures 11(c), 12(c) we show the results obtained with different values of the overlap ratio f. The trade-off involved in choosing f is that it should be small enough to consider all shared drops and large enough to not cause false positives. The results show exactly this behavior for 0.5: the case $f = 0.5$ is close to $f = 0.9$ for the $1 - I$ topology and close to $f = 0.1$ for the $1 - II$ topology.

4) **Sensitivity of Probing Rate:** Figures 14 and 15 show the sensitivity of CD-DJ to the probing rate. The 100Hz probing rate we used generated an overhead of 4KB/sec. In systems where this is prohibitive, probing with lesser frequency could be done with a penalty in accuracy as seen in the figures for the $1 - I$ and $1 - II$ topologies.

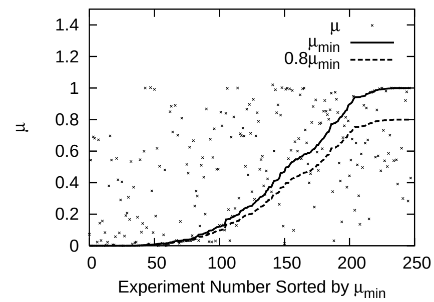
5) **Motivating Conservative Jitter Correlation:** As we discussed in Section III. there are cases when drops on a shared PoC may not be correlated. Hence we augmented the CD scheme (that correlates packet drops only) to include correlation of delay jitter (for



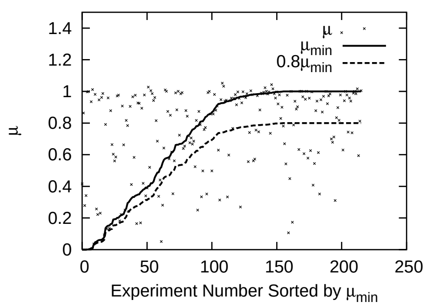
(a) 4-YV Topology Underestimation



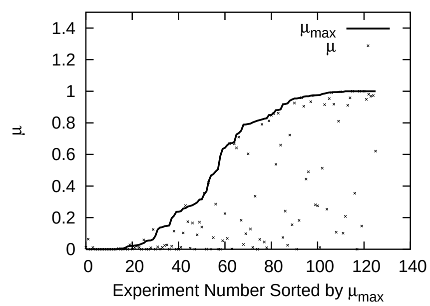
(b) 4-YV Topology Overestimation



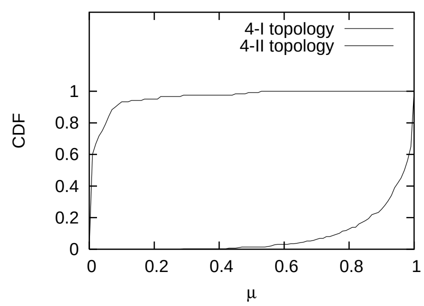
(c) 3-YV Underestimation



(d) 4-Y Underestimation



(e) 4-X Overestimation



(f) 4-I 4-II Topologies - CDF of μ

Fig. 13. Distribution of μ w.r.t μ_{min} and μ_{max} in multi-hop topologies

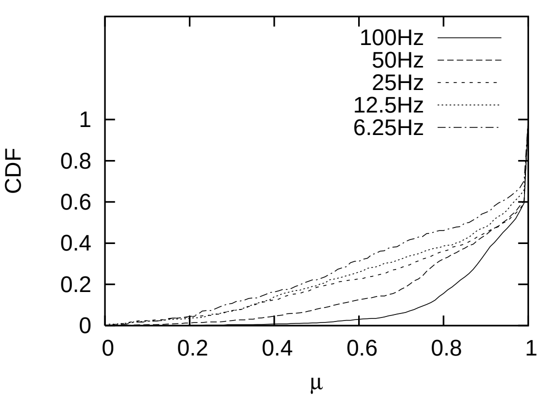


Fig. 14. Sensitivity of probing rate - $1-I$ topology

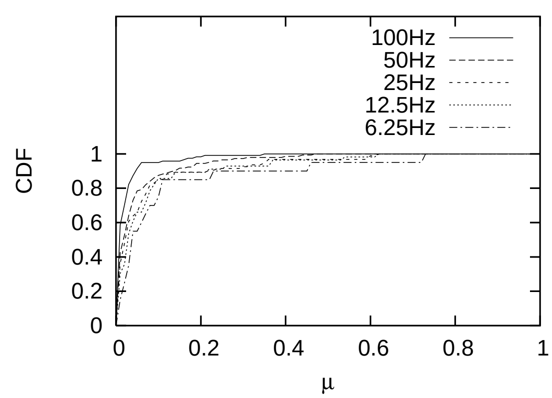


Fig. 15. Sensitivity of probing rate - $1-II$ topology

uncorrelated drops only) to derive the CD-DJ technique. Figures 11(b), 12(b) show the benefits of doing so. We see a clear improvement in the performance of CD-DJ over CD without any significant increase in false positive rate (as seen in the results for the $1-II$ topology). We believe that this is a result of our using delay jitter correlation conservatively.

B. Multi-hop Topologies - UDP Flows

In the previous subsection, we used the results of experiments conducted for two flows in the $1-I$ and $1-II$ topologies to study the validity of our assumptions and our technique. We now provide results of experiments conducted using multi-hop overlay topologies that enabled us to create flows that did not share all PoCs.

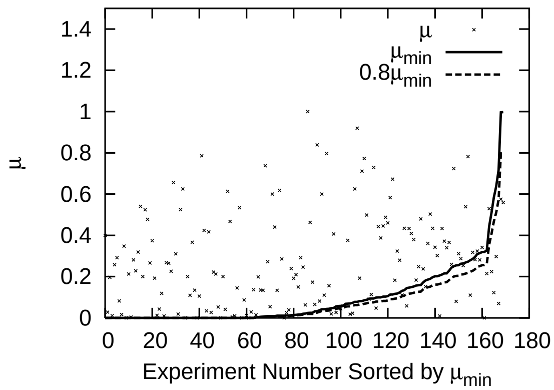


Fig. 16. Distribution of μ w.r.t μ_{min} and μ_{max} for TCP flows in the $3-VY$ topology

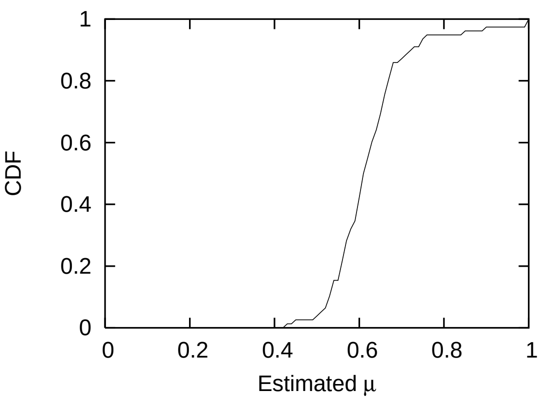


Fig. 17. CDF of μ for TCP flows in the $2-I$ topology

The results of the $1-I$ and $1-II$ topologies could be verified using the CDFs because in every experiment the actual fraction of shared drops $\hat{\mu}$ was a constant (1 and 0 respectively). With multi-hop topologies, we evaluated the CD-DJ technique by verifying that μ lies between $0.8\mu_{min}$ and μ_{max}. We used the 0.8 factor because of our definition of a successful experiment.

Figure 13 shows where μ lies in relation to $0.8\mu_{min}$ and μ_{max} for various multi-hop overlay-based topologies. All figures except Figure 13(f) show for each flow, the estimated value μ, the lower bound μ_{min} that we know from our overlay router traces. The graphs also show the $0.8\mu_{min}$ line, all points above which are deemed successful. Figure 13(f) plots the CDF of μ for the 4-hop topologies $4-I$ and $4-II$. In each of these graphs, the μ calculated in about 80% of the experiments were above $0.8\mu_{min}$.

C. TCP Flows

As we discussed earlier in Section IV, we used the $3-VY$ topology for evaluating the CD-DJ technique with TCP flows. The TCP flows we used were continuously backlogged flows that lived for $600s$. Since both TCP flows may not be active at all times, we did not consider the TCP flows during time intervals when either of them had been inactive (which means that it was not probing the network). The values of μ obtained as compared to $[\mu_{min}, \mu_{max}]$ for the experiments involving the TCP flows are shown in Figure 16. Since the μ_{max} for the $3-VY$ topology is 1, we do not plot the corresponding graph for it. For the $3-VY$ topology, we see that about 80% of the experiments resulted in an estimate that was at least $0.8\mu_{min}$.

Figure 17 shows the results of running the TCP flows using the $2-I$ topology which is the equivalent of the $2-I$ topology for the UDP flows. The graphs shows that most values of μ were between 0.55 and 0.7. This leads us to believe that CD-DJ may have to be tuned for TCP. Studying CD-DJ for TCP is our immediate future focus. Our efforts to collect TCP flow data for topologies with more number of hops were fruitless since the high RTTs involved caused very small amounts of data to be sent. In summary, our evaluation of TCP is limited and a different set of parameters may be required for them.

D. MPEG Experiments

We conducted simple experiments with the $2-VV$ and $3-O$ topologies as explained in Section IV. The purpose of these experiments was to show that video streaming which is more vulnerable to bursty losses produces smaller MSE with independent paths than dependent paths. Table III shows the results of four experiments involving the $2-VV$ and $3-O$ topologies. The MSE of the Y, U and V frames are shown in the table. The most relevant of the three MSEs is the MSE of the Y (luminance) frames. The rest are shown only for completeness sake. Since one of the lossy links has more losses, the number of drops of the independent set of paths is roughly the same for two servers of the same experiment unlike the total losses experienced by the dependent set of paths. The MSE values of the two pairs of paths also show the same behavior. The dependent paths produced better MPEG files when their lossy link was better than the other. However, the results illustrate that independent paths are better not because they will lead to better quality but because the likelihood of a bad quality MPEG is much less if they are used. In the case of all of our MPEG experiments, CD-DJ estimated μ

of the dependent paths to be 1 and independent paths to be 0. These experiments therefore prove that CD-DJ can improve the performance of media streaming applications. As we mentioned earlier in Section II, previous solutions cannot be applied since the MPEG application does not use a Y or Inverted-Y topology.

VI. Discussion

We now discuss miscellaneous issues relevant to the CD-DJ technique.

The following are properties of our technique that make it suitable for use in a distributed network monitoring infrastructure:

- **Asynchrony** - The CD-DJ technique does not require the two senders to be synchronized.
- **Easy Trace Collection** - Our technique uses information about the sending times (and jitter values) of packet drops only which requires simple measurements at the end hosts.
- **Online Computation** - Our technique is sufficiently lightweight for it to be implemented online. For the 10 minute experiments with a few thousand drops that we observed, our technique completed execution in much less than a second on a PIII machine.

As we mentioned in previous sections, we do not know of an easy way to verify our technique. We resorted to an overlay-based approach because evaluation at the IP level was next to impossible. Even with the overlay approach, the range $[\mu_{min}, \mu_{max}]$ is too large in some cases. It is an open question if we can do better given the same infrastructure.

In designing the CD-DJ technique, we assumed that the routers used droptail queues. Our scheme is likely to fail if routers begin to use RED as the queueing discipline in which case long-term metrics such as drop rate, throughput may have to be correlated. We mentioned the case of four sites that consistently gave bad results. We would like to determine the cause behind this. A strong possibility is the use of RED at some router on the route to these hosts. On a related note, an interesting application of our scheme would be to test the deployment of RED on a path.i

VII. Conclusions

Media streaming and other Internet applications can use independent paths i.e., paths that do not share a Point of Congestion to increase their tolerance to bursty losses. Existing solutions to the problem of detecting shared PoCs cannot be used since the applications do not meet the assumption of Y and Inverted-Y topologies made by these solutions. In this paper, we proposed CD-DJ a technique to estimate the amount of shared congestion between two paths in the Internet. CD-DJ can be used with the topologies of applications such as multipath media streaming and possess many novel features. These include the estimated of shared congestion instead of a "yes/no" output, the determination of synclag by maximizing the cross-correlation coefficient. We performed extensive evaluations, which have not been done earlier, using PlanetLab. Our results showed that, with UDP probes, CD-DJ provides an estimate that is within at least a fraction 0.8 of the actual value of shared congestion. Evaluating CD-DJ for TCP probe flows is not easy. Our results have been mixed. The results presented in this paper show that CD-DJ would provide estimates that are more than half the actual amount of shared congestion. Our immediate future work is to analyze CD-DJ with TCP flows more thoroughly and to build more applications that can use CD-DJ.

References

[1] D. G. Andersen, H. Balakrishnan, M. F. Kaashoek, and R. Morris, "Resilient overlay networks," in *Proceedings of ACM SOSP'01*, 2001.

[2] T. Nguyen and A. Zakhor, "Distributed video streaming with forward error correction," in *Packet Video Workshop*, Pittsburgh PA, USA, 2002.

[3] J. Apostolopoulos, T. Wong, W. tian Tan, and S. Wee, "On multiple description streaming with content delivery networks," in *Proceedings of IEEE INFOCOM'02*, July 2002.

[4] Y. J. Liang, E. G. Steinbach, and B. Girod, "Real-time voice communication over the internet using packet path diversity," in *Proceedings of ACM Multimedia*, Ottawa, Canada, 2001, pp. 431–440.

[5] H. Balakrishnan, H. Rahul, and S. Seshan, "An integrated congestion management architecture for internet hosts," in *Proc. of ACM SIGCOMM*, September 1999.

[6] Y. hua Chu, S. G. Rao, S. Seshan, and H. Zhang, "Enabling conferencing applications on the internet using an overlay multicast architecture," in *Proc. of ACM SIGCOMM*, August 2001.

[7] D. Rubenstein, J. F. Kurose, and D. F. Towsley, "Detecting shared congestion of flows via end-to-end measurement," in *Proceedings of ACM SIGMETRICS'00*, June 2000.

[8] D. Katabi and C. Blake, "Inferring congestion sharing and path characteristics for packet interarrival times," MIT-LCSTR-828, MIT LCS, Tech. Rep., December 2001.

[9] K. Harfoush, A. Bestavros, and J. Byers, "Robust identification of shared losses using end-to-end unicast probes," in *Proceeding of IEEE ICNP'00*, Osaka, Japan, October 2000.

[10] 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.

[11] PlanetLab, http://www.planet-lab.org.

[12] W. Jiang and H. Schulzrinne, "Modeling of packet loss and delay and their effect on real-time multimedia service quality," in *Proceedings of NOSSDAV*, 2000.

TABLE III

RESULTS OF MPEG EXPERIMENTS

Topology	Server Number	Losses on Dependent Paths	Losses on Independent Paths	MSE of Dependent Paths (Y,U,V)	MSE of Independent Paths (Y,U,V)
$2-VV$	1	1964	3892	2122,328,160	2558,1036,506
$2-VV$	2	5833	3893	4532,2303,964	2422,995,474
$2-VV$	1	362	1249	377,10,13	1414,39,40
$2-VV$	2	2104	1230	2078,70,83	1343,34,35
$3-O$	1	111	50	85,7,1	39,3,1
$3-O$	2	0	55	0,0,0	39,4,1
$3-O$	1	160	78	20,1,1	11,0,0
$3-O$	2	0	83	0,0,0	10,0,0

[13] H. Sanneck, G. Carle, and R. Koodli, "A framework model for packet loss metrics based on loss runlengths," in *Proceedings of SPIE/ACM SIGMM Multimedia Computing and Networking Conference*, January 2000.

[14] Y. Zhang, N. Duffield, V. Paxson, and S. Shenker, "On the constancy of internet path properties," in *Proceedings of ACM SIGCOMM Internet Measurement Workshop*, November 2001.

[15] M. Yajnik, S. B. Moon, J. F. Kurose, and D. F. Towsley, "Measurement and modeling of the temporal dependence in packet loss," in *Proceedings of IEEE INFOCOM'99*, 1999, pp. 345–352.

[16] V. N. Padmanabhan, L. Qiu, and H. Wang, "Server-based inference of internet performance," in *IEEE INFOCOM'03*, San Francisco, CA, USA, April 2003, (to appear).

[17] O. Younis and S. Fahmy, "On efficient on-line grouping of flows with shared bottlenecks at loaded servers," in *Proceeding of IEEE ICNP'02*, Paris, France, November 2002.

[18] M. Beck, T. Moore, and J. S. Plank, "An end-to-end approach to globally scalable network storage," in *Proceedings of ACM SIGCOMM'02*, Pittsburgh, PA, 2002.

[19] Kazaa, http://www.kaaza.com, 2002.

[20] tcpdump, http://www.tcpdump.org.

[21] Sprint ATL IP Monitoring Project, http://ipmon.sprint.com.

[22] Blanked for Anonymity.

[23] D. Hoffman, G. Fernando, V. Goyal, and M. Civanlar, "Rfc 2250: Rtp payload format for mpeg1/mpeg2 video."

[24] Fourcc, http://www.fourcc.org.