A Paracasting Model for Concurrent Access to Replicated Content

(Invited Paper)

Ka-Cheong Leung
Department of Computer Science
Texas Tech University
Lubbock, Texas 79409-3104, U.S.A.
E-mail: kcleung@cs.ttu.edu

Victor O. K. Li
Department of Electrical and Electronic Engineering
The University of Hong Kong
Pokfulam Road, Hong Kong, China
E-mail: vli@eee.hku.hk

***Abstract*— In this paper, we propose a framework to study how to effectively download a copy of the same document from a set of replicated servers. A generalized application-layer anycasting, known as paracasting, has been proposed to advocate concurrent access of a subset of replicated servers to cooperatively satisfy a client's request. Each participating server satisfies the request in part by transmitting a subset of the requested file to the client. The client can recover the complete file when different parts of the file sent from the participating servers are received. This framework allows us to estimate the average time to download a file from the set of homogeneous replicated servers, and the request blocking probability when each server can accept and serve a finite number of concurrent requests. Our results show that the file download time drops when a request is served concurrently by a larger number of homogeneous replicated servers, although the performance improvement quickly saturates when the number of servers used increases. If the total number of requests that a server can handle simultaneously is finite, the request blocking probability increases with the number of replicated servers used to serve a request concurrently. Therefore, paracasting is effective in using a small number of servers to serve a request concurrently, say, up to four.**

I. Introduction

The convergence of the computer, communications, entertainment, and consumer electronics industry is driving an explosive growth in multimedia applications [8]. The Internet provides a convenient and cost-effective communication platform for electronic commerce, collaboration on research and development, education, and entertainment. The success of the Internet arises from the capabilities to support efficient, survivable, robust, and reliable end-to-end data transfer services for adaptive applications running over a set of end-systems. Popular (multimedia) documents maintained at a non-replicated server can attract tremendous access requests to the server so that the server may not be able to handle the load and becomes the bottleneck.

To improve the user response time, throughput, and reliability, server replication [11] is an effective technique to provide a copy of the same document from a set of, possibly geographically dispersed, replicated servers. However, application-layer anycasting, which chooses the best replicated server for accessing a document is a non-trivial problem as the user response time depends on the load of the selected server and the characteristics (e.g. delay and available bandwidth) of the network path connecting the server to the client making the access request. Existing approaches proposed for the "best" server selection [10], [14], [15] rely on the uses of the round-trip times between client-server pairs and the server response times. This cannot help in spreading requests across a set of replicated servers. The server performance can therefore fluctuate dramatically due to significant load imbalance during a download session [13], resulting in the deterioration of the user's quality of service, such as the response time and the packet drop rate (for multimedia streaming).

Instead of finding the "best" server to fulfill a client's request, concurrent access to a set of servers for satisfying requests has been proposed [13]. The idea is to partition a document into a large number of small blocks and forward block-based requests to all replicated servers until all blocks are received. However, as discussed in [13], issues such as the conditions under which it is beneficial to apply the proposed dynamic parallel access and the optimal/preferred block size are still unresolved.

Since replicated servers are generally located at geographically dispersed locations, packets from different servers can take different paths to the servers. As the response time depends partly on the path delay, it can be inferred from the results of multipath routing[1] that it is effective in using a small number of paths, say up to three [6]. Moreover, the complexity and overhead for maintaining a large number of concurrent requests can be substantial [5].

Thus, we believe that, though we may select more than one server to satisfy a request, only a few replicated servers, instead of all, should be used to achieve load balancing at the servers and networks. We then need to address the questions of how many and which replicated servers should be used to satisfy a client's request, so as to provide a better quality of service to users and improved utilizations for a given set of replicated servers. Therefore, the concept and framework of such a generalized application-layer anycasting, known as *paracasting*, are needed.

A. Our Contributions

The focus of this work is to propose a framework to study the performance of the application-layer paracasting for concurrent access to replicated content over a set of homogeneous servers. This framework allows us to estimate the average time to download a file from the set of homogeneous replicated servers, and the request blocking probability when each server can accept and serve a finite number of concurrent requests.

There is a lack of analytical models to study the behaviour of parallel server access. To the best of our knowledge, there is only

[1]Multipath routing [1], [7], [9], where packets of a source may travel over multiple routes from a source to a destination, is a load balancing technique to spread out the traffic load across the network in order to alleviate network congestion.

0-7803-8239-0/03/$17.00 ©2003 IEEE.

one analytical study to investigate all-server parallel downloading with homogeneous clients and servers [5]. All-server parallel downloading does not necessitate to meet the desired system objectives, but it incurs substantial complexity and overhead for maintaining a large number of concurrent requests. Thus, this cannot provide an answer to the issues raised in paracasting. Hence, an analytical model for paracasting should be developed for optimizing the system performance.

We will investigate the effectiveness of paracasting by examining three basic questions:

- Does paracasting improve the system performance? If so, when?
- How many replicated servers should be participated concurrently to satisfy a download request so as to achieve the best performance?
- What is the cost of employing paracasting?

B. Organization of the Paper

This paper is organized as follows. Section II gives a paracasting model to conceptualize how replicated servers are participated concurrently to satisfy a download request. It also states the assumptions needed to simplify the subsequent discussion. Section III presents an analytical model to compute the average file download time, and the request blocking probability when a replicated server can handle up to a certain finite number of concurrent requests for file download. Section IV examines the analytical results derived from the framework and studies the effectiveness of paracasting. Section V concludes and discusses some possible extensions to our work.

II. Paracasting Model

Application-layer anycasting [15] is a communication service implemented at the application layer so that a sender can choose and interact with a destination belonging to an anycast group for performing the desired communication activity. To perform a file download by application-layer anycasting, a client generally selects a replicated server from a server pool based on the server load and the network path characteristics. This reactive approach cannot help in spreading requests across a set of replicated servers. Application-layer paracasting is thus proposed to exploit further load balancing proactively over a set of replicated servers.

Paracasting is a generalized communication technique for application-layer anycasting. By application-layer paracasting, a communication activity is divided into several tasks. A sender can freely choose a destination within a group of destinations (defined as a paracast group) for performing each of these tasks. In other words, there can have more than one destination within the paracast group involved for the communication activity, though the sender interacts with a single destination for carrying out a task of the given communication activity.

To perform a file download by paracasting, the file is divided into several parts. Each portion of the file is then downloaded from one of the replicated servers. The client requesting the file download can recover the complete file by resembling various parts of the file downloaded from these servers.

The scope of this work is to study the performance of application-layer paracasting for concurrent access to replicated content over a set of homogeneous servers. The following assumptions are made to simplify the discussion:

1) There are N homogeneous replicated file servers in the system. Each server has the maximum capacity of C_s bits per second. It can serve up to K requests concurrently using the processor-sharing policy. An incoming request to a server will be blocked when the server already has K requests. Any blocked request is considered lost and will not be retried.
2) The average file size for a download request is S bits. Whenever w replicated servers are involved, these servers are selected randomly from the server pool and the client will request $\frac{1}{w}$ of the file to be downloaded concurrently from each of these servers.
3) The service rate of each request at a server depends on the number of requests that the server is serving concurrently. The service time conditioned with the state of the server is assumed to be exponentially distributed in Section III-A. This assumption can be relaxed in Section III-B, where the service time is assumed to be generally distributed when an infinite value of K is considered.
4) There are a large number of clients in the system. Each client can make requests independently to the replicated servers for downloading files. The inter-arrival time between any two successive requests to the server pool is assumed to be exponentially distributed with the mean of λ requests per second.
5) The maximum aggregate download capacity for a client's request is C_c bits per second. This bottleneck bandwidth is shared statistically and fairly by all participating servers to the client. The available bandwidth for file download from a participating server to the client is r bits per second, where $0 < r \leq C_c$.

When K is finite, the Markovian service rate as stated in Assumption 3 is necessary so as to make the analysis mathematically tractable. The relaxation of this assumption can be part of the future work.

Assumption 5 is not overly restrictive, because many clients are connected to the Internet over a bandwidth-constrained connection, such as a cable modem connection. This bandwidth-constrained connection is thus the bottleneck for downloading a (large) file from the replicated servers to a client and its bandwidth remains unchanged regardless of how many replicated servers are participating for satisfying a download request. Moreover, active TCP (Transmission Control Protocol) flows tend to share the available bandwidth equally when these flows have the same or similar RTT (round-trip time) [3].

III. Queueing Analysis

In this section, we present the analytical results about the performance of application-layer paracasting for concurrent file access over a set of N homogeneous servers. By paracasting, w out of N replicated servers are selected randomly to satisfy a client's request. Denote λ as the average request arrival rate (in requests per second) to the server pool, which can be computed as:

$$\lambda_s = \frac{w\lambda}{N} \tag{1}$$

Consider each homogeneous replicated server has a capacity C_s bits per second and the average file size for a download request is S bits. The load for the replicated server can be computed as follows:

$$\rho = \frac{\lambda_s \cdot \frac{S}{w}}{C_s} = \frac{\lambda S}{N C_s} \tag{2}$$

This discussion will proceed as follows. Section III-A discusses the analysis by assuming the Markovian request service rate when K is finite. This assumption is relaxed when the analysis is carried out with an infinite value of K in Section III-B.

A. Finite K with Markovian Service Rate

Let $\xi = \frac{C_s}{r}$ and $m = \lfloor \xi \rfloor$, where the available bandwidth for file download from a participating replicated server to a client is r bits per second. The average service rate of a server (in requests per second) when there are i requests being served can be calculated as:

$$\mu(i) = \begin{cases} \frac{i\,w\,r}{S} & \text{if } i = 1, 2, ..., \min(m,\ K); \\ \frac{w\,C_s}{S} & \text{if } m < K, i = m+1, m+2, ..., K. \end{cases} \tag{3}$$

When the request service time conditioned with the number of requests being served is exponentially distributed, each replicated server can be modelled as an M/M/1/K/PS queue. It can be shown [4] that the steady-state probability of having i requests being served by a replicated server, $p(i)$, can be written as:

$$\begin{aligned} p(i) &= p(0) \prod_{j=1}^{i} \frac{\lambda_s}{\mu(j)} \\ &= \begin{cases} \frac{(\xi\,\rho)^i}{i!} \cdot p(0) & \text{if } i = 1, 2, ..., \min(m,\ K); \\ \frac{\xi^m\,\rho^i}{m!} \cdot p(0) & \text{if } m < K, i = m+1, m+2, ..., K. \end{cases} \end{aligned} \tag{4}$$

where $p(0) = \frac{1}{\sum_{i=0}^{m-1} \frac{(\xi\,\rho)^i}{i!} + \sum_{i=m}^{K} \frac{\xi^m\,\rho^i}{m!}}$ since $\sum_{i=0}^{K} p(i) = 1$.

By Little's Theorem [4], the average time (in seconds) to download a file from the set of replicated servers to a client can be determined as:

$$\begin{aligned} T_F(r, S, C_c) &= \frac{\sum_{i=1}^{K} i\,p(i)}{\lambda_s \cdot [1 - p(K)]} \\ &= \frac{S \cdot (\sum_{i=1}^{m-1} \frac{(\xi\,\rho)^i}{(i-1)!} + \sum_{i=m}^{K} \frac{i\,\xi^m\,\rho^i}{m!})}{w\,C_s\,\rho \cdot (\sum_{i=0}^{m-1} \frac{(\xi\,\rho)^i}{i!} + \sum_{i=m}^{K-1} \frac{\xi^m\,\rho^i}{m!})} \end{aligned} \tag{5}$$

By normalizing $T_F(r, S, C_c)$ with respect to the average file size for a download request and the download capacity for a client's request, the normalized average download time is:

$$\begin{aligned} \tilde{T}_F(r) &= \frac{C_c\,T_F(r, S, C_c)}{S} \\ &= \frac{C_c \cdot (\sum_{i=1}^{m-1} \frac{(\xi\,\rho)^i}{(i-1)!} + \sum_{i=m}^{K} \frac{i\,\xi^m\,\rho^i}{m!})}{w\,C_s\rho \cdot (\sum_{i=0}^{m-1} \frac{(\xi\,\rho)^i}{i!} + \sum_{i=m}^{K-1} \frac{\xi^m\,\rho^i}{m!})} \end{aligned} \tag{6}$$

A client's request is considered blocked whenever any one of its requested servers to block its request. Thus, the request blocking probability can be calculated as:

$$P_F(r) = 1 - [1 - p(K)]^w \tag{7}$$

1) Evaluation of r: Denote $B_F(r)$ as the normalized average aggregate download capacity used for file download when the available bandwidth for file download from a participating server to the client is r bits per second. Since the bottleneck download capacity is shared fairly by all participating servers to the client, the optimal value of r, r^*, can be determined by solving the following optimization problem:

$$\begin{aligned} &\text{Minimize} \quad |B_F(r) - 1| \\ &\text{subject to} \quad 0 < r \leq C_c \end{aligned} \tag{8}$$

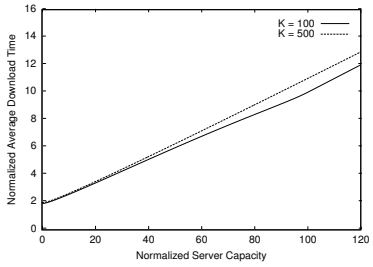


Fig. 1. Normalized average download time against normalized server capacity plot when $\rho = 0.95$, $\frac{C_s}{C_c} = 5$, and $w = 2$.

where $B_F(r) = \frac{1}{\tilde{T}_F(r)}$.

The relationship between the normalized average download time, $\tilde{T}_F(r)$, and the normalized server capacity, ξ, is demonstrated in Fig. 1. The normalized average download time increases when ξ increases from 0 to 120. A larger value of ξ means a smaller available bandwidth for file download from a participating server to the client. This in turns reduces the average aggregate file download capacity and thus increases the normalized download time. Since the normalized average aggregate download capacity used for file download, $B_F(r)$, is inversely proportional to the normalized average download time, $B_F(r)$ decreases when ξ increases. In addition, the available bandwidth for file download from a participating server to the client, r, is also inversely proportional to ξ, $B_F(r)$ increases when r increases. The results follow for all of the numerical experiments we have performed in this paper. Thus, $B_F(r)$ is a monotonic increasing function in r in general.

Intuitively, whenever the available download bandwidth is fully utilized, $B_F(r^*)$ attains a maximum value of 1. If all replicated servers are fully saturated, $B_F(r^*)$ is lower-bounded by $\frac{w\,C_s}{K\,C_c}$. In other words, the normalized average download time varies between 1 and $\frac{K\,C_c}{w\,C_s}$.

When $B_F(r)$ is monotonic increasing in r and $B_F(r)$ is at most r for $0 < r \leq C_c$, there exists an unique solution to the aforementioned optimization problem as exhibited in (8). Numerical root finding techniques like the secant method [12] can be adopted to compute the optimal value of r to the optimization problem. All of our numerical results presented in Section IV can be found correctly based on the captioned numerical root finding techniques and thus the monotonicity property of $B_F(r)$ holds in general.

B. Infinite K with General Service Rate

Let $\xi = \frac{C_s}{r}$ and $m = \lfloor \xi \rfloor$, where the available bandwidth for file download from a participating replicated server to a client is r bits per second. When the request service time is generally distributed and each server can serve any number of file download requests concurrently, each replicated server can be modelled as an M/G/1/∞/PS queue. It can be shown [2], [3] that the average time (in seconds) to download a file from the set of replicated servers to a client is as follows:

$$T_I(r, S, C_c) = \frac{S}{w} \cdot \{\frac{1}{r} + \frac{f(\rho) \cdot [1 - (\xi - m) \cdot (1 - \rho)]}{C_s \cdot (1 - \rho)}\} \tag{9}$$

where $0 \leq \rho < 1$ and $f(\rho) = \frac{\frac{(\xi\,\rho)^m}{m!}}{(1-\rho) \sum_{i=0}^{m-1} \frac{(\xi\,\rho)^i}{i!} + \frac{(\xi\,\rho)^m}{m!}}$ denotes the probability that a replicated server is saturated.

By normalizing $T_I(r, S, C_c)$ with respect to the average file size for a download request and the download capacity for a client's

request, the normalized average download time is:

$$\tilde{T}_I(r) = \frac{C_c}{w\,C_s} \cdot \{\xi + \frac{f(\rho)}{1-\rho} \cdot [1 - (\xi - m) \cdot (1-\rho)]\} \qquad (10)$$

1) Evaluation of r: Denote $B_I(r)$ as the normalized average aggregate download capacity used for file download when the available bandwidth for file download from a participating server to the client is r bits per second. The optimal value of r, r^*, can be determined by solving the following optimization problem similar to (8) in Section III-A.1:

$$\begin{aligned} &\text{Minimize} \quad |B_I(r) - 1| \\ &\text{subject to} \quad 0 < r \le C_c \end{aligned} \qquad (11)$$

where $B_I(r) = \frac{1}{\tilde{T}_I(r)}$.

Intuitively, whenever the available download bandwidth is fully utilized, $B_I(r^*)$ attains a maximum value of 1. The following three lemmas establish that $B_I(r)$ is monotonically increasing in r.

Lemma 1: For any natural number m, $B_I(r)$ is a monotonically increasing function in $r \in (\frac{C_s}{m+1}, \frac{C_s}{m}]$, where $\xi = \frac{C_s}{r}$ and $\xi \cdot (1-\rho) \ge (\xi - m) \cdot f(\rho) \cdot \{[1 + (1-\rho) \cdot (m - \xi\,\rho)] - \rho \cdot [1 - (1-\rho) \cdot (\xi - m)] \cdot f(\rho)\}$.

Proof: Differentiating $f(\rho)$ with respect to ξ,

$$\begin{aligned} \frac{\partial f}{\partial \xi} &= \frac{\frac{m\,\rho \cdot (\xi\,\rho)^{m-1}}{m!}}{(1-\rho)\sum_{i=0}^{m-1} \frac{(\xi\,\rho)^i}{i!} + \frac{(\xi\,\rho)^m}{m!}} \\ &\quad - \frac{\frac{(\xi\,\rho)^m}{m!} \cdot [(1-\rho)\sum_{i=1}^{m-1} \frac{i\,\rho \cdot (\xi\,\rho)^{i-1}}{i!} + \frac{m\,\rho \cdot (\xi\,\rho)^{m-1}}{m!}]}{[(1-\rho)\sum_{i=0}^{m-1} \frac{(\xi\,\rho)^i}{i!} + \frac{(\xi\,\rho)^m}{m!}]^2} \\ &= (\frac{m}{\xi} - \rho) \cdot f(\rho) + (1 - \frac{m}{\xi}) \cdot \rho \cdot [f(\rho)]^2 \end{aligned} \qquad (12)$$

Differentiating $\tilde{T}_I(r)$ with respect to ξ,

$$\begin{aligned} &\frac{\partial \tilde{T}_I}{\partial \xi} \\ &= \frac{C_c}{w\,C_s} \cdot [1 + \frac{1}{1-\rho} \cdot \frac{\partial f}{\partial \xi} + (m - \xi) \cdot \frac{\partial f}{\partial \xi} - f(\rho)] \\ &= \frac{C_c\{\xi \cdot (1-\rho) + (m-\xi) \cdot [1 + (1-\rho) \cdot (m - \xi\,\rho)] \cdot f(\rho)\}}{w\,C_s\,\xi \cdot (1-\rho)} \\ &\quad + \frac{C_c \cdot (\xi - m) \cdot \rho \cdot [1 + (1-\rho) \cdot (m - \xi)] \cdot [f(\rho)]^2}{w\,C_s\,\xi \cdot (1-\rho)} \\ &\ge 0 \end{aligned} \qquad (13)$$

since $\xi \cdot (1-\rho) \ge (\xi - m) \cdot f(\rho) \cdot \{[1 + (1-\rho) \cdot (m - \xi\,\rho)] - \rho \cdot [1 - (1-\rho) \cdot (\xi - m)] \cdot f(\rho)\}$.

Differentiating $B_I(r)$ with respect to r,

$$\begin{aligned} \frac{\partial B_I}{\partial r} &= \frac{\partial B_I}{\partial \tilde{T}_I} \cdot \frac{\partial \tilde{T}_I}{\partial \xi} \cdot \frac{\partial \xi}{\partial r} \\ &= \frac{C_s}{(r\,\tilde{T}_I)^2} \cdot \frac{\partial \tilde{T}_I}{\partial \xi} \\ &\ge 0 \end{aligned} \qquad (14)$$

since $B_I(r) = \frac{1}{\tilde{T}_I(r)}$ and $\xi = \frac{C_s}{r}$.

Thus, the monotonicity of $B_I(r)$ follows. ■

The inequality $\xi \cdot (1-\rho) \ge (\xi - m) \cdot f(\rho) \cdot \{[1 + (1-\rho) \cdot (m - \xi\,\rho)] - \rho \cdot [1 - (1-\rho) \cdot (\xi - m)] \cdot f(\rho)\}$ generally holds for a sufficiently large ξ as $0 \le \xi - m < 1$ and $0 \le f(\rho) \ll \rho < 1$. This result follows for all of the numerical experiments we have performed in this paper.

Lemma 2: $B_I(r)$ is continuous in r, where $0 < r \le C_c$.

Proof: Taking $r = \gamma_+ = \frac{C_s}{m}$,

$$\begin{aligned} f(\rho)|_{r=\gamma_+} &= \frac{\frac{(\xi\,\rho)^{m-1}}{(m-1)!}}{(1-\rho)\sum_{i=0}^{m-2} \frac{(\xi\,\rho)^i}{i!} + \frac{(\xi\,\rho)^{m-1}}{(m-1)!}} \\ &= \frac{\frac{m}{\xi\,\rho} \cdot \frac{(\xi\,\rho)^m}{m!}}{(1-\rho)\sum_{i=0}^{m-1} \frac{(\xi\,\rho)^i}{i!} + \frac{m}{\xi} \cdot \frac{(\xi\,\rho)^m}{m!}} \\ &= \frac{1}{\rho} \cdot f(\rho)|_{r=\gamma} = \frac{1}{\rho} \cdot f(\rho)|_{r=\gamma_-} \end{aligned} \qquad (15)$$

for any positive integer m.

Hence,

$$\begin{aligned} &B_I(\gamma_+) \\ &= \frac{w\,C_s}{C_c} \cdot \frac{1}{m + \frac{f(\rho)|_{r=\gamma_+}}{1-\rho} \cdot \{1 - [m - (m-1)] \cdot (1-\rho)\}} \\ &= B_I(\gamma) = B_I(\gamma_-) \end{aligned} \qquad (16)$$

Thus, the continuity of $B_I(r)$ follows. ■

Lemma 3: $B_I(r)$ is a monotonic increasing function in r, where $0 < r \le C_c$, provided that $\xi \cdot (1-\rho) \ge (\xi - m) \cdot f(\rho) \cdot \{[1 + (1-\rho) \cdot (m - \xi\,\rho)] - \rho \cdot [1 - (1-\rho) \cdot (\xi - m)] \cdot f(\rho)\}$.

Proof: The results follow directly from Lemmas 1 and 2. ■

When $B_I(r)$ is a monotonic increasing in r and $B_I(r)$ is at most r for $0 < r \le C_c$, there exists an unique solution to the aforementioned optimization problem as stated in (11). Numerical root finding techniques like the secant method [12] can be utilized to find the optimal value of r to the optimization problem.

IV. Performance Evaluation

This section discusses the numerical results based on the analytical expressions obtained in Section III. To support paracasting, up to 20 replicated servers are used to serve a single file download request concurrently. The ratio between the maximum server capacity and the maximum aggregate download capacity for a client's request, $\frac{C_s}{C_c}$, takes two different values, namely 5 and 30. These values can be used to simulate the various combinations of link/server speeds in a network. For example, each replicated server connects to the network with the T3 link speed of 45 Mbps, whereas a client connects to the network behind a cable modem connection with the download speed of 9 Mbps and 1.5 Mbps, corresponding to the first and second values respectively.

The results are provided in four sets. The first set examines the effect of the maximum number of concurrent requests a replicated server can handle to the normalized average file download time. The second set investigates the relationship among the normalized average download time, server load, and number of servers used to serve a single request concurrently. The third set studies the relationship among the request blocking probability, server load, and number of servers used to serve a single request concurrently. The fourth set observes the impact of the server load and number of servers used to serve a single request concurrently to the normalized effective download capacity.

A. Effect of K to $\tilde{T}_\bullet(r)$

The effect of the maximum number of concurrent requests a replicated server can handle, K, to the normalized average download time, $\tilde{T}_\bullet(r)$, is demonstrated in Fig. 2. When K is very small, a replicated server can serve all requests such that their download bandwidths are constrained by the maximum aggregate

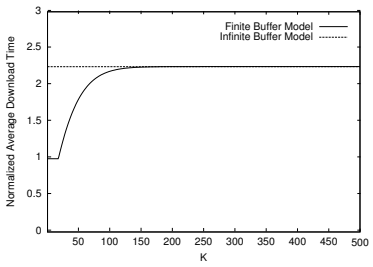


Fig. 2. Normalized average download time against K plot when $\rho = 0.95$, $\frac{C_s}{C_c} = 5$, and $w = 2$.

download capacities at the client sides. Thus, the file download time is determined by the incoming bottleneck bandwidth to the client.

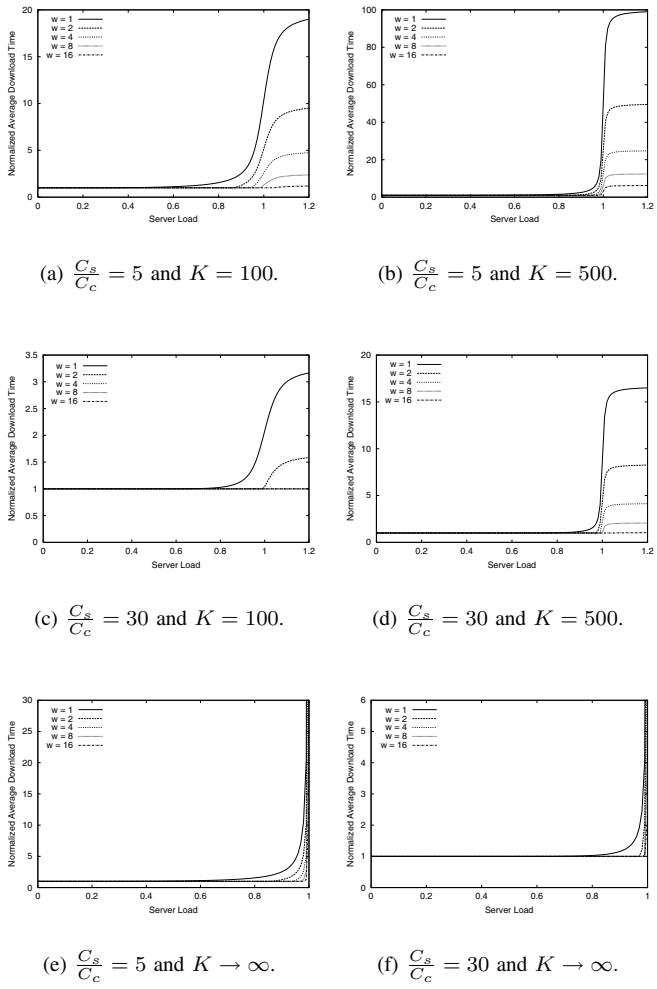
When K is greater than a certain value, the normalized average download time increases as the bandwidths sustained by some of the download requests are limited by the maximum capacities of the replicated servers. When K increases further, the normalized average download time increases slightly and then flattens. The converged download time is the same as the one computed based on the infinite buffer model discussed in Section III-B, as the tail distribution of the number of requests being served concurrently by a server can be captured when K is sufficiently large. Therefore, this shows an agreement between the results produced by both the finite buffer model (where K is finite) and infinite buffer model (where K tends to infinity) as they converge to the same download time when K increases.

B. Relationship Among $\tilde{T}_\bullet(r)$, ρ, and w

Fig. 3 exhibits the normalized average download time when the server load varies between 0 and 1.2 (finite buffer model) or between 0 and 1 (infinite buffer model). The normalized average download time increases with the server load. The download time rises substantially when the server load is high (about 0.9 and larger). When K is finite and the server load exceeds 1, the normalized average download time flattens and is upper-bounded by $\frac{K\,C_c}{w\,C_s}$, since each replicated server can serve up to K concurrent requests at any time. Moreover, the normalized average download time grows unboundedly when K tends to infinity.

Besides, the normalized average download time is approximately inversely proportional to the number of servers used to serve a single request concurrently, w. This suggests that it is beneficial to use more replicated servers to serve a single request concurrently until the minimum normalized average download time (which is 1) is achieved. Thus, this suggests that paracasting is effective in performance improvement when the server load is high.

Fig. 4 shows the normalized average download time when w varies between 1 and 20. The normalized average download time decreases when w increases. However, the improvement flattens with further increases in w. This suggests that the download time is approximately inversely proportional to w. The marginal cost of having an additional server to serve a single request concurrently increases with w. Hence, the argument favors that an appropriate value of w should be chosen to minimize the normalized average download time as well as the system cost, such as the request blocking probability, should be kept to an acceptable level.



(a) $\frac{C_s}{C_c} = 5$ and $K = 100$. (b) $\frac{C_s}{C_c} = 5$ and $K = 500$.

(c) $\frac{C_s}{C_c} = 30$ and $K = 100$. (d) $\frac{C_s}{C_c} = 30$ and $K = 500$.

(e) $\frac{C_s}{C_c} = 5$ and $K \to \infty$. (f) $\frac{C_s}{C_c} = 30$ and $K \to \infty$.

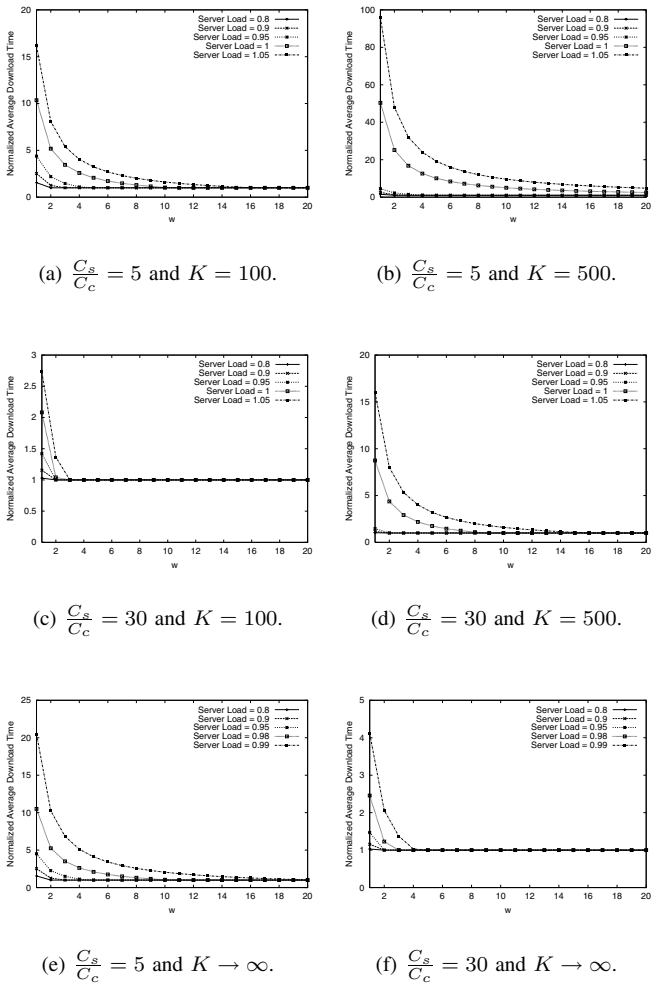
Fig. 3. Normalized average download time against server load plots for various settings.

C. Relationship Among $P_F(r)$, ρ, and w

Fig. 5 and 6 exhibit the impact of the server load and w to the request blocking probability when K is finite. The request blocking probability increases from 0 towards 1 when the server load increases from 0 to 1.2. The blocking probability increases when K decreases from 500 to 100 or w increases from 1 to 20. This means that the server pool administrators need to properly set the minimum value of K in order to limit the request blocking probability at a certain level for a given server load. In addition, the argument does not favour using a large w, say more than four, for paracasting. The determination of an optimal value of w by taking the system cost (including the request blocking probability and the overheads of using more than one server to satisfy a single request) into account is part of the future work.

D. Relationship Among $B_\bullet(r)$, ρ, and w

The relationship between the normalized effective download capacity and w is shown in Fig. 7. The normalized effective download capacity, which is defined as $\frac{w\,r}{C_c}$, represents the sum of all perceived download capacities from a set of participating replicated servers to a client when the bottleneck is at the client's side, such as the last-hop link to the client. Initially, the normalized



(a) $\frac{C_s}{C_c} = 5$ and $K = 100$. (b) $\frac{C_s}{C_c} = 5$ and $K = 500$.

(c) $\frac{C_s}{C_c} = 30$ and $K = 100$. (d) $\frac{C_s}{C_c} = 30$ and $K = 500$.

(e) $\frac{C_s}{C_c} = 5$ and $K \to \infty$. (f) $\frac{C_s}{C_c} = 30$ and $K \to \infty$.

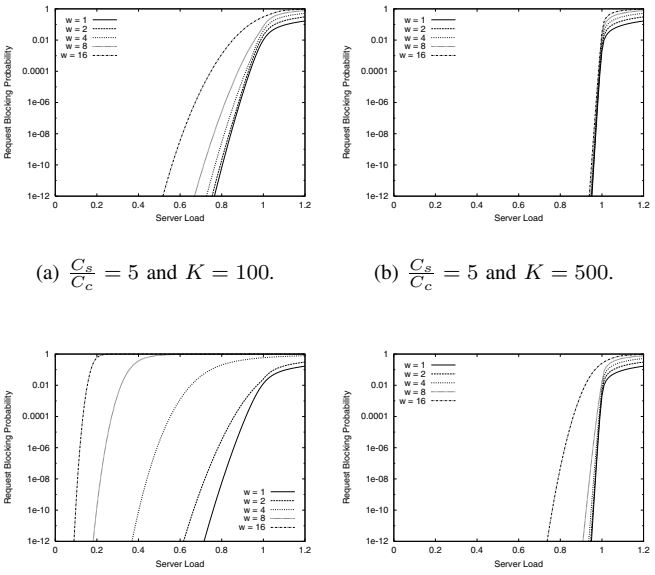
Fig. 4. Normalized average download time against w plots for various settings.

effective download capacity increases with w. The servers may not be able to serve the download request by saturating the client's incoming bottleneck. This means that a faster file download can be realized by using more replicated servers to serve a single request concurrently, i.e. increasing w. Since each replicated server is operated independently with another, statistical multiplexing on the shared bottleneck bandwidth is possible. The metric thus denotes the effectiveness of such statistical multiplexing on the bottleneck for the client's incoming capacity through paracasting.

However, the normalized effective download capacity drops to 1 when w continues to increase. This means that, when the bottleneck for the client's incoming capacity is saturated, it results in no further reduction on the download time by increasing w further. Thus, to improve the file download time, this argument does not favour to setting the value w greater than the one at which the normalized effective download capacity has begun to drop when w increases.

V. Conclusions

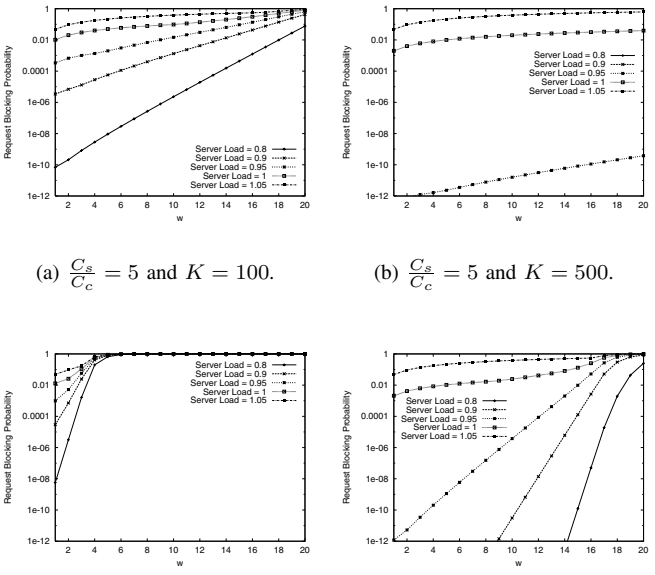
In this paper, we have proposed a framework to study the performance of application-layer paracasting for concurrent access



(a) $\frac{C_s}{C_c} = 5$ and $K = 100$. (b) $\frac{C_s}{C_c} = 5$ and $K = 500$.

(c) $\frac{C_s}{C_c} = 30$ and $K = 100$. (d) $\frac{C_s}{C_c} = 30$ and $K = 500$.

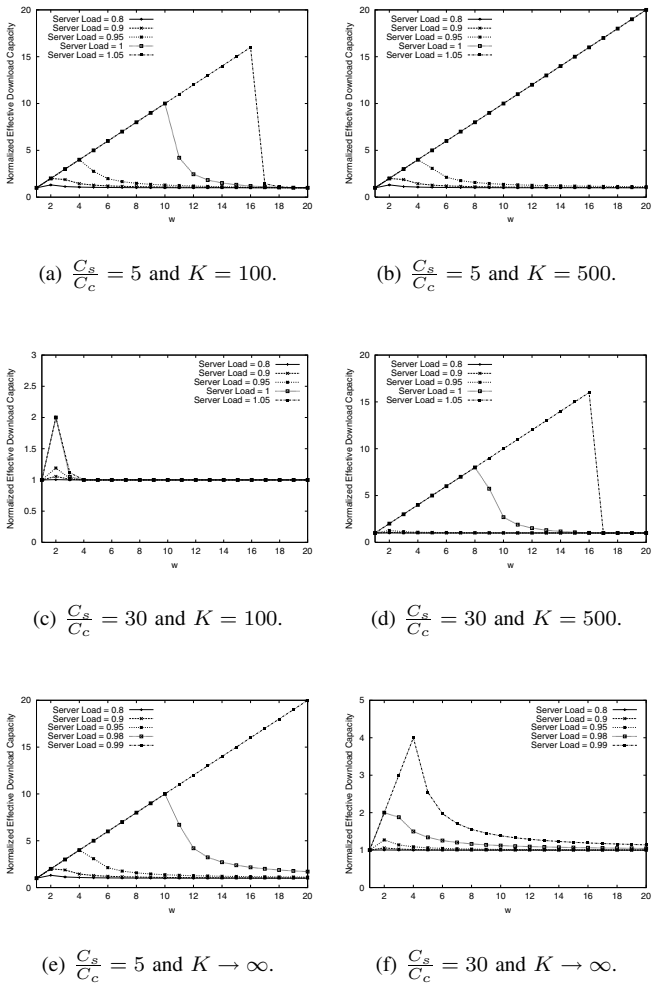
Fig. 5. Request blocking probability against server load plots for various settings.



(a) $\frac{C_s}{C_c} = 5$ and $K = 100$. (b) $\frac{C_s}{C_c} = 5$ and $K = 500$.

(c) $\frac{C_s}{C_c} = 30$ and $K = 100$. (d) $\frac{C_s}{C_c} = 30$ and $K = 500$.

Fig. 6. Request blocking probability against w plots for various settings.



(a) $\frac{C_s}{C_c} = 5$ and $K = 100$.

(b) $\frac{C_s}{C_c} = 5$ and $K = 500$.

(c) $\frac{C_s}{C_c} = 30$ and $K = 100$.

(d) $\frac{C_s}{C_c} = 30$ and $K = 500$.

(e) $\frac{C_s}{C_c} = 5$ and $K \to \infty$.

(f) $\frac{C_s}{C_c} = 30$ and $K \to \infty$.

Fig. 7. Normalized effective download capacity against w plots for various settings.

to replicated content over a set of replicated servers. This framework allows us to estimate the average time to download a file from the set of homogeneous replicated servers, and the request blocking probability when each server can accept and serve a finite number of concurrent requests. The paracasting model has been developed to conceptualize how replicated servers are participated concurrently to satisfy a download request. The queueing analysis to paracasting has been performed to allow us to compute the average file download time, and the request blocking probability when a replicated server can handle up to a certain finite number of concurrent requests for file download.

Our results show that the file download time drops when a request is served concurrently by a larger number of homogeneous replicated servers, although the performance improvement quickly saturates when the number of servers used increases. If the total number of requests that a server can handle simultaneously is finite, the request blocking probability increases with the number of replicated servers used to serve a request concurrently. Therefore, paracasting is effective in using a small number of servers to serve a request concurrently, say, up to four.

Now we re-visit the three questions posed in Section I. As expected, paracasting improves the system performance by serving a request through multiple replicated servers, thereby achieving load balancing. An optimal number of servers involved ensures that the server and network loads are well balanced. However, the use of paracasting may require a client to select a subset of replicated servers to satisfy a request, determine how a request is divided and which server downloads a specific portion of the file to the client, and combine downloaded fragment to recover the file. Moreover, the replicated servers are capable to download a selected portion of a file to a client.

There are several possible extensions to our work, some of which are listed as follows:

- extend the queueing analysis so that the Markovian service rate assumption can be relaxed when the total number of requests that a server can handle simultaneously is finite;
- extend the framework for determining an optimal number of replicated servers employed to serve a request concurrently by taking the system cost (including the request blocking probability and the overheads of using more than one server to satisfy a single request) into account; and
- devise a measurement-based algorithm for paracasting.

REFERENCES

[1] S. N. Chiou and V. O. K. Li. Diversity Transmissions in a Communication Network with Unreliable Components. *Proceedings of IEEE ICC '87*, Vol. 2, pp. 968-973, Seattle, WA, USA, 7-10 June 1987.

[2] J. W. Cohen. The Multiple Phase Service Network with Generalized Processor Sharing. *Acta Informatica*, Vol. 12, pp. 245-284, 1979.

[3] S. B. Fredj, T. Bonald, A. Proutiere, G. Régnié, and J. W. Roberts. Statistical Bandwidth Sharing: A Study of Congestion at Flow Level. *Computer Communication Review*, Vol. 31, No. 4, pp. 111-122, October 2001.

[4] L. Kleinrock. *Queueing Systems (Volume I: Theory)*. John Wiley & Sons, January 1975.

[5] S. G. M. Koo, C. Rosenberg, and D. Xu. Analysis of Parallel Downloading for Large File Distribution. *Proceedings of the* 9^{th} *IEEE International Workshop on Future Trends in Distributed Computing Systems (FTDCS 2003)*, San Juan, Puerto Rico, 28-30 May 2003.

[6] K.-C. Leung and V. O. K. Li. A Resequencing Model for High Speed Networks. *Proceedings of IEEE ICC '99*, Vol. 2, pp. 1239-1243, Vancouver, BC, Canada, 6-10 June 1999.

[7] K.-C. Leung and V. O. K. Li. Generalized Load Sharing for Packet-Switching Networks. *Proceedings of ICNP 2000*, pp. 305-314, Osaka, Japan, 14-17 November 2000.

[8] V. O. K. Li and W. Liao. Distributed Multimedia Systems. *Proceedings of the IEEE*, Vol. 85, No. 7, pp. 1063-1108, July 1997.

[9] N. F. Maxemchuk. Dispersity Routing in High-Speed Networks. *Computer Networks and ISDN Systems*, Vol. 25, No. 6, pp. 645-661, January 1993.

[10] T. S. E. Ng, Y.-H. Chu, S. G. Rao, K. Sripanidkulchai, and H. Zhang. Measurement-Based Optimization Techniques for Bandwidth-Demanding Peer-to-Peer Systems. *Proceedings of IEEE INFOCOM 2003*, San Francisco, CA, USA, 30 March - 3 April 2003.

[11] K. Obraczka and P. B. Danzig. Evaluating the Performance of Flood-d: A Tool for Efficiently Replicating Internet Information Services. *IEEE Journal on Selected Areas in Communications*, Vol. 16, No. 3, pp. 369-382, April 1998.

[12] W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling. *Numerical Recipes in C: The Art of Scientific Computing*. Second Edition. Cambridge University Press, January 1993.

[13] P. Rodriguez and E. W. Biersack. Dynamic Parallel Access to Replicated Content in the Internet. *IEEE/ACM Transactions on Networking*, Vol. 10, No. 4, pp. 455-465, August 2002.

[14] M. Sayal, Y. Breitbart, P. Scheuermann, and R. Vingralek. Selection Algorithms for Replicated Web Servers. *ACM Performance Evaluation Review*, Vol. 26, No. 3, pp. 44-50, December 1998.

[15] E. W. Zegura, M. H. Ammar, Z. Fei, and S. Bhattacharjee. Application-Layer Anycasting: A Server Selection Architecture and Use in a Replicated Web Service. *IEEE/ACM Transactions on Networking*, Vol. 8, No. 4, pp. 455-466, August 2000.