An Efficient Approach for Content Delivery in Overlay Networks

Mohammad Malli, Chadi Barakat, Walid Dabbous
Projet Planète, INRIA-Sophia Antipolis, France
E-mail:{mmalli, cbarakat, dabbous}@sophia.inria.fr

***Abstract*— The replication of digital content in overlay networks makes the identification of the best server an interesting problem. In this paper, our aim is to reduce the content transfer time, which we define as the time required to download a digital content by a client using TCP. Our scheme consists of ranking the servers from the best one to the worst one based on a metric that corresponds to a prediction of the content transfer time from each server. Our prediction function considers the critical performance parameters that have an impact on the quality of the transfer, such as the load of the servers and the characteristics of the path between the client and the servers. Once the servers are ranked, the client can download the content on point-to-point from the best server, or in parallel from a subset of servers at the top of the list (best servers). Our experimental results show that our approach for identifying the best server(s) outperforms the existing classical solutions. Moreover, these results show that the proposed metric predicts the transfer time of a content with around 96% accuracy.**

I. Introduction and Motivation

Service replication is a scalable solution for the distribution of digital content over the Internet. The need for this replication is caused by the increasing number of Internet users and by the desire to improve the QoS. Also, it is important for achieving a high availability of data. Many overlay networks are proposed and installed to realize this replication : (i) Content Distributed Networks (CDN), where client requests are forwarded by request redirectors, and where the contents are stored in mirror servers geographically distributed over the Internet. Many companies, like Akamai [1], provide CDNs to content providers. (ii) Peer-to-peer networks (e.g., Kazaa [2]), where peers behave as clients and servers.

In the following discussion, we consider a *server* as being either a server among a set of replicated servers in a CDN or a peer in a peer-to-peer network. The *best server* is the one which is able to provide the requested service to the client with a better QoS than all other servers. The *central server* is the resolver (or the redirector) in the overlay network. It receives requests from clients and provides them back the address(es) of the best server(s). Also, we mean by *client* a standard client in the client/server paradigm, or a peer that requests a content in a peer-to-peer network. Clearly, the best server varies from one client to another based on the position of the client and the state of networks and servers.

Our aim is to minimize the time required to serve a client. We consider in this work a service that consists of clients downloading files from a set of replicated servers using the TCP protocol and where the QoS provided to clients is maximized if the transfer time was minimized. We call this transfer time the *latency*.

We propose an efficient scheme for ranking the set of servers that are candidates to serve a client from the best one to the worst one. Servers are ranked based on a metric that corresponds to a prediction of the transfer time of a content from each server to the client. The originality of this metric is in the fact that it considers the characteristics of the paths between servers and client together with the server's load. Once the ordered list of servers is obtained, the client can download the content from the best server or in parallel from a certain number of best servers.

Choosing the best server amounts to downloading the file from the server that is able to provide the minimum transfer time. This improves the QoS provided to clients and avoids network and server congestion by distributing the load over servers and network paths that are less loaded than others. Before explaining our scheme, we first present a brief overview of the solutions proposed in the literature for server selection: (1) Using the DNS (Domain Name System) to get the IP address of the best server. This widely used technique is simple: the DNS servers distribute the IP addresses of multiple servers associated to a unique name with a round robin algorithm. It is clear that this solution is not designed to improve the QoS since it does not consider any static or dynamic performance limitations. It only ensures basic load balancing. (2) Offering the client a list of servers and let him choose manually the best server to contact. The client choice in this case is based on his own criteria, for example the geographical proximity. (3) Using more sophisticated techniques that take into account one or many parameters having an impact on the content transfer time. Next, we describe some of these techniques.

The binning solution in [4] requires that the client and servers determine their own bins by measuring their RTT (Round-Trip Time) to a set of landmark points. By knowing the bins of the client and servers, the DNS server can classify the servers (from the best one to the worst one) based on the distance between their bins and the client's one. With this solution, the client does not measure the performance of the paths that connect it to the servers but rather of those that connect it to the landmark points. The drawback of this approach can be observed in the case where a client and a server have a short delay path to the landmark points while not having such a short delay path between them. Moreover, having a short delay may not be enough to obtain

a good transfer time, since other parameters can impact the performance as well, as the available bandwidth and the server load.

[6] uses a technique which combines server push and client probe approaches. The server push consists in the server sending to clients or to DNS servers his current performance information every time there is a considerable change in this performance. The client probe consists in the client measuring the path between it and each server as for example the measurement of the available bandwidth. The main problem of this solution is the fact that it requires installation of proxies that act as probing agents and that characterize the paths between clients and servers. To be efficient, a proxy needs to be installed for each nearby set of clients. The proxy estimates the transfer time from a server and considers this estimate as equal to what the clients located behind it would obtain. But, the proxy estimate of the transfer time does not necessarily correspond to the actual transfer time to the client, even though the proxy and the client can be nearby located. A mismatch can appear in cases when a bottleneck link, which is highly congested or has a high packet loss rate (e.g., a wireless link), exists on the server - client path while it does not exist on the server - proxy path.

This paper is organized as follows. The next section describes our scheme: its goals and the communication protocol. Then, we present our prediction function, the evaluation of this function, and a discussion on the prediction cost in Section III. Finally, the conclusion is presented in Section IV.

II. Communication Protocol

A. Goals

Our scheme enhances the transfer time by providing to the client the address(es) of the best server(s) after ranking the replicated servers from the best one to the worst one based on transfer time prediction. The addresses are provided to the client via the central server and the transfer time prediction is done by each replicated server, so the service is completely transparent to clients. The scheme avoids the congestion in the network and contributes to traffic balancing since it takes into account the available bandwidth on the paths. It also avoids overloading the servers since it takes into account the load of the servers in the selection process.

B. Scheme Description

Our scheme consists of a set of agents located in the central server and the replicated servers. (i) The central server holds the *service relay agent* which is a well-known application-layer gateway, and the *classifier agent* which ranks the servers to provide back to the client the IP address(es) of the best one(s). (ii) Each replicated server holds: a *load-estimator* agent which calculates continuously the request arrival rate and service rate in the server, a *probing agent* which probes the client and predicts the time needed to transfer the requested content to the client, and a *service application* which provides the client with the requested content if the server has been selected by the central server as being the best one.

The following are the basic steps realized before establishing the connection between the client and the best server for content download (see Figure 1):
(1) The client sends its request to the service relay agent in the central server. (2) The service relay agent calls the classifier agent and gives it the client's IP address and client receiving buffer space (obtained from the window size advertised by the client in the TCP header). (3) The classifier agent sends the client's IP address and client receiving buffer space to the probing agents located in a certain set of replicated servers. For example, to serve a French client, the classifier agent sends the client information to the probing agents located in the replicated French servers. (4) Each probing agent in a replicated server probes the client in order to obtain the performance of the path between them, and reads the current server performance parameters determined by the load estimator agent. (5) Each probing agent computes the predicted transfer time metric defined in Section III-A. (6) Each probing agent sends the obtained value of the predicted transfer time to the classifier agent in the central server. (7) The classifier agent compares the predicted transfer time values received from the different probing agents and sorts these values in an increasing order. (8) The classifier agent sends to the client the IP address(es) of the best server(s) (which provide the smallest transfer time value). (9) A TCP connection is established between the client and the best server. In case a set of IP addresses is provided as corresponding to the best servers, the client can open a TCP connection with each of these servers and download the content in parallel. The main

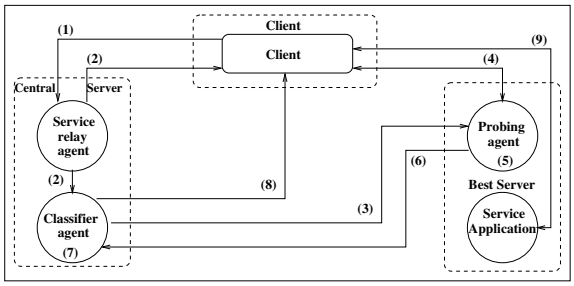

Fig. 1. Communication architecture

reason for which we propose to implement the probing agent on the server side is to make the service transparent to clients. Another reason is that the probing agent can easily read the size S of the requested content and estimate the load of the server without sending to the network any message asking for the values of these parameters.

The probing agent predicts accurately the content transfer time by considering: (i) the characteristics of the path to the client: the available bandwidth denoted by A, the round-trip time denoted by RTT, and the packet loss rate denoted by P. (ii) The performance limitations of the server: the maximum congestion window value W_{max} that can be buffered for transmission, and the idle time lost due to the buffering of the requests in the server (time between the arrival of a request and the establishment of the corresponding TCP

connection). This idle time depends on the server load. (iii) The performance limitation of the client, which is represented by its receiving buffer space communicated to the service relay agent by the advertised window field in the TCP header and then communicated to the server by the classifier agent.

III. Transfer Time Prediction

A. Prediction Function

To predict the content transfer time from a server to a client, we consider that: (i) the server uses TCP New Reno for content download. (ii) The server has an initial congestion window W_1 equal to 1 packet and its congestion window during the i-th round trip time W_i is limited by the value W_{max} which is imposed by server or client buffer limitations. (iii) The client sends an acknowledgment (ACK) for every b data segments received from the server. The value of b is usually equal to 2 due to the Delayed ACK functionality in TCP. (iv) The channel drops packets independently of each other with a constant probability P. Thus, the average number of packets successfully transmitted between the beginning of the transfer and the first packet loss in this case is $\frac{1}{P}$.

The parameters and functions used in our latency prediction function are expressed in the following units: (i) W_i, W_{max}, d, and $E[W_{ss}]$ are in packets of size m bytes. (ii) $E[T_s]$, $E[L_{ss}]$, $E[L_{ca}]$, RTT, and PTT are in seconds, (iii) m, $E[S_{ss}]$, and S are expressed in bytes, and finally (iv) A, R_{max}, and R_{ca} are in bytes per second.

To estimate the transfer time of a content of size S, we propose the following function denoted by PTT (Predicted Transfer Time):

$$PTT = E[T_s] + E[L_{ss}] + E[L_{ca}]. \tag{1}$$

$E[T_s]$ is the mean request waiting time, i.e., the average time that a request spends in the socket's arrival queue of a server before it is handled by a thread. $E[L_{ss}]$ is the transfer time spent during the slow start phase at the beginning of the download, and $E[L_{ca}]$ is the transfer time spent during the congestion avoidance phase.

We model the socket's arrival queue of a server and its associated threads (of number c) as an M/M/c queue, where λ is the mean request arrival rate, and μ is the mean service rate. Using known results from queuing theory [8], we can write:

$$E[T_s] = \frac{(\frac{\lambda}{\mu})^{\sqrt{2\cdot(c+1)}} - 1}{c \cdot (\mu - \lambda)}. \tag{2}$$

Basically, λ is calculated in the kernel of the server platform by marking permanently in a certain file the time when a SYN packet arrives to the socket's arrival buffer. μ is calculated by marking permanently in a certain file the time when a thread begins to serve a request (queued in the socket's arrival buffer) and the time when it finishes serving this request (the TCP connection state is created in the server and the ACK is sent back to the client). Then, λ and μ are updated periodically by the load estimator agent and stored in a file that the probing agent can read when necessary.

We use γ as the rate of exponential growth of TCP congestion window W_i during the slow start phase:

$$W_{i+1} = W_i + \frac{W_i}{b} = \left(1 + \frac{1}{b}\right) \cdot W_i = \gamma \cdot W_i. \tag{3}$$

The end of the slow start phase can be caused by the occurrence of a packet loss along the path in one of three cases (if the content size allows): (i) the bandwidth is saturated on the path server - client (client sending rate reaches A). This case can only happen if the product available bandwidth-delay is less than the maximum window value ($A.RTT < W_{max} \cdot m$). (ii) The congestion window value of the slow start phase reaches the buffering capacity W_{max} (client sending rate reaches $W_{max} \cdot \frac{m}{RTT}$), then after a certain time, a packet is lost on the path server - client due to the random loss process of rate P we are assuming (in opposite to case (i), the available bandwidth A is not reached here). (iii) Before reaching the buffering capacity or the available bandwidth on the path server - client, a packet is lost on the path after an average number of packets equal to $\frac{1}{P}$ has been successfully transmitted to the client. We note W_P the client window size reached at the end of the slow start phase in case (iii):

$$\sum_{i=0}^{log_\gamma W_P} \gamma^i = \frac{1}{P}, \tag{4}$$

so:

$$W_P = \frac{1}{\gamma} \cdot \left(\frac{\gamma - 1}{P} + 1\right). \tag{5}$$

The maximum sending rate that can be reached at the end of the slow start phase can be expressed by taking the minimum over the last three mentioned cases:

$$R_{max} = min\left(A, W_{max} \cdot \frac{m}{RTT}, \frac{1}{\gamma} \cdot (\frac{\gamma - 1}{P} + 1) \cdot \frac{m}{RTT}\right). \tag{6}$$

Small size contents can be completely transferred during the slow start phase. When the content size is large, it starts being transmitted in the slow start phase, then continues its transmission in the congestion avoidance phase. In the next two sections we investigate these two cases.

1) Transfer completed in the slow start phase: In this case, we consider that r_S+1 round-trips are required to complete the download. The download ends before TCP transmission rate reaches R_{max}. The latency components have the following expressions:

$$E[L_{ss}] = (r_S + 1) \cdot RTT, \text{ and } E[L_{ca}] = 0 \quad \text{if } \gamma^{r_S} \cdot \frac{m}{RTT} \leq R_{max}, \tag{7}$$

where,

$$\sum_{i=0}^{r_S} \gamma^i = \frac{S}{m}. \tag{8}$$

It follows that,

$$r_S = log_\gamma \left(\frac{S \cdot (\gamma - 1)}{m} + 1 \right) - 1, \tag{9}$$

and,

$$\gamma^{r_S} \cdot \frac{m}{RTT} = \frac{S \cdot (\gamma - 1) + m}{\gamma \cdot RTT}. \tag{10}$$

Hence, when the transfer is completed in the slow start phase before reaching R_{max}, the transfer time is:

$$E[L_{ss}] = log_\gamma \left(\frac{S \cdot (\gamma - 1)}{m} + 1 \right) \cdot RTT \text{ and } E[L_{ca}] = 0 \text{ if } \frac{S \cdot (\gamma - 1) + m}{\gamma \cdot RTT} \leq R_{max}. \tag{11}$$

2) Transfer completed in the congestion avoidance phase: The content size is longer than being achieved in the slow start phase, so it continues its transmission in the congestion avoidance phase (or in the steady state) where the transmission is completed after that the sender rate has reached R_{max}:

$$\frac{S \cdot (\gamma - 1) + m}{\gamma \cdot RTT} > R_{max}. \tag{12}$$

We evaluate the window expected to be reached at the end of the slow start phase by the following expression:

$$E[W_{ss}] = R_{max} \cdot \frac{RTT}{m} = \gamma^n = \begin{cases} A \cdot \frac{RTT}{m} & \text{if } R_{max} = A \\ W_{max} & \text{if } R_{max} = W_{max} \cdot \frac{m}{RTT} \\ \frac{1}{\gamma} \cdot (\frac{\gamma - 1}{P} + 1) & \text{if } R_{max} = \frac{1}{\gamma} \cdot (\frac{\gamma - 1}{P} + 1) \cdot \frac{m}{RTT} \end{cases} \tag{13}$$

So, we express the number of rounds n which is required to reach the window size $E[W_{ss}]$ (i.e., to reach R_{max}) since the beginning of the transfer as:

$$n = \begin{cases} log_\gamma(A \cdot \frac{RTT}{m}) & \text{if } R_{max} = A \\ log_\gamma W_{max} & \text{if } R_{max} = W_{max} \cdot \frac{m}{RTT} \\ log_\gamma(\frac{\gamma - 1}{P} + 1) - 1 & \text{if } R_{max} = \frac{1}{\gamma} \cdot (\frac{\gamma - 1}{P} + 1) \cdot \frac{m}{RTT} \end{cases} \tag{14}$$

We note $r_n + 1$ the number of slow start rounds required to transfer $E[S_{ss}]$ data bytes in the case where the transmission is completed after that the sending rate reaches R_{max} (see Equation (12)), thus:

$$r_n = \begin{cases} n & \text{if } R_{max} = A \\ n + \frac{d}{W_{max}} & \text{if } R_{max} = W_{max} \cdot \frac{m}{RTT} \\ n & \text{if } R_{max} = \frac{1}{\gamma} \cdot (\frac{\gamma - 1}{P} + 1) \cdot \frac{m}{RTT} \end{cases} \tag{15}$$

d is the average number of packets which is transmitted successfully in the slow start phase between the time when the transmitting buffer is saturated and the time when the transfer is completed or a packet loss is occurred. We evaluate d in these two cases as the following:

$$d = \begin{cases} \frac{S}{m} - \sum_{i=0}^{log_\gamma W_{max}} \gamma^i & \text{if } \frac{S}{m} \leq \frac{1}{P} \\ \frac{1}{P} - \sum_{i=0}^{log_\gamma W_{max}} \gamma^i & \text{if } \frac{S}{m} > \frac{1}{P} \end{cases} \tag{16}$$

so,

$$d = \begin{cases} \frac{S}{m} - \frac{\gamma \cdot W_{max} - 1}{\gamma - 1} & \text{if } \frac{S}{m} \leq \frac{1}{P} \\ \frac{1}{P} - \frac{\gamma \cdot W_{max} - 1}{\gamma - 1} & \text{if } \frac{S}{m} > \frac{1}{P} \end{cases} \tag{17}$$

The time required to transfer $E[S_{ss}]$ data bytes can be expressed as:

$$E[L_{ss}] = RTT \cdot (r_n + 1), \tag{18}$$

Thus, Equations (14), (15), (17), and (18) give:

$$E[L_{ss}] = RTT \cdot \begin{cases} log_\gamma(A \cdot \frac{RTT}{m}) + 1 \\ \quad : \text{if } R_{max} = A \\ log_\gamma W_{max} + \frac{\frac{S}{m} + \frac{1}{\gamma - 1}}{W_{max}} - \frac{1}{\gamma - 1} \\ \quad : \text{if } R_{max} = W_{max} \cdot \frac{m}{RTT} \text{ and } \frac{S}{m} \leq \frac{1}{P} \\ log_\gamma W_{max} + \frac{\frac{1}{P} + \frac{1}{\gamma - 1}}{W_{max}} - \frac{1}{\gamma - 1} \\ \quad : \text{if } R_{max} = W_{max} \cdot \frac{m}{RTT} \text{ and } \frac{S}{m} > \frac{1}{P} \\ log_\gamma(\frac{\gamma - 1}{P} + 1) \\ \quad : \text{if } R_{max} = \frac{1}{\gamma} \cdot (\frac{\gamma - 1}{P} + 1) \cdot \frac{m}{RTT} \end{cases} \tag{19}$$

The maximum number of bytes that can be sent in the case where the transmission is completed after that the sending rate reaches R_{max} (the condition in Equation (12) is satisfied), can be expressed as:

$$E[S_{ss}] = m \cdot \begin{cases} \sum_{i=0}^{n} \gamma^i & \text{if } R_{max} = A \\ \sum_{i=0}^{n} \gamma^i + d & \text{if } R_{max} = W_{max} \cdot \frac{m}{RTT} \\ \sum_{i=0}^{n} \gamma^i & \text{if } R_{max} = \frac{1}{\gamma} \cdot (\frac{\gamma - 1}{P} + 1) \cdot \frac{m}{RTT} \end{cases} \tag{20}$$

Hence, we can evaluate the transfer time required to complete the transfer in the congestion avoidance phase or in the steady state phase (if the content size allows), as:

$$E[L_{ca}] = \frac{S - E[S_{ss}]}{R_{ca}}, \tag{21}$$

where R_{ca} is the TCP average throughput in the congestion avoidance phase (or in steady state). From [7], we use the following expression:

$$R_{ca} = min\left(A, W_{max} \cdot \frac{m}{RTT}, R_p\right) \quad (22)$$

where,

$$R_p = \frac{m}{RTT \cdot \sqrt{\frac{2 \cdot b \cdot P}{3}} + 4 \cdot RTT \cdot min(1, 3 \cdot \sqrt{\frac{3 \cdot b \cdot P}{8}}) \cdot P \cdot (1 + 32 \cdot P^2)} \quad (23)$$

So,

$$E[L_{ca}] = \frac{1}{R_{ca}} \cdot \begin{cases} S - \frac{1}{\gamma - 1} \cdot (\gamma \cdot A \cdot RTT - m) \\ \quad : \text{if } R_{max} = A \\ 0 \\ \quad : \text{if } R_{max} = W_{max} \cdot \frac{m}{RTT} \text{ and } \frac{S}{m} \leq \frac{1}{P} \\ S - \frac{m}{P} \\ \quad : \text{if } R_{max} = W_{max} \cdot \frac{m}{RTT} \text{ and } \frac{S}{m} > \frac{1}{P} \\ S - \frac{m}{P} \\ \quad : \text{if } R_{max} = \frac{1}{\gamma} \cdot (\frac{\gamma - 1}{P} + 1) \cdot \frac{m}{RTT} \end{cases} \quad (24)$$

Therefore,

$$PTT = \begin{cases} Equation\ (2) + Equation\ (11) \\ \quad : \text{if } \frac{S \cdot (\gamma - 1) + m}{\gamma \cdot RTT} \leq R_{max} \\ Equation\ (2) + Equation\ (19) + Equation\ (24) \\ \quad : \text{if } \frac{S \cdot (\gamma - 1) + m}{\gamma \cdot RTT} > R_{max} \end{cases} \quad (25)$$

Computing PTT is of low complexity. Indeed, the parameters used in this function can be determined dynamically by the server without any major difficulty. Practically, the server probes directly the client using any method that can estimate the available bandwidth, the round trip time, and the loss rate on the path with the client [5], [9]. Besides, the server can determine easily its request arrival rate λ and its request service rate μ by implementing the correspondent agents described above.

B. Prediction Evaluation

To determine the accuracy of our PTT function, we compute the ratio of the computed PTT over the measured $ReTT$ by downloading 40 files, of various sizes ranging between $5KB$ and $100MB$, from 20 ftp servers located in Europe, USA, and Asia [10]. We obtain an average ratio (PTT over $ReTT$) equal to 96% with a probability 95% that the real average ratio is in the interval: $[91\%, 99\%]$. We validate in Figure 2 the accuracy of the prediction in the case of short files, where the transfer is very probably completed in the slow start phase, and in Tables I, II, and III in the case of large files, where the transfer is very probably completed in the congestion avoidance phase (or the steady state).

We do not consider in our results the mean request waiting time in the server (Equation (2)) due to the inability to read remotely from the WWW servers the parameters required to compute such information (λ, μ and c). For non highly loaded servers, the request waiting time can be neglected. Concerning the estimation of A, we use a technique similar to that proposed in [11]. We send 4 separate 23-packet streams. The size of the first packet in a stream is 32 bytes, and the size of each other packet in the stream is the size of the previous one incremented by 32 bytes. We calculate the rate of each receiving stream's echos, then we evaluate A as the average value of the obtained rates for the 4 streams.

Figure 2 shows the accuracy of the transfer time prediction for 10 small size contents (between $5KB$ and $400KB$) gathered in Sophia Antipolis from 4 ftp servers located in Canada, Poland, Italy, and Netherlands. In this figure, each point is the average result of 6 times file transfer which are very probably achieved in the slow start phase. Then, we investigate the case of large size file transfer. In this case, we study the impact of the different parameters on the transfer time prediction.

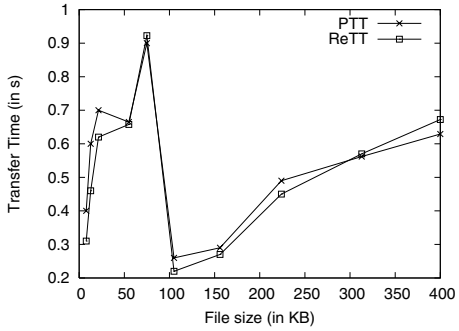

Fig. 2. Transfer time prediction for small size contents

To prove the weakness of the prediction based only on the geographical proximity, number of hops or RTT, we present the following scenario: a client (in Sophia Antipolis) downloads a file of size $75MB$ from two servers, S_{berlin} (in Berlin) and S_{paris} (in Paris), during a congested period. We choose to download such big file size during a congested period in order to observe the effect of the bandwidth limitation. As shown in Table I, S_{paris} is closer to the client than S_{berlin} in term of geographical proximity, number of hops and RTT. While the best server selection based on these criterias must be S_{paris}, the selection based on our PTT function is S_{berlin}, which is the correct choice verified by the real transfer time (ReTT). We observe (in both downloads) that the download rate obtained after dividing the file size by the transfer time, is limited by A. Thus, the good performance of our prediction in this scenario is caused by the fact that our PTT function considers the limitation of the available bandwidth (see Equation (22)).

Considering the available bandwidth alone is not sufficient; W_{max} should also be taken into account in the prediction function. Table II proves this claim, where the transfer is achieved in the congestion avoidance phase for the different

large document sizes collected from S_{berlin} (in Berlin) to the client (in Sophia Antipolis). During these connections, the measured parameters have the following values: A is equal to $24.34Mbps$, P is equal to zero, and W_{max} (imposed by the client maximum receiving window) is equal to $65535bytes$. We observe that the download rate obtained, after dividing each file size by its transfer time, is limited by $W_{max}*m/RTT$ (equal to $12.2Mbps$) even though there is more available bandwidth on the path and a negligible packet loss ratio. This is caused by the receiving window limitation which is taken into account in our metric (see Equation (22)). Thus, the window size limits the transmission rate to much less than the measured bandwidth on the path bottleneck, which is equal to $98.25Mbps$.

	S_{berlin}	S_{paris}
hops	13	10
RTT	41 ms	23 ms
A	10 Mbps	8 Mbps
P	0	0
PTT	60.30 s	75.12 s
ReTT	63 s	77 s
S/ReTT	9.523 Mbps	7.792 Mbps

TABLE I

TRANSFER TIME PREDICTION WHEN A LIMITS THE DOWNLOAD RATE

	PTT (in s)	ReTT (in s)	S/ReTT (in Mbps)
5.4 MB	3.85	4	10.8
14.63 MB	9.90	11	10.64
25.26 MB	16.88	18	11.23
66 MB	43.60	45	11.73
95.81 MB	63.16	64	11.98
102.24 MB	67.63	68	12.03

TABLE II

TRANSFER TIME PREDICTION WHEN W_{max} LIMITS THE DOWNLOAD RATE

S_{hkong}					
S	9.75MB	RTT	381.839ms	P	0.03
A	5.77Mbps	W_{max}	45	R_p	115.93kbps
PTT	688s	ReTT	701s	S/ReTT	113.87kbps
S_{poland}					
S	10.64MB	RTT	96 ms	P	0.04
A	6.6Mbps	W_{max}	45	R_p	370.156kbps
PTT	235.334s	ReTT	239s	S/ReTT	364.789kbps

TABLE III

TRANSFER TIME PREDICTION WHEN P LIMITS THE DOWNLOAD RATE

After showing the critical impact of the available bandwidth (in Table I) and the maximum receiving window size (in Table II) limitations on the transfer time prediction, we present in Table III a trace where the server's maximum sending rate is reduced due to a non-negligible packet loss rate. This trace is the result of 2 files transfer from 2 FTP servers (one located in Hong-Kong and another in Poland) to our end host in Sophia Antipolis. During these connections, the value of the packet loss rate is significant. We observe in Table III that the download rate obtained, after dividing each file size by its transfer time, is limited by R_p (see Equation (23)) which is less than the available bandwidth on the path and the limited rate imposed by W_{max}. Thus, the sending rate limitation is caused by the packet loss rate, which is considered in our metric (see Equation (22)).

C. Prediction Cost

As shown in Section III-B, using our PTT function can provide an accurate transfer time prediction, but it requires the measurement of the available bandwidth, RTT, and packet loss rate. Measuring the available bandwidth accurately may have an expensive cost for applications of short duration. CPROBE [9] probing tool has a default probing overhead approximately equal to $30,000$ bytes and up to 4 seconds of measurement time. Recently developed, ABwE [3] provides quick (< 1 second) measurements of available bandwidth. So, our prediction function can be more useful for applications such as downloading large files (like software from CDN, video from peer-to-peer network, etc.) where the overall response time (including the probing time) that can be obtained by serving from the best server is still less than the transfer time that can be obtained by serving from any other server.

IV. CONCLUSION

In this paper, we propose an efficient scheme to reduce the transfer time of a content stored in a set of replicated servers. Our solution is based on a metric that aims to predict accurately the transfer time of a content transmitted using TCP. Our prediction function considers the critical performance parameters that have an impact on the quality of the transfer such as the load of the servers and the characteristics of the path between the client and the servers. Our experimental results show that the server selection based on our prediction function outperforms the others solutions (e.g., those based only on the geographical proximity or the round trip time). Moreover, the results show that our proposed function predicts the transfer time of a content with around 96% accuracy.

REFERENCES

[1] Akamai, *http://www.akamai.com*.
[2] Kazaa, *http://www.kazaa.com*.
[3] J. Navratil, L. Cottrell, *ABwE: A Pratical Approach to Available Bandwidth Estimation*, PAM 2003, April 2003.
[4] S. Ratnasamy, M. Handly, R. Karp, S. Shenker, *Topologically-Aware Overlay Construction and Server Selection*, Infocom, June 2002.
[5] Cooperative Association for Internet Data Analysis, *http://www.caida.org/*.
[6] Z. Fei, S. Battacharjee, E. W. Zegura, M. H. Ammar, *A Novel Server Selection Technique for Improving the Response Time of a Replicated Service*, IEEE Infocom, March 1998.
[7] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose, *Modeling TCP Throughput: a Simple Model and its Empirical Validation*, in proceedings of ACM SIGCOMM, August 1998.
[8] Leonard Kleinrock, *Queueing systems volume I: theory*, 1975.
[9] BPROBE and CPROBE, *http://cs-people.bu.edu/carter/tools/Tools.html*.
[10] OpenBSD FTP, *http://www.openbsd.org/ftp.html*.
[11] R. Carter, and M. Crovella, *Dynamic Server Selection using Bandwidth Probing in Wide-Area Networks*, technical report, Boston, 1996.