A Paracasting Model for Concurrent Access to Replicated Internet Content

Ka-Cheong Leung, *Member, IEEE,* and Victor O. K. Li*, Fellow, IEEE*

***Abstract*—In this paper, we develop a model to study how to effectively download a document from a set of replicated servers. We propose a generalized application-layer anycasting protocol, known as paracasting, 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 model 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 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 when a small number of servers, say, up to four, are used to serve a request concurrently.**

***Index Terms*—Content distribution, high speed networks, Internet, paracasting, performance modeling, server replication.**

I. Introduction

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 documents maintained at a nonreplicated server can attract tremendous access requests. 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 [13] 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 [18], which chooses the best replicated server for accessing a document is a nontrivial 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 [12], [15], [17], [18] rely on the estimates of the round-trip times between the client-server pairs and the server response times. Since the timing estimates are generally updated periodically and requests are only forwarded to the "best" server based on these estimates, such approaches do not spread these requests across a set of replicated servers. The server performance can therefore fluctuate dramatically due to significant load imbalance during a download session [16], 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, also known as all-server parallel downloading, has been proposed [5], [16]. The general idea is to satisfy a request by the involvement of all available servers. The client may download a certain portion of the desired document from each of the available servers, where the decisions of how much and which portion of the document to be downloaded from which replicated server are determined before the download begins [5]. Alternatively, a document may be partitioned into a large number of small blocks and block-based requests are forwarded to all replicated servers until all blocks are received [16]. A block request can be specified using the range request-header field defined in Hypertext Transfer Protocol (HTTP) Version 1.1 [3]. However, as discussed in [16], 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 receiver. 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 to use only a small number of paths, say up to three [8]. Moreover, the complexity and overhead for maintaining a large number of concurrent requests can be substantial [7].

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 question of how many replicated servers should be used to satisfy a client's request, so as to provide a better quality of service to users and an improved utilization for a given set of replicated servers. Therefore, the concept and model of such a generalized applica-

Manuscript received August 9, 2004; revised April 16, 2005. This work was supported in part by the Research Grants Council of the Hong Kong Special Administrative Region, China, under Grant No. HKU7044/02E. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Ryoichi Komiya.

The authors are with the Department of Electrical and Electronic Engineering, the University of Hong Kong, Hong Kong, China (e-mail: kcleung@ieee.org; vli@eee.hku.hk).

Digital Object Identifier 10.1109/TMM.2005.861288

[1]Multipath routing [1], [9], [11], where packets of a source may travel over multiple routes from a source to a destination, is a load balancing technique to distribute the traffic load across the network in order to alleviate network congestion.

tion-layer anycasting, known as *paracasting*, should be defined and developed.

A. Our Contributions

The focus of this work, first described in [10], is to develop a model to study the performance of application-layer paracasting for concurrent access to replicated content over a set of homogeneous servers. This model 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 behavior of parallel server access. To the best of our knowledge, there is only one analytical study to investigate all-server parallel downloading with homogeneous clients and servers [7]. All-server parallel downloading may not be necessary 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 utilized 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 introduces the idea of paracasting and gives a model for paracasting to conceptualize how replicated servers are utilized 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 model and studies the effectiveness of paracasting. Section V concludes and discusses some possible extensions to our work.

II. Paracasting Model

Application-layer anycasting [18] is a communication service implemented at the application layer so that a client or sender can choose and interact with a destination or server 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. As discussed in Section I, this reactive approach cannot help in spreading requests across a set of replicated servers. We would like to propose the concept of application-layer paracasting to further exploit 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 for performing each of these tasks. In other words, more than one destination within the paracast group may be involved in the communication activity, though the sender interacts with a single destination for carrying out a task of the given communication activity.

A paracast group is defined as a subset of all available destinations for carrying out the given communication activity. At one extreme, the paracast group consists of a single destination. The sender simply picks the destination to perform all tasks for the communication activity. This is the same as what the sender would do for application-layer anycasting. At the other extreme, the paracast group consists of all available destinations. The sender selects all destinations, each of which performs at least one task, for the communication activity. This is the same as what the sender would do for all-server parallel downloading. Hence, paracasting subsumes both application-layer anycasting and all-server parallel downloading.

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 re-assembling various parts of the file downloaded from these servers.

The scope of this work is to develop a simple model to study the performance of application-layer paracasting for concurrent access to replicated content over a set of homogeneous servers. This model can then be used to compare the performance of application-layer anycasting, paracasting, and all-server parallel downloading. The following assumptions are made to simplify the discussion.

1) There are N homogeneous replicated file servers in the system. Each server has a 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 $1/w$ 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 on the number of requests served concurrently by 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 mean $1/\lambda$ seconds.

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 average available bandwidth for file download from a participating server to the client is r bits per second, where $0 < r \leq C_c$. The value of r is determined from the model.
6) Signalling overheads are considered negligible in the system.

A client's request is considered blocked and lost whenever any one of its requested servers blocks its request. It may be more reasonable to try another server when blocked. The study of the performance impact due to retries of blocked requests is part of the future work.

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 in satisfying a download request. Moreover, active Transmission Control Protocol (TCP) flows tend to share the available bandwidth equally when these flows have the same or similar round-trip time (RTT) [4].

III. Queueing Analysis

In this section, we present the analytical results on 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 by λ_s the average request arrival rate (in requests per second) to each server in 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 each replicated server can be computed as follows:

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

This discussion will proceed as follows. Section III-A assumes the Markovian service rate for a request 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 average 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

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

When the service time conditioned on the number of requests being served is exponentially distributed, each replicated server can be modeled as an M/M/1/K/PS queue. It can be shown [6] that the steady-state probability of having i requests being served by a replicated server $p(i)$ can be written as

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

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

By Little's theorem [6], the average time (in seconds) to download a file from the set of replicated servers to a client can be determined as shown in (5), at the bottom of the page.

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 (which is at least 1) is as shown in (6), at the bottom of the next page.

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

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

Next, we evaluate r. Denote by $B_F(r)$ the normalized average aggregate download capacity used for file download when the average available bandwidth for file download from a participating server to the client is r bits per second. $B_F(r)$ falls between 0 and 1. Since the bottleneck download capacity is shared

$$T_F(r, S, C_c) = \frac{\sum_{i=1}^{K} i\, p(i)}{\lambda_s \cdot [1 - p(K)]} = \frac{S \cdot \left[\sum_{i=1}^{\min(m,K)} \frac{(\xi\rho)^i}{(i-1)!} + \sum_{i=\min(m,K)+1}^{K} \frac{i\xi^m \rho^i}{m!} \right]}{wC_s\rho \cdot \left[\sum_{i=0}^{\min(m,K-1)} \frac{(\xi\rho)^i}{i!} + \sum_{i=\min(m,K-1)+1}^{K-1} \frac{\xi^m \rho^i}{m!} \right]} \tag{5}$$

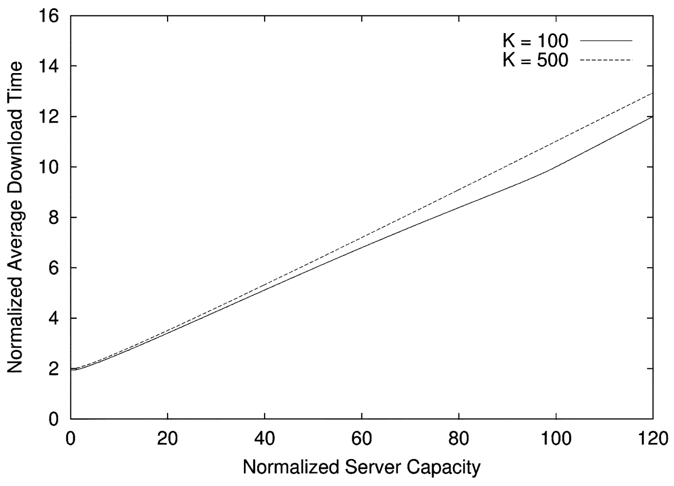

Fig. 1. Normalized average download time against normalized server capacity plot when $\rho = 0.95$, $\frac{C_s}{C_c} = 5$, and $w = 2$.

and consumed fairly by all participating servers to the client, the optimal value of r, r^*, can be determined by solving the following optimization problem:

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

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 average available bandwidth for file download from a participating server to the client. This, in turn, reduces the average aggregate file download capacity and thus increases the normalized average file 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 average 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. Furthermore, the captioned relationship holds 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 bottleneck download capacity is fully utilized, $B_F(r^*)$ attains the maximum value of 1. If all replicated servers are fully saturated, $B_F(r^*)$ is lower-bounded by $\frac{wC_s}{KC_c}$. In other words, the normalized average download time varies between 1 and $\frac{KC_c}{wC_s}$.

When $B_F(r)$ is monotonically increasing in r, there exists a unique solution to the aforementioned optimization problem as exhibited in (8). Numerical root finding techniques like the secant method [14] 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 average available bandwidth for file download from a participating replicated server to a client is r bits per second. When the service time is generally distributed and each server can serve any number of file download requests concurrently, each replicated server can be modeled as an M/G/1/∞/PS queue. It can be shown [2], [4] that the average time (in seconds) to download a file from the set of replicated servers to a client is as follows:

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

where $0 \le \rho < 1$ and

$$f(\rho) = \frac{\frac{(\xi\rho)^m}{m!}}{(1-\rho) \sum_{i=0}^{m-1} \frac{(\xi\rho)^i}{i!} + \frac{(\xi\rho)^m}{m!}} \tag{10}$$

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 (which is at least 1) is

$$\tilde{T}_I(r) = \frac{C_c}{wC_s} \cdot \left\{ \xi + \frac{f(\rho)}{1-\rho} \cdot [1 - (\xi - m) \cdot (1 - \rho)] \right\}. \tag{11}$$

Next, we evaluate r. Denote by $B_I(r)$ the normalized average aggregate download capacity used for file download when the average available bandwidth for file download from a participating server to the client is r bits per second. $B_I(r)$ falls between 0 and 1. The optimal value of r, r^*, can be determined by solving the following optimization problem similar to (8) in Section III-A:

$$\begin{aligned} \text{Minimize} \quad & |B_I(r) - 1| \\ \text{subject to} \quad & 0 < r \le C_c \end{aligned} \tag{12}$$

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

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

$$\tilde{T}_F(r) = \frac{C_c T_F(r, S, C_c)}{S} = \frac{C_c \cdot \left[\sum_{i=1}^{\min(m,K)} \frac{(\xi\rho)^i}{(i-1)!} + \sum_{i=\min(m,K)+1}^{K} \frac{i\xi^m \rho^i}{m!} \right]}{wC_s\rho \cdot \left[\sum_{i=0}^{\min(m,K-1)} \frac{(\xi\rho)^i}{i!} + \sum_{i=\min(m,K-1)+1}^{K-1} \frac{\xi^m \rho^i}{m!} \right]}. \tag{6}$$

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

Proof: From (10), differentiating $f(\rho)$ with respect to ξ

$$\begin{aligned}\frac{\partial f}{\partial \xi} &= \frac{\frac{m\rho \cdot (\xi\rho)^{m-1}}{m!}}{(1-\rho)\sum_{i=0}^{m-1}\frac{(\xi\rho)^i}{i!} + \frac{(\xi\rho)^m}{m!}} \\ &\quad - \frac{\frac{(\xi\rho)^m}{m!} \cdot \left[(1-\rho)\sum_{i=1}^{m-1}\frac{i\rho\cdot(\xi\rho)^{i-1}}{i!} + \frac{m\rho\cdot(\xi\rho)^{m-1}}{m!}\right]}{[(1-\rho)\sum_{i=0}^{m-1}\frac{(\xi\rho)^i}{i!} + \frac{(\xi\rho)^m}{m!}]^2} \\ &= \left(\frac{m}{\xi} - \rho\right) \cdot f(\rho) + \left(1 - \frac{m}{\xi}\right) \cdot \rho \cdot [f(\rho)]^2. \qquad (13)\end{aligned}$$

From (11), differentiating $\tilde{T}_I(r)$ with respect to ξ, we have (14) , shown at the bottom of the page, since $\xi \cdot (1-\rho) \geq (\xi - m) \cdot f(\rho) \cdot \{[1+(1-\rho)\cdot(m-\xi\rho)] - \rho \cdot [1-(1-\rho)\cdot(\xi-m)]\cdot f(\rho)\}$.

Differentiating $B_I(r)$ with respect to r

$$\begin{aligned}\frac{\partial B_I}{\partial r} &= \frac{\partial B_I}{\partial \tilde{T}_I} \cdot \frac{\partial \tilde{T}_I}{\partial \xi} \cdot \frac{\partial \xi}{\partial r} \\ &= \frac{C_s}{(r\tilde{T}_I)^2} \cdot \frac{\partial \tilde{T}_I}{\partial \xi} \\ &\geq 0 \qquad (15)\end{aligned}$$

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 $\xi \cdot (1-\rho) \geq (\xi - m) \cdot f(\rho) \cdot \{[1 + (1-\rho) \cdot (m - \xi\rho)] - \rho \cdot [1 - (1-\rho) \cdot (\xi - m)] \cdot f(\rho)\}$ generally holds for a sufficiently large ξ as $0 \leq \xi - m < 1$ and $0 \leq f(\rho) \ll \rho < 1$. 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 $0 < r \leq C_c$.

Proof: From (10), taking $r = \gamma^+ = \frac{C_s}{m}$,

$$\begin{aligned}f(\rho)|_{r=\gamma^+} &= \frac{\frac{(\xi\rho)^{m-1}}{(m-1)!}}{(1-\rho)\sum_{i=0}^{m-2}\frac{(\xi\rho)^i}{i!} + \frac{(\xi\rho)^{m-1}}{(m-1)!}} \\ &= \frac{\frac{m}{\xi\rho} \cdot \frac{(\xi\rho)^m}{m!}}{(1-\rho)\sum_{i=0}^{m-1}\frac{(\xi\rho)^i}{i!} + \frac{m}{\xi} \cdot \frac{(\xi\rho)^m}{m!}} \\ &= \frac{1}{\rho} \cdot f(\rho)|_{r=\gamma} = \frac{1}{\rho} \cdot f(\rho)|_{r=\gamma^-} \qquad (16)\end{aligned}$$

for any positive integer m.

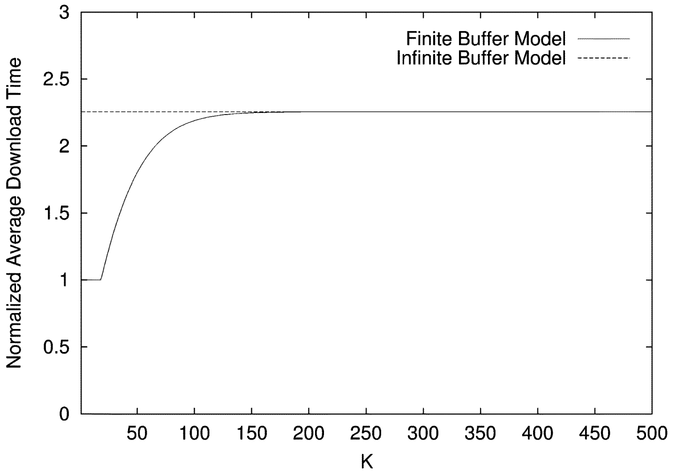

Fig. 2. Normalized average download time against K plot when $\rho = 0.95$, $\frac{C_s}{C_c} = 5$, and $w = 2$.

Hence

$$\begin{aligned}B_I(\gamma^+) &= \frac{wC_s}{C_c} \\ &\quad \cdot \frac{1}{m + \frac{f(\rho)|_{r=\gamma^+}}{1-\rho} \cdot \{1 - [m - (m-1)] \cdot (1-\rho)\}} \\ &= B_I(\gamma) = B_I(\gamma^-). \qquad (17)\end{aligned}$$

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

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

Proof: The result follows directly from Lemmas 1 and 2. ■

When $B_I(r)$ is monotonically increasing in r, there exists a unique solution to the aforementioned optimization problem as stated in (12). Numerical root finding techniques like the secant method [14] 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,

$$\begin{aligned}\frac{\partial \tilde{T}_I}{\partial \xi} &= \frac{C_c}{wC_s} \cdot \left[1 + \frac{1}{1-\rho} \cdot \frac{\partial f}{\partial \xi} + (m - \xi) \cdot \frac{\partial f}{\partial \xi} - f(\rho)\right] \\ &= \frac{C_c\{\xi \cdot (1-\rho) + (m-\xi) \cdot [1 + (1-\rho) \cdot (m - \xi\rho)] \cdot f(\rho)\}}{wC_s\xi \cdot (1-\rho)} \\ &\quad + \frac{C_c \cdot (\xi - m) \cdot \rho \cdot [1 + (1-\rho) \cdot (m - \xi)] \cdot [f(\rho)]^2}{wC_s\xi \cdot (1-\rho)} \\ &\geq 0 \qquad (14)\end{aligned}$$

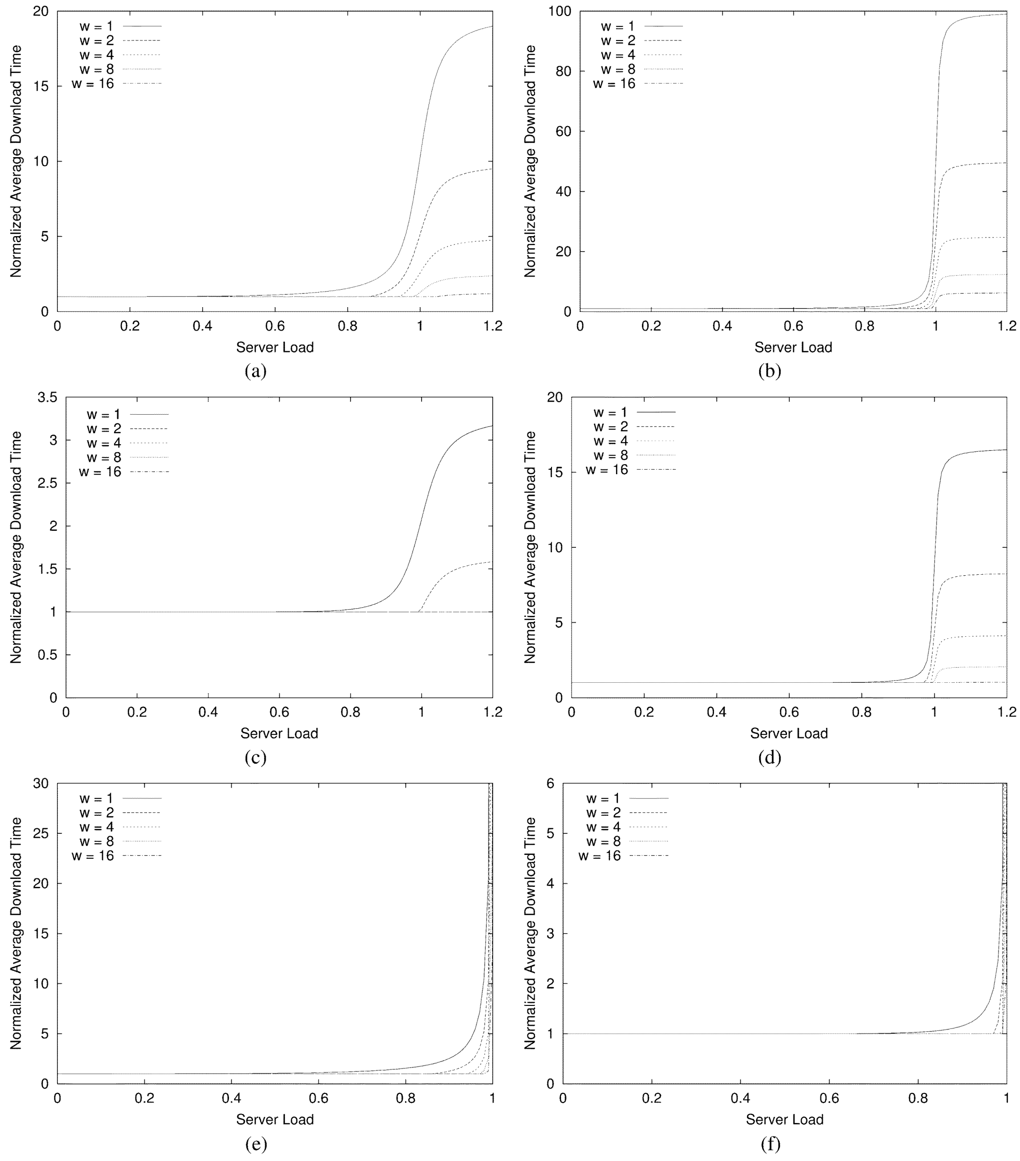

Fig. 3. Normalized average download time against server load plots for various settings. (a) $\frac{C_s}{C_c} = 5$ and $K = 100$. (b) $\frac{C_s}{C_c} = 5$ and $K = 500$. (c) $\frac{C_s}{C_c} = 30$ and $K = 100$. (d) $\frac{C_s}{C_c} = 30$ and $K = 500$. (e) $\frac{C_s}{C_c} = 5$ and $K \to \infty$. (f) $\frac{C_s}{C_c} = 30$ and $K \to \infty$.

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 with a cable modem connection with download speeds of 9 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 *on* $\tilde{T}_{\bullet}(r)$

The effect of the maximum number of concurrent requests a replicated server can handle, K, on 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 download capacities at the clients. Thus, the file download time is determined by the incoming bottleneck bandwidth to the client.

When K is greater than a certain value (i.e., 18), 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

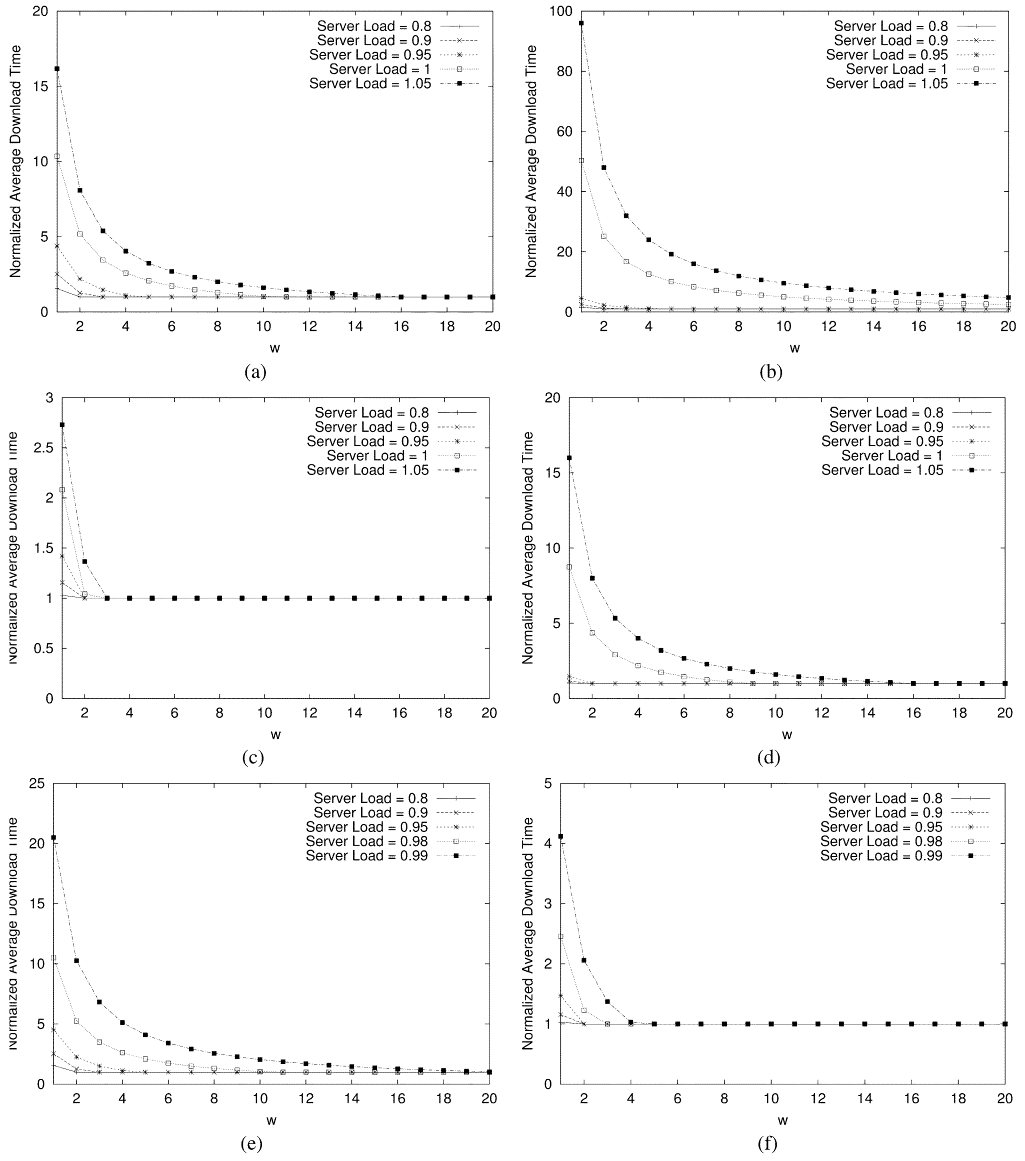

Fig. 4. Normalized average download time against w plots for various settings. (a) $\frac{C_s}{C_c} = 5$ and $K = 100$. (b) $\frac{C_s}{C_c} = 5$ and $K = 500$. (c) $\frac{C_s}{C_c} = 30$ and $K = 100$. (d) $\frac{C_s}{C_c} = 30$ and $K = 500$. (e) $\frac{C_s}{C_c} = 5$ and $K \to \infty$. (f) $\frac{C_s}{C_c} = 30$ and $K \to \infty$.

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 the infinite buffer model (where K tends to infinity) as they converge to the same download time when K is sufficiently large.

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 greater). When K is finite and the server load exceeds 1, the normalized average download time flattens and is upper-bounded by $\frac{KC_c}{wC_s}$, 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, 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 choosing an appropriate value of w to minimize the nor-

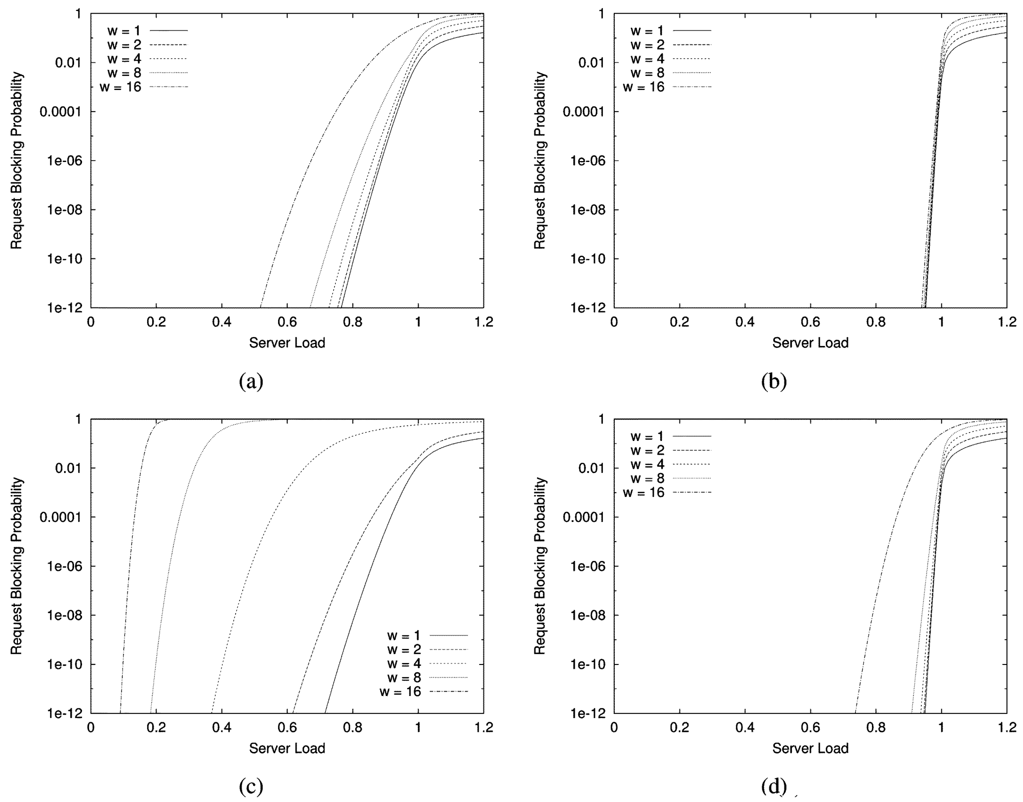

Fig. 5. Request blocking probability against server load plots for various settings. (a) $\frac{C_s}{C_c} = 5$ and $K = 100$. (b) $\frac{C_s}{C_c} = 5$ and $K = 500$. (c) $\frac{C_s}{C_c} = 30$ and $K = 100$. (d) $\frac{C_s}{C_c} = 30$ and $K = 500$.

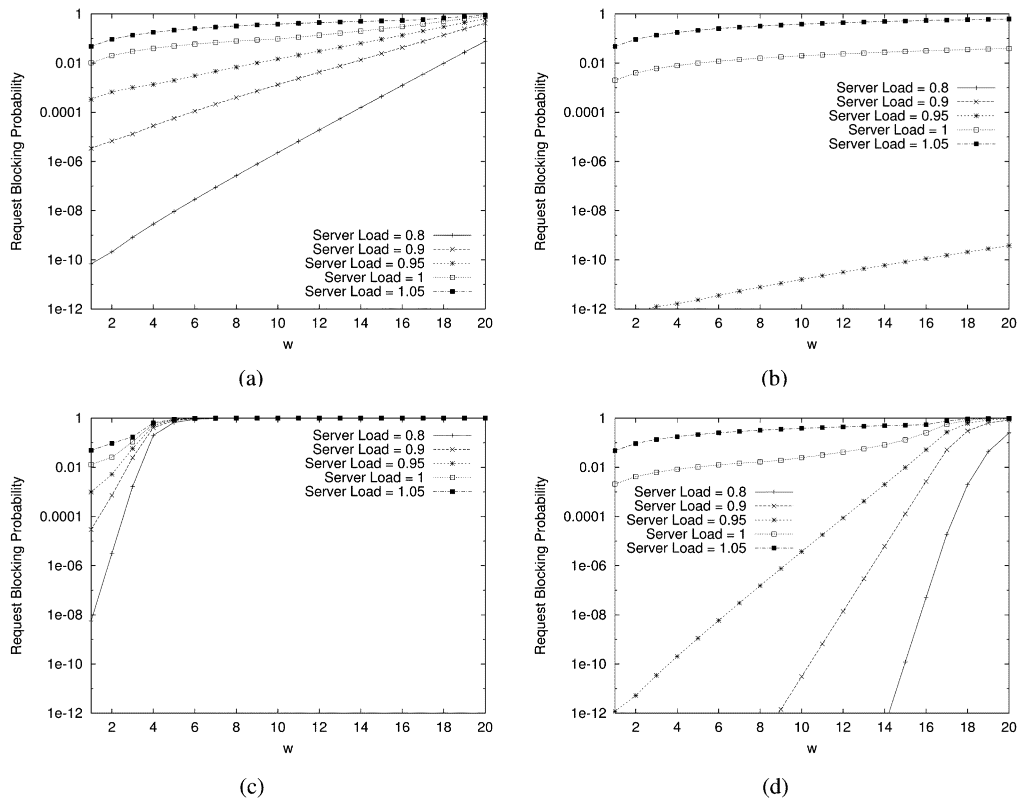

Fig. 6. Request blocking probability against w plots for various settings. (a) $\frac{C_s}{C_c} = 5$ and $K = 100$. (b) $\frac{C_s}{C_c} = 5$ and $K = 500$. (c) $\frac{C_s}{C_c} = 30$ and $K = 100$. (d) $\frac{C_s}{C_c} = 30$ and $K = 500$.

malized average download time while keeping the system cost, such as the request blocking probability, to an acceptable level.

For example, suppose the size of the file to be downloaded is 1.25 MB (i.e., $S = 10^7$) and the maximum aggregate download

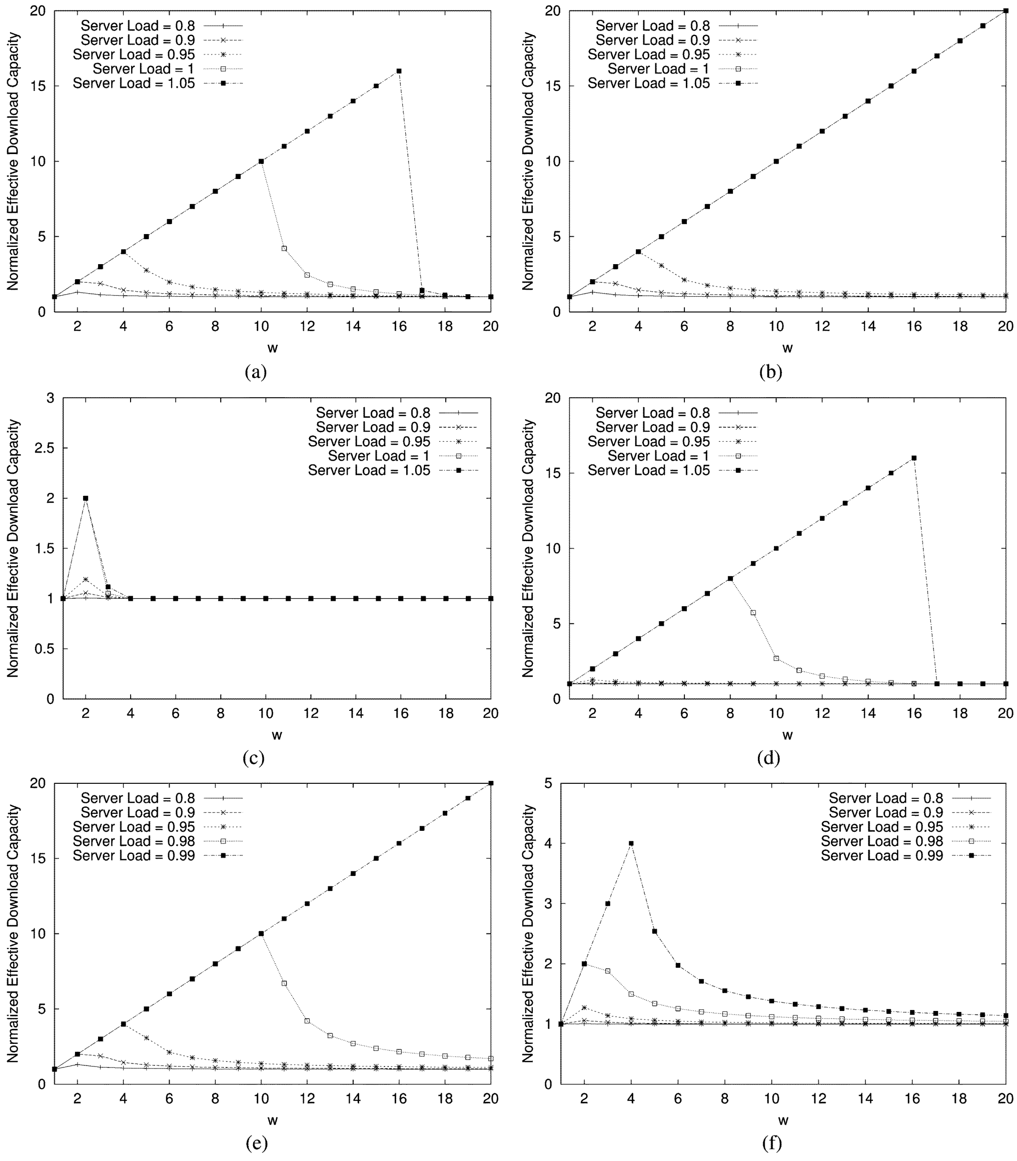

Fig. 7. Normalized effective download capacity against w plots for various settings. (a) $\frac{C_s}{C_c} = 5$ and $K = 100$. (b) $\frac{C_s}{C_c} = 5$ and $K = 500$. (c) $\frac{C_s}{C_c} = 30$ and $K = 100$. (d) $\frac{C_s}{C_c} = 30$ and $K = 500$. (e) $\frac{C_s}{C_c} = 5$ and $K \to \infty$. (f) $\frac{C_s}{C_c} = 30$ and $K \to \infty$.

capacity is 10 Mbps (i.e., $C_c = 10^7$). When $\frac{C_s}{C_c} = 5$, $K = 500$, and the server load is 1, the average download times are 16.8 s when $w = 3$ and 12.6 s when $w = 4$. Thus, at least four servers are needed if a client wants to download the file within 15 s.

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

Figs. 5 and 6 exhibit the impact of the server load and w on the request blocking probability when K is finite. The request blocking probability increases from 0 toward 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 favor using a large w, say more than four, for paracasting. For example, when $\frac{C_s}{C_c} = 30$, $K = 500$, and the server load is 1, the request blocking probability is less than 0.01 provided that w is less than or equal to four. 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 $E_\bullet(r)$, ρ, and w

The relationship between the normalized effective download capacity and w is shown in Fig. 7. The normalized effective download capacity $E_\bullet(r)$, which is defined as $\frac{wr}{C_c}$, represents the sum of all perceived available 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 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 of others, 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, no further reduction on the download time is achieved by increasing w further since the bottleneck bandwidth is fairly shared and consumed among the servers for the file download. Thus, to improve the file download time, this argument does not favor 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 introduced the concept of generalized application-layer anycasting, known as paracasting, for concurrent access to replicated content over a set of replicated servers. By paracasting, a sender can choose a subset of all available destinations (defined as a paracast group) to cooperatively perform a communication activity, such as downloading a document.

We have also developed a model to study the performance of paracasting. This model 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. It conceptualizes how replicated servers are utilized concurrently to satisfy a download request. The queueing analysis of 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 when a small number of servers, say, up to four, are used to serve a request concurrently.

Now we revisit 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. Using an optimal number of servers (generally as small as up to four) ensures that the server and network loads are well balanced. Paracasting also helps to reduce the average file download time by effectively using the available network bandwidth from the servers to the clients.

However, paracasting requires 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 fragments to recover the file. Moreover, the replicated servers must be capable of downloading a selected portion of a file to a client. Indeed, a block request can be specified using the range request-header field defined in HTTP Version 1.1 [3].

In this paper, we assume that a subset of servers is selected randomly from the server pool. Each of the selected servers is responsible to download an equal portion of a document to the client. A more sophisticated server selection and load distribution algorithm will be devised as part of the future work.

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, and a client can try another server when its request is blocked;
- 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.

Acknowledgment

The authors would like to thank the anonymous reviewers for their valuable comments and suggestions which assisted us in improving the quality of the paper.

References

[1] S. N. Chiou and V. O. K. Li, "Diversity transmissions in a communication network with unreliable components," in *Proc. IEEE ICC '87*, vol. 2, Seattle, WA, Jun. 7–10, 1987, pp. 968–973.

[2] J. W. Cohen, "The multiple phase service network with generalized processor sharing," *Acta Informatica*, vol. 12, pp. 245–284, 1979.

[3] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee, "Hypertext transfer protocol—HTTP/1.1," in *Request for Comments, RFC 2616*, Jun. 1999.

[4] S. B. Fredj, T. Bonald, A. Proutiere, G. Régnié, and J. W. Roberts, "Statistical bandwidth sharing: A study of congestion at flow level," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 31, no. 4, pp. 111–122, Oct. 2001.

[5] C. Gkantsidis, M. Ammar, and E. Zegura, "On the effect of large-scale deployment of parallel downloading," in *Pro. 3rd IEEE Workshop on Internet Applications (WIAPP '03)*, San Jose, CA, Jun. 23–24, 2003, pp. 79–89.

[6] L. Kleinrock, *Queueing Systems (Volume I: Theory).* New York: Wiley, 1975.

[7] S. G. M. Koo, C. Rosenberg, and D. Xu, "Analysis of parallel downloading for large file distribution," in *Proc. 9th IEEE Int. Workshop on Future Trends in Distributed Computing Systems (FTDCS 2003)*, San Juan, PR, May 28–30, 2003, pp. 128–135.

[8] K.-C Leung and V. O. K. Li, "A resequencing model for high speed networks," in *Proc. IEEE ICC '99*, vol. 2, Vancouver, BC, Canada, Jun. 6–10, 1999, pp. 1239–1243.

[9] ——, "Generalized load sharing for packet-switching networks," in *Proc. IEEE ICNP 2000*, Osaka, Japan, Nov. 14–17, 2000, pp. 305–314.

[10] ——, "A paracasting model for concurrent access to replicated content," in *Proc. IEEE CCW 2003*, Dana Point, CA, Oct. 20–21, 2003, pp. 105–111.

[11] N. F. Maxemchuk, "Dispersity routing in high-speed networks," *Comput. Netw. and ISDN Syst.*, vol. 25, no. 6, pp. 645–661, Jan. 1993.

[12] 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," in *Proc. IEEE INFOCOM 2003*, vol. 3, San Francisco, CA, Mar.–Apr. 30–3, 2003, pp. 2199–2209.

[13] K. Obraczka and P. B. Danzig, "Evaluating the performance of flood-d: A tool for efficiently replicating internet information services," *IEEE J. Select. Areas Commun.*, vol. 16, pp. 369–382, Apr. 1998.

[14] W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling, *Numerical Recipes in C: The Art of Scientific Computing*, 2nd ed. Cambridge, MA: Cambridge Univ. Press, 1993.

[15] S. Ranjan, R. Karrer, and E. Knightly, "Wide area redirection of dynamic content by internet data centers," in *Proc. IEEE INFOCOM 2004*, vol. 2, Hong Kong, China, Mar. 7–11, 2004, pp. 816–826.

[16] P. Rodriguez and E. W. Biersack, "Dynamic parallel access to replicated content in the internet," *IEEE/ACM Trans. Netw.*, vol. 10, no. 4, pp. 455–465, Aug. 2002.

[17] M. Sayal, Y. Breitbart, P. Scheuermann, and R. Vingralek, "Selection algorithms for replicated web servers," *ACM SIGMETRICS Perform. Eval. Rev.*, vol. 26, no. 3, pp. 44–50, Dec. 1998.

[18] 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 Trans. Netw.*, vol. 8, no. 4, pp. 455–466, Aug. 2000.

Ka-Cheong Leung (S'95–M'01) was born in Hong Kong in 1972. He received the B.Eng. degree in computer science from the Hong Kong University of Science and Technology in 1994, the M.Sc. degree in electrical engineering (computer networks) and the Ph.D. degree in computer engineering from the University of Southern California, Los Angeles, in 1997 and 2000, respectively.

He worked as a Senior Research Engineer at Nokia Research Center, Nokia, Inc., Irving, TX, from 2001 to 2002. He was Assistant Professor at the Department of Computer Science at Texas Tech University, Lubbock, between 2002 and 2005. Since June 2005, he has been with the University of Hong Kong, where he is Visiting Assistant Professor with the Department of Electrical and Electronic Engineering. His research interests include wireless packet scheduling, routing, congestion control, and quality of service guarantees in high-speed communication networks, content distribution, high-performance computing, and parallel applications.

Dr. Leung is listed in the 60th (2006) Edition of *Marquis Who's Who in America*.

Victor O. K. Li (S'80–M'81–SM'86–F'92) was born in Hong Kong in 1954. He received the B.S., M.S., E.E. and D.Sc. degrees in electrical engineering and computer science from the Massachusetts Institute of Technology, Cambridge, in 1977, 1979, 1980, and 1981, respectively.

He joined the University of Southern California (USC), Los Angeles, in February 1981, and became a Professor of Electrical Engineering and Director of the USC Communication Sciences Institute. Since September 1997, he has been with the University of Hong Kong, where he is Chair Professor of Information Engineering. He served as Managing Director of Versitech Ltd., the technology transfer and commercial arm of the University of Hong Kong, from September 1997 to June 2004 and currently serves on various corporate boards. His research interests are in the area of information technology, including all-optical networks, wireless networks, and Internet technologies and applications. Sought by government, industry, and academic organizations, he has lectured and consulted extensively around the world. He is very active in the research community, and has chaired various international conferences and served on the editorial boards of various international journals, has given distinguished lectures at universities around the world, and keynote speeches at many international conferences.

Dr. Li has received numerous awards, including, most recently, the U.K. Royal Academy of Engineering Senior Visiting Fellowship in Communications, the KC Wong Education Foundation Lectureship, the Croucher Foundation Senior Research Fellowship, and the Bronze Bauhinia Star, Government of the Hong Kong Special Administrative Region, China.