Research Article

A Peer-to-Peer Internet for the Developing World

Umar Saif
umar@lums.edu.pk
Associate Professor
Department of Computer Science, LUMS
Opposite Sector U, DHA
Lahore 54600
Pakistan
+92 04 205722670, Ext. 4421

Ahsan Latif Chudhary
Research Associate
Department of Computer Science, LUMS
Opposite Sector U, DHA
Lahore 54600
Pakistan

Shakeel Butt
Research Associate
Department of Computer Science, LUMS
Opposite Sector U, DHA
Lahore 54600
Pakistan

Nabeel Farooq Butt
Research Associate
Department of Computer Science, LUMS
Opposite Sector U, DHA
Lahore 54600
Pakistan

Ghulam Murtaza
Ph.D. Candidate
Department of Computer Science, LUMS
Opposite Sector U, DHA
Lahore 54600
Pakistan

Abstract

Users in the developing world are typically forced to access the Internet at a fraction of the speed achievable by a standard v.90 modem. In this article, we present an architecture to enable offline access to the Internet at the maximum possible speed achievable by a standard modem. Our proposed architecture provides a mechanism for multiplexing the scarce and expensive international Internet bandwidth over higher bandwidth P2P (peer-to-peer) dialup connections within a developing country. Our system combines a number of architectural components, such as incentive-driven P2P data transfer, intelligent connection interleaving, and content-prefetching. This article presents a detailed design, implementation, and evaluation of our dialup P2P data transfer architecture inspired by BitTorrent.

1. Introduction

The "digital divide" between the developed and developing world is underscored by a stark discrepancy in the Internet bandwidth available to end-users. For instance, while a 2 Mbps ADSL link in the United States costs around US$40/month, a 2 Mbps broadband connection in Pakistan costs close to US$400/month.

The reasons for the order of magnitude difference in end-user bandwidth in the developed and the developing world are threefold:

- **Expensive International Bandwidth:** Developing countries often have to pay the full cost of a link to a hub in a developed country, making the cost of broadband Internet connections inherently expensive for ISPs. For instance, more than 40 countries have less than 10Mbps of international Internet bandwidth, whereas in Belgium, a 9Mbps ADSL high-speed Internet package is available for just US$80 a month (Digital Divide, 2005)
- **Lack of Proper Peering Points:** ISPs in the developing world typically have to sign transit agreements with upstream ISPs in which there is typically no notion of a peering agreement between small regional ISPs. Smaller ISPs, therefore, purchase bandwidth at a cost derived from the price of an international Internet link, even when its traffic is destined for a host connected to an ISP in the same region.

  It is worth mentioning that even when an Internet Exchange Point (IXP) exists, it is typically owned by a large upstream ISP. Such an IXP is mostly a marketing term by a transit provider, offering a router configured for BGP4 packet-exchange and "sold" as a transit agreement to the Internet at international bandwidth rates. On the other hand, when no such IXP exists, traffic generated by a user in the developing world has to actually traverse international links even if the recipient is connected to an ISP in the same region.

- **Poor Provisioning for "Pre-paid" Users:** Like cell phone users, Internet users in the developing world predominantly use "pre-paid" scratch cards for accessing Internet over dialup. While the pre-paid model reduces the barrier to entry for users, ISPs
  - cannot easily anticipate and provision their network for their user base.
  - As a result, end-user bandwidth is both expensive (due to conservative ISP provisioning) and unpredictable.

Characterized like this, the scarcity of available bandwidth in the developing world is not a "last-mile" problem. Our experience has shown that a 56Kbps modem can typically achieve an average throughput of greater than 40Kbps, more than twice the bandwidth afforded over a typical (10–15Kbps) dialup Internet connection in Pakistan. Our experiments, conducted with several different ISPs in the second-largest metropolis in Pakistan, repeatedly highlighted the fact that end-user bandwidth lower than 15kb/sec. is typically not due to poor phone lines. Rather, it is because of a choked upstream Internet connection of the ISP.

We conclude that given the high cost of international Internet circuits, local ISP idiosyncrasies, and the economics and politics of routing on the Internet, end-users in the developing world are forced to access the Internet at a fraction of the speed possible over a dialup connection.

Unfortunately, the paucity of Internet bandwidth severely limits the utility of the Internet in the developing world. In general, the Internet is almost never used for accessing or transferring data larger than a few hundred kilobytes. This includes software downloads and online software updates, large email attachments, and sharing and download of large files. On the other hand, such bulk data transfer comprises 70% of overall Internet traffic (Tolia, 2006).

To illustrate the limited role of the Internet in the developing world, consider a simple scenario where two users in Pakistan wish to exchange a 3.5MB PDF file as an email attachment. On a characteristically slow dialup connection in Pakistan of, say 16kb/sec., it would take close to an hour to transfer the file (30 minutes each for uploading and downloading the email attachment). Accounting for possible disconnections over such long dialup connections, the effective time for file transfer may be prohibitively long for most people to even consider the Internet as a medium for exchanging such large content.

Ironically, without the Internet, if the users established a direct peer-to-peer dialup connection between the two computers, they can exchange the file in less than 18 minutes (over a v.90 peer-to-peer dialup connection at 32kb/sec.)—a performance improvement of more than a factor of 3—at a bandwidth equivalent to a typical broadband connection in Pakistan (~35kb/sec.).

However, the contemporary view of a dialup is limited to a "last-mile" connection between the end-host and the Internet Service Provider (ISP) to access the Internet. In this article, we propose a departure from this view of dialup to address the bandwidth limitations in the developing world. The key idea of our proposal is simple: *Instead of using dialup solely as a mechanism to connect to the Internet, at a fraction of bandwidth afforded by the modem, we propose to use point-to-point dialup connections, at modem-speed, to bypass the Internet when transferring large content between end-hosts.* This is achieved by using a network routing layer, dubbed dialup-underlay, which monitors end-host bandwidth and interleaves the (low-bandwidth) ISP dialup connection with modem-speed peer-to-peer dialup connections to download large content on the Internet.

The point-to-point nature of our architecture is reminiscent of routing systems from the pre-Internet days, such as FidoNet, USENET (historically dubbed "poor man's ARPANET"), and UUCP. However, our motivation, design goals, and implementation strategy is very different from these systems. Unlike pre-Internet systems that relied solely on dialup connections for moving content between computers, our goal is to accelerate access to large content on the Internet by utilizing a dialup connection at the maximum bandwidth supported by the modem. In our model, content still "resides" on the Internet, but it may be downloaded using a P2P dialup connection to reduce download time. Figure 1 illustrates the interleaving architecture of our system. Figure 2 illustrates the performance benefit of the proposed interleaving architecture by comparing the download performance of an end-host for a 30-minute

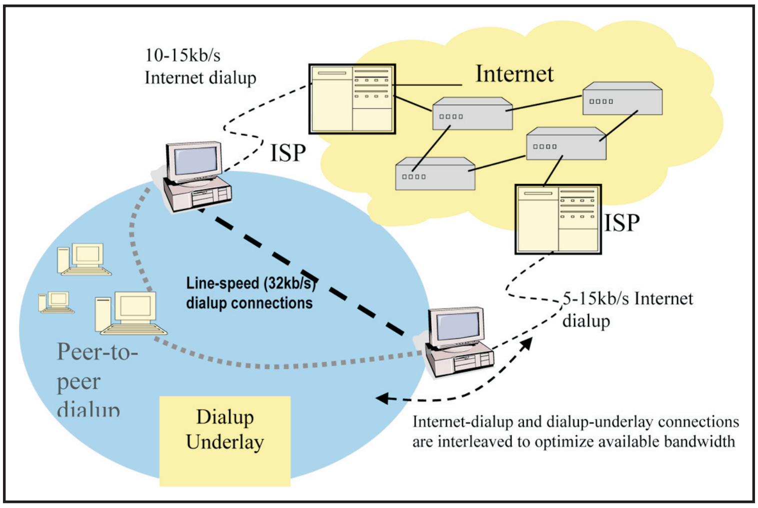

Figure 1. Our system interleaves low-bandwidth Internet connections with modem speed P2P dialup connections.

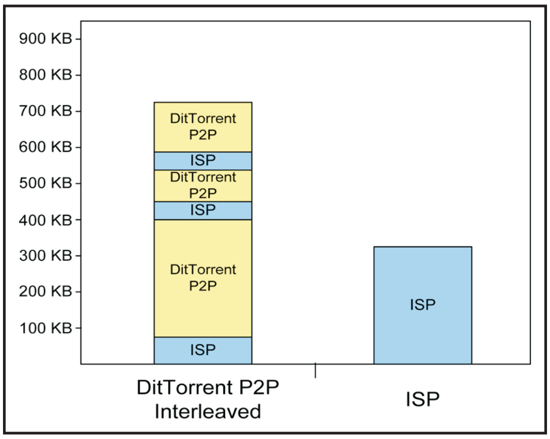

Figure 2. The difference in the data downloaded with and without our interleaving architecture.

session with and without our interleaving architecture.

Perhaps radical, and closer to pre-Internet technologies (such as FidoNet) than current-day broadband networking technologies, we believe that such an approach has practical appeal for the "other four billion" users of the Internet. Furthermore, while this article is focused on using POTS to-modem dialup connections to circumvent extremely low bandwidth Internet connections in the developing world, the crux of our proposal has generic appeal in the developing world. Countries like Pakistan have seen a phenomenal growth in Telecom, resulting in cheaper and better communication infrastructure within the country—including deployment of high-bandwidth Wireless Local Loop (WLL) and WiMAX—while Internet connectivity and bandwidth remains scarce. Our proposed architecture, at an abstract level, provides a mechanism for multiplexing the scarce and expensive international Internet bandwidth over higher bandwidth peer-to-peer connections within a developing country.

Key to the practicality of our approach is a realization that it is not always possible or desirable to make a direct dialup connection between the "client" and "server." For instance, a Web server may not be accessible over dialup; the server may not support a dial-in facility or may be located in a different country, requiring an expensive international phone call to establish a dialup connection. Even when the server is accessible over dialup, it would have a limited number of modems and a finite capacity for handling dial-in connections.

Our system derives its practicality from combining the P2P dialup-underlay with a data transfer architecture that enables dialup clients to cache and collaboratively share downloaded content—a peer-to-peer (P2P) file-sharing systems (Cohen, 2003) (albeit with additional mechanism for direct P2P dialup connections). Our P2P data transfer architecture is derived from and compatible with the hugely popular BitTorrent peer-to-peer file-sharing system (Cohen, 2003). We chose BitTorrent over other P2P file-sharing systems like Gnutella, Kazaa, eDonkey, etc., due to its download performance and robust incentive mechanism (Qiu, 2004; Pouwelse, 2004; Saroiu, 2002). We call our system DitTorrent (Dialup BitTorrent).

The DitTorrent module presents an interface similar to BitTorrent and is designed to be compatible with existing BitTorrent clients and trackers. Additionally, DitTorrent can establish direct point-to-point dialup connections to download content at modem-speed when such a connection can reduce the overall download time of a requested file.

Using DitTorrent as a mechanism for data trans-

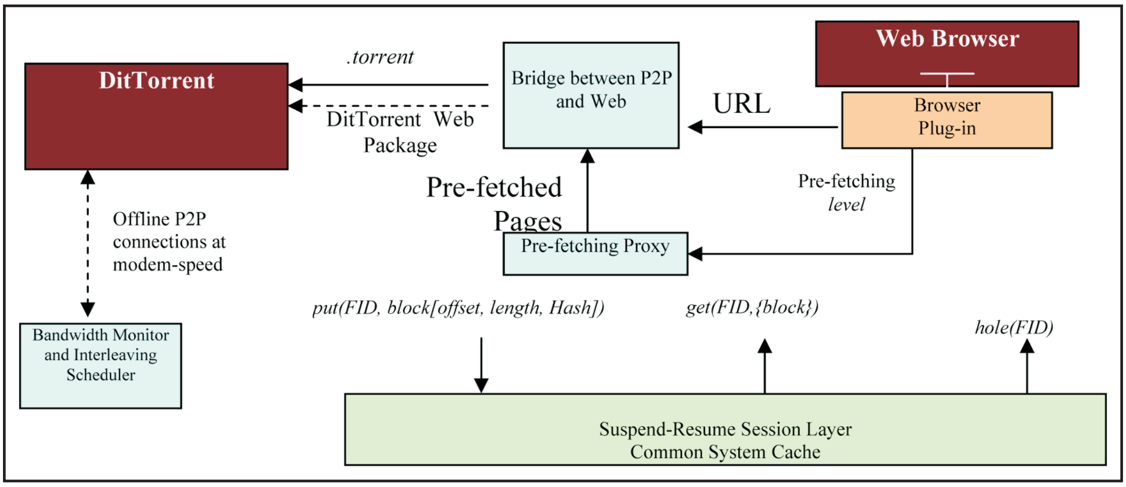
DitTorrent
.torrent
DitTorrent Web Package
Bridge between P2P and Web
URL
Web Browser
Browser Plug-in
Pre-fetched Pages
Pre-fetching level
Pre-fetching Proxy
Offline P2P connections at modem-speed
Bandwidth Monitor and Interleaving Scheduler
put(FID, block[offset, length, Hash])
get(FID,{block})
hole(FID)
Suspend-Resume Session Layer
Common System Cache

Figure 3. Modules of the software installed at a client.

fer, we implement a session layer that enables offline Web access over DitTorrent's modem-speed dialup connections. Offline Internet access in our system is enabled by several components working in concert: (1) browser plug-in that employs the P2P data transfer layer, when possible, to download requested content over DitTorrent (rather than using a TCP/IP connection over the Internet between the client and the Web server); (2) P2P and Web bridge module that converts Web pages into DitTorrent packages to enable offline download; (3) a pre-fetching proxy that mitigates the disruption of online access when interleaving ISP and P2P dialup connections; and (4) suspend-resume download-manager provides a common caching and scheduling mechanism to maintain download sessions across interleaving of ISP (Internet) and peer-to-peer dialup connections. Figure 3 sketches the high-level architecture of our system.

In this article, we describe the design and implementation of our system and present detailed analysis of data transfer over the P2P dialup-underlay. While derived from BitTorrent, the P2P dialup mode of DitTorrent faces a number of unique challenges. Above all, while BitTorrent is designed around the concept of parallel downloads of multiple file chunks from different peers, DitTorrent is "sequential"; a DitTorrent client can connect with only one peer over a P2P dialup connection. Likewise, calling and connecting other peers introduces the additional overhead of 10–30 sec. for modem-to-modem handshake.

The rest of the article describes our architecture and solutions to these challenges and is organized as follows: Section 2 gives an overview of the system architecture and describes the offline access model of our system; Section 3 describes the design and evaluation of our extension to BitTorrent, called DitTorrent, which provides the mechanism for offline data exchange; Section 4 outlines our implementation; and Section 5 presents related and future work.

2. System Overview

Our system is based on a modular architecture diagrammed in Figure 3. The browser plug-in drives the entire system: a user clicks on a URL in his browser; the URL is shipped to the bridge module, which passes this to the DitTorrent module; the DitTorrent module looks up the requested file on the P2P network and may start downloading the file in the offline mode. If the file does not exist on the P2P network, the P2P network responds with a failure code, causing the browser to download the file over the Internet.

Files that are downloaded over the Internet are turned into DitTorrent packages by the bridge module, such that subsequent download requests by other hosts may be served offline by the DitTorrent P2P network. Note that the DitTorrent package typically includes more files than just the one requested by the user; files that are linked by the current page are downloaded by the pre-fetching proxy and are included in the DitTorrent package.

The suspend/resume module provides a common session layer, which permits blocks of a file to be downloaded and checked in parallel by different data transfer mechanisms (offline P2P and Internet in our case).

Below we describe these components in detail.

2.1 Data-oriented Transfer

The internal interfacing between the Web browser and the P2P client embodies a data-oriented architecture; a user chooses a file he wishes to download, while the system automatically explores alternative mechanisms to download the file in the minimum possible time (Tolia, 2006).

To achieve this, the first critical piece is the bridge module, which bridges Web browsing and P2P offline file-download. In order to download a Web page on the P2P network, the bridge module must first translate the requested file to a .torrent file which may be looked up on the DitTorrent network. To achieve this, the browser plug-in passes the URL of the requested file to the bridge module, which looks-up the requested URL on an indexing server[1] to download the corresponding .torrent file. The bridge module then passes the file to the DitTorrent module which contacts the tracker and, subsequently, its swarm to download the file in offline mode at modem speed.

However, the role of the bridge module is more involved when it cannot find the requested URL on the DitTorrent network. In this case, the bridge module waits for the user to download the file over the Internet, and assembles a DitTorrent package to be put on the P2P network for subsequent (offline) downloads by other users.

Before describing the operation of packaging Web pages as DitTorrent packages, it is important to realize that making a single Web page accessible over DitTorrent is typically not feasible. This is because the bandwidth improvement resulting from a transition from a slow-speed Internet connection to a 32kb/sec. P2P modem-to-modem connection is not free. In fact, each such transition incurs an overhead of close to 30 seconds for negotiating the new modem connection. Of course, each subsequent transition between two P2P dialup connections, as well as the transition back to the Internet connection, also incurs the same overhead of modem-handshake. A typical Web page on the Internet, on the other hand, is less than 40KB in size, making it infeasible to incur the overhead of modem-to-modem handshake. To illustrate the point, Figure 4 depicts suitable ISP-P2P interleaving points with varying Internet bandwidth and the number of dialup connections necessary to download a file. These measurements, derived from taking an average of 100 readings of different user traces, illustrate that, for smaller files, the overhead of modem-to-modem negotiation outweighs the time saved in downloading the file at modem speed. With a moderately low-bandwidth dialup Internet connection of 15kb/sec., and between two to four average modem connections (disconnection and reconnection to the Internet, as well as dialup connections between peers), interleaving of ISP-P2P typically becomes feasible for files larger than 50KB.

Another important consideration for off-line Web access is an understanding of user browsing patterns. Typically, users do not access a single Web page on a Web site.

Recent studies (Hu, 2007) show that, on average, a user clicks through 10–30 Web pages on a Web site. Therefore, in order to afford a smooth offline browsing experience on the P2P network, DitTorrent packages must include several pages on a Web site. Given the distribution of crawled pages by users on a Web site (Hu, 2007), the challenge is to balance a smooth browsing experience for a user who may click-through several pages, while minimizing the cost (of downloading a large package) for users who browse fewer pages. Figure 5 shows the percentage of users who get redundant Web pages with respect to the pages included in a single DitTorrent package.[2] The results are derived by taking an average of user traces according to the distribution reported in Hu (2007).

With this background, we assemble a DitTorrent package (for offline browsing) as follows. If the bridge module cannot find a page on the DitTorrent P2P network, it waits for the user to finish her browsing session. If the number of pages browsed by the user (in a single session) exceeds a preconfigured threshold, it simply puts all these pages

1. In DitTorrent, we use the requested URL for indexing and look-up of the file on a Torrent indexing server.

2. User click-through distribution is derived from a recent study by Hu (2007).

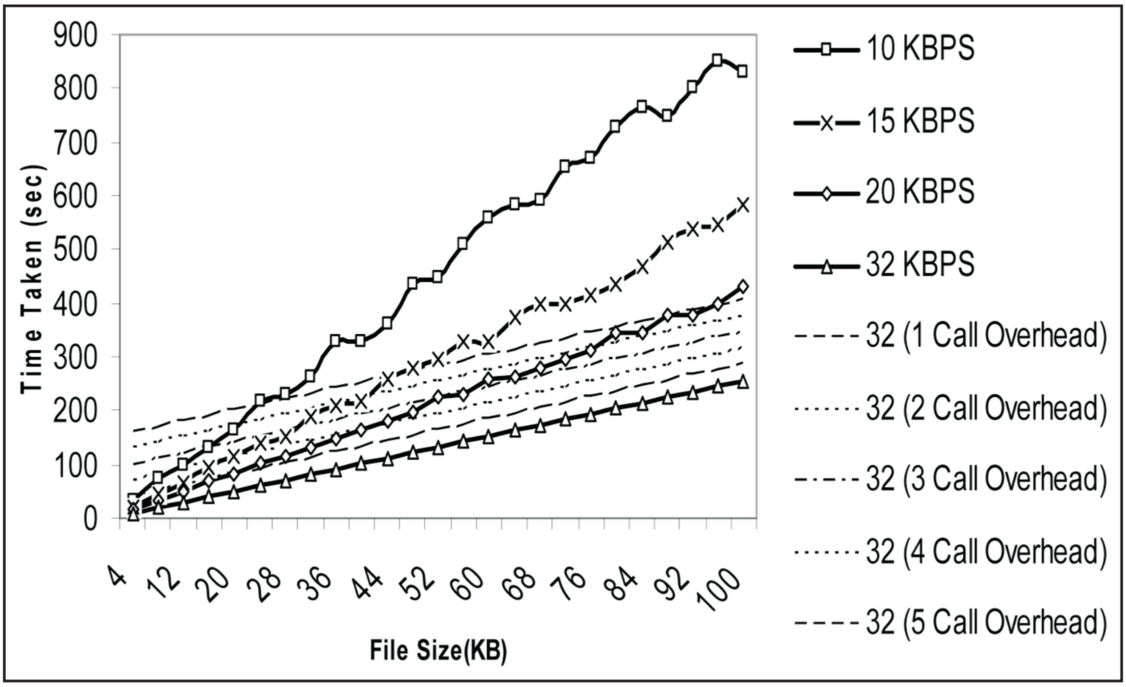

Figure 4. Feasible interleaving points between P2P and Internet.

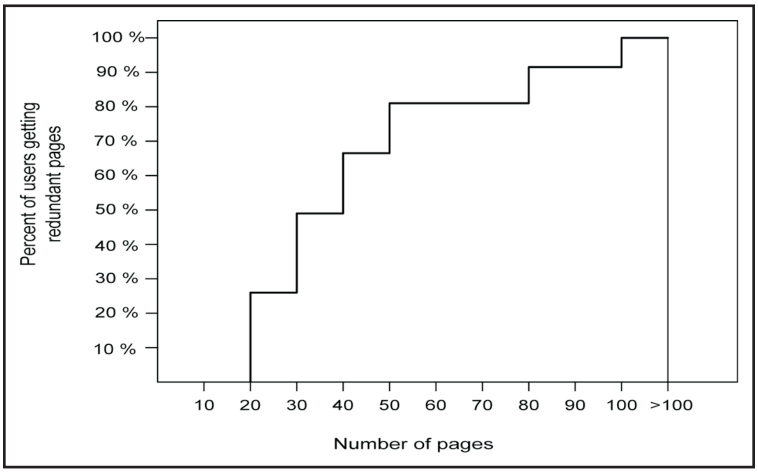

Figure 5. Percentage of users that download "extra" pages with respect to the package size.

in a single DitTorrent package and publishes the package for others to download. However, if the user visits fewer pages, the bridge module requests the pre-fetching proxy to download an additional *k* level of egress links (*k* is configurable), such that the package is also useful for a user who exhibits "average" browsing patterns.

We have currently set the package size to be 40 pages, which corresponds to the 50 percentile mark in Hu (2007). With 40 pages, 50% of the users will have a smooth browsing experience by downloading just a single DitTorrent package.

Web pages in the cache are automatically removed according to the source caching policy (time-to-live). If the pre-allocated cache fills up over time, files with the oldest time stamps are removed first.

2.2 Interleaving of Online-Offline Connection

Web browsing mandates that P2P-ISP connections are interleaved without manifestly disrupting "normal browsing" of the Web. When a user requests to download a file, say, http://ocw.mit.edu, over the Internet (via an ISP), the browser plug-in asks the bridge module to find the file on DitTorrent. If found, it then requests DitTorrent to download the requested file. DitTorrent

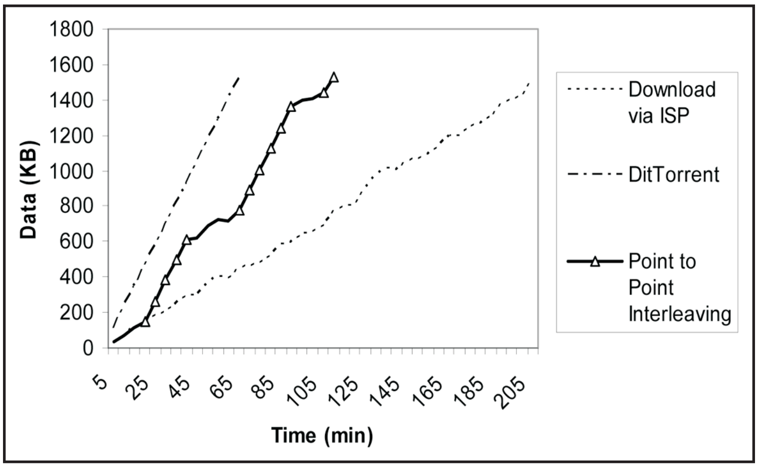

Figure 6. Interleaving of an Internet connection with a point-to-point connection for fetching pages not present in a DitTorrent package.

establishes P2P dialup connection with a peer that has the requested URL package to download the file in offline mode at maximum modem-speed.

However, subsequently, a user may click her way to a page that is not part of a DitTorrent package downloaded by DitTorrent over the P2P network. Therefore, the browser plug-in keeps track of user clicks in the DitTorrent package and switches over to ISP-mode if the user subsequently clicks her way to a level beyond which nothing is locally present. The browser may be configured to switch back to the ISP-mode even before that stage, say, at k-1 levels, to make the browsing experience smoother.

Figure 6 shows a timeline of P2P and Web access interleaving collected from a user trace for the above scenario. A switchover between the slow Internet connection and modem-speed DitTorrent is manifested as the change in the gradient of the line. The dotted lines represent download times without ISP-P2P interleaving.

2.3 Suspend-resume Session Layer

In order to support downloads from both the Internet (via ISP) and DitTorrent's offline mode, our system includes a common suspend-resume session layer. The common suspend-resume session layer permits different download mechanisms, such as DitTorrent and Web browser, to create a file handle (encoded as a unique File ID, FID) and put in different blocks of a file that are pieced together to generate the complete file. The FID is a (MD5) hash of the file contents, permitting both unique identification of file contents for subsequent lookups and verification of the completeness and integrity of the downloaded contents. Different download mechanisms, such as P2P DitTorrent and client-server HTTP connections, invoke the *hole* interface of the session layer to *find* chunks of a file currently not downloaded. The hole method, when invoked with an FID, returns a pair of offsets that indicate a range of bytes in the file, or a *hole,* currently not stored in the cache. Successive invocations of the hole interface return non-overlapping file chunks not stored in the cache; the browser plug-in and DitTorrent calling the hole interface are assigned non-overlapping chunks of a file to download either in parallel (online mode) or sequentially (when interleaving online and offline modes). In our current implementation, we have fixed the hole size to be the same as a BitTorrent block (128kB), simplifying the interoperation of the session layer and our P2P DitTorrent client.

2.4 DitTorrent Peer-to-Peer Client

The centerpiece of our architecture is the DitTorrent P2P client (Saif, 2007). As mentioned earlier, our architecture derives its practicality by enabling clients (peers) to share downloaded data, minimizing the need for direct dialup connections between clients and servers. Our P2P data transfer architecture is derived from and compatible with the hugely popular BitTorrent peer-to-peer file-sharing system (Cohen, 2003). The key attraction of BitTorrent for us is its practical incentive-driven data-sharing model; instead of assuming a volunteer-driven model like FidoNet, where users are expected to voluntarily call one another to copy data between various nodes, our system is based on a more practical incentive-driven, tit-for-tat data-sharing model of BitTorrent (Cohen, 2003). In our model, files are divided into smaller chunks, as in BitTorrent, that are virally replicated in the network, based on opportunistic peer connections; node A may let node B download a chunk of a file as long as node B can offer another chunk in return that node A wishes to download.

However, this model also raises a number of interesting challenges. For one, download of a file in this model does not involve a single switchover from the online Internet world to a point-to-point dialup connection, but several shorter connections with different peers in the same vein as BitTorrent. Therefore, with the 30-second penalty of a modem-to-modem handshake, it is important to minimize the number of connections needed to download the entire contents of the file. Moreover, given the point-to-point nature of the operation over the dialup-underlay, clients must somehow discover other offline clients and establish peer-to-peer connections to exchange chunks of a file. The point-to-point nature of the dialup-underlay also introduces another interesting idiosyncrasy: if two peers are connected to each other, no other peer can connect to them. Hence, once a point-to-point connection is established, there is no mechanism for a client to opportunistically discover a better peer, until it hangs up and connects to another client. Furthermore, BitTorrent's rate-based, tit-for-tat data sharing model—in which a host uploads data to a peer for only as long as it can download data from that peer at a similar data rate—becomes superfluous in a point-to-point dialup connection. This is because upload and download rates in a standard P2P dialup connection are symmetric (e.g., 32kb/s upload and download for typical v.90 modems). Section 3 discusses the challenges and possible solutions in detail.

3. Offline Peer-to-Peer Data Transfer

In this section, we describe the design, implementation, and evaluation of DitTorrent.

3.1 Evaluation Methodology

To understand the behavior of DitTorrent, especially under extreme conditions in the developing world, we use a simulation-based approach. Even though we have released our DitTorrent implementation (Client and Tracker) to a community of users in Pakistan, via Sourceforge (DitTorrent, 2007), we chose a simulation-based evaluation since very little data exist in terms of traces of real torrents for extremely low-bandwidth connections. Moreover, our simulator's controlled environment made it possible to evaluate aspects of our design that are otherwise difficult to infer from tracker logs or by running moderately sized experiments.

For our evaluation, we have implemented a discrete-event simulator of DitTorrent, which extends the BitTorrent simulator implemented by Bharambe (2006). However, while the underlying framework of our simulator is derived from the simulator described in Bharambe, our model of DitTorrent is almost diametrically different from the BitTorrent model he implemented. For instance, the BitTorrent simulator by Bharambe is designed to simulate parallel downloads by a BitTorrent client, while DitTorrent is limited to point-to-point symmetric upload/download between only two hosts. Likewise, Bharambe's simulator assumes perfect knowledge of the location of each block of a file, while a DitTorrent client's knowledge about the location of file blocks is often imperfect. Similarly, a BitTorrent client is designed to maximize download bandwidth, while a DitTorrent client attempts to minimize the time wasted in negotiating new modem-to-modem connections.

To capture the idiosyncrasies of DitTorrent's (offline) operation, we implemented the following new features in the simulator described in Bharambe: point-to-point symmetric connections, call-overhead resulting from modem-to-modem negotiation, busy tones overhead during "flash crowds," offline block-discovery, and greedy peer selection with different "end-game" modes. Below, we describe the motivation, implementation, and evaluation of each of these features in detail.

In our experimental setup, we configured our simulator to use a swarm of 100 nodes (typical BitTorrent swarm size). Our simulations were run on a P4 machine 3.2GHz, with 1GB RAM. Our simulation environment was configured with the following parameters:

- In our simulation environment, all the participating nodes were configured with symmetric download and upload bandwidth, set at 30 Kbps.
- Peers in our simulation were bootstrapped with a single block, unless otherwise stated in the experiments described below.
- In our experiments, we measured the performance of the system by varying the size of the file to be downloaded, and where specifically

mentioned, the number of initial blocks allocated to each peer.

- Given the point-to-point nature of DitTorrent, each node was configured to only connect with one other at a time.
- We set the seed leaving probability to 1 in our experiments; we do not assume that a node stays in the simulation after completing its download.
- In the simulations for *offline block discovery* and *flash-crowds* (described below), nodes were introduced in an ongoing experiment, after a random delay, to simulate late-entering nodes.

3.2 Architectural Overview

DitTorrent is designed to be backward compatible with BitTorrent. Compatibility with BitTorrent has obvious appeal in terms of user adoption, making DitTorrent a vehicle for using BitTorrent in the developing world. Above all, DitTorrent derives its incentive-driven, tit-for-tat data-sharing model from BitTorrent. Furthermore, high-level architectural components of DitTorrent are derived from BitTorrent; files are published by advertising a .torrent metafile; clients make peer-to-peer connections to opportunistically download, cache, and publish blocks of files; and a tracker acts as a directory service for clients to discover peers from which blocks of a file may be downloaded. The use of a .torrent file and a tracker for initial peer discovery and bootstrap provides a basis for compatibility with BitTorrent. DitTorrent tracker, as well as the DitTorrent .torrent file format, is designed to be backward compatible with BitTorrent. As a result, existing BitTorrent clients can interoperate with DitTorrent clients. DitTorrent clients, however, of course have the additional capability to establish point-to-point dialup connections for accelerated download of content.

Before proceeding with the description of DitTorrent, it is instructive to consider the behavior of BitTorrent over characteristically low bandwidth connections in the developing world. Figure 7 plots the download time of a 10 MB file over BitTorrent by a client connected to the Internet on a slow dialup connection. The results shown in Figure 7 were reported by the original BitTorrent simulator of Bharambe et al (Bharambe, 2006), which, most notably, ignores TCP timeouts. Still, the download time of a file goes up sharply as the bandwidth is reduced from 15kb/sec. to 5kb/sec. This is because the number of chokes experienced by the client (snubs by peers as they find better partners) increases as the client bandwidth nears 10kb/sec. For comparison, in the case when there are 30% low-bandwidth hosts in the mix of nodes using BitTorrent, a client with a 10kb/sec. takes close to 8.5 times longer than a cable node with a bandwidth of 100kb/sec. for upload and 250 kb/sec. for download. For these low-bandwidth nodes, point-to-point symmetric dialup connections, at 32kb/sec., can offer a substantial performance improvement. For instance, compared to a 10kb/sec. BitTorrent client that downloads a 10MB file in 2 hours and 21 minutes (derived from the experiments shown in Figure 7), a client using a point-to-point 32kb/sec. connection can download the same file in 41.6 minutes (assuming a single point-to-point connection) for a performance improvement of close to 70%.

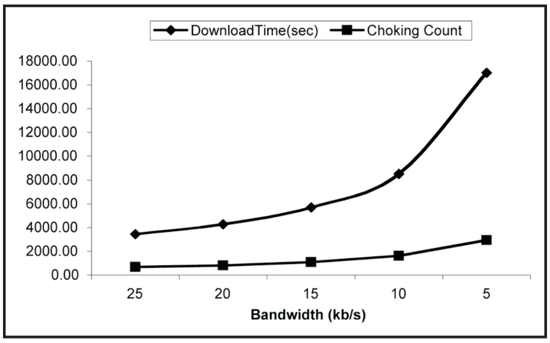

Figure 7. BitTorrent performance deteriorates sharply as the client bandwidth drops to 10kb/sec.

3.3 DitTorrent Tracker-Client Interaction

DitTorrent is designed to interoperate with BitTorrent in its online mode. However, its offline point-to-point mode requires special support from the tracker. Importantly, unlike a traditional BitTorrent tracker that keeps track of currently online hosts, a DitTorrent tracker must keep track of both offline and online hosts interested in downloading a file. While fundamental to the duality of operation of a DitTorrent client, this extension requires only a minor modification to existing BitTorrent trackers. Current BitTorrent trackers require a client to refresh its registration periodically by sending announce messages after an interval number of seconds. A DitTorrent tracker, instead of deleting the record of a client that fails to refresh its registration after interval seconds, simply marks the client offline and retains its entry for future lookups from DitTorrent clients. However, since dialup nodes are prone to frequent disconnections, our DitTorrent tracker marks a client offline only after the client misses successive periodic announcements.

A DitTorrent tracker distinguishes between BitTorrent and DitTorrent clients such that the latter can be given additional information for offline operation. To this end, a DitTorrent tracker accepts an additional event attribute from a DitTorrent client during registration; the event param in the URL is set to "dialup" to indicate that the registering client is a DitTorrent client. A client-announce with event param set to dialup may include two additional parameters to the GET request URL, as described below.

The key additional parameter in a DitTorrent client registration request is a phone number to reach the client in the offline P2P mode. Additionally, a DitTorrent client can also include a list of time windows that specify times of day during which dialup connections may be established with the client. A client can specify the following time windows as part of its request to the tracker: available_time_window specifies the time interval in which the client is available for accepting phone calls; query_time_window specifies the time interval in which the client intends to make calls to download the file; and, optionally, a previous_time_window, in case this request updates existing registration of the client. For instance, a client installed on an office computer may advertise a time-window between 8 P.M. to 7 A.M when phone lines for the office are generally free. Subsequent DitTorrent clients registering with the tracker are given contacts of those peers whose available_time_window overlaps with the query_time_window of the client (as well as online clients). The use of time-windows in our system is reminiscent of "zone mail hour" used by FidoNet (Jennings, 1984) clients to specify a time-window for receiving dialup connections.

3.4 DitTorrent Peer-to-Peer Interaction

DitTorrent's offline point-to-point operation is fundamentally different from BitTorrent. Where a BitTorrent client attempts to minimize the download time of a file by opportunistically connecting with and disconnecting (choking and unchoking) from a large set of peers (peer swarm) in search of better download bandwidth, a DitTorrent client must minimize the number of peer connections when downloading a file. This is because "trying" new peers in BitTorrent is an almost zero-overhead operation (an unchoke message sent to the peer over a long-running TCP connection), while a new dialup connection in DitTorrent incurs a 30-second overhead in negotiating the dialup connection. To illustrate the point, consider the simulation results of BitTorrent shown in Figure 7. The BitTorrent client shown in Figure 7, with a 10kb/sec. symmetric upload/download bandwidth, makes close to 564 connections (with rate-based choking period set at 10 sec. and opportunistic unchoking every 30 sec.) during the download of a 10 MB file. If these were dialup connections, the time spent in negotiating new dialup connections would equal the actual time spent in downloading the contents of the file (an overhead of 100%).

Furthermore, given a symmetric upload/download bandwidth in a P2P dialup connection, DitTorrent's offline operation does not need to include BitTorrent's rate-based choking. Instead, a DitTorrent client (in offline mode) should only choke a peer when the peer can no longer upload newer blocks needed by the client; peer connections last for as long as peers can reciprocate each other with non-overlapping blocks of file, while there is no need for opportunistic unchokes. This approach of peer-choking, similar to pairwise block-level tit-for-tat (BLTFT) proposed in Bharambe (2006)—as opposed to rate-based, tit-for-tat implemented by BitTorrent—is a natural fit in a point-to-point dialup setup and is implemented by DitTorrent.

However, while disabling opportunistic unchokes and using BLTFT avoids unnecessary connections in DitTorrent's point-to-point mode, the overhead of negotiating dialup connections must be carefully managed, especially for smaller files. This is because dialup connections, worth 30 seconds each, incur an overhead equivalent to exchange a whole block (128KB) on BitTorrent at 32kb/sec.

3.4.1 Peer-to-Peer Dial-up Overhead

In order to understand the overhead of modem-to-modem negotiation in DitTorrent, consider the theoretical best and worst cases for downloading a file of size N blocks, with a client starting with a single block of the file.

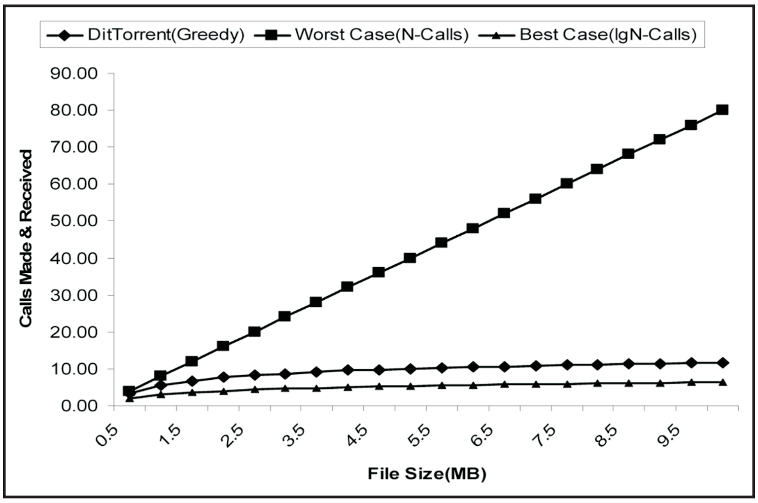

Figure 8. Comparison of the modem-to-modem negotiation overhead of the greedy peer selection with the best and worst cases.

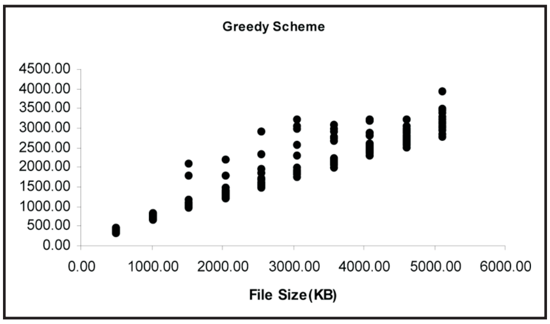

Figure 9. Variability in file download times due to the last block problem in the simple greedy strategy.

The theoretical best case, aimed at minimizing the number of calls required to download a file, may be understood as peer selection policy that exactly matches peer needs: a client calls only that peer which has exactly the same number of *complementary* blocks. In this case, with BLTFT, the first dialup connection of a new client with a single initial block will result in the exchange of one block, making it two blocks at the client. The next call will result in two blocks exchanged, making it four blocks at the client. Likewise, the next call will result in eight blocks at the client and so on. Therefore, for a file of size N blocks, it will take at least $\log_2 N$ calls to download all blocks of the file. Unfortunately, this theoretical best case assumes a perfect match of peer needs on every call made by the client; any mismatch would result in additional future calls for either the caller or the callee.

The theoretical worst case, on the other hand, may arise in a scenario in which a client must make N calls to download a file of N blocks, incurring a modem-to-modem negotiation overhead for each block of the file.

To come close to the theoretical best case, our system uses a simple *greedy strategy* for peer selection. In the greedy strategy, a client grabs the maximum it can at any point in time, regardless of whether it is an exact or a suboptimal match on either side of perfect-match. Figure 8 compares the performance of this simple greedy-strategy with the best (Log_2N calls) and worst case (N Calls). The clients in the simulation are bootstrapped with an initial set of one to five random blocks for a swarm size of 100.

However, while a simple greedy strategy performs adequately well on average, we quickly realized the impact of the "last-block-problem" in P2P systems (Bharambe, 2006). Figure 9 plots the download times from 20 different simulations as we varied the file size. The variability in the measured times for a given file size reflects the time for which different nodes in the swarm may be "stuck" trying to download the last few blocks of a file. BitTorrent employs two techniques to help nodes that are near completion to finish quickly: (1) End-game mode, which enables a client close to finishing to quickly search for the last few missing blocks; and (2) Local Rarest First (LRF), which helps balance the rarity of different blocks by requiring clients to download the rarest block first from a connected peer.

In DitTorent, we experimented with analogues of both of these schemes. In the first implementation, we mimicked the effect of the end-game mode by modifying the greedy policy, dubbed *greedy-completor.* With this modification, a greedy client favors those peers that will finish the file download at the end of the connection. Intuitively, this scheme is aimed at enhancing the chances of relative newcomers to expedite the completion of peers closer to finishing. Conversely, peers relatively early in the

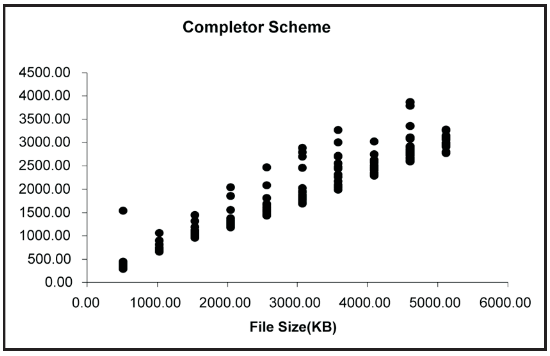

Figure 10. Variability in the file download times due to the last block problem in the greedy-completor strategy.

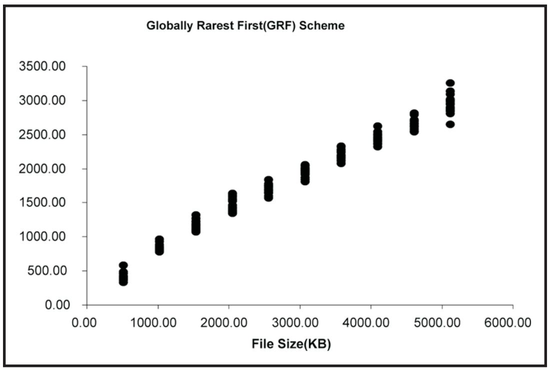

Figure 11. Variability in file download times due to the last block problem with Global-Rarest-First peer prioritization.

race get blocks from those near completion, causing rare blocks (typically stored at seeds or near-seeds) to be transferred from the old to the new. Figure 10, like Figure 9, plots download times across 20 simulations, illustrating the reduction in the variability of measured times for *greedy-completor.*

Our second strategy to combat the last-block problem is inspired by BitTorrent's LRF strategy. However, LRF's rationale is that a peer should grab the rarest block first from a peer in case it is choked prematurely. This has little impact in DitTorrent since a P2P connection in DitTorrent lasts as long as peers have something to exchange; there is no danger of unanticipated choking due to a dip in the client's upload bandwidth. Instead, we implement Global Rarest First (GRF) strategy in DitTorrent. Unlike LRF, in which a client grabs the rarest blocks stored at a peer, a client using GRF *chooses* a peer that has the rarest blocks stored at it. Viewed differently, LRF is a block prioritization strategy, while GRF is a peer selection policy; a client using GRF prioritizes its connections with peers according to the rarity of blocks stored at them. For instance, in our greedy strategy, a client that could exchange three blocks each with two of its peers will first call that peer which can offer comparatively rarer blocks.

Figure 11 shows that our greedy peer selection policy with GRF performs better than *greedy-completor* in terms of the variability in download times across 20 simulations.

3.4.2 Flash Crowds in DitTorrent

As mentioned earlier, the point-to-point nature of DitTorrent's offline operation precludes multiple peer connections. In fact, a client calling a peer that is already connected to another peer would get a "busy-tone." Unfortunately, however, this operation of checking the availability of a peer incurs an extra overhead of close to 10 seconds (time to call the number and receive a busy tone). This overhead becomes particularly significant during the so-called flash crowds (Bharambe, 2006) in which many clients want to quickly download a newly accessible file. In a point-to-point setup, flash crowds result in a large fraction of the call attempts failing with busy-tones.

Intuitively, this may be addressed by introducing a wait-time between calls during times of congestion. Our implementation combats flash crowds by using two simple heuristics: (1) Each client must wait for n seconds before trying again if it finds all of its potential peers (peers with non-overlapping chunks) busy; and (2) A client must wait for n seconds between successive calls. The former introduces a back-off period in the times of congestion, while the latter is aimed at giving clients a chance to receive calls in between making calls.

In our simulation experiments, we found that the back-off time was more useful than the introduction of wait time between calls. For instance, Figure 12 shows the file download time with respect to the back-off time and a fixed time to wait-between-calls (WBC). The performance of the system (file download time) improves significantly by introducing a back-off time, but quickly tapers off at around eight seconds for a setup in which the average call time is three minutes. It is worth highlighting, though, that while we achieved best performance with WBC set

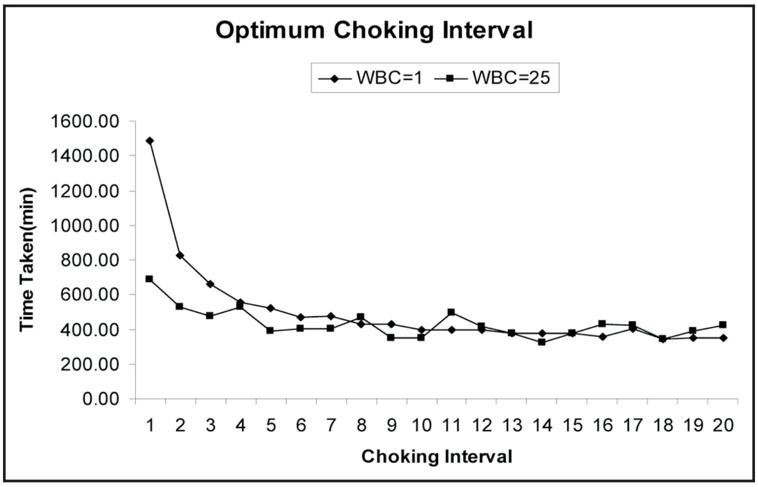

Figure 12. Analysis of overhead due to "busy tones" in flash crowds.

to a nominal one second, a WBC of zero (no wait between calls) makes the performance exponentially worse. A WBC of, say, 25 seconds, however, makes the performance more variable, without resulting in any real performance advantage. We are currently exploring an adaptive policy for WBC such that the value of WBC is dynamically adjusted according to the congestion in the network. For instance, the value of WBC may be increased multiplicatively by a constant factor upon a failed call, and reduced correspondingly if a call succeeds.

3.4.3 Offline Block Discovery

In BitTorrent, a client starts the download of a file by acquiring (from the tracker) a list of peers that have chunks of the file. The client then connects with the set of 80–100 peers returned by the tracker (its peer swarm) and acquires from each a list of blocks currently stored at the peer (called "peer handshake" in BitTorrent). Subsequently, peers in a swarm keep each other informed about newly acquired blocks by sending "have" messages. All this mechanism of discovering file blocks rests on a BitTorrent client's ability to talk to multiple peers in parallel. Hence, this cannot be directly mapped to our P2P dialup architecture.

We implement block discovery in DitTorrent's offline mode by using a scheme inspired by distributed gossip protocols (Ganesh, 2003). In this scheme, when a client calls a peer, it not only exchanges complementary file blocks, the peers also exchange lists of blocks discovered at hosts that previously connected to the clients (including their own). The aim is to virally spread the knowledge about the blocks stored at various hosts, while minimizing the number of connections required to spread the information. For instance, consider a DitTorrent client A that has previously connected with (either as a caller or callee) nodes B and C, each with the following sets of blocks: B{0,5,3}, C{1,2,9}. With our gossip-based approach, when a newly arrived client D (with zero current blocks) calls A, it is not only given a single block of file (c.f. DitTorrent bootstrap mode), it is also given lists of blocks stored at B and C (including the blocks exchanged in their connection with A). Once understood like this, it quickly becomes apparent that a client in this scheme will greatly benefit by initially calling nodes that have been around for a long time (i.e., nearing completion).

Given this background, block discovery in DitTorrent's offline mode works as follows. Newly arrived clients "scrape" the tracker to find out the percentage of file downloaded by each client in its swarm (as reported by their last update message to the tracker). Armed with this information, the client goes into offline mode. In the offline mode, it first calls the host that has downloaded the maximum number of blocks of the file. Of course, this could be a seed node, but a DitTorrent client only calls non-source seeds since they have acquired a diverse knowledge while working their way up from a single block to the completion of the file.

In our implementation, a DitTorrent client initially calls a fixed number of peers, three in our current setup, that have downloaded the maximum fraction of the file. After this initial bootstrap, a client using offline block discovery proceeds in this mode until it has acquired information about all N blocks of the file. Subsequently, the client simply follows the greedy scheme when choosing peers to exchange blocks. Figure 13 compares the performance of DitTorrent's gossip-style, offline block discovery with a setup in which a client has perfect knowledge of its peers as in BitTorrent. The performance of

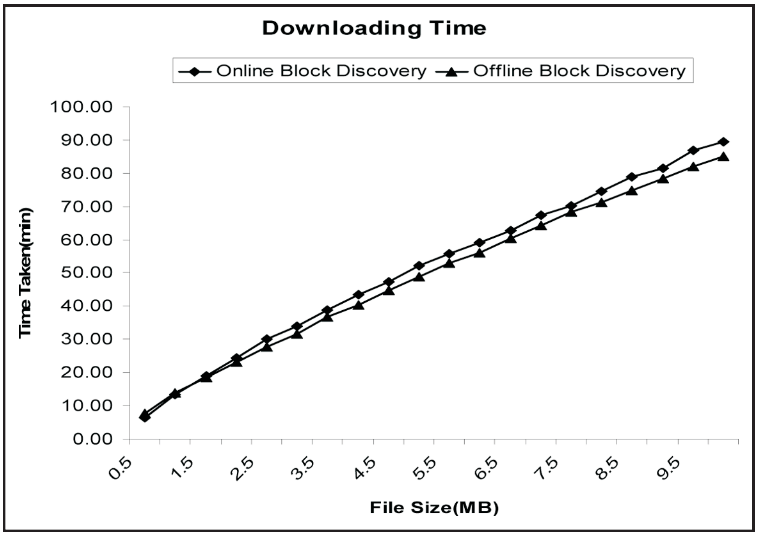

Figure 13. Comparison of DitTorrent's offline block-discovery with BitTorrent "perfect knowledge" about block locations.

DitTorrent closely mirrors BitTorrent with its gossip-style, offline block discovery, with the overhead of block discovery becoming more visible as we increase the number of file blocks (that must be discovered).

3.4.4 Budget-based Download

Often in our work, we were asked about the economic feasibility of our approach. This concern was raised because the Internet is typically a flat-rate service, while phone calls are charged by the minute. First, it is worth noting that POTS is increasingly becoming like the Internet in terms of service charges; it is typical for telephone service providers to offer flat-rate regional or national plans. In case of such a flat-rate subscription, our approach offers "near-broadband" speed for no additional cost as long as calls are localized in the flat-rate region. DitTorrent clients can be configured to only call those peers that are within their calling "region" by, for instance, matching the ISP calling code with available peers. Importantly, given the P2P nature of our data transfer scheme, the burden of making a phone call is shared between peers; a peer downloads data both as a caller and a callee.

Initially, we considered providing an interface to the users to limit the number of calls to be made by the client when downloading the file. The intuition was that after the client has made the specified number of calls, it goes into a passive mode in which it simply waits to be called to acquire more file blocks. If a client fails to download the file in a specified time, it prompts the user to increase the call budget. In some sense, this scheme exposes a tradeoff of calling cost vs. timely download to the user. It is worth highlighting that, with the current download speeds in the developing world, such an architecture is still very practical; more often than not, a user may be happy with a one-time extra cost to quickly download an important file, otherwise not possible with an extremely slow and intermittent Internet connection. However, we found that limiting the number of calls this way not only leads to starvation of peers, it results in an increase in the average number of calls each client has to make to download a file. Figure 14 shows the percentile of nodes that complete a file download as we increase the call budget. With a call-budget restriction, a 100% completion rate is achieved for a call-budget that is slightly worse than the average number of calls made in a setup without a call-budget (average number of calls without a call-budget in this simulation is nine).

4. Implementation

DitTorrent's implementation comprises the following three components: DitTorrent Daemon, Browser plug-in, and the DitTorrent Tracker.

1. The DitTorrent Daemon is written in python 2.4 and is an extension of Bram Cohen's BitTorrent version 4.4.0. The Daemon includes the additional capability to make point-to-point telephone calls using PPP connections for data transfer.
2. The Browser plug-in is a plug-in for Firefox (xpi file) that detects and passes links .torrent files to the DitTorrent Daemon. The Daemon component also provides the URLs of the Web pages for prefetching to the prefetching proxy. The prefetching proxy in our

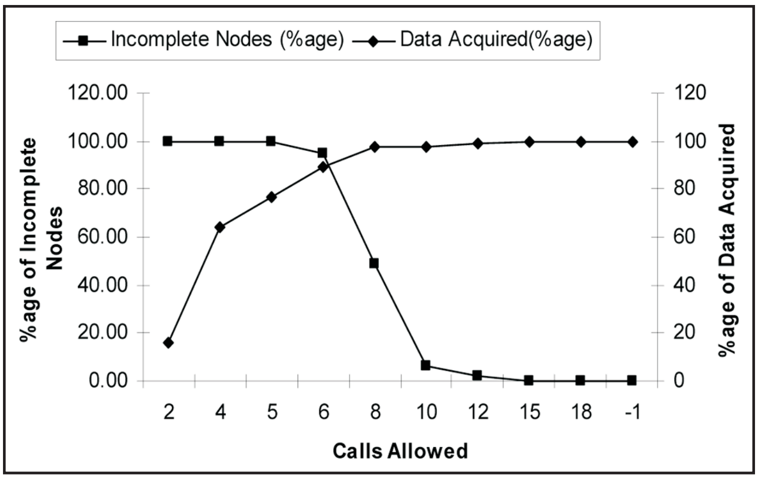

Figure 14. Analysis of budget-based download.

implementation extends the open source wwwoffle (wwwoffle, 2003) proxy, specifically designed as a dialup offline proxy server.

3. The DitTorrent Tracker is written in C++ as an extension to the open source BNBT tracker (BNBT, 2004). Our extension to the BNBT tracker includes time-window-based lookups, using efficient interval-trees, as well as compliance with additional parameters introduced to enable point-to-point dialup connections.

The source code of our implementation is available at DitTorrent (2007). The distribution at sourceforge also includes the DitTorrent simulator described in the paper.

5. Conclusion and Future Work

In the current age of broadband networks, dialup networking is mostly a forgotten technology. Perhaps the most relevant technologies for our system date back to the pre-Internet days, including FidoNET (Jennings, 1984), UUCP (Kolstad, 1984) and USENET (USENET, 1980). However, our use of an incentive-driven P2P data exchange mechanism makes our architecture fundamentally different from such systems. Importantly, unlike pre-Internet systems that relied solely on dialup connections for moving content between computers, our goal is to accelerate access to large content on the Internet by utilizing a dialup connection at the maximum bandwidth supported by the modem. In our model, content still "resides" on the Internet, but may be downloaded using a P2P dialup connection to reduce download time. Our system combines a number of architectural components, such as P2P data transfer, intelligent connection interleaving, and content-prefetching to make a practical system.

Currently, our system lacks comprehensive security architecture. We are currently working on rendezvous servers much in the same vein as the recently introduced Google click-to-call service that hides the identities of the peers connected on a dialup.

While this paper is focused on using POTS modem-to-modem dialup connections to circumvent extremely low bandwidth Internet connections, we intend to explore high-bandwidth intra-country deployments of Wireless Local Loop (WLL) and WiMAX as well. Our proposed architecture, at an abstract level, provides a mechanism for multiplexing the scarce and expensive international Internet bandwidth over higher bandwidth peer-to-peer connections within a developing country.

6. Summary and Discussion

In this article, we describe and evaluate an architecture for an accelerating Internet access in the developing-world.

Users in the developing world often face a paucity of Internet bandwidth due to three reasons: (1) expensive Internet bandwidth, (2) interplay and politics of ISPs, and (3) poor network planning for "prepaid" dialup customers.

As a result, the Internet is rarely used for accessing exchanging or disseminating large data. Most people in the developing world use the Internet primarily for low-bandwidth browsing, chatting, or textual email. Applications that require high bandwidth access to the Internet are rarely used.

We propose an architecture that circumvents the Internet bottleneck in the developing world. Our architecture is derived from the observation that, in the developing world, the bottleneck in Internet bandwidth is not the "last mile." The dialup last mile and the user's dialup Internet modem can support up to 64 kb/sec. However, the bottleneck is the ISP that rate limits the user in the face of limited upstream bandwidth.

Our architecture essentially permits users in the developing world to "share" or multiplex their Internet bandwidth by sharing downloaded content over peer-to-peer dialup connections. The idea is really simple: use the Internet for downloading content only once and share the content by directly connecting with other machines in the region to bypass the ISP bottleneck. Using our peer-to-peer architecture, a user's computer can automatically discover other computers in its local region that have a (full or partial) copy of the desired Internet content and download the file at the maximum speed supported by a modem, which is typically twice as much as the speed afforded by an ISP (40–50kb/sec.). The P2P nature of the system is derived from the hugely popular BitTorrent system, enabling a file to be broken up in pieces such that the overall operation is distributed and computers can download a file piece by piece from different peers. The P2P underpinnings of our architecture are based on a data exchange scheme that is incentive driven, and hence, sustainable, in which a callee permits a caller to download a chunk of a file only if the caller can furnish a chunk that is of interest to the callee.

However, the practicality of this peer-to-peer nature of our scheme depends on a sizeable number of peers (20–100) wishing to download a file at the same time. While this assumption is largely valid for most P2P file exchange systems, the performance of our system may be adversely affected if the user-base in a region has very diverse interests.

The system is designed to be compatible with BitTorrent, and the interleaving of the Internet and P2P connections is designed to be imperceptible to a Web surfer. The aim of the overall architecture is to make large data transfer faster, over our P2P architecture, without causing visible disruption to a user surfing the Web.

Our proposed system represents one design point in the spectrum of architectures for accessing the Internet via non-traditional channels. In particular, it enables users in the developing world to leverage intra-country high-bandwidth communication channels to speed up access over the high-cost international Internet links.

To achieve this, our system takes content from the Internet and makes it accessible over a P2P (offline) network. The offline behavior of the system is both reminiscent of pre-Internet networks such as FidoNet and modern-delay tolerant networks (Jain, 2004). However, where delay-tolerant-networks (DTN) are focused on protocols and architectures for transport of data between computers connected by "sneaker networks" (Jain, 2004), the focus of our work is to provide access to data on the Web, not between disconnected computers. We also rely on real-time dialup connections between computers, with the aim to reduce the time needed to download data from the Web. In comparison, DTN architectures focus on reliable transfer of data over slow, unreliable channels, with performance being a tertiary issue.

Our architecture may be used as the underlying "plumbing" for data-intensive applications, such as digital library systems, and learning management systems such as Sakai and Moodle. In our system, the remote reference material may be downloaded only once, while other users behind slow dialup connections can share the content over modem-speed connections without having to use a rate-limited ISP connection. ■

Acknowledgments

The work presented in this paper was, in part, funded by the Microsoft Research (MSR) Digital Inclusion Grant, 2006.

References

Bharambe, A. R., Herley, C., & Padmanabhan, V. N. (2006). Analyzing and Improving a BitTorrent Network's Performance Mechanisms. In *IEEE Conference on Computer Communications* (INFOCOM). Barcelona, Spain.

Bharambe, A., Herley, C., & Padmanabhan, V. N. Microsoft Research Simulator for the BitTorrent Protocol. Retrieved from http://www.research.microsoft.com/projects/btsim

BNBT. Retrieved from http://bnbt.depthstrike.com/

Cohen, B. (2003). Incentives Build Robustness in BitTorrent. In *Workshop on Economics of Peer-to-Peer Systems.* Berkeley, CA.

The Digital Divide at a Glance. *World Summit on the Information Society,* Tunis 2005.

Ganesh, A. J., Kermarrec, A. M., & Massoulie, L. (2003). Peer-to-peer membership management for gossip-based protocols. In *IEEE Transactions on Computers.* 52(2): 139–149.

Hu, J., Zeng, H., Li, H., Niu, C., & Chen, Z. (2007). Demographic prediction based on user's browsing behavior. *16th International World Wide Web Conference,* 151–160. New York.

Jain, S., Fall, K., & Patra, R. (2004). Routing in a Delay Tolerant Network. In *Proceedings of ACM SIGCOMM.* Portland, OR, USA.

Jennings, T. FidoNet. Retrieved from http://en.wikipedia.org/wiki/FidoNet

Kolstad, R., & Summers-Horton, K. (1984). UUCP. Retrieved from http://en.wikipedia.org/wiki/UUCP/

Pouwelse, J. A., Garbacki, P., Epema, D. H. J., & Sips, H. J. (2004, April). A Measurement Study of the BitTorrent Peer-to-Peer File-Sharing System. *Technical Report PDS-2004-003.* Delft University of Technology, The Netherlands.

Qiu, D., & Srikant, R. (2004, September). Modeling and Performance Analysis of BitTorrent-like Peer-to-Peer Networks. In *Proceedings of ACM SIGCOMM.* Portland, OR, USA.

Saif, U., Chudhary, A. L., Butt, S., & Butt, N. F. (2007a, October). DitTorrent. Retrieved from http://dittorrent.sourceforge.net/

Saif, U., Chudhary, A. L., Butt, S., & Butt, N. F. (2007b, October). Poor Man's Broadband: Peer-to-Peer Dialup Networking. In *ACM SIGCOMM CCR,* vol. 37, no. 5.

Saroiu, S., Gummadi, P. K., & Gribble, S. D. (2002). A Measurement Study of Peer-to-Peer File Sharing Systems. In *Multimedia Computing and Networking (MMCN).* San Jose, CA, USA.

Tolia, N., Kaminsky, M., Andersen, D. G., & Patil, S. (2006, May). An Architecture for Internet Data Transfer. *Proc. 3rd Symposium on Networked Systems Design and Implementation (NSDI).* San Jose, CA.

Usenet. Retrieved from http://en.wikipedia.org/wiki/Usenet/

WWWOFFLE. (2006). Retrieved from http://www.gedanken.demon.co.uk/wwwoffle/