

# Stability of QoS

Avinash Varadarajan, Subhransu Maji  
{avinash,smaji}@cs.berkeley.edu

## Abstract

Given a choice between two services, rest of the things being equal, it is natural to prefer the one with more stable Quality of Service(QoS). For certain applications, stability would weigh more than other factors. For example, in the case of a streaming media application that has a limited buffer, a server offering stable moderate bandwidth is preferred to one that offers high bandwidth most of time, but has outages every now and then. In this work, we study the problem of a client choosing one out of a set of servers that offer time varying QoS, dynamically updating its choice, in order to achieve stable service. The QoS metric we consider in this work is bandwidth.

## 1 Introduction

Consider the problem of a client deciding to choose one among  $n$  servers, that offer bandwidths  $B_1(t), B_2(t) \dots B_n(t)$ , at any given instant  $t$ . Let an interruption period be defined as one in which the bandwidth received by the client drops below a threshold  $\tau$ . At any instant the client can decide to switch from one server to another. When the client switches from one server to another, there is a period of duration  $P$  during which the client receives 0 bandwidth. This serves as a *switch cost*, that models the overhead involved in the protocol messages that need to be sent when switching from one server to another. The value of  $P$  is application dependent, based on how “easy” it is to switch from one server to another. The *available bandwidth* for the client,  $B(t)$ , is a function of the client’s choices, and  $B_1(t), B_2(t) \dots B_n(t)$ . The goal of the client is to decide which server to receive service from at time  $t$ , such that the total interruption period, including the interruptions occurred during switching, is low.

In the above formulation, we have made an implicit assumption that the bandwidths offered by the servers are independent of the client’s choices. This will definitely not hold in a general case, as there would be several clients trying to use these same set of servers. Nevertheless, studying the restricted case of the problem is useful, as it would help us in understanding some aspects of this problem, before tackling the general case. Also, the assumption that we make would be reasonable, when the total volume of traffic generated by all the clients of the particular application is negligible compared to the volume of traffic needed to alter available bandwidths on the network.

We consider two instances of the above problem in our work. In one of them the “servers” are the set of possible relay nodes for a VoIp application [1] and the bandwidth of interest is end-to-end bandwidth between the two clients of the application via a relay node. In the other case, the “servers” are the possible parents in an overlay multicast tree used in streaming media applications [5, 6, 7, 8], and the bandwidth under consideration is the bandwidth between the client and the root of the tree via a parent.

Our work is related to work in [2], where they study the problem of choosing  $k$  servers from a set of  $n$ , dynamically replacing those servers, within the chosen set of  $k$  servers that go down by the remaining servers that are up. The choice of initial set of servers and the choice of the server used for replacement should be made in such a way that the chosen set remains “stable” requiring replacements as few times as possible.

## 2 Traces

For our simulations we use the bandwidth measurements reported in [3] made using the NIMI measurement infrastructure [4]. NIMI is a measurement framework, in which a number of measurement platforms are deployed across the Internet and used to perform end-to-end measurements. For bandwidth measurements they use TCP transfers between random pairs of NIMI hosts, making a 1 MB transfer between the same pair of hosts every minute for a 5-hour period. We took all measurement

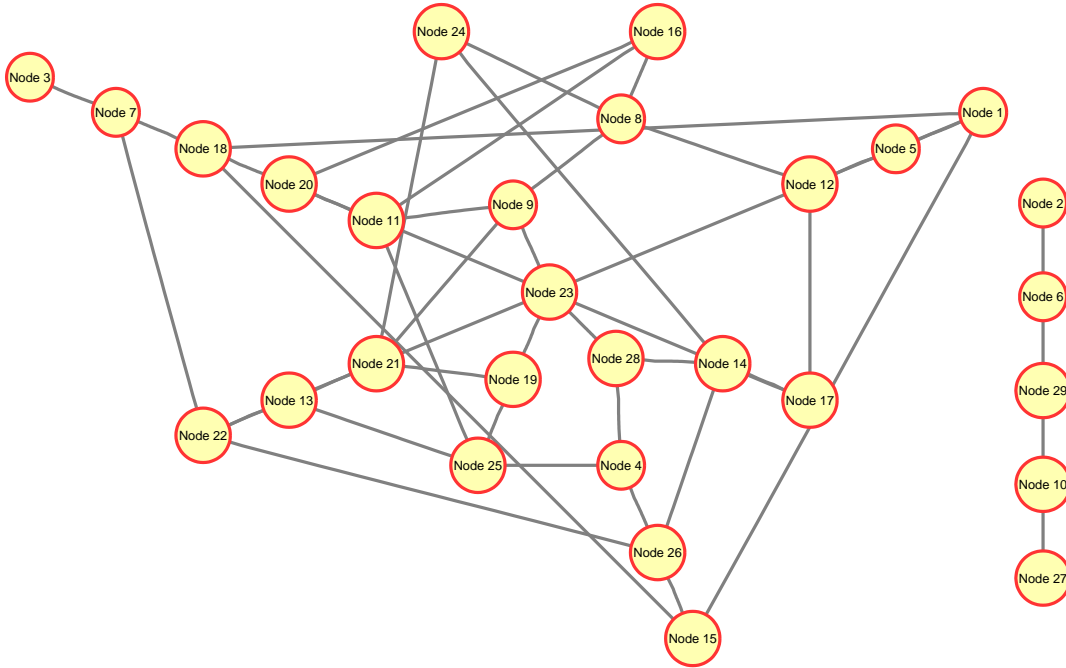


Figure 1: The graph describing pairwise measurement traces.

traces that had less than 10 errors out of the 300 measurements and interpolated these missing values. Figure 1 shows a graph with nodes representing the NIMI hosts, and edges connecting pairs of nodes involved in a measurement trace that was used.

The Bandwidth functions  $B_1(t), B_2(t) \dots B_n(t)$  introduced in Section 1 are piecewise constant approximations derived from the 300 measurement points. We assume that time  $t$  runs from 0 to 300 minutes, and that the value of the  $k^{th}$  measurement between a pair of nodes is the bandwidth from  $t = k - 1$  to  $t = k$  for that pair. The *switch cost*, namely the time for which an interruption is observed on switching, is assumed to be 1 minute. For the two applications we consider, we believe that 1 minute would be reasonable upper bound for time taken to switch. Given these assumptions, the available bandwidth  $B(t)$  for the client can also be represented as a set of 300 values, and the total interruption time in minutes will simply be the total number of these 300 values that are below the threshold  $\tau$ . Hence, the the bandwidth functions are fully defined by these 300 values, and we use the terms “number of interruptions” and “total interruption time” interchangeably. Also, given these assumptions, it is clear that the decisions about switching need to be done only at the beginning of each minute.

### 3 Selection Strategies

#### 3.0.1 Optimal

Given the bandwidth functions  $B_1(t), B_2(t) \dots B_n(t)$ , make the set of decisions that result in the least possible total number of interruptions. Let the  $M(t, i)$  be the least possible total number of interruptions for the first  $t$  minutes given that in the  $t^{th}$  minute the client receives service from the  $i^{th}$  server. Then

$$\begin{aligned}
 M(1, i) &= thresh(B_i(1)) \\
 M(t, i) &= Min_i(M(t - 1, i) + thresh(B_i(t), \tau)) \quad 1 < t \leq 300
 \end{aligned}$$

And finally the minimum number of interruptions is  $Min_i(M(300, i))$ , where  $thresh(b, \tau)$  takes the value 1 when  $b \geq \tau$  and 0 otherwise. We use this definition for the function  $thresh$  throughout the paper.

In the static strategies, no switch is ever made - the client chooses a particular server and receives service from the same server for the whole time period.

### 3.0.2 Static-Best

Choose that server whose bandwidth function has the least number of interruptions. The number of interruptions in this case would be  $Min(\sum_{t=1}^{300} thresh(Bi(t), \tau)), 1 \leq i \leq n$

### 3.0.3 Static-Second-Best

Choose that server whose bandwidth function has the second least number of interruptions.

The above strategies cannot be implemented, as the bandwidth function is not known in advance. We consider the above two strategies simply for the sake of comparing them with the other strategies.

### 3.0.4 Static-Heuristic

Evaluate the different servers in a window of the first few minutes, and choose the server that has the least number of interruptions within this window.

### 3.0.5 Replace-With-Best

For the first minute, choose the server that offers the highest bandwidth in the first minute. For any subsequent minute  $t$ , if the bandwidth offered by the current server drops below the threshold  $\tau$ , switch to that server which offers the highest bandwidth in the  $t^{th}$  minute. If no server offers a bandwidth  $\geq \tau$  do not switch.

### 3.0.6 Random-Replacement

For the first minute, randomly choose any server that offers a bandwidth of at least  $\tau$ . For any subsequent minute  $t$ , if the bandwidth from the current server drops below the threshold  $\tau$ , among the server that offer a bandwidth  $\geq \tau$ , pick one randomly, and switch to it. If no such server exists, do not switch.

### 3.0.7 Random-Subset-Replacement

For the first minute, select a random subset of  $s$  server from the  $n$  server, and choose the server within this subset that offers the highest bandwidth for the first minute. If the bandwidth from the current server drops below  $\tau$  in any subsequence minute, repeat the same procedure to choose a server to switch to.

The above three strategies make use of as few values of the measurements as possible, in order to make decisions. These are the attractive strategies for implementing in an application, as the bandwidths have to be measured online, and the a strategy that requires it to be done as few times as possible is preferable.

### 3.0.8 Replace-With-Best-History

For the first minute, choose the server that offers the highest bandwidth in the first minute. For any subsequent minute  $t$ , if the bandwidth offered by the current server drops below the threshold  $\tau$ , switch to that server which has had the least number of interruptions in the first  $t$  minutes.

*Replace-With-Best-History* is a *predictive strategy*, that tries to predict the server's future performance, based on its performance so far.

## 4 Relay Node selection in VoIp

### 4.1 Simulation Setup

Consider the topology shown in figure 2. The "client" in this case is one of  $C1$  or  $C2$ , and "servers" are the  $n$  relay nodes labeled  $R_1 \dots R_{10}$ . Each edge in this graph represents a path in the underlying IP network. The end-to-end bandwidth between  $C1$  and  $C2$  via  $R_i$  is  $Min$  of the two bandwidths along the two edges -  $C1 - R_i$  and  $R_i - C2$ . From the bandwidth data set, we use a randomly chosen trace of 300 minutes to model the bandwidth along one of the edges. For each strategy, we simulated 1000 trials for different values of the threshold  $\tau$ , where a trial consists of choosing a set

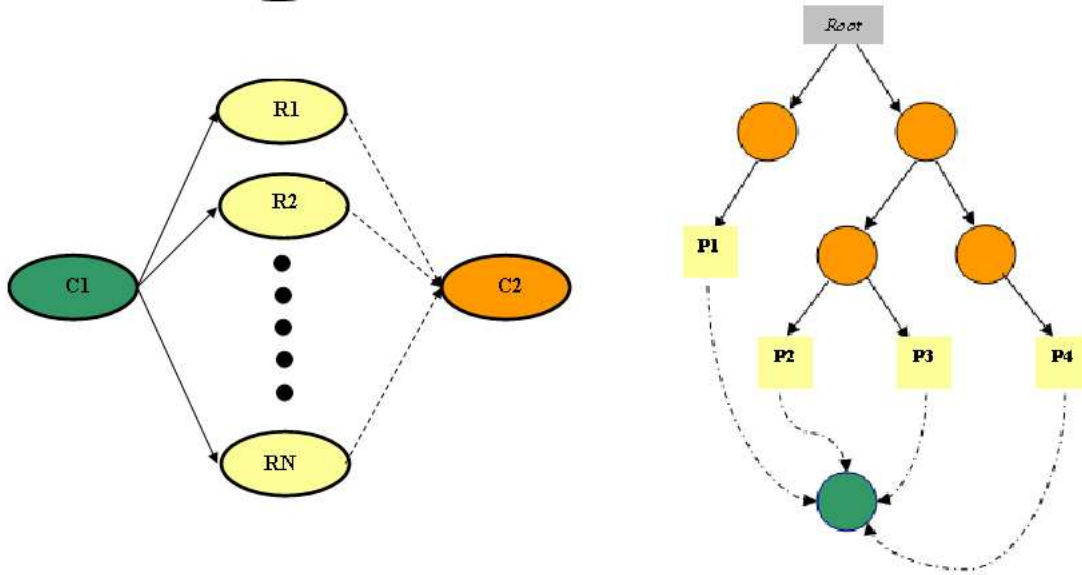


Figure 2: Overlay Topology for VOIP(left) and Multicast tree(Right).

of 20 traces to model all the edges, and simulating the strategy for the 300 minutes. We consider thresholds from **0.1** to **0.4** at a step size of 0.05Mbps. For *Static-Heuristic*, we try three different window sizes - 1, 5, and 10.

## 4.2 Results

The results of the experiments are shown in figures 3 and 4. In general, the dynamic strategies, i.e, strategies that adaptively switch from one relay node to another perform better than even *Static-Best*. The improvements over *Static-Second-Best* and *Static-Heuristic* are higher. There is huge difference between the number of interruptions for *Static-Best* and the number of interruptions for *Static-Second-Best*, suggesting that strategies that make switches are preferable to those that do not make any switches, as in a static strategy the penalty for choosing the wrong relay node is very high. *Replace-With-Best-History*, does not perform a lot better than *Replace-With-Best* - making use all the extra measurements does not make the performance too much better. On a similar note, *Random-Subset-Replacement* for a subset size of 5, performs nearly as good as *Replace-With-Best*, although it uses just half of the measurements. Increasing the subset size to more than 5 gives only fractional improvement in the number of interruptions.

## 5 Parent Selection in a Overlay-Multicast Tree

### 5.1 Simulation Setup

In this case, the “servers” are the possible parents in an overlay multicast tree as shown in Figure 2, and the bandwidth under consideration is the bandwidth between the client and the root of the tree via a parent. We model the overlay network as a graph with an edge between the nodes if there is bandwidth measurement data between the two nodes. This is a directed graph, however if we assume that the bandwidths are symmetrical then we can ignore the directionality of the edges. Figure 1 shows the undirected graph representing the overlay topology.

Multicast trees are constructed by a breath-first tree rooted at every node of the graph. This graph has 29 nodes and 44 edges. Thus we have 29 multicast trees with each of these nodes as the root. For purposes of our simulation we only consider the trees which have greater than one leaf nodes. There are 27 trees with an average of 8 leaf nodes. A new node has to choose one of the leaf nodes as a parent to become a part of the multicast tree. The effective bandwidth is now the minimum of the bandwidths along the path of the leaf node to the parent. The number of interruptions in service is the interval of time the bandwidth is below a threshold. We run the simulations for various thresholds from 0.01 to 0.1 MBps with increments of 0.01 MBps using strategies as described earlier.

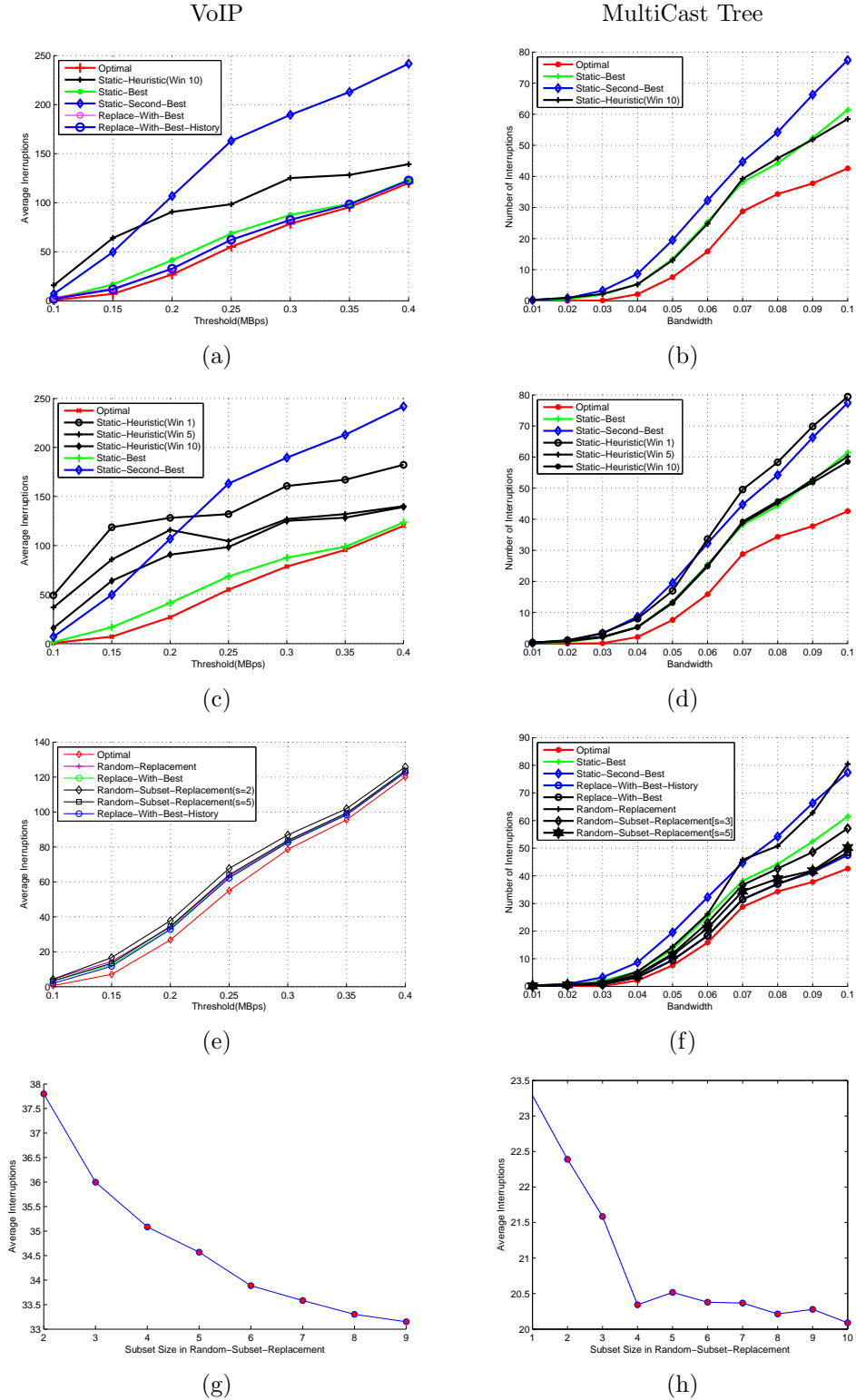


Figure 3: Comparison of various strategies on VoIP and Overlay Multicast Tree Topologies (a),(b) Comparison of *Optimal*, *Static-Best*, *Static-Second-Best*, *Static-Heuristic*. The Heuristic functions does almost as good as the static best. The penalty of choosing the second best is high as can be seen from the difference between the blue and the green curves. (c),(d) Comparison of *Static-Heuristic* Strategy, with different window sizes - 1, 5, 10. The window size 1 is much worse than the window size of 5 and 10 both of which perform similar. (e),(f) Comparison of various dynamic strategies and *Replace-With-Best-History*. Most of the dynamic strategies do better than any of the static strategies and the performance is comparable with the *Replace-With-Best-History* strategy which is close to the optimal. (g),(h) Plots the number of interrupts as a function of subset size. Increasing the subset size beyond 4 does not give any significant improvement in performance in OverlayTree while a subset size of 5 gives close to optimal performance. In both cases this is approximately half the maximal choices.

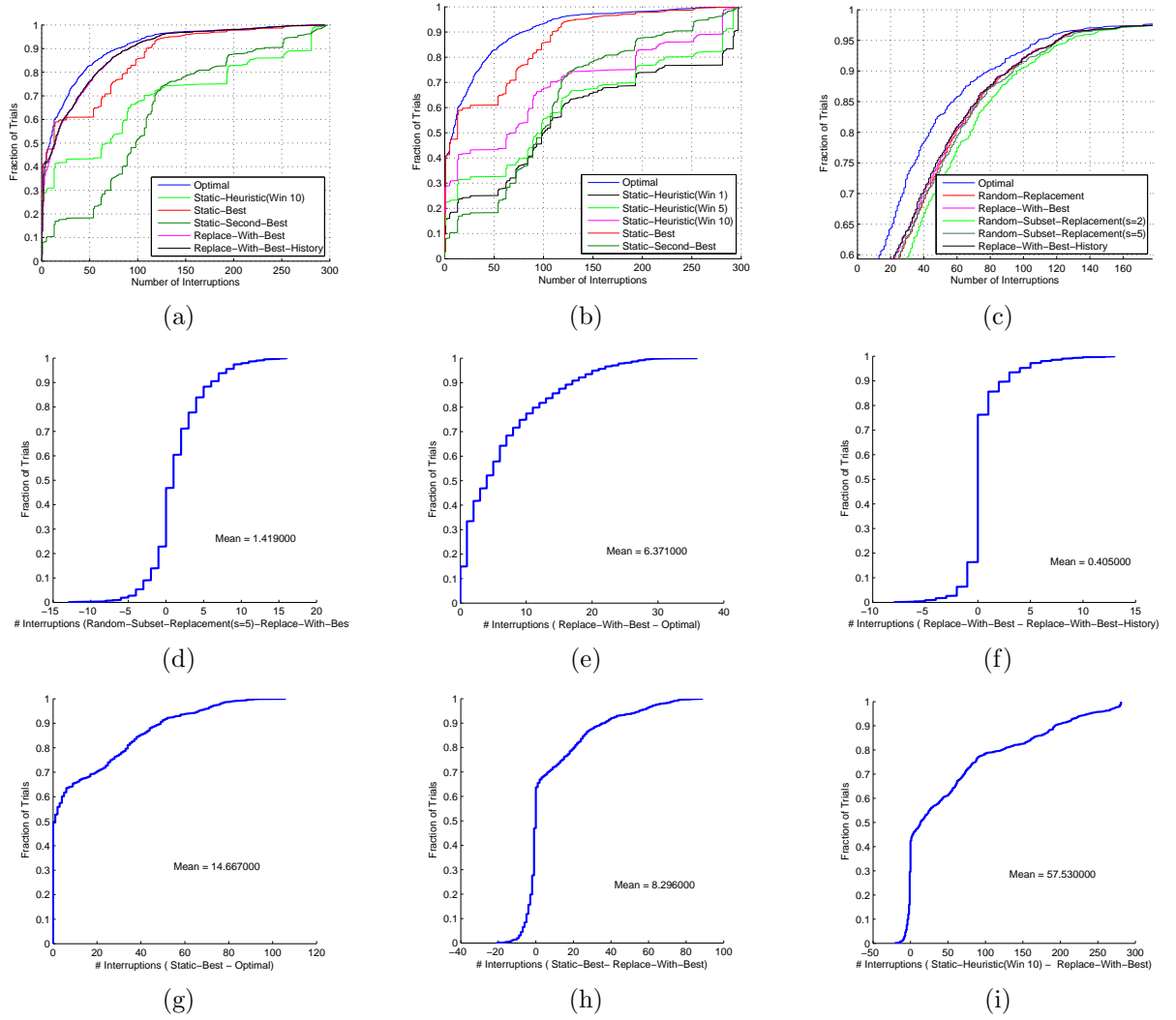


Figure 4: (a),(b),(c) Fraction of trails(in 1000 trials) Vs Number of Interruptions at threshold of 0.2 MBps for the various strategies. Fraction of Trials Vs Difference in Number of interruptions at threshold of 0.2MBps for (d) Random-Subset-Replacement, Replace-With-Best (e) Replace-With-Best, Optimal (f) Replace-With-Best, Replace-With-Best-History (g) Static-Best Optimal (h) Static-Best, Replace-With-Best (i) Static-Heuristic(Window Size 10), Replace-With-Best

## 5.2 Results

Figure 3 shows the results of our simulations. We observe similar trends in this dataset too. The random strategies outperform all the static strategies consistently. The *Replace-With-Best* performs same as the *Replace-With-Best-History* strategy suggesting that the a node does not require to maintain a lot of state. Here too we observe a significant difference between the *Static-Best* and *Static-Second-Best*, suggesting that strategies are better off switching. *Random-Subset-Replacement* with subset size of 4 has comparable performance with the *Replace-With-Best* strategy but requires only half of its measurements on an average. Also *Random-Subset-Replacement* gives significantly lesser interrupts than any static strategy.

## 6 Conclusions and Future Work

We perform a comparative study of various algorithms for the simplified version of the problem of selecting a link to receive service over time. Our experiments show that the strategies that make switches are better than static strategies. It is possible to reduce interruptions with a few measurements at coarse granularity. Our work can be extended by running experiments on other data traces, preferably on measurements at finer granularity. Possible future work also includes studying the stability of other other QOS parameters like Packet Loss, Jitter etc, and combinations of two or more parameters, and implementing these ideas in a real system to analyze performance.

## References

- [1] Skype. <http://www.skype.com>
- [2] Minimizing Churn in Distributed Systems, P. Brighten Godfrey, Scott Shenker, and Ion Stoica. SIGCOMM 2006.
- [3] Yin Zhang, Nick Duffield, Vern Paxson, and Scott Shenker. Internet Measurement Conference '01.
- [4] V. Paxson, J. Mahdavi, A. Adams, and M. Mathis, An Architecture for Large Scale Internet Measurements, IEEE Communications,36(8), pp 48-54, Aug. 1998.
- [5] S. Banerjee, B. Bhattacharjee, and C. Kommareddy. Scalable Application Layer Multicast. In Proceedings of ACM SIGCOMM, August 2002.
- [6] Y. Chu, S. G. Rao, and H. Zhang. A Case for End System Multicast. In Proceedings of ACM Sigmetrics, June 2000.
- [7] P. Francis. Yoid: Your Own Internet Distribution, <http://www.aciri.org/yoid/>. April 2000.
- [8] J. Jannotti, D. Gifford, K. L. Johnson, M. F. Kaashoek, and J. W. O'Toole Jr. Overcast: Reliable Multicasting with an Overlay Network. In Proceedings of the Fourth Symposium on Operating System Design and Implementation (OSDI), October 2000.