Oscillations in a Bio-Inspired Routing Algorithm

Prof. Erol Gelenbe, Dr. Michael Gellman
Dept. of Electrical & Electronic Eng.,
Imperial College London
Email: {e.gelenbe,m.gellman}@imperial.ac.uk

***Abstract*— Adaptive routing is once again becoming of interest because of the possibility to couple on-line probing in networks with real-time dynamic and distributed control of paths and flows using the biologically-inspired Reinforcement Learning. Wireless networks, with their rapidly changing network conditions also create a need to revisit this issue. This paper uses measurements in a wired, bio-inspired adaptive network test-bed, the Cognitive Packet Network (CPN), to investigate the pros and cons of adaptive routing. CPN routes packet flows through a store and forward network according to their Quality of Service (QoS) needs through an on-line, distributed reinforcement learning mechanism. This paper investigates routing oscillations which occur due to the interaction of multiple flows and studies their effect on QoS in the context of CPN. Our results indicate that routing oscillations can be easily controlled by randomising the route switching, and that from an overall QoS viewpoint increased switching can also lead to improved performance.**

I. Introduction

Reinforcement learning is a key biological mechanism that enables people and animals to explore their environment and exploit decisions that lead to the greatest reward. We use the same principles in our routing algorithm, the Cognitive Packet Network [11], to learn the best paths in a network and use them to route data packets. Previous work has tended to look at how the reinforcement learning algorithm works with one flow in the network; this work examines the impact of multiple flows on each other, leading to what are known as *routing oscillations*.

Routing oscillations are a networking phenomenon that have been observed since some of the earliest packet-switched networks (such as the ARPANet) where they caused performance to suffer under medium to high load [14]. These oscillations were due to the use of a load-sensitive routing metric, which has led to a commonly taken-for-granted assumption among researchers that routing protocols should ensure that oscillations do not occur, and furthermore, load-sensitive metrics should not be used in such a way that they could occur. We set out to test this assumption using an adaptive routing protocol, with some surprising results which question whether it still holds.

This work was supported by the UK Engineering and Physical Sciences Research Council under Grant GR/S52360/01.

The routing protocol in question is the Cognitive Packet Network (CPN) [11], which is a biologically inspired packet routing protocol that uses on-line measurement and reinforcement learning to adapt its routing to changing traffic or network conditions. It constantly probes paths in the network using smart packets, and selects the "best" one for a given flow in a distributed manner based on a user-specified QoS metric. It has been experimentally shown in previous work [8], [13] that CPN can effectively offer best-effort QoS routing according to metrics that are selected for specific application's needs. Being an *adaptive* routing protocol, we set out to question how routing oscillations impact performance, and what methods exist towards controlling the rate at which routes oscillate.

In this paper, we challenge the assumption that routing oscillations always result in poor network performance. Using experiments conducted with CPN on a large networking testbed configured with real-world topology data, we show that increased switching does not always result in poor performance, and can actually result in improved QoS for network flows. While an increase in route-switching results in a corresponding increase in packet desequencing, we demonstrate two factors which function to trade-off between network performance (i.e. delay and loss) and desequencing.

We begin our paper with an overview of the CPN protocol, focusing on its reinforcement learning-based approach to routing (Section II). In Section III, we show that CPN in effect can *benefit* from frequent route switching, yielding its best performance when it is allowed to switch whenever it finds a better-performing route. This observation seems to indicate that myopic and selfish behaviour of users in a network, leading to frequent "changes of mind" can under certain circumstances result in better performance for all parties concerned. This affirmation needs to be tempered when oscillations can have other side effects, such as packet desequencing for flows that are highly sensitive to sequence, such as real-time video or voice, or even TCP traffic (Section III-C).

A. Related Work

Routing oscillations are hardly a new phenomenon, and there have been many contributions to the subject in the field of networks. The ARPANET, one of the predecessors of the modern Internet was susceptible to routing oscillations under heavy load [14] which led to poor performance. The effects of Internet routing instability were discussed in [15], [16] which summarized their impact by saying that "Overall, instability has three primary effects: increased packet loss to unstable destination, delays in the time for network convergence, and additional overhead (memory, CPU, etc.) within the Internet infrastructure." A discussion of the negative impact of synchronization of periodic routing messages is found in [5] who show that even if traffic sources are initially not synchronized, they can abruptly become synchronized. In [18] route instability is discussed in the presence of network congestion, causing increased packet loss and latency.

The negative impact upon TCP performance of disturbances at the routing layer is examined in [17] which considers both the case of routing oscillations due to the self-load effect and link failures. TCP is affected by these events because of how it responds to asymmetric paths (i.e. the path that its data packets take is different from those of its ACKs), and out-of-order packet delivery.

Routing oscillations in Intelligent Route Control (IRC) systems have been explored in [6] through simulation. They define IRC systems as aiming "to optimize the cost and performance of outgoing traffic, based on measurement-driven dynamic path switching techniques". CPN also carries out path switching as needed for QoS purposes based on dynamic measurement and control, and (as previously described) it constantly explores the network state using SPs.

In [6], two causes of the oscillations are identified. The first is what they term the *self-load* effect, where the impact of a flow on the metric that is measured is not taken into account in the probing process so that once a path whose load appears to be light is actually used, the resulting load is significantly higher than the one that was previously observed. This first phenomenon has also been studied in [1] with respect to the use of CPN in admission control algorithms. Of more difficulty is the second cause they outline: that routers' measurement windows can overlap, leading to persistent switching as flows interfere with each other, preventing the network from stabilizing, and causing uneven utilization and poor QoS.

II. The Cognitive Packet Network

In this section, we briefly present an overview of the Cognitive Packet Network (CPN) routing protocol as described in [10], focusing on the on-line path searching algorithm and on the manner with which it allows edge routers to source route the traffic. A more detailed description can be found in [11], [10].

As opposed to routing protocols which rely on pre-computing routes to all possible destinations based on a fixed criterion, such as the choice of paths which are of minimum length, CPN works in an on-demand fashion, dividing the routing task among three distinct classes of packets: Smart (SP), Acknowledgement (ACK), and Dumb (DP) packets. The SP's role is simply to discover paths, while DPs carry payload and are source routed. ACKs bring back information that SPs have discovered so that both intermediate routers and the source router can update the information that they have.

In CPN, communication sources (which are supported by edge routers) create SPs at regular intervals, whose final destination is the address of the edge router which supports the communication user's destination. SPs contain the destination node identifier and the QoS criterion that they seek, and are routed in a distributed manner without any form of centralized control. Each intermediate node decides on the next node for a SP based on the final destination address and the QoS criterion. The intermediate router's decision is taken by consulting a neural network whose weights have been updated with previous measurement data on routes to that destination for the same QoS criterion, using a reinforcement learning algorithm described in [11]. A fraction of SPs are routed at random at each intermediate router so that a wider range of paths may be discovered. SPs collect both the router's identity, and data relevant to QoS at each router visited.

For instance, if the relevant QoS metric is "delay" the SP will collect the local (not global) time at which it entered the node, so that if and when it reaches its final destination it will contain a list of all nodes visited together with the times of the corresponding visits. The destination will then generate an ACK packet, to be sent back to the source along the reverse of the path that the SP followed so that all intermediate nodes may be visited. The ACK packet would contain all the information stored in the SP so that when the ACK visits a node, the current local time minus the time when the SP visited that same node in that order can be used as the "penalty" of the reinforcement learning algorithm.

Each ACK that arrives at the source contains a complete route from the source to the destination, along with its QoS; for instance in the above example when QoS is delay, the ACK will have the time at which the SP was generated at the source, and the current time when the ACK is received by the source provides an estimate of the round-trip delay from source to destination when that path is used. Paths are stored at the source, and allow it to select among all recently tested paths the one that is "best" and which it will use to forward its DPs. The source can revise its decision more or less frequently based on the ACKs that it receives, and based on the difference between the "best" QoS path, and the one it may be currently using. Thus the source node is free to reduce path switching when the perceived improvement in QoS does not exceed a given threshold.

A. The CPN Reinforcement Learning Algorithm (RL)

In this section we will detail the RL algorithm that is implemented in CPN. Each CPN router contains a (fully recurrent) Random Neural Network (RNN) [7], [9] whose weights are updated using RL. The RNN functions as a decision maker where each neuron corresponds to the choice of one of the possible output links of the router, and at any time the most excited neuron is selected to indicate the output port that the current SP should take.

The RNN weights are updated whenever an ACK visits the router. From the QoS information (loss, delay, etc.) brought back by an ACK, the relevant information is stored in the router it is currently visiting and a new value of the "reward" R is calculated. Successive values of R are denoted as R_l, $l = 1, 2, \ldots$. These are used to compute a *threshold* or historical value of the reward

$$T_l = \alpha T_{l-1} + (1 - \alpha) R_l \qquad (1)$$

where α is some constant close to (but less than) 1.

Thus T_{l-1} represents our *expectation of reward* so that if $R_l > T_{l-1}$, we *reward* the previous decision by increasing the excitatory weights leading to its corresponding neuron, and slightly increase the inhibitory weights of the other neurons. On the other hand, if the reward is below the threshold, $R_l < T_{l-1}$, than we *punish* the decision by increasing its inhibitory weights, and also slightly increasing the excitatory weights of the other neurons so that the other choices for output ports can compete more effectively in the next round of decision making.

B. Switching routes at the source

We incorporate two additional parameters which influence when the source switches to a new route. The first of these enhancements (also described in [6] though not explicitly used to control oscillations) we call a *reward threshold*, where we specify that a newly discovered route must be better than the active route by a fixed percentage in order for a switch to occur (this is a similar approach to the one used in [12]). The second control we use is a *fixed switching probability* (FSP) which was also used in [6]. When a new route is discovered which is better by the reward threshold than the active one then with some probability it will cause a switch. As an example, in traditional CPN the FSP is equal to 1, whereas if no switching were allowed, the FSP would be 0.

III. Oscillations under stationary load

We wanted to conduct experiments using a realistic environment where routing oscillations would be observed and their impact on the performance of a large number of flows could be observed. In this section we experimentally investigate the assumption that oscillations result in poor performance, with surprising results.

A. Experimental setup

Our networking testbed consists of 46 Pentium IV-class machines, each equipped with one (or more) 4-port 10/100 Ethernet interfaces. The Operating System is Linux 2.6.15, where CPN is implemented as a loadable kernel module. Each link is full-duplex, and is configured to run at 10Mb/s. The topology we use is based on information we received about the Swiss Education & Research Network depicted in Figure 1.

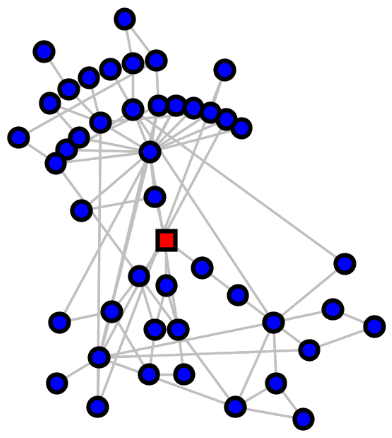

Fig. 1. CPN testbed topology constructed using data from the Swiss Education & Research Network. The square node is the traffic sink in the experiments, and 24 of the remaining nodes act as traffic sources.

Our experiments use 24 constant bit-rate flows, each of $1.66Mb/s$, which makes for (just above) $40Mb/s$ of application data (not including SP and ACK overhead). This yields similar parameters to the 105% of capacity case studied in [6]. Each client sends traffic to the same destination, which has 4 ingress interfaces that provide it with $40Mb/s$ of available incoming bandwidth. For each Dumb packet sent, the probability of generating a Smart packet is 0.1. In order to estimate the impact of a unresponsive traffic, we configured this probability to be 0 for two of the flows (this matches the 90% case in [6]). Each CPN flow is configured to optimize delay and each individual experiment lasted 15 minutes.

B. Impact of Switching on Performance

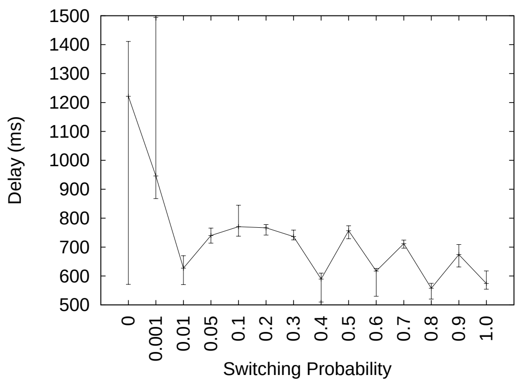

Fig. 2. CPN's performance when increasing switching. The average delay of each client is computed individually, and the median is plotted in the graph. Error bars indicate the 1st and 3rd quartile.

In order to study the extent to which CPN's performance is affected by routing oscillations, we began by experimenting with the fixed switching probability (FSP) parameter. When the switching probability is zero (i.e. the path of every flow is chosen at random at the beginning of the experiment, and then never switched), the delay is at its highest. As CPN is allowed more freedom to switch routes, it is able to better optimize its performance, yielding a median value half that obtained without switching.

We wanted to confirm that the increase in the switching probability corresponded to an increase in not only switching, but also oscillations. We defined an oscillation as a special pattern of route switching where a route is used, then another, then back to the original route. For instance, if route A is used, followed by route B, and then by route A again, we define that as a single oscillation. The rate of these occurrences over the experiment length is plotted in Figure 3. As the switching probability increases so does the rate at which the routes oscillate.

When taking Figures 2 and 3 together, we can see some interesting relationships. The difference in performance between probabilities 0.01 and 1 is not very dramatic, yet their rate of oscillation differs by a factor of 100. Thus, our results are not indicating that oscillations are *necessary* for good performance – rather we are saying that they do not degrade performance in our network.

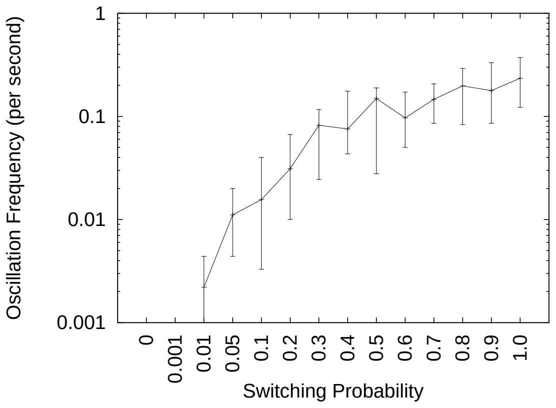

Fig. 3. Oscillation Frequency as a function of FSP

C. Packet Desequencing

One of the disadvantages to frequent path switching is that it can cause packets to arrive at the destination in a different order from which they were generated. This desequencing negatively impacts many traffic types' performance. The negative impact of reordering on a TCP connection is discussed at length in [3], and it is an on-going research activity to design a TCP implementation that can withstand desequencing without suffering a performance penalty [4]. It is also well-known that real-time traffic such as Voice over IP needs its packets to be received in-order, or else they will be buffered, increasing the memory and processing requirements at the destination, and, in cases of excessive desequencing, it leads to dropped packets and poor performance.

Because of the negative impact on application performance, we measured the amount of desequencing as a result of increasing the switching probability. We make the assumption that our client streams each represent a single flow of real-time traffic. While in our traces, the application has not used a monotonically increasing sequence number, we can deduce the order in which each packet was generated by looking at the *id* field, which is the time-stamp at the source when the packet was originally sent. By considering the sequence of received *id*s, we can reconstruct the order in which the packets were generated, and, in turn, the amount to which it became desequenced by the network.

In order to quantify the level of desequencing, and in keeping with our focus on real-time traffic, our metric use a *reordering buffer* that acts to store out-of-order packets. When an out-of-order packet arrives at the destination[1], it is put into a buffer until it can be freed by the arrival of the expected sequence number. If an attempt is made to add an arriving packet to a full buffer, then it is assumed that the expected packet has been lost, and the packet with the lowest sequence number in the reorder buffer becomes the new expected packet. This acts to timeout packets which arrive excessively late. Packets arriving with sequence number less than the expected one are simply dropped. The reorder buffer has the ability to smooth out small amounts of desequencing, but, under higher levels it can lead to packet drops. This approach is similar to the Reorder Buffer Density metric proposed in [2], except we are concerned with the number of packets dropped by the buffer - not its size distribution.

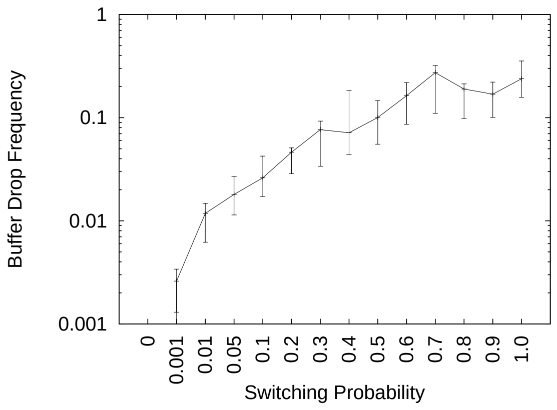

Fig. 4. Impact of switching on desequencing. The number of packets which would be dropped by a reorder buffer increases with the switching rate.

For a reorder buffer of size 10, we obtained the drop frequencies in Figure 4. As the switching frequency increases, the drop rate quickly exceeds the tolerable limits of many applications.

D. Reward Threshold

So far, we have demonstrated the impact that the switching probability has on performance, both in terms of delay and desequencing. Another factor which has a large impact on CPN's ability to select the best-performing routes is the reward threshold which we introduced in Section II-B (which we refer to as simply as *threshold* for the rest of the paper), which is how much better the estimated reward for a newly discovered route must be than the active route in order to result in a route switch. We express the threshold in terms of a percentage; thus, the scenario where a new route would have to be twice as good as the current route corresponds to a threshold value of 100%. Here, we discuss the impact that this threshold has both on performance and oscillations.

The threshold serves as a parameter which can be tuned to make CPN more or less sensitive to improvements in route quality. If the value is too low, any improvement in route quality, no matter how small, will result in a route change. This can lead to unnecessary route switches due to, for instance, the *self-load effect* where the Smart packets of a flow do not impose the entire load of the flow on a path, thereby seeing improved levels of QoS that are not experienced by the flow after it switches, subsequently leading to further switches and high rates of oscillation. Similarly, if the threshold is too high, CPN will not be sensitive enough and will not take advantage of routes which offer improved performance. Thus reduced switching may also lead to reduced performance. Both of these phenomenon are observed in Figure 5. There appears to be a definite value of the relative threshold, in this case around 14.3% of the current value, which provides better performance than with greater or smaller values.

When Figure 5 is considered along with the rate of oscillations in Figure 6 we can confirm our analysis above, but also draw a stronger relationship between oscillations and performance. When the threshold is 0 the oscillation rate is nearly 2 times per second, a factor of 10 increase over all the other switching values. Similarly, at the highest threshold value, the oscillation rate is at its lowest. It is when we analyse these two graphs in tandem however, that we see some interesting trends. First of all, for oscillation rates that are very nearly identical, we see significant performance differences. This tells us that the rate of oscillation is not by itself enough to impact performance, and that there are other factors at work, corroborating our analysis of the results in Section III-B.

IV. Summary

This paper has examined routing oscillations and their impact on performance, based on experiments on an adaptive network test-bed which uses the CPN routing protocol. We configured our test-bed network as realistically as possible given our resource (total number of nodes and links) constraints. Our measurements indicated that the CPN routing protocol provided somewhat different results from those that we would have expected based on long-held views (dating back to the ARPANet), that path adaptation will lead to oscillations and that this will result in poor performance under medium to

[1]Any arriving packet having a sequence number greater than the expected packet is out-of-order.

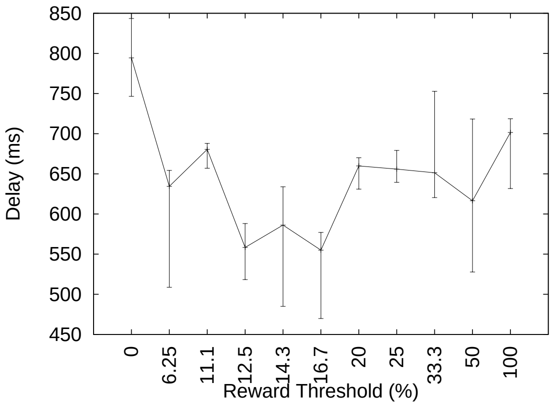

Fig. 5. Impact of reward threshold on delay

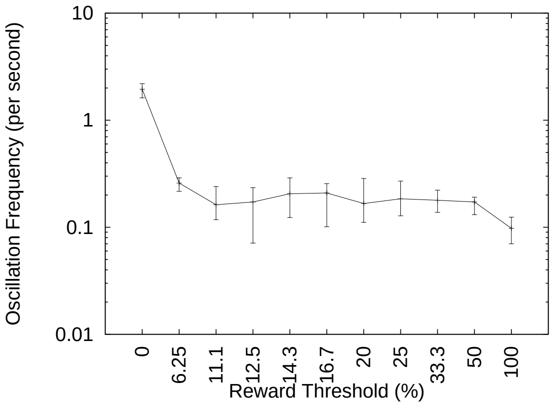

Fig. 6. Reward threshold's impact on oscillation rate

high load. Our results indicate that routing oscillations do not severely degrade performance as would be expected; rather we show that even when they are present we can still obtain high performance.

We also examined how oscillations can be controlled in the network, and tested two different parameters which largely impact the rate at which oscillations are observed. In particular we studied the use of probabilistic path switching we can be used both to make path switching more asynchronous, and to vary the rate at which switching decisions are made. We also examined the value of introducing a decision threshold which will only allow path switching if the gain expected from switching exceeds a certain minimal value. Both of these controls schemes are easy to implement, and provide an effective way to limit oscillations and their negative consequences.

More generally, we feel that this paper brings into question whether some long-held assumptions in the networking community regarding the viability of load-sensitive routing metrics are applicable for new classes of routing protocols which are based on self-monitoring and adaptation such as CPN.

References

[1] M. D' Arienzo, E. Gelenbe, and G. Sakellari. Admission control in self aware networks. In *49th annual IEEE Global Telecommunications Conference, GLOBECOM 2006, San Francisco*, November 2006.

[2] T. Banka, A.A. Bare, and A.P. Jayasumana. Metrics for degree of reordering in packet sequences. In *Local Computer Networks, 2002. Proceedings. LCN 2002. 27th Annual IEEE Conference on*, pages 333–342, November 2002.

[3] Jon C. R. Bennett, Craig Partridge, and Nicholas Shectman. Packet reordering is not pathological network behavior. *IEEE/ACM Trans. Netw.*, 7(6):789–798, 1999.

[4] S. Bohacek, J.P. Hespanha, Junsoo Lee, C. Lim, and K. Obraczka. A new TCP for persistent packet reordering. *IEEE/ACM Transactions on Networking*, 14(2):369–382, April 2006.

[5] Sally Floyd and Van Jacobson. The synchronization of periodic routing messages. *IEEE/ACM Trans. Netw.*, 2(2):122–136, 1994.

[6] Ruomei Gao, Contantine Dovrolis, and Ellen Zegura. Avoiding oscillations due to intelligent route control systems. In *IEEE Infocom*, 2006.

[7] E. Gelenbe. Random neural networks with negative and positive signals and product form solution. *Neural Computation*, 2:239–247, Feburary 1990.

[8] E. Gelenbe, M. Gellman, R. Lent, P. Liu, and Pu Su. Autonomous smart routing for network QoS. In *Proceedings of the First International Conference on Autonomic Computing*, pages 232–239, New York, NY, May 2004.

[9] E. Gelenbe and K. Hussain. Learning in the multiple class random neural network. *IEEE Transactions on Neural Networks*, 13(6):1257–1267, November 2002.

[10] E. Gelenbe, R. Lent, and A. Nunez. Self-aware networks and QoS. *Proceedings of the IEEE*, 92(9):1478–1489, September 2004.

[11] E. Gelenbe, R. Lent, and Z. Xu. Measurement and performance of a cognitive packet network. *Journal of Computer Networks*, 37:691–791, 2001.

[12] E. Gelenbe, P. Liu, and J. Laine. Genetic algorithms for autonomic route discovery. In *Proceedings of IEEE 2006 Workshop on Distributed Intelligent Systems*, June 2006.

[13] M. Gellman and P. Su. Using adaptive routing to achieve quality of service. *Performance Evaluation*, 57(2):105–119, June 2004.

[14] A. Khanna and J. Zinky. The revised arpanet routing metric. *SIGCOMM Comput. Commun. Rev.*, 19(4):45–56, 1989.

[15] C. Labovitz, G.R. Malan, and F. Jahanian. Internet routing instability. *IEEE/ACM Transactions on Networking*, 6(5):515–528, 1998.

[16] C. Labovitz, G.R. Malan, and F. Jahanian. Origins of internet routing instability. In *INFOCOM '99. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, volume 1, pages 218–226, New York, NY, 1999.

[17] U. Ranadive and D. Medhi. Some observations on the effect of route fluctuation and network link failure on TCP. In *Computer Communications and Networks, 2001. Proceedings. Tenth International Conference on*, pages 460–467, Scottsdale, AZ, 2001.

[18] Aman Shaikh, Anujan Varma, Lampros Kalampoukas, and Rohit Dube. Routing stability in congested networks: experimentation and analysis. In *SIGCOMM '00: Proceedings of the conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, pages 163–174, New York, NY, USA, 2000. ACM Press.