The Case for Servers in a Peer-to-Peer World

Shirshanka Das, Saurabh Tewari, Leonard Kleinrock

Computer Science Department
University of California Los Angeles
Los Angeles, CA 90095-1596
{shanky, stewari, lk}@cs.ucla.edu

***Abstract*— The increasing ease of self-expression and web-publishing has resulted in an explosion in the amount of content being generated in the current Internet. Besides traditional sources such as news portals, regular users are documenting their lives and thoughts and other people are subscribing, downloading and viewing this content. A lot of content therefore is being generated at the edge and consumed by the edge.**

Traditional client-server architectures are known to be ineffective in handling large correlated bursts of user demands. However, with RSS becoming more popular, such flash crowd scenarios will be more and more commonplace due to automated polling and downloads. Peer to peer protocols such as BitTorrent provide an attractive solution for such scenarios. BitTorrent networks are scalable, and the expected download time is independent of the arrival rate of peers (content consumers). However, the base performance of a BitTorrent network may not be fast enough from a user or content publisher's perspective. Besides, BitTorrent gives poor results towards the end of a flash crowd when most of the large burst of arrivals have downloaded and left, and there are not too many peers online.

We motivate the need for a content delivery network with well connected servers to participate in BitTorrent delivery streams. The servers are dynamically added and function as cushions to handle increase in demand as well as bolster a delivery stream when there is a paucity of users. We analytically model the approach and determine the average download time both in steady state and in the transient state. Our analysis shows that a content delivery provider can provision the deployment of servers on-the-fly based on the arrival rate of peers to provide a certain pre-agreed average download time.

I. Introduction

The enormous popularity of weblogs and peer to peer applications over the last couple of years points to a paradigm shift in the traditional formats of publishing on the world wide web. While media houses and news providers such as CNN and BBC remain significant providers of content for the web, freelancing individuals such as Adam Curry have become famous overnight through their strong online presence through weblogs or blogs as they are commonly termed.

Users are generating and publishing content on the Web faster than ever and in much larger amounts. While many people are using free online servers like Xanga.com, these free online sites are getting seriously overloaded by the millions of hits they get. Some of these sites such as livejournal.com are getting seriously bottlenecked over time. In most cases, storage is not the bottleneck; it's just the server not being able to handle the huge number of client accesses.

We believe that users will be willing to adopt more and more decentralized ways of publishing content in order to get around the scaling problems of the traditional client-server architecture. For example, they could depend on the main server only to point to or host the metadata related to the content (e.g. the torrent file corresponding to a BitTorrent download) allowing everyone to retrieve the data in a cooperative manner. Swarming peer-to-peer protocols such as BitTorrent have proven very successful at such edge-to-edge delivery of content. However, the massive popularity of BitTorrent have been chiefly with respect to illegal, copyrighted material, for which users are willing to wait an extra night so long as the download is free. As more traditional applications on the web transition to swarming downloads, users will demand a more spontaneous experience, driving incentive for content delivery systems to provide some of the resources needed to maintain high download rates. One example, Prodigem [10] aims to do exactly that.

II. Related Work

There has been a significant amount of commerical as well as research interest in the content delivery problem. HTTP-based Content Distribution Networks (CDNs) have successfully been used to serve Web pages, offloading origin servers and reducing download latency. These systems, such as those from Akamai [1], generally consist of a distributed set of caching Web proxies and a set of request redirectors. The proxies are responsible for caching requested objects so that they can be served from the edge of the network instead of contacting the origin server. The redirectors, often modified DNS servers, send clients to appropriate CDN nodes, using a number of factors, such as server load, request locality (presence of object in the cache), and proximity (network latency to the node).

BitTorrent [3], a peer-to-peer protocol proposed and implemented in 2001 has rapidly gained popularity. Millions of people use it regularly to share legal as well as copyrighted content with millions of others. There are already efforts to harness the power of BitTorrent to create full fledged content delivery sites with proper rights management for large files at a fraction of the cost incurred by heavyweight content delivery providers like Akamai.

Cooperative content delivery has also been the focus of several research efforts recently. SplitStream [2] uses BitTorrent-like swarming coupled with network coding to get greater robustness, CoDeploy [8] uses a set of cooperative HTTP proxies which execute partial downloads using the latest HTTP protocol features to assemble the complete file in a somewhat similar fashion to BitTorrent. All the commercial efforts and the several papers that have proposed swarming like mechanisms to deploy the content would benefit directly from the work we describe in this paper.

There has been a fair amount of research on modeling BitTorrent [6], [7], [9], [11], [12]. Reference [11] models the rate of change in the number of downloaders and seeds in the system and derives the expected download time (for any one downloader) in the steady-state using Little's result [4] to obtain the number of downloaders in the steady-state. We use their modeling approach for our work in this paper and provide the relevant extensions to include the effect of overlay nodes. Reference [12] uses a branching process model to obtain the total time to serve a certain number of downloaders all of whom are assumed to have arrived at the same time. Their base model assumes that the clients stay on in the system even after their downloads are complete. This assumption is later replaced by a model where a certain fraction of downloaders leave immediately after their downloads are complete while the remaining remain in the system forever. We believe that it is more realistic to assume that the clients who have completed downloading will remain in the system only for a (short) random amount of time as assumed by [11]. Reference [6] presents a very generalized model for BitTorrent-like systems. However, for the metric of our interest, the expected download time, the conclusions are similar to those in the more simplified models of [11], [12] so we chose to extend the model in [11]. The model in [12] is extended for the limited network capacity case in [7] which models swarming-based content distribution in adhoc wireless networks. While the network capacity is limited in wired networks also, we do not include a limit on the network capacity in our case as the service capacity is the limiting factor in the download time in wired networks. The model in [11] is extended in [9] to include the effect of heterogenous client link capacities on the average download time. In our model, the link capacities are heterogenous between the overlay nodes and the client nodes but we do not distinguish between client nodes themselves since our goal is to understand the effect of overlay nodes.

Our Contributions

- We formulate the problem faced by a content delivery service provider seeking to optimize its resource usage while maintaining the expected download time constraint as set by its agreement with the customer.
- We propose the concept of hierarchical swarming, where certain nodes in the BitTorrent overlay act as high-speed always-on seeds, while the clients exhibit more realistic behaviours, such as logging off as soon as their download is complete.
- We model the problem using a simple deterministic fluid model extending the model proposed in [11].
- We model the effect of the number of hierarchical nodes on the average download time, and thus outline a simple algorithm for a content delivery provider to optimize its resources.

Organization of the Paper

The paper is organized as follows. In Section III we give a brief overview of the BitTorrent protocol. Section IV defines a projected use case for swarming content downloads and outlines our assumptions about the scenario. In Section V, we describe a deterministic fluid model for hierarchical swarming and derive the expressions for the expected download time in the steady-state and the transient phase. Finally, Section VI concludes the paper.

III. Overview of BitTorrent

A good description of the BitTorrent protocol from its creator is given in [3]. In this section, we only describe the aspects of BitTorrent that are directly relevant to our discussion in this paper. BitTorrent is typically used to distribute large files. A key feature of BitTorrent is its division of the content into several small pieces. Users wishing to download the content obtain a (partial) list of other downloaders from a tracker which keeps track of current downloaders and the pieces of the content they have already obtained. The user wishing to download connects to other downloaders based on the list provided by the tracker and these peers download and upload from each other at the same time. The protocol enforces sharing by limiting the download rate of users depending on their upload rate. Division of the content into a large number of small pieces allows the downloaders to have content to upload to other downloaders while they are downloading the remaining portion of the content [11]. Thus, the service capacity of the system scales in proportion to the number of downloaders.

IV. Content Delivery using BitTorrent

In this section we put forward the case for combining the power of BitTorrent and the ease of RSS to create a self-swarming self-updating content delivery mechanism [5]. The idea is that RSS feeds enclose torrent files which are updated when new content is created and published. Client applications polling the RSS feeds will automatically download the torrent and start downloading the actual file referred to by the torrent using any Bittorrent-compliant client. Thus, asynchronous notification follows cooperative downloading leading to a much better download experience. We argue for the case of having content delivery providers take part in this communal download to provide power (in terms of upload bandwidth) and reliability (through always-"up" nodes) when the BitTorrent network is unable to achieve satisfactory download times.

To formally describe the system, we assume that we have :

- A BitTorrent overlay of *clients* who want to download a certain file.
- A content delivery provider who provides M *overlay nodes*. The overlay nodes provide more capability, for

example, we assume that the upload rate of the overlay nodes is m times (100 to 1000) that of the clients (e.g. 1 Gbps versus 1.5 Mbps).

	Time	New seeds generated	Seeding clients starting new cycle	Total seeds so far	
	t	*1*	*1*	*1*	
	2t	*1*	*1*	*2*	
	3t	*1*	*1*	*3*	
	.	.	.	.	
	.	.	.	.	
	.	.	.	.	
T	*mt*	*1*	*1*	*m*	*m*$(2^1$-1)
	T+t	*1+1*	*3**	*m+2*	
	T+2t	*1+1*	*3*	*m+4*	
	.	.	.	.	
	.	.	.	.	
	.	.	.	.	
2T	*T+mt*	*2*	*3*	*3m*	*m*$(2^2$-1)
	2T+t	*1+3*	*7***	*3m+4*	
	2T+2t	*1+3*	*7*	*3m+8*	
	.	.	.	.	
	.	.	.	.	
	.	.	.	.	
3T	*2T+mt*	*4*	*7*	*7m*	*m*$(2^3$-1)
	3T+t	*1+7*	*15****	*7m+8*	
	3T+2t	*1+7*	*15*	*7m+16*	
	.	.	.	.	
	.	.	.	.	
	.	.	.	.	
4T	*3T+mt*	*8*	*15*	*15m*	*m*$(2^4$-1)
	4T+t	*1+15*	*31*****	*15m+16*	
	.	.	.	*15m+32*	
	.	.	.	.	
	.	.	.	.	

** 1 done, starting new + 1 newly created by regular node + 1 newly created by overlay*
*** 3 done, starting new + 3 newly created by regular node + 1 newly created by overlay*
**** 7 done, starting new + 7 newly created by regular node + 1 newly created by overlay*
***** 15 done, starting new + 15 newly created by regular node + 1 newly created by overlay*

Fig. 1. Download Time for $M = 1$ with ideal schedule

While the subsequent sections provide analysis of the download time in these systems for realistic scenarios (such as having the clients leave immediately after finishing their download, or staying on for only a short random amount of time after they finish), we can see the power of our approach with the simple idealized approach proposed in [12]. Let us consider the case where there is a single overlay node, i.e. $M = 1$. Assume that the download time for a client is constrained only by the upload speed to that client and not by the client's download speed. Let t be the time for an overlay node to upload the entire file to a client, and T be the time for a regular client to upload the entire file to another client. If the upload speed of the overlay node is m times that of the client, then $T = mt$. Let there be N clients wishing to download the file at time 0. As discussed in [12], to increase the service capacity as quickly as possible, the overlay nodes should upload to a single client at a time at the fastest rate the client can download the file. Thus, at time t, one more replica of the file is created and by time T, m additional replicas of the file have been created. Fig. 1 shows the exponential increase in the service capacity with such an approach. Notice that each of the m clients that the overlay node downloaded to in time T have become seeds and are downloading to other clients. Thus, by time $2T$, the m seeding clients will download to an additional m clients who also become seeds. The overlay node would also have downloaded to an additional m clients in this time. Thus, by time $2T$, $3m$ clients will have the entire file. As shown in Fig. 1, the number of completed downloads in time kT grows as $m(2^k - 1)$. Hence, the time to serve a burst of N file requests is $Tlog_2(N/m)$. Recall that [12] shows the exponential increase in service capacity for a "pure" BitTorrent network, but the time to serve a burst of N file requests in their case is $Tlog_2(N)$. Thus, we see that the factor of m increase in uploading is equivalent to having a factor of m fewer requests leading to a $log_2 m$ decrease in the total service time.

V. Simple Fluid Model

We use a simple fluid model based on [11] for our hierarchical-swarming content delivery network. While a stochastic model would be more accurate, the stochastic analysis in [11] suggests that the stochastic variations do not lead to any qualitative difference in conclusions. While a service provider offering such a hierarchical-swarming content delivery service may wish to conduct further analysis accommodating stochastic variations, since the service capacity of the underlying BitTorrent network scales linearly with the number of downloaders, we expect that the stochastic variations in the number of downloaders would not change the expected download time at a given client arrival rate in the steady-state.

In our model we use the following quantities to characterize a content-delivery network that uses BitTorrent-like swarming techniques to disseminate data to its clients. All analysis is with regard to the serving of a single file whose size is assumed to be 1.

$x(t)$: Number of clients (who are downloading the content) in the system at time t

$y(t)$: Number of overlay nodes in the system at time t

$z(t)$: Number of clients who have finished downloading but have not yet left the system at time t

λ: The arrival rate of new clients. We assume that clients arrive according to a Poisson process.

μ: The uploading bandwidth of a given client. We assume that all clients have the same uploading bandwidth.

c: The downloading bandwidth of a given client. We assume that all clients have the same downloading bandwidth and $c >= \mu$.

$m\mu$: The uploading bandwidth of a given content delivery node. We assume that $m >> 1$, and all content delivery nodes have the same upload bandwidth.

θ: The rate at which clients abort the download.

γ: The rate at which clients that have finished downloading leave the system.

η: The effectiveness of the file sharing (if the swarming clients are constantly uploading the pieces of the content they have to other clients while they are downloading the remaining pieces of the content, file sharing is 100% effective i.e. η is the fraction of the upload capacity of swarming clients that is being utilized with values in [0,1]).

In a swarming content-delivery network, the provider has to make provisions for the worst case which involves the client logging off as soon as he completes his download. We have included the possibility of some clients remaining in the system even after they have finished downloading ($z(t)$) for completeness. Specifically, the model allows for clients who have finished downloading to remain in the system for an exponentially distributed random time with mean $\frac{1}{\gamma}$. However, the provider must assume the rate at which the clients who have completed downloading leave the system to be very high for provisioning purposes i.e. $\gamma \to \infty$.

Without any constraint on downloading bandwidth, the total uploading rate of the system can be expressed as $\mu(\eta x(t) + my(t) + z(t))$. If $\eta = 0$, then we have a simple content delivery network (like Akamai), where the overlay nodes act as mirrors for the content. When the downloading bandwidth constraint is considered, the total uploading rate will be $min\,\{cx(t), \mu(\eta x(t) + my(t) + z(t))\}$.

Occasionally, a client may leave before the downloading is complete in case the client is unsatisfied with the download experience. We assume that each client independently aborts its download after a certain amount of time which is exponentially distributed with mean $1/\theta$. Equivalently, θ is the rate at which clients abort their download and leave the system. In a fluid model, the rate of departures of downloaders will be given by :

$$min\,\{cx(t), \mu(\eta x(t) + my(t) + z(t))\} + \theta x(t)$$

We now describe the evolution of x and z based on the above model. A deterministic fluid model for the evolution of the number of peers (clients) is given by :

$$\frac{dx(t)}{dt} = \lambda - \theta x(t) - min\,\{cx(t), \mu(\eta x(t) + my(t) + z(t))\} \tag{1}$$

$$\frac{dz(t)}{dt} = min\,\{cx(t), \mu(\eta x(t) + my(t) + z(t))\} - \gamma z(t) \tag{2}$$

A. Steady State Analysis

At the steady state, $\frac{dx(t)}{dt} = 0$ and $\frac{dz(t)}{dt} = 0$. Hence, from (1) and (2), we have

$$\lambda - \theta\bar{x} - min\,\{c\bar{x}, \mu(\eta\bar{x} + mM + \bar{z})\} = 0 \tag{3}$$

$$min\,\{c\bar{x}, \mu(\eta\bar{x} + mM + \bar{z})\} - \gamma\bar{z} = 0 \tag{4}$$

where $\bar{x}$ and $\bar{z}$ are the equilibrium values of $x(t)$ and $z(t)$, respectively and M is the number of overlay nodes deployed by the provider at the steady-state.

Like [11], we also assume $\eta > 0$. When the downloading speed is the constraint, i.e. if $c\bar{x} <= \mu(\eta\bar{x} + mM + \bar{z})$, (3) yields

$$\bar{x} = \frac{\lambda}{\theta + c} \tag{5}$$

When the downloading speed of clients is the constraint, the overlay nodes are not being utilized fully and the provider would prefer to not over-provision the number of overlay nodes at least in steady-state. When the upload bandwidth is the constraint, i.e., if $c\bar{x} \geq \mu(\eta\bar{x} + mM + \bar{z})$, equations (3) and (4) yield

$$\bar{x} = \frac{\lambda}{\nu(1 + \frac{\theta}{\nu})} - \frac{mM}{\eta(1 + \frac{\theta}{\nu})} \tag{6}$$

$$\bar{z} = \frac{\lambda}{\gamma(1 + \frac{\theta}{\nu})} + \frac{\theta mM}{\gamma\eta(1 + \frac{\theta}{\nu})} \tag{7}$$

where $\frac{1}{\nu} = \frac{1}{\eta}(\frac{1}{\mu} - \frac{1}{\gamma})$ (we assume that clients remain in the system after completing their download for a shorter time than their expected download time in the "pure" BitTorrent network, i.e. $\frac{1}{\mu} > \frac{1}{\gamma}$).

One can easily verify that (6), (7) reduce to (4) in [11] when no overlay nodes are deployed and only BitTorrent is used for content delivery, i.e. $M = 0$.

To calculate the average downloading time for a client in steady state, we can use Little's Result as :

$$\frac{\lambda - \theta\bar{x}}{\lambda}\bar{x} = (\lambda - \theta\bar{x})T, \tag{8}$$

where T is the average downloading time, $(\lambda - \theta\bar{x})$ is the average rate at which downloads are completed, and $\frac{\lambda - \theta\bar{x}}{\lambda}$ is the fraction of interesting downloaders, that is the downloaders that will continue to completion of the download. Using (5) and (6), we get the average download time as

$$T = \frac{1}{\theta + c} \tag{9}$$

for the download constrained region and

$$T = \frac{1}{\nu(1 + \frac{\theta}{\nu})}(1 - \frac{mM}{\lambda\eta}\nu) \tag{10}$$

for the upload constrained region.

Comparing this expression to that derived in [11], we see that the existence of overlay nodes linearly decreases the expected download time compared to a regular edge-node based BitTorrent network. However, the reduction in the expected download time on account of the overlay nodes decreases as the rate of arrivals λ increases. Thus, the content provider would need to increase M linearly with increasing λ to keep the average download time the same.

A content delivery provider can use (10) to calculate the number of overlay nodes to deploy for a given expected download time target, given the arrival rate of new nodes, and the upload and download bandwidth. As discussed earlier, the provider should assume $\gamma \to \infty$ for overlay capacity provisioning. We can rewrite (10) for this case as

$$T = \frac{1}{\mu\eta(1 + \frac{\theta}{\mu\eta})}(1 - \frac{mM}{\lambda\eta}\mu\eta) \tag{11}$$

Since $\mu\eta < \nu$, we see that the presence of clients who have finished downloading (i.e. $\frac{1}{\gamma} > 0$ and $z(t) > 0$) increases the effect of overlay nodes on the download time[1]. This is in line with our analysis in Section IV where we found for the case where all clients remain in the system after completing their download that addition of overlay nodes is equivalent to having fewer requests by a factor equal to the additional capacity added by the overlay nodes.

[1]The percentage reduction in the download time on account of the overlay nodes is $\frac{T - T_0}{T} = \frac{mM}{\lambda\eta}\nu$ where T is the expected download time for non-zero M and T_0 is the expected download time when $M = 0$. When $\frac{1}{\gamma} = 0$ (and $z(t) = 0$), $\frac{T - T_0}{T} = \frac{mM}{\lambda\eta}\mu\eta = \frac{mM\mu}{\lambda}$.

Finally, it would also be prudent to assume that no clients leave on account of frustration. With $\theta = 0$ and including the download constraint, we write the expected download time to be

$$T = max\left\{\frac{1}{c}, \frac{1}{\mu\eta} - \frac{mM}{\lambda\eta}\right\} \tag{12}$$

We plot (12) in Fig. 2. As [11] discusses, for large files and multiple downloaders, η, the effectiveness of the BitTorrent file sharing is nearly 1 and we choose $\eta = 1$ in Fig. 2. The overlay node's upload capacity is likely to be 100 times more than that of a client's upload capacity. We plot (12) in Fig. 2 with M varying from 1 to 6, thus illustrating how one can achieve a target expected download time by deploying more nodes. Guided by the measurements on the BitTorrent network reported in [11], we choose $\mu = 0.001$. The downloading speed in a typical broadband connection to home is three times or higher than the uploading speed so we choose $c = 0.003$.

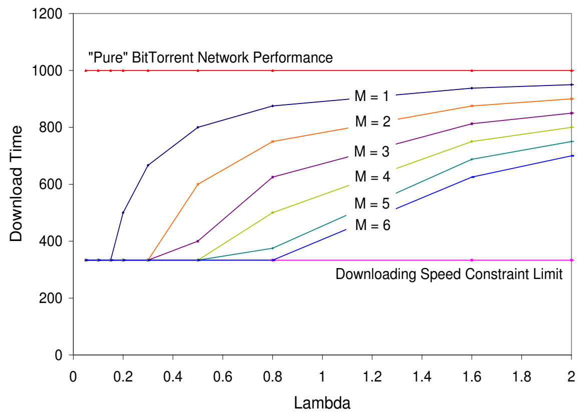

Fig. 2. Steady-State Expected Download Time

We see from Fig. 2 that when λ, the client arrival rate, is very small, there is no congestion and the download time is constrained only by the downloading speed c resulting in an expected download time of $\frac{1}{c}$. This is a direct result of the fixed upload capacity offered by overlay nodes. In contrast, in a "pure" BitTorrent network, the download time is limited by the upload bandwidth of peers independent of the client arrival rate.

We also see that the impact of the service capacity added by a fixed number of overlay nodes diminishes as λ increases. Therefore, the number of overlay nodes a service provider needs to deploy to maintain a target expected download time increases linearly as λ increases[2]. A service provider can estimate the required number of servers given the expected value of λ using (12).

Our analysis in this section targets one service model where the service provider agreement is on the maximum download time up to a maximum client arrival rate. In many content delivery scenarios, the client arrival rate changes suddenly (e.g. the content may become very popular suddenly for a short time or has very high popularity in the initial period). An alternative service model for these scenarios is for the service provider to guarantee a maximum download time independent on the client arrival rate and dynamically adjust the number of overlay servers dedicated to a particular content. In the next section, we investigate the transient behavior of swarming-download-assisted content delivery and find that, even if the client arrival rate increases suddenly a service provider is likely to have the opportunity to detect the change and react appropriately.

B. *Transient Analysis*

We know that the presence of the clients who have finished downloading improves the expected download time. However, since a service provider making average download time guarantees must not assume this help from the clients who have finished downloading. Therefore, for our transient analysis, we assume $z(t) = 0$. We also assume $\theta = 0$ in this section for similar reasons[3] and $\eta \sim 1$ when $x(t) > 0$ in this section.

For the expressions derived below, we assume that the system starts in an empty state, i.e. $x(t = 0) = 0$. At least one overlay node will be deployed to seed the system. In the initial period, when there are very few clients in the system, the downloads can be served by the overlay node itself and the downloading speed of the clients is the constraint in the expected download time. For this download constraint region, from (1) and $x(t = 0) = 0$, we obtain

$$x(t) = \frac{\lambda}{c}(1 - e^{-ct}) \tag{13}$$

The expected download time in this region will be $\frac{1}{c}$.

As more clients arrive (i.e. $cx(t) > \mu x(t) + m\mu M$), the uploading speed will become the constraint. Specifically, the uploading speed becomes the constraint when

$$x(t) = \frac{m\mu M}{c - \mu} \tag{14}$$

Substituting (14) in (13), we obtain the time τ after which the uploading speed becomes the constraint

$$\tau = \frac{1}{c}ln(1 - \frac{m\mu cM}{\lambda(c - \mu)}) \tag{15}$$

From (1) and $x(t = \tau) = \frac{m\mu M}{c-\mu}$, we obtain $x(t)$ for $t > \frac{1}{c}ln(1 - \frac{cm\mu M}{\lambda(c-\mu)})$

$$x(t) = \frac{\lambda - m\mu M}{\mu} - \frac{\frac{\lambda}{\mu} - \frac{cm\mu M}{\mu(c-\mu)}}{(1 - \frac{cm\mu M}{\lambda(c-\mu)})^{\frac{\mu}{c}}} e^{-\mu t} \tag{16}$$

[2]While having to increase servers in proportion to the request rate is same as that required in client-server systems, we note that a "pure" client-server-based content delivery network is completely unscalable with a fixed number of overlay servers (e.g. a 6 overlay server system can handle a λ of only up to 0.6 beyond which the download time becomes unbounded) while a swarming-download-assisted content delivery network can handle arbitrarily large λ (even with $\lambda \rightarrow \infty$, the download time is no more than $\frac{1}{\mu\eta}$, the download time for a "pure" BitTorrent network). In fact, comparing to the case of 6 overlay servers, we see that the swarming downloads allow the system to provide the best possible download time (i.e. download time constrained only by the client download speed) up to $\lambda \sim 0.8$.

[3]One can easily solve (1) and (2) for the general case to obtain the $x(t)$ and $z(t)$ expressions but the analysis of the general case has little value for our application.

Since the clients complete the download at rate $\mu x(t) + m\mu M$ when $x(t)$ clients are downloading, the expected download time can be estimated as $\frac{x(t)}{\mu x(t)+m\mu M}$. Hence, from (16), we obtain the expected download time $\bar{d}(t)$ for $t > \frac{1}{c} ln(1 - \frac{cm\mu M}{\lambda(c-\mu)})$

$$\bar{d}(t) = \frac{1}{\mu}[1 + \frac{m\mu M}{(\lambda - m\mu M)(1 - e^{-\mu t})}]^{-1} \quad (17)$$

Thus, we can write the expected download time in the transient phase as

$$\bar{d}(t) = max\left\{\frac{1}{c}, \frac{1}{\mu}[1 + \frac{m\mu M}{(\lambda - m\mu M)(1 - e^{-\mu t})}]^{-1}\right\} \quad (18)$$

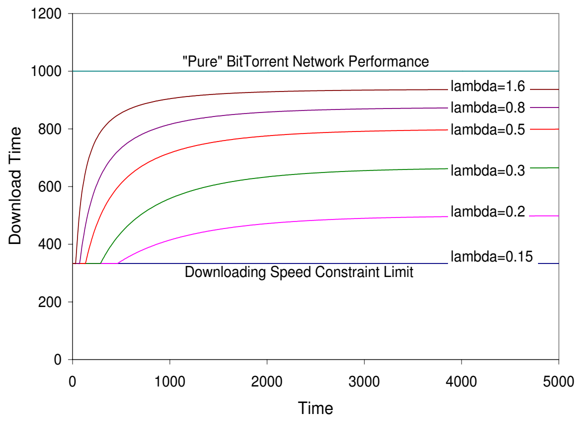

Fig. 3. Expected Download Time in the Transient Period with $M = 1$

We plot (18) in Fig. 3 for $\mu = 0.001, c = 0.003, m = 100$, and $M = 1$, the same parameters that were used for Fig. 2. As we can see from Fig. 3, as the arrival rate λ increases, the region where τ, the expected download time is limited by the downloading speed, diminishes. In addition, in the upload speed constraint region, we find that the expected download time reaches the steady-state download time faster as the arrival rate λ increases. Let us now estimate the duration of the transient phase in the upload constraint region.

From (17), we find that the asymptote for the transient phase in the upload constraint region (Fig. 4) is $(\frac{\lambda}{m\mu M} - 1)t$. Equating to the steady-state download time expression in (12)

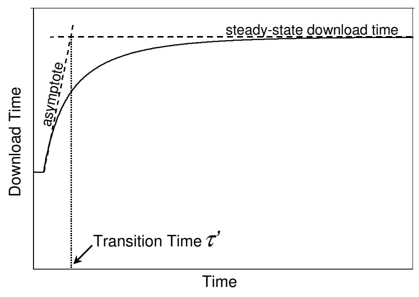

Fig. 4. Transition Period Estimation in Upload Constraint Region

with $\eta = 1$, we get the transient period in the upload constraint region to be

$$\tau' = \frac{mM}{\lambda} \quad (19)$$

Both (15) and (19) show that the transient time is inversely proportional to the client arrival rate. However, we note from (14) that the number of downloading clients in the system at τ' is independent of λ. Further, we see from (6) that the number of downloading clients in the system grows proportionally with λ and from (12) that the number of servers needs to be increased proportionally with λ. Thus, a service provider can keep track of the number of downloading clients in the system and follow the simple strategy of increasing the number of overlay nodes as the number of downloading clients crosses (pre-defined) threshold levels.

VI. Conclusions

In this paper, we make a case for bringing together traditional client/distributed-server paradigms and pure BitTorrent-based peer-to-peer delivery models. The content delivery provider benefits as it can provide guarantees about expected download times while deploying a much lower number of servers and the end user benefits by getting a much shorter download time compared to a pure BitTorrent download.

After motivating the need for using traditional content servers alongside BitTorrent peer-to-peer delivery, we extend a deterministic fluid model for pure BitTorrent network to analyze this new scenario in the steady-state as well as the transient phase. Our results illustrate the benefits of using traditional content servers alongside BitTorrent peer-to-peer delivery over using either of the two approaches by themselves.

A service provider can use our results for download time in the steady-state to determine the number of servers needed to guarantee a target download time up to a certain maximum client arrival rate. Our transient analysis shows that the service provider can dynamically adapt the number of servers they need to deploy to meet the guaranteed download time even when the popularity of a content increases suddenly.

References

[1] Akamai Technologies, *http://www.akamai.com.*
[2] M. Castro, P. Druschel, A-M. Kermarrec, A. Nandi, A. Rowstron and A. Singh, "SplitStream: High-bandwidth multicast in a cooperative environment," In Proc. of ACM SOSP, October 2003.
[3] B. Cohen, "Incentives Build Robustness in BitTorrent," In IPTPS, February 2003.
[4] L. Kleinrock, Queueing Systems, Volume I: Theory, Wiley Intersciece, New York, 1975.
[5] Legal Torrents, *http://www.legaltorrents.com/rss.xml.*
[6] L. Massouli and M. Vojnovic, "Coupon replication systems," In Proc. of SIGMETRICS, June 2005.
[7] A. Nandan, S. Das, G. Pau, M.Y. Sanadidi and M. Gerla, "Cooperative Downloading in Vehicular Wireless Ad Hoc Networks," In Proc. of Wireless On-Demand Networks and Services (WONS), January 2005.
[8] K. Park and V. S. Pai "Deploying Large File Transfer on an HTTP Content Distribution Network," In Proc. of the First Workshop on Real, Large Distributed Systems (WORLDS), December 2004.
[9] F. L. Piccolo, G. Neglia and G. Bianchi, "The Effect of Heterogeneous Link Capacities in BitTorrent-Like File Sharing Systems," In Proc. of HOT-P2P, October 2004.
[10] Prodigem Hosting Solutions, *http://www.prodigem.com.*
[11] D. Qiu and R. Srikant, "Modeling and Performance Analysis of BitTorrent-Like Peer-Peer Networks," In Proc. of SIGCOMM, September 2004.
[12] X. Yang and G. de Veciana, "Service Capacity of Peer to Peer Networks," In Proc. of INFOCOM, March 2004.