

Optimizing Video Streaming Against Transient Failures and Routing Instability

Gene Cheung
Hewlett Packard Laboratories

Chen-Nee Chuah
University of California, Davis

Dan-Jue Li
University of California, Davis

Abstract—In addition to network congestion, a link/node failure is another major cause of performance degradation for video streaming over the Internet. Such failures may be followed by a long routing instability period, during which packets can be black-holed due to invalid paths or caught in routing loops. This paper proposes a routing proxy approach to improve media streaming adaptation against both link/node failures and network congestion. In particular, we first argue that it is important to distinguish between network performance degradation due to network congestion versus link/node failures, and then model link/node failures using empirical models derived from measurements. We then show how by means of proper congestion control, such timely notifications from the network layer can be exploited at the streaming server to improve the performance of a rate-adaptive Automatic Retransmission reQuest (ARQ) video streaming scheme. Simulation results show that a rate-adaptive streaming scheme using feedbacks from our proposed proxy can recover much faster from link/node failures than a scheme without such feedbacks.

I. INTRODUCTION

There are two major causes of network performance degradation when streaming video over the Internet: network congestion and routing instability due to link/node failures. Past literature on video streaming over Internet has been focusing on coping with bandwidth variation and data losses due to network congestion [1]–[4]. However, recent studies have found that the level of congestion in core IP backbone is always negligible and has relatively small impact on latency sensitive applications such as Voice-over-IP (VoIP) [5]. On the other hand, link/node failures have been observed to be fairly common in the day to day operation of a network [6] due to fiber cuts, faulty equipments, or router misconfigurations. Re-routing after a link/node failure can take tens of seconds within a single domain [6], while inter-domain route changes through Border Gateway Protocol (BGP) can take up to minutes to converge [7]. During this transient period of route convergence, packets can be dropped because of invalid paths or caught in routing loops leading to additional delays. Such discontinuity in routing results in service disruptions at the application layer and can adversely affect the quality of real-time video streaming.

Our hypothesis is that if we can distinguish losses and delays due to link/node failure versus network congestion at the network layer, we can perform proper congestion control to provide the best video quality at the receiver. For example, a typical transmission strategy reduces video transmission rate after deducing network congestion from increased packet loss.

However, if losses are due to failures, then a better solution is to maintain sending rate but increase video packet protection using automatic retransmission request (ARQ), etc. Hence, it is important to re-evaluate the various optimized video streaming schemes by considering realistic Internet failure scenarios.

Our main contributions in this paper are twofold. First, we identify the importance of differentiating two causes of network performance degradation: link/node failures and network congestion. Two feasible mechanisms are discussed as to how the network infrastructure can obtain this differentiation and inform the application. We then construct realistic empirical models of failure patterns based on collected network traces. Second, we show how a version of the rate-distortion optimized streaming algorithm proposed in [4] can be modified to benefit from such network feedbacks. We performed simulation studies using ns-2.26 to evaluate the performance of the modified scheme under realistic failure scenarios. We compare its performance to a scheme that does not differentiate between congestion and failures.

The paper is organized as follows. Section II provides a general overview of the considered network infrastructure and discusses the two mechanisms for inferring network failures/congestion: (a) router-assisted, and (b) proxy-assisted. We then describe the adopted empirical models for link/node failures and routing instability. In Section III, beginning with a discussion on source model and objective measure, we present the rate-distortion optimized ARQ streaming scheme which we modify to rate-adapt based on link/node failure notifications from network layer. Our simulation results using ns-2.26 are discussed in Section IV. We conclude the paper and discuss future research directions in Section V.

II. SYSTEM OVERVIEW

We focus our discussion on non-interactive, server-client streaming services.

A. Inferring Network Failures and Congestion

To determine the cause of the performance degradation, we propose two mechanisms to infer the *loss-mode*—whether it is network congestion or link/node failure induced.

Router-assisted approach: The first mechanism relies on the deployment of active queue management at network routers to provide congestion indications through marking of packets instead of dropping them. This uses an Explicit Congestion Notification (ECN) [8] field in the IP header with two bits

making four ECN code-points, '00' to '11'¹. The video server first sets the ECN-Capable Transport (ECT) code-points '10' or '01' to indicate that the end-points of the transport protocol are ECN-capable. When the router's buffer is approaching full occupancy and the router is prepared to drop a packet to inform end nodes of incipient congestion, the router first checks to see if the ECT code-point is set in that packet's IP header. If so, then instead of dropping the packet, the router sets the Congestion Experienced (CE) code-point in the IP header ('11'). Slight modification is needed at the video client side to interpret ECN marking. We tag a one-bit *congestion notification* field in the RTCP report for the receiver to notify the sender when a packet with CE code-point is received. When a sender receives an RTCP report with *congestion notification* marked, it will attribute the cause of subsequent delays/losses to network congestion. On the other hand, if the *congestion notification* is missing, it *infers* that the losses are due to failures. The main disadvantage of the router-assisted approach is that it requires the cooperation of network routers. In addition, the loss of ECN-marked packets may cause the sender to inaccurately attribute network congestion induced losses to link/node failures.

Proxy-based approach: We propose to deploy routing proxies in the network to detect link or node failures directly. The proxy contains a listener software, such as Python Routing Toolkit (PyRT) [9] or Zebra [10], that allows it to receive routing messages from an adjacent network router. Since most network domains run link-state routing protocols, such as IS-IS [11] or OSPF [12], any changes in the routing topology (link/node addition or deletion) are flooded throughout the network, e.g., via Link-State Announcements (LSAs) in OSPF. The routing proxy associated with a video server keeps track of the end-to-end paths used for video streaming to its various clients. Such path information can be obtained by running *traceroute* from the video server to its client. When the proxy receives an LSA announcing a link or node failure that affects the path of the video streaming, it notifies the video server of the failure. This marks the start time of a routing instability period, which can take up to tens of seconds [6] before the network converges to the backup routing paths.

We have illustrated the availability of timely feedback mechanisms (Figure 1) to allow the video server to *infer* network loss-modes, but a detailed comparison of the two methods is out of scope of our discussion. The focus of our work is to model link/node failures and to evaluate video streaming strategies under realistic failure scenarios.

B. Empirical Models for Failures and Routing Loops

Previous empirical studies have shown that congestion losses can be modelled with low order Markov chains, and the number of lost packets in a loss period is approximately geometric [13]. However, the use of similar model for loss patterns during link/node failure and routing instability has not

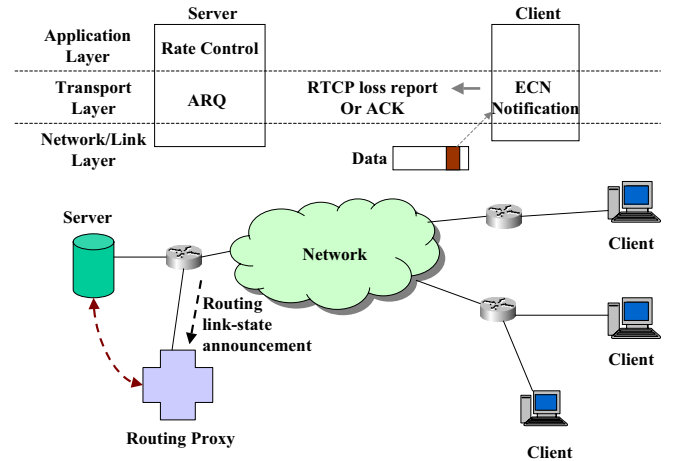


Fig. 1. Available feedback mechanisms to a video server.

been validated. When a link or node fails, it is often followed by a transient routing instability (or route re-convergence) period during which all network routers are notified of the failure, recompute their routing tables and reconfigure their forwarding paths. There are two distinct stages during the convergence period:

- **Black-out stage:** All packets traversing the failed link are dropped initially due to invalid forwarding path.
- **Routing-loop stage:** Subsequently, some of these packets may be caught in routing loops because of inconsistent forwarding tables at different routers. Routing loops from which packets do not escape contribute to an increase packet loss rate but have no effect on the delay performance. However, delay of packets that do escape a routing loop will be increased.

Once the routing protocol converges, the traffic will be forwarded on the backup path, which could be longer than the original path and result in increased end-to-end delay. After the link or node recovers, traffic forwarding will resume on the original path.

The distribution of the frequency and duration of link failures observed in a Tier-1 ISP backbone are reported in our previous work [6]. It was observed that failures are fairly well spread out across weeks, days, and even over the course of a single day. Clearly, they need to be taken into account as part of every day operations. The cumulative distribution of the duration of failures observed over the same period show that most failures are *transient* (i.e., short-lived): 50% last less than a minute and 85% last less than ten minutes. However, as discussed earlier, the actual failure duration is much less important since the packets will be forwarded correctly on the recomputed (alternate) paths once the routing protocol converges. Hence, we only concentrate on modeling the loss behavior during the *re-convergence period* following a failure, but not the failure duration itself. The experiments conducted in the same study indicate that this instability period can last

¹The four ECN code-points: Non-ECN-Capable Transport ('00'), ECN-Capable Transport ('01' and '10'), and Congestion Experienced ('11').

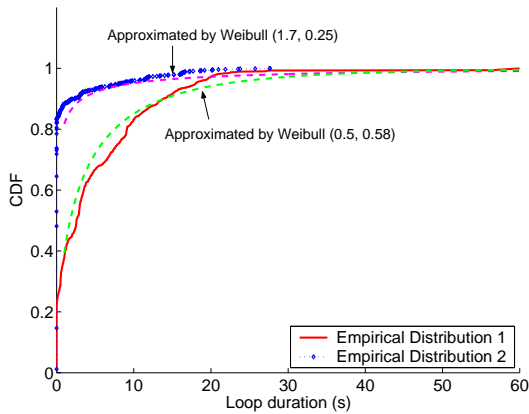


Fig. 2. Empirical distributions of routing loop durations that we consider in our simulation studies.

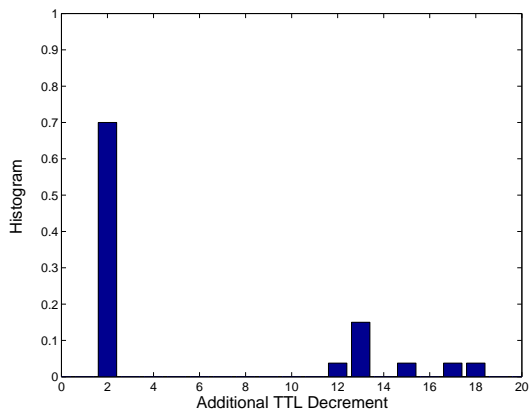


Fig. 3. Distributions of TTL decrements/loop for packets caught in routing loops.

between 2 to 6.6 seconds. This is in agreement with the parallel work that the duration of routing loops is mostly under 10 seconds [14].

For simplicity, we collapse the two stages and model the impact of network failures on losses and delays as follows:

- Immediately after a failure, packet loss burst duration is uniformly distributed between 1 to 2 seconds.
- This is followed by a transient routing loop with finite duration. The duration of routing loops is chosen based on empirical distributions reported in [14] (two examples are shown in Figure 2). We observe that the empirical CDF is well approximated by a Weibull distribution: $F(x) = 1 - \exp(-(x/\alpha)^\beta)$, $x \geq 0$. The Weibull parameters can be derived for each set of the empirical data based on maximum-likelihood estimation.
- If the routing loop duration is non-zero, the packets caught in the loop will traverse a random number of extra hops and hence experience extra delay before it is successfully delivered to its destination. Since the Time-To-Live (TTL) field of a packet is decrement by each router that it goes through, the additional TTL decrements (or extra hops) experienced by a packet caught in a

routing loop can be used to estimate the increase in end-to-end delay. We adopt the empirical distribution of the Time-to-Live (TTL) decrements observed in Hengartner's studies [14](Figure 3) and estimate the additional delay per hop as 25 ms. Hence, packets that manage to escape routing loops will incur between 50-500 ms extra delay.

III. RATE-DISTORTION OPTIMIZATION OF VIDEO STREAMING

Given the models for link/node failures, we now discuss how a rate-distortion optimized video streaming scheme can be modified to rate-adapt during link/node failure. We begin with a discussion on source model and objective measure, follow by a formalization of the streaming optimization problem.

A. Video Source Model and Quality Measure

Objective Quality Measure: We will use the most commonly used metric in the video processing literature [1]–[4], i.e., peak signal to noise ratio(PSNR), to evaluate the visual quality at the receiver. When the receiver is unable to decode frame i , the most recent correctly decoded frame j is used for display for frame i , and we can calculate the PSNR using original frame i and encoded frame j instead. If no such frame is available, the the PSNR for this frame is simply 0.

We assume there is a predictively coded (IPPP...) video sequence with a fixed I-frame frequency. We use the directed acyclic graph (DAG) based source model introduced in [4] to model the pre-coded video sequence. Each frame i is abstractly represented by one data unit (DU_i). For simplicity of discussion, we assume for now that each DU is transmitted in one RTP packet. Each DU_i is characterized by three numbers: delivery deadline T_i , size in byte B_i and reduction in distortion d_i , where d_i is defined by: $d_i = PSNR(i, i) + \sum_{j=i+1}^L PSNR(i, j) - \sum_{j=i}^L PSNR(i-1, j)$ for $i \geq 2$. $PSNR(j, i)$ is the PSNR between original frame i and encoded frame j . L is the last P-frame in the GOP. DU_i is correctly decoded if each DU_j , $k \leq j \leq i$, is correctly delivered by the corresponding delivery deadline T_j to the client, where DU_k is the most recent I-frame. If correctly decoded, DU_i reduces distortion at the client by d_i .

B. Problem Formulation

Using the network and source models derived previously, we now formalize the optimization problem for video streaming. Our discussion is loosely based on a simplified version of the rate-distortion optimized streaming framework (RaDiO) in [4].

At any given optimization instance T_{start} , an optimization window equal to M -frame time is selected. The window is defined to be the set of data units whose delivery deadline falls within start time $start(t)$ and end time $end(t)$. $start(t)$ brings data units into the optimization window; by keeping the window small, it keeps the optimization computationally feasible and the instantaneous client buffer small. $end(t)$ expires data units when they cannot be reasonably be expected to be delivered to the client on time. The slope of both functions — the rate at which the window advance in time

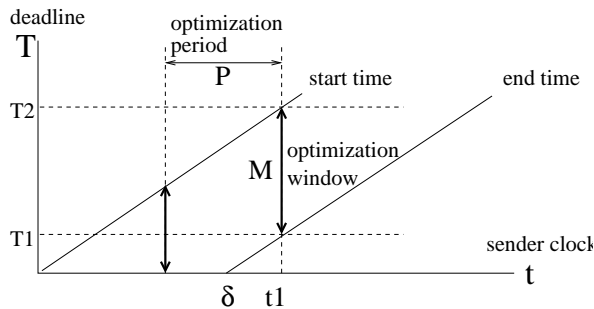


Fig. 4. Optimization Window.

— is the playback speed at the client. The optimization is performed again in P seconds (to be discussed). See Figure 4 for a plot of data unit deadline T against sender running time t .

C. Data Unit Selection Algorithm

Data units within the optimization window are selected for transmission as follows. For each DU_l in the window, we find the associated $\lambda_l = \lambda'_l S_l / B_l$. λ_l can be loosely interpreted as the *benefit* of delivering DU_l at this optimization instance. It is a function of both λ'_l , the increase in successful delivery likelihood of DU_l if one transmission is sent at this optimization instant, and S_l , the *sensitivity* of DU_l — the overall expected decrease in distortion if DU_l is successfully delivered. The top H data-units with the largest benefit value λ are selected at each round of the optimization.

In mathematical terms, we let π_l to be the number of transmissions of DU_l to date, and $\epsilon(\pi_l)$ be the loss probability of DU_l given π_l . If we let ϵ_R be the estimated end-to-end packet loss rate, we can write

$$\epsilon(\pi_l) = \begin{cases} 0 & \text{if client ACK received} \\ (\epsilon_R)^{\pi_l} & \text{o.w.} \end{cases} \quad (1)$$

We can now write λ'_l simply as follows:

$$\lambda'_l = \epsilon(\pi_l) - \epsilon(\pi_l + 1) \quad (2)$$

Using the above definitions and the source model, we can write the sensitivity S_l of DU_l as:

$$S_l = \sum_{l \preceq l'} \Delta D_{l'} \prod_{\substack{l'' \preceq l' \\ l'' \neq l}} (1 - \epsilon(\pi_{l''})) \quad (3)$$

where $\{l \preceq l'\}$ are the set of P-frames that depends on the correct decoding of DU_l , and $\{l'' \preceq l' | l'' \neq l\}$ are the set of frames that $DU_{l'}$ depends on for correct decoding, but not equal to DU_l .

D. Failure-aware Rate-adaptive Congestion Control

To be “TCP-friendly” and not claim more bandwidth than what a normal TCP connection would use under the same network condition, the transmitted data units will be spaced by T_s using a well-known TCP-friendly congestion control equation [15]:

$$T_s = \mu_R \sqrt{2\epsilon_R/3} + 3(\mu_R + 4\sigma_R)\epsilon_R(1 + 32\epsilon_R^2)\sqrt{3\epsilon_R/8} \quad (4)$$

where μ_R and σ_R^2 are the estimated mean and variance of RTT, respectively. After sending H data units, the optimization will be run again after $P = HT_s$ seconds with a new (and possibly overlapping) optimization window of data units.

As we learn from Section II-B, there are two stages following a link/node failure: *black-out stage* where a burst of consecutive packets are dropped, and *routing-loop stage* where packets are delayed due to routing loops. As such, black-out stage increases the estimate of the packet loss rate ϵ_R , and routing-loop stage increases the estimate of mean and variance of RTT, μ_R and σ_R^2 , respectively. This translates to a larger packet spacing using (4), resulting in a lower sending rate. As these are transient negative effects not attributed to network congestion, it would be unwise to include these contributions in the calculation of ϵ_R , μ_R and σ_R^2 . So a better strategy for the streaming server is to maintain the current observable statistics, and therefore the same sending rate, upon receipt of failure notification from the network layer, for a time duration equals to the expected combined length of the black-out stage and the routing-loop stage. After such time, normal operation will resume. We will show in Section IV that this rate-adaptive strategy enables the video stream to recover much faster than a scheme without such strategy.

IV. SIMULATION STUDIES

We use network simulator ns-2.26 to generate realistic failure scenarios, as described in Section II-B, and characterize the received video quality when different transmission strategies are used. We track successfully delivered video frame sequences in the presence of network failure and congestion, and then compare it with the original video streams to compute the PSNR.

A. Simulation framework

We consider the setup shown in Figure 1. All the links have equal capacity of 10^5 bps. The client will send an ACK packet back to the sender when the packet is received correctly. We also introduce a routing proxy node that periodically sends the video server control packets (notifications) when there is a network failure. To simulate a background level of network congestion, we increase the rate of background traffic to overload the link between the video server and the client.

Based on feedback control packets from the proxy, the video server will decide whether the network is in congestion or failure mode. We consider two transmission strategies: (a) failure-unaware and (b) failure-aware rate-adaptive ARQ scheme described in Section III. In case (a), the source will rate-adapt by interpreting all packet loss as network congestion. In case (b), the source will rate-adapt by distinguishing between network congestion and route failure.

B. Simulation Results

In this section, we present and discuss our simulation results. For source, two 300-frame standard video sequences, *foreman* and *container*, are encoded using H.263 version2 at QCIF, 30 frames per second and 120kps. The I-frame frequency in each sequence is 1 in 25 frames. For each

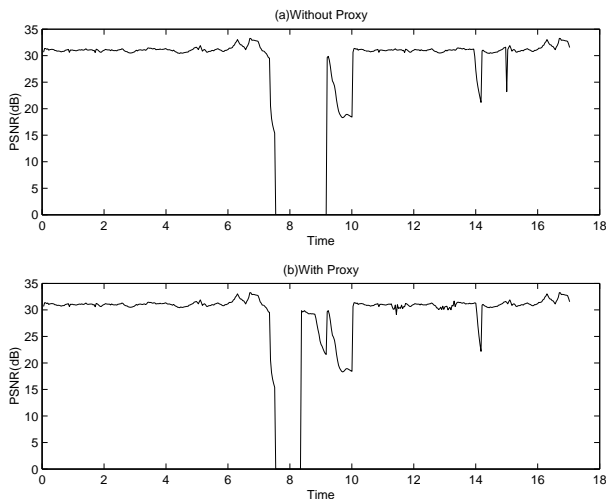


Fig. 5. 1 run of Streaming Simulation w/o and w/ Proxy ('foreman').

sequence	container	foreman
PSNR w/o Proxy w/ Failure	14.12	13.47
PSNR w/ Proxy w/ Failure	22.18	18.54
PSNR w/o Failure	34.31	30.70

Fig. 6. Average Streaming Performance w/o and w/ Proxy

sequence, PSNR between original frame i and reconstructed frame j is calculated for every combination i and j for $i \leq j$ and input into matrix $dArray[i, j]$. The matrix is then loaded into the simulated client in the simulator to estimate client's performance based on successfully received data units. We assume initial network parameters $(\epsilon_R, \mu_R, \sigma_R^2) = (3\%, 200ms, 0ms)$ before link/node failure which results in the initial packet spacing of $T_s = 30.2248ms$ using (4). The Weibull parameters (α, β) used in the calculation of the routing-loop stage duration are $(0.5, 0.58)$. Figure 5 shows the typical performance of the discussed optimized streaming scheme not using the proposed route proxy and the one using the proxy for rate adaptation. In this specific simulation run, we see that starting at $7.0s$, both schemes began to experience the effects of the black-out stage and PSNR in both cases was soon driven to zero. Because packet loss rate is heightened during blackout stage and RTT is lengthened during routing-loop stage, scheme (a), not knowing the effects were due to failure instead of congestion, had packet spacing unnecessarily enlarged using equation (4), and the low streaming performance persisted at the client for a long period. Scheme (b), on the other hand, knew the losses were due to failure and not congestion, and hence it did not perform unnecessary congestion control and recovered from the failure much quicker.

Figure 6 shows the mean PSNR of the two sequences for the two competing schemes in the failure scenario and the mean PSNR in the failure-free scenario. In the simulation, we used the same Weibull parameters and same network parameters as in previous part, averaged from the beginning of the blackout stage to 5 seconds after the end of the blackout stage. The

blackout stage lasts for $1.00s$ and the looping stage lasts $104ms$. We again see that because scheme (b) recognized the loss were due to failure instead of congestion, its quick recovery led to a much higher streaming performance than scheme (a).

V. CONCLUSIONS AND FUTURE WORK

We propose two approaches to differentiate network loss modes, i.e., whether it is failure or congestion induced, by inferring (router-assisted) or directly detecting (proxy-based) routing failures. We then present a rate-adaptive ARQ transmission scheme at the video server that employs routing-layer feedbacks to provide resiliency against both network failures and congestion. We formulate the problem using a rate-distortion optimization framework and considering realistic delay and loss patterns observed in a Tier-1 IP backbone. Numerical investigations through ns-2.26 simulations show that our optimal scheme can recover much faster from routing failures than unprotected scheme by avoiding the unnecessary rate-reduction introduced by blind congestion control. We intend to generalize our analytical model in the near future to include the effects of different source coding schemes, e.g., distributed scalable coding and multiple description coding.

ACKNOWLEDGMENTS

The authors are grateful to S. Moon for sharing the empirical distributions of routing loops and packet delay patterns observed in her paper. This research was supported by funding from Hewlett Packard and CITRIS.

REFERENCES

- [1] W. Tan and A. Zakhor, "Real-time internet video using error resilient scalable compression and tcp-friendly transport protocol," *IEEE Transactions on Multimedia*, vol. 1, pp. 172–186, June 2000.
- [2] Z. Miao and A. Ortega, "Optimal scheduling for streaming of scalable media," in *Asilomar*, 2000.
- [3] P. Frossard and O. Verscheure, "Joint source/fec rate selection for quality-optimal mpeg-2 video delivery," *IEEE Transactions on Image Processing*, vol. 10, pp. 1815–1825, December 2001.
- [4] P. A. Chou and Z. Miao, "Rate-distortion optimized streaming of packetized media," February 2001. Microsoft Research Technical Report MSR-TR-2001-35, Submitted to IEEE Trans. MM.
- [5] C. Boutremans, G. Iannaccone, and C. Diot, "Impact of link failures on VoIP performance," in *ACM NOSSDAV*, May 2002.
- [6] G. Iannaccone, C.-N. Chuah, R. Mortier, S. Bhattacharyya, and C. Diot, "Analysis of link failures in an IP backbone," in *Proc. ACM SIGCOMM Internet Measurement Workshop*, November 2002.
- [7] C. Labovitz, A. Ahuja, A. Abose, and F. Jahanian, "An experimental study of bgp convergence," in *ACM SIGCOMM*, 2000.
- [8] K. K. Ramakrishnan, S. Floyd, and D. Black, "The addition of explicit congestion notification (ECN) to IP," September 2001. IETF RFC 3168.
- [9] "Python Routeing Toolkit." <http://www.cl.cam.ac.uk/rmm1002/pyrt.html>.
- [10] "GNU Zebra, free routing software." <http://www.zebra.org/>.
- [11] D. Oran, "OSI IS-IS intra-domain routing protocol." RFC 1142, February 1990.
- [12] J. Moy, "OSPF Version 2." RFC 2328, April 1998.
- [13] M. Yajnik, S. Moon, J. Kurose, and D. Towsley, "Measurement and modelling of temporal dependence in packet loss," in *IEEE INFOCOM*, March 1999.
- [14] U. Hengartner, S. B. Moon, R. Mortier, and C. Diot, "Detection and analysis of routing loops in packet traces," in *Proc. ACM SIGCOMM Internet Measurement Workshop*, (Marseilles, France), November 2002.
- [15] S. Floyd, M. Handley, J. Padhye, and J. Widmer, "Equation-based congestion control for unicast applications," in *ACM SIGCOMM*, 2000.