Analysis of Parallel Downloading for Large File Distribution*

Simon G. M. Koo, Catherine Rosenberg
School of Electrical and Computer Engineering
Purdue University, West Lafayette, IN 47907
{koo, cath}@ecn.purdue.edu

Dongyan Xu
Department of Computer Science
Purdue University, West Lafayette, IN 47907
dxu@cs.purdue.edu

Abstract

Recently, the scheme of parallel downloading (PD) has been adopted by a number of Internet file downloading applications. With the wide deployment of content distribution networks and peer-to-peer networks, PD is expected to be more commonly used for file distribution. There have been experiments showing that PD results in higher aggregated downloading throughput and therefore shorter downloading time experienced by clients. However, these experimental studies focused on the performance experienced by a particular user and did not consider the impact of PD on the network when it is largely deployed. In this paper we present our efforts toward an in-depth understanding of large-scale deployment of PD through simulation and analysis. Our results suggest that while PD may achieve a shorter downloading time, its impact on the network and server is significant. Our analysis is also used for network dimensioning and content distribution service provisioning. We show that with proper admission control and dimensioning, single-server downloading can perform just as well as PD, without the complexity and overhead incurred by PD.

Keywords – Parallel Downloading, CDN, Large File Distribution, Network Dimensioning.

1 Introduction

Recently, the scheme of Parallel Downloading (PD) has been adopted by a number of Internet file downloading applications. Under the PD scheme, a client requesting a file will open concurrent connections to multiple senders, which can be servers or peers, and different parts of the file will be transmitted from the senders to the client. There have been experiments showing that PD results in higher aggregated downloading throughput and therefore shorter downloading time experienced by the clients [11, 12] at the expenses of more signalling due to the need to coordinate the servers. With the wide deployment of content distribution networks (CDNs) and peer-to-peer (P2P) networks, PD is expected to be more commonly adopted for file distribution, especially when the volume of a file is large. The introduction of Tornado Code [1, 2] popularizes the use of PD, as Tornado Code reduces the requirement for coordination among servers (and hence for signalling).

Previous work on parallel downloading has focused on downloading time experienced by an *individual* client. There has been no performance study on parallel downloading when it is used by a large number of clients. As a result, we cannot conclude if PD is a good scheme based on previous work. In fact, there are two common but conflicting views: the first view is that parallel downloading is generally effective because it speeds up downloading by exploiting the servers' capacity in a more balanced fashion. However, the other opinion is that parallel downloading is no better, if not worse, than the single-server downloading scheme, because the downloading time reduction will become less significant if *every* client chooses to perform PD. It is also not clear what is the trade-off between average downloading time experienced by each client and the overall request blocking rate. Unfortunately, previous experimental studies did not answer these fundamental questions.

We believe that to answer these questions, rigorous analytical models are needed, which capture the relation between key system parameters and performance metrics. In this paper we present our efforts toward an in-depth understanding of PD, especially in comparison with traditional downloading from single server (SD). We propose analytical models, which will be applied to guide decisions in content distribution service provisioning – for example, we want to answer the following questions: Which scheme to adopt (PD or SD)? How many servers should be used? How many concurrent sessions should be allowed per server? How large should be the capacity of the link connecting a server to the network?

In the following section we will review previous works

*This work has been supported in part by a grant from the e-Enterprise Center at Purdue University.

on PD. We present our analytical models and compare results from our model to simulation results in Section 3. In Section 4, we apply our analytical models to practical network design problems including network dimensioning and admission control. Finally, Section 5 concludes the paper.

2 Related Works

The idea of parallel access to multiple CDN servers has been proposed in the recent literature as an empirical technique. In [12], Rodriguez and Biersack study a dynamic parallel-access scheme to access multiple mirror servers. In their study, a client downloads files from mirror servers residing in a wide area network. They show that their dynamic parallel downloading scheme achieves significant downloading speedup with respect to a single server scheme. However, they only study the scenario where *one* client uses parallel downloading. The authors fail to address the effect and consequences when all clients choose to adopt the same scheme. Other works [8, 11, 16] on parallel downloading provide experimental results on the performance gain from an *individual* client's standpoint, without addressing the impact if *all* clients use parallel downloading. In addition, their results are drawn from experiments under specific system setup and parameters. In our study, we will investigate the effectiveness of parallel downloading assuming wide adoption of the scheme. Furthermore, we will perform rigorous analysis to compare the performance of parallel downloading and traditional downloading from a single server.

In order to perform parallel access to multiple servers, the client must either determine the part of the document it needs from each of the servers, or encode the document such that it requires less synchronization between client and servers. Nebat and Sidi in [10] consider the scenario where `packets' are requested from multiple servers, and analyze the amount of client-side buffer required to handle the parallelism. Byers *et al.* propose a new coding scheme, Tornado Code [1, 2], which works like FEC but with a lower computation complexity to minimize the need for coordination between servers. The advantage of using Tornado Code is that clients receiving data from multiple servers do not need complicated signaling to determine how much and what data to expect from each server contrary to [12]. A client can simply tell the servers to stop once it receives `enough' encoded data segments to reconstruct the file. The down side of Tornado Code is that client must buffer all the data before being able to reconstruct the file. All works suggest that parallel downloading incurs non-trivial overhead (either synchronization overhead or coding/decoding overhead), and that parallel downloading should be adopted only if it improves the downloading performance significantly. We will show in Section 4 that with a properly dimensioned CDN, parallel downloading is *not* the clear winner in terms of performance, overhead, and resource consumption.

Notation	Definition
C_{in}	In-bound bandwidth of each client.
C_{out}	Outgoing bandwidth of each server.
λ	Arrival rate of file requests.
F	Average size of files being distributed.
K	Number of servers.
N	Limit on the number of simultaneous connections per server.
$\overline{D}$	Average downloading time achieved by the system
P_B	Blocking rate achieved by the system

Table 1. Definitions of system parameters in the system model

3 Models for Downloading Systems

In this section, we derive an abstraction for modelling parallel downloading. We want to understand how the key system parameters impact the system performance, including the average file downloading time, and the downloading request blocking rate. The system parameters are listed in Table 1. We will conduct a comparison study on PD and SD based on these models. The analytical results will help determining the optimal tunable parameters and the expected performance under different system configurations.

Before describing our models, we will first state the assumptions about the file downloading system. We considered a homogeneous environment in which each client has an incoming capacity of C_{in} and each of the K servers has an outgoing capacity of C_{out}, where $C_{out} \gg C_{in}$. We assume that each server has its outgoing capacity equally shared by all active connections at any time, i.e., processor sharing is used. The intermediate network is assumed to be bottleneck-free. The bottleneck is either the server's outgoing or the client's incoming link. We also assume that the arrival rate of requests follows a Poisson process and that a client has at most one download going on at a given time, i.e., the population of clients is infinite.

For a parallel downloading scheme using K servers, the client is requesting $1/K$ of the file from each of the K servers. This scheme is not in general the most efficient way to download a file from multiple servers, but it allows us to come up with closed-form results. In the homogeneous case, it is equivalent to the scheme where the K servers use Tornado code and send data till they receive a stop message from the client. For SD, the server selection policy used in

the study is round-robin. It can be shown that, under our homogeneous network model, round-robin yields the minimum downloading time by selecting the server with minimum remaining work. For PD, a new client is connected to all K servers. For both schemes, each server has a limit on the number of concurrent connections N. A request for a download session is considered blocked in SD if each of the K servers has N active connections. For PD, if a request arrives when N clients are in the system (each of them connected to all K servers), the request is blocked. Any blocked request is considered lost. There is no retry mechanism and we assumed that the time for admission process is negligible.

We will compare PD and SD in terms of downloading time and request blocking rate. It is important to note that PD is significantly more difficult to manage (even if Tornado code is used) and hence its use would only make sense if PD yields much better performances than SD.

In [7] and [13] it has been shown that the mean response time of a processor-sharing system with Poisson arrivals is *independent* of the service time distribution and only depends on the mean service time. This result gives us freedom to choose *any* distribution for the file size (as long as it has a finite and deterministic mean). We have chosen the distribution to be exponential with mean size F to facilitate our analysis. The average downloading time is defined to be the average time between a request is made (and accepted) and that particular request is finished. The blocking rate is defined to be the ratio of rejected requests to the total number of requests. These two metrics will be the major performance metrics of interest throughout the paper.

Under these assumptions, we can use birth-death processes to model the two schemes. For SD, we will model the system as K identical queues, each being a M/G/1/N/PS with an arrival rate of λ/K and each request will receive an exponentially distributed amount of data with mean size F before leaving the system. Similarly, PD is also modelled by K identical queues, except the arrival rate for each queue is λ and each request will require an exponentially distributed amount of data with mean F/K. We will only study one of the K queues for both SD and PD. The detailed arrival/service rates and stationary distribution of these birth-death processes, as well as the closed-form expressions for average downloading times ($\overline{D}$) and blocking rates (P_B) are derived as follows. We denote λ_i and μ_i as the birth rate and death rate respectively out of state i. The system is in state i if the number of sessions in the queue under study is i. When i clients share a server's outgoing link, each of them will get $1/i$ of the link's capacity, or the incoming capacities of their own, whichever smaller. This is equivalent to say that when the systems in state i, each client will receive $min\{C_{in}, \frac{C_{out}}{i}\}$ amount of capacity.

For SD:

$$\begin{cases} \lambda_i = \frac{\lambda}{K} \\ \mu_i = \frac{i \times min\{C_{in}, \frac{C_{out}}{i}\}}{F} \end{cases}$$

$$P_i = (\frac{\lambda}{K})^i (\frac{1}{\prod_{j=1}^{i} \mu_j}) P_0 \quad (1)$$

Since $\sum_{i=0}^{N} P_i = 1$, we have

$$P_0 = (1 + \sum_{i=1}^{N} (\frac{\lambda}{K})^i (\frac{1}{\prod_{j=1}^{i} \mu_j}))^{-1}$$

Therefore, the average downloading time of SD will be

$$\overline{D} = \frac{\sum_{i=0}^{N} i P_i}{\lambda(1 - P_N)} \quad (2)$$

with P_i's from (1), and the blocking rate of SD, which is equal to P_N, will be

$$P_B = \frac{(\frac{\lambda}{K})^N (\frac{1}{\prod_{j=1}^{N} \mu_j})}{1 + \sum_{i=1}^{N} (\frac{\lambda}{K})^i (\frac{1}{\prod_{j=1}^{i} \mu_j})} \quad (3)$$

Similarly, for PD:

$$\begin{cases} \lambda_i = \lambda \\ \mu_i = \begin{cases} \frac{i \times C_{in}}{F} & \text{if } 1 \le i \le \eta \\ \frac{K \times C_{out}}{F} & \text{if } \eta \le i \le N \end{cases} \end{cases}$$

where $\eta = \frac{K \times C_{out}}{C_{in}}$, we have

$$P_i = \begin{cases} (\frac{1}{i!})(\frac{\lambda F}{C_{in}})^i P_0 & \text{if } 0 \le i \le \eta \\ (\frac{1}{\eta!})(\frac{\lambda F}{C_{in}})^\eta (\frac{\lambda F}{K C_{out}})^{i-\eta} P_0 & \text{if } \eta < i \le N \\ 0 & \text{otherwise} \end{cases} \quad (4)$$

$$\Rightarrow P_0 = (\sum_{i=0}^{\eta} (\frac{1}{i!})(\frac{\lambda F}{C_{in}})^i + \sum_{i=\eta+1}^{N} (\frac{1}{\eta!})(\frac{\lambda F}{C_{in}})^\eta (\frac{\lambda F}{K C_{out}})^{i-\eta})^{-1}$$

Therefore, the average downloading time for PD will be

$$\overline{D} = \frac{\sum_{i=0}^{N} i P_i}{\lambda(1 - P_N)} \quad (5)$$

with P_i's from (4), and the blocking rate for PD will be

$$P_B = \frac{(\frac{1}{\eta!})(\frac{\lambda F}{C_{in}})^\eta (\frac{\lambda F}{K C_{out}})^{N-\eta}}{\sum_{i=0}^{\eta} (\frac{1}{i!})(\frac{\lambda F}{C_{in}})^i + \sum_{i=\eta+1}^{N} (\frac{1}{\eta!})(\frac{\lambda F}{C_{in}})^\eta (\frac{\lambda F}{K C_{out}})^{i-\eta}} \quad (6)$$

It can be shown that for the same set of parameters, the blocking rate of SD is never higher than that of PD, based

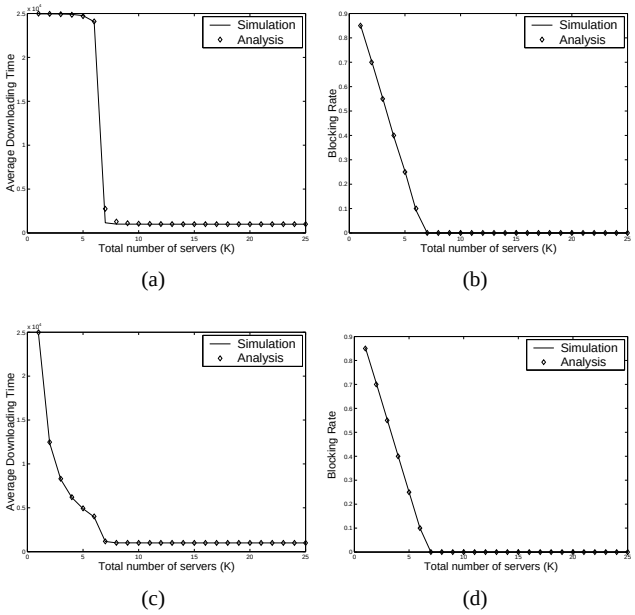

Figure 1. Comparison between simulation and analysis for (a) SD on $\overline{D}$; (b) SD on P_B; (c) PD on $\overline{D}$; and (d) PD on P_B

on equations (3) and (6); and that the average downloading time of PD is never larger than that of SD, based on equations (2) and (5).

We perform simulations on SD and PD using a fixed file size F (to check the insensitivity of the results to the file size distribution) with the following network parameters: $C_{in} = 100$kbps, $C_{out} = 10$Mbps, $F = 125$MBytes, $N = 750$, $\lambda = \frac{1}{15}$ requests/second and K varying. The simulation results, which are shown in Figure 1, match our analysis very well for both SD and PD, and confirm the insensitivity to the file size distribution and the preciseness of our model.

There are some fundamental properties derived from the analytical models that are noteworthy. The following propositions summarize the observations:

Proposition 3.1 *The minimum downloading time achievable by each scheme is F/C_{in}, and a sufficient condition to achieve this minimum is when $N \leq C_{out}/C_{in}$ for SD and $N \leq KC_{out}/C_{in}$ for PD.*

Proof. The inbound capacity of each client is C_{in}, so the minimum time to download F will be F/C_{in}. For SD, when x clients are connected to a server, the capacity that each client will receive is $\min\{C_{in}, \frac{C_{out}}{x}\}$, and $\frac{C_{out}}{x}$ is minimum when $x = N$. Therefore, if

$$C_{in} \leq \frac{C_{out}}{N}$$

$$\Rightarrow N \leq C_{out}/C_{in}$$

each client will receive exactly C_{in} amount of capacity from the server, and be able to download the file in F/C_{in} time. Similarly for PD, when $N \leq KC_{out}/C_{in}$, each client will receive C_{in} amount of capacity from the server and experience minimum downloading time. ■

Corollary 3.1 *If $N < \frac{C_{out}}{C_{in}}$ for SD (or $N < \frac{KC_{out}}{C_{in}}$ for PD), when the system does not accept requests it still has unused capacity.*

Proof. For SD, assume $N < C_{out}/C_{in}$. When a server is full, i.e., N clients are connecting to the server concurrently, the server will block further request. The server capacity in-use is

$$N \times C_{in} < (C_{out}/C_{in}) \times C_{in} = C_{out}$$

which proves our claim. The proof for PD is similar and will be omitted. ■

Corollary 3.1 shows that, if K servers are deployed in the system, letting N to be less than the critical value of the respective scheme will cause the under-utilization of servers and is a waste of server capacity. Hence when $\frac{C_{out}}{C_{in}}$ is large, N could become an issue in that having an acceptable N, which is greater than the critical value, could be expensive or infeasible. We will denote the critical values C_{out}/C_{in} as N_{SD}^C and KC_{out}/C_{in} as N_{PD}^C in the rest of the paper.

Proposition 3.2 *For SD and PD, if $\rho = \frac{\lambda F}{KC_{out}} > 1$, P_B is strictly greater than 0 for any finite N.*

Proof. It can be shown that the blocking rate is a non-increasing function of N given other parameters are fixed, and we can rewrite equation (3) (and 6) as:

$$P_B = (\frac{P_0}{P_N} + \frac{P_1}{P_N} + \cdots + \frac{P_{N-1}}{P_N} + 1)^{-1} \qquad (7)$$

When $\rho > 1$, $P_N > P_i$ for all $i < N$. Therefore, $\frac{P_i}{P_N} < 1$ for all $i \neq N$, and

$$P_B > \frac{1}{N+1} > 0$$

■

Here ρ can be viewed as the load of the system, as λF is the data request rate and KC_{out} is the total capacity of the system. We will use this definition of ρ in the rest of the paper.

Corollary 3.2 *If $\rho < 1$, P_B could be made arbitrarily small by increasing N.*

Proof. By letting N in Equation (3) and (6) go to infinity, when $\rho < 1$, $P_B = P_N \to 0$. ■

These propositions have significant influences on system dimensioning as discussed next.

4 Performance Evaluation

In the models we presented in Section 3, with given C_{in}, C_{out}, and λ, which represent the physical link capacities and the request rate, it is clear that different settings of K and N will lead to different performance of SD and PD. These results allow CDN providers to properly dimension the network, and select the corresponding downloading scheme (PD or SD) under different constraints. In practice, the number of servers K is constrained by cost while the value of N is constrained by server resources such as server memory and CPU. In this section we use our analytical models to provide a guideline to dimension the network resources so that we can improve the cost-effectiveness of CDN. We consider the scenario when CDN providers want to decide the number of servers and the scheme given target values for $\overline{D}$ and P_B, and the scenario when the number of servers is fixed and a certain target performance is desired for P_B and we want to minimize $\overline{D}$. For the later case, we consider both situations when N is limited and N is not limited.

4.1 Network Dimensioning

We consider the following dimensioning problem. For a given scheme:

$$\begin{aligned} &\min K \\ &\text{s. t. } \overline{D} \leq \overline{D}^t \\ &\qquad P_B \leq P_B^t \end{aligned}$$

where $\overline{D}^t$ and P_B^t denote the target average downloading time and the target blocking rate respectively. This scenario corresponds to the case in which CDN providers try to determine the number of servers to deploy in a service area as a function of a target average downloading time and a target blocking rate assuming that they have a good estimate of λ and F. A dimensioning example for this scenario is presented in Figure 2.

The network parameters we have chosen are: $C_{in} = 100$kbps, $C_{out} = 10$Mbps, $F = 125$MBytes, and $\lambda = \frac{1}{15}$ requests/second. In this scenario, we consider N unbounded. Target blocking rate is set at 5% and the minimum possible K as a function of the target average downloading time is shown in Figure 2(a). The corresponding values of N required to achieve such performance are shown in Figure 2(b). We will denote these two values as K_{min} and N^* respectively for a given pair $(\overline{D}^t, P_B^t)$. Figure 2(c) and 2(d) show the case for P_B^t set at 1%.

For a fixed set of network parameters and K, there is a minimum achievable blocking rate for both SD and PD which can be found by letting N in Equations (3) or (6) go to infinity. This property comes from the fact that P_B is a non-decreasing function of N. Therefore, by determining a P_B^t, the *lower bound* of K_{min} is also determined. The remaining question is with such K_{min}, whether $\overline{D}^t$ can be achieved by changing the value of N. For example, in the case where network parameters are specified as above, we have computed the lower bound of P_B achievable by SD and PD for $K = 6$, they are both about 0.1 and those for $K = 7$ are both 0. This suggests that for any P_B^t less than 10%, at least 7 servers will be needed.

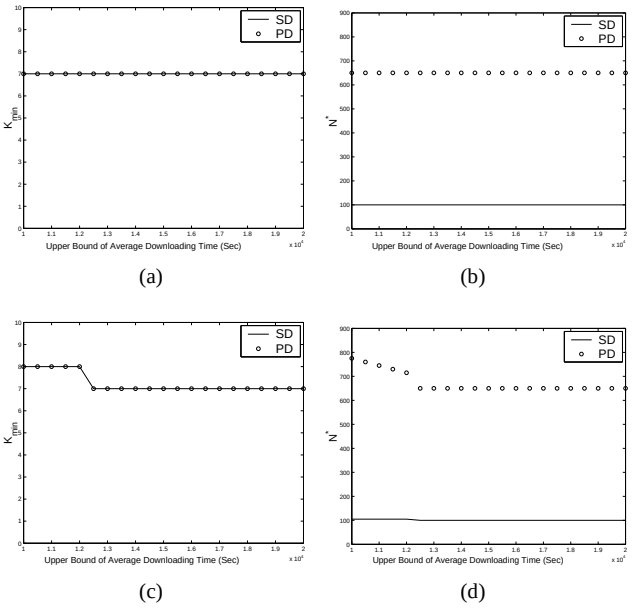

Figure 2. Network Dimensioning with different $\overline{D}^t$ and P_B^t: (a) K_{min} for P_B^t =5%; (b) Corresponding N^* for P_B^t = 5%; (c) K_{min} for P_B^t = 1%; (d) Corresponding N^* for P_B^t = 1%

Figure 2(a) shows that for P_B^t =5%, the minimum number of servers required will be 7 for both SD and PD, even if $\overline{D}^t$ is set to its minimum possible (F/C_{in}). Since relaxing the requirement of $\overline{D}^t$ can only reduce K_{min}, and K_{min} has reached its lower bound in this case, further relaxing $\overline{D}^t$ will have no effect on K_{min}. On the other hand, in Figure 2(c) where the P_B^t is 1%, 7 servers are not enough to meet a $\overline{D}^t$ of F/C_{in} and from our computation we find that 8 servers will be needed. The value of K_{min} decreases to its lower bound of $K_{min} = 7$ when we relax $\overline{D}^t$ requirement to about 125% of F/C_{in}. SD and PD require the same number of servers to achieve the targets.

Figures 2(b) and 2(d) present N^* as a function of $\overline{D}^t$. N^* is the number of concurrent sessions each server should be able to accommodate in order to allow the system to reach the targets with K_{min} servers. As we can see, the value of N^* for SD and PD are significantly different

for the same target values. SD requires an N^* of about $C_{out}/C_{in} = N^C_{SD}$, while PD requires an N^* of about $KC_{out}/C_{in} = N^C_{PD}$. It is possible for the values of N^* for PD to be less than N^C_{PD}, which is shown in the figures. However, from Corollary 3.1, setting any value of N less than N^C_{PD} will cause the under-utilization of servers and may create unnecessary blocking, so for any PD system with K servers, having an N value less than N^C_{PD} will be a waste of resources.

Figure 2 shows that the values of N^* are equal to N^C of the respective schemes, meaning that for the same network setting, deploying PD will requires K_{min} times more resources on each server than deploying SD since $N^C_{PD} = K \times N^C_{SD}$. The difference in server resource requirement for each server is significant even though both schemes require similar number of servers. This observation holds regardless of the values of λ, C_{out}, C_{in}, and other network parameters. If the overhead and complexity that PD imposes are also considered, SD appears to be a much better candidate.

4.2 Server Resource Dimensioning

We now consider the scenario where K, the number of servers, is fixed. This corresponds to the case where a certain number of CDN servers are already deployed, and the goal is to dimension N, the amount of server resources, such that the network performance is optimized. The problem can also be viewed as an admission control problem, where the network needs to limit the number of concurrent connections in order to achieve the desired system performance. We formulate the problem as follows:

$$\begin{aligned} &\min \overline{D} \\ &\text{s. t. } P_B \leq P^t_B \end{aligned}$$

with K, λ, F and $\frac{C_{out}}{C_{in}}$ fixed. In Section 4.2.1, N is unbounded while in Section 4.2.2, N is bounded by N_{max}, which adds a constraint to our optimization problem. The system parameters used in both sections are the same as before, and the value of K is fixed at 6 ($\rho > 1$) and 7 ($\rho < 1$). In general, $\rho > 1$ indicates the network is not well dimensioned. This situation can happen, but it is highly undesirable.

4.2.1 When N is unbounded

Figure 3(a) shows the minimum mean downloading time as a function of P^t_B with $K = 7$. The first observation is that there is a small difference between SD and PD when P^t_B is low, and the two schemes yield similar results when P^t_B is larger than 5%. The difference becomes insignificant when K is larger, or in other words, ρ is much smaller than 1. PD has a slight advantage in terms of performance when ρ

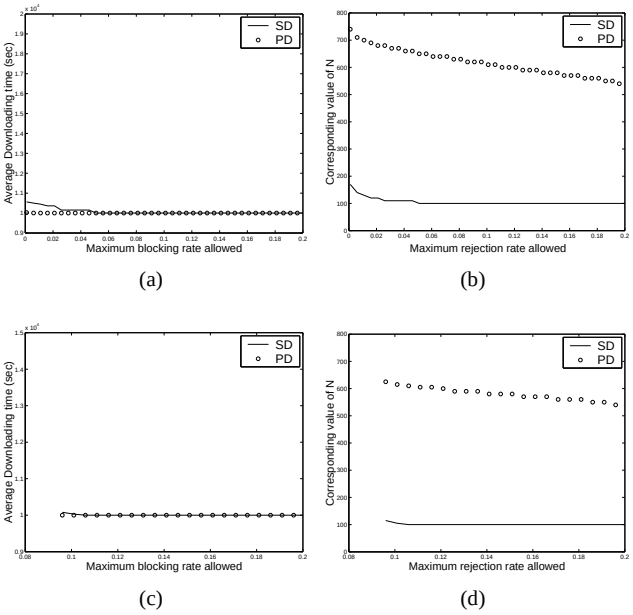

Figure 3. (a)Minimum $\overline{D}$ as a function of P^t_B for K = 7; (b) Corresponding N; (c) Same scenario as (a) for K = 6; and (d) Corresponding N

is close to 1 ($\rho = 0.95$ in this case) and the target performance criteria is tight. When ρ is small (less than 0.9), the performance of SD and PD are not significantly different even with a tight target. Figure 3(b) shows the corresponding minimum server resources N^* required to achieve such performance. Like the network dimensioning problem, PD requires an N^* that is K times larger than that of SD to achieve the same target. This minimum value of N^*, even though less than N^C_{PD}, is not a realistic minimum as discussed in the previous section. Fixing any value of N less than N^C of the corresponding scheme makes the servers under-utilized and will not enhance performance.

For the case where ρ is greater than 1, which is shown in Figure 3(c), there exists a range of target blocking rates that are not achievable due to capacity limitation, which means the lower bound of blocking rate achievable is greater than zero for the particular K. Also, regardless of the choice of N, as long as it is larger than N^C and is finite, the blocking rate will remain the same. With this property, we can dimension the system to provide an arbitrary low downloading time, which cannot be less than the minimum possible, with the same blocking rate. The best strategy is to limit N to N^C of the scheme of choice such that the blocking rate is indifferent and the average downloading time is guaranteed minimum possible (Proposition 3.1).

Considering the fact that the requirement of server resources for PD is higher than that for SD by an order of K,

as well as the potential overhead imposed by PD, the choice of downloading scheme in this scenario should be SD.

4.2.2 When N is bounded

Now we consider the following optimization problem:

$$\begin{aligned} &\min \overline{D} \\ &\text{s. t. } P_B \leq P_B^t \\ &\quad N \leq N_{max} \end{aligned}$$

with K, λ, F and $\frac{C_{out}}{C_{in}}$ fixed. Figure 4(a) and 4(b) show the average downloading and blocking rate achievable with 6 servers ($\rho > 1$) with varying N. For SD, P_B reaches its minimum approximately at $N = 100 = N_{SD}^C$ and $\overline{D}$ starts to increase at the same value of N. The floor for P_B has been explained in the previous section, and the increase of $\overline{D}$ as N increases is a consequence of such phenomenon. As a result, increasing N will decrease the capacity received by each client and thus increase the service time. The same behavior is also observed in PD, except it happens at $N = 600 = N_{PD}^C$. Figure 4(c) and 4(d) show the case for K = 7, in which $\rho < 1$, and again P_B reaches minimum at the a similar value of N ($\approx N^C$) for both SD and PD. The increase of $\overline{D}$ does not happen in this case, mainly due to the fact that the expected number of concurrent clients is finite (because $\rho < 1$), which is less than N. Therefore, the increase of N will not allow more clients into the system. So the value of N, does not make a different in terms of system performance for $\rho < 1$, as long as it is greater than N^C of the respective downloading scheme.

From the above observation we can conclude that, when $\rho < 1$, the bound on N will not make any difference as long as it is greater than N^C of the chosen downloading scheme, and there will be no performance improvement by forcing N to reach its upper bound. Conversely, when $\rho > 1$, if the bound for N is greater than N^C, admission control should be performed so that the value of N is limited to N^C, otherwise system performance will begin to degrade with the increment of N.

In this scenario, we again see that SD and PD performs similarly, but PD requires more server resources as in the previous cases. This observation, like what we have in Section 4.1, holds regardless of the values of the network parameters. Therefore, after considering all scenarios presented in this section, it is reasonable to conclude that SD is a more appropriate scheme to deploy for large file downloading.

5 Conclusion

Parallel downloading (PD) has been adopted recently in some Internet file downloading systems, and is expected to

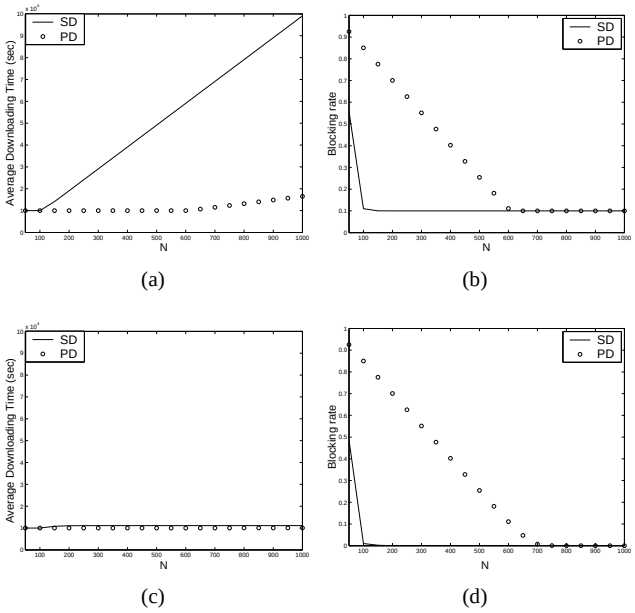

Figure 4. Performance of SD and PD as a function of N: (a) $\overline{D}$ when K =6; (b) P_B when K=6. (c) and (d) are the counterparts when K = 7.

be more commonly adopted with the increasing deployment of CDN and peer-to-peer networks. The work reported in this paper was initiated by the lack of an in-depth analysis of the system performance and the impact on the whole system of such a popular scheme. In this paper, we have built mathematical models for both PD and single server downloading (SD), and we have applied the models to address practical problems such as network dimension and admission control. It should be noticed that our conclusions are drawn based on a homogeneous network scenario and on the *average* downloading time. For heterogeneous scenarios where clients have different connectivity, average downloading time may not be a suitable performance metric to study.

We have shown in the paper that SD and PD, if adopted on a properly dimensioned network, perform similarly in terms of average downloading time and blocking rate. However, we conclude that SD is a better candidate scheme than PD because PD requires more server resources to achieve a similar performance as SD, and the scheme itself imposes complexities and overhead in synchronization, buffering, re-sequencing, etc. This does not happen with SD. The result shows that PD is not necessarily such a great scheme to adopt.

When the servers are constrained in terms of number of concurrent sessions that they can serve, we have shown that admission control should be deployed to prevent unneces-

sary system degradation. We have presented in the paper that if the number of servers is limited, the system should limit the number of users regardless of the downloading scheme. This admission control process will prevent the average downloading time from rising without increasing the blocking rate.

References

[1] J. Byers, J. Considine, M. Mitzenmacher, and S. Rost. Informed Content Delivery Across Adaptive Overlay Networks. In *Proc. of ACM SIGCOMM*, 2002.

[2] J. W. Byers, M. Luby, and M. Mitzenmacher. Accessing Multiple Mirror Sites in Parallel: Using Tornado Codes to Speed Up Downloads. In *Proc. of INFOCOM*, pages 275–283, New York, NY, Mar. 1999.

[3] J. W. Byers, M. Luby, M. Mitzenmacher, and A. Rege. A Digital Fountain Approach to Reliable Distribution of Bulk Data. In *Proc. of ACM SIGCOMM*, pages 56–67, Vancouver, Canada, 1998.

[4] R. L. Carter and M. Crovella. Server Selection Using Dynamic Path Characterization in Wide-Area Networks. In *Proc. of INFOCOM*, volume 3, pages 1014–1021, Kobe, Japan, Apr. 1997.

[5] Digital Fountain Inc. Digital Fountain' s Meta-Content Technology. White Paper.

[6] Z. Fei, S. Bhattacharjee, E. W. Zegura, and M. H. Ammar. A Novel Server Selection Technique for Improving the Response Time of a Replicated Service. In *Proc. of INFOCOM*, volume 2, pages 783–791, 1998.

[7] L. Kleinrock. *Queueing System*, volume 2 (Computer Applications). John Wiley and Sons, fi rst edition, Apr. 1976.

[8] Y. Liu, W. Gong, and P. Shenoy. On the Impact of Concurrent Downloads. In *2001 Winter Simulation Conference*, Arlington, VA, 2001.

[9] A. Myers, P. A. Dinda, and H. Zhang. Performance Characteristics of Mirror Servers on the Internet. In *Proc. of INFOCOM*, volume 1, pages 304–312, 1999.

[10] Y. Nebat and M. Sidi. Resequencing Considerations in Parallel Downloads. In *Proc. of INFOCOM*, 2002.

[11] S. Philopoulos and M. Maheswaran. Experimental Study of Parallel Downloading Schemes for Internet Mirror Sites. In *Thirteenth IASTED International Conference on Parallel and Distributed Computing Systems (PDCS '01)*, pages 44–48, Aug. 2001.

[12] P. Rodriguez and E. Biersack. Dynamic parallelaccess to replicated content in the Internet. *IEEE/ACM Transactions on Networking*, 10(4), Aug. 2002.

[13] M. Sakata, S. Noguchi, and J. Oizumi. An Analysis of the M/G/1 Queue Under Round-Robin Scheduling. *Operations Research*, 19:317–385, 1971.

[14] S. Saroiu, K. P. Gummadi, R. J. Dunn, S. D. Gribble, and H. M. Levy. An analysis of internet content delivery systems. In *Proc. of the Fifth Symposium on Operating Systems Design and Implementation (OSDI)*, Boston, MA, Dec. 2002.

[15] D. Villela and D. Rubenstein. A Queuing Analysis of Server Sharing Collectives for Content Distribution. Technical Report EE200412-1, Columbia University, Apr. 2002.

[16] A. Zeitoun, H. Jamjoom, and M. El-Gendy. Scalable Parallel-Access For Mirrored Servers. In *The 20th IASTED International Conference on Applied Informatics (AI 2002)*, Innsbruck, Austria, Feb. 2002.