Multimedia Systems and Applications

Multimedia Distribution - BitTorrent

James Wang

http://www.bittorrent.org/protocol.html/

http://en.wikipedia.org/wiki/BitTorrent

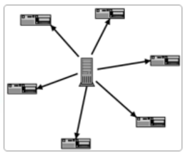
What is BitTorrent?

- BitTorrent is a free speech tool.
- You have something terrific to publish -- a large music or video file, software, a game or anything else that many people would like to have. But the more popular your file becomes, the more you are punished by soaring bandwidth costs.
- The key to scaleable and robust distribution is cooperation.
  - With BitTorrent, those who get your file tap into their upload capacity to give the file to others at the same time.
  - Those that provide the most to others get the best treatment in return. ("Give and ye shall receive!")
- Cooperative distribution can grow almost without limit, because each new participant brings not only demand, but also supply.

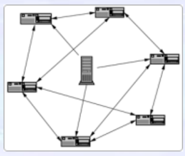
2

BiTTorrent Architecture

- The Problem with Publishing:
  More customers require more bandwidth



- The BitTorrent Solution:
  Users cooperate in the distribution



3

HTTP vs. BT

- Though both ultimately transfer files over a network, a BitTorrent download differs from a classic full-file HTTP request in several fundamental ways:
  - BitTorrent makes many small P2P requests over different TCP sockets, while web-browsers typically make a single HTTP GET request over a single TCP socket.
  - BitTorrent downloads in a random or "rarest-first" approach that ensures high availability, while HTTP downloads in a contiguous manner.

4

BitTorrent Protocol

- BitTorrent is a protocol for distributing files.
- It identifies content by URL and is designed to integrate seamlessly with the web.
- Its advantage over plain HTTP is that when multiple downloads of the same file happen concurrently, the downloaders upload to each other, making it possible for the file source to support very large numbers of downloaders with only a modest increase in its load.

5

BitTorrent File Distribution

- An ordinary web server
- A static 'metainfo' file
- A BitTorrent tracker
- An 'original' downloader
- The end user web browsers
- The end user downloaders

6

Data Partition and Management

- The peer distributing the file breaks it down into a number of identically-sized pieces, typically between 64 KB and 1 MB each.
- Pieces over 512 KB are used to reduce the size of torrent files for very large payloads, but also reduce the efficiency of the protocol.
- The peer creates a checksum for each piece, using a hashing algorithm, and records it in the torrent file.
- When a peer receives the piece, it compares the recorded checksum to the actual checksum of the received piece to test that it is error-free.
- Peers that provide the complete file are called seeders, and the peer providing the initial copy is called the initial seeder.

7

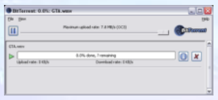
BitTorrent Hosting

- Start running a tracker (or, more likely, have one running already).
- Start running an ordinary web server, such as apache, or have one already.
- Associate the extension .torrent with mime type application/x-bittorrent on their web server (or have done so already).
- Generate a metainfo (.torrent) file using the complete file to be served and the URL of the tracker.
- Put the metainfo file on the web server and link to the metainfo (.torrent) file from some other web page.
- Start a downloader which already has the complete file (the 'origin').

8

BitTorrent Downloading

- Install BitTorrent (or have done so already).
- Surf the web and click on a link to a .torrent file.
- Select where to save the file locally, or select a partial download to resume.
- Wait for download to complete.
- Tell downloader to exit (it keeps uploading until this happens).



9

BitTorrent Connectivity

- The web site is serving up static files as normal, but kicking off the BitTorrent helper app on the clients.
- The tracker is receiving information from all downloaders and giving them random lists of peers. This is done over HTTP or HTTPS.
- Downloaders are periodically checking in with the tracker to keep it informed of their progress, and are uploading to and downloading from each other via direct connections. These connections use the BitTorrent peer protocol, which operates over TCP.
- The origin is uploading but not downloading at all, since it has the entire file. The origin is necessary to get the entire file into the network. Often for popular downloads the origin can be taken down after a while since several downloads may have completed and been left running indefinitely.

10

Bencoding (Bee Encoding)

- Metainfo file and tracker responses are both sent in a simple, efficient, and extensible format called bencoding (pronounced 'bee encoding').
- Bencoded messages are nested dictionaries and lists (as in Python), which can contain strings and integers.
- Extensibility is supported by ignoring unexpected dictionary keys, so additional optional ones can be added later.

11

How to Bencoding

- Strings are length-prefixed base ten followed by a colon and the string. For example 4:spam corresponds to 'spam'.
- Integers are represented by an 'i' followed by the number in base 10 followed by an 'e'. For example i3e corresponds to 3 and i-3e corresponds to -3. Integers have no size limitation. i-0e is invalid. All encodings with a leading zero, such as i03e , are invalid, other than i0e , which of course corresponds to 0.
- Lists are encoded as an 'l' followed by their elements (also bencoded) followed by an 'e'. For example l4:spam4:eggse corresponds to ['spam', 'eggs'].
- Dictionaries are encoded as a 'd' followed by a list of alternating keys and their corresponding values followed by an 'e'. For example, d3:cow3:moo4:spam4:eggse corresponds to {'cow': 'moo', 'spam': 'eggs'} and d4:spaml1:a1:bee corresponds to {'spam': ['a', 'b']} . Keys must be strings and appear in sorted order (sorted as raw strings, not alphanumerics).

12

Metainfo files

- **bencoded dictionaries with the following keys:**
  - **announce:** The URL of the tracker.
  - **info:** a dictionary with the following keys.
    - **name**: suggested name for the file.
    - **piece length**: number of bytes in each piece.
    - **piece:** a string whose length is a multiple of 20. It is to be subdivided into strings of length 20, each of which is the SHA1 hash of the piece at the corresponding index.

13

Metainfo files (cont.)

- **There is also a key *length* or a key *files* , but not both or neither.**
  - If length is present then the download represents a single file, otherwise it represents a set of files which go in a directory structure.
- **In the single file case, length maps to the length of the file in bytes.**
- **The multi-file case is treated as only having a single file by concatenating the files in the order they appear in the files list. The files list is the value files maps to, and is a list of dictionaries containing the following keys:**
  - ***Length:*** The length of the file, in bytes.
  - ***Path:*** A list of strings corresponding to subdirectory names, the last of which is the actual file name (a zero length list is an error case).
- **In the single file case, the name key is the name of a file, in the multiple file case, it's the name of a directory.**

14

Tracker queries

- **Tracker GET requests have the following keys:**
  - **info_hash:** The 20 byte SHA-1 hash of the bencoded form of the info value from the metainfo file.
  - **peer_id:** A string of length 20 which this downloader uses as its id. Each downloader generates its own id at random at the start of a new download.
  - **ip:** An optional parameter giving the IP (or DNS name) which this peer is at, used for the origin if it's on the same machine as the tracker.
  - **port:** The port number this peer is listening on. Common behavior is for a downloader to try to listen on port 6881 and if that port is taken try 6882, then 6883, etc. and give up after 6889.

15

Tracker queries (cont.)

- **uploaded:** The total amount uploaded so far, encoded in base ten ascii.
- **downloaded:** The total amount downloaded so far, encoded in base ten ascii.
- **left:** The number of bytes this peer still has to download, encoded in base ten ascii.
- **event:** This is an optional key which maps to started , completed , or stopped (or empty, which is the same as not being present). If not present, this is one of the announcements done at regular intervals.

16

Tracker Responses

- **Tracker responses are bencoded dictionaries.**
- **If a tracker response has a key *failure reason* , then that maps to a human readable string which explains why the query failed, and no other keys are required.**
- **Otherwise, it must have two keys:**
  - **interval:** the number of seconds the downloader should wait between regular rerequests.
  - **Peers:** a list of dictionaries corresponding to peers, each of which contains the keys peer id , ip , and port , which map to the peer's self-selected ID, IP address or DNS name as a string, and port number, respectively.

17

Peer Protocol

- **Peer connections are symmetrical. Messages sent in both directions look the same, and data can flow in either direction.**
- **The peer protocol refers to pieces of the file by index as described in the metainfo file, starting at zero.**
- **When a peer finishes downloading a piece and checks that the hash matches, it announces that it has that piece to all of its peers.**
- **Connections contain two bits of state on either end: *choked* or not, and *interested* or not.**
- **Choking is a notification that no data will be sent until *unchoking* happens.**

18

Peer Protocol (cont.)

- **Data transfer takes place whenever one side is *interested* and the other side is not *choking*. Interest state must be kept up to date at all times.**
- **Connections start out *choked* and not *interested*.**
- **When data is being transferred, downloaders should keep several piece requests queued up at once in order to get good TCP performance (this is called 'pipelining'.)**
- **On the other side, requests which can't be written out to the TCP buffer immediately should be queued up in memory rather than kept in an application-level network buffer, so they can all be thrown out when a choke happens.**

19

Piece Selection

- **Strict Priority:** Once a single sub-piece has been requested, the remaining sub-pieces from that particular piece are requested before sub-pieces from any other piece.
- **Rarest First:** When selecting which piece to start downloading next, peers generally download pieces which the fewest of their own peers have first.
- **Random First Piece:** An exception to rarest first is when downloading starts, a random piece is selected.
- **Endgame Mode:** Sometimes a piece will be requested from a peer with very slow transfer rates. This could potentially delay a download's finish. To keep that from happening, it sends requests for all sub-pieces to all peers. This strategy can only be used at the end of downloading.

20

Choking Algorithms

- **BitTorrent does no central resource allocation. Each peer is responsible for attempting to maximize its own download rate. Peers do this by downloading from whoever they can and deciding which peers to upload to via a variant of tit-for-tat.**
- **To cooperate, peers upload, and to not cooperate they 'choke' peers.**
- **Choking is a temporary refusal to upload; It stops uploading, but downloading can still happen and the connection doesn't need to be renegotiated when choking stops.**
- **A good choking algorithm should utilize all available resources, provide reasonably consistent download rates for everyone, and be somewhat resistant to peers only downloading and not uploading.**

21

Limitations and security vulnerabilities

- **BitTorrent does not offer its users anonymity. It is possible to obtain the IP addresses of all current, and possibly previous, participants in a swarm from the tracker. This may expose users with insecure systems to attacks.**
- **Another drawback is that BitTorrent file sharers, compared to users of client/server technology, often have little incentive to become seeders after they finish downloading. The result of this is that torrent swarms gradually die out, meaning a lower possibility of obtaining older torrents. Some BitTorrent websites have attempted to address this by recording each user's download and upload ratio for all or just the user to see, as well as the provision of access to older torrent files to people with better ratios.**
- **Also, users who have low upload ratios may see slower download speeds until they upload more. This prevents users from leeching, since after a while they become unable to download much faster than 1-10 kB/s on a high-speed connection. Some trackers exempt dial-up users from this policy, because they cannot upload faster than 1-5 kB/s.**
- **BitTorrent is best suited to continuously connected broadband environments, since dial-up users find it less efficient due to frequent disconnects and slow download rates.**

22