Measurement-Based Optimization Techniques for Bandwidth-Demanding Peer-to-Peer Systems

T. S. Eugene Ng, Yang-hua Chu, Sanjay G. Rao, Kunwadee Sripanidkulchai, Hui Zhang
Carnegie Mellon University, Pittsburgh, PA 15213
{*eugeneng,yhchu,sanjay,kunwadee,hzhang*}*@cs.cmu.edu*

***Abstract*— Measurement-based optimization is one important strategy to improve the performance of bandwidth-demanding peer-to-peer systems. However, to date, we have little quantitative knowledge of how well basic light-weight measurement-based techniques such as RTT probing, 10KB TCP probing, and bottleneck bandwidth probing may work in practice in the peer-to-peer environment. By conducting trace-based analyses, we find that the basic techniques can help achieve 40 to 50% optimal performance. To deepen our understanding, we analyze some of the intrinsic properties of these techniques. Our analyses reveal the inherent difficulty of the peer selection problem due to the extreme heterogeneity in the peer-to-peer environment, and that the basic techniques are limited because their primary strength lies in eliminating the low-performance peers rather than reliably identifying the best-performing one.**

However, our analyses also reveal two key insights that can potentially be exploited by applications. First, for adaptive applications that can continuously change communication peers, the basic techniques are highly effective in guiding the adaption process. In our experiments, typically an 80% optimal peer can be found by trying less than 5 candidates. Secondly, we find that the basic techniques are highly complementary and can potentially be combined to better identify a high-performance peer, thus even applications that cannot adapt may benefit. Using media file sharing and overlay multicast streaming as case studies, we have systematically experimented with several simple combined peer selection techniques. Our results show that for the non-adaptive media file sharing application, a simple combined technique can boost performance to 60% optimal. In contrast, for the continuously adaptive overlay multicast application, we find that a basic technique with even low-fidelity network information is sufficient to ensure good performance. We believe our findings will help guide the future designs of high-performance peer-to-peer systems.

***Index Terms*— Network measurements, peer-to-peer systems**

I. Introduction

Research on systems based on the peer-to-peer architecture such as distributed hash tables [1][2][3][4], file sharing systems (e.g. Napster and Gnutella) and overlay multicast systems [5][6] has flourished. The peer-to-peer architecture promises freedom for innovations and rapid large-scale deployment. This architecture is appealing also because it challenges the limit on what powerful systems can be built end-to-end [7] without the provisioning of special network resources or extending the minimal unicast datagram service model of the Internet. These fundamental driving forces ensure an important place for peer-to-peer systems in the future.

One of the core challenges in building peer-to-peer systems is how to achieve high performance efficiently. For systems that require low communication latency only (such as distributed hash tables), there has been previous work addressing the performance challenges [8]. In contrast, previous work has not explored the solution space for systems that are bandwidth-demanding (such as media file sharing and overlay multicast streaming). For these systems, a basic strategy is to measure the network using light-weight techniques and use the measured information to organize peers such that whenever possible data traffic is transmitted between peers who are connected by high bandwidth low congestion network paths. Despite the importance of this problem, to date, we have very little systematic knowledge of how well this basic strategy may work in practice.

Although some characteristics of peer paths such as bottleneck bandwidth and round-trip time distributions have been studied in detail [9], we are still a step away from understanding how these properties translate into performance. Fundamentally, to understand how to achieve high performance in bandwidth-demanding peer-to-peer systems, we must study how one can identify peer paths with high TCP available bandwidth. To add another dimension, some bandwidth demanding peer-to-peer systems are continuously adaptive (e.g. overlay multicast streaming) while others are not (e.g. media file sharing), hence the fidelity of network information they need to achieve high performance can be vastly different.

The goal of this paper is to provide systematic quantitative information about light-weight measurement-based optimization techniques in the peer-to-peer environment through trace-based experiments and Internet-based experiments. We believe this information is valuable to the designers of high-performance peer-to-peer systems. We study three basic light-weight techniques (round-trip time probing, 10KB TCP probing, and bottleneck bandwidth probing) at multiple levels. At the basic level, we treat the techniques as "black boxes" and quantify how well they can identify a peer with high TCP available bandwidth. To deepen our understanding, we conduct analyses to address a set of important questions. For instance, are there unique aspects of the peer-to-peer environment that contribute to the performance of a technique?

This research was sponsored by DARPA under contract number F30602-99-1-0518, and by NSF under grant numbers Career Award NCR-9624979, ANI-9730105, ITR Award ANI-0085920, and ANI-9814929. Additional support was provided by Intel. Views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of DARPA, NSF, Intel, or the U.S. government.

What are the fundamental limitations of a technique? Can adaptive applications benefit from using the basic techniques? Can we simultaneously exploit multiple basic techniques? Finally, at the system level, we use media file sharing and overlay multicast streaming as two case studies to demonstrate the benefits of different light-weight techniques in real applications.

Our results show that peer selection is an inherently hard problem due to the extreme heterogeneity in the peer-to-peer environment, and the basic techniques are limited because their primary strength lies in eliminating the low-performance peers rather than reliably identifying the best-performing one. Despite these findings, our analyses also reveal two key insights that can potentially be exploited by applications. First, for adaptive applications that can continuously change communication peers, the basic techniques are highly effective in guiding the adaption process. In our experiments, typically an 80% optimal peer can be found by trying less than 5 candidates. Secondly, we find that the basic techniques are highly complementary and can potentially be combined to better identify a high-performance peer, thus even applications that cannot adapt may benefit.

We have systematically experimented with several simple combined peer selection techniques in media file sharing and overlay multicast streaming. Our results show that for the non-adaptive media file sharing application, a simple combined technique can boost average performance from 40-50% optimal to 60% optimal and double the worst case performance. In contrast, for the continuously adaptive overlay multicast application, we find that a basic technique with even low-fidelity network information is sufficient to allow 30% more peers to converge to good performance within 5 seconds. This highlights the different needs of applications depending on their characteristics.

The rest of this paper is organized as follows. In Section II, we briefly describe the media file sharing system and the overlay multicast streaming system that we are studying. We describe the methodology of our work in Section III and present our analytical findings in Section IV. In Section V, we turn our attention to study the application specific use of light-weight measurement-based techniques in both media file sharing and overlay multicast streaming systems. We present related work in Section VI. Finally, we summarize our work in Section VII.

II. Bandwidth-Demanding Peer-to-Peer Systems and Performance Metrics

To initiate our discussion, we briefly review the basic functionalities of the two bandwidth-demanding peer-to-peer systems that we are studying and their corresponding performance evaluation metrics.

A. Media File Sharing

We assume a Napster-like media file sharing system. In such a system, a client peer issues a request for a desired media file to a directory service. The directory service then returns a list of server peers (potentially a large number of them) that have the wanted media file. The client peer will then pick a server peer to download the media file from via TCP. The goal of the client peer is to choose a server peer that can achieve the highest TCP throughput. For simplicity, we assume that the client peer is not continuously adaptive. That is, once it chooses a server peer, it downloads the entire media file from that server peer.

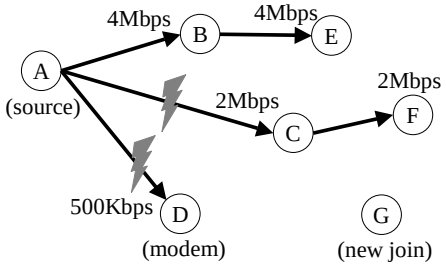


Fig. 1. Example of an overlay tree where *A* is the source.

1) Performance Metric: To quantify how well a client peer picks a server peer, we use a metric called the *optimality ratio* (O.R.). Given a set of candidate server peers to choose from, the O.R. is simply the ratio between the TCP bandwidth achieved by downloading from the selected server peer and the TCP bandwidth achievable from the best server peer in the candidate set.

B. Overlay Multicast Streaming

In overlay multicast streaming, peers self-organize into an overlay tree rooted at the source, and data is sent along links in the overlay tree. We assume traffic on each overlay link is congestion controlled by running a TCP-friendly protocol for streaming media applications [10]. An example of an overlay tree is depicted in Figure 1.

Note that overlay multicast streaming is continuously adaptive, and the need for intelligent peer selection can occur in two situations. First, when a member joins the group, it needs to decide which parent it should attach itself to in the overlay tree (consider the newly joining peer *G* in Figure 1). Second, even after making an initial decision, members still need to periodically re-evaluate their choice of parent in the overlay tree. For example, in Figure 1, *C* is not satisfied with its choice of parent as it does not receive the source rate, and hence considers moving to a better performing parent.

As a first step in deciding which peer to select as the parent, we can leverage on knowledge specific to the application. First, a member does not select any descendant as a parent. Second, a member can filter out those peers that receive worse application level performance. For example, in Figure 1, *C* filters out its descendant *F*, and filters out *D* because its observed performance is poor. It is now left with the task of choosing one of *B* and *E*.

1) Performance Metric: To select a good performance metric for overlay multicast streaming, we need to consider some of its unique characteristics. First of all, a peer is satisfied with a parent peer as long as it can receive data at the source rate. The maximum achievable throughput is not important. Secondly, as we will show in Section V-B.3, due to the

continuously adaptive nature of overlay multicast streaming, given sufficient time, the overlay structure will converge to a stable configuration, where each peer is receiving data at as high a quality as possible. Thus, the key performance metric is the *convergence time*.

We define the convergence time of a peer to be the amount of time after the initial join it takes for the peer to receive more than 95% of the *stable bandwidth* for 30 seconds. We determine the stable bandwidth of a peer based on the bandwidth it receives at the end of a 5-minute experiment. The 30 second window is necessary because a peer's performance may momentarily dip below 95% of the stable bandwidth due to network performance fluctuations.

III. Methodology

This study is about understanding the peer-to-peer system performance optimization problem at multiple levels. At the analytical level, we want to know how much a light-weight technique such as 10KB TCP probing generally helps to pick a good neighbor peer for data transfer. We also study the properties of light-weight techniques to shed some new insights about their behavior. At the system level, we focus on two different bandwidth-demanding peer-to-peer systems – media file sharing and overlay multicast streaming – and try to find good optimization strategies for them.

To achieve these goals, we use trace-based analyses, trace-based experiments, and Internet-based experiments. First, to conduct a reasonably general analytical study of the various optimization techniques, we need to repeatedly experiment with a large collection of realistic Internet peers. Thus instead of on-line experimentation, we have collected several peer performance traces on the Open Napster system (an open source version of Napster). These traces allow us to conduct basic analyses of measurement-based techniques, and to conduct off-line experiments to show how well different techniques perform in the media file sharing system. The details of these traces are described in Section III-A below. Since the peer traces are not collected at one instant in time, a necessary assumption we make when analyzing the performance attributes of peers in these traces is that the underlying processes that govern the relationships among the peer performance attributes were unchanged over the trace collection period.

In contrast, network traces cannot facilitate the study of overlay multicast streaming because the complexity of the dynamics cannot be accurately modeled from network traces. Therefore, to study the performance of overlay multicast streaming, we conduct Internet-based experiments on a distributed and heterogeneous Internet testbed, which is described in Section V-B.

A. Peer Traces Collection

From October 2001 to January 2002, we collected network performance data of roughly 10,000 peers distributed around the world. We used 4 data collection hosts located in Carnegie Mellon University (CMU), University of Illinois Urbana-Champaign (UIUC) and University of Alberta, Canada. Their characteristics are described in Table I. To enable our data collection hosts to interact with a large number of realistic peers in the Internet, we took advantage of individual operated Open Napster servers. Open Napster servers are central directories where Open Napster clients can log-on to submit databases about their digital music collections, and search each other's music collections for download. We used an instrumented Linux-based Open Napster client to log-on to any of a list of roughly 15 Open Napster servers, and repeatedly searched for common words (drawn from a list of 100) such as "they", "me" and "he". After each search, up to 20 resulting matches were returned, and the client would attempt to collect data about the 2nd, 10th and 19th peers in the resulting list. Note that the Open Napster platform is quite ideal for collecting performance traces because the control traffic overhead of the Napster protocol is very low and does not prevent slower peers from participating. The result is that we are able to collect information about a truly diverse set of peers.

Host	Location	Link Speed	# Peers	TCP Avg
1	CMU	10 Mbps	2705	129 kbps
2	CMU	640/90 kbps ADSL	880	111 kbps
3	UIUC	10 Mbps	3116	140 kbps
4	U of Alberta	10 Mbps	3805	130 kbps

TABLE I
COLLECTION HOST PROPERTIES

The data we collected about each peer consists of the following. First we recorded the round-trip times of ten 36-byte pings to the peer. Subsequently, 500KB of data was downloaded from the peer via TCP, and we recorded the total transfer time and the time-stamp for the first 10KB chunk, which is used to simulate the result of a 10KB TCP probe. When a download was successful, a traceroute of up to 30 hops was performed (note that this records the reverse path of the TCP data traffic). Besides these active measurements, we used the *nettimer* tool [11] at CMU (since *nettimer* requires super-user access to the collection host) to passively monitor the TCP traffic to obtain the approximate raw bottleneck link bandwidth of the data path. We collected the DNS host names of the peers and the geographical locations of the peers using the NetGeo tool [12] from CAIDA off-line. The traceroute data was used to compute the AS hop count in conjunction with an IP address prefix to AS number database.

When a peer is sampled multiple times by the same collection host, only the last sample is used in the analyses that follows. Also, because our trace collection hosts are all connected to the Internet 2 high speed backbone, to eliminate this bias, we remove all peers that are also connected to the Internet 2 backbone in our trace.

There are a few interesting observations about the data sets. Based on the NetGeo data, we found that 49% of the peers we encountered are in North America, 30% are in Europe, 14% are in Asia, 2% in Oceania, and 1.2% in South America. This shows that the set of peers we study is a very good sample of the global Internet. An interesting fact we have observed is that, the North American peers' average available bandwidth does not decrease during business hours (Monday to Friday, 9am to 8pm Eastern time) like other systems such as web servers; in fact the average bandwidth

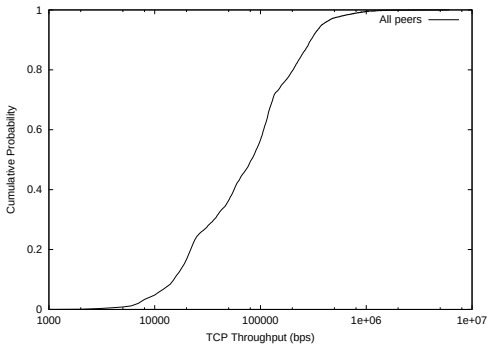


Fig. 2. TCP Throughput distribution of peers.

Host	Location	Random	36B RTT	10KB Probe	BNBW
1	CMU	0.13	0.42	0.47	0.48
2	CMU (ADSL)	0.24	0.52	0.66	0.33
3	UIUC	0.13	0.27	0.43	N/A
4	U of Alberta	0.15	0.40	0.48	N/A

TABLE II

AVERAGE O.R. OF BASIC TECHNIQUES (95% CONFIDENCE INTERVALS $<$ 0.02)

Host	Location	Last Hop RTT	Network Hops RTT
1	CMU	72%	22%
2	CMU (ADSL)	68%	43%
3	UIUC	141%	70%
4	U of Alberta	113%	106%

TABLE III

PERFORMANCE (BY O.R.) OF DECOMPOSED RTT AS A PERCENTAGE OF PERFORMANCE OF FULL PATH RTT

increases! The reason is that slower peers (typically dial-up modem and ADSL modem connected hosts in people's homes) are not encountered as often during business hours. Thus the composition of the peer population is changing during the course of the day and the week, and the effect is reflected in the average observed performance. Finally, in Figure 2, we plot the cumulative distribution of TCP available bandwidth of all peers. Observe the characteristic shifts at 30 kbps and 150 kbps in the distribution correspond well with the bandwidth ranges of dial-up modem and ADSL modem technologies. Clearly a large portion of Internet peers have very limited performance.

IV. ANALYSES OF PEER SELECTION TECHNIQUES

In this section, we present analytical results for three basic light-weight peer selection techniques, namely round-trip time (RTT) probing, 10KB TCP probing, and bottleneck bandwidth (BNBW) probing. Our goal is not only to quantify the performance of these techniques, but also to understand the techniques' inner-workings, weaknesses, and complementarity. We believe this level of understanding will reveal the inherent difficulty of the peer selection problem and provide valuable guidelines to peer-to-peer application designers.

In RTT probing, we measure the RTT to each candidate peer using a *single* 36 byte ICMP ping message; the candidate peer with the smallest RTT is then selected. In 10KB TCP probing, a TCP connection is opened to each candidate peer and 10KB of data is downloaded; the peer that provides the fastest download is selected. In BNBW probing, *nettimer* [11] is used to measure the BNBW to each candidate peer; the peer with the largest BNBW is selected.

Note that we have also analyzed other simple techniques such as hop count probing, autonomous system hop count probing, and NetGeo geographical distance probing. We found that AS hop count probing has performance similar to random peer selection, while hop count probing and NetGeo geographical distance probing have performance that is less than half of the RTT probing technique. Therefore, in this paper, we will not focus on these alternative techniques.

A. Performance Analyses of Basic Techniques

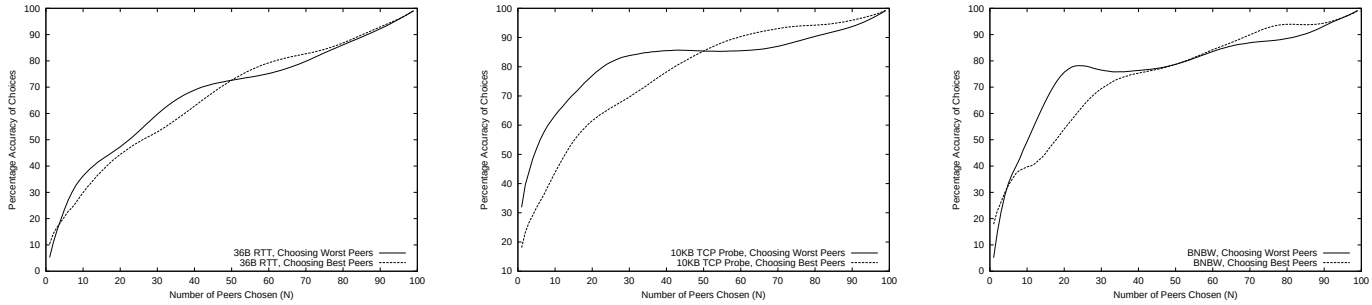
To provide some basic understanding of the peer selection techniques, we compute their optimality ratios (O.R.) (defined in Section II) using our peer traces. First, we randomly pick 100 peers from a peer trace. Then based on the recorded probe values in our trace, the three techniques are applied in turn to select a peer. Given the best candidate peer and the selected peer's TCP available bandwidth values as recorded in our trace, the O.R. of each technique can be computed. This experiment is repeated 1000 times and the mean O.R. and the 95% confidence interval are reported.

Table II is a summary of the results. The main observation is that these basic techniques can typically achieve 40 to 50% of optimal performance (or 3 to 4 times better than random selection). Somewhat surprising is that 10 KB TCP probing and BNBW probing perform only a bit better than RTT probing. We will turn to explore this finding in Section IV-A.1.

Host 2 is a special case since it only has a 640/90 kbps ADSL Internet connection. Because of the relatively low maximum possible throughput, the diversity of the peers' performance is greatly reduced and so the peer selection problem is generally easier (reflected in the higher O.R. values). However, even so, RTT probing and 10KB TCP probing are still worthwhile because they do provide substantial improvements over random selection. In contrast, BNBW probing does not work well simply because the measurements are upper bounded by the ADSL host's own access bottleneck and thus BNBW probing has lost most of its differentiation power.

1) Understanding RTT Probing in Peer-to-Peer: An interesting question to raise is, how come RTT probing can perform comparably to 10KB TCP probing and BNBW probing? How does low RTT translate into high TCP throughput? Is it merely because TCP performs better when RTT is small [13]? Our conjecture is that this result has a lot to do with the differences in network access technologies used by many peers today. By performing some simple measurements, we learn that while an Ethernet connection has a one hop RTT of less than 1 ms, the one hop RTT of a dial-up modem connection is typically over 150 ms, that of an ADSL modem connection is typically over 15 ms, and that of a cable modem connection is typically over 7 ms. This demonstrates that the overall RTT value to a peer may have an unusually high correlation to its access speed, therefore the predictive power of RTT probing may also be significantly enhanced.

To better understand this phenomenon, we decompose the peer-to-peer path RTT into two parts: (1) the last hop RTT at



(a) (b) (c)

Fig. 3. Percentage accuracy of choices versus number of peers (best or worst) chosen. (a) 36B RTT probing. (b) 10KB TCP probing. (c) BNBW probing.

the candidate peer, and (2) the RTT of the remaining network hops. To compute these values, we use the minimum of the 3 path RTT measurements from our traceroute data as the regular full path RTT, and the minimum of the 3 traceroute RTT measurements up to the second last hop as the network hops RTT. We consider only the cases where the resulting full path RTT is greater than the network hops RTT, and compute the last hop RTT as the difference of the two.

Table III summarizes the performance of using the decomposed RTT values for peer selection as a percentage of the performance of using the full path RTT. What is striking is that the last hop RTT value can predict the TCP throughput much better than the network hops RTT. In the case of the UIUC trace, the last hop RTT can even perform significantly better than the full path RTT. In the U of Alberta case, the decomposed RTT values perform similarly to the full path RTT. Clearly the predictive power of RTT in the peer-to-peer environment is mainly coming from the last hop.

B. Limitations of Basic Techniques

Although we have quantified the performance of the three basic techniques, ultimately, we want to gain a deeper understanding of how the techniques work and their limitations. In this section, we first discuss the inherent difficulty of the peer selection problem, then we analyze the properties of the techniques and discuss their limitations.

1) Inherent Difficulty of Peer Selection: While the 40 to 50% optimal performance achieved by the basic techniques does not seem very impressive, we must also consider the inherent difficulty of the peer selection problem. From Figure 2, we can see just how diverse the performance characteristics of peers are. In the experiments we have conducted on the 10 Mbps CMU trace, we find that the O.R. of the second best peer choice among the 100 random candidates is already only 0.73, the O.R. of the third best choice is only 0.60, and the O.R. of the fourth best choice is 0.51. As can be seen, the drop in performance by picking a slightly lower ranked peer is huge. Thus, all things considered, the basic techniques are actually performing fairly decently.

2) Peer Selection or Peer Elimination?: Our intuition is that a good technique must do two jobs well, the first is to identify peers with good performance, and the second equally important job is to avoid peers with poor performance. What we are interested in is, do the basic techniques work mostly by making good selections, or do they work mostly by eliminating the bad choices? Answering this question will also help shed light on the limitations of these techniques.

To address this question, for each technique, we conduct 1000 experiments on the three 10 Mbps peer traces, each with a different set of 100 randomly chosen peers. In each experiment, we apply a technique to choose the best N peers and the worst N peers, where $N \in (1..99)$. For each experiment, we compute the percentage accuracy of the choices by counting the number of choices that are indeed correct. Then we average the percentage accuracy for each N across the 1000 experiments. To summarize the overall result across all three traces, we take another average of the percentage accuracies from each trace. By comparing a technique's accuracy in choosing the best peers versus the worst peers, we can observe whether the technique works by selection or by elimination.

In Figure 3, for each technique, we plot the cross-trace average percentage accuracy against N for selection of best peers and selection of worst peers. We note that only the 10 Mbps CMU trace has data for BNBW probing. Observe that the two curves in each graph are complementary to each other and thus they cross over at N equals 50. We will focus our attention on the region where N is between 1 and 50. Recall that a good technique must both accurately select good peers and eliminate poor peers. Clearly, none of the three techniques are extremely good at either tasks. In fact, all three techniques are generally more accurate at selecting the worst peers than at selecting the best peers. The only exception is when N is less than 4, RTT and BNBW probing are slightly better at selecting the best peers.

These results reveal the fundamental limitations of the three techniques. For RTT probing, it is mediocre at both selection and elimination because in many cases RTT simply cannot provide very accurate information about TCP throughput. On the other hand, we can see that a 10KB TCP probe is quite good at identifying the poorly performing peers, but the probe size is still too small to accurately measure the best peers. Similarly, BNBW can identify peers that are limited by the bottleneck link bandwidth very well, but when the bottleneck link bandwidth is sufficiently large, many other dynamic factors such as network congestion will heavily influence a peer's performance, and they are not captured by the BNBW probe.

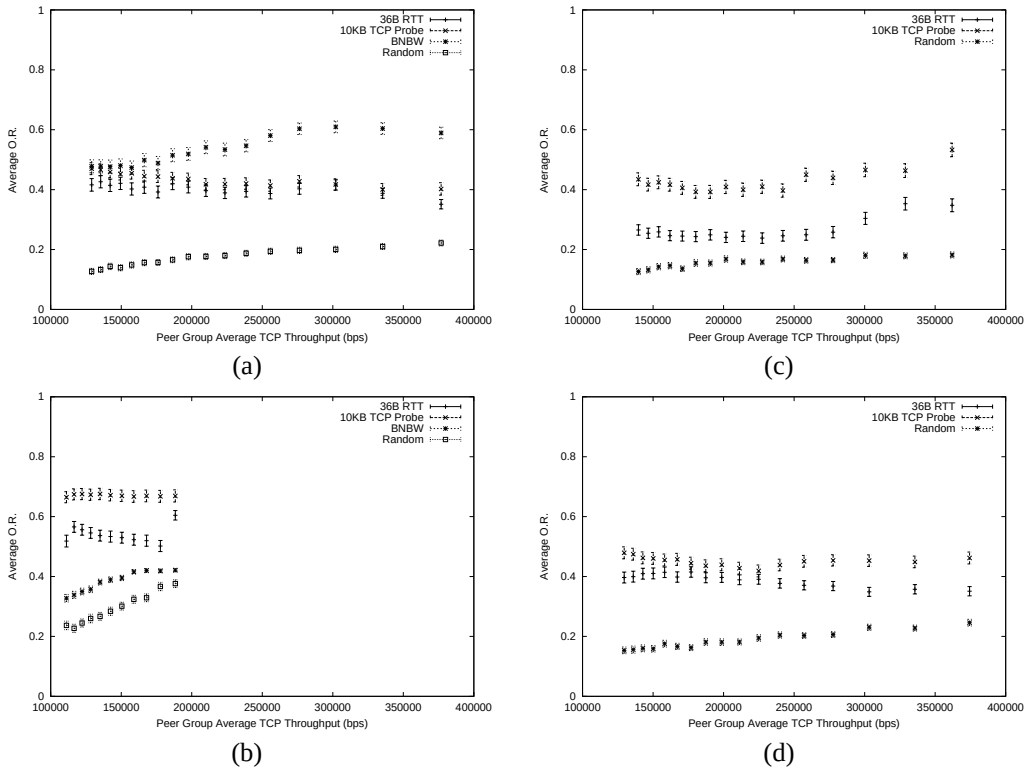


Fig. 4. Average O.R. of techniques, with 95% confidence interval, versus average TCP throughput of groups of peers. (a) CMU 10 Mbps trace. (b) CMU ADSL trace. (c) UIUC trace. (d) U of Alberta trace.

3) Inability to Differentiate Good Peers: With the intuition given in the previous section, we want to quantify how well basic techniques can differentiate among the good peers. To do this, we conduct experiments with different groups of peers, created by progressively removing 5% of the worst performing peers at a time, so that we are gradually left with only good peers. In Figure 4, we plot the average O.R., with 95% confidence intervals, for each technique in each trace against the average TCP throughput of the resulting peer groups.

The first thing to notice is that as the worst performing peers are removed, the peers' performance gradually becomes more homogeneous. As a result, the O.R. of random peer selection increases noticeably. Since the peer selection problem is becoming easier, we might expect a blanket improvement in performance for all probing techniques similar to that for random selection, but this is clearly not the case. In fact, only BNBW probing shows improvement when the worst peers are removed. The reason is as follows. The ADSL CMU trace is a special case, where BNBW probing is similar to random selection because the BNBW is locally limited. Thus the performance gain observed is also consistent with that of random selection.

In the 10 Mbps CMU trace, we know 50% of the peers have TCP throughput of less than 82 kbps, and in 125 of the 1000 experiments on all peers, BNBW probing picks a peer with less than 82 kbps. Thus after removing the major "false positive" slow peers, BNBW probing is able to pick much better peers. Note that this performance gain stops once all major "false positives" are removed from the data set.

In contrast, RTT probing and 10KB TCP probing do not benefit from removing the slow peers. In the 10 Mbps CMU trace, in only 18 of the 1000 experiments 10KB TCP probing picks a peer with less than 82 kbps, thus the effect of the removal of slow peers is minimal. For RTT probing, although in 144 of the 1000 experiments it picks a peer with less than 82 kbps, even after the slower peers are removed, RTT probing is still not good at picking a peer that is much better.

These results show that the basic techniques (including BNBW probing once all major "false positives" are removed) are mainly limited by their inability to accurately differentiate among the good peers. This is the main reason why the O.R. performance of these techniques is only 40 to 50% of optimal.

Host	Location	Random	36B RTT	10KB Probe	BNBW
1	CMU	0.33	0.82	0.80	0.83
2	CMU (ADSL)	0.52	0.91	0.90	0.58
3	UIUC	0.33	0.58	0.81	N/A
4	U of Alberta	0.35	0.69	0.80	N/A

TABLE IV
AVERAGE O.R. OF BEST PEER AMONG TOP 5 CANDIDATES (95% CONFIDENCE INTERVALS < 0.02)

Host	Location	Random	36B RTT	10KB Probe	BNBW
1	CMU	0.46	0.86	0.91	0.95
2	CMU (ADSL)	0.67	0.95	0.96	0.69
3	UIUC	0.44	0.80	0.95	N/A
4	U of Alberta	0.47	0.83	0.91	N/A

TABLE V
AVERAGE O.R. OF BEST PEER AMONG TOP 10 CANDIDATES (95% CONFIDENCE INTERVALS < 0.02)

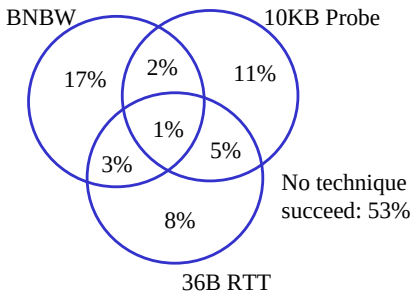


Fig. 5. Percentage of cases where each technique chooses a peer within 90% of optimal.

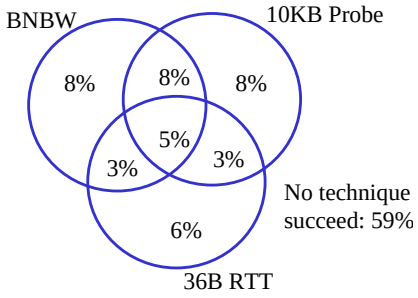


Fig. 6. Percentage of cases where, after using RTT to select 5 candidates, each technique chooses a peer within 90% of optimal.

C. Using Basic Techniques in Adaptive Applications

To cope with the limitations of light-weight peer selection techniques, one strategy is to adaptively select a peer based on observed TCP performance. For example, in the overlay multicast streaming application, a peer can adapt by changing its parent peer selection until a satisfactory parent peer is found.

In this section, we present analytical results to show how well light-weight peer selection techniques can potentially guide adaptive applications in finding a good peer. As before, we conduct 1000 experiments, each with 100 randomly chosen candidate peers. Table IV shows the average O.R. of the chosen peer if an adaptive application is able to observe the performance of the top 5 candidate peers recommended by each technique. Table V shows the results when the top candidate set size is increased to 10. Except for the BNBW probing technique in the bottleneck bandwidth limited ADSL case, we can see that all the light-weight techniques can successfully guide an adaptive application in finding an 80% optimal peer in less than 10 tries, and in many cases this is achieved in less than 5 tries. Moreover, simple RTT probing is only slightly less effectively than the other bandwidth probing-based techniques, making RTT probing a good practical choice. These results demonstrate that adaptive peer selection guided by light-weight techniques is a promising approach to improve the performance of peer-to-peer systems.

D. Complementarity Analyses of Basic Techniques

So far, we have analyzed the three basic peer selection techniques in isolation. A key question is, can we do better than these basic techniques? Although we have learned that these techniques have comparable performance, we do not know whether the information revealed by each technique is mutual or complementary. To understand how much complementary information is revealed by the three basic techniques, we further analyze the results of our peer selection experiments over the 10 Mbps CMU trace. The idea is to compute the percentage of time where each technique is selecting a peer that is within 90% of the optimal peer, and how often the techniques agree in picking the same 90% optimal peer. The result is a Venn diagram as shown in Figure 5.

First, notice that individually, RTT probing can pick a 90% optimal peer 17% of the time, 10KB TCP probing can pick a 90% optimal peer 19% of the time, and BNBW probing can pick a 90% optimal peer 23% of the time. In contrast, if we can leverage the ability of all three techniques, we would be able to pick a 90% optimal peer 47% of the time, more than double of what each individual technique can achieve. Clearly the three techniques complement each other. For instance, only 6% of the time RTT probing and 10KB TCP probing agree with each other when picking a 90% optimal peer. In terms of O.R., in a separate analysis, we find that if we always follow the recommendation of the most successful technique among the three, we can achieve an O.R. of 0.73. We have also conducted these analyses with the other 3 traces. While BNBW probing does not work well in the CMU ADSL trace (see Section IV-A for the explanation), we still observe that RTT probing and 10KB TCP probing are highly complementary in the other 3 traces.

1) Practical Considerations: Although we have revealed the possibility of exploiting the three basic techniques simultaneously to achieve better performance, a practical concern is that probing all candidate peers using all three techniques may consume a significant amount of time, ultimately reducing the actual benefits. What we are interested in is, how much performance may be lost if we use inexpensive RTT probing to eliminate most of the candidate peers from consideration to save time?

To answer this question, we conduct a new set of 1000 experiments on the 10 Mbps CMU trace. In each experiment, we use RTT probing to eliminate 95 candidate peers from consideration. Then we apply the 10KB TCP probing and BNBW probing techniques to choose among the remaining 5 candidate peers. In Figure 6, we again show the results as a Venn diagram of the percentage of time each technique picks a peer within 90% optimal after the RTT-based filtering. Of course the optimal peer is still defined as the best peer among the original 100 candidates.

Comparing to Figure 5, although the amount of overlap between techniques has changed after the RTT-based filtering, the overall chance of selecting a 90% optimal peer is only reduced by 6% to 41%. In terms of O.R., again in a separate analysis, we find that if we always follow the recommendation of the most successful technique, we can still achieve an O.R. of 0.68. Thus RTT-based filtering is an excellent way to reduce the overhead without significantly reducing performance. Another interesting observation is that RTT-based filtering actually does not negatively affect the performance of 10KB TCP probing and BNBW probing techniques. Their

individual accuracies are both 24%, slightly higher than in Figure 5. Therefore, we see that the reason for the 6% reduction in overall accuracy is mainly due to the fact that some complementarity is lost (i.e. more overlapping between techniques exists) by applying the RTT-based filter.

V. Application Case Studies

We have quantified the performance of the basic peer selection techniques and analytically shown the potentials of exploiting them simultaneously. However, how much sophistication is needed and how to apply the techniques are ultimately application dependent. Therefore in this section, we conduct two case studies with two very different peer-to-peer applications. The first is media file sharing, which is non-adaptive and must make the best peer selection possible. The second is overlay multicast streaming, which is a self-organizing application that continuously adapts itself to improve performance. Our main goal is to find out what are the best performance optimization strategies for these two different applications. We believe these case studies can provide some guidelines for future application designs.

A. Media File Sharing

As discussed in Section II, a peer in the media file sharing application tries to pick the best candidate peer to download a media file from. Because we assume the entire media file is downloaded from one chosen peer, it is important to make the choice carefully. In Section IV-A, we have essentially analyzed how well individual techniques can improve the performance of media file sharing. Therefore, in this section, we focus on how much performance can be gained by using the techniques in an integrated fashion.

We propose *joint ranking* as one plausible way of exploiting the complementarity of the basic techniques and evaluate its effectiveness in the media file sharing application. The idea of joint ranking is very simple. In an experiment, we first rank the candidate peers based on each technique. For each candidate peer, we sum up the rank values assigned by the basic techniques. We then choose the peer with the smallest sum of ranks. In essence, the chosen peer should have good ranking by all techniques, thus we are less likely to choose a bad peer that just happen to have high bottleneck bandwidth, for instance. We conduct these experiments on the 10 Mbps CMU trace and each experiment is repeated 1000 times as usual, and the average O.R. results are reported.

Table VI summarizes the results from a set of systematic experiments that we have conducted. For comparison, the top portion of the table is simply a copy of the results for individual techniques from Section IV-A. The middle portion of the table shows the performance of different joint ranking techniques when applied to all 100 candidate peers in each experiment. Finally, the bottom portion of the table shows the performance of various techniques when applied to only 5 candidate peers that have the smallest RTT among the 100 candidates.

Technique		O.R.
100 candidates	(a) Random	0.13
	(b) 36B RTT	0.42
	(c) 10KB Probe	0.47
	(d) BNBW	0.48
100 candidates	(e) 36B RTT joint BNBW	0.49
	(f) 36B RTT joint 10KB Probe	0.54
	(g) BNBW joint 10KB Probe	0.50
	(h) 36B RTT joint BNBW joint 10KB Probe	0.55
5 out of 100 candidates by 36B RTT	(i) Oracle	0.82
	(j) 10KB Probe	0.53
	(k) BNBW	0.50
	(l) 36B RTT joint BNBW	0.51
	(m) 36B RTT joint 10KB Probe	0.55
	(n) BNBW joint 10KB Probe	0.55
	(o) 36B RTT joint BNBW joint 10KB Probe	0.58

TABLE VI

Average O.R. of joint ranking techniques (95% confidence intervals < 0.02)

1) Benefits of Joint Ranking & RTT Filtering: First, consider the middle section of Table VI, where various joint ranking techniques are applied to all 100 candidate peers. We can see that, though not by much, all joint ranking techniques out-perform the individual basic techniques. This demonstrates that we can indeed benefit from the complementarity of the basic techniques. However, the reason why the average O.R. is not very high is discussed in Section V-A.2.

Now consider the bottom portion of Table VI. Our main motivation to use RTT to pick 5 candidates is to reduce the measurement overhead of BNBW probing and 10KB TCP probing techniques. However, one pleasant surprise is that the performance of all techniques actually improves slightly with the RTT filter. For example, comparing case (j) to case (c), 10KB TCP probing performs 13% better with the RTT filtering. This may seem contradictory to the results presented in Section IV-D.1. Although RTT filter reduces the theoretical maximum performance, it actually helps improve the O.R. of sub-optimal choices, thus the overall effect on O.R. can be positive.

2) Best Technique for Media File Sharing: The best technique we have found turns out to be case (o) in Table VI, where RTT is used to first select 5 candidates, then the three basic techniques are used in joint ranking. This combined technique has an O.R. of 0.58, which is a reasonable improvement over the basic techniques. In addition, this combined technique has significantly better robustness. More precisely, the 20 percentile O.R. for RTT probing is 0.09, in contrast the 20 percentile O.R. for this combined technique is 0.23. We believe this combined technique is suitable for the media file sharing application since its overhead is not much more than RTT probing, and the average performance gain over RTT probing is over 38% (i.e. from O.R. of 0.42 to 0.58).

Note that an O.R. of 0.58 is still far from the theoretically possible O.R. of 0.82 (case (i)), where an oracle is picking the best peer from the 5 remaining candidates. We suspect this is a fundamental limitation of using light-weight measurement-based techniques for performance optimization. Light-weight measurement-base techniques again are generally good at eliminating bad choices, but are not very reliable in picking the best one. More precisely, we found that 73% of the time, the combined technique (o) can rank the best of the 5 candidates

either as the best or the second best. However, only 44% of the time it correctly ranks the best candidate at the top. Additionally, the drop in performance from the best candidate to the second best candidate among the 5 candidates is very large. On average, the second best candidate among the 5 has performance only 60% of the best candidate. Thus, it is not hard to see that in order to do much better than an O.R. of 0.58, a technique must be able to choose the best peer correctly extremely often, which unfortunately is very difficult because the light-weight measurements cannot provide this level of pin-point accuracy.

B. Overlay Multicast Streaming

In this section, we evaluate the effectiveness of different peer selection techniques in overlay multicast streaming. Compared to media file sharing, the key advantage of overlay multicast streaming is that it can continuously adapt to achieve better performance. Given overlay multicast protocols can dynamically adapt when performance is sub-optimal, is it necessary to use any intelligent peer selection technique? If so, are sophisticated techniques based on combining basic techniques required?

1) Evaluation Methodology: To conduct our study, we extend Narada [14], an overlay multicast protocol that adapts to both bandwidth and latency, to use various peer selection techniques to optimize performance. Our study is conducted on a wide-area heterogeneous Internet testbed consisting of 29 hosts. 22 hosts are in North America and 7 hosts are outside. Of the 22 hosts in North America, 16 hosts are ".edu" hosts. There are 3 hosts connected by ADSL modems.

To compare the performance of different peer selection techniques, we adopt the methodology suggested in [14]. We interleave experiments with the various techniques that we compare, and repeat the same set of experiments at different times of the day. We then aggregate the results obtained from several runs. Given that experiments with various techniques cannot be conducted concurrently, such an approach helps to minimize biases due to the varying nature of Internet performance.

Every individual experiment is conducted in the following fashion. Members join the group in a sequential order, spaced by 2 seconds apart. The order in which members join is randomized in each experiment. The source multicasts data at a constant rate of 1.5Mbps. Each experiment lasts for 5 minutes.

2) Implementation of Peer Selection Techniques: We describe below how the three basic peer selection techniques are implemented in our overlay multicast protocol. In addition, we contrast with random peer selection (*Random*) to helps us understand if any measurement-based technique is required while selecting peers.

- *RTT*: This technique uses single-packet RTT probe to select peers. In our experiment, we conduct RTT probes in parallel to all peers of interest, and select the one with the smallest RTT value.
- *RTT filter + 10K*: This technique selects at most 5 of the candidate peers based on the best RTT estimates and perform 10KB of file transfer using TCP in parallel. The peer with the shortest transfer time is selected. Use of 10KB probes has been suggested in Overcast [6].
- *RTT filter + 1-bit BNBW*: This technique selects at most 5 peers with the lowest RTT estimates and then chooses the peer with the highest bottleneck bandwidth. Ideally we would like to measure the bottleneck bandwidth between peers during our experiments. However, the bottleneck bandwidth measurement tools we are aware of all require super-user access, which is a privilege we do not have on the majority of our testbed machines. Hence, we decide to use only a single bit to differentiate between ADSL and non-ADSL peers and study how much can such minimal information help. Note that tie-breaking is according the RTT values.

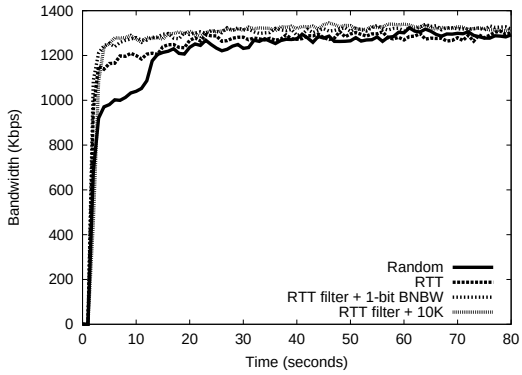


Fig. 7. Mean receiver bandwidth as a function of time at 1.5 Mbps source rate.

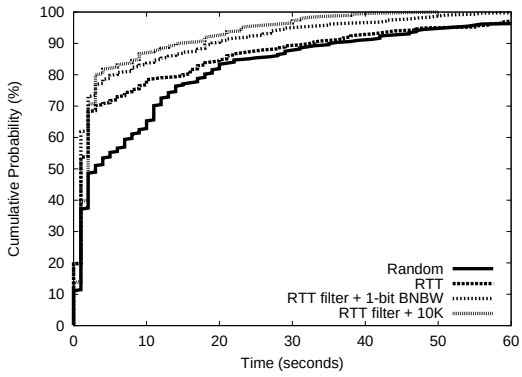


Fig. 8. Cumulative distribution of convergence time for basic techniques.

3) Results of Basic Techniques: While a self-improving overlay eventually converges to stable performance, an accurate peer selection technique can help to improve overlay convergence. This is shown in Figure 7, which plots the mean bandwidth as a function of time. Each curve corresponds to one technique. The mean bandwidth is averaged across all receivers and all experiments using that technique. For clarity, only the result of the first 80 seconds of the experiments is shown. We observe that for all techniques, the overlay reaches similar stable average performance within 30 seconds. As expected, random peer selection leads to a longer convergence time compared to techniques that exploit network information.

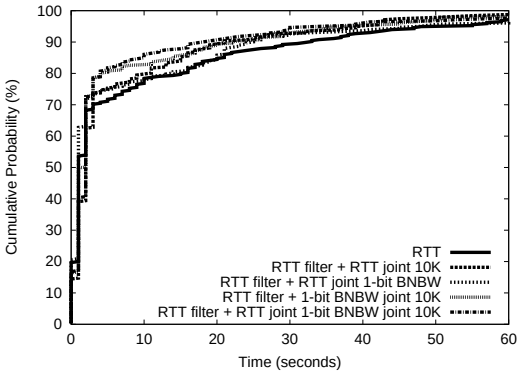


Fig. 9. Cumulative distribution of convergence time: a comparison between RTT and combined techniques. The lowest curve is RTT.

Next we zoom into the first 60 seconds of the experiments, and analyze how individual receivers perform at the initial stage of the multicast session. Figure 8 shows the cumulative distribution of convergence time (defined in Section II-B.1) for all receivers. As can be seen, *Random* results in longer convergence time with as many as 30% of the peers taking longer than 15 seconds to converge. This is because by selecting peers at random, it may take several trials before finding a peer that provides good performance. The use of a simple RTT technique greatly improves convergence time, with 70% of the peers take less than 5 seconds to converge. This is an indication that RTT can successfully help peers to select good parent peers when they first joined the group. Finally, RTT filter with either 10K or 1-bit BNBW results in better convergence property than RTT alone, with 80% of the peers take less than 5 seconds to converge. Note that even though our low-fidelity 1-bit BNBW metric can only differentiate the 3 ADSL host from the rest, it is shown to be already quite effective. This shows that with application adaptivity, even low-fidelity network information may be good enough.

4) Results of Combined Techniques: Our results so far indicate that simple light-weight techniques such as RTT filter with 1-bit BNBW can achieve good convergence properties compared to random peer selection. The question is whether the performance can be further improved by combining multiple techniques. We adopt a similar method of joint ranking as in Section V-A to combine multiple techniques. Figure 9 shows the cumulative distribution of convergence time for the combined techniques we compare. We observe that although all the combined techniques perform better than RTT alone (5 – 10% more peers converge in 5 seconds), the improvement is limited and is no better than simple techniques like RTT filter with 1-bit BNBW or RTT filter with 10K probe.

Based on these results we make the following observation: in adaptive applications such as overlay multicast, using a peer selection technique that has slightly better accuracy may not significantly improve application performance. Given that a peer stays in an overlay multicast group for at least a few minutes, the peer can always switch to another peers if the initial selection is bad. So the key to achieving good performance in overlay multicast is to apply useful hints such as RTT and 1-bit BNBW to allow peers to make quick adaptation decisions.

VI. Related Work

Performance studies of the peer-to-peer environment are at an early stage of research. Perhaps the most detailed study to date was conducted by Saroiu et al [9]. Their study characterizes the distribution of bottleneck bandwidths between peers, and the degree of cooperation among peers. A unique aspect of their work is that it exposes the behavior of human users in media file sharing systems. In contrast, our work is focused on optimizing for TCP throughput in peer-to-peer systems by using peer selection techniques. To our knowledge, our study is the first that systematically studies TCP throughput in the peer-to-peer environment, quantifies light-weight probing techniques' effectiveness for peer selection, and analyzes their intrinsic properties.

Parallel to the peer selection problem is the web server selection problem, and there has been numerous studies on this problem, ranging from server characteristics [15], system architecture [16][17], probing techniques [18][19][20][21][22], to parallel download techniques [23][24]. However, we believe that the peer-to-peer environment is quite different from the web server environment, and new studies are needed. The reason is that web servers in server selection problems are usually identical replicas, they are typically operated by the same organization and are likely to have homogeneous performance characteristics (e.g. network access speed, CPU speed, etc). In contrast, peers, by definition, are simply some random hosts out there on the Internet and as Saroiu et al and our study have shown, they are highly heterogeneous in their performance characteristics.

Our study is based on some of the same light-weight measurement-based selection techniques that have been studied in the server selection context, and using RTT as a filter has also been suggested before in server environments [22]. A new technique that we have considered is bottleneck bandwidth probing. This choice has been motivated by the fact that a significant fraction of the peers use bandwidth constrained access technologies such as dial-up modems, ADSL modems and cable modems. In addition, this heterogeneity in access technology has also motivated us to investigate the usefulness of the last hop RTT in revealing information about the performance of peers.

Our analysis methodology builds on top of those previously used in server selection studies. In addition to quantifying the "black box" performance of various selection techniques, we have also analyzed some of their intrinsic properties. For example, our analyses quantified the complementarity among basic techniques, which led to new combined techniques for improving performance. Our hope is that additional insights like this will help guide the designs of future peer-to-peer systems.

VII. Summary

To the best of our knowledge, this is the first study of light-weight measurement-based optimization techniques in the

peer-to-peer environment. By analyzing thousands of Internet peers and conducting trace-based and Internet-based experiments, we are able to shed quantitative insights on the peer selection problem that we believe are valuable to application designers. First, peer selection is inherently a very challenging problem due to the diversity in peers' performance. In fact the performance loss in picking the second best peer among 100 random candidates is already 30%. Even so, light-weight measurement-based techniques such as RTT probing, 10KB TCP probing and BNBW probing can perform reasonably well, achieving 40 to 50% optimal performance in media file sharing. We have also shown that the unique characteristics of the network access technologies used by peers today (e.g. dial-up modems, ADSL modems, cable modems, and Ethernet) actually contribute to the performance of RTT probing. The main factor that limits the performance of these techniques is that while they are fairly reasonable at eliminating poor peer choices, they are not very reliable in differentiating among the top peer choices. However, our analyses also reveal two key insights that can potentially be exploited by applications. First, for adaptive applications that can continuously change communication peers, the basic techniques are highly effective in guiding the adaption process. In our experiments, typically an 80% optimal peer can be found by trying less than 5 candidates. Secondly, we find that the basic techniques are highly complementary and can potentially be combined to better identify a high-performance peer, thus even applications that cannot adapt may benefit. We show that by combining all techniques together in a simple fashion, we can boost the performance of media file sharing to 60% optimal and double the worst case performance. However, we suspect this may be as close to optimal as a one-time peer selection based on light-weight techniques can achieve. This is due to the inaccuracies in differentiating top candidates based on light-weight measurements. In contrast, because overlay multicast streaming is continuously adaptive, even very basic techniques are quite sufficient in ensuring fast overlay convergence and robust performance. This highlights the different choices application designers must make depending on the adaptivity of the application.

ACKNOWLEDGEMENTS

We are deeply grateful to our contacts at over twenty institutions who allowed us to use their networks for collecting data and conducting experiments for this study. We also thank the anonymous reviewers for their helpful comments.

REFERENCES

[1] B. Zhao, J. Kubiatowicz, and A. Joseph, "Tapestry: An infrastructure for wide-area fault-tolerant location and routing," *U.C. Berkeley Technical Report UCB//CSD-01-1141*.

[2] A. Rowstron and P. Druschel, "Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems," in *IFIP/ACM International Conference on Distributed Systems Platforms (Middleware)*, 2001.

[3] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker, "A scalable content-addressable network," in *Proceedings of ACM SIGCOMM*, 2001.

[4] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan, "Chord: A scalable peer-to-peer lookup service for Internet applications," in *Proceedings of ACM SIGCOMM*, 2001.

[5] Y. Chu, S. G. Rao, and H. Zhang, "A case for end system multicast," in *Proceedings of ACM SIGMETRICS*, June 2000.

[6] 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)*, Oct. 2000.

[7] J. H. Saltzer, D. P. Reed, and D. D. Clark, "End-to-end arguments in system design," *ACM Transactions on Computer Systems*, vol. 2, no. 4, pp. 277–288, Nov. 1984.

[8] S. Ratnasamy, M. Handley, R. Karp, and S. Shenker, "Topologically-aware overlay construction and server selection," in *Proceedings of IEEE INFOCOM*, New York, NY, 2002.

[9] S. Saroiu, P. K. Gummadi, and S. D. Gribble, "A measurement study of peer-to-peer file sharing systems," in *Proceedings of Multimedia Computing and Networking (MMCN)*, Jan. 2002.

[10] S. Floyd, M. Handley, J. Padhye, and J. Widmer, "Equation-based congestion control for unicast applications," in *Proceedings of the ACM SIGCOMM*, Aug. 2000.

[11] K. Lai and M. Baker, "Nettimer: A tool for measuring bottleneck link bandwidth," in *Proceedings of the 3rd USENIX Symposium on Internet Technologies and Systems*, Mar. 2001.

[12] D. Moore, R. Periakaruppan, and J. Donohoe, "Where in the World is netgeo.caida.org," in *Proceedings of the Internet Society Conference (INET)*, 2000.

[13] J. Padhye, V. Firoiu, D. Towsley, and J. Krusoe, "Modeling TCP throughput: A simple model and its empirical validation," in *Proceedings of ACM SIGCOMM*, Vancouver, CA, 1998, pp. 303–314.

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

[15] A. Myers, P. A. Dinda, and H. Zhang, "Performance characteristics of mirror servers on the internet," in *Proceedings of IEEE INFOCOM*, 1999, pp. 304–312.

[16] S. Bhattacharjee, M. H. Ammar, E. W. Zegura, V. Shah, and Z. Fei, "Application-layer anycasting," in *Proceedings of IEEE INFOCOM*, 1997.

[17] S. Seshan, M. Stemm, and R. Katz, "SPAND: Shared passive network performance discovery," in *Proceedings of the USENIX Symposium on Internet Technologies and Systems*, 1997.

[18] R. L. Carter and M. Crovella, "Server selection using dynamic path characterization in wide-area networks," in *Proceedings of IEEE INFOCOM*, 1997.

[19] M. Sayal, Y. Breitbart, P. Scheuermann, and R. Vingralek, "Selection algorithms for replicated web servers," in *Proceedings of the Workshopon Internet Server Performance*, 1998.

[20] Z. Fei, S. Bhattacharjee, E. W. Zegura, and M. H. Ammar, "A novel server selection technique for improving the response time of a replicated service," in *Proceedings of IEEE INFOCOM*, 1998.

[21] K. Obraczka and F. Silva, "Network latency metrics for server proximity," in *Proceedings of IEEE Globecom*, Dec. 2000.

[22] K. Hanna, N. Natarajan, , and B. N. Levine, "Evaluation of a novel two-step server selection metric," in *Proceedings of IEEE ICNP*, Nov. 2001.

[23] J. Byers, M. Luby, and M. Mitzenmacher, "Accessing multiple mirror sites in parallel: Using tornado codes to speed up downloads," in *Proceedings of IEEE INFOCOM*, 1999.

[24] P. Rodriguez, A. Kirpal, and E. W. Biersack, "Parallel-access for mirror sites in the internet," in *Proceedings of IEEE INFOCOM*, 2000.