Analysis of Distributed Algorithms to Remove Correlations for Reducing Average Download Time in PEER–TO–PEER Networks

P. Satheesh
Associate Professor
CSE Department
M.V.G.R College of Engineering

B. Srinivas
Assistant Professor
CSE Department
M.V.G.R College of Engineering

M. V. S. Narayana
CSE Department
M.V.G.R College of Engineering

ABSTRACT
The Peer-to-Peer (P2P) networks are widely used for internet file sharing. In general the file download can take minutes or hours depending on the level of network congestion or the service capacity fluctuations. 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 fluctuations in service capacities of a single source peer. We show 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 reduce the file download time. Here, we analyzes a new algorithms called that effectively remove the negative factors of the existing systems i.e. Parallel downloading, Chunk based switching, periodic switching, thus reduce the average download time. Our algorithm removes correlations in the capacity fluctuations and the heterogeneity in space, thus greatly reducing the average download time.

General Terms
Network performance, spatial heterogeneity, average download time, Peers.

Keywords
P2P networks, Peer Selection Strategy, Service Capacity.

1. INTRODUCTION
PEER-TO-PEER (P2P) technology is widely 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. 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].As the measurement study shows [4], the per-user performance in aP2P network may be even worse than that of 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. 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]–[17].

1.1 Limitations of 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 S 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……N). Notice that the actual value of C_i is unknown before the downloading peer actually connects to the source peer. The average service capacity of the network,

$\hat{C}$ is given by $\hat{C} = \sum_{i=1}^{N} C_i/N.$

Intuitively, the average download time, T, for a file of size F would be

$$T = F/\hat{C}. \quad (1)$$

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.

1.2 Impact of Heterogeneity
Suppose that there are two source peers with service capacities of C_1 = 100 and C_2 = 150, 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 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 1 MB, we predict that the average download time is 64 seconds from (1) However, the actual average download time is 1/2(1 MB/100 kbps) + 1/2(1 MB/150 Mbps) = 66.7 seconds! Hence, we see that the spatial heterogeneity actually makes the average download time longer. 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 with the average capacity C_j ($C_j \geq C_i \forall$ i)), as the average download time T_i over source peer would be given by F/C_i. Consider again the previous two-source peer example with C_1 = 100 and C_2 = 150. 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 250 kbps with equal probability, thus giving $E\{C_2(t)\} = C_2 = 150$ 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 (1 MB/50 kbps + 1 MB/250 kbps)/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. Thus it is evident that the average download time is increased due to heterogeneity and temporal correlation of service capacity.

1.3 Our Contribution

The main contribution of this paper is to show that the predicted value given in (1) is actually achievable, regardless of the heterogeneity in service capacity and the temporal correlation of service capacity of single source peer. We have designed and analyzed new distributed algorithms 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. Our dynamic distributed algorithm limits the amount of time each peer spends on a bad source peer thus minimizing the average download time for each user. The rest of the paper is organized as follows. In Section 2, 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 3, 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 4, 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 5 provides simulation results to test our algorithm and compare with others under various network settings, and we conclude our work in Section 6. we conclude our design of algorithm and prove that our scheme greatly reduces the average download time.

2. FACTORS OF AVERAGE DOWNLOAD TIME

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.

2.1 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.). The limitation in the processing power can limit how fast a peer can service others and hence limits the service capacity.

2.2 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.

Fig. 1 shows a typical available end-to-end capacity fluctuation similar to that presented in [23] and [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.

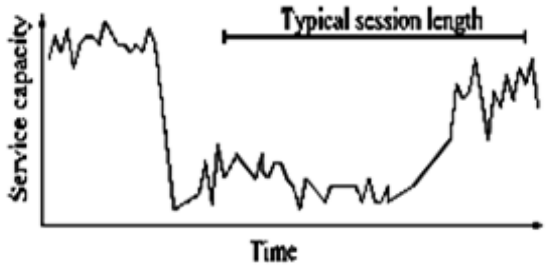

Fig 1: Typical variations in end-to-end available bandwidth based on the results in [23] and [24]. Drastic changes usually occur in the scale of minutes.

3. CHARACTERIZING DOWNLOAD TIME IN A P2P NETWORK

We consider our network as a discrete-time system with each time slot of length. 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) denotes the time-varying service capacity(available end-to-end bandwidth) of a given source peer at time slot t (t=1,2…) over the duration of a download. Then, the download time T for a file of size F is defined as

$$T = \min \{ s > 0 \sum_{t=1}^{s} C(t) \geq F \}. \quad (2)$$

Note that 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..are constant or identically identified distributed, then by assuming an equality in (2), we obtain from Wald's equation [25] that

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

The expected download time, measured in slots, then becomes $E(t) = F/E\{c(t)\}$. Note that (3) also holds if C (t) is constant (over). Thus, when the service capacity is constant or independent and identically distributed. 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.

4. MINIMIZING AVERAGE DOWNLOAD TIME

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. We will analyze the performance of (i) parallel downloading; (ii)

random chunk-based switching; and (iii) random time-based (periodic) switching.

4.1 Effect of Parallel Downloading

Parallel downloading is one of the most noticeable ways to reduce the download time [28], [16]. If the file F is divided into K chunks of equal size, and simultaneous connections are used, the capacity for this download session becomes $c_1 + c_2 + \dots + c_k$, where $C_i$is the service capacity of 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 $\{ t_1 + t_2 + ..+ t_k \}$ rather than $F/(c_1 + c_2 + \dots + c_k)$. 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

$$T_p = \max \left\{ \frac{F}{2C_1}, \frac{F}{2C_2} \right\} = \frac{F}{2C_1}. \qquad (4)$$

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

$$E\{T_s\} = \frac{1}{2}\left(\frac{F}{C_1} + \frac{F}{C_2}\right) > E\{T_p\} = T_p. \qquad (5)$$

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 us recall the two-source peers example. From (1), the predicted download time is

$$E\{T\} = \frac{F}{A(\hat{C})} = \frac{2F}{C_1 + C_2}. \qquad (6)$$

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.

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

$$E\{T_{chunk}\} = \sum_{j=1}^{m} \frac{1}{N} \sum_{i=1}^{N} \frac{F/m}{C_i} = \frac{F}{H(\hat{C})}. \qquad (7)$$

The result in (10) 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. 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.

4.3 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. In our model, there are N possible source peers for a fixed downloader. Let $C_i(t)$ (t=1,2,…. and i=1,2,..N). denote the available capacity during time slot of source peer. 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. In other words, U(t) = U, where U is a random variable uniformly distributed over {1,2,…N}.For the random periodic switching, the downloader randomly chooses a source peer at each time slot, independently of everything else. Fig. 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

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

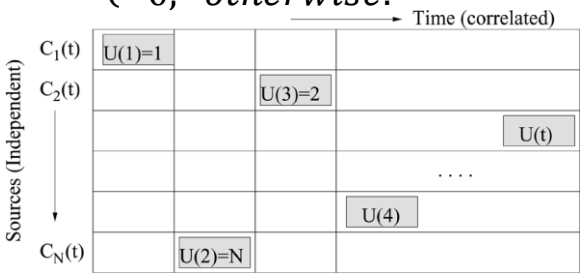

Fig 2: Operation of source selection function U(t) for Random Periodic Switching.

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

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

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 E { I_u (t) } = 1/N. for any u, we have

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

$$= \sum_{u=1}^{N} E\{C_u(t)\}\, E\{I_u(t)\} \qquad (8)$$

$$= \sum_{u=1}^{N} \frac{C_u}{N} = A(\hat{C})$$

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

4.4 Limitations of the existing methods

In Parallel Downloading, if the downloader stuck with any one of the bad source peer over k peers, then it waits for long time until getting the chunk. The download time of this method is the maximum time taken by any of the k peers that take the longest time to complete. The main disadvantage of the Chunk-Based Switching is that if we get stuck in a bad source peer with very low service capacity, downloading a fixed amount of bytes from that source peer may still take a In Random Periodic Switching, the downloader randomly chooses a source peer at each time slot and it may get stuck with bad source peer. So this method cause too much

overhead associated with switching to many source peers and integrating those many chunks into a single file.

4.4.1 Distributed parallel switching algorithm

Distributed parallel switching algorithm has two methods.(i)parallel connection (ii)parallel random chunk based switching. This method outperforms all the previous schemes in the presence of heterogeneity of service capacities over space and fluctuation of service capacity of each source peer. Our two schemes are described below.

4.4.2 Dynamically Distributed Parallel Connection

In this method, the source selection function does not change in fixed time slot t as in Permanent Connection of existing Periodic Switching. But instead of choosing a single source peer, here the downloader chooses multiple fixed k source peers over N possible source peers and it makes permanent connection for the fixed time slot t.

4.4.3 Dynamically Distributed Parallel Random Chunk Based Switching

In this method, the source selection function changes for each randomly selected time slot as in Random Chunk Based Switching of existing Periodic Switching. But instead of choosing a single source peer, here the downloader randomly chooses multiple fixed k source peers over N possible source peers and it makes parallel connection with that k source peers for each randomly selected time slot. So in this method, if the downloader chooses the k source peers over N possible source peers at randomly selected time to, then it will stay with that k source peers permanently until the download completes. After the time slot gets completed, the downloader again performs the 6source selection function to download the remaining file.

5. DESIGN OF DYNAMICALLY DISTRIBUTED PARALLEL CONNECTION

In this method, We designed new Parallel Connection and our algorithm implemented at each downloading peer in a distributed fashion and we focus on a single downloader throughout this method. In our method , there are N possible source peers for a fixed downloader. Let $C_i(t)$ where t=1,2… and i=1,2…k over N possible source peers. Let U(t) is a source selection function for the downloader. If U(t) = i ,then it indicates the ith path and available capacity it receives is $C_i(t)$ during the time slot t. we however aloe that they have different distributions of service capacities and the average service capacity of the network is given as follows,

$$\overrightarrow{A(C)} = \frac{1}{N}\sum_{i=1}^{N} C_i$$

Our new Dynamically Distributed Parallel Permanent Connection scheme has the following modules,

5.1 Downloading Peer Initiation Module

In this Downloading Peer Initiation Module, the user requests the file from the downloading peer and the file is given to the Source Selection Function Module. Fig. 3 illustrates the functionality of the Downloading Peer Initiation Module.

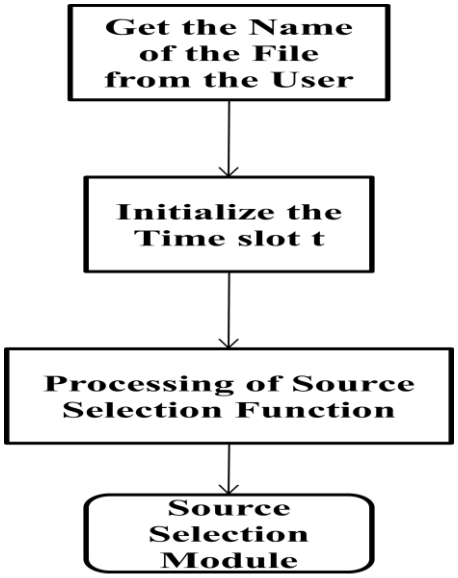

Fig 3: Downloading Peer Initiation Module

5.2 Source Selection Function Module

In Source Selection Function Module, the file name is received as input. The fixed time slot t is initialized. The downloading peer identifies all the source peers and randomly selects the k source peers over N possible source peers and divides the file into chunks. Fig. 4 illustrates the functionality of the Source Selection Function Module.

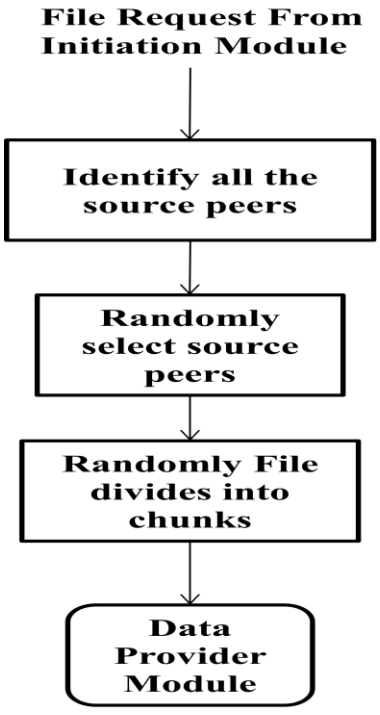

Fig 4: Source Selection Function Module

5.3 Data Provider Module

In this Data Provider module, the chunk size of the file is identified and divides into k chunks of equal size. The chunk ratio is calculated and it is given to each Source Peer Module. Fig. 5 illustrates the functionality of the Data Provider Module.

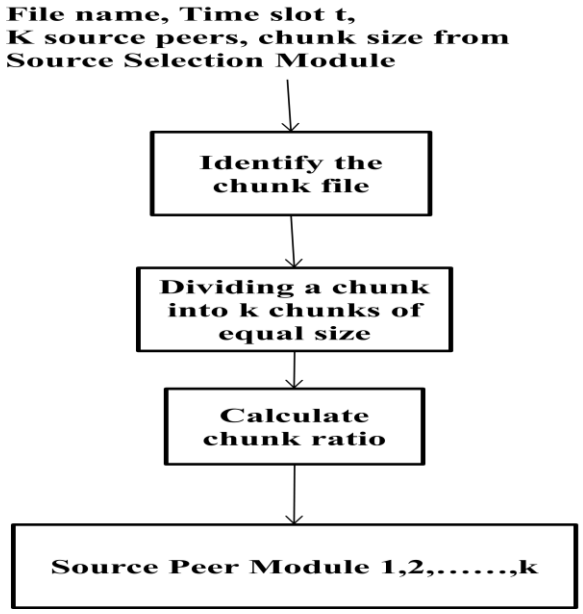

Fig 5: Data Provider Module

5.4 Source Peer Module

In this Source Peer Module, the File name, Time slot t, and the chunk ratio is received as input from the Data Provider Module. Then it finds the total chunk size and calculates the number of bytes from the chunk ratio. Then the file content is given to the Data Receiver Module. Fig.6 illustrates the functionality of the Source Peer Module.

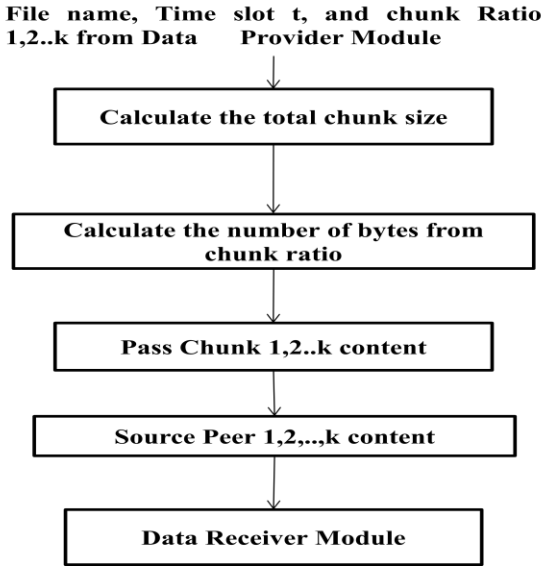

Fig 6: Source Peer Module

5.5 Data Receiver Module

In this Data Receiver Module, the file content from the randomly selected k source peers is received as input. This module verifies whether each chunk over k chunks is received fully. If the k chunks are received fully, then the file content is received. Fig.7 illustrates the functionality of the Data Receiver Module.

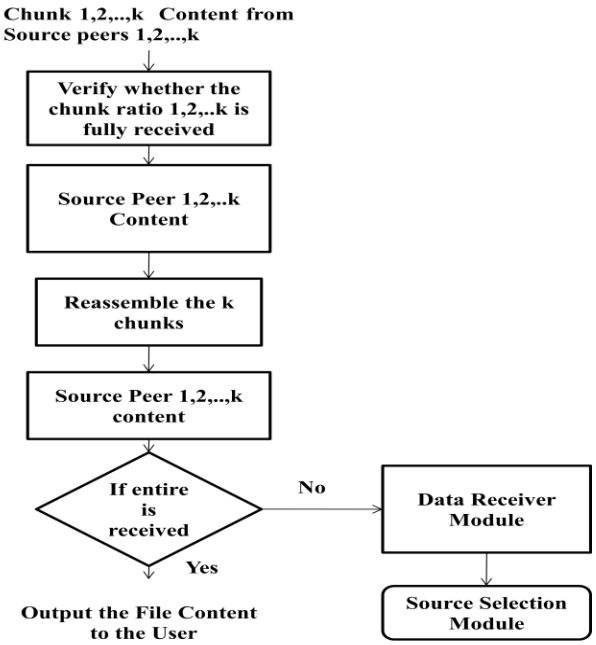

Fig 7: Data Receiver Module

6. DYNAMICALLY DISTRIBUTED PARALLEL RANDOM CHUNK BASED SWITCHING

6.1 Downloading Peer Initiation Module

In this Downloading Peer Initiation Module, the user requests the file from the downloading peer and the file is given to the Source Selection Function Module. The Fig.8 illustrates the initial module

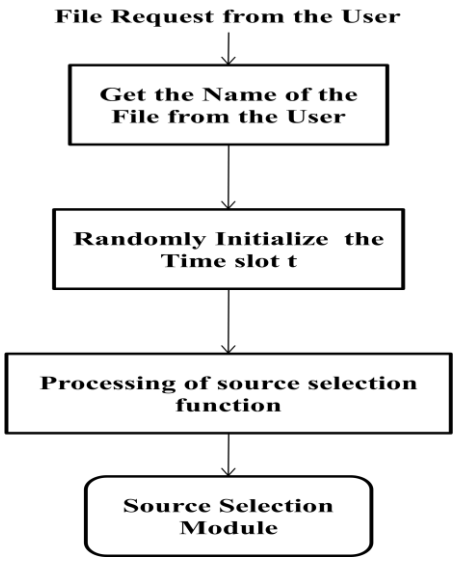

Fig 8: Downloading Peer Initiation Module

6.2 Source Selection Function Module

The downloading peer identifies all the source peers and randomly selects the k source peers over N possible source peers and divides the file into chunks. Fig. 9 illustrates the functionality of the Source Selection Function Module.

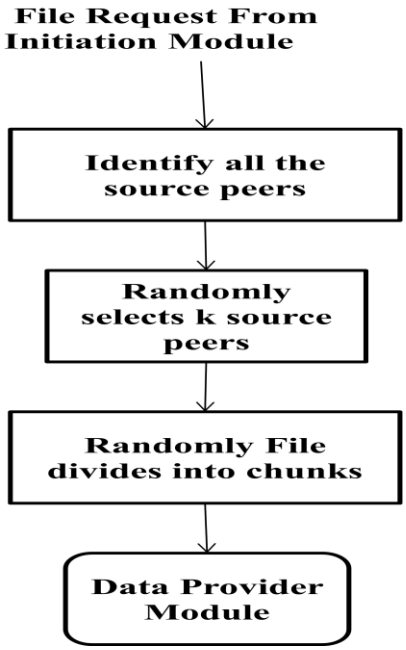

Fig 9: Source Selection Function Module

6.3 Data Provider Module

In this Data Provider module, the chunk size of the file is identified and divides into k chunks of equal size. The chunk ratio is calculated and it is given to each Source Peer Module. Fig. 10 illustrates the functionality of the Data Provider Module.

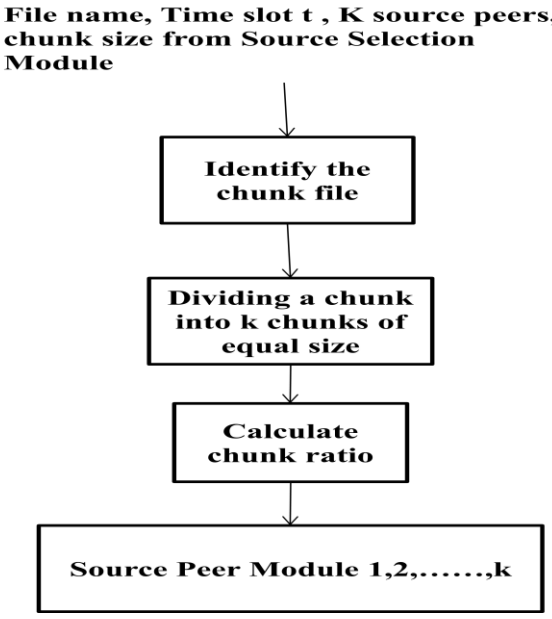

Fig 10: Data Provider Module

6.4 Source Peer Module

In this Source Peer Module, the File name, Time slot t, and the chunk ratio is received a input from the Data Provider Module. Then it finds the total chunk size and calculates the number of bytes from the chunk ratio. Then the file content is given to the Data Receiver Module. Fig. 11 illustrates the functionality of the Source Peer Module.

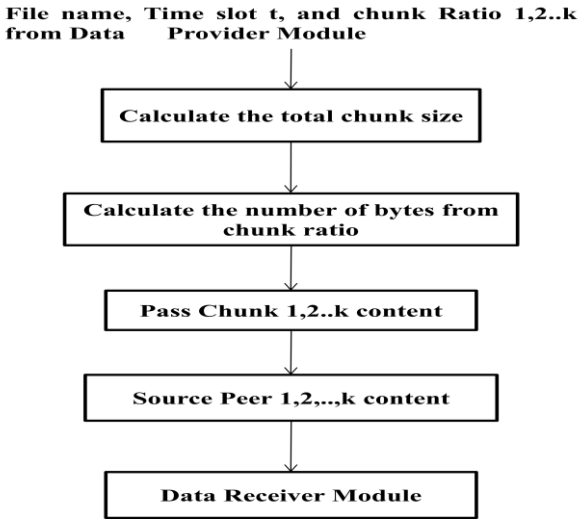

Fig 11: Source Peer Module

6.5 Data Receiver Module

In this Data Receiver Module, the file content from the randomly selected k source peers is received as input. This module verifies whether each chunk over k chunks is received fully. If the entire file is not received, then it starts downloading the next chunk from the file by continuing the process from the Source Selection Module. Fig. 12 illustrates the _functionality of the Data Receiver Module.

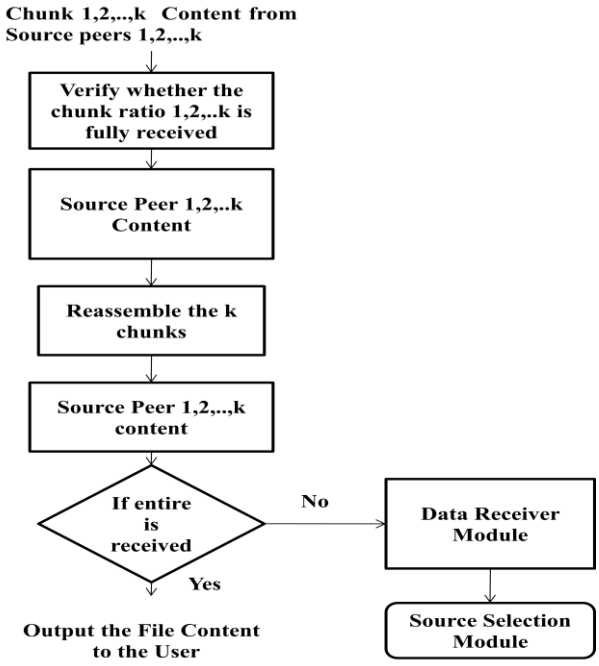

Fig 12: Data Receiver Module

7. CONCLUSION AND FUTURE WORK

In this paper, we have designed a new scheme that effectively removes correlations in the capacity fluctuation and the heterogeneity in space, thus greatly reducing the average download time. It is highly desirable to improve the network efficiency by reducing each user's download time. Our future enhancement is to overcome the negative factor in the existing Random Chunk Based Switching by providing the Dynamic Distributed Algorithm.

8. REFERENCES

[1] Y. M. Chiu and D. Y. Eun, "Minimizing file download time over stochasticchannels in peer-to-peer networks," in *Proc. 40th Annu. Conf.Information Sciences and Systems (CISS)*, Princeton, NJ, Mar. 2006.

[2] D. Qiu and R. Srikant, "Modelling and performance analysis of Bit-Torrent-like peer-to-peer networks," in *Proc. ACM SIGCOMM*, Aug.2004.

[3] X.Yang and G. deVeciana, "Service capacity of peer to peer networks,"in *Proc. IEEE INFOCOM*, Mar. 2004, pp. 2242–2252.

[4] K. P. Gummadi, R. J. Dunn, and S. Saroiu, "Measurement, modeling,and analysis of a peer-to-peer file sharing workload," in *Proc. ACMSymp. 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 *Proc. Workshop on Economics of Peer-to-Peer Systems (P2PEcon)*Cambridge, MA, Jun. 2004.Brown, L. D., Hua, H., and Gao, C. 2003. A widget framework for augmented interaction in SCAPE.

[6] M. Adler, R. Kumar, K. Ross, D. Rubenstein, T. Suel, and D. D. Yao,"Optimal peer selection for P2P downloading and streaming," in *Proc.IEEE INFOCOM*, Miami, FL, Mar. 2005, pp. 1538–1549.

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

[8] S. G. M.Koo, K. Kannan, and C. S. G. Lee, "A genetic-algorithm-basedneighbor-selection strategy for hybrid peer-to-peer networks," in *Proc.IEEE Int. Conf. Computer Communications and Networks (ICCCN2004)*, Rosemont, IL, Oct. 2004, pp. 469–474.

[9] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker, "AScalable Content Addressable Network," Univ. California, Berkeley,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 applications,"in *Proc. ACM SIGCOMM*, 2001.

[11] B. Y. Zhao, L. Huang, J. Stribling, S. C. Rhea, A. D. Joseph, and J.D. Kubiatowicz, "Tapestry: A resilient global-scale overlay for servicedeployment," *IEEE J. Sel. Areas Commun.*, vol. 22, no. 1, pp. 41–53,Jan. 2004.

[12] J. Byers, J. Considine, M. Mitzenmacher, and S. Rost, "Informed contentdelivery across adaptive overlay networks," in *Proc. ACM SIGCOMM*,2002.

[13] C. Gkantsidis and P. R. Rodriguez, "Network coding for large scalecontent distribution," in *Proc. IEEE INFOCOM*, Miami, FL, Mar.2005, pp. 2235–2245.

[14] "Peer-to-peer content distribution: Using client PC resources to storeand distribute content in the enterprise" Intel Corp., Tech. Rep.,Sep. 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 modelingpeer to peer networks," in *Proc. IEEE INFOCOM*,Mar. 2005, pp.215–269.

[16] S. G. M. Koo, C. Rosenberg, and D. Xu, "Analysis of parallel downloadingfor large file distribution," in *Proc. IEEE Int. Workshop on FutureTrends in Distributed Computing Systems (FTDCS)*, May 2003,pp. 128–135.

[17] F. Lo Piccolo and G. Neglia, "The effect of heterogeneous link capacitiesin BitTorrent-like file sharing system," in *IEEE Int. Workshop onHot Topics in Peer-to-Peer Systems (HOT-P2P)*, Oct. 2004, pp. 40–47.

[18] T. Ng, Y. Chu, S. Rao, K. Sripanidkulchai, and H. Zhang, "Measurement-based optimization techniques for bandwidth-demandingpeer-to-peer systems," in *Proc. IEEE INFOCOM*, Apr. 2003, pp.2199–2209.

[19] Y. Kulbak and D. Bickson, "The eMule Protocol Specification," Jan.2005 [Online]. Available: http://leibniz.cs.huji.ac.il/tr/acc/2005/HUJICSE-LTR-2005-3_emule.pdf

[20] R. Sherwood, R. Braud, and B. Bhattacharjee, "Slurpie: A cooperativebulk data transfer protocol," in *Proc. IEEE INFOCOM*, Hong Kong,Mar. 2004, pp. 941–951.

[21] B. Cohen, "BitTorrent Protol Specification," [Online]. Available:http://ww.bittorrent.com/protocol.html.

[22] S. Saroiu, K. P. Gummadi, and S. D. Gribble, "A measurement studyf peer-to-peer file sharing systems," in *Proc. ACM Multimedia Computingand Networking (MMCN)*, 2002.

[23] M. Jain and C. Dovrolis, "End-to-end estimation of the available bandwidthvariation range," in *Proc. ACM Sigmetrics*, Jun. 2005.

[24] N. Hu and P. Steenkiste, "Evaluation and characterization of availablebandwidth probing techniques," *IEEE J. Sel. Areas Commun.*, vol. 21,no. 6, pp. 879–894, Aug. 2003.

[25] S. M. Ross, *Stochastic Processes*, 2nd ed. New York: Wiley, 1996.

[26] I. Rhee and L. Xu, "Limitations of equation-based congestion control,"in *Proc. ACM SIGCOMM*, Aug. 2005.

[27] A. Müller and D. Stoyan, *Comparison Methods for Stochastic Modelsand Risks*. New York: Wiley, 2002.

[28] C. Gkantsidis, M. Ammar, and E. Zegura, "On the effect of large-scaledeployment of parallel downloading," in *Proc. IEEE Workshop on InternetApplications (WIAPP)*, Jun. 2003, pp. 79–89.