Technical Report
Number: NU-EECS-15-04

November, 2015

Towards Application-Level Multi-Homing

Yong Wang, Hao Wu, Weihua Zhang, Yanwei Li, Aleksandar Kuzmanovic

Abstract

Hundreds of millions of users are attracted to many of the thriving Internet applications and services, such as e-mail, online social networking, or video sharing. In an attempt to effectively serve the large number of users, the application providers started deploying their own application-specific network infrastructures. We demonstrate that the significant proliferation of applications and the growth of associated networking infrastructures creates a considerable application-induced diversity in end-to-end performance, measured in terms of the effective throughput. We conduct a large-scale measurement study in an attempt to quantify, understand, and utilize such application-driven network performance diversity. Moreover, we demonstrate that the observed end-to-end performance diversity can be effectively utilized in numerous networking contexts.

Keywords

Content providers; effective throughput; private networks; measurements; multi-homing;

Towards Application-Level Multi-Homing

ABSTRACT

Hundreds of millions of users are attracted to many of the thriving Internet applications and services, such as e-mail, online social networking, or video sharing. In an attempt to effectively serve the large number of users, the application providers started deploying their own application-specific network infrastructures, thus improving user-perceived performance. In this paper, we demonstrate that the significant proliferation of applications and the growth of associated networking infrastructures creates a considerable application-induced diversity in end-to-end performance, measured in terms of the effective throughput. We conduct a large-scale measurement study in an attempt to quantify, understand, and utilize such application-driven network performance diversity.

Our contributions are threefold. First, we extensively measure and analyze some of the most popular content and application providers including YouTube, Flickr, Hotmail, and Gmail. We characterize their underlying network infrastructures as well as their data replication mechanisms, and explain how the two properties impact the observed end-to-end performance diversity. Second, we find that application-specific network paths are stable, i.e., show little variability in effective throughput over longer timescales, which makes them attractive alternatives to the default Internet paths. Finally, we demonstrate that the observed end-to-end performance diversity can be effectively utilized in numerous networking contexts. The common primitive in all such scenarios is the one that helps the end users to opportunistically select one of their applications to transfer data (e.g., via FTP/TCP vs. as an attachment to a Gmail or Hotmail e-mail message, etc.) to a particular destination or a group of users.

1. INTRODUCTION

Popular applications and services are thriving in the Internet, attracting millions of users on a daily basis. Facebook alone attracts more than 800 million users, more than half of whom log on to Facebook in any given day [5]. YouTube attracts 800 million unique users each month, such that stunning 48 hours of video are uploaded every minute, resulting in nearly 8 years of content uploaded every day [8]. Email is among the popular applications as well. Hotmail serves 360 million unique users each month, followed by Yahoo! Mail and Gmail, which attract 300 million and 200 million users, respectively [4].

In an attempt to serve the large number of users and to effectively cope with the huge amount of traffic, the largest among these application providers (e.g., Google, Facebook, Microsoft) started building and deploying their own application-specific network infrastructures. These application networks typically consist of large-scale data centers interconnected by high capacity backbone links. In addition, such networks are also expanding dramatically towards end-users, i.e., by aggressively pursuing direct peerings with last-mile consumer networks [18]. As a result, more than 60% of Google's traffic flows directly between Google and consumer networks [18]. Facebook applies the same approach, such that more than 25% of its traffic runs through direct peering links with last-mile providers [19]. Microsoft also began building its own application network [7].

The growing application-specific networks necessarily created a *network-level* Internet path diversity. Indeed, one network path is selected if data is sent via FTP/TCP from a source to a destination, another path is selected if data is sent as an attachment to an e-mail provider, yet another path is selected if data is transferred via a third application network. Our key hypothesis is that the network-level path diversity creates an *end-to-end* performance diversity, measured in terms of the effective throughput. For example, transferring data via FTP/TCP is the optimal (fastest) approach for a given source-destination pair, e-mail enabled transfer is optimal for another source-destination pair, while transferring the same data via a third application network is optimal for a third pair of endpoints.

The main contributions of this paper are the following. First, we systematically quantify the application-induced end-to-end performance diversity in terms of effective throughput, i.e., time needed to transfer the same data via different application networks. Second, we analyze the application network's internal data transfer and replication mechanisms and explain the key underlying factors that cause the end-to-end performance diversity. Finally, we demonstrate that the application-driven end-to-end performance diversity can be effectively utilized in various networking contexts to help end users select the best among their applications to transfer data over. While in this paper we do *not* implement such an automatic application selector, we show that such a service can be feasibly deployed with negligible measurement overhead. This is because predicting the right application to use is straight-forward in the vast majority of scenarios due to high stability of the effective end-to-end throughput in application networks.

1.1 Our Findings

Our findings are the following:

First, we evaluate four application networks, i.e., Gmail, Hotmail, Youtube, and Flickr, and confirm that there exists significant end-to-end performance diversity. We show that application-specific paths can often and significantly outperform regular Internet paths. We find that 65.2% of Internet paths are improved by at least one of the explored application networks. Moreover, we find that the fully transparent e-mail application networks alone improve 44.3% of the paths.

Second, we provide a comprehensive analysis of the application-network transfer properties in an attempt to understand the root causes for the observed performance diversity. We decompose the end-to-end transfer times into the upload, replication, and download latencies. We analyze the key reasons that impact each of the three components in different application networks. We find that aggressive replication applied by Gmail and Youtube incurs the increased replication latency. Still, this approach pays off since it brings the content closer to end users who can download it at high rates.

Third, we explore the consistency of application-specific path properties. We find that that contrary to the Internet paths, the application-based paths show consistency in effective throughput, and consequently data transfer times. This is a very important feature because the observed performance diversity can be effectively utilized in various networking contexts. In particular, to help end users select the best among their applications to transfer data over, yet without incurring a significant measurement overhead. We empirically confirm this result by transferring $\sim 100\,$GBytes of data over various application-network paths and by analyzing the corresponding path properties.

Fourth, we explore how much a user can benefit by using multiple application networks. We find that using multiple applications is certainly beneficial, because it increases a chance to improve end-to-end performance. Still, we find that if a default Internet path is improved by one application network, it is likely improved by some other application network as well. In addition, we find that the achieved performance improvements in our experiments were strongly dominated by single application networks, *i.e.*, Gmail for the transparent transfers, Youtube for the video transfers, and Flickr for the photo transfers.

Fifth, we explore the properties of the application-network-enabled multicast. In such a scenario, data is pushed to an application network, then replicated towards multiple recipients, and finally downloaded by them. We show that this "native" multicast service dramatically outperforms the basic direct multicast transfer. We find that as the number of receivers increases, even the least effective among the application networks, Hotmail, manages to significantly outperform the direct transfer multicast performance.

Sixth, we evaluate application-network-hopping paths, *i.e.*, scenarios in which a path over two or more application networks shows better performance than over a single application network. While such paths are not particularly frequent, they reveal inefficiencies of the inner-application-network replication and routing. We demonstrate that such inefficiencies can be effectively resolved by application-network-based overlays in the same way the Internet overlays improve the default Internet path performance.

Seventh, we explore application-level multi-homed overlays that opportunistically combine and utilize the best Internet- and application-network paths. We show that the use of application-network paths can further improve the performance of the existing overlays, particularly for file transfers that otherwise experience the poorest performance, i.e., have the longest transfer times.

2. BACKGROUND AND MOTIVATION

2.1 The New Internet

The structure of the Internet has experienced a significant change in the last decade [20]. Indeed, a decade ago, the core of the Internet was largely dominated by the global transit backbone operators, *e.g.*, Level 3, Global Crossing, or AT&T, while customer IP networks were scattered at the edges of the Internet. The regional and local access providers — that connect the Internet core on one side, and the customer IP networks on the other side — were placed in between [20]. Consequently, clients and content were placed at the opposite edges of the Internet. Thus, requests from clients from one customer IP network would usually transit the upper networks to access the content hosted at servers in other customer networks.

A pioneering effort in bringing content closer to end-users has been conducted by Akamai [1] and other CDN providers. By deploying a large number of edge servers

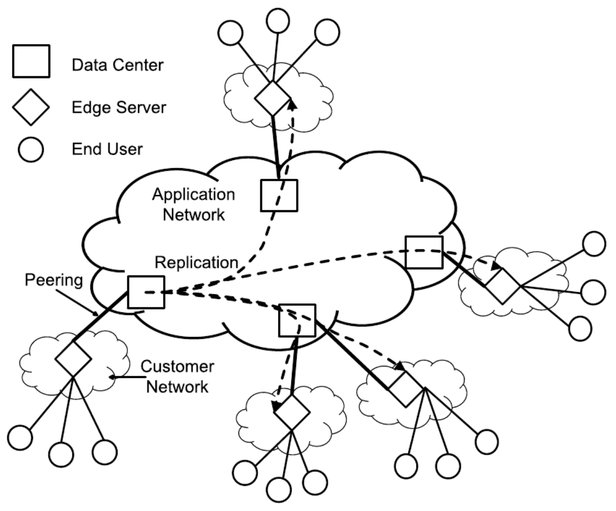

Figure 1: A “Hyper Giant” Morphing into a CDN

deep into customer ISPs, by further replicating content (*e.g.*, web or streaming) at these servers, and by effectively redirecting end users to close-by replicas, such systems manage to fundamentally improve user-perceived performance in both delay and throughput. Delay is improved because the corresponding content replicas are topologically closer to end users. Throughput is improved both because the corresponding round-trip times between clients and edge servers are decreased (hence the TCP throughput increases [22]), and because network bottlenecks near the origin servers or elsewhere in the Internet are effectively avoided.

A major shift relevant both for the Internet structure and content distribution started happening in the last 3 years [20]. In essence, this shift is caused by a significant growth of a small number of content providers, which started attracting majority of the clients. Indeed, according to a recent report [20], the top 150 content providers nowadays account for as much as 50% of the entire Internet traffic, while the top 30 content providers alone are now responsible for generating as much as 35% of the Internet traffic. Consequently, the structure of the Internet started changing because the largest among these providers (*e.g.*, Google, Facebook, Microsoft) started building their own application networks, by interconnecting their Internet data centers by high speed links [9]. In addition, these content ”hyper giants” [19] started to aggressively deploy their servers deep into customer ISPs, effectively applying Akamai’s CDN model [6]. As a result, the corresponding content networks started morphing into CDNs [7].

Figure 1 illustrates this phenomenon. A content provider’s application network consists of Internet data centers (four shown in the figure) that are interconnected by high-speed links, effectively creating an application network core. The content providers typically purchase unused fiber optic cable known as ”dark fiber” [9] to connect their data centers. In addition, the provider installs a number of edge servers in consumer networks (five shown in the figure), thus effectively expanding their application networks towards end users. An edge server is connected to a data center via a peering link. Currently, more than 60% of Google’s traffic flows directly between Google and consumer networks [18]. Facebook is also aggressively peering with a number of last-mile network providers [19].

Our key hypothesis is that these growing application networks effectively create a diversity in end-to-end data transfer performance. In particular, that the end-to-end performance of the application networks, used by millions of Internet users, can be better than that provided by the regular Internet paths. We argue here (and empirically show later in the paper) that this is indeed the case. The main arguments on behalf of this hypothesis are the following. First, breaking an end-to-end path into several shorter paths (*i.e.*, one from the sender to the application-network edge server, one or more paths within the application network, and finally one path from the receiving application-network edge server to the recipient) should increase the effective TCP throughput. As mentioned above, shorter RTTs increase TCP’s throughput [22]. Second, traffic that goes through the application networks can avoid potential bottlenecks on the regular Internet paths, hence improve the data transfer times. Finally, if the replication within the application network is sufficiently agile, then the application-network-based data transfer can indeed be beneficial.

In the following sections we conduct an extensive study in an attempt to confirm the above hypothesis. Our main goal is to understand the key architectural and design properties of different application networks that induce the application-driven end-to-end performance diversity. Then, we analyze different networking contexts in which the end users can opportunistically utilize the observed end-to-end diversity, i.e., by selecting the application that shows the best performance for a given destination or a group of users.

3. CHARTING APPLICATION NETWORKS

In this section, our goal is to collect the data about the physical structure of the application networks that we evaluate. In particular, we are interested in discovering the number and geographical locations of the corresponding edge servers or data centers that users get redirected to. This knowledge helps us to isolate the key factors that lead to application-driven end-to-end diversity that we evaluate in the next section.

3.1 Application Network Selection

We evaluate four application networks, associated with the following applications: Gmail, Hotmail, Youtube, and Flickr. The applications could be divided into two categories, (*i*) e-mail based application networks, *i.e.*, Gmail and Hotmail, and (*ii*) online social networking applications, *i.e.*, Youtube and Flickr. E-mail based application networks are transparent in the sense that they support all file types (that can be sent as attachments to e-mails). On the other side, online social networking application networks support a subset of file types. In particular, Youtube supports video file types, *e.g.*, `mp4` and `mov`, while Flickr supports popular photo formats, *e.g.*, `jpg` and `png`.

In addition to supporting selective file types, both Flickr and Youtube opportunistically apply compression. In particular, the resolution of the picture or video uploaded to the application network is typically larger than the resolution of the same picture or video downloaded from the network. Hence, the upload size is typically larger than the download size. Despite these effects, we selected the two application networks for the following reasons. First, because analyzing them provides important insights about upload, download, and replication performance, which have distinctive properties relative to e-mail application networks. Second, as we demonstrate later in the paper, in certain scenarios transparent e-mail application networks manage to outperform the YouTube and Flickr, despite the lack of compression. We analyze the underlying reasons that lead to such scenarios.

3.2 Discovering Edge Servers

To comprehensively characterize the underlying application networks, we strive for a platform that has a large geographical distribution. This helps to successfully emulate users' location diversity. We "create" such a platform by using a set of open recursive DNS servers. Open recursive DNS servers are public DNS servers in the Internet that provide DNS resolution service to any requester, without any source-based filtering. To obtain a list of such DNS servers, we retrieve the list of one million most popular Web sites from Alexa [2], and find their authoritative DNS servers. Next, we check if they are open recursive DNS servers. By using this approach, we successfully locate 13,816 open recursive DNS servers located in six continents and 130 countries.

3.2.1 Finding CNAMEs, Servers, and their Locations

In order to chart the application networks, we must first discover the CNAMEs associated with upload and download servers for each of the networks. To this end, we record the DNS traces while uploading content to and downloading from these application networks (Section 4), and extract the CNAMEs. In this way, we obtain one CNAME for Gmail download and upload, respectively; one CNAME for Hotmail download and upload, respectively; 256 CNAMEs for Youtube download and one CNAME for upload; and six CNAMEs for Flickr download and one for its upload.

Table 1: Geographical distribution of the application network servers

Continent	# of IPs							
	Gmail		Hotmail		Youtube		Flickr	
	D	U	D	U	D	U	D	U
N. America	17	17	4	1	1,917	15	12	1
Europe	6	6	0	0	2,591	8	0	1
Asia	0	0	0	0	1,186	4	0	0
S. America	0	0	0	0	154	1	0	0
Oceania	0	0	0	0	285	0	0	0
Unknown	0	0	2	1	99	0	0	0
Total	23	23	6	2	6,232	28	12	2

Next, we query these CNAMEs from the recursive DNS platform, and obtain the IPs corresponding to the upload and download servers or data centers, which we call *serving points*. Table 1 summarizes the number of IPs of both upload and download serving points for each application network. Starting with Google's services, Youtube has 6,232 download serving points and 28 upload points, while Gmail has both 23 download and upload serving points. Flickr has 12 points for download and 2 for upload. Hotmail has the smallest number of serving points, six for download and two for upload. We will show in Section 4 that the significant difference among application networks affects the data upload, download, and replication performance of each application network.

Finally, we attempt to geolocate the discovered serving points. Because we were unable to resolve these IPs using geolocation databases, we utilize a constraint-based geolocation approach [15] to discover the locations of IPs for each application network at the continent level. Table 1 shows the results. More than 90% of the download and the upload points for YouTube are located in North America, Europe, and Asia. In comparison with Gmail and Flickr whose servers are located in North America and Europe, Hotmail's servers only have presence in North America.

4. UNDERSTANDING APPLICATION-DRIVEN END-TO-END PERFORMANCE DIVERSITY

In this section, we first explore the overall performance of application-specific paths for the four application networks. Next, we decouple the transfer times into components, *i.e.*, upload, download, and replication to understand application-specific data transfer performance of each component. We find that the application networks' generic internal replication mechanisms and properties can fundamentally impact the end-to-end data transfer performance.

4.1 Preliminaries

4.1.1 Measurement Infrastructure

The platform of open recursive DNS servers is effective in charting application networks. However, it is technically impossible to measure the performance of application networks, *i.e.*, actively push and pull the data from it. Hence, we use 46 servers from PlanetLab, which are geographically distributed over 5 continents and 25 countries. We use this platform for measurements and evaluations in Sections 4 and 5.

4.1.2 Methodology

For each pair of PlanetLab nodes, we measure the total time needed to transfer data over the four application networks. The total transfer time consist of three components, (*i*) upload delay, *i.e.*, time needed to upload data to an application-network edge server, (*ii*) replication delay, *i.e.*, time needed by the application network to replicate the data towards designated edge servers, and (*iii*) download delay, *i.e.*, time needed by the receiver to download data from the designated edge server.

In the upload process, the sender uploads files to an application network, and specifies the name of recipients. In the e-mail application scenarios, the sender specifies a single or multiple recipients by providing appropriate e-mail addresses. On the contrary, in Youtube and Flickr scenarios, the sender simply uploads a chunk of data to the application network, without explicitly indicating the receivers. The time needed to upload the file to the application network is referred to as the upload time.

Once the data is uploaded, the sender immediately notifies the receiver by sending a message. In particular, the receiver first creates a server thread listening message from sender. Then the sender uploads data to application networks as explained above, and further creates a client thread to notify the receiver once data is uploaded.

The receiver records the time when the signal is received. Next, in the e-mail application network scenario, the receiver starts checking its mail box to see if the data arrived. Once it arrives, the receiver records the time. In the Youtube and Flickr application scenarios, the receiver starts checking sender's public space at these Web sites to see if there is any newly uploaded data by the sender. If the receiver observes new data, it records the time. Finally, we consider the difference between the time when the signal is received and the

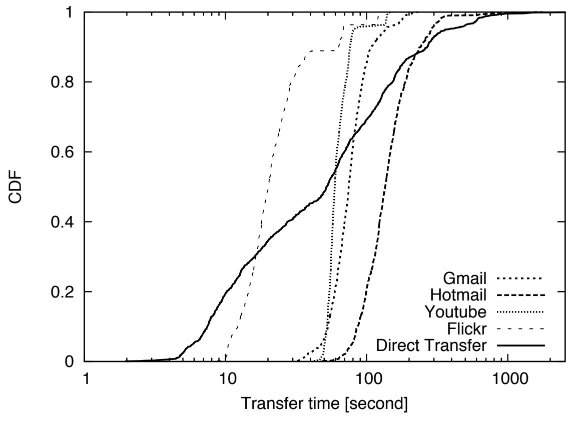

Figure 2: The overall performance comparison between application-network transfer and direct transfer

time when the uploaded data is visible to the receiver as the replication delay.

In the download process, the receiver downloads the requested data from either its mail box in the e-mail case, or sender's public space in the Youtube and Flickr case, and records the time when the download is finished. Finally, we consider the difference between the time when the data is visible to the receiver and the download finish time as the download time.

In this way, we are able to measure the total transfer time, and the time for its three components between any pair of PlanetLab nodes for each of the four application networks.

4.2 Overall Performance

Here, we evaluate the properties of the application-specific data transfers when different application networks are used. In all cases, we transfer different 10 MBytes files from a source to a destination to avoid any WAN acceleration technologies that may potentially be applied by the application networks. We then compare the application-specific data transfer results to those achieved when regular Internet paths are used. Finally, we analyze the key reasons standing behind the performance of each of the underlying application networks.

4.2.1 Per-Application-Network Performance

Figure 2 shows the cumulative distribution function (CDF) of the data transfer time for the four application networks as well as for regular Internet paths (denoted by "direct transfer" in the figure). The most important insight from the figure is that data transfers over application networks can *often*, *consistently*, and *significantly* outperform regular Internet paths. Indeed, whenever a curve for an application based system is above the one for the direct transfer, this implies better performance for the given application networks. We evaluate this issue in depth later in the paper. Below, we provide additional insights.

First, the variance of transfer times for the Internet paths is significantly higher than for application networks. Indeed, the Internet transfer times range across almost three orders of magnitude, *i.e.*, from several seconds to almost one thousand seconds. This is not a surprise given the highly distributed nature of our measurement platform and the known heterogeneous properties of Internet paths. At the same time, the data transfer performance for the application networks is much more smoother. For example, the transfer times for Youtube are almost deterministic. This is because the transfer times are determined by near deterministic replication and download delays, as we explain in detail later in the text. The transfer time variance for the other three application networks is more variable than Youtube's, yet significantly less variable than that of the direct transfer.

Next, Figure 2 shows that the data transfers over Flickr show the best performance. Indeed, about 80% of transfers are shorter than 30 seconds. This is counter-intuitive given the results from Section 3, which showed that Flicker's infrastructure is not particularly widely distributed. The reason for the superior performance is the file (picture) size compression, used by Flickr to reduce the file sizes to 10% of its original size. Figure 2 further shows that Youtube and Gmail have similar data transfer performance, *i.e.*, the median delays are around 65 seconds. This is despite the fact that Youtube also applies video files compression and typically reduces the file size to 50% of its original file size. Finally, from the two transparent application networks, Gmail outperforms Hotmail. This is because Gmail has a larger-scale infrastructure, and aggressively replicates data towards end users, as we show later.

Finally, we quantify the differences between the application network-based data transfers and the direct transfer. In particular, we compute the *per-path* statistics (not directly viewable from Figure 2, but viewable from Figure 9). It simply provides the percent of end-to-end paths for which a given application network-based transfer outperforms the direct transfer. For the transparent file-transfer systems, we find that Gmail outperforms direct transfers in 41% of cases, while Hotmail outperforms direct transfers in 19.4% of cases. For the remaining two application networks, Flickr's data transfer time is better than the direct transfer in 63.5% of scenarios, while Youtube surpasses direct transfer in 42.3% of cases. Below, we analyze the properties of the paths that application networks are both able and not able to improve.

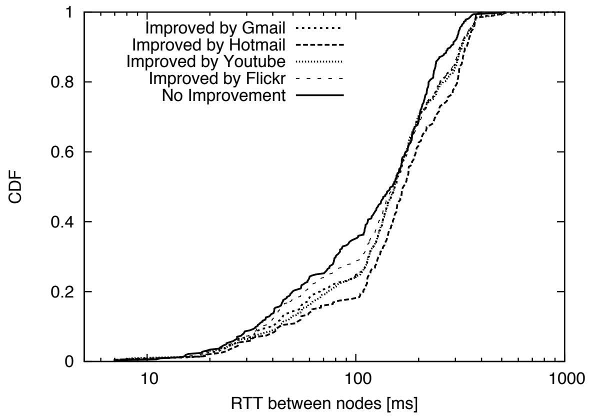

Figure 3: The role of RTTs

4.2.2 The Role of RTTs

Here, we explore the properties of paths, *i.e.*, the RTTs between the corresponding nodes, that application network-based data transfer can and cannot improve. In particular, Figure 3 shows the CDF of RTTs of the paths that are improved by particular application networks, as well as the CDF of RTTs for the paths (34.8% of all paths) that are not improved by any of the application networks. Our initial hypothesis was that the long Internet paths, which are known to have poorer performance, would be the ones that are more likely to be improved by application network systems. At the same time, we expected that paths with shorter RTTs would less likely be improved, because they in general provide better performance.

Figure 3 shows that our hypothesis is only partially true. In particular, the "no improvement" curve is shifted to the left, *i.e.*, towards shorter RTTs relative to the application network-based curves which are shifted towards longer RTTs. However, the figure also shows that RTTs are not the only factor that affects the performance. For example, the figure shows that more than 60% of paths that are *not* improved have RTTs longer than 100 ms (curve "no improvement", point (x,y) = (100ms, 0.38)). This shows that other factors, such as proximity of an endpoint to the corresponding application network edge server, also matters. If the edge server is not close-by, even longer RTT paths may not have better alternatives over application networks. This is further confirmed by the fact that application network-based paths are able to improve even short-RTT regular Internet paths. For example, Hotmail, an application network that improves the smallest number of Internet paths relative to other application networks, still can improve shorter RTT paths. In particular, 20% of paths improved by Hotmail have RTTs below 100 ms.

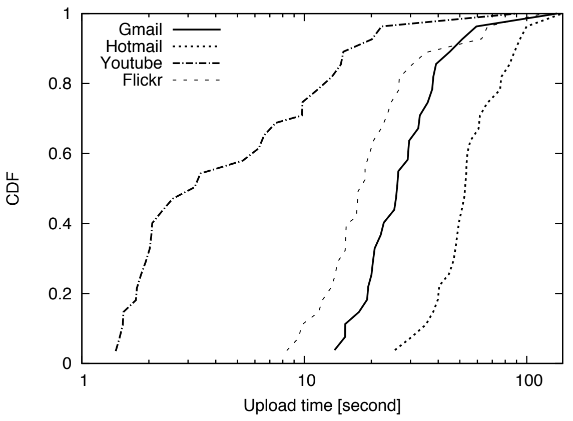

Figure 4: The upload performance comparison among application networks

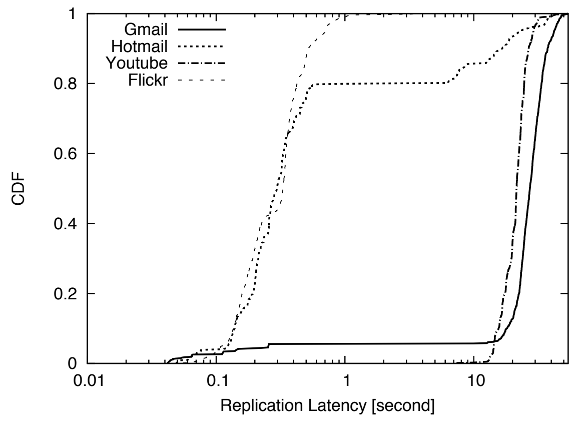

Figure 5: The replication latency comparison among application networks

4.3 Decomposing Application network-Based Data Transfer Delays

Here, we investigate the reasons standing behind different data-transfer performance for different application networks. To that end, we decompose the application network-based data transfer delays into three components: (*i*) upload delay, (*ii*) replication delay, and (*iii*) download delay.

4.3.1 Upload Performance

Figure 4 shows the CDF of upload times for the four application networks. The first insight is that Youtube has by far the shortest upload times. Indeed, approximately 50% of endpoints can upload the test file to Youtube in less than 3 seconds, *i.e.*, point (x,y) = (3, 0.5) in the figure. This is due to a significant presence of the Youtube edge servers discussed above. Indeed, providing fast file uploads is essential for a service such as Youtube. A similar feature is needed by the Flickr service. Still, because it is much less distributed than Youtube, the upload times are up to 8 times longer.

Regarding e-mail based systems, Gmail outperforms Hotmail by 2 times on average. While Gmail uses the same serving points for upload and download, Hotmail uses a smaller number of upload points, hence, the upload performance is the worst among the four application networks. In particular, the median upload time is around 55 seconds, which is approximately 17 times longer than Youtube's median.

4.3.2 Replication Latency

Figure 5 shows the CDF of the replication latency for the four application networks. Flickr shows the best performance among the four application networks. There are several reasons for this phenomenon. First, as explained above, Flickr compresses the file sizes by 10 times. Hence, replicating short files is necessarily faster than long ones. Moreover, given that all the replication times are below 1 second, and the median replication time is around 0.3 seconds, we conclude that the replication likely happens within the same server or data center.

Next, Hotmail shows a comparable performance to Flickr in 80% of cases. This again implies that the replication is likely happening within the same server or data center. Indeed, given that Hotmail provides a transparent data transfer service without any compression, the median transfer time of 0.3 seconds for a 10 MBytes file implies effective replication rate of 266 Mbps. It follows that the data likely does not leave a server or a data center, but is replicated within. At the same time, Figure 5 shows that the replication among edge servers does happen in 20% of cases for Hotmail. This corresponds to the ~10 seconds long tail at $y = 0.8$ for Hotmail. We find that when both the sender and the receiver are outside the US, the additional latency arises and generates the tail.

Figure 5 shows that Gmail and Youtube have similar replication delays. Still, Gmail shows a tail towards shorter replication times. In particular, in ~5% of cases, Gmail's replication is below 0.3 seconds, hence the data is likely replicated locally. Indeed, given that there are 23 Gmail locations, the probability that the sender and receiver are associated with the same Gmail server is around 5%. When that is not the case, the data is replicated towards the designated receiver, *i.e.*, the corresponding edge server. While this necessarily incurs additional replication latency (at the order of ~ 20 seconds), we will demonstrate later that such proactive replication brings data closer to end users and reduces the overall transfer times.

Youtube shows a more deterministic behavior, *i.e.*,

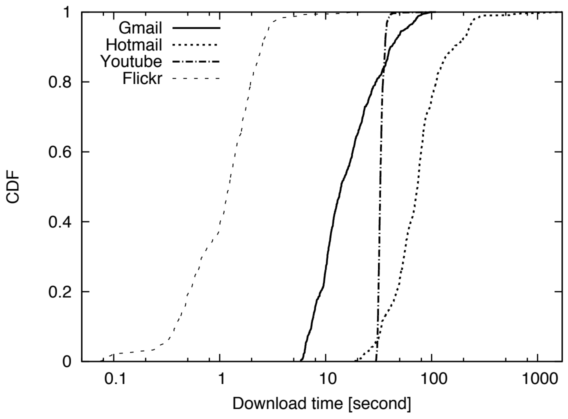

Figure 6: The download performance comparison among application networks

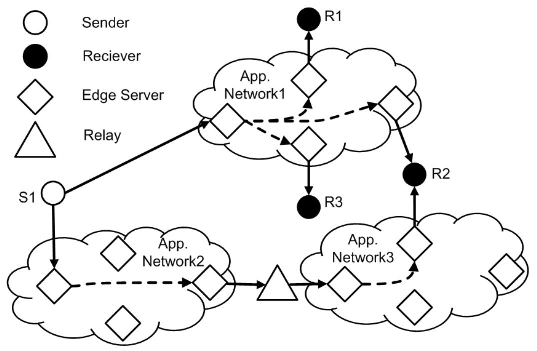

Figure 7: Application-Level Multi-Homing Data Transfer Scenarios

the corresponding CDF is rather steep, and does not have a tail as Gmail's CDF does. This happens due to obvious synchronization within the Youtube application network, *i.e.*, the uploaded file will become available to all edge servers at nearly the same time. On the contrary, the e-mail based application networks' receivers experience different replication and processing delays depending on their locations. Nonetheless, we will show later in Section 5.1 that the deviation of the transfer time for a *given* application network-based end-to-end path is very small. This has tremendously positive effects on application-level multi-homing system design and control, as we demonstrate later in the paper.

4.3.3 Download Performance

Figure 6 shows the CDF of download times for the four application networks. Given that Youtube and Flickr compress files, we expected that their download times would be much shorter than for the e-mail application networks that do not compress data but transfer it transparently. This is true in the Flickr case. It achieves superior download times, *i.e.*, the median download time is $\sim$ 1.5 second. This happens because Flickr aggressively compresses files (pictures) by a factor of ten.

Contrary to the Flickr case, we find that Youtube, despite the compression rate of two, does not achieve short download times. We find that Youtube rate-limits the download transfers by 1.44 Mbps. Hence, the almost vertical Youtube CDF curve in Figure 6 corresponds to the time needed to download 5 MBytes at the rate of 1.44 Mbps, *i.e.*. $\sim$ 27 seconds. Indeed, as an online video provider, Youtube does not need to provide the download rate that is much faster than the rate at which users can perceive videos in real time.

Gmail and Hotmail, the two application networks that provide transparent data transfers, show significantly different download performance. Indeed, as we demonstrated above, Gmail aggressively replicates data among edge servers, thus making the data closer to end users. In particular, the median download time is 14 seconds in Gmail's case, and 73 seconds in Hotmail's case, shown in Figure 6. Recall that Gmail typically spends $\sim$ 20 seconds for replication within the application network (Figure 5). Still, pushing data closer to end users pays off, because the median total transfer time is twice as shorter for Gmail than for Hotmail (Figure 2).

5. UTILIZING APPLICATION-DRIVEN END-TO-END PERFORMANCE DIVERSITY

Here, we explore several ways in which end-users can benefit from the end-to-end performance diversity shown above. Figure 7 illustrates several applications, that we explain later in the text. One of the applications is the one in which an endpoint (e.g., S1) opportunistically selects one of its underlying application networks (e.g., application network 1) to transfer data over. The key question for this and other related applications to be feasible in the first place, is that the end-to-end performance of the underlying application-network path is sufficiently stable, such that the selection process cab be simple and accurate. Below, we first analyze application-networks' path persistence properties, and then evaluate other networking contexts in which application networks can be utilized.

5.1 The Consistency of Application-Network Path Properties

Here, we explore the consistency of the application-network-based path properties. This issue is essential for application-level multi-homing (outlined above) as well as other application scenarios that we outline below, such as overlay networks. This is because low variability in performance implies smaller measurement

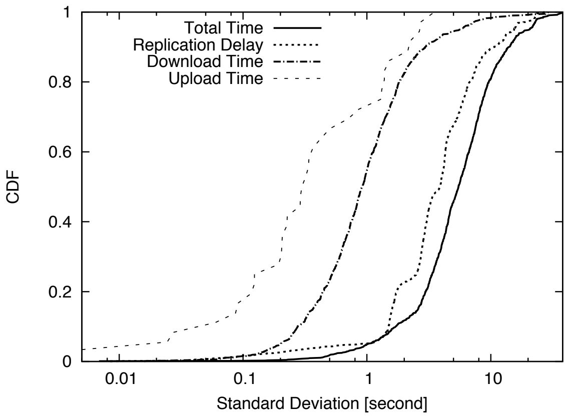

Figure 8: Standard deviation of the data-transfer delay components for the same node pairs over time

overhead. For the Internet paths, it is known that when predicting throughput, remembering observations from a number of minutes in the past is fine, but remembering for more than an hour can mislead the estimator [26]. Hence, Internet-based overlays have to evaluate underlying Internet paths often, which creates a significant overhead [11], hence scalability challenges.

Figure 8 plots the CDF of the standard deviation of data transfers and its components for the Youtube application network. In particular, we repeatedly transfer a 10 MByte video between all pairs of PlanetLab nodes via the Youtube application network twice in a day in a period of 8 days. We then compute the standard deviation over the given samples (16 for each path), and show the CDF of the standard deviation over the paths.

Figure 8 shows a negligible standard deviation for the upload and download times relative to the standard deviation for the replication times. Hence, the variance of the total time attributes to that of the replication delay. The relatively higher deviation of the replication delay is because it involves several processes: processing delay, *e.g.*, the compression of data as well as the replication over dispersed data centers. Still, the bottom line is that the standard deviation of the total transfer time is small overall, *i.e.*, much smaller than that shown in the Internet [26]. This is because each and every link of an end-to-end path over the public Internet can be affected by numerous issues and other applications that use these paths. On the contrary, the application network transfer effectively has three hops, where the first and the third hop that pass the public Internet are very short, while the middle one is handled by the application network. Hence, the "effective" end-to-end throughput is highly stable, hence predictable.

The implications of the above result are significant for the application-level multi-homing applications. Due to the small variance in the total transfer time, paths need to be re-evaluated over much longer time scales. Later in the paper, we verify this finding in an operational overlay scenario.

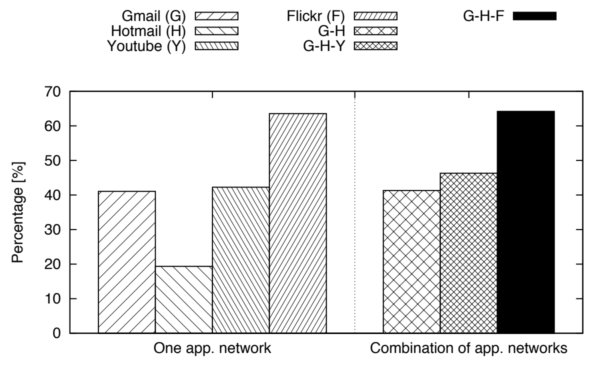

Figure 9: The percent of paths for which application network-based transfers outperform direct transfers

5.2 Utilizing Multiple Application Networks

Here, we explore if a combination of application networks could improve more Internet paths than when single application networks are used.

We explore three types of application network combinations. The first combination is the e-mail application network, consisting of Gmail and Hotmail application networks, which enable transparent (non-compressed) data transfers. The second combination is the video application network, consisting of Gmail, Hotmail, and Youtube. Because Youtube is used, this application network only supports the transmission of video file types. The third combination is the picture application network, consisting of Gmail, Hotmail, and Flickr. In this case, because Flickr is used, only picture file types are supported.

Figure 9 on the left side shows the performance for individual application networks, discussed earlier at the end of Section 4.2.1. The right side shows the performance of application network combinations. The figure shows that the Google-Hotmail application network improves 41.3% of paths, only 0.3% more than the Gmail application network itself. This is because most of the paths that can be improved by Hotmail, can also be improved by Gmail. Still, there are cases when a path is improved by Hotmail, but not Gmail. For example, the path `planetlab1lannion.elibel.tm.fr` (France) — `pl1.planetlab.uvic.ca` (Canada) achieves by 28 seconds faster transfer via Hotmail than via Gmail.

Next, Figure 9 shows that the Google-Hotmail-Youtube "video" application network improves by 4% more paths relative to the Youtube application network alone. An example scenario when the e-mail application network outperforms Youtube is the path `planetlab1.s3.kth.se` (Sweden) — `planetlab1.citadel.edu` (US), which achieves by 76 seconds faster transfer via Gmail than via Youtube.

Finally, the figure also shows that the Google-Hotmail-Flickr application network mostly improves the paths (64.2% of total paths) that are already improved by Flickr itself (63.5% of total paths). Despite significant compression applied by Flickr, there still exist picture transfers over Internet paths that Flickr does not improve, while Gmail does. For example, the path `planetlab1.csg.uzh.ch` (Switzerland) — `planet1.pnl.nitech.ac.jp` (Japan) achieves by 96 seconds faster transfer via Gmail than via Flickr.

5.3 Multicast

Here, we explore the properties of the application-network-based multicast. Indeed, as shown for application network 1 in Figure 7, application networks naturally provide a multicast service by concurrently replicating content to multiple edge servers. In particular, Figure 7 shows that the data from sender 1 is sent to an edge server in application network 1, and then replicated at the remaining three edge servers in the application network. Finally, the data is uploaded to the three recipients, R1, R2, and R3. This is possible to achieve in all investigated application networks. Youtube and Flickr support multicast by design. For the e-mail application networks, sending the same message (with an attached file) to multiple receivers triggers replication at the appropriate edge servers closest to anticipated receivers.

In our experiments, we use 46 PlanetLab nodes, each of which sends data to a subset of randomly chosen receivers in a multicast manner. In the application-network-based multicast scenario, a sender sends a single copy of a file to the application network. The application network replicates the data, which are then downloaded by the designated receivers. In the direct transfer multicast scenario, a sender sends a file simultaneously (using a separate TCP connection for each of the receivers) to the same group of receivers as in the application network case.

Figure 10 shows the median transfer times computed over all sender-receiver pairs as a function of the number of receivers. Given that there are 46 nodes, the maximum number of receivers is 45. When the number of receivers is 1, *i.e.*, in the unicast scenario shown previously in Figure 2, the performance is as previously explained. The median transfer time is the shortest for Flickr, than for the direct transfer, then for Youtube, Gmail, and Hotmail. As we increase the number of recipients, the performance for the application-network-based multicast does *not* change. This is because only a

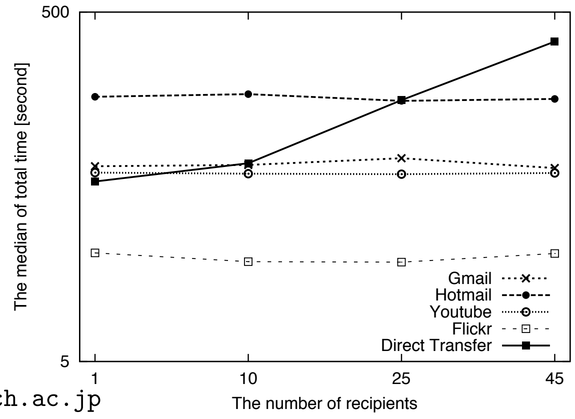

Figure 10: Multicast transfer times as a function of the number of receivers

single file is uploaded by a sender to the application network, and a single copy of a file is downloaded by each of the receivers from the corresponding edge servers.

On the contrary, in the direct transfer case, the number of connections that share the upload bandwidth increases with the number of receivers. Because the upload capacity of the sender is shared by multiple TCP connections, it becomes a bottleneck. Hence, the transfer times necessarily increase with the number of receivers. Figure 10 quantifies this effect. For 10 receivers, the direct transfer multicast that uses regular Internet paths is comparable to Gmail's multicast performance. Still, for 25 receivers, the direct transfer multicast performance falls far below Gmail and becomes comparable to Hotmail. For 45 receivers, the multicast direct transfer performance falls far below the Hotmail multicast service as well.

Much more effective direct transfer (non-application-network-based) multicast protocols than the basic one we applied above do exist, *e.g.*, [3, 12, 17]. We do not compare against these protocols for space constraints. Nonetheless, we argue that the "native" multicast primitive enabled by application networks is quite effective. Indeed, it requires the sender to send a single copy of a file once over the upload link. Also, it requires each of the receivers to download the file via the download link. We observe that these two requirements are the minimal possible requirements for any multicast system. Thus, contrary to application-level multicast systems [3, 12, 17], which require collaboration of the participating nodes and the use of their upload bandwidth, this is not the requirement for the application-network-based multicast. This makes the application-network-based multicast more attractive, and capable of achieving superior performance.

5.4 Overlays

Here, we explore application-level multi-homing in the context of overlay networks. As shown in Figure 7, data can be effectively transferred via multiple application networks, *i.e.*, in an overlay network scenario. As shown in the figure, data from S1 can be uploaded to application network 2, replicated towards a relay overlay node, uploaded to application network 3, replicated towards R2, and finally transferred to it. Such end-user-enabled use of relay nodes is common in various networking scenarios (*e.g.*, [3, 17, 23]). Later in the paper we explore if such an approach can, and in which scenarios, improve the performance. Intuitively, because different application networks have different presence in different parts of the world, hopping over multiple application networks appears as a viable approach. We evaluate this hypothesis below.

In particular, we first empirically verify the path consistency properties and explore the application-network-hopping path properties. Finally, we explore the benefits of application-level multi-homing approach in the scenarios where regular and application-network paths are combined.

5.4.1 *Application-Network-Based Overlays*

Here, we construct overlay networks on top of application networks. In an overlay network, nodes are PlanetLab nodes, and the underlying paths between overlay nodes traverse application networks. In particular, we have three types of application networks – transparent, video, and picture. For the case of transparently transferring any type of data from a source to a destination, we choose a path between the Gmail and the Hotmail application networks that shows better performance, *i.e.*, shorter transfer time. We assign this transfer time value as the weight of the edge between the pair of nodes over the given e-mail application network. Similarly, when transferring a video file, we choose the minimum value from Gmail, Hotmail, and Youtube, and assign the corresponding value as the weight of the edge. Finally, we apply the same approach for the "picture" application network which is a combination of Gmail, Hotmail, and Flickr.

For each of the above three overlay networks, we deploy a routing algorithm to compute the shortest path between any pair of nodes, *i.e.*, by minimizing the total time to transfer data among any two pairs. Certainly, multiple hop paths are supported. We use Dijkstra's algorithm to compute the paths [14]. Finally, we transfer the data over the paths computed by the algorithm.

Transfer Time Predictability Here, we empirically and systematically evaluate how the path consistency properties can be explored to predict transfer times. In particular, for each of the three networks,

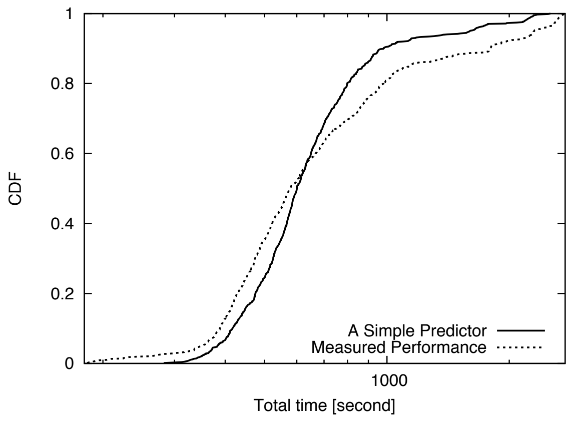

Figure 11: Achieved performance in theory VS in practice

and for any pair of nodes for which the performance is improved by application-network-based overlay over the regular Internet paths, we transfer the data as follows. For $N = 1, ..., 12$, we transfer $N * 10$ MBytes of data via the pre-determined application networks over the calculated paths. In the experiments, we send in total ~ 100 GBytes of data. We record the times taken by all such transfers, and evaluate whether a simple estimator can predict the transfer times. In particular, denote by T_{10M} the transfer time corresponding to transmitting 10 MBytes of data. We evaluate whether $T_N = N \times T_{10M}$ is an accurate predictor.

Figure 11 shows the results, confirming that the transfer times are predictable even with a very simple estimator. The results show that in 50% of cases, the difference between the predicted time and the real time is less than 12%. The key reason is the small deviation of transfer times shown in Section 5.1 above. Still, we observe that the prediction can be further improved. Indeed, for the longer file sizes for which the transfer last longer, the simple estimator underestimates the actual transfer times. This happens because the application-network-based overlay is a store-and-forward network, where *entire* files are first stored at ingress and egress application network edge servers, before being forwarded. Hence, the transfer times do not linearly increase with the file sizes, as assumed by our predictor.

Application-Network Hopping. Here, we focus on a subset of application-network-based overlay paths that travel more than a single application network from a source to a destination. Indeed, the shortest path algorithm explained above aims to optimize the end-to-end transfer times, which sometimes means using two or more application networks, as illustrated for a path traveling application networks 2 and 3 in Figure 7. We explore this phenomenon below.

We find that among the paths that improve regular Internet paths, the percent of application network-hopping paths, *i.e.*, those that travel over at least two application networks, is 3%. The reasons for the small percent are the following. First, regular Internet paths are already significantly improved by single application network paths, as we demonstrated above. Second, application network transfers are store-and-forward in nature, and the data transfer unit is the entire file. Before getting forwarded, a file is first stored at the ingress edge server, and then at the egress one. Repeating such processes in multiple application networks necessarily increases the end-to-end transfer delays. Nonetheless, there are cases when using two or three application networks between a source and a destination is beneficial.

We find that within the application network hopping paths, 94.5% hop over two application networks, while the remaining 5.5% of paths hop over three application networks. A similar result has been shown for regular Internet paths [11]. We evaluate an example three-application network path next in the "picture" category. In particular, when a 10 MByte picture needs to be sent from `planetlab-n1.wand.net.nz` to `planetlab1.cs.umass.edu`, the best direct application network path is the one via Flickr. The original file is transmitted in its full size to a Flickr server in US, and then a 1 MByte compressed version is downloaded by the receiver at Umass. The whole transfer takes 68.2 seconds. However, if the same file is sent by Gmail from New Zealand to `planetlab1.arizonagigapop.net`, then transferred via Flickr to `planetlab1.cs.wayne.edu`, and then transferred to `planetlab1.cs.umass.edu` via Flickr again, the received file size is 1 MB again yet the transfer lasts 57.7 seconds.

There are two reasons for the improved transfer time. First, sending the 10 MB file from New Zealand via Gmail's application network to a well-connected PlanetLab node improves the transfer time on this long-RTT path. Moreover, the result further shows that inner-application network routing is not always effective. In this particular case, hopping through an external node and then going back to the Flickr application network is better than downloading a file from a suboptimal edge server. Indeed, application networks can optimize their internal routing decisions based on their local policies. Moreover, they may be unaware of the properties of external Internet paths and their upload and download characteristics. Because the application-network-based overlay has a global view, it is able to more effectively route data over application networks.

5.4.2 *A Hybrid Overlay*

Here, we do not constrain ourselves to application network-based overlays, but rather explore hybrid, application network-supported overlays, that can combine the best Internet - and application network-based paths. In particular, it has been shown [11, 24] that taking an alternate Internet path can be beneficial. Thus, we explore a scenario in which an existing overlay, *e.g.*, RON [11], would combine Internet and application network-based paths, and opportunistically choose the best direct or indirect paths to improve the performance. To avoid any issues with compressed files, we only use Gmail and Hotmail application network paths, which transfer data transparently. Then, we compute the shortest paths among all node pairs.

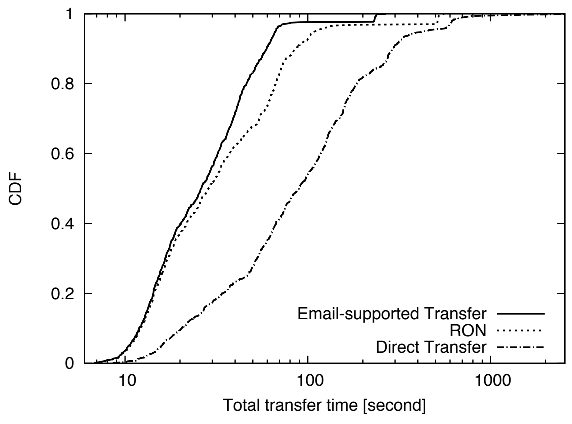

Figure 12: Email-supported transfer VS RON

Figure 12 shows the results. The "Direct Transfer" curve shows the performance of default Internet paths. Next, the curve "RON" shows that alternative Internet paths are capable of improving the default Internet performance. Finally, the curve "E-mail-supported Transfer" demonstrates that using application network-based paths (e-mail in this case), brings additional benefits. In particular, the figure shows that the transfer times are improved by approximately 35% for the 30% of files that experience the longest transfers relative to RON. This corresponds to the gap shown in Figure 12 between RON and e-mail-supported transfer for $y \in (0.7, 1)$. Given that application network-based paths are much more stable and provide predictable performance, hence require small overhead, they could be very attractive for existing Internet overlays.

6. DISCUSSION AND RELATED WORK

6.1 Discussion

Uneven traffic distribution? One might think that the application-level multi-homing approach can lead to uneven traffic distribution in the sense that some application providers (e.g., Gmail, which shows the best performance in our study) will "pick up" most of the traffic from users that apply this approach. There are several issues here. First, attracting users is one of the main goals of the application providers. Hence, those application providers that manage to provide better networking services should certainly attract more user traffic, which is the measure of application networks' success. Second, in this paper we evaluated 4 representative application providers and their network infrastructures. Because the number of such providers is much larger, it should be expected that the traffic distribution will be more even. Third, different application providers have different presence in different regions, which should further help more evenly distribute the traffic. Finally, a lot of traffic micro-level "load balancing" is naturally happening due to inherent randomness in the Internet, i.e., due to bottlenecks and round-trip time variability, which are the factors that help different applications to achieve different performance among different endpoints.

6.2 Related Work

Akella *et al.* [10] showed that peering with multiple ISPs is beneficial for reliability and performance reasons. In particular, they demonstrated that cleverly scheduling traffic across the ISPs can improve Internet RTTs and throughputs by up to 25% and 20% respectively. In our work, we explore a similar, yet fundamentally different concept in which an end user can opportunistically utilize one of its applications to send data. Because the throughput performance in our case is very stable over longer time scales, the application selection process does not have to be particularly clever in order to be successful.

Savage *et al.* [24] demonstrated that taking an alternate Internet path, and not a default one, can be quite beneficial in terms of delay, packet loss, and bandwidth. To an extent, our work here is a modern "detour" version that reuses content application networks by utilizing the corresponding application protocols. Indeed, we demonstrate that high-bandwidth application-network-based paths that exhibit superior quality and incur much smaller measurement overhead relative to the Internet paths are feasible.

Anderson *et al.* [11] designed a resilient overlay network (RON), which improves the end-to-end path characteristics by opportunistically selecting Internet paths with optimal performance. In particular, RON monitors the functioning and quality of Internet paths among its nodes, and selects the optimal route between any pair of nodes. Thus, an application-level multi-homed overlay and RON share the same goals. Still, contrary to RON, the application-level multi-homed overlay works *on top* of several users' application networks and leverages the fast replication properties of content application networks to transfer data, while RON utilizes the regular Internet infrastructure. In Section 5.4.2, we demonstrated synergistic potentials between the two approaches.

Application networks provide an effective multicast service. Hence, it relates to a huge body of work dedicated towards designing and deploying such a service. Our overview here is necessarily not comprehensive – we simply select two representative designs. Deering *et al.* [13] proposed IP multicast where packet replication happens at the router level. Because such a feature has not for a long time been actually enabled on the Internet, a natural alternative was an end-point multicast system, *e.g.*, [12]. In such a system end-points themselves replicate data without requiring any underlying network support. Our work relates to both approaches. On one hand, we reuse a "native" multicast support enabled by content application networks such as Youtube, which replicates content to a number of edge servers. At the same, by using this "native" underlying multicast service, we further improve it by providing an application-level multicast service where application-network-based paths are also supported.

Our work to some extent relates to the "split TCP" idea applied in mobile ad hoc networks [16] or satellite networks [21] to improve end-to-end performance. In particular, it is known that TCP's fairness and throughput suffer when it is used in such networks. This is a direct consequence of long RTTs or TCP wrongly attributing packet losses due to link failures to congestion. Hence, by introducing proxies the split TCP emulates shorter TCP connections and achieves higher throughput. One issue with this approach is that it does not retain the end-to-end semantics, hence creates problems to applications [25]. Application-level multi-homing does not experience such issues because it operates transparently at the application layer. More fundamentally, by routing traffic through the content application networks, application-level multi-homing is capable of effectively avoiding the Internet bottlenecks, thus achieving better performance. Moreover, contrary to split TCP approaches, a generic application network-based transfer does not require any dedicated proxy infrastructure, but rather reuses existing application network systems.

7. CONCLUSIONS

In this paper, we demonstrated that the significant proliferation of applications and the growth of associated networking infrastructures created a significant application-level diversity in end-to-end performance, measured in terms of the effective throughput. We conducted a large-scale measurement study in an attempt to quantify, understand, and utilize such application-driven network performance diversity.

Our findings are the following: (*i*) application-network-based paths can often and significantly outperform regular Internet paths, *i.e.*, we find that 65.2% of Internet paths are improved by at least one of the four application networks. (*ii*) The fully transparent e-mail application networks manage to improve 44.3% of the paths. (*iii*) Longer RTT Internet paths are more likely to be improved, but the key factors determining the performance of application-network-based data transfers is the proximity of the endpoints to the application network edge servers and an application network's replication agility. (*iv*) Using multiple application networks can improve the performance, even though the achieved performance improvements in our experiments were strongly dominated by single application networks, *i.e.*, Gmail for the transparent transfers, Youtube for the video transfers, and Flickr for the photo transfers. (*v*) Application-network-based multicast can significantly outperform the direct transfer multicast. With the increase in the number of receivers, even the least effective among the four application networks, Hotmail, manages to significantly outperform the direct transfer multicast performance.

We have further explored application-level multi-homing in various networking scenarios. We have found that (*i*) contrary to Internet paths, application-network-based paths show significant consistency in effective throughput. This is because the upload, replication, and download latencies show very small variability. (*ii*) This can significantly simplify the application-network-based overlay control and design, and reduce the measurement overhead. We have empirically demonstrated the high predictability of the data transfer times. (*iii*) Application-network hopping paths rarely outperform single application network paths. This is because single application networks already perform well. (*iv*) Application-network-based overlays are capable of improving inefficiencies of the inner-application network replication and routing decisions in the same way the Internet overlays improve the default Internet performance. (*v*) Hybrid overlays can improve the performance of regular overlays. For example, we have demonstrated that an e-mail-supported overlay can help improve RON's performance.

All the above properties demonstrate that application-level multi-homing is a viable approach that can help end-users improve their end-to-end performance. We plan to deploy a multi-homing application-selector plugin and make it publicly available.

8. REFERENCES

[1] Akamai. `http://www.akamai.com/`.

[2] Alexa. `http://www.alexa.com/`.

[3] BitTorrent. `http://www.bittorrent.com/`.

[4] Email and webmail statistics. `http://www.email-marketing-reports.com/metrics/email-statistics.htm`.

[5] Facebook statistics. `http://www.facebook.com/press/info.php?statistics`.

[6] Google Peering Policy. `http://lacnic.net/documentos/lacnicxi/presentaciones/Google-LACNIC-final-short.pdf`.

[7] Is Google's Network Morphing Into a CDN? `http://www.datacenterknowledge.com/archives/2010/03/18/google-boosts-peering-to-save-on-bandwidth/`.

[8] YouTube statistics. `http://www.youtube.com/t/press_statistics`.

[9] YouTube's Bandwidth Bill Is Zero. Welcome to the New Net. `http://www.wired.com/epicenter/2009/10/youtube-bandwidth/`.

[10] A. Akella, B. Maggs, S. Seshan, and A. Shaikh. On the performance benefits of multihoming route control. *IEEE/ACM Transactions on Networking*, 2006.

[11] D. Anderson, H. Balakrishnan, M. Kaashoek, and R. Morris. Resilient overlay networks. In *ACM SOSP '01*.

[12] Y. Chu, S. Rao, and H. Zhang. A case for end system multicast. In *ACM SIGMETRICS '00*.

[13] S. Deering, D. Estin, D. Farinacci, V. Jacobson, C. Liu, and L. Wei. An architecture for wide-area multicast routing. In *ACM SIGCOMM '94*.

[14] E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1959.

[15] B. Gueye, A. Ziviani, M. Crovella, and S. Fdida. Constraint-based geolocation of internet hosts. *IEEE/ACM Transactions on Networking*, 2006.

[16] S. Kopparty, S. Krishnamurthy, M. Faloutsos, and S. Tripathi. Split TCP for mobile ad hoc networks. In *GLOBECOM '06*.

[17] D. Kostic, A. Rodriguez, J. Albrecht, and A. Vahdat. Bullet: high bandwidth data dissemination using an overlay mesh. In *ACM SOSP '03*.

[18] C. Labovitz. How Big is Google? `http://asert.arbornetworks.com/2010/03/how-big-is-google/`.

[19] C. Labovitz. The Battle of the Hyper Giants (Part I). `http://asert.arbornetworks.com/2010/04/the-battle-of-the-hyper-giants-part-i-2`.

[20] C. Labovitz, S. Iekel-Johnosn, D. McPherson, J. Oberheide, F. Jahanian, and M. Karir. ATLAS Internet Observatory 2009 Annual Report. `http://www.nanog.org/meetings/nanog47/presentations/Monday/Labovitz_ObserveReport_N47_Mon.pdf`.

[21] M. Luglio, M. Sanadidi, M. Gerla, and J. Stepanek. On-board satellite "split TCP" proxy. *IEEE Journal on Selected Areas in Communications*.

[22] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose. Modeling TCP throughput: A simple model and its empirical validation. In *ACM SIGCOMM '98*.

[23] M. Piatek, T. Isdal, T. Anderson, A. Krishnamurthy, and A. Venkataramani. Do incentives build robustness in BitTorrent? In *NSDI, '07*.

[24] S. Savage, A. Collins, E. Hoffman, J. Snell, and T. Anderson. The end-to-end effects of Internet path selection. In *ACM SIGCOMM '99*.

[25] F. Xie, N. Jiang, Y. Ho, and K. Hua. Semi-split TCP: Maintaining end-to-end semantics for split TCP. In *LCN '07*.

[26] Y. Zhang, N. Duffield, V. Paxson, and S. Shenker. On the consistency of Internet path properties. In *IMW '01*.