INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

A Dynamic Architecture for Reducing the Response Time and Avoiding the Congestion

Mohammad Malli, Chadi Barakat, and Walid Dabbous

N° 5195 – version 2

version initiale May 2004 – version révisée October 2004

Thème COM

ISSN 0249-6399

INRIA
SOPHIA ANTIPOLIS

A Dynamic Architecture for Reducing the Response Time and Avoiding the Congestion

Mohammad Malli, Chadi Barakat, and Walid Dabbous

Thème COM — Systèmes communicants
Projet Planète

Rapport de recherche n° 5195 – version 2* — version initiale May 2004 — version révisée October 2004 23 pages

Abstract: The replication of digital content over the Internet makes the identification of the best server an interesting problem. In this paper, our aim is to reduce the content transfer time from a server to a client with avoiding network and server congestion using an efficient, scalable server selection scheme. First, we review the limitations of the anycasting schemes proposed in the literature. Then, we propose an application-layer anycasting scheme which is a scalable, transparent, and dynamic solution. Our anycasting scheme is based on a metric which can predict the transfer time of a content transmitted using TCP. Our metric considers the server load, the server's maximum sending window size, and the client's maximum receiving window size. Also, it considers the critical performance parameters on the path server - client (i.e., the available bandwidth, the round trip time, and the packet loss rate). Our empirical results show the weakness of the other prediction schemes. Moreover, the results show that our proposed function predicts the transfer time of a content with an 96% accuracy compared to the real transfer time.

Key-words: Prediction, Transfer Time, Anycasting, Service replication

* This second version contains more experimental results than the first version of the same report

Unité de recherche INRIA Sophia Antipolis
2004, route des Lucioles, BP 93, 06902 Sophia Antipolis Cedex (France)
Téléphone : +33 4 92 38 77 77 — Télécopie : +33 4 92 38 77 65

Une architecture dynamique pour réduire le temps de réponse et éviter la congestion

Résumé : La réplique du contenu numérique au-dessus de l'Internet fait à l'identification du meilleur serveur un problème intéressant. Dans cet article, notre but est de réduire le temps de transfert d'un contenu d'un serveur à un client avec l'évitation de la congestion de réseau et de serveur en utilisant une approche dynamique, efficace et scalable pour choisir un serveur. D'abord, nous passons en revue les limitations des mécanismes anycasting proposés dans la littérature. Puis, nous proposons un mécanisme anycasting au niveau applicative qui est une solution scalable, transparente, et dynamique. Notre mécanisme est basé sur une métrique qui peut prévoir la période de transfert d'un contenu transmis en utilisant TCP. Notre métrique considère la charge de serveur, la taille maximum du fenêtre d'envoi du server, et la taille maximum du fenêtre de réception du client. En outre, il considère les paramètres critiques concernant la performance sur le chemin serveur - client (c.à.d., la largeur de bande disponible, le temps rond de voyage, et le taux de perte de paquet). Nos résultats empiriques montrent la faiblesse de prévision des autres approches. D'ailleurs, les résultats prouvent que notre fonction proposée prévoit la période de transfert d'un contenu avec une exactitude de 96% comparée au vrai temps de transfert.

Mots-clés : Prédiction, Temps de tranfer, Service Replié, Anycast

1 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 offered to the clients. Several network environments can be used for service replication such as Content Distributed Networks (CDN), where client requests are forwarded by request redirectors and contents are stored in mirror servers which are geographically distributed over the Internet, or peer-to-peer overlay networks (e.g. Kazaa [6]), where peers behave as clients and servers.

Thus, a client that needs a service from a certain content provider within a certain name (e.g., www.download.com) needs to be served by the best server among the set of all the replicated servers of this provider. Similarly, a client that needs content from its overlay network should be served by the best overlay node.

In the following discussions, a server is either a server among a set of replicated servers or a node in an overlay network, which holds the desired content. The *best server* is the one which is able to provide to the client the desired service with a better QoS than it can be provided by the other servers or nodes. 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 localize the best server in order to minimize the time required to serve a client. We consider in this work a service that consists in 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 is minimized. Choosing the best server amounts then 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. For determining the best server, we consider the service availability, the load, and the buffering limitations of the server and the current performance on the server - client directed path (i.e., the round-trip time, the available bandwidth, and the loss rate).

One main contribution in this paper is a metric for the path between clients and servers that incorporates several server and path parameters and that allows to localize the best server in an efficient way. Based on this performance metric, we propose a new application-layer anycasting scheme for localizing the best server. Our scheme takes into account several issues such as (i) the service friendliness regarding the client, (ii) the feasibility to implement the scheme, (iii) the simplicity of the required measurements and gathering parameters, and

(iv) the location of the best server prediction. Before presenting our solution, we first present a short overview on the proposed solutions in the literature so as to motivate our work.

Many techniques have been proposed in the literature for optimal server selection. We mention some of them:

1. IP anycasting. RFC 1546 [10] defines anycasting as: *a stateless best effort delivery of an anycast datagram to at least one host, and preferably only one host, which serves the anycast address.* An IP anycast address identifies a set of replicated servers which offers the same service. A client, which is requesting a service using an IP anycast address as a destination address, is served by one server among the set of servers which have the same IP anycast address. This is done after that the request is routed by the anycast-aware routers until that the client request arrives to the first server having the same address marked in the packet as destination address. Thus, IP anycast consists in implementing in each routing table of each router a set of IP anycast addresses in order to have the capability of routing the packets which have IP anycast address as destination address. This requirement makes the deployment of IP anycast difficult over the Internet. However, even if it is deployed, this technique is not good enough since it does not consider the QoS criteria into account.

2. Using the DNS (Domain Name System) to get the IP address of the optimal server. This widely used technique is simple: the DNS servers distribute the multiple 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.

3. 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.

4. Using more sophisticated techniques that take into account one or many parameters having an impact on the content transfer time. [19] uses a binning strategy with landmark points to locate the server whose bin is the nearest to the client's bin. The bin of a network entity (client or server) is constructed by measuring the delays to landmark points. [20] 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 a server as for example the measurement of the available bandwidth. [11] proposes a prediction of the data transfer time between a client and a server based on the measurement of the delay and the available bandwidth on the path between them. [12] considers only the number of hops and round-trip time parameters for choosing the best server.

The binning solution in [19] requires that the client determines its own bin by measuring the RTT values to a set of landmark points. Knowing the bin of the client and servers, the DNS server sends to the client the IP address of the best server which is placed in the nearest bin to the client's one. This binning solution, even though it does not require that the client knows the list of servers, still requires tasks from the client for determining its bin. Moreover, the client does not measure the performance on the path to the servers but rather to landmark points. The drawback of this behavior can be illustrated 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 must be considered as well, as the available bandwidth and the server load. Thus, the binning solution may not determine the best server due to the weak of characterization of the performance on the path between the client and the server.

[11] requires that a home server sends the list of replicated servers to the client which must probe each server in order to choose the best one. This solution only considers the RTT and the available bandwidth for characterizing the path client - server. It does not take into account the packet loss rate on the path server - client which can have a bad impact on the performance of the content delivery. Moreover, this scheme does not consider the server load.

The main problem of the solution presented in [20] is the fact that it requires installation of probing agents to act as client proxies in order to characterize the paths to the servers. To be scalable, this scheme requires installation of a proxy for each nearby set of clients. It estimates a content transfer time from a server to a proxy, but this is not necessarily the same transfer time in the reality since the transfer is realized from the server to the client, even though the proxy and the client can be nearby located. The drawback of this assumption appears 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 path server - client while it does not exist on the path server - proxy.

In contrast with the other propositions, our solution localizes the best server by satisfying the following requirements: (i) our solution resides at the application layer. This avoid any changing in the existing infrastructure (i.e., it does not require any modification in the routers). Indeed, it does not require to install proxies over the Internet. (ii) the best server is selected after taking into account the current performance on the paths directed servers - client rather than the performance on the paths directed client-servers or those directed client proxy-servers. (iii) the path server - client is characterized by taking into account its round trip time, its current available bandwidth, and its packets loss rate, (iv) the best server is selected after taking into account the server's buffering capacity and load as well as the client's receiving window limitation. (v) the best server is characterized transparently to the client. (vi) the solution is scalable and does not require an expensive implementation in term of hardware installations.

This paper is organized as follows. The next section describes our application-layer anycasting scheme: the goals, and the communication protocol. Then, we present in Section 3: our prediction function, the prediction evaluation, and the prediction cost. Finally, the conclusion is presented in Section 4.

2 Application-layer Anycasting Scheme

2.1 Goals

Our application layer anycasting scheme provides a scalable, transparent, and dynamic solution for serving a client request. It is a scalable scheme because its implementation does not depend on the number of clients nor on their locations. It is transparent to the client since it provides him the IP address of the best server without requiring any particular effort from the client other than sending its request to the central server (see Figure 1). Besides, it provides a dynamic server selection since it characterizes the current performance on the server and on the paths between the servers and the client in order to determine the best server. This behavior contributes not only to improve the QoS perceived by the client, but also to avoid the congestion in the Internet and to traffic balancing since it takes into account the available bandwidth on the links. It also contributes to avoid the congestion of the servers since it takes into account the load of the servers in the server selection process. Keeping these features in mind, we describe next the behavior of our server selection scheme.

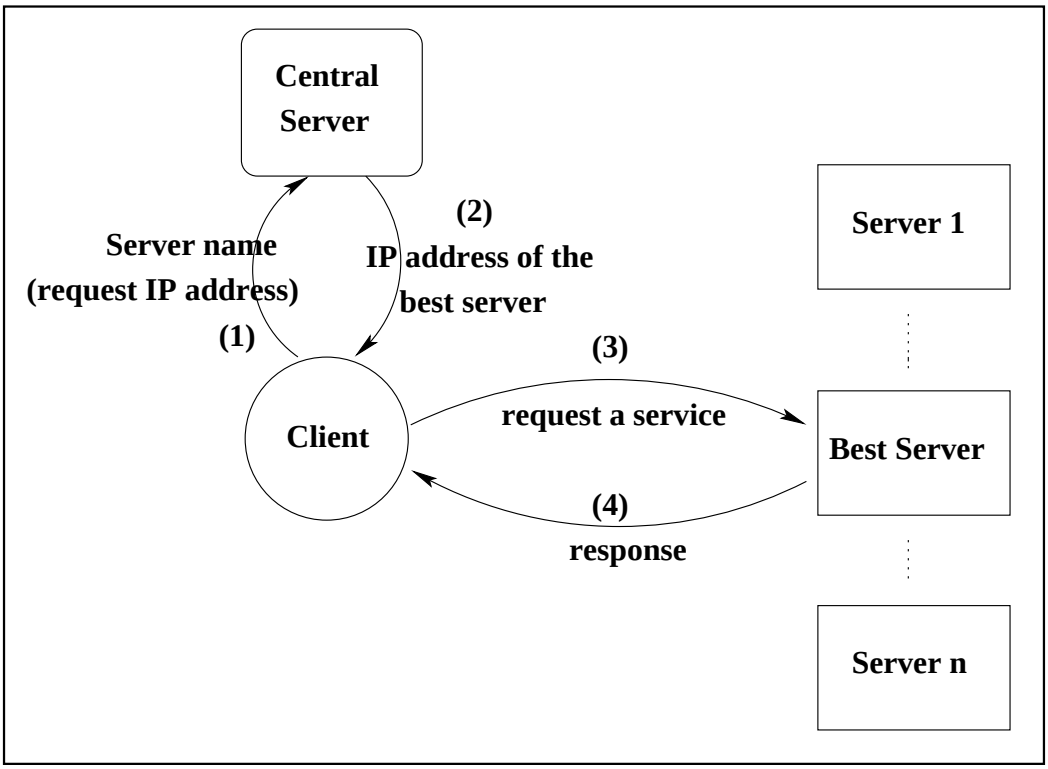

Figure 1: The determination of the best server is hidden regarding the client

To access a service, a client sends a request to the central server. The central server acts as an application-layer gateway for the requested service which is stored in a set of replicated servers distributed over the Internet. The transfer time prediction and the required performance measurement (described in Section 3.1) are then accomplished by the set of replicated servers and the result is transmitted to the central server. The central server delivers to the client in a certain way the IP address of the best server whose prediction of the content transfer time is the minimum compared to those values predicted by the other servers. To realize this task, the central server can send to the client an HTML page that contains hyperlinks pointing to the location of the requested content on the best server selected.

2.2 Communication Protocol

In this section, we describe the communication protocol between the basic application entities of our scheme: (i) in the client part is located: the *anycast aware client* which demands a service, (ii) in the central server part is located: the *service relay agent* which is the well-known application-layer gateway, and the *classifier agent* which identifies the best server to send its IP address to the client, (iii) in each server part is located: a *load-estimator* agent which calculates continuously the requests arrival rate and service rate in the server, the *probing agent* which probes the client and predicts the time needed to transfer the requested content to the client, and finally the *service application* which is obviously located in each server. We note that these three agents, located in the replicated servers, can also be implemented in the central server, which can predict the content transfer time like any other server and thus it can serve the client if its predicted transfer time is found to be the smallest one.

Figure 2 shows the basic steps realized before establishing the connection between the client and the best server:

1. The anycast aware client sends his request to the service relay agent located in the central server to establish a TCP connection between them.

2. The service relay agent responds by sending an HTML page to the client. For example, this page can be the home page of the server which contains general information about the services provided by the server and a notice which informs the client to wait a moment until receiving the HTML page which contains a link to the desired content

or service. Besides, the service relay agent calls the classifier agent and gives it the client IP address and the client receive buffer (obtained from the window size advertised by the client).

3. The classifier agent sends the client's IP address and receiving buffer space (i.e., maximum receiving window) to the probing agents located in a certain set of servers. For example, to serve a French client, the classifier agent sends the client's IP address to the probing agents located in the replicated French servers.

4. Each probing agent, among the chosen set, probes the client in order to obtain the performance on the path server - client and reads the current server performance parameters determined by the load estimator agent.

5. Each probing agent, among the chosen set, computes our predicted transfer time metric defined in Section 3.1.

6. Each probing agent, among the chosen set, sends the obtained value of the predicted transfer time to the classifier agent.

7. The classifier agent compares between the different values received from the different probing agents in order to obtain an ordered list of the content transfer time.

8. The classifier agent sends to the client the IP address of the best server selected (which corresponds to the smallest transfer time value) explicitly or implicitly via a hyperlink in an HTML page.

9. The TCP connection is established between the client and the best server.

Basically, each server predicts the time required to serve the requested client by using our predicted transfer time function defined in Equation (25). To obtain the parameters of this function, the probing agent reads the current server performance parameters determined by the load-estimator agent and probes the client requesting the service to obtain the performance parameters on the path server - client. Then, it sends the obtained value to the classifier agent which compares between the values received from the different servers in order to know the best one which can serve the client in the smallest time. Hence, the classifier agent sends to the client the IP address of the best server explicitly or implicitly via an hyperlink in an HTML page.

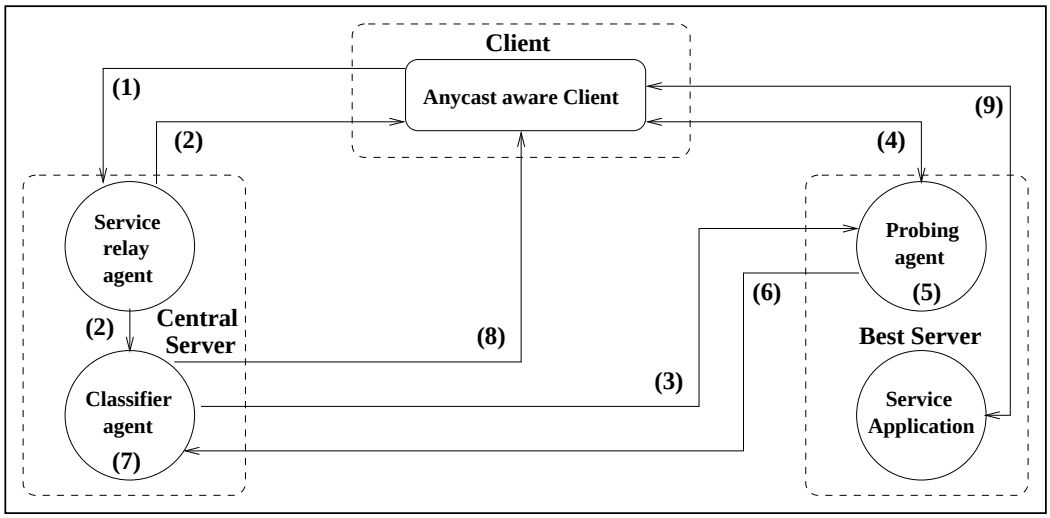

Figure 2: Application-layer anycasting architecture

The probing agent considers, in predicting the service transfer time: (i) the performance on the path with the client: the available bandwidth noted by A, the round trip time noted by RTT, and the packet loss rate noted by P, (ii) the performance 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). It is obvious that this idle time depends on the server load. (iii) the performance of the client, which is represented by its receive buffering space communicated to the central server via the advertised window field in the TCP header.

We propose that the probing agent is implemented in the server part, for these two main reasons: (i) the probing agent can easily read the size S of the requested content and the performance parameters of the server: λ and μ without sending to the network any message asking the values of these parameters, (ii) providing the service transparently to the client.

3 Transfer Time Prediction

3.1 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 noted 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 requests arrival rate, and μ is the mean service rate. Using known results from queuing theory [16], 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 (residing 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: (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 that 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.

3.1.1 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:

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

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:

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

3.1.2 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 (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 the steady state). From [14], we use the following expression:

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

where,

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

So,

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

Therefore,

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

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 [1, 4]. Besides, the server can determine easily its requests arrival rate λ and its requests service rate μ by implementing the correspondent agents described above.

3.2 Prediction Evaluation

We propose to use our PTT function (Equation (25)) to predict the transfer time of a content transmitted using TCP. It can reflect the transfer time more precisely than the other proposed policies which consider geographical proximity, number of hops, or RTT measurements, since it considers the critical performance limitations on the server and on the path server - client (see Section 2.2).

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 [8]. 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 3 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 1, 2, and 3 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 [?]. 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 3 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.

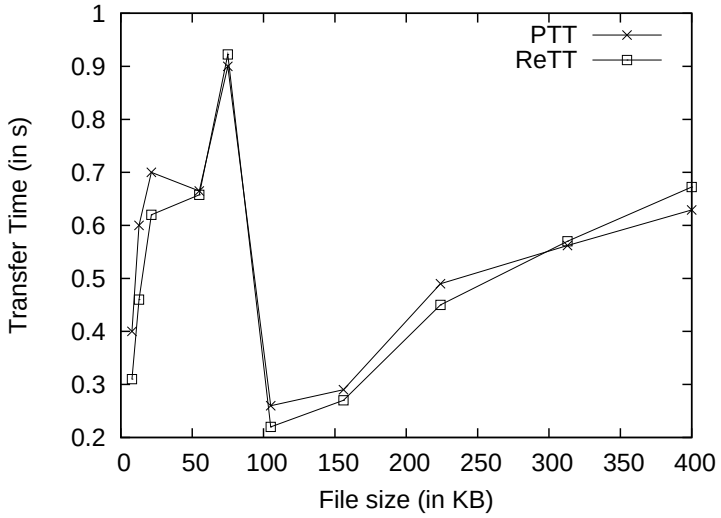

Figure 3: Transfer time prediction for small size contents

To prove the weakness of the prediction based on the geographical proximity, number of hops (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 1, 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 parameters 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 is not sufficient if W_{max} is not taken into account in the prediction function. Table 2 shows this case, 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 sending 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 1: 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 2: Transfer time prediction when W_{max} limits the download rate

After showing the critical impact of the available bandwidth (in Table 1) and the maximum sending window size (in Table 2) limitations on the transfer time prediction, we present

in Table 3 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 3 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, as it is considered in our metric (see Equation (22)).

3.3 Prediction Cost

To define a prediction method, we must take into account the tradeoff between the two following constraints: (i) improve the prediction capability of the algorithm by taking into account the critical performance parameters that can have an impact on the QoS, (ii) reduce the cost that can result from such prediction. While a weak best server prediction can not provide a better QoS since the client is not served from the real best server, the cost of an accurate prediction can have a bad impact on both: (i) the service response time due to the extra time passed in the prediction phase, and (ii) the network due to the extra bytes injected for probing. Therefore, the cost of the prediction must depend on the features of the service requested by the client. Briefly, the number of bytes injected to the network for probing must depend on the size of the content to be transfered to the client. Similarly, the time passed on probing must depend on the content transfer time. It is obvious that while a small content size may require an approximate prediction in order to reduce the number of bytes and the extra time used for probing, a large content size may allow a more accurate prediction for the best server with a limited cost.

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 3: Transfer time prediction when P limits the download rate

After proving that the number of hops is a poor predictor of latency, [11] proposes the OnePercent policy for measuring the round trip time as pursue: files under 10,000 bytes require a single ping, files under 20,000 bytes require the mean of two pings, etc until 50,000 bytes, and all the larger documents require the mean of 5 pings. Thus, they propose to use at most one ping per 10 Kbytes to be transferred. While a policy like the OnePercent can be efficient in term of reducing the probing cost, measuring RTT can not provide always an accurate best server selection. As shown in Section 3.2, even measuring RTT with more than 5 repetitions (the tools, that we are used, measure RTT with more than 5 repetitions) is not enough to have an accurate prediction for the best server specially when the content size is large and the service require a long client - server connection duration.

As shown in Section 3.2, using our PTT function can provide an accurate best server prediction but it requires the measurement of available bandwidth, RTT, and packet loss rate. Measuring the available bandwidth accurately may have an expansive cost for applications of short duration. CPROBE [1] probing tool has a default probing overhead approximately equal to 30,000 bytes and up to 4 seconds of measurement time. Recently developed, ABwE [13] 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.

4 Conclusion

In this paper, we propose a scalable, transparent, and dynamic application-layer anycasting scheme to reduce the transfer time of a content transmitted using TCP connection. Our solution is based on a metric which aims to predict accurately the content transfer time in order to be served from the best server among a set of replicated servers spread over the Internet. Our prediction function considers the critical performance parameters which have an important impact on the quality of the transfer: the load, and the maximum sending window of the server as well as the client maximum receiving window, also the available bandwidth, the round trip time, and the packet loss rate on the path with the client. Our obtained traces show that the server selection based on our prediction function is more accurate than 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 an 96% accuracy compared to the real transfer time.

References

[1] http://cs-people.bu.edu/carter/tools/tools.html.

[2] http://dsd.lbl.gov/ncs.

[3] http://www-nrg.ee.lbl.gov.

[4] http://www.caida.org.

[5] http://www.ietf.org/html.charters/ippm-charter.html.

[6] http://www.kazaa.com.

[7] http://www.microsoft.com.

[8] http://www.openbsd.org/ftp.html.

[9] http://www.veritest.com.

[10] T. Mendez C. Partridge and W. Milliken. Host anycasting service. *RFC 1546*, November 1993.

[11] R. Carter and M. Crovella. Server selection using dynamic path characterization in wide-area networks. *IEEE INFOCOM*, April 1997.

[12] J. Guyton and M. Schwartz. Locating nearby copies of replicated internet servers. *SIGCOMM '95*, August 1995.

[13] L. Cottrell J. Navratil. Abwe: A pratical approach to available bandwidth estimation. *PAM*, April 2003.

[14] D. Towsley J. Padhye, V. Firoiu and J. Kurose. Modeling tcp throughput: a simple model and its empirical validation. *ACM SIGCOMM*, August 1998.

[15] V. Jacobson. Congestion avoidance and control. *SIGCOMM '88*, August 1988.

[16] Leonard Kleinrock. Queueing systems volume i: theory. 1975.

[17] V. Paxson M. Allman and W. Stevens. Tcp congestion control. *RFC 2581*, April 1999.

[18] J. Postel. Transmission control protocol. *RFC 793*, September 1981.

[19] R. Karp S. Shenker S. Ratnasamy, M. Handly. Topologically-aware overlay construction and server selection. *IEEE INFOCOM*, June 2002.

[20] E. W. Zegura M. H. Ammar Z. Fei, S. Battacharjee. A novel server selection technique for improving the response time of a replicated service. *IEEE Infocom*, March 1998.

Unité de recherche INRIA Sophia Antipolis
2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex (France)

Unité de recherche INRIA Futurs : Parc Club Orsay Université - ZAC des Vignes
4, rue Jacques Monod - 91893 ORSAY Cedex (France)
Unité de recherche INRIA Lorraine : LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex (France)
Unité de recherche INRIA Rennes : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex (France)
Unité de recherche INRIA Rhône-Alpes : 655, avenue de l'Europe - 38334 Montbonnot Saint-Ismier (France)
Unité de recherche INRIA Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex (France)

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399