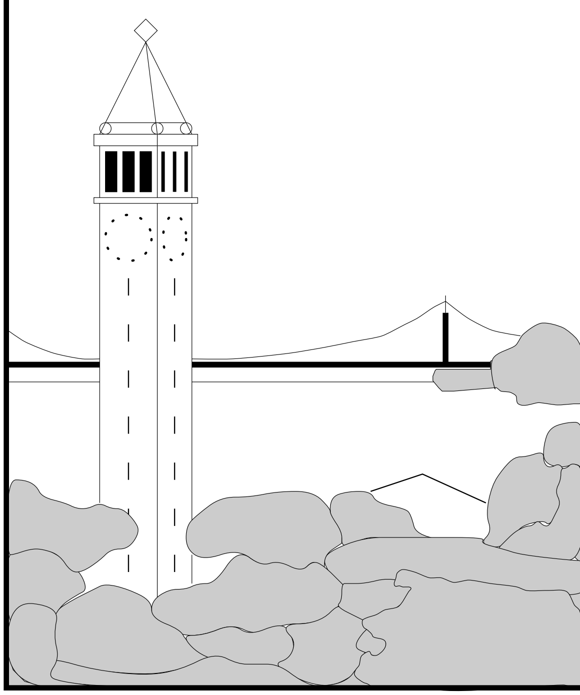
Dynamics of Simultaneous Overlay Network Routing

Mukund Seshadri and Randy H. Katz



Report No. UCB//CSD-03-1291

November 2003

Computer Science Division (EECS)
University of California
Berkeley, California 94720

This research was supported under grant number (NSF)EIA-0122599.

Dynamics of Simultaneous Overlay Network Routing

Mukund Seshadri, Randy H. Katz
EECS Department, University of California, Berkeley
Email: {mukunds,randy}@cs.berkeley.edu

***Abstract*— Peer-to-peer and overlay networks allow routing to be controlled at the application layer. Consider several *independent* overlay flows, each with a set of available overlay routes to send their data on. If they each select the route with the most available bandwidth, i.e., they are "greedy", a significant degree of instability could result, leading to degraded performance. We investigate this possibility, and a wide variety of factors that affect routing performance, by simulations. We find that some measure of "restraint" is crucial for obtaining acceptable performance of route selection in such scenarios. Specifically, we investigate three forms of restraint - randomization of route selection, utilizing an appropriate hysteresis threshold when switching routes, and increasing the time intervals between route-change considerations.**

Our results indicate that randomization can significantly reduce loss-rates (typically by half) - more importantly, it is sufficient to utilize load information from a small subset of overlay paths to obtain such results. This approach would significantly reduce the path measurement overhead imposed by applications. Secondly, we find that appropriate values of the hysteresis threshold (H) can be heavily dependent on the parameters of the system. Therefore we propose that flows determine H dynamically; we suggest and evaluate an algorithm based on multiplicative increase and decrease of H for this purpose. This algorithm is found to reduce loss rates of the basic greedy method by at least half. Finally, we investigate the scenario when a subset of unrestrained overlay flows (the "cheaters") select the best of all available routes, while the remainder use suitable hysteresis thresholds or randomization.

I. INTRODUCTION

Peer-to-peer systems and overlay networks allow data or queries to be routed through peers or overlay hosts to the ultimate destination. Such systems often possess a degree of freedom in choosing the overlay-level path. This allows applications to control the performance of their routes, and seek better routes or greater functionality than is available from the Internet. This approach is convenient since it does not require changes in the IP infrastructure. Several overlay networks and peer-to-peer systems have been proposed which follow this approach. Some examples are Gnutella [1] (a peer-to-peer overlay for file-sharing), the Resilient Overlay Network [2] or RON (which attempts to provide better connectivity than BGP does), and End System Multicast (which attempts to provide efficient multicast capability using only end-hosts). In addition, proposed infrastructures like *i3* [3] would make it easier for end-hosts to control routing [4], and testbeds like PlanetLab [5] increase the likelihood that a large number of independently routed flows use the same physical resources.

Typically, in a performance-oriented system, overlay-level routing involves the measurement of all available overlay paths; data is then sent along the path that is the best, in terms of the metric of interest to the application. While this "greedy" method works well for a single end-host or decision maker, the effects of simultaneous route-selection by several independent end-hosts have not been adequately studied. Path characteristics can change during, and after, the time required for measurement, due to other end-hosts making route changes. If these end-hosts do not explicitly communicate these decisions to each other, it is likely that the path information is inaccurate at the instant when the selection of the best route is made. It is possible that this could lead to unstable behavior and poor performance of route selection. For example, several end-hosts could find a path to be unloaded, and start sending traffic on that path; now this path would become heavily loaded, and another path would seem more desirable, and this "herd" behavior could continue for the life-time of these end-hosts' traffic flows.

Our goal in this paper is to investigate the occurrence and extent of performance degradation when a large number of end hosts independently perform route selection. We performed simulations using a system model of multiple arriving and departing flows, each with some number of available paths from its source to the destination, having bottleneck physical links in common with paths of other flows. Each of these flows periodically selects a route using the $GREEDY$ method, i.e., it always sends its data on the best path in terms of the metric of interest, which is available bandwidth in our case.

We investigated the factors that affected performance, and based on this, we made simple modifications to the route-selection method, to improve its performance.

Unrestrained GREEDY route selection, as described above, is very unstable, and performs very poorly, even in a dynamic system where the routing decisions are not synchronized. Obviously, the algorithm would benefit from a hysteresis threshold (denoted by H), which regulates the decision to make a route change. When the optimal value of H was employed, we observed significant loss-rates (7% or more) only when the following conditions were true: (a) overlay paths belonging to different flows shared physical bottleneck links to a large extent, (b) the overlay traffic was a large fraction (more than 25%) of the total traffic on the bottleneck links, and (c) the overall bandwidth demand was as high as the overall capacity.

The hysteresis threshold represents only one form of restraint that can be applied to $GREEDY$ selection. We also explored two other forms of restraint: randomizing route selection, and increasing the interval between route change decisions (T_r). Introducing some randomness into route-selection significantly reduced the loss-rates observed (typically by half). Our most interesting observation here was that it is adequate to use measurements of a small subset of all the available paths at any given time to achieve the afore-mentioned improvements. This would significantly reduce measurement traffic and overhead associated with the overlay flows.

Next, we found that the optimal value of H is very sensitive to system parameters. Therefore we propose that all the end-hosts in the system dynamically and independently discover a suitable value for H. We suggest and evaluate a simple adaptive algorithm for this purpose, based on multiplicative increase and decrease of H in response to the rate of route-changes performed. We found that this algorithm yields very low loss rates (less than 1%), and is often better than having a fixed value of H.

Finally, we investigate the performance gained by a small number of "cheating" overlay flows which perform unrestrained $GREEDY$ route selection when all the remaining "good" flows adopt a restrained approach (using the right hysteresis thresholds or randomization). We find that the dynamic discovery algorithm for H performs the best in the context of the good flows, and their performance does not suffer unless the cheating flows approach half the total number of flows. The cheating flows obtain performance benefits only when they number less than 10% of total flows, and these benefits are small.

The rest of the paper is organized as follows. In Section II, we provide some background on overlay networks, and describe related work. In Section III, we describe the system model and simulation methodology that we use to perform our investigations. Section IV presents our results on greedy route selection, and its dependence on H and T_r. Section V presents our results on randomized selection and dynamic discovery of H. Section VI presents our results on the performance obtained by flows that "cheat". Section VII concludes, and outlines future work.

II. Background

There have been several proposals for overlay network protocols. In some cases, the goal was simply incremental deployment of new functionality in the Internet. The MBone [6] was an early example of such an "overlay". Subsequent proposals implemented multicast and group communication solely at end-hosts, at the application-level. For example, End System Multicast [7] organizes a group of end-hosts into a mesh and eventually into a tree. The source (the tree root) sends packets which are replicated and re-sent by the end-hosts corresponding to the nodes of the tree. The tree topology is reactive to properties like latency of the paths corresponding to tree edges. These properties are inferred either by passive measurement (if data is being sent on that path), or active probing.

Overlay networks can also improve performance (e.g., in terms of effective bandwidth), or provide more reliable connectivity. The Resilient Overlay Network (RON) project [2] provides reliable connectivity and quick recovery from path outages for a small group of overlay nodes. Each node monitors the properties of its paths to every other node of a RON by frequent probing or passive monitoring. Upon detection of a path outage between two nodes A and B (or a severe performance drop), packets from A are sent to a third node C, which forwards it to B. The choice of the node C is made by selecting the best composite path (A-C-B) in terms of the application's metric of interest. The RON system claims to be able to recover from path outages in around 20 seconds, while BGP recovery times can be several minutes. Systems like RON and End System Multicast which have a large number of overlay links (compared to the number of nodes) do not scale to a large number of nodes (around 100 in the case of RON, around 1000 in the case of End System Multicast).

Owing to the level of interest in overlay networks, there has been a recent proposal [4] to provide generic overlay functionality in the infrastructure. This proposal advocates provision of infrastructure primitives that would allow an end-host to control packet replication and routing *through the infrastructure*. This makes it easier to deploy an overlay network. The infrastructure is built on a single large basic overlay network. Such an infrastructure can lead to a large number of overlay networks or flows sharing a limited number of physical nodes (and consequently, the links between those nodes), compared to the number of links in the entire Internet. This would increase the chance of adverse interactions between different overlay networks' routing processes if there was no explicit coordination.

Several network-wide measurement services ([4],[8]) have been proposed that eliminate the need for each overlay network to perform measurements. Nevertheless, the potential for instability in route selection still remains, unless there is a single point of serialization or reservation of routing decisions.

A. Related Work

The problem of simultaneous overlay networks or flows is similar to that of server load-balancing. In [9], Mitzenmacher observes instability in the presence of stale load data, and advocates introducing randomization into the server selection method. He observes that the combination of random selection and a small amount of load data is effective in reducing imbalances in server load. This method is particularly attractive in our context, since it involves a lower measurement overhead than other methods. We study the performance of this method (among others) in our problem domain as a possible solution to observed problems. We note, however, that there are several differences between our study and Mitzenmacher's. We study a greater parameter space and use a work model of long-lived flows which make route-change decisions periodically, based on a hysteresis threshold on the improvement of path bandwidth; and we aim for low loss-rates. On the other hand, Mitzenmacher considered jobs with lifetimes much shorter than system life, and was interested in the time to completion of these jobs. There was no notion of job migration in his study.

Selfish routing has also been studied in the past. Roughgarden et al. [10], and more recently, Qiu et al. [11] study the effects of selfish route selection on latency. To our knowledge, corresponding studies of the effect on bandwidth, for overlay networks with dense connectivity graphs, incorporating factors like cross-traffic, hysteresis, and randomization, have not yet been done.

Much work has been done on dynamic routing for networks [12], which typically involves a trade-off between reactivity to load and link state changes, and stability. While a certain lack of coordination does exist between different different nodes of a network (due to transmission delays and out-of-date information), our problem is different in that it involves no explicit communication between the different decision-makers; these different decision-makers observe and usually perturb the same physical links.

III. Simulation Model

In this section we specify the model of overlay network flows that we use for our simulations. Our aim is to use a general model that captures the features important to routing stability, rather than a particular protocol with application-specific details. While our model is a simplification of the real world, and the simulator is not packet-level, it nevertheless provides valuable insight into the relative effects of different factors and schemes on routing performance.

We consider several *independent* flows, either from different overlay networks, or from the same network, but with no explicit coordination with other flows of that network. The key aspect is that the flows do not communicate (regarding their decisions) with each other or with any central entity, and make their decisions independently.

Each flow f seeks to send data traffic from a source to a destination with a certain desired bandwidth. It has several *potential* overlay-level paths that it can use for this purpose. We denote the number of such paths by P_f. Potential paths of different flows may share a bottleneck physical link.

Each flow sends traffic on one of its potential paths, called the *current data path*, and measures the remaining potential paths. After each window of time T_r (the *routing window*), the flow decides whether to continue using the current data path or whether to use one of the other potential paths for sending its data during the next window. This decision is based on measurements of the potential paths' available bandwidth. If the period between routing decisions is different from the measurement period, we call the latter the *measurement window* T_m. A routing change from the current data path to a potential data path will be made only if the improvement in bandwidth is greater than some hysteresis threshold H. We define the actual methods of route selection (and evaluate them) in subsequent sections. In Sections IV and V, all flows use the same route selection method. This assumption is relaxed in Section VI.

Flows arrive and depart from the system with inter-arrival times and lifetimes drawn uniformly at random from a specified interval. We only consider flows with lifetimes much larger than the routing window; it does not make sense for short-lived flows to make route changes, because they would depart before reaping the benefits of the changes. We note that since the flows arrive at different times, their routing decisions are usually not made at the same time. We also assume the presence of *cross-traffic* on the bottleneck links; the cross-traffic bandwidths are drawn uniformly at random from a specified interval. The capacities and loads are always chosen such that it is theoretically possible to route all flows with zero loss rate. Unless specified otherwise, the parameters used in our simulations are the values shown in Figure 1. We consider flows with reasonably long life-times relative to the routing window time, since only such flows would be able to benefit from route-changes. We choose the default inter-arrival times such that the number of flows in the system is around 1000, given the lifetime. These are only the default parameters, and the effect of changing them is also explored.

We choose the average loss rate of the flows after the system has warmed up as our metric of performance. We also present the rate of route-changes when relevant.

Unless specified otherwise, our graphs show data points corresponding to results averaged over 10 simulation runs; also depicted for these results are 95% confidence intervals.

IV. Performance of $GREEDY$ Route Selection

A flow which uses the $GREEDY$ method considers *all* its potential paths, and selects the potential path with the most available bandwidth The flow then sends data on this selected path if that path's available bandwidth is greater than the bandwidth obtained on the current data path by a factor of H. We add random noise to the available bandwidth calculations to partially account for measurement errors that would occur with real-world measurement tools [13]. In this section, we explore the effect of several factors on $GREEDY$'s performance.

We start off with a simple illustration of the role of H in the stability and performance of the routing scheme. Figure 2 plots the number of route changes observed over 10 second windows, versus the progressed time of a simulation, for 3 values of H. The uppermost line represents $H = 2$, while the two (nearly identical) lower lines represent $H = 8.75$ and $H = 15$. Figure 3 plots loss rate similarly, with the three lines from top to bottom representing H values of 2,15, and 8.75, in that order. We see that setting $H = 2$ leads to a high loss rate and rate of change of routes. $H = 15$ causes a more stable system, but the loss rates are still high since the flows remain satisfied with poor routes. $H = 8.75$ represents the best value of H for this scenario.

Mean Inter-Arrival-Time	1 sec
Mean Flow Lifetime	1000 sec
Avg. No. of Flows at any time	1000
No. of bottleneck links	50
Mean P_f	25
Mean Cross-Traffic Bandwidth	50% of capacity
Excess capacity	10% of load
Variation in flow bandwidths	1:4
T_r	10 sec
Variation in T_r	10%
K_g, K_s	4
H for $GREEDY$	8.75
Avg. Measurement Error	10%

Fig. 1. Default Parameters

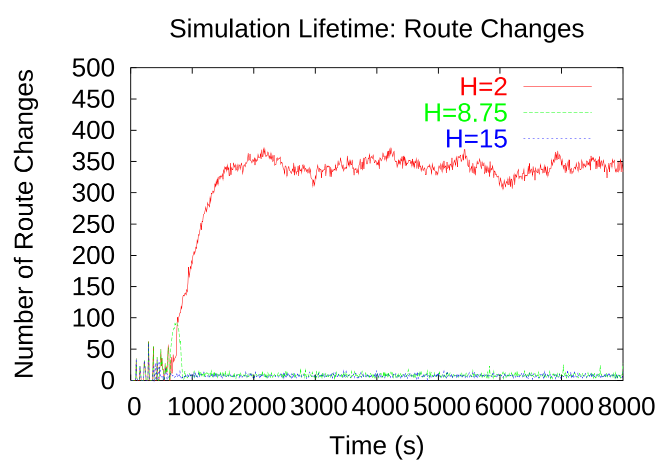


Fig. 2. Number of Route Changes over system time

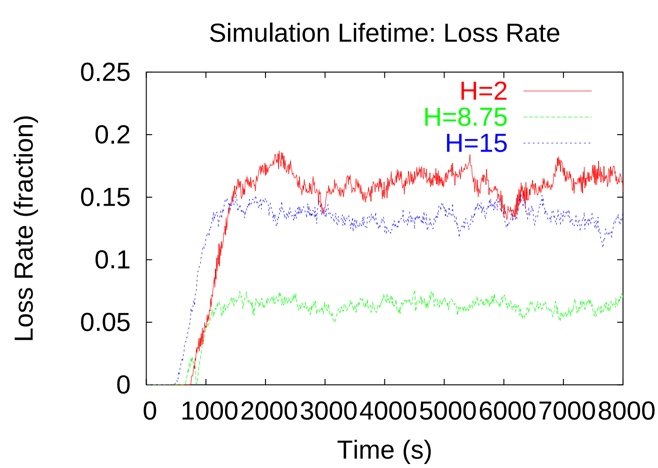


Fig. 3. Average Loss Rate over system time

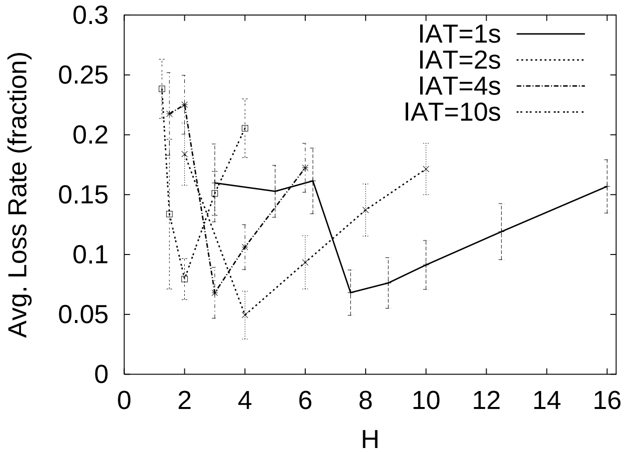


Fig. 4. Loss Rate variation with H, for different mean inter-arrival-times (seconds)

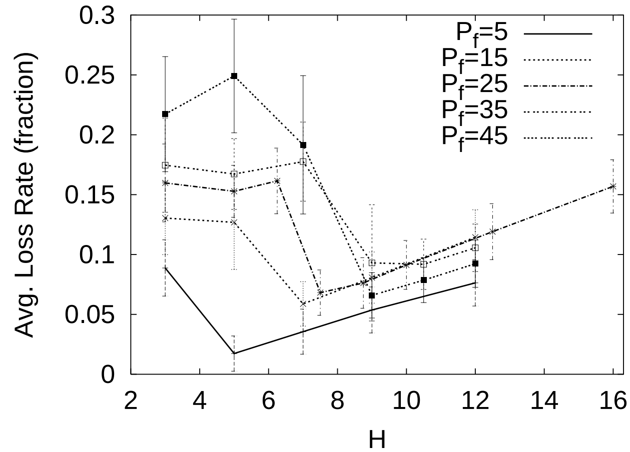


Fig. 5. Loss Rate variation with H, for different mean values of P_f

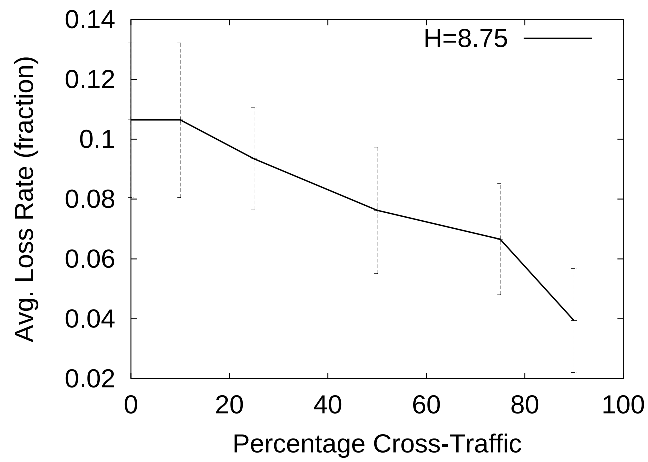


Fig. 6. Loss Rate variation with Percentage Cross-traffic

Figure 4 shows the variation of average loss rate (on the Y-axis) with different values of H (on the X-axis), with each line representing a different rate of arrival of flows. As the rate of arrival decreases, we keep the mean lifetimes fixed, and scale the flow bandwidths so that the total load remains fixed. We observe that, for each value of inter-arrival-time (IAT), the line starts high, drops to a minimum, and then rises again. If the value of H is too low, excessive routing instability leads to poor performance. If the value of H is too high, the flows remain satisfied with poor quality routes, and poor performance results.

Figure 4 also illustrates that the optimal value of H varies significantly with respect to IAT. This optimal value is hard to predict a priori and this remains an open problem. Figure 5 illustrates a similar point. This graph plots the variation of average loss rate with H (on the X-axis), with each line corresponding to a different mean value of P_f. Again, we see that, for a given P_f, the loss-rate can be high unless we choose the optimal value of H. The optimal value of H varies significantly as the the number of potential paths per flow (P_f) changes. This graph also indicates that higher values of P_f lead to worse performance. This is due to increased probability of interaction between two flows.

Figure 6 plots the variation of average loss rate as the mean percentage of link bandwidth occupied by cross-traffic is varied (on the X-axis). We see that GREEDY performs quite well if the cross-traffic is larger than 75%. This is because the total bandwidth effect of route changes comprise an insignificant fraction of the link capacity.

From these results, we believe that $GREEDY$ overlay routing with fixed H would probably work well in the real world if the overlay flows comprise a small fraction of the traffic on physical links, and if paths belonging to different flows do not share bottleneck links to a large extent. However, peer-to-peer traffic has been reported to comprise more than 42% of total traffic to and from a domain [14] - this gives rise to the possibility that overlay routing flows could occupy a large fraction of link bandwidths, in the future. Also, scenarios with high probability of path-sharing could arise if several flows use a common overlay infrastructure ([4]), or if several flows of a peer-to-peer or overlay network do not coordinate with each other. This could also arise if the distribution of overlay nodes was skewed toward certain domains, like universities; e.g., [14] reported that there were more than 4500 peers of a peer-to-peer application inside the campus network they studied. For shared bottleneck links to exist, the bottleneck links of each flow's potential paths would have to lie on some link other than the physical link connecting the overlay nodes

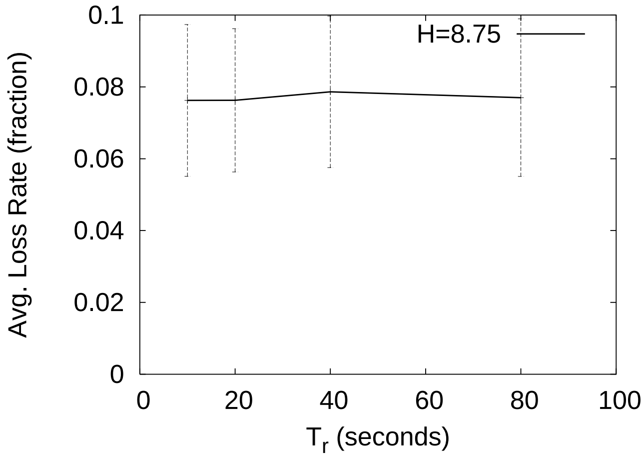


Fig. 7. Loss Rate variation with different routing window times (seconds)

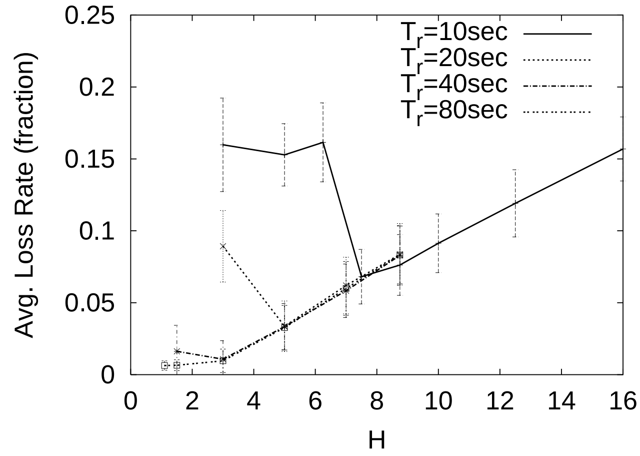


Fig. 8. Loss Rate variation with H, for different routing window times (seconds)

to the Internet - unless different *independent* overlay flows used the same physical machine as an overlay node. The latter possibility could arise in the case of multi-user distributed testbeds like PlanetLab [5].

Next we consider the effect of increasing routing window time while keeping the measurement window fixed. This has the effect of increasing the accuracy of measurement, since fewer flows would change their routes during the measurement window. However, this also leads to decreased reactivity which can adversely affect performance. If we fix the value of H at the optimal value for $T_r = 10s$ and then vary T_r, there is no significant variation in performance, as shown in Figure 7.

However, better performance might be obtained at lower values of H, since increasing T_r adds some "restraint" in itself. Figure 8 shows the variation of average loss rate on the Y-axis, with different values of H on the X-axis. Each line represents a different value of T_r, with T_m fixed at 10 seconds. We observe that, using the optimal values of H (i.e., the minimum of each line) , the loss rate reduces as T_r increases, up to a point. Thus, careful choice of the routing and measurement windows can improve performance, if we also choose the right value of H. However, we notice that the optimal value of H is quite different for each value of T_r. This motivates our proposal of a more effective method of route selection in Section V-B.

V. Improvements to $GREEDY$

In this section, we consider two methods of improving the performance of route selection.

A. Randomized Route Selection Methods

Intuitively, adding an element of randomization can add some stability to the system. We consider three candidate methods to achieve this:

$ARAND$**:** The flow randomly selects a path from its set of potential paths (with probabilities weighted by the *available* bandwidth of potential paths).

$GRAND$**:** The flow randomly selects a path from the "best" K_g potential paths (where K_g is small compared to the total number of potential paths, and "best" is defined in terms of greatest available bandwidth).

$SRAND$**:** As observed by Mitzenmacher in study of server load-balancing [9], a small amount of load information is likely to be significantly better than no load information. In addition, we observed in Section IV that using *all* load information greedily is not always beneficial. Therefore we explore the performance of the following method: each flow selects a subset of K_s paths, at random, from all its potential paths, and then selects the path with the highest available bandwidth in this subset. We make the probability of a path's inclusion in the subset proportional to its capacity, to account (partially) for link heterogeneity.

$SRAND$ has the additional benefit that each flow needs to only probe K_s paths at a time for their load (or available bandwidth) information, if it selected the random subset at the beginning of the routing window. It could then select the path with the highest measured available bandwidth from this subset, at the end of the routing window. The capacities of the paths need not be probed at the same frequency as the load information, since the capacities are unlikely to change over flow lifetimes. Therefore if we assume that the measurement overhead is dominated by the probing for load informa-

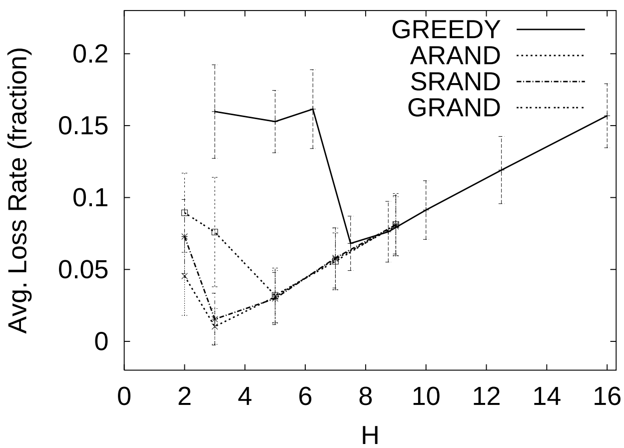


Fig. 9. Loss Rate variation with H, for different randomization methods

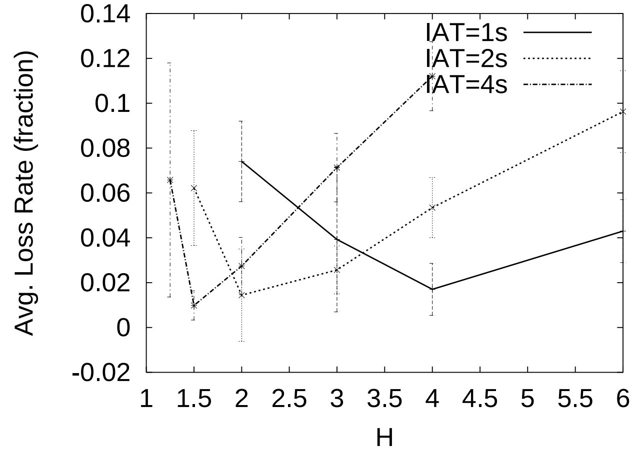


Fig. 10. Loss Rate variation of $SRAND$ with H, for different values of mean inter-arrival-time

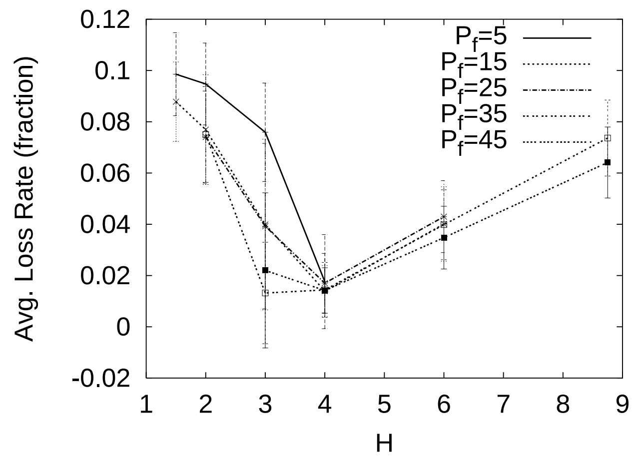


Fig. 11. Loss Rate variation of $SRAND$ with H, for different values of P_f

tion, this method reduces the overhead by a factor of P_f/K_s compared to the other methods.

Figure 9 shows the loss rates for different values of H (on the X-axis), with each line representing one of the methods of randomization for a system with the default parameters, as in Figure 1. We notice that, while the performance is still sensitive to H, it is possible to obtain much lower loss-rates than GREEDY, using the randomized methods. In addition, the $SRAND$ method, which in this case reduces the overhead by $\frac{25}{4}$, works as well as the other methods.

We now explore the performance of the $SRAND$ method in more detail. Figure 10 shows the loss-rate as H is varied on the X-axis, for different inter-arrival-times (represented by the different lines). Similarly, Figure 11 depicts different values of P_f by different lines. We observe that $SRAND$ exhibits trends similar to $GREEDY$, but is less sensitive to H. More importantly, the loss-rates for all these scenarios are much lower than $GREEDY$ loss-rates.

B. Discovery of Threshold H

We observed, in Section IV, that poorly chosen values of H could lead to high loss rates, and that the value of H was sensitive to many parameters. Therefore, we propose that end-hosts or flows dynamically discover the value of H best suited to their current deployment scenario. Since we assume that the end-hosts do not explicitly communicate, they have to perform this discovery independently. We propose the following algorithm: each flow maintains its own value of H. When the flow makes a route change, it increases the value of H, and when a routing window passes by with no route change (i.e. no path offers a bandwidth improvement factor greater than H), it decreases the value of H.We experimented with several combinations and modifications of additive/multiplicative increase and decrease algorithms, and we found that the best performance was obtained in most cases by multiplicative increase and decrease. In addition, there is a trade-off between quickly moving from the initial choice of H to a "suitable" value, and subsequent stability; therefore, we settled on a large (i.e., conservative) initial choice of H,and added a quick-start phase with a multiplicative decrease factor of 2. Figure 12 shows the flow diagram for this algorithm.

Figure 13 shows the performance of this algorithm for our default system, with different decrease parameters (on the X-axis), and increase parameters (one per line). We find that we obtain very good performance for increase parameters of 1.9 or greater, and for decrease parameters of 1.01. We use these as our settings for subsequent simulations.

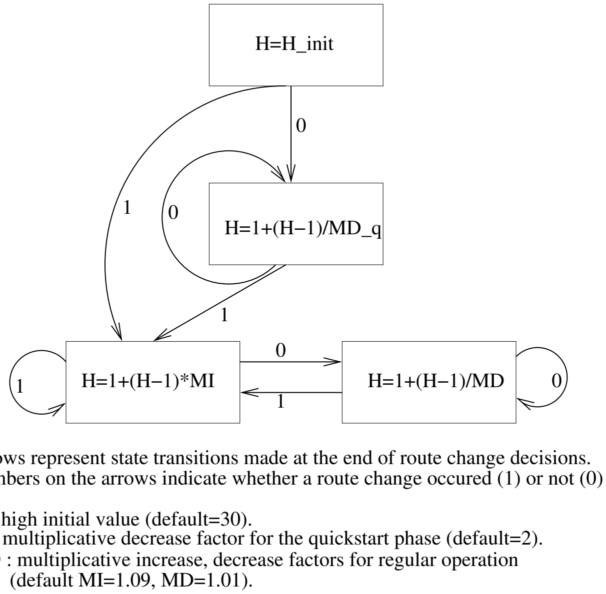


Fig. 12. Flow diagram for the MIMD H-discovery algorithm

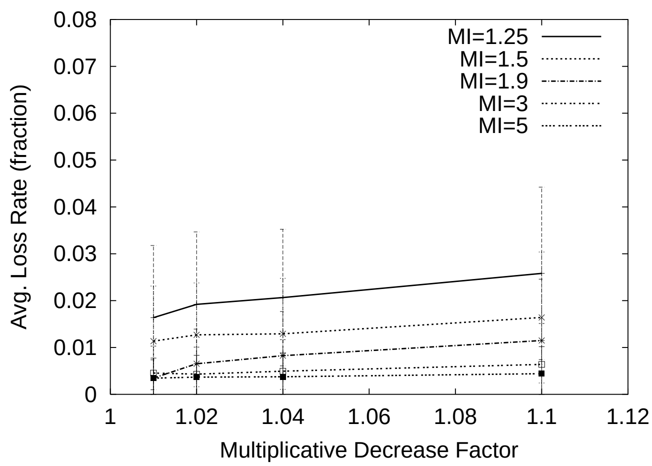


Fig. 13. Loss Rate variation with increase and decrease parameters

Figures 14 and 15 illustrate the performance of this algorithm in a system with default parameters as in Figure 1. The first graph plots the average loss rate for different values of P_f (on the X-axis), while the second does the same for different flow arrival rates (and consequently, flow bandwidths). These figures show that the discovery method reduces loss rates by more than half compared to basic $GREEDY$(Section IV). While the loss-rate still varies with P_f and the arrival rate, the magnitude and variation of the loss-rate is much smaller than $GREEDY$ and often less than 1%.

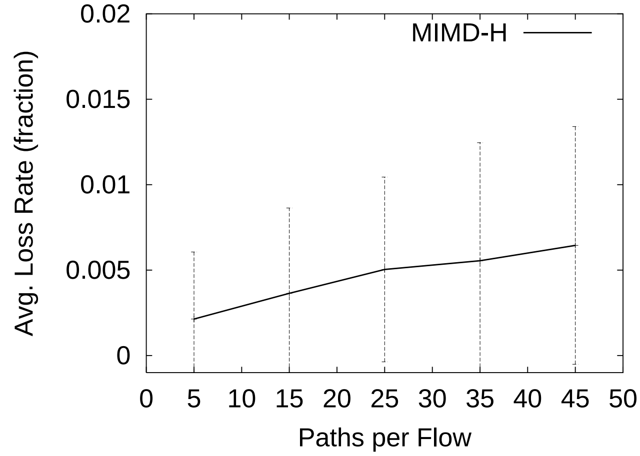


Fig. 14. Average Loss Rate versus mean value of P_f

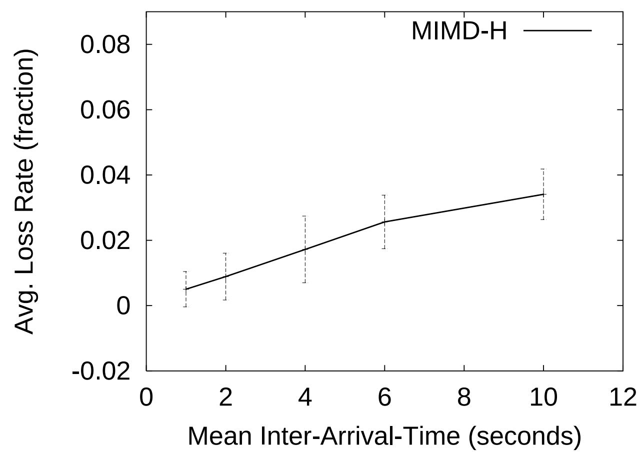


Fig. 15. Loss Rate variation with mean inter-arrival-time(seconds)

VI. FLOWS THAT "CHEAT"

Until now in this paper, we have assumed that all the overlay flows in a system utilize the same route selection method. We now relax that assumption. Let us assume that all the overlay flows in the system are expected (much like the expectation of TCP-friendliness) to use some method of restraint, like an appropriate value of H. What would happen if a subset of the overlay flows decided to "cheat", and attempt to obtain higher bandwidth by using unrestrained $GREEDY$ with low values of H? Clearly this is a possibility - there have been examples of people taking advantage of loopholes in TCP [15] to obtain higher throughputs.

Figure 16 shows the loss rates for our default system on the Y-axis, while the percentage of cheating flows, out of the total number of flows, is varied on the X-axis. The two different lines in each graph correspond to

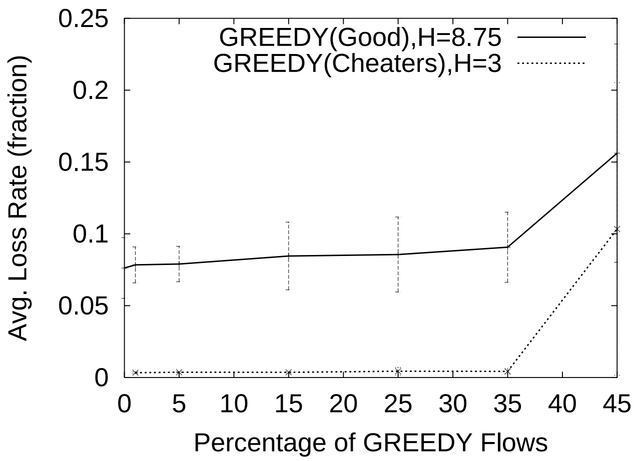


Fig. 16. Loss Rates with different percentages of “cheaters”

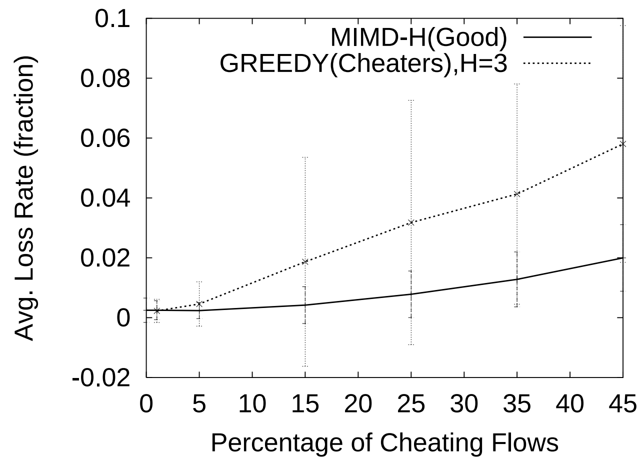


Fig. 18. Loss Rates with different percentages of “cheaters” for the dynamic discovery method

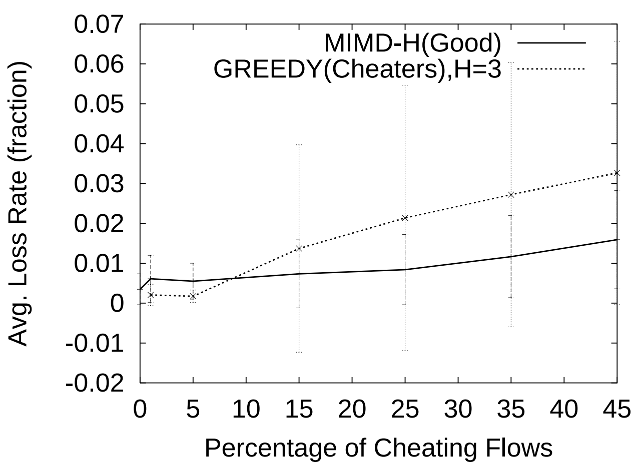


Fig. 17. Loss Rates with different percentages of “cheaters” for the dynamic discovery method

the cheating flows that use $GREEDY$ with low values of H, and the “good” flows, that use $GREEDY$ with the value of H that would be optimal if there were no cheaters. From this we observe that the cheating flows derive significant benefit. The performance degradation of the good flows becomes significant when the number of cheating flows approaches half the total number of flows.

Figure 17 shows the loss rates when the “good” flows use the dynamic discovery method for H, and the “cheating” flows use fixed low values of H. We see that the dynamic method is more resistant to cheating - the performance of the good flows does not suffer as much relatively. The cheating flows benefit to a small extent, and only when they are less than 10% of the total flows. This reduces the incentive of flows to cheat. When we combine randomized selection and the dynamic discovery of H, we find that the “cheating” flows obtain virtually no benefit in performance at all, even when they are few in number, as shown in Figure 18.

VII. Conclusions and Future Work

We have investigated the performance of independent $GREEDY$ route selection by end-hosts at the overlay level, for a wide variety of scenarios. We find that greedy route selection performs well under light load, when the overlay flows comprise a small fraction of link capacities, or when the paths of different overlay flows do not share bottleneck physical links to a large extent. However, when these conditions are not met, we find that greedy route selection can perform poorly, unless some restraint or stabilizing factor is used. We have investigated three methods of applying this restraint - randomization, using a hysteresis threshold, and increasing routing windows.

Our two most interesting observations are: (a) it is sufficient to measure a small subset of paths to obtain good routing performance (half the loss-rates of $GREEDY$), and (b) $GREEDY$ performance is heavily dependent on the value of the hysteresis threshold H, the optimal value of which is dependent on other factors like flow arrival rates or P_r. The latter observation motivates our proposal of a simple dynamic algorithm for each flow to discover a suitable value of H, based on multiplicative increase (and decrease) of H in response to routing change (and lack thereof), with a more rapidly decreasing quick-start phase. This is found to perform well, reducing loss rates to less than half of $GREEDY$'s loss rates.

As future work, we intend to consider more complicated network models, derived from topologies and workloads of testbeds that are used for deployment of peer-to-peer or other distributed systems, like PlanetLab [5]. We also plan to investigate the behavior of these routing methods and the threshold discovery algorithm in the presence of more dynamic types of cross traffic.

References

[1] "The gnutella home page," http://gnutella.com.

[2] David G. Andersen, Hari Balakrishnan, M. Frans Kaashoek, and Robert Morris, "Resilient overlay networks," in *Proc. ACM SOSP '01*, Oct. 2001.

[3] Ion Stoica, Daniel Adkins, Shelley Zhuang, Scott Shenker, and Sonesh Surana, "Internet Indirection Infrastructure," in *Proceedings of ACM SIGCOMM 2002*, Aug. 2002.

[4] Karthik Lakshminaryanan, Ion Stoica, and Scott Shenker, "Building a flexible and efficient routing infrastructure: Need and challenges," Technical Report CSD-03-1254, University of California, Berkeley, CA, Mar. 2003.

[5] Intel Research, "The PlanetLab Testbed," http://www.planet-lab.org.

[6] H. Eriksson, "MBone: The Multicast Backbone," *Communications of the Association for Computing Machinery*, pp. 54–60, 1994.

[7] Yang hua Chu, Sanjay Rao, and Hui Zhang, "A Case For End System Multicast," in *Proceedings of ACM Sigmetrics '00*, Santa Clara, CA, June 2000.

[8] Aki Nakao, Larry Peterson, and Andy Bavier, "A routing underlay for overlay networks," in *To appear in Proc. ACM Sigcomm '03*, Aug. 2003.

[9] Michael Mitzenmacher, "How useful is old information?," in *Proc. PODC '97*, 1997.

[10] T. Roughgarden and . Tardos, "How bad is selfish routing?," in *In Proc. FOCS '00*, 2000.

[11] Lili Qiu, Yang Richard Yang, Yin Zhang, and Scott Shenker, "On selfish routing in internet-like environments," in *To appear in Proc. ACM Sigcomm '03*, Aug. 2003.

[12] A.Shaikh, *Efficient Dynamic Routing in Wide-Area Networks*, Ph.D. thesis, University of Michigan, May 1999.

[13] Manish Jain and Constantinos Dovrolin, "End-to-end available bandwidth: Measurement methodology, dynamics, and relation with tcp throughput," in *Proc. ACM Sigcomm '02*, Aug. 2002.

[14] Stefan Saroiu, Krishna P. Gummadi, Richard J. Dunn, Steven D. Gribble, and Henry M. Levy, "An analysis of internet content delivery systems," in *Proc. ACM OSDI '02*, Dec. 2002.

[15] Stefan Savage and Neal Cardwell and David Wetherall and Tom Anderson, "TCP Congestion Control with a Misbehaving Receiver," *ACM Computer Communication Review*, vol. 29, no. 5, Oct. 1999.