On the Impact of Mobile Hosts in Peer-to-Peer Data Networks

Zhenyun Zhuang[†], Sandeep Kakumanu[†], Yeonsik Jeong[‡][*], Raghupathy Sivakumar[†], Aravind Velayutham[§]
[†]{zhenyun, ksandeep, siva}@ece.gatech.edu, Georgia Tech, Atlanta, USA [†]
[‡] ysjeong@mail.skhu.ac.kr, SungKongHoe University, Seoul, Korea
[§]vel@asankya.com, Asankya Inc., Atlanta, USA

Abstract

*Peer-to-peer (P2P) data networks dominate Internet traffic. In this work, we study the problems that arise when mobile hosts participate in P2P networks. We primarily focus on the performance issues as experienced by the mobile host, but also study the impact on other fixed peers. Using BitTorrent as a key example, we identify several unique problems that arise due to the design aspects of P2P networks being incompatible with typical characteristics of wireless and mobile environments. We then present a wireless P2P (*wP2P*) client application that is backward compatible with existing fixed peer client applications, but when used on mobile hosts can significant improve performance.*

1 Introduction

Over the last few years, peer-to-peer (P2P) data sharing applications have experienced an explosive growth. By the end of 2005, a staggering 60% of the Internet data traffic constituted of P2P file sharing [6]. While copyright concerns had earlier brought down popular P2P applications such as Napster, several content owners and providers have of late started embracing what is being seen as a technology that has come to stay [1].

With P2P data sharing applications securing a dominant position in the Internet landscape, an interesting question is to ask is: *what is the performance of mobile users when participating in P2P data sharing?* The question is significant because of two reasons: (a) as with any Internet application with emerging or established popularity, wireless and mobile users are increasingly adopting P2P data sharing applications on devices such as laptops and PDAs [9]; And (b) there are several efforts underway to deliver P2P data sharing as the next killer application for mobile devices [17, 16]. Initial instantiations of such efforts focus on sharing of ringtones and music files, but are expected to evolve into other types of content including video.

Thus, in this work, we first investigate the following question: *what is the performance of a mobile user in a wireless environment using a P2P data sharing application?* As a corollary, we also investigate the following question: *what is the performance of a fixed peer in a P2P network when using a mobile host as a corresponding peer?* In answering these questions, we find that several of the fundamental design principles and peculiarities of P2P data sharing applications are inconsistent with the key limiting characteristics of typical wireless and mobile environments. Briefly, these issues include: (i) P2P applications, unlike in typical scenarios where a mobile host functions as a client, creates a scenario *requiring the mobile host to function as a server*. This raises several implications. (ii) P2P data sharing uniquely involve *simultaneous bi-directional data transfer*. This consequently results in the use of bi-directional TCP, a form of TCP not studied extensively for wireless environments. (iii) P2P data networks, by virtue of being almost entirely supported by end-hosts, typically use *incentives based performance delivery*. Such a mechanism exposes issues when applied as-is to a wireless and mobile environment. (iv) While incentives encourage P2P users to share data longer, P2P data fetching is also adapted to increase the uniquely shareable data available at a user. One such approach is performing *random* or *rarest-first fetching*. However, such techniques have severe implications to the mobile user, especially during disconnections. Using experiments on a real-life P2P network, we profile the performance of a mobile user with respect to these issues.

Using insights gained, we present a deployable solution suite called wireless P2P (*wP2P*) that addresses the issues using changes only to the P2P application at the mobile host. *wP2P* uses techniques transparent to the fixed peer, but uniquely relevant to the specific issues pertaining to wireless and mobile hosts functioning in a P2P data network. While we elaborate on the specifics of the solutions later in the paper, *wP2P* uses a combination of multiple proposed techniques in tandem including age based manipulation, incentive aware operations, and mobility aware fetches. We evaluate *wP2P* with an implemented prototype,

[*]Work conducted when author was a visiting research scientist in the Georgia Tech Networks and Mobile Computing (GNAN) Research Group.

[†]This work is supported in part by the National Science Foundation under grants CNS-0721296, CNS-0519733, and CNS-0519841.

and show that significant performance improvements can be achieved for mobile hosts and fixed peers. Thus, the contributions of this work are twofold:

• We consider the specific scenario of mobile hosts participating in P2P data sharing applications and investigate performance issues such hosts face due to the unique design elements with real-life experiments.

• We present *wP2P* that runs only at mobile hosts. We show *wP2P* addresses the identified issues and delivers enhanced performance with a prototype implementation.

We presents the scope of this work and describes key background material in Section 2. Section 3 presents the motivation results that show the limiting performance existing P2P application design impose on mobile users. Section 4 outlines the key design basis and describes the *wP2P* solution in detail, while Section 5 presents the performance evaluation. Finally, Sections 6 and 7 discuss related-work and conclusions, respectively.

2 Scope and Background

2.1 Scope of this work

• *P2P Networks:* While there are several forms of P2P networks ranging from those that help in computing (grids) to those that help in communication (e.g. skype) to those that help in data-sharing, this work is entirely focused on P2P data sharing networks. Data sharing P2P networks are primarily used for sharing files containing audio (e.g. mp3 files), video (e.g. mpeg2 files), or data (e.g. linux distributions). Examples of such networks include BitTorrent [1], eDonkey, Gnutella, and FastTrack. Recent study show that P2P traffic is dominating Internet traffic and specifically, the BitTorrent [1] P2P network accounts for 30% of the overall Internet traffic [6]. Measurement studies conducted recently observe far more wireless and mobile users on the network than ever before [9].

• *BitTorrent:* In this work, while we identify characteristics of P2P networks that are generically applicable to all four of the above networks, we use BitTorrent as the primary example for all discussions, experiments, and trials. However, as necessary we also step back and investigate relevance of our discussions and interpretations for the other networks as well. We believe that this choice of BitTorrent as the key representative is justifiable from multiple standpoints including its dominance in terms of traffic carried, and its relative sophistication.

• *Wireless Technologies and Mobile Devices:* While mobile users with any type of wireless access can participate in P2P networks, the access technology typically used is wireless LANs (WLANs). This is both because of the higher bandwidths available and the relatively lower or no cost models associated with such networks. Hence, we consider WLANs for the wireless environment in this paper. Similarly, while other mobile devices such as PDAs and IP enabled cellphones are fair game for assuming membership in P2P networks, we primarily consider laptops as the mobile device in this work.

• *Metrics:* We consider throughput performance as the main metric in our evaluation and optimization considerations. The focus is more on the question: *what is the performance of a mobile host when it participates in a P2P network?* In addition, we also consider a corollary question: *what is the performance of a fixed peer when it uses a mobile host as a peer to download data from?*

2.2 BitTorrent

BitTorrent, like other P2P data sharing protocols, uses peers that have downloaded a certain content as the sources for the content subsequently for other peers that need the same content. We now outline some of the key elements of the BitTorrent protocol relevant to the focus of this work.

• *Torrent, tracker, seeds and leaches:* (i) Any peer that wants to share a file through the BitTorrent network creates a ".torrent" file that consists of some meta data information and the address of the *tracker* that will act as the directory server for the file. (ii) A tracker maintains all current peers that have a specific file either in its entirety or in parts. When it receives a request from a client for a specific file, it furnishes the client with the addresses of the peers associated with the file. The list of peer addresses is updated periodically. (iii) All peers downloading a specific file form a swarm, which consists of seeds (peers having the entire file) and leeches (only in parts).

• *Tit-for-Tat and Rarest-first:* A client uploads to peers in the swarm and downloads from other peers. (i) A tit-for-tat incentive policy is used to control the upload rate of one peer to another peer based on the download rate the peer enjoys from that other peer. (ii) BitTorrent peers do not fetch parts of the file in sequence. Instead, each peer picks the *rarest* of the blocks (in terms of the number of peers in the swarm that have the block) preferably to download. This ensures that the rarest blocks of a file are propagated in the swarm faster, reducing bottlenecks at the few peers that have the block and increasing the availability of those blocks if the peers that have them shutdown.

3 Motivation

We perform experiments on a BitTorrent network to study the performance of a mobile host. We identify several design characteristics of P2P data networks that are incompatible with typical characteristics of a wireless environment. While we present all the discussions in the context of BitTorrent, we revisit the implications of the discussions on the other P2P networks at the end of the section. The network testbed used is shown in Figure 1. The testbed consists of six locally controlled BitTorrent peers which run popular

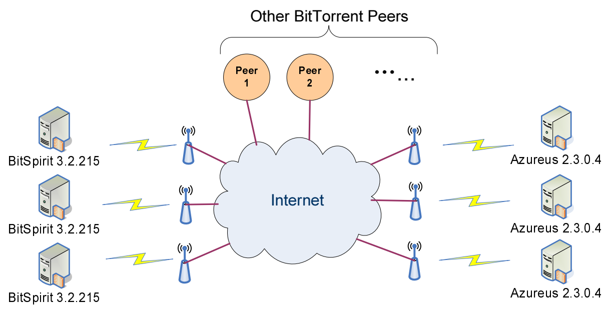


Figure 1. Network Testbed for P2P Evaluation

BitTorrent clients: three run the Azureus client 2.3.0.4 on Linux, and the other three peers run the BitSpirit 3.2.215 client on Windows.

3.1 Bi-directional TCP

As described in Section 2, most peers in BitTorrent upload and download at the same time, and because of the tit-for-tat policy used by BitTorrent, several of the uploads are to peers from which downloads are being done. Hence, *it is common for data to be exchanged simultaneously between peers in both directions*. Given that BitTorrent (or for that matter the other P2P applications) uses TCP, and that TCP is inherently designed to be a bi-directional protocol, BitTorrent uses TCP in its true bi-directional mode. In other words, a single TCP connection is used to transfer data in both directions between the peers. While TCP is designed to be a bi-directional transport protocol, few extensive studies of its behavior have been performed. More importantly, in the context of this work, very little is understood about the behavior of bi-directional TCP in a wireless environment. In this context, we identify the issue of *ACK piggybacking* with the use of bi-directional TCP by BitTorrent.

When bi-directional TCP is used, ACKs in the reverse path are almost always piggybacked on the data packets being sent in the reverse direction. In a wireless environment, where random errors rates can be non-trivial, this potentially has an adverse impact on the connection performance. More specifically, when ACKs are piggybacked on data packets, the effective "length" of the ACKs is longer than if they were sent as *pure* ACKs (non-piggybacked). Hence, for the same bit error rate (BER) in a wireless environment, the effective packet error rate for the ACK traffic is larger. This in turn results in more number of ACK packets being lost on the reverse path just because piggybacking.

While it is true that TCP uses cumulative ACKs, and hence is relatively robust to ACK losses, there still is a negative impact in terms of the overall throughput enjoyed by a connection in the presence of higher number of ACK losses. More importantly, in a P2P network peers typically have a large number of TCP connections ongoing even for a single swarm (BitTorrent trackers typically provide addresses of 50 peers in response to a request, but the overall swarm size can be greater than 1000), resulting in the average congestion window size of a TCP connection to be relatively small. And, it is for connections with small congestion window sizes that a higher ACK loss rate can result in a non-trivial degradation in throughput. In other words, the download rate for a TCP connection from a particular peer will be smaller just because of ACKs being piggybacked in the reverse direction.

We set up two peers which hold different portion of data and measure the throughput when either uni-TCP or bi-TCP is used. Specifically, when two peers hold different data, they exchange data over bi-TCP. When only one peer has the data the other needs, data are downloaded using uni-TCP. Figure 2(a) presents the 5-run averaged results of the download rate experienced by a peer under varying conditions of bit error rate on the wireless leg. Note that bi-directional connections will in general suffer in throughput when compared to uni-directional connections because of the self-contention between packets being sent in the upstream and downstream directions. However, that difference is captured by the data-point at BER=0.

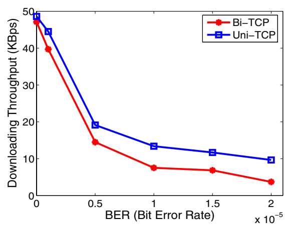
3.2 Uploads based Incentives

The tit-for-tat mechanism in BitTorrent encourages higher rates for uploads to enjoy better download rates. In a wired environment, it can be shown that peers enjoy their best download rates when their upload rates are high. Figure 2(b) shows the aggregate download rate of five simultaneous tasks as a function of the upload rate limit in a wired setting. The upload rate limit is represented as a fraction of the physical link capacity. The network is Comcast Cable High-speed Internet with 4Mbps downloading rate and 384Kbps upload rate. We observe that the download rate is an increasing function of the upload limit.

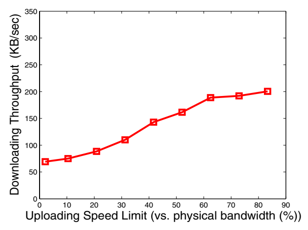
However, for a wireless environment, the relationship between the enjoyed download rate and the upload rate limit changes. Figure 2(c) shows the aggregate download rate of the same five tasks. The network is Georgia Tech Wireless LAN 802.11G. As shown, while the download rates initially increase with higher upload rates, beyond a much smaller upload rate (than 80%) the download rates actually drop. This is due to the *shared* channel nature of the wireless link, where the uploads and downloads are contending for the same wireless channel bandwidth. This is in contrast to a wired setting where the uploads and downloads do not share the same bandwidth resources. The same figure also demonstrates that shutting down the upload is not a solution either as the tit-for-tat strategy of BitTorrent will kick-in.

3.3 Incentives and Mobility

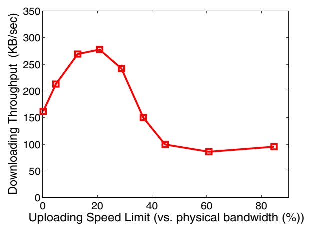
The tit-for-tat mechanism in BitTorrent is associated with a unique identifier for the peer called the *peer-id*. The



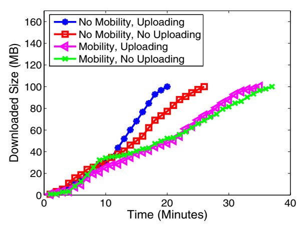
(a) Bi-Directional TCP



(b) Wired



(c) Wireless



(d) Incentive and mobility

Figure 2. Impact of {Bi-TCP(a), upload traffic on downloads (b, c) and incentive and mobility (d)}

peer-id is typically constructed as either a function of the IP address of the host and a random value, or simply as a function of a mobile host specific random value. The peer-id is regenerated every time fetch tasks are reinitiated. Thus when a mobile host experiences a hand-off and receives a new IP address, the ongoing tasks are terminated and the tasks are re-initiated [1] thus generating a new peer-id. However, since the peers track the *goodness* of corresponding peers based on the peer-id, this results in the mobile peer losing all the credit it has built with its corresponding peers.

Figure 2(d) show the effect of incentives on the download size for a 100MB file as a function of time of a typical run. Under the no mobility scenario, we observe that the download size is lower when there is no upload traffic. This is the normal incentive behavior. However when we introduce mobility (IP address changes periodically) we see that the incentive mechanism is rendered ineffective. Not only is the actual download size lower than the no-mobility case, there is marginal difference between the download rates with or without uploading. This is because every time the IP address changes, tasks are re-initiated and thus the host acts as a new peer without any previous incentives. Thus, the mobility of a peer can have an adverse impact on the incentive mechanism of a P2P network.

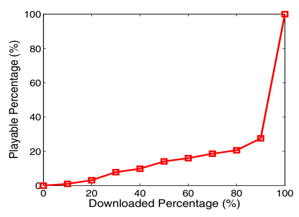
3.4 Rarest-first Fetches

As outlined in Section 2, BitTorrent employs a rarest first fetching paradigm. This results in any snapshot of the downloaded content for a file not having any significant "in-sequence" data from the head of the file till a large percentage of the file download is completed. Many media formats, on the other hand, allow for partial playback of content provided the partial information is in sequence. For example, for an MPEG file of a 2 hour video, the download of the first 30 minutes worth of the video will still allow for a playback of that part of the video. Figures 3(a,b) show the *playable* fraction of two files being downloaded with increasing fraction of the actual downloads using rarest-fetch. The piece length is the default value of 256 KB, and the results are averaged over 10 runs. It can be observed that until a large percentage of the whole file download is complete, a significant percentage of the file still remains unplayable. For a 5MB file, even with a 60% download fraction, less than 10% of the file remains playable. For the 100MB file, more than 90% of the file size needs to be downloaded to playback the first 2% of the video.

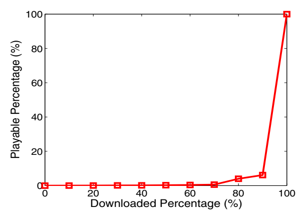
While this property of BitTorrent is an irritant even for a fixed peer, it is justifiable for two reasons: (a) this enables the peer to contribute well to the P2P network as it is likely to have blocks that are different from those at other peers; and (b) fixed peers do not have to concern themselves with wireless disconnections thus ensuring that the downloads will eventually complete and will not be in vain. However, for a mobile peer, this property can have more serious implications. In the example of the 100MB file, if the mobile peer gets disconnected from its wireless network (and remains so) after 90% of the file has been downloaded, the user still cannot playback more than 5% of the content. Furthermore, the 90% of the file size downloaded thus far using the rarest-first algorithm in the interest of the well being of the rest of the P2P network cannot be served back to the P2P network anyway because of the disconnection!

3.5 Relevance to Other P2P Networks

Thus far, we have investigated properties of the BitTorrent P2P data network that negatively impact performance of a mobile peer. While not all the properties discussed thus far are directly relevant in the context of the other three popular P2P data networks (Gnutella, FastTrack, and eDonkey), a majority of the issues discussed apply. This is especially true for the other third-generation P2P network - eDonkey, where a majority of the above issues still hold true except for the problem with rarest first fetching. For the second-generation P2P networks represented by Gnutella and Fast-



(a) 5MB file



(b) 100MB file

Figure 3. Impact of rarest first fetching

[1] We assume here that mobile IP is not used to handle mobility due to its slow deployment.

Track, a subset of the issues apply including the impact of mobility on incentives and the impact of uploads on downloads (for other applications).

4 Design and Solution

In this section, we outline some key design aspects of the proposed *wP2P* solution that are targeted to address the limitations of existing P2P data networks identified in Section 3. One key property of all the principles we present is that they are *mobile host only changes* that do not require any support from the fixed peers, and are fully backward compatible with already existing versions of the P2P (BitTorrent) protocols. Also, we present any specific implementation details with respect to the BitTorrent protocol. However, all the solutions presented are *purely local to the mobile host* and backward compatible to all existing BitTorrent P2P client applications on fixed peers.

4.1 Age-based Manipulation

The bi-directional TCP problem arises because of specific quirks of the TCP design and how they relate to the wireless environment. However, bi-directional TCP's performance otherwise is desirable since it eliminates ACK overheads under normal conditions. In other words, the solution to the problems with bi-directional TCP is not to switch back to dual unidirectional TCP connections as that would render the overall performance worse than when using bi-directional TCP as pure ACKs in both directions consume precious bandwidth resources.

In this context, the *Age-based Manipulation* (AM) design principle of *wP2P* involves the adaptive manipulation of the bi-directional TCP connections for better performance. Essentially, recalling the discussion on ACK loss rates in Section 3, an argument can be made that TCP's throughput performance is vulnerable to ACK losses *only when the congestion window is small.* At larger congestion windows, the higher ACK loss rates do impact progress, but not significantly. Hence, under age-based manipulation, explicit conversion of piggybacked ACKs to pure ACKs is performed *when the connection congestion window (cwnd) is small*[2] and piggybacked ACKs are let through as-is when the congestion window is larger than a threshold[3].

The AM component constantly monitors the congestion window of the TCP connection and if the current connection congestion window is less than a specified threshold value γ (set to 6 in our evaluations as suggested in [10]), the connection status is set to *YOUNG*. Otherwise the connection status is set to *MATURE*. The AM component also maintains state about the TCP connection and captures TCP packets transmitted by the mobile host. If the connection status is *YOUNG*, the AM module conveys any new ACK information piggybacked on DATA packets transmitted by the mobile host as separate pure ACKs. This achieves better robustness for the ACKs given a finite error rate in the wireless channel. The captures and manipulates TCP packets in *wP2P* can be achieved using the WinpkFilter [4] framework that acts transparently to the existing protocol stack of the network.

4.2 Incentive-Aware Operations

The problem of failure of incentives stems from the two distinct conditions of the self contention in a wireless link and mobility related identity loss. The *Incentive-Aware operations* (IA) principle in *wP2P* addresses both problems. Essentially, one technique under incentive aware operations in *wP2P* involves the adaptation of the upload rate in order to find the smallest upload rate possible to achieve the maximum download rate. While this value for the upload rate is trivial to determine in a wired setting, a more sophisticated algorithm is required in a wireless environment.

Since a wireless host uses a shared channel, the upload and download traffic contend with each other. In order to strike the optimal balance between the two competing issues (incentives and self-contention) *wP2P* performs a Linear Increase History-based Decrease (LIHD) algorithm that adapts the uploading rate to an optimal value. The intuition behind the LIHD algorithm is that while increasing the upload rate, it is better to be conservative so that the mobile host does not upload more than necessary. At the same time while reducing the uploads it is desirable to be aggressive. LIHD hence increases upload rates linearly when there is a positive correlation between the uploads and downloads, while decreasing the upload rates with increasing aggressiveness when decreasing the uploads does not cause a decrease in the downloads.

The Incentive-Aware (IA) component monitors upload and download rates achieved by the P2P application. It uses window-averaged throughput to determine the upload rate control of the P2P application. It controls the upload rate in a way as to optimize the downloads achieved by the mobile host. In order to achieve optimal performance we need to operate at the peak of Figure 2(c). If the download rate in the current time window is greater than the download rate achieved in the preceding time window, then the IA component increments the upload-rate counter. On the other hand, if the download rate in the current time window is less than the previously recorded download rate, then the upload rate counter is decremented by a value proportional to the number of consecutive cutdowns of the upload rate. This procedure achieves the optimal trade-off between incentive driven P2P downloads and wireless contention of the uplink and downlink transmissions.

Another technique *wP2P* uses that falls under this design

[2]Note that although this manipulation is done at the receiver, standard techniques exist to track the sender congestion window at the receiver.

[3]A straightforward value for the threshold is 6 as congestion windows less than 6 are highly vulnerable to losses in either direction [10].

principle is identity retention across hand-offs and within the same swarm. The rationale for generating uniquely different peer-ids in BitTorrent is to be able to identify and distinguish between clients with the same IP address (say, if the clients are behind a NAT), but at the same time confine the benefits of incentives accumulated by a peer to only that swarm in which the peer contributed. Since the typical scenario for task initiation in wired environments is when a peer wants to download another data file, generating a new peer-id is reasonable. However, in mobile environments task re-initiations can occur just because IP addresses have changed. *wP2P*, in this context, performs identity retention within a swarm, whereby even when task re-initiation is performed, as long as it is for a swarm the mobile peer was a member of before, the old peer-id is retained. This enables the mobile peer to leverage its previously accumulated incentives. Thus, IA component stores the peer ID of the mobile host when the application is started and when there is IP layer handoff, the IA component restores the stored peer ID to maintain incentives.

4.3 Mobility-Aware Fetching

In Section 3 we observed how in a mobile peer (as a client) mobility can impact the performance of downloads in BitTorrent. *wP2P* uses a *Mobility-Aware Fetching* (MA) principle to deal with the problems associated with mobility. This principle explicitly controls how data is fetched. The mechanism is that of *exponentially increasing altruism* or *exponentially decreasing selfishness*. Essentially, a mobile peer fetches blocks in sequence with a probability p_s (=$1 - p_r$), and fetches the rarest-first block with a probability p_r. This probability P_r is a function of the network stability of the mobile host as measured by the amount of time elapsed since the last network disconnection of the mobile host (or the start of the download). During the initial phases of the download, the mobile peer uses a small value (say, 20%) for p_r, and exponentially increases p_r as it downloads increasing fractions of the total file.

The rationale for this design is as follows: during the initial stages of downloads, if the mobile host gets disconnected, there is no benefit due to the rarest-fetch mechanism *either for the mobile host (in terms of playability) or to the P2P network (in terms of availability)*. Hence, it is more desirable to fetch sequentially. However, as the mobile host stays connected for a longer period of time, *its utility to the P2P network* has more stability and hence it is more meaningful to have available rare blocks. Furthermore, if the mobile host now gets disconnected, the user still has a considerable portion of the data in-sequence for playback. Thus, This mobility-aware adaptive content fetching achieves a more desired tradeoff between *sequential content availability for disconnected usage of content* and *usability of content for other peers to download.*

4.4 Integrated Operations

For a particular connection, all the three components of the *wP2P* framework work in tandem to achieve optimal performance. We can classify the operation of the components with respect to the different periods of the P2P connection, as illustrated in Figure 4. After the connection is setup, Age-based Manipulation component kicks in during early stages of the connection. It also works during congestion recovery periods and after reconnection in case of mobility. The operations of the Incentive-Aware component are performed during steady-state of the TCP connections. Finally, the Mobility-Aware operations component is active during the steady-state period of the TCP connection and also after IP address change due to reconnection.

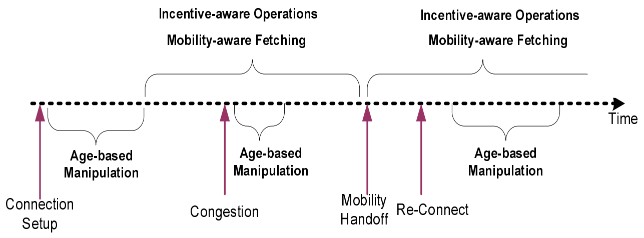


Figure 4. Integrated operations

5 Evaluation

5.1 Implementation

We use a prototype implementation of *wP2P* that is built as an enhancement to the CTorrent client version 1.2 [2], which is a lightweight C++ implementation of BitTorrent protocol with about 10K lines of code. All the three components of *wP2P* are implemented by either modifying the source code of CTorrent or adding a separate module which works with a packet filtering utility widely available in Linux distributions[3].

• *Network Setup:* We use two wireless clients on a popular BitTorrent network to compare the performances of *wP2P* with a default version of BitTorrent. One client runs the modified CTorrent version and the other runs plain vanilla version of the CTorrent (we call this as the default client). The clients are connected to the Internet through ns-2 based wireless emulators. An illustration of the network setup is shown in Figure 5. We use ns-2 emulation to study

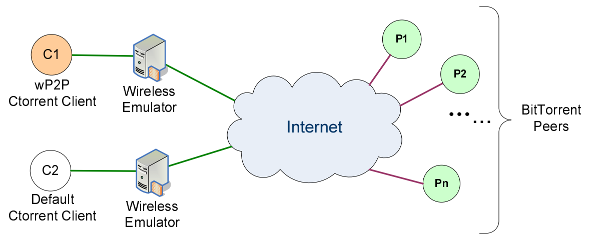


Figure 5. Testbed used in evaluation

the impact of various issues of wireless environments. We emulate random wireless losses using random bit errors. We emulate mobility by changing the IP addresses of the clients using the "ifup/ifdown" commands in Linux. We also monitor the bandwidth consumed at each client to enforce bandwidth limitations.

• *Age-based Manipulation:* The operations of this component require the determination of the connection's age. The determination is based on the measurement of current congestion window. Since the information of congestion window typically is not available to the application itself, the realization of this component has to obtain such information from certain networking entities. Specifically, we choose Netfilter utility to assist the implementation of the component partly due to its wide deployment in Linux distributions. A module in the user space keeps track of the amount of data sent by the remote peer in every round trip time (rtt), and uses the current value as an estimate of that peer's TCP congestion window for the next rtt. We chose a congestion widow of size 9k bytes (approximately 6 full packets) as an indicator of the age of the flow.

• *Incentive-Aware Operations:* This component employs two techniques: identity retention and Linear Increase History based Decrease (LIHD) rate control. For the first technique a static peer ID is used in lieu of a randomly generated peer ID every time the IP address changes. For the second technique we modify CTorrent's in-built capability to control upload and download limits. The default CTorrent client allows uses to specify the upload and downloads limits. We modify it to allow the adaptive LIHD rate control algorithm described in the previous Section. We use bandwidth monitors to check the current upload and download rates for the algorithm.

•*Mobility-Aware Fetching:* The basic CTorrent client does not implement the rarest first fetching algorithm commonly used by BitTorrent clients. Hence we first implement the rarest first fetching algorithm for the default client. We then modify this algorithm to include sequence information and awareness of download progress. Specifically, at the start of the download higher probability is given for low sequence numbered pieces as opposed to high sequence numbered ones. As the download progresses, the default rarest first search algorithm gains prominence.

5.2 Evaluation of *wP2P*

We now look at the effectiveness of the specific mechanisms employed by *wP2P* to address the limitations of default BitTorrent clients in wireless environments.

5.2.1 Age-based Manipulation

wP2P addresses the problems of Bi-directional TCP in wireless environments using the AM component. We study the impact of this component under varying random loss conditions emulated by varying the BERs ranging from $1e-6$ to $1.5e-5$. Initially we setup a single seed for a 100MB file and use the two CTorrent clients as leeches. We allow each of the leech to initially download about 50% from the seed (and the other leech). We then remove the seed so that any further data transfer is just between the two leech peers. We compare the download rates observed by the two leech peers for five runs and show the averaged results in Figure 6(a). We observe that *wP2P* outperforms the default CTorrent under all bit error rates because decoupling ACKs result in smaller ACK losses for the connection, and in turn, larger throughput. Specifically, with all the four BER values, *wP2P* achieves about 20% more throughput.

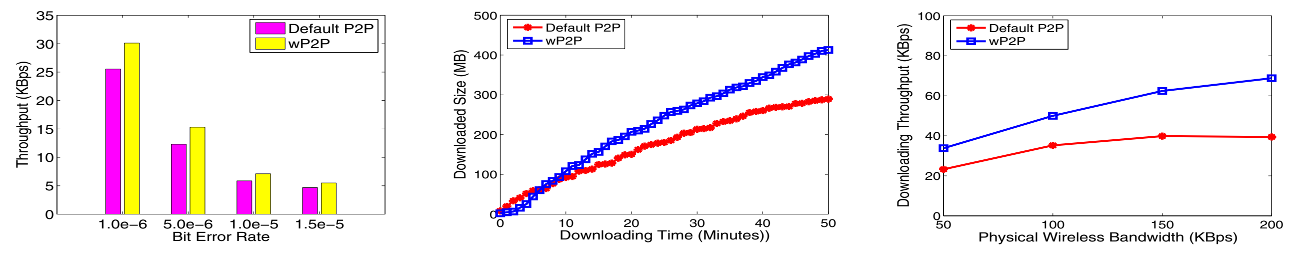
5.2.2 Incentive-Aware Operations

As discussed in the previous Section identity retention and LIHD rate control address the issue of failure or loss of incentives. To evaluate identity retention we use the two CTorrent clients to simultaneously download a Fedora-7-KDE-Live-i686.iso (http://torrent.fedoraproject.org/) image, a 688MB file shared among more than two hundreds peers when our experiments were conducted. The IP addresses of the two clients are changed every one minute to emulate mobility. Figure 6(b) shows the total downloaded size of a typical run for these two peers. The downloaded size is plotted as a function of time. For the default client whenever incentives are lost the download rate is reset to the initial nominal value unlike in *wP2P* where the incentives are maintained. Hence we find a higher downloads for *wP2P* compared to the default for the same time. After 50 minutes of download we observe that *wP2P* downloaded about 100MB more than the default.

To evaluate LIHD we vary the bandwidth of the wireless emulator from 50KBps to 200KBps. Figure 6(c) shows the averaged results over 10 runs for the case when $\alpha = \beta = 10KBps$. We observe that, initially as the available bandwidth increases both *wP2P* and the default client show increased download throughput, but beyond a certain point the default client starts losing achieved throughput. With a bottleneck bandwidth of 200KBps we observe that *wP2P* outperforms the default by as much as 70%.

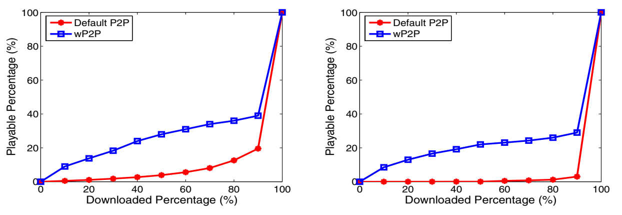
5.2.3 Mobility-Aware Fetching

Figures 7 show the results of the Mobility-Aware Fetching technique for different file sizes of the content being downloaded and compare them against the default rarest first fetch algorithm. The results are averaged over 20 runs. In these experiments we set the value of p_r to be equal to the downloaded percentage of file. We observe that MF can achieve significantly better performance compared to the default. For instance, for a 5MB file, when 50% of the data has been downloaded, MF can result in about 30% of playable content while the default rarest first technique can achieve only about 5%. The improvements are even more prominent when the file size increases.



(a) Age-based manipulation (b) Maintaining peer ID (c) LIHD

Figure 6. Age-based manipulation (a), Incentive-aware operations (b, c)



(a) 5 MB (b) 100 MB

Figure 7. Mobility-aware Fetching

6 Related Work

P2P Enhancements: There are works in related literature that propose to improve performance in P2P systems([15, 11, 12, 13, 14]. For example, Reputation based trust systems ([12]) focuses on incorporating better incentive schemes to encourage cooperative behavior and penalize free riders. Some work also ([15, 5, 11]) analyze the performance characteristics of the BitTorrent protocol. In comparison with these works, our work focus the unique challenges arise as mobile hosts join the p2p networks. These challenges are not seen in fixed hosts and in wired networks.

Recently more and more p2p users go mobile and are connected with wireless links. The authors in [9] analyze the traffic pattern of a well established 802.11 WLAN network and show that P2P traffic including P2P data sharing and streaming has increased dramatically. The authors in [7] propose a cross-layer optimization of Gnutella for deployment in purely mobile ad hoc networks.

Other Related Work: [18] addresses the issue of mobility in an ongoing transport connection by providing transparent network connection mobility using reliable sockets (rocks) and reliable packets (racks). [8] designs a mobility-aware file system for partially connected operation. Specifically, it allows applications to maintain consistency on only the critical portions of its data files.

7 Conclusions

In this paper we investigated the issues when using mobile hosts as peers in a P2P network. We identified several insights into the issues such hosts face using a real-life BitTorrent P2P data network. We proposed a solution called *wP2P* that significantly improves performance and also evaluated the solution using a real life implementation.

References

[1] *BitTorrent*. http://www.bittorrent.org/.

[2] *Enhanced CTorrent, a lightweight C++ implementation*. http://www.rahul.net/dholmes/ctorrent/.

[3] Netfilter project. http://www.netfilter.org/.

[4] *WinpkFilter*. http://www.ntkernel.com/.

[5] A. Bharambe, C. Herley, and V. Padmanabhan. Understanding and deconstructing bittorrent performance. In *ACM SIGMETRICS*, Banff, Alberta, Canada, 2005. ACM Press.

[6] CacheLogic, http://www.cachelogic.com/home/pages/ research/p2p2005.php. *Peer-to-Peer in 2005*, 2005.

[7] M. Conti, E. Gregori, and G. Turi. A cross-layer optimization of gnutella for mobile ad hoc networks. In *ACM MobiHoc*, 2005.

[8] D. Dwyer and V. Bharghavan. A mobility-aware file system for partially connected operation. *ACM Operating Systems Review*, 31(1):24–30, Jan. 1997.

[9] T. Henderson, D. Kotz, and I. Abyzov. The changing usage of a mature campus-wide wireless network. In *ACM MobiCom*, 2004.

[10] H.-Y. Hsieh, K.-H. Kim, and R. Sivakumar. On achieving weighted service differentiation: An end-to-end perspective. In *IEEE IWQoS*, 2003.

[11] A. Legout, N. Liogkas, E. Kohler, and L. Zhang. Clustering and sharing incentives in bittorrent systems. In *ACM SIGMETRICS 2007*.

[12] Q. Lian, Z. Zhang, M. Yang, B. Y. Zhao, Y. Dai, and X. Li. An empirical study of collusion behavior in the maze p2p file-sharing system. In *IEEE ICDCS*, 2007.

[13] Y. Liu, L. Xiao, X. Liu, L. M. Ni, and X. Zhang. Location awareness in unstructured peer-to-peer systems. *IEEE Trans. Parallel Distrib. Syst.*, 16(2):163–174, 2005.

[14] Y. Liu, L. Xiao, and L. Ni. Building a scalable bipartite p2p overlay network. *IEEE Trans. Parallel Distrib. Syst.*, 18(9):1296–1306, 2007.

[15] D. Qiu and R. Srikant. Modeling and performance analysis of bit torrent-like peer-to-peer networks. In *ACM SIGCOMM 2004*.

[16] Ringtonia, http://www.ringtonia.com/. *Melodeo's mobile phone P2P to launch*.

[17] Roadcasting, http://www.roadcasting.org/. *A new type of radio*.

[18] V. C. Zandy and B. P. Miller. Reliable network connections. In *MobiCom '02*, 2002.