Reducing File Download Delay by

QoS Driven Parallelization of Resources

Shaleeza Sohail, Sanjay Jha, Salil S. Kanhere and Chun Tung Chou

CSE, UNSW, Sydney, Australia

Email: (sohails,sjha,salilk,ctchou)@cse.unsw.edu.au

UNSW-CSE-TR-0518

September 2005

THE UNIVERSITY OF
NEW SOUTH WALES

SYDNEY • AUSTRALIA

Abstract

In this paper, we propose a novel approach, for reducing the download time of large files over the Internet. Our approach, known as Parallelized File Transport Protocol (P-FTP), proposes simultaneous downloads of disjoint file portions from multiple file servers. P-FTP server selects file servers for requesting client, on the basis of a variety of QoS parameters, such as, available bandwidth and server utilisation. The sensitivity analysis of our file server selection technique shows that it performs significantly better than random selection. The scalability study of P-FTP server shows that it can handle queries of large number of P-FTP clients, without becoming bottleneck. During the file transfer, P-FTP client monitors the file transfer flows to detect slow servers and congested links and adjusts the file distributions accordingly. P-FTP is evaluated with simulations and real-world implementation. The results show at least 50% reduction in download time, when compared to the traditional file-transfer approach. Moreover, we have also carried out a simulation-based study to investigate the issues related to large scale deployment of our approach on the Internet. Our results demonstrate that large number of P-FTP users has no adverse effect on the performance perceived by non P-FTP users. In addition, the file servers and network are not significantly affected by large scale deployment of P-FTP.

1 Introduction

Downloading large files, can be a very time-consuming and slow process. As a result, they are highly susceptible to congestion, that results from the high network dynamism that is prevalent in the Internet. Numerous studies of Internet traffic characteristics, have shown that file size distributions are heavy-tailed [7]. This implies that a significant number of large files are downloaded over the Internet. Hence, reducing the download time for large files is an important and relevant problem.

A common approach, to address this problem involves provisioning multiple mirror servers with replicated content. With multiple copies existing at these mirror servers, a popular server selection strategy involves choosing the geographically closest server, assuming that this will result in the least delay [2]. However, this can result in poor performance if other mirror servers, not necessarily geographically closest, have a higher resource availability at that time and hence, can be accessed for less download delay. Further, there is no guarantee that a server which provides optimal performance at the start of the file download will necessarily be able to provide the same level of performance at the end of the download. Rather than downloading the entire file from a single server, we can take advantage of the fact that the files are replicated across multiple servers and hence disjoint portions of a single file can be simultaneously downloaded in parallel.

In this paper, we propose a novel parallel download scheme called Parallelized File Transfer Protocol (P-FTP), with these capabilities. Our approach involves the use of a special P-FTP server running within the client's Autonomous System (AS) [12]. This server is capable of communicating with the file servers, located anywhere in the Internet and measuring the Quality of Service (QoS) parameters, along the paths to these servers. As opposed to traditional FTP, a client in our case first contacts the P-FTP server when a particular file is to be downloaded. The P-FTP server then selects a set of suitable file servers from which

the file can be downloaded in parallel. This information is sent back to the client, who then initiates parallel file transfer sessions. In addition to this, the P-FTP server also sends a list of backup file servers, that will be contacted by the client if there is congestion in the network or some particular file server fails or slows down. This makes our technique dynamically adaptable to the changing network conditions.

The file server selection, is based on the server utilisation and the QoS characteristics along the network path to the servers. The server utilisation is measured based on its CPU and memory utilisation. The path quality is determined by QoS parameters such as available bandwidth, round trip time and packet dropping probability. However, additional parameters can be easily introduced to account for any other factors that may affect the performance. A database of these parameters is maintained at the P-FTP server and is regularly updated to ensure the freshness of the information. The P-FTP server ranks all the available servers based on these parameters and selects a subset of these servers based on the available bandwidth between the client and its Internet Gateway. Given this information, the P-FTP server, determines the respective file portions that can be downloaded from the selected file servers. This information is then sent to the client.

The P-FTP client is able to dynamically adapt to the changing network conditions. In particular, we focus on two such conditions: congested common links and the file server's throughput. Our client can detect servers that are sharing a common congested link on their path to the client. The connections to such servers are terminated upon successful detection and new connections with the backup servers nominated by the P-FTP server are initialised. Moreover, the throughput of the file servers, relative to themselves, is regularly monitored to detect low-performing or failed file servers. In such an event, the file portions to be downloaded from slow / failed servers are reduced and new sessions are initiated with the backup servers to download the remaining file portions. This approach has two additional advantages: 1) It distributes the load and reduces the congestion in the Internet, and 2) The download time does not vary significantly when the network conditions change. Following are the primary contributions of this paper:

- The paper discusses the distinguishing features of P-FTP. In particular, network parameter based selection of file servers and dynamic adaptation ability, are the two main features that distinguish P-FTP from other parallel downloading approaches.

- A thorough evaluation of P-FTP is presented using a mix of Internet experiments and simulations. Comparisons with traditional FTP and other parallel downloading techniques are also conducted. The results show that P-FTP significantly reduces the download time of large files in all testing scenarios.

- Large scale Internet deployment issues of P-FTP are investigated with the help of simulations. The paper presents convincing evidence that if a large number of users are downloading files using P-FTP, there is no detrimental impact on the performance of other non-P-FTP clients. In addition, our experiments indicate that P-FTP does not significantly impact the servers and the network due to its self-tuning ability.

The rest of the paper is organised as follows: Section 2 provides a brief description of the work related to our approach. Section 3 provides details on the P-FTP approach. In particular, the working of the P-FTP server (Section 3.2) and the P-FTP client (Section 3.3). Section 4 discusses the simulation carried out to measure the P-FTP performance. Section 5 discusses the Internet implementation of P-FTP and provides the results. Section 6 briefly discusses the issues of P-FTP's large scale deployment and provides the details of the simulations carried out to study the effect of such deployment. Section 7 discusses several issues related to P-FTP deployment. Finally, section 8 concludes the paper and provides some future work options.

2 Related Work

Numerous research groups have proposed different criteria for the selection of a single mirror server for any client ([3], [4] and [5]). We propose the use of multiple mirror servers when retrieving a single file to reduce the download time. Our approach uses QoS parameters of the network and the file-servers to select

the suitable set of mirror servers for file download.

In the client-server scenario, a few researchers have used the approach of transferring a single file from multiple mirror servers simultaneously ([37], [8] and [6]). The work that closely relates to our proposal is of Dynamic Parallel Access Technique (DPAT) [8]. It proposes downloading large files by connecting to multiple HTTP servers simultaneously. The file is partitioned into small blocks which are downloaded from all the selected servers. The server with highest throughput sends the largest number of blocks. The mechanism for calculating block size, storing file server information and failure recovery are not clearly discussed by the authors. Plank et al. [26] evaluated the performance of DPAT like approach with progress driven redundancy of the different block sizes. Collins and Plank [27], evaluated different server selection algorithms for DPAT. The authors, concluded, that the file server algorithms based on bandwidth prediction performed best. In addition, the authors anticipated that incorporating server load in server selection may further improve the performance. We also propose using multiple QoS parameters related to network and mirror servers for selection of servers. However, our approach is based on continuous connections with the file servers to download file portions without requiring any redundant connections. Most approaches, that propose parallel downloading ([19] and [34]), require either the change in the operation of the servers, or change in the content encoding method. P-FTP can work even if no support from the file servers is available. Internet experiments without any support from traditional FTP servers are discussed in section 5.

Grid-FTP [47] is designed for grid environment, by extending the basic FTP to enable terabyte transfers in grid systems. It adds new modes and commands to the basic FTP standards. Grid-FTP is specifically designed for the Grid environment and it requires modifications to the FTP servers to support the Grid-FTP protocol. P-FTP uses standard FTP and is designed to work on the existing client-server model of the Internet. Allen and Wolski [28], compared performance of four server selection techniques for file transfer in Grid environment. The authors, demonstrated that for fastest downloads network resource prediction

techniques, based on network probing, performed best. P-FTP system has also based its server selection technique on prediction of resource availability in network and at file servers. The resource prediction requires network and server probing for resource usage.

Many peer-to-peer (P2P) applications ([35], [36], [43] and [44]), provide the advantage of simultaneous partial file download. Some P2P applications require change in file format or special files on the servers, in order to operate successfully. The P2P approach is limited in the sense that P2P application-program must be running on all machines sharing the files. Unknown machines must be trusted for some P2P applications, which require the users to allow uploading from their machine, in order to get good download speed. P-FTP is not on competing grounds with P2P application as P-FTP addresses the file download in client-server paradigm of the Internet. Secondly, our approach can work even if there is no support available from the file servers.

Research groups have also addressed the issue of large scale deployment of parallel downloading approaches; two such efforts due to their similarity to our work are discussed here. Koo et al. [20], present analytical and simulation results to show that parallel downloading approaches degrades the performance of servers and the network significantly. Gkantsidis et al. [21] compare the performance of three parallel downloading techniques in large scale deployment scenario using simulation. They have shown that by introducing many simultaneous parallel downloading clients in the network, other clients experience degradation of performance. In contrast to the research work discussed in [20] and [21], we have carried out rigorous simulation based testing, to show that introducing a large number of P-FTP users has no adverse effect on the performance perceived by users of other applications.

3 Parallelized File Transfer Protocol (P-FTP)

We describe P-FTP approach in detail in this section. The P-FTP design has been modified, from our initial design in [9], for scalability and ease in real-world implementation.

3.1 P-FTP Features

P-FTP is a protocol to download files on the Internet by efficiently using the available network and mirror server resources, in order to reduce the download time. The salient features of P-FTP are:

- The dynamic use of distributed resources in the network limits, the file transfer time in a predictable fashion.

- The placement of a central P-FTP server in the user's AS allows us to serve many clients by maintaining a single database of network and file servers characteristics. We analyse the feasibility of a single P-FTP server in large ASs in section 3.4.

- P-FTP checks the available bandwidth at the last hop i.e., between the client and its Internet Gateway, before starting the file transfer. This consideration is introduced to avoid unnecessary parallel download initiations, in the situation when the client is attached to slow access link, such as a dial-up connection to the Internet.

- Simultaneously, downloading parts, of a large file from different FTP servers distributes the load on the mirror servers. This distribution of the load, among the servers improves performance for all users accessing those servers.

- The P-FTP clients monitor the TCP flows, after starting the transfer of file portions, from the mirror servers. If multiple flows are sharing a common congested link on their path to the client, the client terminates selected flows to reduce congestion.

- P-FTP system can work even without any support from the file servers. The tests conducted on the Internet without file server's support are discussed in section 5. As previously discussed, P-FTP system requires a special client at the file servers to collect file server utilisation information.

However, Naf [29] has demonstrated with Internet experiments that the load on the file server does not significantly affect the file download time for the clients. Hence, the file server monitoring module is not mandatory and we expect it to be deployed incrementally.

3.2 P-FTP Server

The P-FTP server is the central entity in the system. It is placed in the client's autonomous system (AS) and is accessible by all users of that AS. Balakrishnan et al. [23] demonstrated that Internet hosts close to each other (within 2-4 hops) often experience almost the same throughput. The P-FTP server is placed in the same AS as we assume that for small to medium ASs, where hosts are at 2-4 hops away from each other, the measured network characteristics are valid for all hosts of that AS (section 7 discusses issues related to large ASs).

When the P-FTP server receives a request from a P-FTP client, it gathers relevant information regarding the mirror servers containing a copy of the requested file. This information is maintained by the P-FTP server in a database and is used to rank the mirror servers in order of resource availability. Depending upon the available bandwidth between client and its Internet Gateway, P-FTP server selects a number of high ranked mirror servers ensuring that the aggregate P-FTP traffic does not produce congestion in the client's AS. The P-FTP server also calculates the file portions to be downloaded from each selected mirror server. The highest ranked mirror server is allocated the largest file portion and vice versa, the ranking and selection processes are discussed in section 3.2.2. The information about mirror servers and the file portions is sent to the requesting client. This also includes a set of backup or additional mirror servers, in case the client detects congestion and wants to dynamically adjust the load. We now describe each step in detail in the following subsections.

3.2.1 P-FTP Database

The P-FTP database plays a vital role in the ranking and selection process of the P-FTP server. The database contains data on the network, file server utilisations and the list of files on these file servers. The P-FTP server can use any publicly available network measurement tool to collect network information (e.g., [14], [15], [16], [17]), *Ping* [49] and *Pathchar* [38] are two examples of commonly used tools. The network information consists of network characteristics along the paths between the client's AS and different mirror servers. The utilisation information consists of the mirror server's memory and CPU utilisation. This information can be controlled by the mirror server's administrator. A special client is placed at mirror servers that send information about mirror server utilisation to the P-FTP server at a configured rate. The file information consists of a file replica map of mirror servers that indicates which file is replicated at which server. Some issues regarding the P-FTP database design, are discussed in sections 3.4 and 8.

3.2.2 Ranking and Selection Process

The ranking process makes use of the following information:

- Network Characteristics: The QoS parameters, i.e., the available bandwidth, the round trip time and the packet dropping probability between user's AS and different mirror servers, provide the basis for the ranking process. The mirror servers that have more network resources along their path to the client's AS are ranked higher.

- Utilisation: The P-FTP server considers the CPU and memory utilisation level of the mirror servers. Highly utilised mirror servers have lower rank.

- However, other user defined parameters (such as the use of selected servers) can be easily incorporated as well.

We call the parameters used in ranking process, optimisation variables. For every optimisation variable, the suitability of each mirror server is calculated and then a combined rank is allocated. Suppose there are m mirror servers and their suitability on the basis of an optimisation variable, OV is to be calculated. The method to calculate the suitability for mirror server k, S_k, depends upon the type of OV: direct (DOV) or inverse (IOV). DOV directly influences the suitability of mirror servers. Examples of DOV are bandwidth along the path, and the mirror server's available resources. The suitability of mirror server S_k using DOV is calculated by the following equation:

$$S_k = \frac{DOV_k}{\sum_{i=1}^{m} DOV_i} \quad (1)$$

IOVs affect the suitability of the mirror server inversely. End-to-end delay along a path and mirror server utilisation are examples of IOV. The suitability of mirror server S_k on the basis of IOV is calculated by the following equation:

$$S_k = \frac{\frac{1}{IOV_k}}{\sum_{i=1}^{m} \frac{1}{IOV_i}} \quad (2)$$

The P-FTP server calculates the suitability for all mirror servers on the basis of each optimisation variable. The simple average or weighted average of all suitability values is calculated depending upon the relative importance of the optimisation variables. The mirror server with highest average suitability is ranked first and so on.

The P-FTP server selects the mirror servers according to their ranks, high-ranked mirror servers are selected first. The selection is limited by the available bandwidth between the client and its Internet gateway. The mirror servers are selected such that the aggregate P-FTP flow does not produce congestion at the link between the client and its gateway. The file portions to be downloaded from each server are calculated, based on the suitability values, of the selected servers. The server with highest suitability transfers the largest file portion and vice versa.

When the request arrival rate at the P-FTP server exceeds a predefined threshold, the P-FTP server changes the selection criteria. The reason behind this is that in case of large number of requests if only the best servers are selected then these servers may get congested and all clients will experience poor performance. To avoid this condition, the P-FTP server tries to load balance the requests among all available FTP servers when request rate is high. The load balancing is a well tested concept, hence, we do not discuss it any further in this paper.

3.2.3 P-FTP Server Failure Recovery

Single P-FTP server for a complete AS is susceptible to failure. In which case all the clients of that AS would be unable to use P-FTP. The simplest solution for efficient recovery in case of failure would be to have a backup server that takes over in case of failure of primary P-FTP server. Since these servers can be built using low cost personal computers, it will not add a lot to the infrastructure cost.

3.2.4 P-FTP Server Discovery

P-FTP server is a central entity in an AS and all P-FTP clients of that AS, contact it, when they require to download a file. This raises the issue of discovering the P-FTP server by P-FTP clients in order to contact it. The network administrator can use any resource discovery technique [24] [25] to facilitate P-FTP clients in discovering P-FTP server. DNS Resource Record (RR) [30] can also be used to specify location of servers for specific protocols and domains in the DNS response to a requesting client.

A scalable and easily adaptable solution to this problem is to fix a port for communication among P-FTP server and clients. P-FTP server can inform about its presence to all the P-FTP clients of that AS via a periodic broadcast message on the fixed port. The network administrator, can adjust the frequency and size of the broadcast message to minimise the overhead of P-FTP. In case of failure of primary P-FTP server (lack of broadcast message for a while), the backup P-FTP server takes over, by immediately broadcasting

its address to all P-FTP clients of that AS.

3.3 P-FTP Client

The P-FTP client is a FTP client with additional capabilities of partial file transfer and flow monitoring. To download arbitrary portion of file, P-FTP client uses standard FTP command *REST* to initiate file transfer from a required point in the file and terminates that flow after collecting the required file data.

P-FTP client measures the available bandwidth to its Internet Gateway, before requesting file server information from the P-FTP server. We used *Ping* messages of different sizes to calculate the bandwidth of the link [33], however any other tool can be used. P-FTP client sends a request containing the file name and the available bandwidth to the P-FTP server. After receiving a reply from the P-FTP server, the client starts simultaneous FTP connections with the nominated mirror servers and starts downloading disjoint file portions. The P-FTP client is able to dynamically adjust to the changing network and file server conditions. The client monitors the inter-packet arrival rate to detect the servers sharing a congested link on their path to the client. The client terminates connections to all detected mirror servers except one (as explained below). The client starts the same number of new connections, to the additional mirror servers nominated as back up servers by the P-FTP server. The client, continuously monitors the received data rate from all file servers. If one or more of these servers falls below their expected rate then it reduces the size of the respective allocated file portions and downloads the remaining file portions from the backup servers. The effect of dynamic adaptation ability of P-FTP clients on file transfer time in wireless networks has been evaluated with simulations [1]. We now explain these steps in detail.

3.3.1 Shared Congestion Detection and Dynamic Adaptation

The P-FTP server selects the file servers on the assumption that no links from the file servers to the clients are shared, except the last link. In reality this may not be the case resulting in congestion on shared links.

Our client is capable of detecting and dynamically adjusting to this situation.

Katabi and Blake [13] proposed the use of packet inter-arrival rate, to detect a common congested link and evaluated their approach with simulation study. We implemented an algorithm at the application-layer level, based on the same approach. The client monitors the data packet arrival time of all the flows. The data packets of the flows reach the client with random inter-packet intervals. However, when packets traverse a congested link on their path, the inter-packet arrival times of such packets follow a pattern and do not show randomness. The client detects the servers, which share a common congested link, on their path to the client by measuring the entropy of inter packet intervals.

After detecting the servers sharing the congested links, the client terminates the connections with all such file servers except the server ranked highest in that set of servers. It should be noted that only those servers are selected by the P-FTP server that have enough network resources to support at least a single FTP session, hence the justification for leaving one connection intact.

3.3.2 Slow Server Detection and Dynamic Adaptation

The P-FTP server calculates the file portions, such that all file servers finish sending the file-data at approximately the same time. If any of the file server slows down or fails during a P-FTP session, then that will increase the total file download time. Our P-FTP client can detect this situation and act accordingly for better P-FTP performance. The client continuously monitors the throughput of each server connection. The throughput should be proportional to the allocated file size. If this is not the case for a particular server, the client proportionally reduces the size of the remaining file portion to be downloaded. The backup servers are invoked to download rest of the file portion. In case any of the file server stops the file transfer, the P-FTP client terminates that connection and downloads remaining file portion from a backup server.

3.3.3 Complexity of P-FTP Client Algorithm

P-FTP client starts simultaneous file transfer connections to download required file quickly. The dynamic adaptation ability of the P-FTP client, is based on shared congestion detection algorithm and slow server detection algorithm. These algorithms at the P-FTP client add to the complexity of its implementation. We have calculated the complexity of each algorithm and determined the overall complexity of P-FTP client.

Shared congestion detection algorithm determines a congested link shared by two or more flows by aggregating different flows and determining the entropy of inter packet arrival time of aggregated flows. Calculation and aggregation of entropy values require a constant number of steps. Let the number of file transfer flows be n. Since shared congestion detection requires comparing each pair of flows so the overall complexity of the algorithm is n^2, in terms of number of flows.

The slow server detection algorithm detects the slow file server, by comparing the throughput of all the flows. The steps required in determining the throughput of n flows takes a constant time. The number of comparisons of the throughput values have the same complexity of n^2. Combining this with shared congestion detection algorithm, the total complexity will be $O(n^2)$. As the number of flows is small (typically 5-10), the combined complexity of both the algorithms can be ignored for practical purposes.

3.4 Modeling P-FTP Server Workload using MVA

The P-FTP server is a centralised entity in a user's AS. As all users of that AS query the P-FTP server when they need to download a large file, the P-FTP server may become a bottleneck in a large AS. To study the scalability issues of the P-FTP server, we have designed and implemented a P-FTP server for the UNSW.edu.au domain and have analysed its performance from the system-level point of view with the help of Mean Value Analysis (MVA) technique [22]. MVA technique was chosen to compute various performance metrics as we are mainly concerned with the mean values of the performance metric. We

consider the system as a “Black Box” [22]. We also assume that the system being analysed is in operational equilibrium. The P-FTP server processes two distinct types of workload, the user’s queries and the database update. Thus, from the system level performance analysis, the P-FTP server is considered as a multiple class infinite queue system [22].

For the estimate of the query rate at the P-FTP server, we studied the logs of all FTP transfers in the School of Computer Science and Engineering (CSE) of University of New South Wales (UNSW). It was not possible for us to get network traffic logs from any ISP as the data of such commercial organisations is not available publicly to our knowledge. Secondly, CSE can be considered as one of the worst case example, as the network users pay little / no download charges and hence, large files are downloaded frequently. We studied the traffic logs for a week from 1st June 2004 to 7th June 2004, in which total 49 files of size 1 MByte or more were downloaded. We also analysed the log files for the maximum number of large files (larger than 1 MByte) downloaded at any time. We observed that at most 4 file-transfers occurred in a minute. The school of CSE of UNSW has 3,500 users and there are approximately 35,000 total users in the UNSW domain. By assuming the same download pattern for UNSW AS, we estimated that the maximum number of requests, λ_{Query}, at the P-FTP server of UNSW AS will be 40 requests per minute or approximately 1 request per second.

The processing of a request at the P-FTP server consists of the following three phases. Phase one is the connection establishment and initial processing of the request. Phase two consists of the P-FTP database access. Phase three involves processing of the file-server data (ranking and selection, query reply formation and reply delivery) and connection termination. The UNSW P-FTP server implementation was tested on a Pentium 3 machine with 256 MBytes RAM. The individual processing times of 100 different queries were measured at the P-FTP server. During the testing, the maximum times required for the processing of phases one and three of any request were 0.2 milliseconds and 20 milliseconds respectively. We conducted a simple

study to measure the database access time of P-FTP server. For 200,000 files and 200,000 file-servers, the average time to retrieve the file-server parameters, given a filename as an input, was measured to be 0.1 millisecond assuming that on average each file was replicated on 20 servers. The processing time for a request at the P-FTP server is the sum of processing time of all three phases. On the test machine, the UNSW P-FTP server's request processing time of user's query, D_Q, was 20.3 msec (phase 1 = 0.2 msec, phase 2 = 0.1 msec, phase 3 = 20 msec).

We did not perform a study to analyse the time taken to populate the database as it is a one time process. However, a number of experiments were conducted to study the update process of the database. For the database update, we only considered updating the network information for a large number of servers (10,000). Various publicly available tools, were tested to calculate the average time required to measure network characteristics. We present results of two representative tools here: *Ping* and *Pchar*. The service rate D of *Ping* and *Pchar* to update one server record was 800 msec and 7 sec respectively. Balakrishnan et al. [23] demonstrated that the throughputs to the Internet hosts do not change much within the time frame of tens of minutes. During the tests, the P-FTP database was updated once after few hours, as the tests were conducted on a simple desktop machine. If a high power dedicated server is used as P-FTP server, then the P-FTP database can easily be updated in the matter of tens of minutes. Moreover, the tools that provide more information are used less frequently to minimise the load in user's AS. Hence, we considered the database update frequency of *Ping* to be high as every four hours ($\lambda_{ping} = 1/(4*3600)*10,000(fileservers) = 0.69/sec$) and that of *Pchar* to be low as every thirty hours ($\lambda_{pchar} = 1/(30*3600)*10,000(fileservers) = 0.09/sec$). The utilisation of the P-FTP server U_{PFTP} is the sum of the utilisation due to query U_{Query} and that due to database update U_{Update}. The utilisation of the system for the database update depends upon the tool used to gather network information.

$$U_{PFTP} = U_{Query} + U_{Update} \quad (3)$$

$$U_{Query} = \lambda_{Query} * D_{Query} = 1 * 0.0203 \quad (4)$$

No of Users	Request Arrival Rate req/sec	Ping		Pchar	
		P-FTP server Utilisation	Query Response Time (sec)	P-FTP Server Utilisation	Query Response Time (sec)
35,000	1	57%	0.0474	65%	0.058
50,000	1.5	58%	0.0487	66%	0.059
100,000	3	61%	0.0524	69%	0.065
200,000	6	67%	0.0622	75%	0.817

Table 1: The P-FTP Server Response Time for Large ASs.

$$Ping : -U_{Update} = \lambda_{ping} * D_{ping} = 0.69 * 0.8 = 0.552 \quad (5)$$
$$Hence : -U_{PFTP} = 0.572 \quad (6)$$
$$Pchar : -U_{Update} = \lambda_{pchar} * D_{pchar} = 0.09 * 7 = 0.63 \quad (7)$$
$$Hence : -U_{PFTP} = 0.6503 \quad (8)$$

The above calculations show that most of the processing of the P-FTP server is dedicated to the update of the database. The time taken by a server to reply to the query of the requesting client is called response time. The P-FTP server's response time for query, R_{Query} is:

$$Ping : -R_{Query} = \frac{D_{Query}}{1 - U_{PFTP}} = \frac{0.0203}{1 - 0.57} = 0.0474sec \quad (9)$$
$$Pchar : -R_{Query} = \frac{D_{Query}}{1 - U_{PFTP}} = \frac{0.0203}{1 - 0.65} = 0.058sec \quad (10)$$

We have also analysed the performance of the P-FTP server for ASs larger than UNSW, where the population is much bigger and hence the request arrival rate is much higher. The calculated response times of the P-FTP server in such large ASs, using same database specifi cations are shown in table 1.

The response time of the P-FTP server, serving a very large population of say 200,000 users using a resource intensive tool like *Pchar* is very small (817 msec). The P-FTP server response time (in milliseconds) is almost negligible compared to the download times of large fi les, which are usually in the order of seconds or minutes. Hence, we can safely state that the P-FTP server does not create any bottleneck for process of the P-FTP downloads even when it is implemented on a simple desktop machine and serves a large population of users.

P-FTP server utilisation and query response time shown in table 1 are calculated with update frequency

Tool Used	Update Frequency (hours)	λ for 10,100 servers	U_{update}	P-FTP Server Utilisation $U_{update} + U_{Query}$	Query Response Time (sec)
Ping	2.5	1.11	0.88	95%	.406
Ping	3	0.925	0.74	81%	.106
Ping	5	0.55	0.44	51.5%	.042
Ping	6	0.46	0.37	44%	.036
Pchar	21	0.13	0.925	99.5%	4.060
Pchar	26	0.106	0.75	82%	0.112
Pchar	35	0.08	0.56	63%	0.055
Pchar	40	0.07	0.49	56%	0.046

Table 2: The P-FTP Server Response Time for Different Update Frequencies.

of *Ping* and *Pchar* as 4 and 30 hours respectively. Table 2 shows, query response time, and utilisation of P-FTP server, serving 100,000 users with different update frequencies of *Ping* and *Pchar*, while the P-FTP database consists of 10,000 FTP server's records. By increasing the update frequencies of *Ping* and *Pchar* more than 2.5 and 21 hours respectively, P-FTP server becomes unstable as its utilisation is more than one.

3.5 Evaluation of Shared congestion Detection Ability

We conducted laboratory tests on isolated networks, to evaluate P-FTP client's shared congestion detection ability and the effect of that ability / adaptation on clients performance. We chose four topologies, to test the shared congestion detection ability of P-FTP client (topology 1 - 4). Each topology consisted of seven FTP servers with different number of shared links among those servers. During the experiments, client started file transfer from servers 1 to 4 in the beginning. If shared congestion was detected, backup servers 5 - 7 were contacted. The results were collected for every topology, for three different cases. The first case was when there were no common congested links present. The second case was, when the servers shared congested links but P-FTP client did not use the adaptation technique. The third case was when the servers shared the congested links and the P-FTP client dynamically changed the file transfer sessions after successful detection. More details of these tests are not provided due to space restrictions but can be found in [31].

The tests results of P-FTP client's shared congestion detection ability are shown in table 3. We only

discuss the results of topology 1 as an example as the results of other topologies follow the same pattern. For topology one (two servers 1 and 2 shared common links), in case 2, transfer time was 320 secs, almost double from case 1, 165 secs. The huge difference in the download time of case one and two showed the degradation in P-FTP client's performance when P-FTP client did not have the ability to adjust dynamically. Large increase in transfer delay, justified the integration of shared congestion detection with P-FTP functionality. In third case, the client started connections with additional servers after terminating connections to the servers sharing common congested links. The transfer delay was 180 sec in this case, which was much less than second case of 320 sec. The difference of transfer delay among second and third case enforced the validity of shared congestion detection algorithm. There was, a 15 sec difference in download time for case 1 and case 3. It was the time P-FTP client took to: 1) detect shared congested links, 2) terminate connection to the servers sharing congested links, and 3) start new connections with backup servers. Hence, the overhead caused by the shared congestion detection ability of P-FTP client for topology one network, was less than 10% (180 secs from 165 secs). The small overhead was justified as without this ability the download delay would increase by more than 90 % (320 secs from 165 secs). The maximum overhead due to P-FTP adaptation observed during all tests was 20% (topology 4, 193 sec from 165 sec).

The column 7 of table 3, shows the minimum number of packets required to detect the presence of shared congestion in each case. The packets referred here are the ones that came from the common congested links. The implementation of shared detection algorithm for the P-FTP client has some limitations. The percentage of error, in terms of wrong detection, of shared congested links is indicated in last column of table 3. The error rate was highest, when there were multiple occurrences of common congested link, among disjoint sets of mirror servers, as in topology two.

Topo-logy	Servers Sharing Common Link	Conges-tion at Common Link	Shared Conge-stion Detection	Servers Added After Detection	Down-load Time (Sec)	Packets Needed	Error %
1	1 and 2	No	-	-	165	-	-
1	1 and 2	Yes	No	-	320	-	-
1	1 and 2	Yes	Yes	One	180	100	5
2	1 and 2, 3 and 4	No	-	-	165	-	-
2	1 and 2, 3 and 4	Yes,Link	No	-	300	-	-
2	1 and 2, 3 and 4	Yes, Link	Yes	Two	188	150	12
3	1,2 and 3	No	-	-	165	-	-
3	1,2 and 3	Yes	No	-	495	-	-
3	1,2 and 3	Yes	Yes	Two	190	160	8
4	1,2,3 and 4	No	-	-	165	-	-
4	1,2,3 and 4	Yes	No	-	654	-	-
4	1,2,3 and 4	Yes	Yes	Three	193	180	10

Table 3: Results For Shared Congestion Detection Algorithm.

4 Simulation

Simulation study was performed to evaluate the performance of P-FTP clients, with widely used discrete event network simulator, Network Simulator 2 (NS-2) [40]. A new application and a new agent were introduced to test the idea of multiple FTP transfers to a single client simultaneously. The simulation results were validated, using the quantitative stochastic simulator Akaroa-2 [46], in independent observation mode due to the independent nature of the observed parameters. The simulation source code is available at [42]. As we previously discussed, the initial design [9] of P-FTP has been modified but the simulation results discussed in section 4.1 are identical to the results presented in [9] as these modifications were implementation specific and affected large scale deployment only.

Topologies: - Different network topologies were generated with the Boston University Representative Internet Topology Generator (BRITE) [45]. To test the performance of the P-FTP approach, we used two topologies. Topology 1 contained 10 nodes and 19 links and was created to compare P-FTP with another parallel downloading approach. Topology 2 was created to study the effects of P-FTP on the download

Mirror Server	Delay (sec)	Suitability	Ranks
3	0.03024	0.2691	2
4	0.0452311	0.17994	3
5	0.02016	0.40372	1
6	0.0553004	0.1471	4

Table 4: Mirror Servers Delay and Suitability.

Mirror Server	Download Time (sec)
3,4,5,6 (P-FTP)	2.3163
3 (FTP)	6.23067
4 (FTP)	9.33167
5 (FTP)	4.15397
6 (FTP)	11.4014

Table 5: Download Times.

time for very large files in a large network. Topology 2 was a flat router-level topology based on Waxman's probability model with 900 network nodes, where each network node was a router and the capacities of the network links were distributed in the interval [5, 155] Mbps. Both topologies contained ten mirror servers and five requesting clients, which were connected to the routers randomly with 10 Mbps links having 5 msec delays. Multiple traffic sources and sinks were introduced in the network randomly to produce competing traffic.

Methodology: - During the simulation, file transfer times were calculated for the P-FTP and traditional FTP approaches. The file sizes ranged from 500 KBytes to 5 MBytes for topology 1 and 5 MBytes to 50 MBytes for topology 2. Each file was replicated on at most five mirror servers. The end-to-end delay between each mirror server and each client was calculated by monitoring agents introduced on the servers and the clients. The P-FTP clients sent the request to the P-FTP server, which contained the file name and the available last hop bandwidth. The P-FTP server calculated the file portions to be downloaded, from selected mirror servers and sent this information to the requesting client. The mirror servers started transferring the allocated file portions to the client after receiving a FTP request from the client. On receiving the complete file, the client calculated the time taken to completely download the file. The time was calculated from the instant the client first requested the P-FTP server, to the instant the complete file was transferred.

4.1 Simulation Results

We explain the P-FTP file transfer process by taking the example of client 5 and file 4 in topology 1. File 4 was replicated on four mirror servers 3, 4, 5 and 6. The end-to-end delay values between client 5 and the

mirror servers (taken from the database), are shown in table 4. For this particular case, end-to-end delay was taken as the only optimisation variable. The end-to-end delay is an inverse optimisation variable so the S_k values for all mirror servers were calculated using equation 2:

$$S_3 = \frac{\frac{1}{IOV_3}}{\sum_{i=3}^{6} \frac{1}{IOV_i}} = \frac{33.0687}{122.862} = 0.2691 \quad (11)$$

Similarly, the suitability values for mirror servers 4, 5 and 6 were calculated as shown in table 4.1. The mirror servers were ranked based on their suitability values. Mirror server 5 was ranked first, with highest suitability value (0.40372). The access link bandwidth was not the bottleneck in this case; hence all mirror servers containing the file were selected for the P-FTP transfer. Due to the use of a single IOV, the respective mirror server's suitability value indicated the file portion to be downloaded from that mirror server. Hence, it can be seen that mirror server 6, which had the highest end-to-end delay from the client, transferred the smallest file fraction (only 0.14). Client 5 calculated the download delay after receiving the complete file using P-FTP. The client also downloaded the same file from each mirror server separately using traditional FTP and measured the download time. Table 5 shows the download time for P-FTP and the traditional FTP approach.

Figure 1 shows the improvement in terms of time when files of different sizes were downloaded using the P-FTP as compared to the traditional FTP in topology 2. The time difference was more obvious for large files than small files; in the latter case the overhead of the message exchange between the P-FTP entities overshadowed the delay measurements. The time measurements in figure 1 with P-FTP did not follow a straight line, although the download time for normal FTP was directly proportional to the file size. The reason was that for P-FTP each file was downloaded from a different set of servers whereas normal FTP downloaded all files from the geographically closest mirror server.

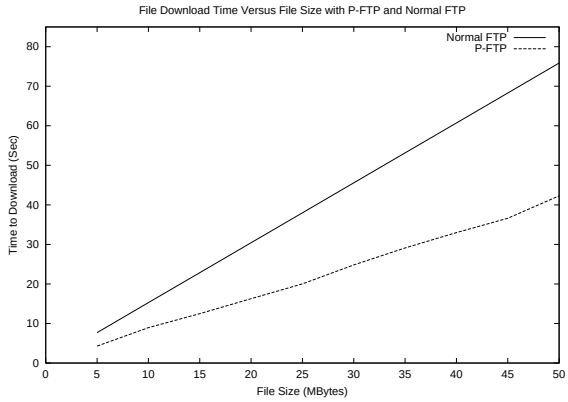

Figure 1: Simulation Results for Topology 2.

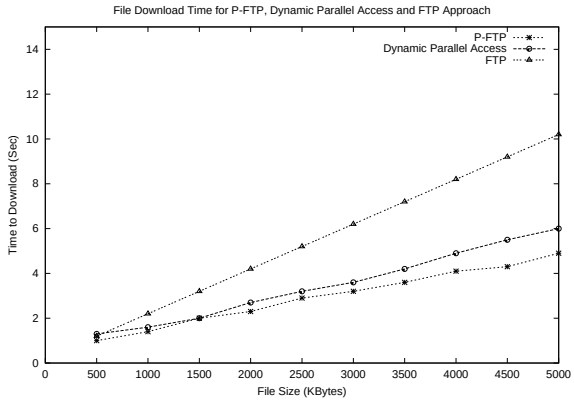

Figure 2: P-FTP and DPAT.

4.2 Comparing P-FTP with DPAT

Other parallel downloading techniques have been briefly discussed in section 2. In this section we discuss and compare P-FTP with the Dynamic Parallel Access Technique (DPAT), proposed in [8], which closely relates to our work. We compare both approaches on the basis of their working model and simulation results.

DPAT proposes downloading large files by connecting to multiple HTTP servers simultaneously. The file is partitioned into small blocks and the client requests a block from each server. When transfer of one block from any mirror server is completed, the client requests the next block from the same server, until the complete file is received. The fast HTTP servers that transfer file blocks rapidly are the ones that send the most blocks.

The most significant difference among two approaches is that P-FTP, in contrast to DPAT, considers network characteristics and mirror server's utilisations before selecting the most appropriate mirror servers. Collins and Plank [27] demonstrated that the server selection techniques based on bandwidth prediction results in better performance than random selection, even for DPAT like approaches. DPAT introduces some ambiguities, regarding the calculation of block size, number of mirror servers to be used and the mechanism to collect information about mirror servers. Another important limitation of dynamic parallel

access technique is that the mirror servers do not have any knowledge about the next block of file they are expected to send until the transfer is requested. This introduces inter-block gaps [8]. The excessive number of seeks on hard disks can produce performance degradation, even when few sessions are in progress, as each session requires multiple seeks. The results presented in [26] were sensitive to block size for block based download approach like DPAT. Dynamic parallel access technique initiates multiple TCP sessions at the beginning of the file download. This can produce additional congestion over the last-hop link when the last hop link is the bottleneck, such as when client is connected via a dial-up connection to the Internet. In contrast, P-FTP is fair as the selection of mirror servers is based on the available bandwidth between client and its Internet gateway. When the client is connected to the Internet via a slow link, P-FTP initiates a single connection to download complete file. DPAT requires re-transfer of a complete block from another server when a server fails to fully transfer a block. In case of large blocks, a lot of data can be lost. Whereas, P-FTP is a byte oriented approach and in case of server failure no data is lost. For P-FTP the penalty is in the form of extra time requirement to establish connection to a new server.

For comparison of both approaches, files ranging from 0.5 - 5 MBytes were downloaded using both approaches and traditional FTP for reference. The simulation setup was same as the one discussed in section 4.1. The P-FTP and dynamic parallel access technique showed significant impact on download delay compared to normal FTP as shown in figure 2. The simulation results showed that both approaches were better alternatives, for downloading large files than the traditional download method. Figure 2 also shows up to 30% less download time for the P-FTP approach compared to DPAT. This demonstrates that P-FTP is better than either of the other approaches. The impact is more vivid for large files as for small files the message passing among P-FTP entities overshadows the download delay.

We would like to emphasis that approaches like DPAT derives the network and system status implicitly and adapts dynamically. Due to which, they do not need additional infrastructure support and hence, are

much easy to deploy. P-FTP is an approach that explicitly requires infrastructure support and hence, is more complex to deploy. For large files the performance gain with P-FTP as compared to DPAT, is significant (up to 30 %). Hence, both of these approaches can be used by future tool developers depending upon the availability of infrastructure support.

5 Empirical Evaluation of P-FTP

To study the behaviour of P-FTP on the Internet, we implemented the P-FTP server and the client. The implementation source code is available at [41]. Planet-Lab was selected to test our P-FTP implementation because of its vast deployed infrastructure [39]. After careful observation of various Planet-lab nodes and their access links, sixteen nodes were selected and configured as FTP mirror servers. Five nodes were in Asia (Asia-1 to Asia-5), seven nodes were in Europe (Europe-1 to Europe-7) and four were in America (USA-1 to USA-4).

The *UNSW.edu.au* domain was used as client's AS and the P-FTP server and the client were placed in this network. The P-FTP server collected QoS parameters along the path between host network and mirror servers using *Pathchar* [38], once every hour. *Pathchar* measures bottleneck bandwidth, Round Trip Time (RTT), number of hops and packet dropping probability, along the path, between the host network and each mirror server [38]. Several experiments were performed to download an 8-MB file with P-FTP and traditional FTP. The experiments were conducted from 20 to 40 times for each test, with a randomly chosen interval of time between each run.

5.1 Sensitivity Analysis of Network Parameter Based Server Selection

A set of tests was conducted to find the relationship between the file download time and the network parameters measured with *Pathchar*. As mentioned earlier, *Pathchar* measures bottleneck bandwidth, RTT and dropping probability along a network path. The dropping probability of the packets, was not used for file

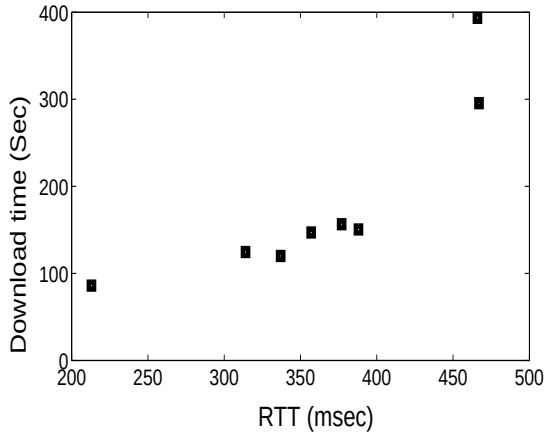

Figure 3: RTT vs Download Time.

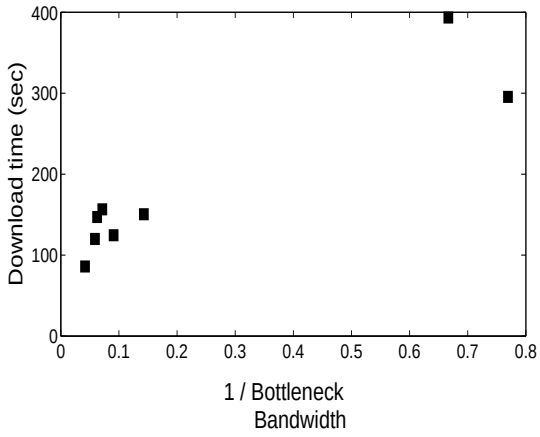

Figure 4: 1/Bandwidth vs Download Time.

server ranking process. The reason is that *Pathchar* measures dropping probability of ICMP packets, which is not applicable to data packets as network routers drop ICMP messages quickly even in case of minor congestion. The average transfer time of an 8 MBytes file downloaded from eight Planet-Lab nodes was measured. The average bottleneck bandwidth and average round trip time measured with *Pathchar*, were used for ranking of those file servers. Figures 3 and 4 show the relationship of file download time with RTT and inverse of bottleneck bandwidth respectively. We used linear regression in Matlab [51], to predict the file download time of the servers based on bottleneck bandwidth, RTT and their combination. The coefficient of determination (R^2) [32] for these three cases were 0.2, 0.5 and 0.9 respectively. Hence, we considered the combination of bottleneck bandwidth and RTT for predicting the download time. Following equation shows linear relation calculated by Matlab:

$$DownloadTime = \frac{247.44}{BottleneckBandwidth(Mbps)} + 344RTT(sec) \tag{12}$$

Figure 5 compares the measured download time with the one predicted using equation 12. The difference of the measured and predicted values in figure 5 is very low, thus, equation 12 gives good prediction of file download times. The linear relation needs to be readjusted after a predefined time interval, depending upon the database update.

We have performed sensitivity analysis of the predicted download time. The four servers with lowest predicted time (calculated with equation 12) were selected for file download. The S_k values of the selected servers were calculated with equation 2, using predicted time as the inverse optimisation variable. The suitability values of the selected servers indicated the file fractions to be downloaded from each respective server. Table 6 provides the comparison of P-FTP's file server selection and file portion calculation processes with two other methods. Method 1 downloaded equal portions of the file from the best four file servers. We considered this method to evaluate the effect of P-FTP's file partitioning process on the download time. Method 2 downloaded equal file portions from randomly selected file servers. The results with all methods were collected while avoiding the shared bottlenecks among different servers for fair comparison. The experiments were conducted for 10 - 30 times for each method. The results in table 6 shows that average download time was smallest with P-FTP (47 sec), followed by method 1 (52 sec) and finally method 2 (77 sec). From the results it is clear that the network parameter based selection of the file servers greatly affected the file download time as P-FTP performed significantly better than method 2. However, the file partitioning did not significantly impact the download performance as the performance of method 1 was almost same as P-FTP. The download times, measured with P-FTP showed least standard deviation as shown in table 6. Figure 6 shows, the Cumulative Distribution Function (CDF) of the download time of three methods. Figure 6 shows that the results measured with P-FTP were distributed in a considerably smaller range than other two methods which imply that most values measured with P-FTP were close to the average value. Hence, the download time was more predictable with P-FTP, than other two methods. Even though the average download time measured with method 1 was almost same as P-FTP, the standard deviation and CDF of download time of the three methods showed that P-FTP performed file transfers, in most predictable time.

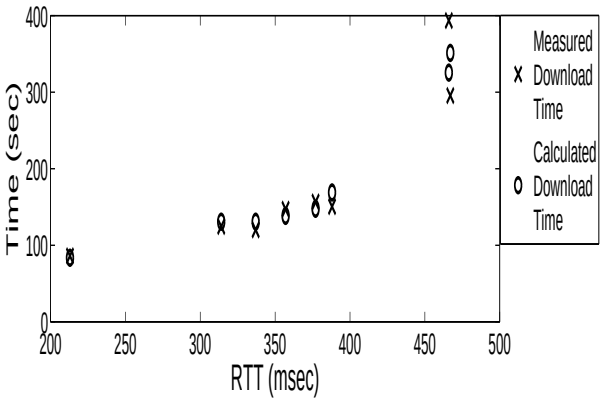

Figure 5: Comparing Download Times.

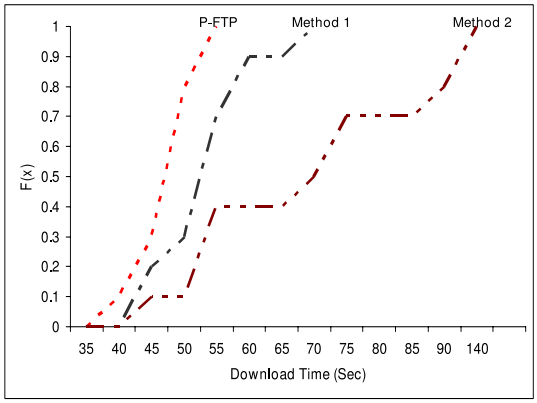

Figure 6: Cumulative Distributive Function.

Method	Server Selection	Partitioning	Download Time (Sec)	Std Dev
P-FTP	P-FTP	P-FTP	47	5.3
1	P-FTP	Equal	52	7.6
2	Random	Equal	77	31.7

Table 6: Sensitivity Analysis.

5.2 Number of Servers and Download Time

Different numbers of servers were selected for multiple P-FTP tests, in order to study the effect of number of servers on the download delay. Results are shown in figure 7. The vertical bars in the background indicate the average file fraction downloaded from each server (left-hand scale). For example, when there are two servers involved in P-FTP, about half of the file is transferred by each server on average, i.e., there are two almost equal sized vertical bars at position 2 on X-axis. The thick black line in the foreground of figure 7 shows the average download delay (right-hand scale). For example, when four servers are involved in a P-FTP session, the average download time is around 43 seconds.

The results indicate that the download time with P-FTP depends upon the number of servers involved to some extent. The download time reduces from 120 to 75 seconds when the number of servers increases from two to three. Similarly, the download time of the file reduces considerably when the number of servers

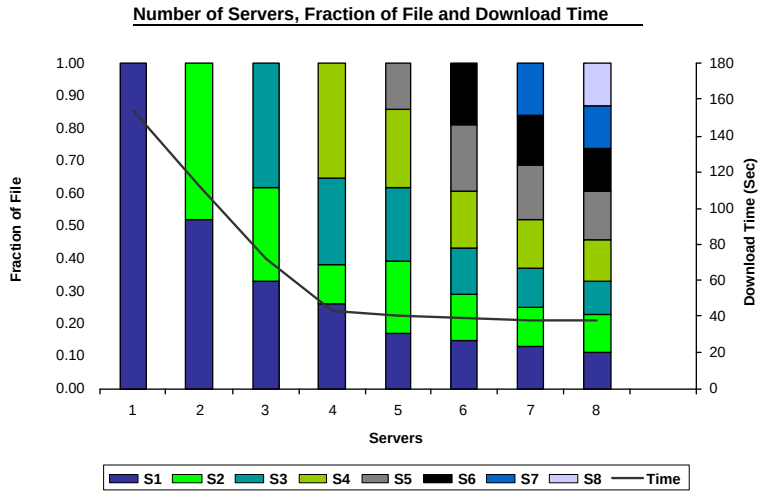

Figure 7: Download Time/ File Fraction VS No. of Servers.

in the P-FTP was incremented from three to four. The download time did not change much when the number of servers was further increased. The client's access link was able to support up to four parallel file transfer connections without congestion. Hence, the file transfer time reduced with increase of connections from one to four. However, congestion was produced at the client's access link when connections were further increased and the download time did not change. Hence, choosing the appropriate number of parallel connections for efficiently downloading a single file is crucial and P-FTP considers this factor before starting the download process (discussed in section 3.2.2).

5.3 Comparing P-FTP and FTP

P-FTP was tested with a number of mirror servers under different conditions. Due to space limitations we are presenting results of one representative case only, more results can be found in [31]. Table 7 shows the time taken to download an 8-MB file with P-FTP, using three servers. The table also shows the time taken to download the same file from each of those servers individually using FTP. The minimum, maximum, average and standard deviation of the measured download times are also shown in table 7. The ranking

Downloaded From	Approach Used	Min (Sec)	Max (Sec)	Avg (Sec)	Std Dev
Asia-1	FTP	146	229	185	16
Europe-1	FTP	94	217	150	31
Europe-2	FTP	74	207	135	32
Asia-1, Europe-1, Europe-2	P-FTP	48	91	72	10

Table 7: Download Times.

and selection process during the tests used the predicted download time (calculated with equation 12) as the only IOV. The server availability was not considered during the file portion calculations because the Planet-Lab infrastructure provides limited root access to the users. It was impossible for us to find out CPU and memory usage of the mirror servers during tests. The file download time with P-FTP was 48 to 91 sec, with an average of 72 sec. When the same file was downloaded from each mirror server independently, the average FTP transfer even from the fastest server required more than 135 seconds. Hence, the download time reduces to about half when P-FTP was used.

An interesting aspect of the results shown in table 7 is the difference in the standard deviations of the download times. Download time measured with P-FTP had significantly less standard deviation than download times measured with FTP. The small standard deviation of download time shows that files can be downloaded in a predictable range of time with P-FTP.

6 Large Scale Deployment of the P-FTP

As discussed in the earlier section, the implementation results of P-FTP show significant improvement in terms of the download time for the P-FTP clients. However, it is important to note that these results were collected from the perspective of a single client. The natural question that follows is whether one would expect similar performance improvements if a large number of clients were downloading files using P-FTP. Another issue that needs to be addressed before P-FTP can be widely deployed on the Internet is the effect of P-FTP on other clients not using this approach. It is of paramount importance that the large user base of P-FTP clients does not affect the performance of other clients who are downloading files using the traditional

single client approach. Our main goal in this paper is to answer these questions and to highlight the fact that P-FTP can indeed be suitably deployed on a large scale in the Internet. To this end, we have carried out a rigorous set of simulations and analysed the results.

6.1 Simulation Study

Ideally, we would have liked to perform Internet experiments similar to the ones conducted in [10] using PlanetLab. We are interested, however, in considering scenarios with large number of clients and we also desire to provide more control and repeatability than would be possible using Internet experiments. We have, therefore, conducted a set of simulations as described below.

Topology: -The network topology considered for these simulations is similar to the one used by the authors of [21]. A topology consisting of 100 routers and 10 mirror file servers, was generated with Boston University Representative Internet Topology Generator (BRITE) [45]. The file servers, the clients and the P-FTP server were connected to a randomly chosen router. The clients connect to the file servers using the Full-TCP model of ns-2. Background load was introduced into the network using on/off UDP flows at random paths of the network. The simulations were conducted with three different numbers of clients depicting three level of load in the network; 1) 100 clients as low load, 2) 200 clients as medium load, and 3) 300 clients as high load. All the results were collected at steady state and average values were calculated.

Methodology : -In our experiments, we compare the download delays perceived by clients using two download techniques: Single Download Technique (SDT) and P-FTP. SDT clients download files using TCP/IP protocol suit from a single file-server. Two approaches were used to select the file-server for SDT file transfers, the random selection (SDT-R) and the selection of the geographically closest server (SDT-L) from the client. P-FTP clients send a request to the P-FTP server before starting the file download process. On receiving the request, the P-FTP server calculates the file portion values for the selected file-servers on the basis of the information maintained in the database. During simulations, the database only consisted of

information about end to end delay between the clients and the file servers. New monitoring agents were designed for NS-2 to measure the end-to-end delay between the file servers and the clients at predefined intervals. The file download time for the P-FTP clients was measured from the instance the request was sent to the P-FTP server to the instance the complete file was received at the client. This is to incorporate the delay due to the message passing among P-FTP server and the clients in file download.

The inter-arrival time between the file transfer requests were chosen randomly for every request of each client. For accurate comparison of the download techniques it is important that the behaviour of the clients is independent of the approach being used for file transfer. As a result, the request inter-arrival time is kept independent of the download technique and the download delay.

6.2 Effect of Load Increase

A set of experiments was conducted to measure the file download time for the P-FTP and SDT clients for low, medium and high load. During simulations P-FTP and SDT clients randomly chose a file size between 1-10 MBytes to download. We calculated the file download time for different files for the P-FTP and SDT approaches under different load conditions. Under all types of load the download time for the P-FTP clients was much lower than the download time for both types of the SDT clients. The load increase influences the performance of both the approaches. To study this effect in greater detail, we calculated the percentage increase in download time for P-FTP and SDT-L approaches when number of users was increased from 100 to 200 and 300 respectively. The results are shown in figure 8. The base value for the calculations was the download time of the respective approach with 100 users. Figure 8 shows that the impact of the load on both approaches was almost similar. Figure 9 shows the mean percentage increase in the download time for the P-FTP, SDT-L and SDT-R approaches. The mean increase in download delay for SDT-L and P-FTP for medium and high loads (200 and 300) was almost the same. This indicates that an increase in load degrades the performance of both the approaches in a similar manner. The mean increase in delay of the SDT-R

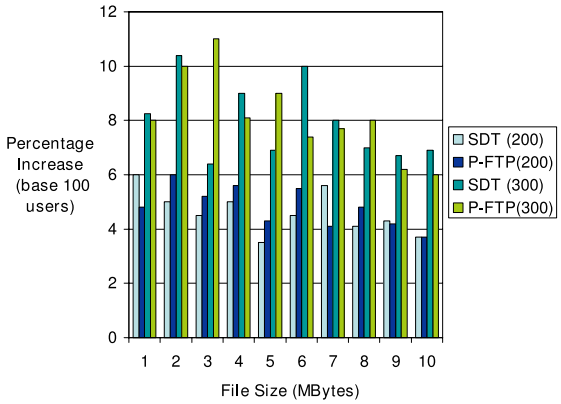

Figure 8: Percentage Increase in Download Time.

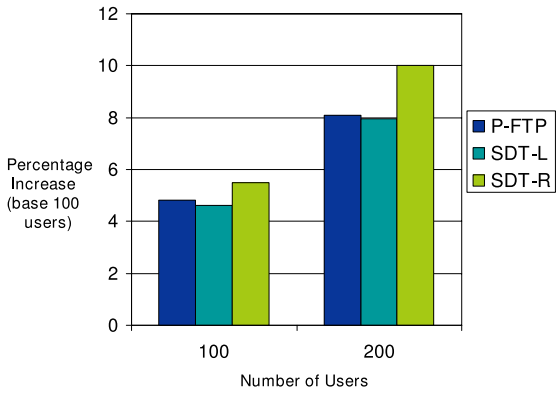

Figure 9: Mean Percentage Increase.

approach was higher for 300 users. This was possibly due to the fact that the randomly selected server in this approach could be geographically distant from the client and under high load this could increase the download time considerably.

It is important to point out that the performance degradation of the P-FTP approach with increase in load, is similar to what one would expect with any other application in a similar scenario. The effect of load increase on the P-FTP approach, is not a disadvantage of its large scale deployment but is the inherent behaviour of any application. It is especially important to highlight the fact that the increase in download time with P-FTP is not an exponential function of the load, but is rather some form of linear function for the tested scenarios. This implies that the saturation effect is not very severe and hence indicates that P-FTP is actually quite scalable. This set of experiments also convincingly shows that the performance of the P-FTP clients is signifi cantly better than the traditional fi le transfer clients, even when large number of users adopt the respective approach. Thus, P-FTP is highly scalable and suitable for large scale Internet deployment. These results are signifi cant for other parallel downloading approaches like DPAT as well.

6.3 Effect on File servers and non P-FTP Users

P-FTP is based on the initiation of multiple file transfer sessions by the clients to accelerate the download process. In other words, the P-FTP clients initiate more short file transfer sessions as compared to a single long file transfer session by the traditional clients. Hence, there are more connection initiation requests handled by the file servers when users use P-FTP. The processing time of the file transfer initiation request at the FTP server was calculated on a Pentium 3 machine with 256-MBytes RAM. The effect of already established FTP connections on the overall response time of the request was also studied by sending different number of simultaneous requests to the server. For every connected client, the processing cost increased by approximately 2-3 msec. The difference in processing time of a P-FTP request from that of a SDT request, is due to different number of already established connections at the file servers in each case. In case of all P-FTP clients the number of connections in the network and at the file server is more than the case of all SDT clients. We monitored the number of simultaneous connections in the network during the simulations discussed in section 6.2. The maximum number of simultaneous connections for P-FTP was 85 for high load and that for SDT was 55. On average each of the ten file servers had 3 more connections in case of P-FTP, which added a delay of 6-9 msec to the processing time of the P-FTP request as compared to the SDT request. However, one can readily observe that this small difference in processing time does not produce any noticeable impact on the download time of both approaches as the difference was in multiple seconds.

Due to initialisation of more file transfer sessions at the file servers, the P-FTP approach can produce extra load when large number of clients are using this approach. If there is large number of P-FTP clients requesting for new connections to the file servers, the excessive load can degrade the performance of the file servers and all the users using those file-servers may get affected. We have considered this factor and hence, restricted P-FTP for transfer of large files only as: firstly, the large files constitute a small ratio of

total downloads on the Internet, and secondly, the download time of large files is reduced significantly when downloaded with P-FTP. We have performed some simulations to study the effect of these factors. For the study of the file size distribution pattern, we used the same FTP traffic logs of school of CSE of UNSW, as discussed in section 3.4. From those traffic logs we were able to come up with the file size and frequency distribution for FTP transfers for the users of school of CSE, shown in figure 10. We performed simulations for this real world file size distribution (taken from figure 10) with 300 clients (heavy load). Files with sizes 1 MByte or more were called large files and all other files were called small files. Three cases were used for simulations: 1) all files were downloaded with SDT-L approach, 2) all files were downloaded with P-FTP approach, and 3) large files were downloaded with P-FTP and small files with SDT-L. The download times measured during simulations for these three cases are shown in figure 10. Figure 10 shows that:

1. The download time of small files was slightly improved with P-FTP as compared to the traditional approach. When the small files were downloaded with P-FTP (*ALL P-FTP* in figure), the change in download time was very small as compared to when they were downloaded with SDT-L (*ALL SDT-L* in the figure).

2. Large files were downloaded in significantly less time with P-FTP as compared to the traditional approach. The difference in download time of large files was significant when downloaded with P-FTP (*ALL P-FTP,P-FTP + SDT-L* in the figure) as compared to the SDT-L approach.

3. If only the large files were downloaded with P-FTP, the download process of the small files did not get influenced. The download time of small files was the same for *P-FTP + SDT-L* and *SDT-L* cases. This shows that the download delay of the large files can be reduced considerably with P-FTP without affecting other downloads.

Figure 10 shows that, by restricting P-FTP for large files, we accomplish two goals: 1) The burden of extra

file transfer session initiations on the file servers is minimised, and 2) the large files are downloaded in reduced amount of time without affecting the download process of other files. Therefore, we propose that P-FTP is suitable for transfers of large files only. When only the 10% of the files are downloaded with P-FTP then the excessive load on the file servers is very low. This small extra load on the FTP servers (usually dedicated low cost PCs) can be justified against the gain in the download time for the P-FTP clients. Secondly, the P-FTP server tries to distribute load among different set of file servers during intense load conditions.

6.4 P-FTP Behaviour under High Network Congestion

We have conducted a set of simulations, to study the behaviour of P-FTP system, under high network congestion. The access link bandwidth of the file servers was reduced, for these simulations to quickly achieve network congestion. Figure 11 shows the average download time of an 1 MByte file for P-FTP and SDT-L clients with the increase in load. The download time of SDT-L clients increased linearly with the increase in load until 400 users. After that, due to network congestion, the download time increased exponentially, with the increase in load until 500 users. When the load increased further than 500 users, the system became unstable and download times were unlimited for SDT-L clients. The download time of the P-FTP users increased linearly and in same proportion as the SDT-L clients with the increase in load until 400 users. When the load increased to more than 400 users, download time of P-FTP clients increased exponentially with the increase in load. The rate of increase was greater than SDT-L clients in this case after 450 clients. The reason for this was that there was a higher probability for P-FTP clients to contact the highest congested file servers than SDT-L clients, as P-FTP clients downloaded one file from four servers and SDT-L clients downloaded entire file from a single server. The dynamic adaptation ability of the P-FTP clients did not affect the download time much as all the backup servers were also very congested most of the time due to very heavy load conditions. Hence, near maximum capacity of the system P-FTP's performance

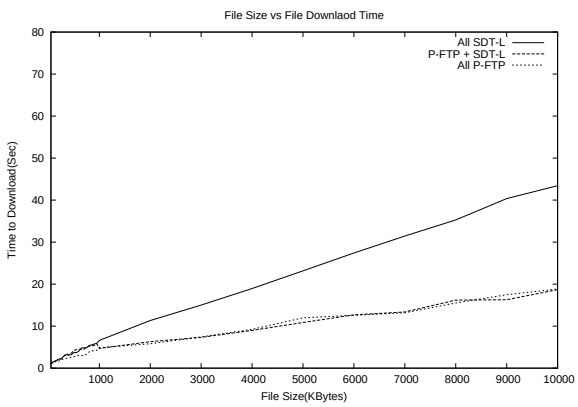

Figure 10: Result for Real-World File Size Distribution.

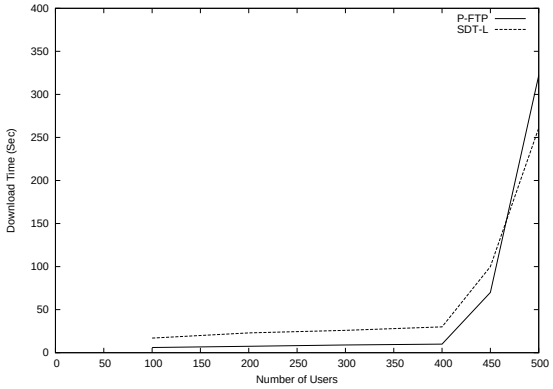

Figure 11: P-FTP vs SDT-L.

degraded signifi cantly. P-FTP clients experienced unlimited download times due to system instability when the load was increased beyond 500 users similar to SDT-L clients. The system collapsed at the same load for both approaches.

We can conclude, from the results that P-FTP performs much better than traditional approach, when the network load is low or medium. However, as the load increases to the maximum capacity of the system P-FTP's perform worst than traditional approach. Therefore, under such conditions, either traditional approaches should be considered, or the P-FTP's server selection process should change to load balancing among all available fi le servers, as discussed in section 3.2.2. In addition, P-FTP does not affect the network stability under extreme load conditions.

7 Discussion

There are certain issues related to P-FTP that we have not discussed in this paper due to space restrictions. We proposed the decentralisation of P-FTP server on AS basis for feasibility and scalability. However, for very large ASs a single P-FTP server may not be adequate and a set of hierarchal P-FTP servers [24] [25] may be considered. In this case, a separate P-FTP server is provisioned for different parts of the AS. The most suitable hierarchal structure would be the one based on network measurements, specifi cally number of hops. This would enable users to query the P-FTP server that is closest to them geographically. Network

traffic, generated by multiple P-FTP servers, can be minimised by using statically configured topological information and using database mining techniques. Other issues related to hierarchal servers like database sharing, interaction among servers etc are out of the scope of our work.

Parallel access approaches, like P-FTP and DPAT, can only work when mirror servers contain identical copies of the requested file. A hashing tag can be used by the mirror servers to provide information about the version of the requested document [8]. P-FTP targets file transfer, in client-server paradigm of the Internet. However, multiple other applications can benefit from P-FTP such as, proxy caching and large storage systems.

P-FTP initiates single connection to multiple servers, for parallel downloading. Another parallel downloading method is to start multiple connections, to a single server. The selection of the fastest server in this case, can result in very good performance. However, the server selection process is based on previously measured network parameters and resource availability changes very quickly due to Internet dynamism. The wrong selection can enormously increase download time in this case. By selecting multiple servers the load is shared among servers and large number of users can experience better performance [8].

8 Conclusion and Future Work

In this paper, we have presented a novel file transfer protocol called Parallelized File-Transfer-Protocol (P-FTP), which reduces the download time for large files on the Internet. The design and implementation of our approach focuses on the reduction of transfer delays by simultaneously downloading disjoint file portions from multiple file servers. A distinguishing feature of P-FTP is that the file servers are selected on the basis of the QoS parameters along the network path and the server utilisations. The approach is tested via simulations and real-world experiments on the Internet. The results of our approach show that the large file transfer delays can be reduced by more than 50% in most cases. An important issue, concerning the deployment scalability of P-FTP, is also studied in this paper. A rigorous simulation based evaluation is

presented to highlight the important properties of P-FTP, which make it suitable for widespread deployment in the Internet. We, conclusively, show that the download time for P-FTP clients, is signifi cantly lower as compared to traditional clients, even when large number of simultaneous clients is present in the network. Secondly, the effect on the network and the fi le servers is minimised due to the self tuning ability of the approach. In addition, by restricting the use of P-FTP for downloading large fi les only, we also show that there is no degradation in the performance of other clients who are using the traditional single server fi le transfer techniques.

There are a number of directions for future work. During the experimental testing we have considered a small number of fi le servers. Thus the P-FTP server does not need any specialised database. Instead, it maintains a data-fi le containing the measured values. However, as the number of servers increases, we anticipate the need of a dedicated database in the P-FTP system. The design issues of P-FTP database are its size, the method to collect fi le server information at start and minimising maintenance overhead. We are currently developing a scalable and effi cient mechanism for the P-FTP database. Multiple factors contributing to the overhead of the approach are also being studied. Merging P-FTP with a peer-to-peer application can contribute to P-FTP effi ciency and we are investigating such a case.

References

[1] S. Sohail, S. Kanhere, S. Jha, A. Baig and M. Malik, *A Parallelized File-Transfer-Protocol for On-Board IP Networks*, In Proceedings of Vehicular Technology Conference, Spring 2005.

[2] G. Pultar, *Automatically Selecting a Close Mirror Based on Network Topology*, In Proceedings of the 12th Systems Administration Conference, 1998, pages 159-165.

[3] J. Guyton and M. Schwartz, *Locating Nearby Copies of Replicated Internet Servers*, In Proceedings of SIGCOMM 1995, pages 288-298.

[4] R. Carter and M. Crovella, *Server Selection Using dynamic path characterization in Wide Area Networks*, In Proceedings of INFOCOM 1997, pages 1014-1021.

[5] Z. Fu and N. Venkatasubramanian, *Directory Based Composite Routing and Scheduling Policies for Dynamic Multimedia Environment*, In Proceedings of 15th International Parallel and Distributed Processing Symposium 2001, page 11.

[6] J. Byers et al., *Informed Content Delivery Across Adaptive Overlay Networks*, In Proceedings of IEEE/ACM Transaction on Networking, Vol 12, Issue 5, 2004, pages 767-780.

[7] M. Crovella and A. Bestavros, *Self-Similarity in World Wide Web Traffic: Evidence and Possible Causes*, IEEE/ACM Transaction on Networking, Vol 5, Issue 6, 1997, pages 835-846.

[8] P. Rodriguez and E. Biersack, *Dynamic Parallel Access to Replicated Content in the Internet*, IEEE/ACM Transaction on Networking, Vol 10 no 4, Aug 2002, pages 455-465.

[9] S. Sohail, S. Jha and H. Elgindy, *Parallelized File Transfer Protocol*, In Proceedings of IEEE LCN, Workshop of HSLN, 2003, pages 624-631.

[10] S. Sohail and S. Jha, *QoS Driven Parallelized File-Transfer-Protocol*, In Proceedings of Australian Telecommunication, Networks and Application Conference, 2004.

[11] S Sohail, C Chou, S Kanhere and S Jha, *On Large Scale Deployment of Parallelized File Transfer Protocol*, In Proceedings of IPCCC, 2005, pages 225-232.

[12] J. Doyle, *Routing TCP/IP Volume I (CCIE Professional Development)*, Cisco Press, 1998.

[13] D. Katabi, I. Bazzi, and X. Yang, *A Passive Approach for Detecting Shared Bottlenecks*. IEEE International Conference on Computer Communications and Networks 2001, pages 174-181.

[14] S. Savage, *Sting: a TCP-Based Network Measurement Tool*, In Proceedings of the USENIX Symposium on Internet Technologies and Systems 1999, pages 71-79.

[15] P. Francis, et al. *IDMAPS: a Global Internet Host Distance Estimation Service*, In Proceedings of IEEE/ACM Transaction on Networking 2001, Vol 9, pages 524-540.

[16] K. Gummadi et al. *King Estimating Latency Between Arbitrary Internet End Hosts*, In Proceedings of the SIGCOMM Internet Measurement Workshop 2002, pages 5-18.

[17] E. Ng and H. Zhang, *Predicting Internet Network Distance with Coordinates-based Approaches*, In Proceedings of IEEE INFOCOM 2002, pages 170-179.

[18] J. Postel and J. Reynolds, *File Transfer Protocol (FTP)*, IETF RFC 959, 1985.

[19] J. Buyers et al., *Accessing Multiple Mirror sites in Parallel: Using Tornado Code to Speed up Downloads*, In Proceedings of IEEE INFOCOM 1999, pages 275-283

[20] Simon Koo, Catherine Rosenberg, Dongyan Xu, *Analysis of Parallel Downloading for Large File Distribution*, In Proceedings of IEEE International Workshop on Future Trends in Distributed Computing Systems 2003, pages 128-138.

[21] Christos Gkantsidis, Mostafa Ammar, Ellen Zegura, *On the Effect of Large-Scale Deployment of Parallel Downloading*, In Proceedings of The Third IEEE Workshop on Internet Applications 2003, pages 79-90.

[22] D. Menasce and V. Almeida, *Capacity Planning for Web Services: Metrics, Models and Methods*, Prentice Hall, Upper Saddle River, New Jersey, 2001.

[23] H. Balakrishnan, M. Stemm, S. Seshan and R. Katz, *Analyzing Stability in Wide-Area Network Performance*, In Proceedings of ACM SIG METRICS Conference on Measurement and Modeling of Computer Systems 1997, pages 2-12.

[24] S. Czerwinski, B. Zhao, T. Hodes, A. Joseph and R. Katz,*An Architecture for a Secure Service Discovery Service*, Proceedings of ACM/IEEE International Conference on Mobile Computing and Networking, 1999, pages 24-35.

[25] D. Xu, K. Nahrstedt, and D. Wichadakul, *QoS-Aware Discovery of Wide-Area Distributed Services*, In Proceedings of First IEEE/ACM International Symposium on Cluster Computing and the Grid, page 92.

[26] J. Plank, S. Atchley, Y. Ding and Y. Beck,, *Algorithms for High Performance, Wide-area Distributed File Downloads*, In Parallel Processing Letters, June 2003, Vol 13, no 2, pages 207-224.

[27] R. Collins and S. Plank, *Downloading Replicated, Wide-Area Files – a Framework and Empirical Evaluation*, In Proceedings of 3rd IEEE International Symposium on Network Computing and Applications 2004, pages 89-96.

[28] M. Allen and R. Wolski, R., *The Livny and Plank-Beck Problems: Studies in Data Movement on the Computational Grids*, In Proceedings of ACM/IEEE SC2003 Conference on High Performance Networking and Computing, 2003, page 43.

[29] M. Naf, *Dynamic Server Selection*, Diploma Thesis, Institute for Computer Systems ETH Zrich, 1999.

[30] A. Gulbrandsen, P. Vixie and L. Esibov, *A DNS RR for specifying the location of services (DNS SRV)*, IETF RFC 2782, 2000.

[31] S. Sohail and S. Jha, *Parallelized FTP:- Effective approach for Solving Huge Download Delay Problem over Internet*, Technical Report 0411, School of Computer Science and Engineering, University of New South Wales, April 2004.

[32] R. Jain, *The Art of Computer Sytems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation and Modeling*, Wiley-Interscience, 1991.

[33] *Bandwidth Measurements by Ping*, http://www.oreilly.com /catalog/ nettroubletools/chapter/ch04.html Apr-2005.

[34] Jigsaw download, http://atterer.net/jigdo/ Jun-2004.

[35] Kazaa, www.kazaa.com Dec-2003

[36] Further Network, www.furthurnet.com Dec-2003

[37] Download Accelerator Plus, www.speedbit.com/DAPDL.asp Dec-2003

[38] Pathchar, http://www.caida.org/tools/utilities/others/pathchar/ Jun-2004

[39] Planet Lab, http://www.planet-lab.org/ Jun-2004

[40] Network Simulator-2, http://www.isi.edu/nsnam/ns/ Jun-2004

[41] P-FTP Implementation Source Code, http://www.cse.unsw.edu.au/šohails/pftp-23jan.tar.gz Jun-2004

[42] P-FTP Simulation Source Code, http://www.cse.unsw.edu.au/šohails/pftp-simula.tar.gz Jun-2004

[43] Bittorent, http://bitconjurer.org/BitTorrent/ Dec-2003

[44] Slurpie, http://slurpie.sourceforge.net/ Jun-2004

[45] BRITE, http://www.cs.bu.edu/brite/ Jan-2004

[46] Akaroa, http://www.tkn.tu-berlin.de/research/ns-2-akaroa-2/ns.html Sep-2003

[47] Globus, http://www.globus.org/datagrid/gridftp.html Jun-2004

[48] Pchar, http://www.kitchenlab.org/www/bmah/Software/pchar/ jan-2004

[49] Ping, http://ftp.arl.mil/ mike/ping.html Jun-2004

[50] Pure FTP Daemon, http://www.pureftpd.org Jul-2004

[51] Matlab, http://www.mathworks.com/ May-2005