Minimizing File Download Time in Stochastic Peer-to-Peer Networks

Yuh-Ming Chiu and Do Young Eun
Department of Electrical and Computer Engineering
North Carolina State University
{ychiu, dyeun}@eos.ncsu.edu

***Abstract*—The peer-to-peer (P2P) file-sharing applications are becoming increasingly popular and account for more than 70% of the Internet's bandwidth usage. Measurement studies show that a typical download of a file can take from minutes up to several hours depending on the level of network congestion or the service capacity fluctuation. In this paper, we consider two major factors that have significant impact on average download time, namely, the spatial heterogeneity of service capacities in different source peers and the temporal fluctuation in service capacity of a single source peer. We point out that the common approach of analyzing the average download time based on *average service capacity* is fundamentally flawed. We rigorously prove that both spatial heterogeneity and temporal correlations in service capacity increase the average download time in P2P networks and then analyze a simple, distributed algorithm to effectively remove these negative factors, thus minimizing the average download time. We show through analysis and simulations that it outperforms most of other algorithms currently used in practice under various network configurations.**

I. Introduction

Peer-to-peer (P2P) technology is heavily used for content distribution applications. The early model for content distribution is a centralized one, in which the service provider simply sets up a server and every user downloads files from it. In this type of network architecture (server-client), many users have to compete for limited resources in terms of bottleneck bandwidth or processing power of a single server. As a result, each user may receive very poor performance. From a single user's perspective, the duration of a download session, or the download time for that individual user is the most often used performance metric.

P2P technology tries to solve the issue of scalability by making the system distributed. Each computer (peer) in the network can act as both a server and a client at the same time. When a peer completes downloading some files from the network, it can become a server to service other peers in the network. It is obvious that as time goes on, the service capacity of the entire network will increase due to the increase in the number of servicing peers. With this increasing service capacity, theoretical studies have shown that the average download time for *each user* in the network is much shorter than that of a centralized network architecture in ideal cases [2], [3]. In other words, users of a P2P network should enjoy much faster downloads.

This work was supported in part by NSF CAREER Award CNS-0545893.
A subset of results reported in this paper has appeared in the proceedings of Conference in Information Science and Systems (CISS) 2006 [1]

However, the measurement results in [4] show that a file download session in a P2P network is rather long and varies a lot from user to user. For instance, downloading an 100MB file in a Gnutella network can range from several hours to a *whole week*. While theoretical studies provide performance bounds for ideal cases, there are many factors that make the real world performance much worse than the theoretical prediction. Some of the major challenges facing a P2P network in the real world include peer selection [5], [6], [7], [8] and data search and routing [9], [10], [11], [12], [13].

Due to the distributed nature of the P2P network, searching and locating data of interest in the network has been an important issue in the literature. In reality, data searching time only contributes a very small portion of a download session while the most delay is caused by actually transferring the file from source peers as shown in [14]. Thus, if we want to minimize the download time for each user, reducing the actual file transfer time would make more noticeable difference. Most recent studies, however, have focused on reducing the total download duration, i.e. the time required for *all users* to finish their download. This total download time is a system-wide performance metric. On the other hand, there are very few results in analyzing the performance of *each individual user*. As the measurement study shows [4], the per-user performance in a P2P network may be even worse than that of a centralized network architecture. Those results suggest that there is much room for improvement in the P2P system in terms of per-user performance, i.e. the file download time of each user.

However, there have been very few results in minimizing the download time for *each user* in a P2P network. In recent work [5], [6], the problem of minimizing the download time is formulated as an optimization problem by maximizing the aggregated service capacity over multiple simultaneous active links (parallel connections) under some global constraints. There are two major issues in this approach. One is that global information of the peers in the network is required, which is not practical in real world. The other is that the analysis is *based on the averaged quantities*, e.g., average capacities of all possible source peers in the network. The approach of using the average service capacity to analyze the average download time has been a common practice in the literature [2], [3], [5], [6], [15], [16], [17].

A. Limitations of Approach via Average Service Capacity

We here illustrate limitations of the approach based on averaged quantities in a P2P network by considering the following examples. Suppose that a downloading peer wants to download a file of size F from N possible source peers. Let c_i be the average end-to-end available capacity between the downloading peer and the i^{th} source peer ($i = 1, 2, \ldots, N$). Notice that the actual value of c_i is *unknown* before the downloading peer actually connects to the source peer i. The average service capacity of the network, $\bar{C}$, is give by $\bar{C} = \sum_{i=1}^{N} c_i/N$. Intuitively, the average download time, T, for a file of size F would be

$$T = F/\bar{C}. \tag{1}$$

In reality, however, (1) is far different from the true average download time for each user in the network. The two main reasons to cause the difference are (i) the *spatial heterogeneity* in the available service capacities of different end-to-end paths and (ii) the *temporal correlations* in the service capacity of a given source peer. We first consider the impact of heterogeneity. Suppose that there are two source peers with service capacities of $c_1 = 100$kbps and $c_2 = 150$kbps, respectively, and there is only one downloading peer in the network. Because the downloading peer does not know the service capacity of each source peer [1] prior to its connection, the best choice that the downloading peer can make to minimize the risk is to choose the source peers with equal probability. In such a setting, the average capacity that the downloading peer expects from the network is $(100+150)/2 = 125$kbps. If the file size F is 1MB, we predict that the average download time is 64 seconds from (1). However, the actual average download time is 1/2(1MB/100kbps)+1/2(1MB/150Mbps) = 66.7 seconds! Hence, we see that the spatial heterogeneity actually makes the average download time longer.

Suppose now that the average service capacity can be known *before* the downloading peer makes the connection. Then, an obvious solution to the problem of minimizing the average download time is to find the peer with the *maximum average capacity*, i.e., to choose peer j with the average capacity c_j ($c_j \geq c_i$ for all i), as the average download time T_i over source peer i would be given by F/c_i. We assume that each peer can find the service capacity of its source peers via packet-level measurements or short-term in-band probing [18].

Consider again the previous two-source peer example with $c_1 = 100$kbps and $c_2 = 150$kbps. As we want to minimize the download time, an obvious choice would be to choose source peer 2 as its average capacity is higher. Now, let us assume that the service capacity of source peer 2 is not a constant, but is given by a stochastic process $C_2(t)$ taking values 50 or 250kbps with equal probability, thus giving $\mathbb{E}\{C_2(t)\} = c_2 = 150$kbps. If the process $C_2(t)$ is strongly correlated over time such that the service capacity for a file F is likely to be the same throughout the session duration, it takes on average (1MB/50kbps + 1MB/250kps)/2 = 96 seconds, while it takes only 80 seconds to download the file from source peer 1. In other words, it may take longer to complete the download when we simply choose the source peer with the maximum average capacity! It is thus evident that the impact of correlations (second-order statistics) or higher-order statistics associated with the capacity fluctuation in time will need to be taken into account, even for finding a source peer with minimum *average* download time.

[1]Although the fluctuation seen by a downloader can be caused by change both in the status of the end-to-end network path and in the status of the source peer itself, we use "service capacity of a source peer" to unify the terminology throughout the paper.

B. Our Contribution

The examples in Section I-A give us a motivation to seek methods that can reduce the download time of each individual user. The main contribution of this paper is to show that the predicted value given in (1) is actually achievable without requiring any global information, regardless of the distribution of service capacities and correlations in a P2P network.

In this paper, we first characterize the relationship between the heterogeneity in service capacity and the average download time for each user, and show that the degree of diversity in service capacities has negative impact on the average download time. After we formally define the download time over a stochastic capacity process, we prove that the correlations in the capacity make the average download time much larger than the commonly accepted value F/c, where c is the average capacity of the source peer. It is thus obvious that the average download time will be reduced if there exists a (possibly distributed) algorithm that can efficiently eliminate the negative impact of both the heterogeneity in service capacities over different source peers and the correlations in time of a given source peer.

In practice, most P2P applications try to reduce the download time by minimizing the risk of getting stuck with a 'bad' source peer (the connection with small service capacity) by using smaller file sizes and/or having them downloaded over different source peers (e.g., parallel download).[2] In other words, they try to reduce the download time by minimizing the *bytes* transferred from the source peer with small capacity. However, we show in this paper that this approach cannot effectively remove the negative impact of both the correlations in the available capacity of a source peer and the heterogeneity in different source peers. This approach may help to reduce average download time in some cases but not always. Rather, a simple and distributed algorithm that limits the amount of *time* each peer spends on a bad source peer, can minimize the average download time for each user almost in all cases as we will show in our paper. Through extensive simulations, we also verify that the simple download strategy outperforms all other schemes widely used in practice under various network configurations. In particular, both the average download time and the variation in download time of our scheme are smaller than any other scheme when the network is heterogeneous (possibly correlated) and many downloading peers coexist with source peers, as is the case in reality.

[2]For example, Overnet, BitTorrent, and Slurpie divide files into 9500KB, 256KB, and 256KB file segments (chunks), respectively [19], [20], [21], and a downloader can transfer different chunks from different source peer.

The rest of the paper is organized as follows. In Section II, we provide some background on service capacity characteristics in a P2P network in terms of the heterogeneity over different connections and correlations over time for a given connection. In Section III, we analyze the impact of heterogeneity in service capacities as well as the correlations in a given connection on each user's average download time. In Section IV, we show that our simple and distributed algorithm and can virtually eliminates all the negative impacts of heterogeneity and correlations. Our scheme thus greatly reduces the average download time and achieves the simple relation in (1) regardless of network settings. Section V provides simulation results to test our algorithm and compare with others under various network settings, and we conclude our work in Section VI.

II. Background

In this section we briefly describe the characteristics of the service capacity that a single user receives from the network from the user's perspective. Specifically, we consider the heterogeneity of service capacities over different network paths and the stochastic fluctuation of the capacity over time for a given source peer.

A. Heterogeneity of Service Capacity

In a P2P network, just like any other network, the service capacities from different source peers are different. There are many reasons for this heterogeneity. On each peer side, physical connection speeds at different peers vary over a wide range [22] (e.g., DSL, Cable, T1, etc). Also, it is reasonable to assume that most peers in a typical P2P network are just personal computers, whose processing powers are also widely different. The limitation in the processing power can limit how fast a peer can service others and hence limits the service capacity.

On the network side, peers are geographically located over a large area and each logical connection consists of multiple hops. The distance between two peers and the number of hops surely affect its round-trip-time (RTT), which in turns affects the throughput due to the TCP congestion control. Moreover, in a typical P2P network, this information is usually 'hidden' when a user simply gets a list of available source peers that have contents the user is looking for.

Note that the aforementioned factors do not change over the timescale of any typical P2P session (days or a week). Hence, we assume that those factors mainly determine the long-term average of the service capacity over a given source peer.

B. Correlations in Service Capacity

While the long-term average of the service capacity is mainly governed by topological parameters, the actual service capacity during a typical session is never constant, but always fluctuates over time [23], [24]. There are many factors causing this fluctuation. First, the number of connection a source peer allows is changing over time, which creates a fluctuation in the service capacity for *each user*. Second, some user applications running on a source peer (usually a PC), such as online games, may throttle the CPU and impact the amount of capacity it can offer. Third, temporary congestion at any link in the network can also reduce the service capacity of all users utilizing that link.

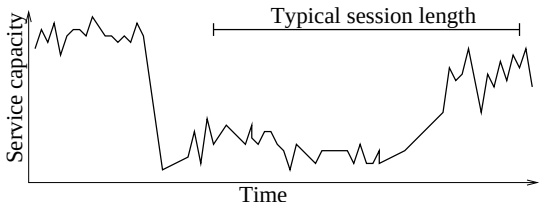


Fig. 1. Typical variation in ene-to-end available bandwidth based on the results in [24], [23]. Drastic changes usually occur in the scale of minutes.

Figure 1 shows a typical available end-to-end capacity fluctuation similar to that presented in [23], [24]. The time scale for the figure in the measurement study is on the order of minutes. We know from [4] that a typical file download session can last from minutes to hours for a file size of several megabytes. This implies that the service capacity over the timescale of one download session is stochastic and correlated. In Figure 1, the short-term variations in the capacity are mainly due to the window size fluctuation in TCP, while the long-term variations are due to network congestion, changes in workload or the number of connecting users at the source peer, etc. The long-term fluctuation typically lasts over a longer time scale, say, few minutes up to several hours.

As illustrated in the introduction, both the heterogeneity over different source peers and the correlations of the capacity in a given source peer have significant impact on the average download time. To the best of our knowledge, however, there has been no result available in the literature addressing these issues. All the existing studies have simply assumed that the service capacity is given by a constant (its average value) for the duration of a download. Consequently, the download time of a file of size F is simply given by F/c, where c is the average service capacity. As will be seen later on, however, this is true only when the service capacity is constant or $i.i.d.$ over time, neither of them is true in reality. In the next section, we will analyze the impact of these two factors on the per-user performance in terms of the average download time.

III. Characterizing the Download Time in a P2P Network

We consider our network as a discrete-time system with each time slot of length Δ. For notational simplicity, throughout the paper, we will assume that the length of a time slot is normalized to one, i.e., $\Delta = 1$. Let $C(t)$ denote the time-varying service capacity (available end-to-end bandwidth) of a given source peer at time slot t ($t = 1, 2, \ldots$) over the duration of a download. Then, the download time T for a file of size F is defined as

$$T = \min\left\{ s > 0 \;\middle|\; \sum_{t=1}^{s} C(t) \geq F \right\}. \tag{2}$$

Note that T is a stopping time or the *first hitting time* of a process $C(t)$ to a fixed level F.

If $C(t)$, $t = 1, 2, \ldots$ are independent and identically distributed (i.i.d.), then by assuming an equality in (2), we obtain from Wald's equation [25] that

$$F = \mathbb{E}\left\{\sum_{t=1}^{T} C(t)\right\} = \mathbb{E}\{C(t)\}\mathbb{E}\{T\}. \quad (3)$$

The expected download time, measured in slots, then becomes $\mathbb{E}\{T\} = F/\mathbb{E}\{C(t)\}$. Note that (3) also holds if $C(t)$ is constant (over t). Thus, when the service capacity is $i.i.d.$ over time or constant, there exists a direct relationship between the average service capacity and the average download time, as has typically been assumed in the literature.

A. Impact of Heterogeneity in Service Capacity

We first consider the impact of heterogeneous service capacities of different source peers. In order to decouple the effect of correlations from that of heterogeneity, in this section, we assume that Wald's equation holds true for *each source peer* (i.e., the service capacity of a given source peer is either constant or $i.i.d.$ over time). But we allow the average capacities for different source peers to be different. We will consider the impact of correlations in Section III-B.

Let N be the number of source peers in the network (N different end-to-end paths) and $C_i(t)$ be the service capacity of source peer i at time slot t. We assume that $C_i(t)$ is either constant or $i.i.d.$ over t such that (3) holds. Let $c_i = \mathbb{E}\{C_i(t)\}$ be the average capacity of source peer i. Then, the average service capacity the network offers to a user becomes

$$A(\vec{c}) = \frac{1}{N}\sum_{i=1}^{N} c_i, \quad (4)$$

where $\vec{c} = (c_1, c_2, \ldots, c_N)$ and $A(\vec{c})$ is the arithmetic mean of the sequence $c_1, c_2, \ldots, c_N$. Thus, one may expect that the average download time, $\mathbb{E}\{T\}$, of a file of size F would be

$$\mathbb{E}\{T\} = \frac{F}{A(\vec{c})}. \quad (5)$$

As we mentioned earlier, however, the actual service capacity of each source peer remains hidden unless a network-wide probe is conducted. So the common strategy for a user is to randomly pick one source peer and keep the connection to it until the download completes. If the user connects to source peer i (with service capacity $C_i(t)$), the average download time over that source peer becomes F/c_i from (3). Since the user can choose one of N source peers with equal probability, the actual average download time in this case becomes

$$\mathbb{E}\{T\} = \frac{1}{N}\sum_{i=1}^{N}\frac{F}{c_i} = \frac{F}{H(\vec{c})}, \quad (6)$$

where $H(\vec{c})$ is the harmonic mean of $c_1, c_2, \ldots, c_N$ defined by $H(\vec{c}) = [\frac{1}{N}\sum_{i=1}^{N}\frac{1}{c_i}]^{-1}$. Because $A(\vec{c}) \geq H(\vec{c})$ [3], it follows that (6) $\geq$ (5). This implies that the actual average download time in a heterogeneous network is always larger than that given by 'the average capacity of the network' as in (5).

To quantify the difference between (6) and (5), we adopt similar techniques as in [26]. Let C be the random variable taking values of $c_1, c_2, \ldots, c_N$ with equal probability, i.e. $\mathbb{P}\{C = c_i\} = 1/N$ for all i. Consider the following Taylor expansion of the function $f(x) = 1/x$ around some point x_0:

$$f(x) \approx f(x_0) + f'(x_0)(x - x_0) + \frac{1}{2}f''(x_0)(x - x_0)^2. \quad (7)$$

Letting $x = C, x_0 = \mathbb{E}\{C\}$ and taking expectation in both sides of (7) give

$$\mathbb{E}\left\{\frac{F}{C}\right\} - \frac{F}{\mathbb{E}\{C\}} \approx \frac{F \cdot \mathrm{Var}\{C\}}{(\mathbb{E}\{C\})^3}. \quad (8)$$

From (8), we see that the difference between the predicted average download time using (1) and the actual average value is governed by two factors, the file size F and the variance of the service capacity, $\mathrm{Var}\{C\}$. First, the actual average download time will be different from (5) if the file is large. Second, more importantly, if the service capacities over different source peers vary over a wide range, the actual download time will be much larger than (5).

B. First Hitting Time of a Correlated Stochastic Process

In this section we show that the expected first hitting time of a 'positively correlated process' is larger than that of an $i.i.d.$ counterpart. Consider a fixed network path between a downloading peer and its corresponding source peer for a file of size F. Let $C(t)$ be a stationary random process denoting the available capacity over that source at time slot t. We will assume that $C(t)$ is positively correlated over time. Then, as before, we can define the download time of a file (or the first hitting time of the process $C(t)$ to reach a level F) as T_{cor}, where the subscript 'cor' means that $C(t)$ is a correlated stochastic process.

Suppose now that we are able to remove the correlations from $C(t)$. Let $C'(t)$ be the resulting process and T_{ind} be the stopping time for the process $C'(t)$ to reach level F, where the subscript 'ind' now means that $C'(t)$ is independent over time. Then, again from Wald's equation, we have

$$\mathbb{E}\{T_{ind}\} = \frac{F}{\mathbb{E}\{C'(t)\}} = \frac{F}{\mathbb{E}\{C(t)\}}.$$

First, as introduced earlier, consider the case that $C(t)$ is 100% correlated over time, i.e., $C(t) = C$ for some random variable C for all t. Then, the download time T_{cor} becomes $T_{cor} = F/C$ assuming an equality in (2). Hence, from Jensen's inequality, we have

$$\mathbb{E}\{T_{cor}\} = F\mathbb{E}\left\{\frac{1}{C}\right\} \geq \frac{F}{\mathbb{E}\{C\}} = \mathbb{E}\{T_{ind}\},$$

i.e., the average first hitting time of an 100% correlated process is always larger than that of an $i.i.d.$ counterpart. In order to characterize any degree of positive correlations in $C(n)$, we need the following definition [25]:

[3]The arithmetic mean is always larger than or equal to the harmonic mean, where the equality holds when all c_i's are identical.

Definition 1: Random variables $X_1, X_2, \ldots, X_n$ are said to be 'associated' if for all increasing functions f and g

$$\mathbb{E}\left\{f(\vec{X})g(\vec{X})\right\} \geq \mathbb{E}\{f(\vec{X})\}\mathbb{E}\{g(\vec{X})\} \tag{9}$$

where $\vec{X} = (X_1, \ldots, X_n)$, and we say h is an increasing function if $h(x_1, \ldots, x_n) \leq h(y_1, \ldots, y_n)$ whenever $x_i \leq y_i$ for $i = 1, \ldots, n$. □

Relation (9) characterizes the positive dependence among the random variables $X_1, X_2, \ldots, X_n$. In words, if some of them become larger, then the other random variables are also likely to be larger. Note that (9) implies positive correlations in $C(t)$ by setting $f(\vec{X}) = X_i$ and $g(\vec{X}) = X_j$. Definition 1 can be generalized to a stochastic process as follows.

Definition 2: The stochastic process $\{X(t), t = 1, 2, \ldots\}$ is said to be associated if for all k and $t_1, \ldots, t_k$, the random variables $X(t_1), \ldots, X(t_k)$ are associated. □

In fact, the set of associated processes comprises a large class of processes. Perhaps, the most popular example is of the following type:

Theorem 4.3.13. in [27]: Let $\{X(t)\}$ be a stochastic process with static space $S = \mathbb{R}^d$ of the form

$$X(t+1) = \varphi(X(t), Z(t)), \quad \text{for} \quad t = 0, 1, \ldots \tag{10}$$

If the $\{Z(t)\}$ are mutually independent and independent of $X(0)$, then $\{X(t)\}$ is associated if $\varphi(x, z)$ is increasing in x. □

Stochastic processes of the form (10) constitute large portion of Markov processes. For example, any auto-regressive type model with positive correlation coefficient can be written in the form of (10). Specifically, for an AR-1 sequence $X(t)$ defined by

$$X(t+1) = \rho X(t) + b\xi(t),$$

where $0 < \rho < 1$ and $\xi(t)$ $(t = 0, 1, \ldots)$ is a sequence of $i.i.d.$ random variables and independent of $X(0)$, we can write $X(t+1) = \varphi(X(t), \xi(t))$ where $\varphi(x, \xi) = \rho x + b\xi$. Since φ is increasing in x, the process $\{X(t)\}$ is associated.

We now present our theorem.

Theorem 1: Suppose that $\{C(t), t \geq 1\}$ is associated. Then, we have

$$\mathbb{E}\{T_{cor}\} \geq \mathbb{E}\{T_{ind}\}.$$

Proof: First, for any given k, we set $f(\vec{C}) = C(k)$ and $g(\vec{C}) = 1_{\{C(1)+\cdots+C(k)>F\}}$, where $\vec{C} = (C(1), C(2), \ldots, C(k))$. Note that both functions f and g are increasing. Observe that

$$\{T_{cor} < k\} \equiv \{C(1) + \cdots + C(k) > F\}. \tag{11}$$

Thus, we have, for any k,

$$\begin{aligned}
\mathbb{E}\left\{C(k)1_{\{T_{cor}<k\}}\right\} &= \mathbb{E}\left\{C(k)1_{\{C(1)+\cdots+C(k)>F\}}\right\} \\
&= \mathbb{E}\left\{f(\vec{C})g(\vec{C})\right\} \geq \mathbb{E}\{f(\vec{C})\}\mathbb{E}\{g(\vec{C})\} \qquad (12)\\
&= \mathbb{E}\{C(k)\}\mathbb{P}\left\{C(1) + \cdots + C(k) > F\right\} \\
&= \mathbb{E}\{C\}\mathbb{P}\left\{T_{cor} < k\right\}, \qquad (13)
\end{aligned}$$

where the inequality in (12) follows since $C(t)$ is associated, and (13) is from the stationarity of $C(k)$ in k and (11). Since $\mathbb{E}\{C(k)\} = \mathbb{E}\{C(k)1_{\{T_{cor}<k\}}\} + \mathbb{E}\{C(k)1_{\{T_{cor}\geq k\}}\}$, it follows that

$$\mathbb{E}\left\{C(k)1_{\{T_{cor}\geq k\}}\right\} \leq \mathbb{E}\{C\}\mathbb{P}\{T_{cor} \geq k\}. \tag{14}$$

Now, let us assume that an equality holds in the definition of T_{cor} (see (2)). Then, we have

$$\begin{aligned}
F &= \mathbb{E}\left\{\sum_{k=1}^{T_{cor}} C(k)\right\} = \mathbb{E}\left\{\sum_{k=1}^{\infty} C(k)1_{\{T_{cor}\geq k\}}\right\} \\
&= \sum_{k=1}^{\infty} \mathbb{E}\left\{C(k)1_{\{T_{cor}\geq k\}}\right\}. \qquad (15)
\end{aligned}$$

Substituting (14) into (15) yields

$$\begin{aligned}
F &\leq \sum_{k=1}^{\infty} \mathbb{E}\{C\}\mathbb{P}\{T_{cor} \geq k\} \\
&= \mathbb{E}\{C\}\sum_{k=1}^{\infty} \mathbb{P}\{T_{cor} \geq k\} = \mathbb{E}\{C\}\mathbb{E}\{T_{cor}\}.
\end{aligned}$$

Thus, we have

$$\mathbb{E}\{T_{cor}\} \geq \frac{F}{\mathbb{E}\{C\}} = \mathbb{E}\{T_{ind}\}.$$

This completes the proof. ■

Theorem 1 states that the average download time of a file from a source peer with correlated service capacity (in the sense of association defined in (9)) is always larger than that of an $i.i.d.$ counterpart. In the subsequent section, we show the relationship between the degree of correlation of a process and the average first fitting time of that process, and illustrate how much $\mathbb{E}\{T_{cor}\}$ can be larger than $\mathbb{E}\{T_{ind}\}$. From previous discussions, we know that in general the average download time, $\mathbb{E}[T]$, should be calculated using $\mathbb{E}[F/C(t)]$ rather than the commonly used $F/\mathbb{E}[C(t)]$.

C. First Hitting Time and Degree of Correlation

To illustrate the relationship between the average download time and the degree of correlation in the available bandwidth $C(n)$, assume that $C(t)$ is given by a stationary first-order autoregressive process (AR-1), i.e.,

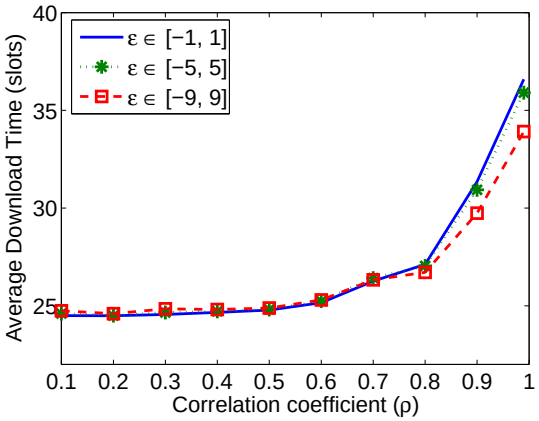
$$C(t+1) = \rho \cdot C(t) + \epsilon(t) + \alpha. \tag{16}$$

Here, $\epsilon(t)$ is a sequence of $i.i.d.$ random variables with zero mean, which represents a noise term of the process. Then, from the stationarity of the process, we get

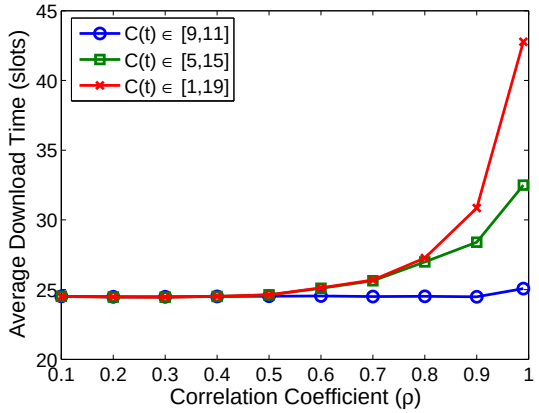
$$\mathbb{E}\{C(t)\} = \mu = \alpha/(1-\rho). \tag{17}$$

We vary the constant α such that the average capacity is always fixed to $\mathbb{E}\{C(t)\} = \mu = 10$ under different ρ. Since the available bandwidth cannot be negative, we limit the range of $C(t)$ such that $C(t) \in [0, 20]$, while keeping the same mean. The file size is $F = 250$ and the noise term, $\epsilon(t)$, is chosen to be uniformly distributed over $[-1, 1]$, $[-5, 5]$, and $[-9, 9]$ to see how the noise term affects the average download time.

Remark 1: The choice of the autoregressive process is for the sake of presentation, not to actually reflex the real fluctuation in an end-to-end available bandwidth in real world.



(a) under different noise term $\epsilon(t)$ in (16).



(b) under different range for $C(t)$.

Fig. 2. Relationship between the average download time and different degrees of correlation ρ.

It is easy to generate AR-1 process with the same mean but different correlation structures. Similar results can be obtained if the AR-1 process is replaced by other processes with more complicated correlation structures. □

Figure 2 (a) shows the relationship between the average download time and the degree of correlation of the process (16) for different ρ and $\epsilon(t)$. As the degree of correlation increases, the average download time increases. In particular, for a heavily correlated process, the average download time can be about 40% larger than that for a uncorrelated or weakly correlated process, regardless of different noise terms. In other words, the long term variation in the service capacity is the main determining factor of the average download time, and the short-term random noise in the service capacity, such as the one caused by TCP congestion control mechanism over short time scales (RTTs), does not have significant impact on the average download time.

To see the impact of the variance of $C(t)$ itself, we restrict the range of $C(t)$ to some fixed interval. For example, $C(t) \in [9, 11]$ means that we set $C(t) = 9$ whenever it becomes smaller than 9 and $C(t) = 11$ when larger than 11. Figure 2 (b) shows the relationship between the average download time and the degree of correlation of $C(t)$ under different variation range for $C(t)$. When the range of fluctuation of $C(t)$ gets smaller, the download time is less affected by the correlation of the process. This is well expected since the process $C(t)$ fluctuates only within $[9, 11]$ and thus behaves more like a constant process. In contrast, when the range for $C(t)$ is large, the impact of correlation becomes apparent as shown in Figure 2(b).

In real data networks, the available capacity of a connection typically shows wild fluctuation; it becomes very low when congestion occurs, and it can reach up to the maximum link bandwidth when things go well. In addition, as technology advances, people are getting links of higher and higher speed, hence the range of available capacity fluctuation is also likely to increase. Therefore, it is very important to consider the effect of correlation in capacity over time when we calculate the average download time of a file transfer.

IV. Minimizing Average Download Time Over Stochastic Channels

Intuitively, if a downloader relies on a single source peer for its entire download, it risks making an unlucky choice of a slow source resulting in a long download. Since the service capacity of each source peer is different and fluctuates over time, utilizing different source peers either simultaneously (parallel downloading) or sequentially within one download session would be a good idea to diversify the risk. Parallel downloading improves the performance by reducing the file size over the 'worst' source peer and also may increase the service capacity one receives from the network by utilizing 'unused' capacities of other source peers. If a downloader utilizes one source peer at a time, switching around seems to be a good strategy to avoid the 'bad' source peer. Now, the question is, "What is the criterion for switching, i.e., is it chunk-based or time-based?" In this section we will analyze the performance of (i) parallel downloading, (ii) random chunk-based switching, and (iii) random time-based (periodic) switching.

Different strategies have different impact on the average download time of each peer, which may result in different system dynamics as well, e.g., how fast a downloader can start to contribute (become a source peer) or how fast a peer leaves the system after finishing download. If there is no peer leaving the system and all peers are willing to share after they complete their download (either the entire file or a chunk), the aggregate service capacity in the system keeps increasing as time goes on because the number of source peers continuously grows. In this case, the dynamics in the increase of aggregate service capacity becomes the dominent factor in the average download time for each peer. On the other hand, if no peer is willing to share after download, the aggregate capacity will then eventually drop to zero, thus throttling all the performance metrics. In reality, however, the P2P network will reach a steady-state at some point in which the peer arrivals and departures are balanced and the aggregate service capacity remains around some constant with little variation as shown in [3]. This suggests that the number of source peers in the system will also be around some constant with little fluctuation in the steady-state. In this paper, we are mostly

interested in the impact of stochastic variations of capacities on the average download time of each peer in the steady-state, rather than in the impact of sources-downloaders dynamics in the transient period, which is beyond the scope of this paper.

Before we start our analysis, we have the following assumptions:

(i) The service capacity of a source is constant within one time slot.
(ii) Each downloader selects its source independently.
(iii) Each downloader makes blind choice, i.e. the sources are randomly chosen uniformly over all available sources.

Assumption (i) is reasonable since it is expected that there is no major event that triggers dramatic fluctuation in the service capacity within a short period of time. There may be small short-term fluctuations, on the order of seconds, in the service capacity due to the nature of the network protocol, such as TCP congestion window changes, or OS interrupt handling, etc. These changes however do not impose serious impact on the service capacity. Thus, we are not interested in such small short-term variations, but are more interested in the fluctuation on a longer time scale caused by change in the number of connections at a source peer or change in network congestion status, which all usually last for longer time (say, minutes to hours). We have the assumption (ii) because it is impractical for any downloader to know how other downloaders choose their source peers in the network. Hence the downloader cannot not make its source selection decision based on other downloaders' decision. Assumption (iii) is based on the fact that the downloader does not know the service capacity of each source peer a priori. Although some protocols require peers to broadcast information about its physical connection speed, it is hard to tell the "true" instant service capacity of each source peer due to many factors such as competition among other peers, changing workload of the source peer, or the network congestion status. Therefore, a simple way to select a source peer is just to make blind choice.

A. Effect of Parallel Downloading

Parallel downloading is one of the most noticeable way to reduce the download time [28], [16]. If the file F is divided into k chunks of equal size, and k simultaneous connections are used, the capacity for this download session becomes $c_1 + c_2 + \ldots + c_k$, where c_i is the service capacity of i^{th} connection. Intuitively, this parallel downloading seems to be optimal in all cases. But, it is worth noting that the download time for parallel downloading is given by $\max\{t_1, t_2, \ldots, t_k\}$ rather than $F/(c_1 + c_2 + \ldots + c_k)$, where t_i is the download time of a chunk over i^{th} connection. This is because the chunk that takes the longest time to complete determines the entire download session.

To illustrate that parallel downloading is better than single download, we consider the following simple example. Assume that there are only two source peers in the network, and c_1, c_2 are the service capacities of the two source peers. Without loss of generality, we assume that $c_1 \leq c_2$. If parallel downloading is used for downloading a file of size F from the network, the download time T_p is given by

$$T_p = \max\left\{\frac{F}{2c_1},\ \frac{F}{2c_2}\right\} = \frac{F}{2c_1}.$$

For the case of single download, the average download time $\mathbb{E}\{T_s\}$ is

$$\mathbb{E}\{T_s\} = \frac{1}{2}\left(\frac{F}{c_1} + \frac{F}{c_2}\right) > \mathbb{E}\{T_p\} = T_p.$$

Now, given that parallel download is better than single download, one may ask whether it is as good as the predicted value in (1). To answer this, let's recall the two-source peers example. From (1), the predicted download time is

$$\mathbb{E}\{T\} = \frac{F}{A(\vec{c})} = \frac{2F}{c_1 + c_2}.$$

An easy calculation shows $\mathbb{E}\{T\} < \mathbb{E}\{T_p\}$ if $c_2 > 3c_1$. Thus, even in the network with one user, parallel downloading may not reduce the download time to the predicted value in all cases. Instead, the performance of parallel download depends upon the distribution of the underlying service capacities and could be much worse than the ideal case, $F/A(\vec{c})$. Indeed, it is shown in [15] that if we can make the chunk-size proportional to the service capacity of each source peer, parallel downloading can yield the optimal download time. But such scheme requires global information of the network. One of our goals is to find a simple and distributed algorithm with *no global information* such that the value in (1), or $F/A(\vec{c})$, can be achieved under almost all network settings.

We have already seen that parallel downloading may not achieve $F/A(\vec{c})$ even when there is only one user in the network. Further, it is shown [28], [16] that in a multi-user network, maintaining just a few parallel connections, say, 4 to 6, is better than having parallel connections to all possible source peers. Hence, if there is an algorithm that can increase the performance of *each individual* connection among such a few parallel connection, then each individual user may achieve the download time predicted by (1) or even better.

B. Random Chunk-based Switching

In the random chunk-based switching scheme, the file of interest is divided into many small chunks just as in the parallel download scheme. A user downloads chunks sequentially one at a time. Whenever a user completes a chunk from its current source peer, the user randomly selects a new source peer and connects to it to retrieve a new chunk. In this way, if the downloader is currently stuck with a bad source peer, it will stay there for only the amount of time required for finishing one chunk. The download time for one chunk is independent of that of the previous chunk. Intuitively, switching source peers based on chunk can reduce the correlation in service capacity between chunks and hence reduce the average download time. However, there is another factor that has negative impact on the average download time, the spatial heterogeneity.

First, suppose that there is no temporal correlation in service capacity and Wald's equation holds for each source peer. A file of size F is divided into m chunks of equal size, and let t_j be the download time for chunk j. Then, the total download time,

T_{chunk}, is $T_{chunk} = \sum_{j=1}^{m} t_j$. Since each chunk randomly chooses one of N source peers (with equal probability), the expected download time will be

$$\mathbb{E}\{T_{chunk}\} = \sum_{j=1}^{m} \frac{1}{N} \sum_{i=1}^{N} \frac{F/m}{c_i} = \frac{F}{H(\vec{c})}. \qquad (18)$$

The result in (18) is identical to the download time given in (6) where a user downloads the entire file from an initially randomly chosen source peer. In other words, the chunk-based switching is still subject to the 'curse' of spatial heterogeneity. While there is no benefit of the chunk-based switching from the average download time point of view, it turns out that this scheme still helps reduce the variance of the download time under a relatively smaller number of users by diversifying the risk with smaller chunks. (See Figure 5(b) in Section V.)

In the chunk-based switching, if we get stuck in a source peer with very low service capacity, downloading a fix amount of bytes from that source peer may still take a long time. We could avoid this long wait by making the size of each chunk very small, but this then would cause too much overhead associated with switching to many source peers and integrating those many chunks into a single file. Therefore, instead of waiting until we finish downloading a fixed amount of data (chunk or file), we may want to get out of that bad source peer after some fixed amount of time. In other words, we randomly switch based on time. In the subsequent section, we will investigate the performance of this random switching based on time and show that it outperforms all the previous schemes in the presence of heterogeneity of service capacities over space and temporal correlations of service capacity of each source peer.

C. Random Periodic Switching

In this section, we analyze a very simple, distributed algorithm and show that it effectively removes correlations in the capacity fluctuation and the heterogeneity in space, thus greatly reducing the average download time. As the algorithm will be implemented at each downloading peer in a distributed fashion, without loss of generality, we only focus on a single downloader throughout this section.

In our model, there are N possible source peers for a fixed downloader. Let $C_i(t)$ ($t = 0, 1, 2, \ldots$ and $i = 1, 2, \ldots, N$) denote the available capacity during time slot t of source peer i. Let $U(t) \in \{1, 2, \ldots, N\}$ be a source selection function for the downloader. If $U(t) = i$, this indicates that the downloader selects path i and the available capacity it receives is $C_i(t)$ during the time slot t. We assume that each $C_i(t)$ is stationary in t and $C_i(t)$ of different source peers $i = 1, 2, \ldots, N$ are independent.[4] We however allow that they have different distributions, i.e., $\mathbb{E}\{C_i(t)\} = c_i$ are different for different i (heterogeneity). For any given i, the available capacity $C_i(t)$ is correlated over time t. As before, when each connection has the same probability of being chosen, the average service capacity of the network is given by $A(\vec{c}) = \frac{1}{N}\sum_{i=1}^{N} c_i$.

In this setup, we can consider the following two schemes: *(i) permanent connection*, and *(ii) random periodic switching*. For the first case, the source selection function does not change in time t. When the searching phase is over and a list of available source peers is given, the downloader will choose one of them randomly with equal probability. In other words, $U(t) = U$ where U is a random variable uniformly distributed over $\{1, 2, \ldots, N\}$. For example, if the downloader chooses u ($u \in \{1, 2, \ldots, N\}$) at time 0, then it will stay with that source peer *permanently* ($U(t) = u$) until the download completes.

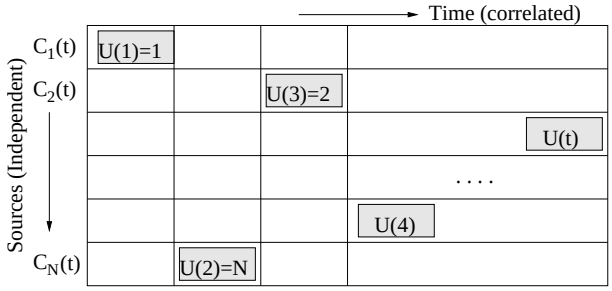


Fig. 3. The operation of source selection function $U(t)$ for random periodic switching

For the random periodic switching, the downloader randomly chooses a source peer at each time slot, independently of everything else. In other words, the source selection function $U(t)$ forms an $i.i.d.$ sequence of random variables, each of which is again uniformly distributed over $\{1, 2, \ldots, N\}$. Figure 3 illustrates the operation of the source selection function $U(t)$ for random periodic switching. In this figure, source 1 is selected at time 1, source N is selected at time 2, and so on.

Let us define an indicator function

$$I_u(t) = \begin{cases} 1, & \text{if } U(t) = u \\ 0, & \text{otherwise.} \end{cases}$$

Then, since $U(t)$ can take values only from $\{1, 2, \ldots, N\}$, the actual available capacity at time t can be written as

$$X(t) = C_{U(t)}(t) = \sum_{u=1}^{N} C_u(t) I_u(t)$$

for both the permanent connection and the random periodic switching strategies. Since each downloader chooses a source peer independently of the available capacity, $U(t)$ is also independent from $C_u(t)$, and so is $I_u(t)$. Note that, from $\mathbb{E}\{I_u(t)\} = 1/N$ for any u, we have

$$\begin{aligned} \mathbb{E}\{X(t)\} &= \sum_{u=1}^{N} \mathbb{E}\{C_u(t) I_u(t)\} \\ &= \sum_{u=1}^{N} \mathbb{E}\{C_u(t)\}\mathbb{E}\{I_u(t)\} = \sum_{u=1}^{N} \frac{c_u}{N} = A(\vec{c}), \end{aligned} \qquad (19)$$

i.e., the average available capacity for the two source selection strategies are the same.

[4]We note that different paths (overlay) may share the same link at the network core, but still, the bottleneck is typically at the end of network, e.g., access network type, or CPU workload, etc. Thus, the independence assumption here is reasonable.

In order to analyze how the two different strategies affect the correlation in $X(t)$, we consider the correlation coefficient of $X(t)$ defined as

$$r(\tau) = \frac{\mathrm{Cov}\{X(t), X(t+\tau)\}}{\mathrm{Var}\{X(t)\}}.$$

Then, we have the following result.

Proposition 1: Let $r_{per}(\tau)$ and $r_{ran}(\tau)$ denote the correlation coefficient of $X(t)$ under the permanent connection and the random periodic switching, respectively. Then, we have

$$r_{ran}(\tau) \quad = \quad \frac{1}{N}\, r_{per}(\tau), \quad \forall\ t \geq 1.$$

□

Proof: Since the average capacity for both strategies remains the same (see (19)), without loss of generality, we can assume that $C_u(t)$ for any source peer u has zero mean by subtracting $\mathbb{E}\{C_u(t)\}$ if necessary. From the independence among different source peers, we have, for any $u \neq v$,

$$\mathbb{E}\{C_u(t) \cdot C_v(t')\} = \mathbb{E}\{C_u(t)\}\mathbb{E}\{C_v(t')\} = 0. \tag{20}$$

Then, the covariance of $X(t)$ becomes

$$\begin{aligned}
&\mathrm{Cov}\{X(t), X(t')\} \\
&\quad = \mathbb{E}\left\{\sum_{u=1}^{N} C_u(t) I_u(t) \cdot \sum_{v=1}^{N} C_v(t') I_v(t')\right\} \\
&\quad = \mathbb{E}\left\{\sum_{u=1}^{N}\sum_{v=1}^{N} C_u(t) C_v(t') I_u(t) I_v(t')\right\} \\
&\quad = \sum_{u=1}^{N}\sum_{v=1}^{N} \mathbb{E}\{C_u(t)C_v(t')\}\mathbb{E}\{I_u(t)I_v(t')\}. \qquad (21)
\end{aligned}$$

From (20), we can rewrite (21) as

$$\sum_{u=1}^{N} \mathbb{E}\{C_u(t)C_u(t')\}\mathbb{E}\{I_u(t)I_u(t')\}. \tag{22}$$

First, consider the case of $t = t'$. Then, it follows that

$$\mathbb{E}\{I_u(t)I_u(t)\} = \mathbb{E}\{I_u(t)\} = \frac{1}{N}.$$

Hence from (22) with $t = t'$, the variance of $X(t)$ is given by

$$\begin{aligned}
\mathrm{Var}\{X(t)\} &= \frac{1}{N}\sum_{u=1}^{N} \mathbb{E}\{C_u(t)C_u(t)\} \\
&= \frac{1}{N}\sum_{u=1}^{N} \mathrm{Var}\{C_u(t)\}, \qquad (23)
\end{aligned}$$

regardless of the strategies for $U(t)$.

Now, consider the case of $t \neq t'$. Under the permanent connection strategy, since $I_u(t) = I_u(t')$ all the time, we get

$$\mathbb{E}\{I_u(t)I_u(t')\} = \frac{1}{N}.$$

On the other hand, for the random periodic switching, we have

$$\mathbb{E}\{I_u(t)I_u(t')\} = \mathbb{E}\{I_u(t)\}\mathbb{E}\{I_u(t')\} = \frac{1}{N^2},$$

since $I_u(t)$ and $I_u(t')$ for $t \neq t'$ are independent.

Finally, set $t' = t+\tau$. Then, from (22) and since the variance of $X(t)$ remains the same for both strategies as in (23), we have $r_{ran}(\tau) = r_{per}(\tau)/N$ and this completes the proof. ■

From Proposition 1, we see that under the random periodic switching strategy, the correlation of $X(t)$ is N times smaller than that of permanent connection strategy. For example, when each downloader has about 10 available source peers ($N = 10$), the correlation coefficient of the newly obtained capacity process under our random periodic switching is no more than 0.1 regardless of the correlations present in the original capacity fluctuation. So, by using our random periodic switching, we can always make the capacity process *very lightly correlated*, or almost independent. From Figure 2, we see that the average download time for a lightly correlated process is very close to that given by Wald's equation. It is thus reasonable to assume that Wald's equation holds for the lightly correlated process $X(t)$ under our random periodic switching strategy. Specifically, if we define T_{ran} as the download time for a file of size F under the random periodic switching, we have

$$\begin{aligned}
F = \mathbb{E}\left\{\sum_{t=1}^{T_{ran}} C_{U(t)}(t)\right\} &= \mathbb{E}\{T_{ran}\}\mathbb{E}\{C_{U(t)}(t)\} \\
&= \mathbb{E}\{T_{ran}\}\mathbb{E}\left\{\mathbb{E}\left\{C_{U(t)}(t) \mid U(t)\right\}\right\} \\
&= \mathbb{E}\{T_{ran}\}\frac{1}{N}\sum_{u=1}^{N}\mathbb{E}\{C_u(t)\} \\
&= \mathbb{E}\{T_{ran}\}\frac{1}{N}\sum_{u=1}^{N} c_u = \mathbb{E}\{T_{ran}\}A(\vec{c}). \qquad (24)
\end{aligned}$$

We then have the following comparison result between the permanent connection and periodic switching.

Proposition 2: Suppose that the process $C_u(t)$ for each u is associated (i.e., it is correlated over time t). Let T_{per} and T_{ran} be the download time for the permanent connection and for the random periodic switching, respectively. Then, we have

$$\mathbb{E}\{T_{per}\} \geq \mathbb{E}\{T_{ran}\}.$$

□

Proof: Assume that the file size is F. Since $C_u(t)$ is associated, from Theorem 1, we have

$$\mathbb{E}\{T_{per}|U = u\} \geq \frac{F}{\mathbb{E}\{C_U(t)|U = u\}}, \tag{25}$$

for any given source peer u. Observe now that

$$\begin{aligned}
\mathbb{E}\{T_{per}\} = \mathbb{E}\{\mathbb{E}\{T_{per} \mid U\}\} &\geq \mathbb{E}\left\{\frac{F}{\mathbb{E}\{C_U(t)|U\}}\right\} \qquad (26) \\
&\geq \frac{F}{\mathbb{E}\{\mathbb{E}\{C_U(t)|U\}\}} = \frac{F}{\mathbb{E}\{C_U(t)\}} \qquad (27) \\
&= \frac{F}{A(\vec{c})} = \mathbb{E}\{T_{ran}\}, \qquad (28)
\end{aligned}$$

where (26) is from (25), (27) is from Jensen's inequality and the convexity of a function $f(x) = 1/x$ for $x > 0$, and (28) is from (24). This completes the proof. ■

Proposition 2 shows that our random periodic switching strategy will always reduce the average download time compared to the permanent strategy and that the average download

time under the random periodic switching is given by $F/(\vec{c})$ (see (27)). Note that this was made possible since the random periodic switching removes the negative impact of both the heterogeneity and the correlations. In addition, our algorithm is extremely simple and does not require any information about the system.

D. Discussion

So far, we have analyzed the performance of three different schemes that utilize the spatial diversity of the network to improve per-user performance in terms of the average download time. We have considered (i) parallel downloading, (ii) random chunk-based switching, and (iii) random periodic switching. The parallel downloading may perform well if the capacity of each possible source peer is known so as to allocate larger chunks to faster connections and smaller chunks to slower connections. But this method is not practical as one cannot know a priori the service capacity of all source peers. In addition, the service capacity is stochastically fluctuating all the time, and our analysis show that the performance of parallel downloading depends much upon the heterogeneity of the service capacities in different source peers if the chunks are equal in size.

Many P2P applications nowadays use chunk-based file transfer with equal chunk size. As mentioned earlier, the benefit of chunk-based switching is to speed up the conversion from downloading peers to uploading peers and thus indirectly affect the average download time. But, in terms of reducing the average download time directly, it does not help much. Random chunk-based switching may reduce the correlations in the service capacity, but it still cannot eliminate the effect of spatial heterogeneity in different source peers.

In current practice, the chunk based transfer and the parallel download are often combined. Taking BitTorrent and Overnet for examples, a file is first divided into 256KB and 9.5MB chunks of equal size, respectively, and then different chunks are downloaded from different source peers simultaneously. However, we separate the analysis of the two strategies to show how each is different in combating spatial heterogeneity and temporal correlations. Please note that we are not trying to compare the performance of parallel downloading with chunk based transfer since they can be easily combined to yield better performance. Rather, we are comparing the performance of the two strategies with our random periodic scheme. Further, we will present the performance comparison of the combined strategy with the random periodic scheme in Section V-B.

The idea of time-based switching scheme is in fact not new. Such strategy has been implemented in BitTorrent [21] but with some other purpose in mind. In BitTorrent application, by using its optimistic choking/unchoking algorithm, a peer changes one of its servicing neighbors with the lowest upload capacity every 10 seconds in hope to find some peers offering higher service capacity. However, the idea of switching source peer periodically in the BitTorrent's optimistic choking/unchoking algorithm is to discover new potential sources rather than to explicitly remove the negative impact of temporal correlations and spatial heterogeneity in service capacity. To the best of our knowledge, we are the first to point out that the random periodic switching gives us the average download time of $F/A(\vec{c})$, while all the other schemes considered so far yield larger average download time.

Our study leads us to believe that the random switching decision should be based on time rather than 'bytes' because we are interested in the download time, not the average capacity itself. Indeed, any algorithm based on bytes or a fixed amount of data will suffer the *curse of bad source peer* in that it has to wait until that amount of data is completely received from the 'bad' source peer. On the other hand, when the decision is based on time, we don't need to wait that long as we can jump out of that source peer after a fixed amount of time (one period).

V. Numerical Results

In this section we provide numerical results to support our analysis and compare the performance of the four schemes for file download under various network configurations. In any case, in our configuration, different source peers have different average service capacities, and the service capacity of each source peer is correlated in time. We consider a single downloading peer as well as multiple downloading peers to allow competition among the downloading peers for limited service capacity of each source peer.

A. Single Downloader with Heterogeneous Service Capacities

We first show the impact of both heterogeneity and correlations in service capacities on the average download time when there is a single user (downloader) in the network. There are $N = 4$ source peers in the network, each offering different average service capacities. Let c_i be the average service capacity of source peer i and $\vec{c} = (c_1, c_2, c_3, c_4)$. The average service capacity of the whole network is then $A(\vec{c}) = (c_1 + c_2 + c_3 + c_4)/4$. We change the heterogeneity in service capacity by changing each c_i, while keeping $A(\vec{c}) = 200kbps$ the same. We measure the degree of heterogeneity in term of $\delta = \sqrt{\mathrm{Var}\{\vec{c}\}}/A(\vec{c})$, the normalized standard deviation. Table I shows the different settings used in our simulation in this subsection.

	1	2	3	4	5	6	7	8
c_1	185	170	140	110	80	50	35	20
c_2	195	190	180	170	160	150	145	140
c_3	205	210	220	230	240	250	255	260
c_4	215	230	260	290	320	350	365	380
δ	0.05	0.11	0.22	0.33	0.45	0.56	0.61	0.67

TABLE I
AVERAGE SERVICE CAPACITY OF EACH SOURCE PEER UNDER DIFFERENT CONFIGURATIONS.

To demonstrate the impact of correlation in each fixed source peer, we use a class of AR-1 random processes to model the stochastic fluctuation in the service capacity. It is reasonable to assume that if the average service capacity is large, the service capacity is more likely to fluctuate over a wider range. For instance, for a high-speed source peer(e.g., 1Mbps), the actual service capacity of the end-to-end session

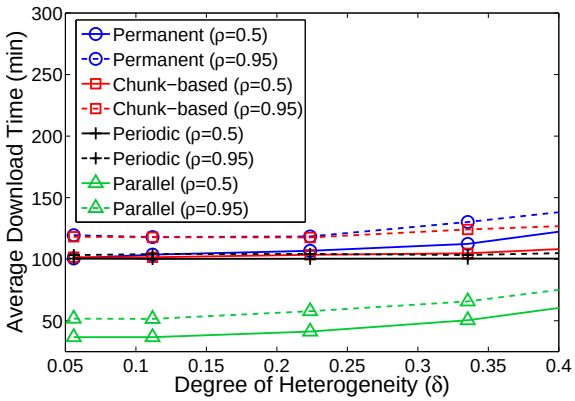
may drop down to somewhere around 50kbps and stays there for a while due to network congestion or limited CPU resources at the source peer. In this regard, we assume that the amount of fluctuation in $C_i(t)$ is proportional to its mean value c_i. Specifically, for source peer i, we set $\epsilon_i(t)$ in (16) to be uniformly distributed over $[c_i - \theta_i, c_i + \theta_i]$ where θ_i is chosen such that $\sqrt{\text{Var}\{C_i(t)\}}/\mathbb{E}\{C_i(t)\}$ remains the same for all i.

In our simulation, the length of each time slot (one period) is chosen to be 5 minutes. We set the file size to 150MB, which is the typical size of some small video clips or multimedia files. As the average service capacity (of the network) is 200kbps, we set the chunk-size for chunk-based switching to be 7.5MB ($=$ 200kbps $\times$ 5 minutes). The purpose of simulating the chunk-based switching is to show the impact of switching based on "data size", hence we choose 7.5MB to allow fair comparison with the random periodic switching with 5 minute period. We will show the performance of using smaller chunk size later in Section V-B.

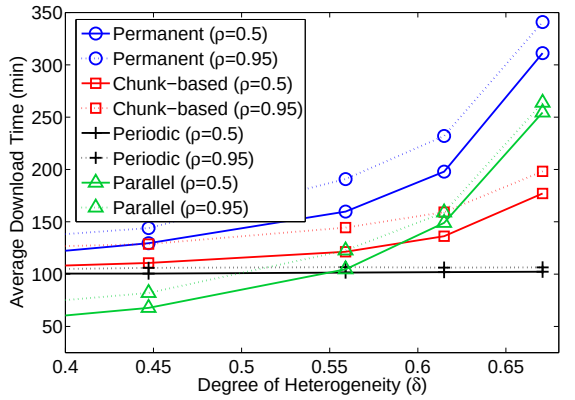
We consider all three download strategies discussed so far in comparison with permanent connection. For permanent connection, the user initially chooses one of four sources randomly and stays there until the download completes. For chunk-based switching, the user switches to a new randomly selected source peer whenever a chunk is completed. Although we simulate the system as a discrete time system, the user is allowed to switch to a new source peer anytime within a time slot whenever it finishes the current chunk. For parallel download, the file is divided into 4 equal-sized pieces and the downloading peer connects to all 4 source peer and download each piece from each source peer simultaneously. Finally, for periodic switching, a user switches to a new randomly chosen source peer every 5 minute to further download the remaining parts of the file.

Figures 4 (a)–(b) show the average download time vs. the degree of heterogeneity in the average service capacities (δ) when there is a single downloader in the network. Dashed lines are for strong correlations ($\rho = 0.95$) and solid lines represent the case of light correlations ($\rho = 0.5$). In Figure 4(a), when the degree of heterogeneity is small, all three single-link download strategies (permanent, chunk-based, periodic) under light correlations perform the same. This is well expected since the service capacities of all source peers are almost $i.i.d.$ over space and time, so switching doesn't make any difference and the average download time becomes $F/A(\vec{c})$ = 150MB/200kbps = 100 minutes, as commonly used in practice. On the other hand, when there exists strong correlations in the service capacity, the download time is longer for all strategies except the periodic switching. For example, when $\delta = 0.1$, the correlation alone can cause more than 20% of increase in the average download time. Thus, when the network is more like homogeneous (i.e., small δ), the temporal correlation in the service capacity of each source peer becomes a major factor that renders the average download time longer. However, the average download time remains the same under the random periodic switching.

Figure 4 (a) also shows the performance of parallel downloading. Intuitively, parallel downloading should perform better than single link downloading because (i) it utilizes more than one link at the same time and (ii) if the connection is poor, parallel downloading reduces the amount of data getting through that bad source peer. Since there is only a single user, it utilizes all the service capacity the network can provide ($c_1 + c_2 + c_3 + c_4$). In this case, the average download time should be 150MB/($c_1 + c_2 + c_3 + c_4$) = 150MB/800kbps $\approx$ 25 minutes. We see from Figure 4(a) that parallel downloading can actually achieve the performance close to our expectation when the service capacities of different source peers are close to $i.i.d.$ Still, parallel downloading is prone to the negative effect of correlations.



(a) Low degree of heterogeneity: $0 < \delta < 0.4$



(b) High degree of heterogeneity: $0.4 < \delta < 0.7$

Fig. 4. Average download time vs. degree of heterogeneity under different download strategies and different degree of correlations.

As the degree of heterogeneity increases, the average download time sharply increases for all the schemes except the periodic switching. Figure 4 (b) shows this when δ is between 0.4 and 0.7 (see Table I). All but periodic switching suffer from the negative effect of heterogeneity. When both heterogeneity and correlation are high ($\delta = 0.65$ and $\rho = 0.95$), permanent connection takes about 350 minutes to complete the download. This time is about 250 minutes, or 4 hours more than using periodic switching! It is expected that that the performance of parallel downloading degrades fast when there is a large degree of heterogeneity. It is more likely that one of the parallel connections is 'poor' with very small capacity. Thus, even though the size of chunk (37.5MB) is smaller than the whole

file (hence reducing the risk of staying with the bad source peer for too long), this is still not as good as the idea of averaging capacities all the time, as used in the periodic switching. We note that temporal correlations still negatively affect in all these three schemes. However, it should be pointed out that the random periodic switching performs the same *regardless of heterogeneity and correlations*, and in fact it outperforms all the other schemes when the network is heterogeneous with a wide range of service capacities as in the current network.

B. Multiple Downloaders with Competition

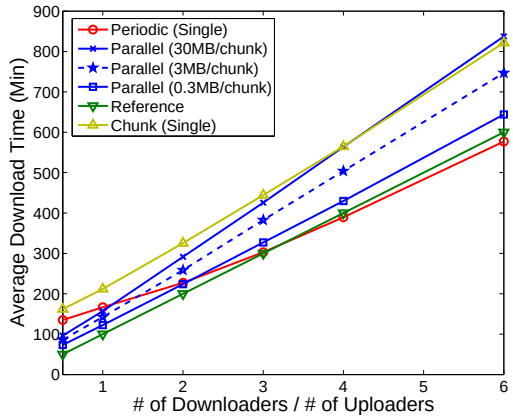
In this section, we consider the performance of different download strategies under a multi-user environment. In our multi-user setting, we set the number of source peers to $N = 100$. The source peers are divided into 4 groups and each source peer within the same group will have the same average service capacity. In reality, the service capacity of each source peer may vary a lot, much greater than the ones that are presented in Table I. We choose the service capacity of the four groups as 1Mbps, 500Kbps, 100Kbps, and 50Kbps, representing typical capacities of LAN, cable, DSL, and modem connections, respectively. In contrast to the setting in the previous section, each group now may consist of different number of source peers to reflect a more realistic distribution of service capacity. We choose the number in each group as 10, 5, 65, 20, respectively. This is to reflect the situation in real world that only few source peers have very high service capacity while most others have the capacity of typical DSL (100Kbps) lines or slower modems. The average service capacity of the network is then $\mathbb{E}\{C\} = 1M \cdot 0.1 + 500K \cdot 0.05 + 100K \cdot 0.65 + 50K \cdot 0.2 = 200$Kbps. The degree of heterogeneity (δ) in our setting is $\delta = 0.99$. The fluctuation in the service capacity is represented by AR-1 process with correlation coefficient of each source peer set to 0.9. We want to see the performance of different strategies under the impact of spatial heterogeneity and temporal correlation.

In our simulation, service capacity of a source peer is equally divided among all the users connected to that source peer. The effect of dividing capacity among users gives us an idea of how different strategies will perform when users compete for limited resources in the network. To represent the level of competition, we use the *downloader-source* ratio, i.e. the ratio between the number of users (downloading peers) to the number of source peers. Since the service capacity of a source peer is equally among the users the source peer serves, we can expect that the service capacity of the system is equally divided among all users as well. Hence, the *average per-user service capacity* can be calculated as the average system service capacity divided by the downloader-source ratio. For example, if the number of users is 200, then the downloader-source ratio 2. The average per-user service capacity will then be $200\text{Kbps}/2 = 100\text{Kbps}$.

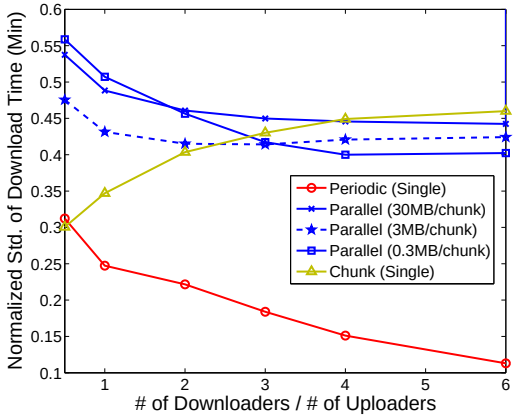
We simulate three strategies. First one is the *combined* strategy of parallel download and the chunk-based transfer. Since we know from [28], [16] that keeping only a small number of parallel active connections is better than maintaining connections to all source peers, we set the number of parallel connections to 5 for all the combined strategies. We vary the chunk size to see its impact on the average download time in conjunction with parallel download. Further, the users are allowed to request the same chunk from different source peers when the number of untransferred chunks is less than the number of active parallel connections. For example, if a user is three chunks away from completing the entire file, s/he can request all three chunks from all currently connected source peers. Although making the same chunk requests to different source peers will reduce the download time for that specified chunk, this is at the expense of some waste of the system resource. Note that we do not allow users to make duplicate requests to all connected source peers for every chunk, as this will waste too much resource. This notion of making requests to different source peers for the same chunk when a user's download is nearly complete has been already implemented in BitTorrent called the "end-game" mode [29]. The second strategy is the random chunk-based switching with a single connection. The chunks size is chosen to be 7.5MB, which is identical to what we used in the previous section to allow fair comparison with the periodic switching. Finally, the third strategy is the random periodic switching. The switching period is still 5 minute, but we reduce the length of the system time slot to 1 min. In this case, there will be capacity variations within each switching period.

Figure 5(a) shows the average download time for the strategies considered so far. The reference line is given by the file size divided by the average per-user service capacity. First, we can clearly see that the periodic switching performs a lot better than the chunk-based switching. We have a reduction of 40% in average download time by using periodic switching. Next, the combined strategies are shown to outperform the chunk-based switching. Note that when the level of competition is low, the combined strategy outperforms both the chunk-based and the periodic switching schemes. This is well expected because parallelism increases the service capacity each user can achieve in an under-utilized network. As the level of competition increases, however, the random periodic switching readily starts to outperform the combined strategy.Further, it is interesting to see that chunk based transfer (7.5 MB per chunk) even outperform parallel downloading using large chunks (30MB per chunk) when the competition in the system is heavy. This is due to the fact that the average download time for parallel downloading is still determined by the slowest link. In a system where there are already many downloading peers, parallelism actually increase the level of competition even more, hence the service capacity of a slow source peer is further divided among its downloading peers.

Another noticeable trend in Figure 5(a) is that the performance of the combined strategy gets better with smaller chunk size. Recall that the users can download the same chunk when there are only several chunks left before the completion of the entire file, so the last few chunks will be transferred over the fastest source peer. This method may reduce the negative impact of spatial heterogeneity a little, but at the price of wasting some of the system resource transferring duplicate chunks. The larger the chunk size, the more waste of recourses in sending the duplicate chunks. In addition, a



(a) Average download time



(b) Standard deviation / average download time

Fig. 5. Performance comparison of different strategies under different levels of competition.

larger chunk is more prone to the spatial heterogeneity as the user downloading that larger chunk will have to wait long if it is from a 'bad' source. Certainly, very small chunk sizes would make the performance of the combined strategy better and approach the reference line. However, *this comes at a cost*; having small chunks means more protocol overheads because more negotiations between downloaders and source peers are required. Take the combined strategy using 0.3MB chunks as an example, a downloader has to make requests for chunks at least 150MB/0.3MB = 500 times in the entire download session. However, the downloader using the periodic switching only needs to make data transfer requests about 120 times in the extreme case (when the downloader-source ratio is 6). From our simulation result, we can see that random periodic switching is the optimal strategy when the network is over-utilized (downloader-source ratio is 3 or higher).

Figure 5(b) shows the normalized standard deviation (standard deviation divided by its mean) of the download time for different strategies as the level of competition varies. Larger value of the normalized standard deviation means that the download time among different users will vary more; some can complete the transfer in a very short period while others have to wait for a long time to complete with high probability. Thus, if there is a large variation in the download time, it is very hard for a user to predict what kind of service s/he will receive. It would be better to have small variations in the download time so that the performance is more predictable and fair. We can clearly see that the periodic switching yields the smallest variation in download time comparing with other strategies we have considered so far.

In summary, the periodic switching not only gives downloaders the minimal average download time in most network configurations and introduces less overhead, but it is fair with more predictable performance as well.

VI. Conclusion

In this paper we have focused on the average download time of each user in a P2P network. With the devastating usage of network resources by P2P applications in the current Internet, it is highly desirable to improve the network efficiency by reducing each user's download time. In contrast to the commonly-held practice focusing on the notion of average capacity, we have shown that both the spatial heterogeneity and the temporal correlations in the service capacity can significantly increase the average download time of the users in the network, even when the average capacity of the network remains the same. We have compared several 'byte-based' (file size based) schemes widely used in practice, including chunk-based file transfer, parallel downloading, as well as their combination, and have shown that all those byte-based schemes are not so effective in reducing the two negative factors that increase the average download time. From our study, it becomes apparent that all P2P algorithms regarding the download time should focus directly on 'time' rather than on 'bytes', and the notion of average service capacity alone is not sufficient to describe each user's average performance in a P2P network.

References

[1] Y. M. Chiu and D. Y. Eun, "Minimizing File Download Time over Stochastic Channels in Peer-to-Peer Networks," in *Proceedings of the 40th Annual Conference on Information Sciences and Systems (CISS)*, Princeton, NJ, March 2006.

[2] D. Qiu and R. Srikant, "Modelling and performance analysis of bittorrent-like peer-to-peer networks," in *Proceedings of ACM Sigcomm*, Aug. 2004.

[3] X. Yang and G. de Veciana, "Service capacity of peer to peer networks," in *Proceedings of IEEE Infocom*, Mar. 2004.

[4] K. P. Gummadi, R. J. Dunn, and S. Saroiu, "Measurement, modeling, and analysis of a peer-to-peer file sharing workload," in *Proceedings of ACM Symposium on Operating Systems Principles (SOSP)*, 2003.

[5] M. Adler, R. Kumar, K. Ross, D. Rubenstein, D. Turner, and D. D. Yao, "Optimal peer selection in a free-market peer-resource economy," in *Proceedings of Workshop on Economics of Peer-to-Peer Systems (P2PEcon)*, Cambridge, MA, Jun. 2004.

[6] ——, "Optimal peer selection for p2p downloading and streaming," in *Proceedings of IEEE Infocom*, Miami, FL, Mar. 2005.

[7] D. S. Bernstein, Z. Feng, and B. N. Levine, "Adaptive peer selection," in *Proceedings of International Workshop on Peer-to-Peer Systems (IPTPS)*, Berkeley, CA, Feb. 2003.

[8] S. G. M. Koo, K. Kannan, and C. S. G. Lee, "Neighbor-selection strategy in peer-to-peer networks," in *Proceedings of IEEE International Conference on Computer Communications and Networks (ICCCN)*, Rosemont, IL, Oct. 2004.

[9] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker, "A scalable content addressable network," UC Berkeley, Berkeley, CA, Tech. Rep. TR-00-010, 2000.

[10] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan, "Chord: A scalable peer-to-peer lookup service for internet applicationss," in *Proceedings of ACM Sigcomm*, 2001.
[11] B. Y. Zhao, L. Huang, A. D. J. J. Stribling, S. C. Rhea, and J. D. Kubiatowicz, "Tapestry: A resilient global-scale overlay for service deployment," *IEEE Journal on Selected Areas in Communications*, vol. 22, no. 1, pp. 41–53, Jan. 2004.
[12] J. Byers, J. Considine, M. Mitzenmacher, and S. Rost, "Informed content delivery across adaptive overlay networks," in *Proceedings of ACM Sigcomm*, 2002.
[13] C. Gkantsidis and P. R. Rodriguez, "Network coding for large scale content distribution," in *Proceedings of IEEE Infocom*, Miami, FL, Mar. 2005.
[14] Intel, "Peer-to-peer content distribution: Using client pc resources to store and distribute content in the enterprise," Intel, Tech. Rep., September 2003. [Online]. Available: http://www.intel.com/it/digital-enterprise/peer-peer-content-distribution.pdf
[15] K. K. Ramachandran and B. Sikdar, "An analytic framework for modelling peer to peer networks," in *Proceedings of IEEE Infocom*, Mar. 2005.
[16] S. Koo, C. Rosenberg, and D. Xu, "Analysis of parallel downloading for large file distribution," in *Proceedings of IEEE International Workshop on Future Trends in Distributed Computing Systems (FTDCS)*, May 2003.
[17] G. B. F. Lo Piccolo, G. Neglia, "The effect of heterogeneous link capacities in bittorrent-like file sharing system," in *IEEE International Workshop on Hot Topics in Peer-to-Peer Systems (HOT-P2P)*, Oct. 2004.
[18] T. Ng, Y. Chu, S. Rao, K. Sripanidkulchai, and H. Zhang, "Measurement-based optimization techniques for bandwidth-demanding peer-to-peer systems," in *Proceedings of IEEE Infocom*, Apr. 2003.
[19] Y. Kulbak and D. Bickson, *The eMule Protocol Specification*, Jan. 2005, "http://leibniz.cs.huji.ac.il/tr/acc/2005/HUJI-CSE-LTR-2005-3_emule.pdf".
[20] R. Sherwood, R. Braud, and B. Bhattacharjee, "Slurpie: A cooperative bulk data transfer protocol," in *infocom*, Hong Kong, Mar. 2004.
[21] B. Cohen, *BitTorrent Protol Specification*, "http://www.bittorrent.com/protocol.html".
[22] S. Saroiu, K. P. Gummadi, and S. D. Gribble, "A measurement study of peer-to-peer file sharing systems," in *Proceegins of ACM Multimedia Computing and Networking (MMCN)*, 2002.
[23] M. Jain and C. Dovrolis, "End-to-end estimation of the available bandwidth variation range," in *Proceedings of ACM Sigmetrics*, Jun. 2005.
[24] N. Hu and P. Steenkiste, "Evaluation and characterization of available bandwidth probing techniques," *IEEE Journal on Selected Areas in Communications*, vol. 21, no. 6, Mar. 2003.
[25] S. M. Ross, *Stochastic Processes*, 2nd ed. New York: John Wiley & Son, 1996.
[26] I. Rhee and L. Xu, "Limitations of equation-based congestion control," in *Proceedings of ACM Sigcomm*, Aug. 2005.
[27] A. Müller and D. Stoyan, *Comparison Methods for Stochastic Models and Risks*. New York, NY: John Wiley & Son, 2002.
[28] C. Gkantsidis, M. Ammar, and E. Zegura, "On the effect of large-scale deployment of parallel downloading," in *Proceedings of IEEE Workshop on Internet Applications (WIAPP)*, Jun. 2003.
[29] B. Cohen, "Incentives build robustness in bittorrent," 2003, "http://www.bittorrent.org/bittorrentecon.pdf".

Yuh-Ming Chiu received his B.S. degree from the Department of Communication Engineering, National Chiaotung University, Taiwan, in 1997, and the M.S. degree from the Department of Electrical Engineering, National Tsinghua University, Taiwan, in 2000. He has been a graduate student in the Department of Electrical and Computer Engineering, North Carolina State University Since 2004. His research interests include queueing analysis and Peer-to-Peer networks.

Do Young Eun received his B.S. and M.S. degree in Electrical Engineering from Korea Advanced Institute of Science and Technology (KAIST), Taejon, Korea, in 1995 and 1997, respectively, and Ph.D. degree in Electrical and Computer Engineering from Purdue University, West Lafayette, IN, in 2003. Since August 2003, he has been an Assistant Professor with the Department of Electrical and Computer Engineering at North Carolina State University, Raleigh, NC. His research interests include network modeling and analysis, congestion control, resource allocation, and ad-hoc/sensor networks. He is a member of Technical Program Committee of IEEE INFOCOM 2005–2007, IEEE ICC 2005, 2006, IEEE Globecom 2005, and IEEE IPCCC 2006. He received the Best Paper Awards in the IEEE ICCCN 2005 and the IEEE IPCCC 2006, and the NSF CAREER Award 2006.