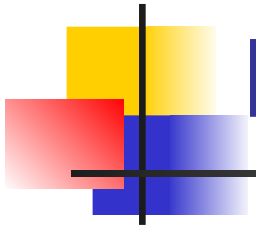
Peer Distributed Transfer Protocol

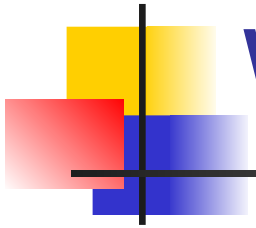


Presenter: Tony Arcieri
<tarcieri@pdtp.org>

DEFCON 12

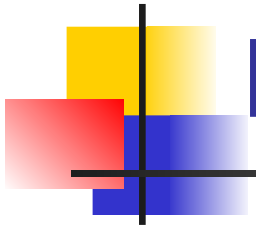
July 30th – August 1st

http://www.pdtp.org/



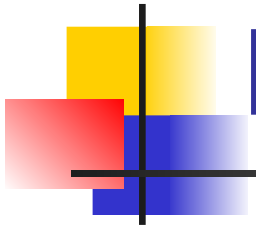
What is PDTP?

- Replacement for anonymous FTP
- A protocol designed to leverage server networks instead of individual systems
- Bringing BitTorrent-like technology to the web and FTP mirror networks



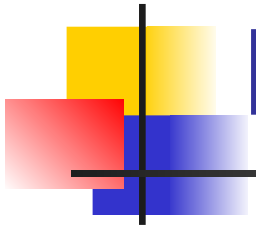
Existing Technologies

- FTP provides nice mechanisms for querying directory structures but requires manual mirror selection and individual servers to handle all traffic.
- HTTP provides a more elegant transfer protocol but makes it harder to browse directories and still requires individual servers to handle all traffic.



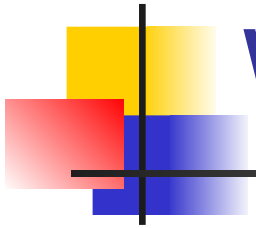
BitTorrent

- BitTorrent became popular in 2003 as a means to transfer large, high demand files, such as operating system ISOs or often copyrighted content.
- BitTorrent is designed to serve static file groupings. The data from the grouping is divided into “pieces” shared over a peer network.



Problems with BitTorrent

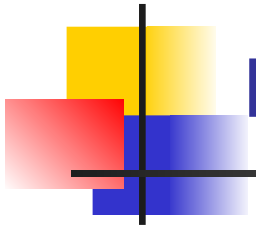
- Static file groupings are hard to integrate into the dynamic nature of web and FTP servers.
- Existing BitTorrent implementations are difficult to effectively integrate into web browsers.
- BitTorrent’s HTTP-based tracker protocol is resource intensive.



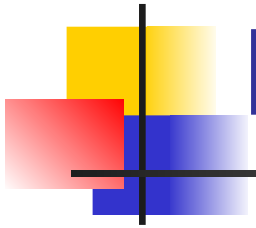
Where is BitTorrent going?

- BitTorrent 2 will feature hash trees as opposed to hash lists, making it easier to checksum parts of files.
- The new tracker protocol will be UDP based, which is much lower overhead than HTTP but requires complex packet management code.

PDTP – BitTorrent for the masses



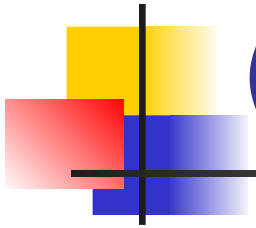
- PDTP provides content transfer through a peer network while also exporting dynamically changing directory mappings, just like HTTP/FTP.
- PDTP provides tools needed for usage within corporate or other large networks, such as a proxy server.



PDTP vs. BitTorrent

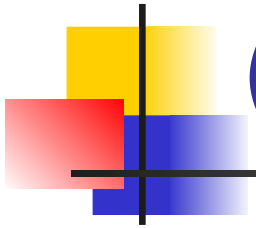
- BitTorrent currently uses HTTP as a heavyweight transaction mechanism for its protocol. BitTorrent 2 will move to UDP to mitigate scalability issues with the HTTP-based tracker protocol.
- PDTP uses a lightweight TCP-based transaction protocol.

PDTP vs. BitTorrent (cont'd)



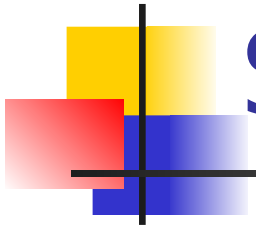
- BitTorrent networks are comprised of a "tracker" which manages the transfer and one or more "seeders" serving pieces of files to the network.
- PDTP provides additional scalability by allowing multiple servers to manage transfers or serving pieces of files to the network.

PDTP vs. BitTorrent (cont'd)

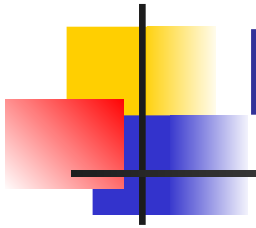


- A detailed protocol specification for BitTorrent does not yet exist, nor has BitTorrent received port assignments from IANA. Implementers of third party clients are more or less on their own.
- PDTP has applied to IANA for port assignments. The protocol is documented as an IETF Internet-Draft (i.e. RFC format), HTML, and RFC2629 XML format.

PDTP Components - Servers

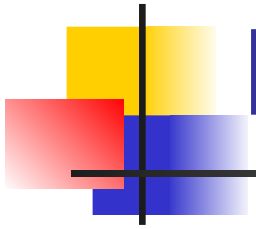


- PDTP servers function like BitTorrent trackers, but also provide caching for server directory maps and file information.
- One or more PDTP servers connect to PDTP hubs, allowing transfer management to be distributed. Clients are automatically load balanced between servers at connection, eliminating mirror selection.



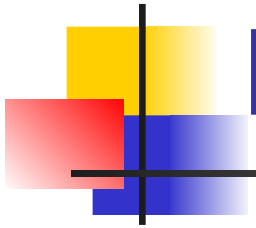
PDTP Components - Hubs

- Hubs are the heart of any PDTP network. They provide network maps, directory listings, and file service to the network.
- The file service function of a PDTP hub is similar to a BitTorrent “seed”, however piece proxies may be used for additional piece service scalability.



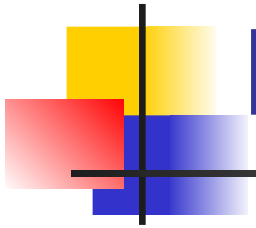
PDTP Piece Proxies

- Piece proxies download and cache file pieces from a hub, then serve them to the network on demand, reducing the bandwidth requirements of hubs.
- BitTorrent solves this problem through the use of multiple seeds. However, if no seeds are available for a torrent, content becomes inaccessable.



PDTP Proxy Servers

- PDTP proxies allow multiple users behind a firewall to accept incoming peer connections and, if necessary, make outgoing connections through the proxy rather than directly.
- PDTP proxies also allow multiple users on a network transferring the same file from the same server to share the file among each other rather than each downloading it separately, good for OS patches.

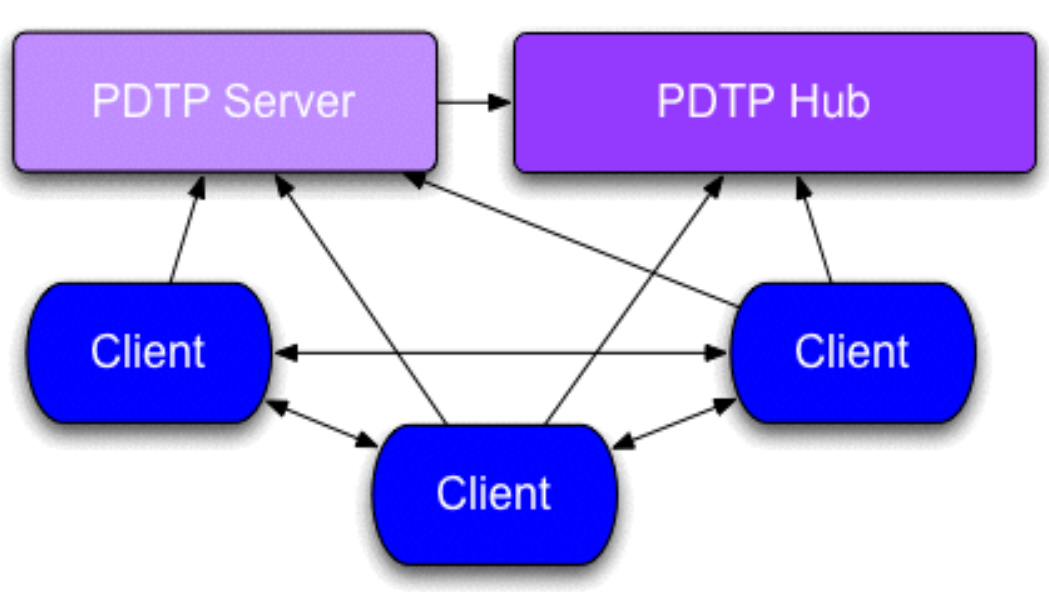


PDTP Clients

- PDTP clients can browse directory lists and request file transfers.
- Clients transfer pieces either from a hub/proxy or to/from each other in a peer-to-peer manner.
- Clients are either “passive” (behind a firewall and cannot accept incoming connections) or “active” (can make both outgoing and incoming connections)

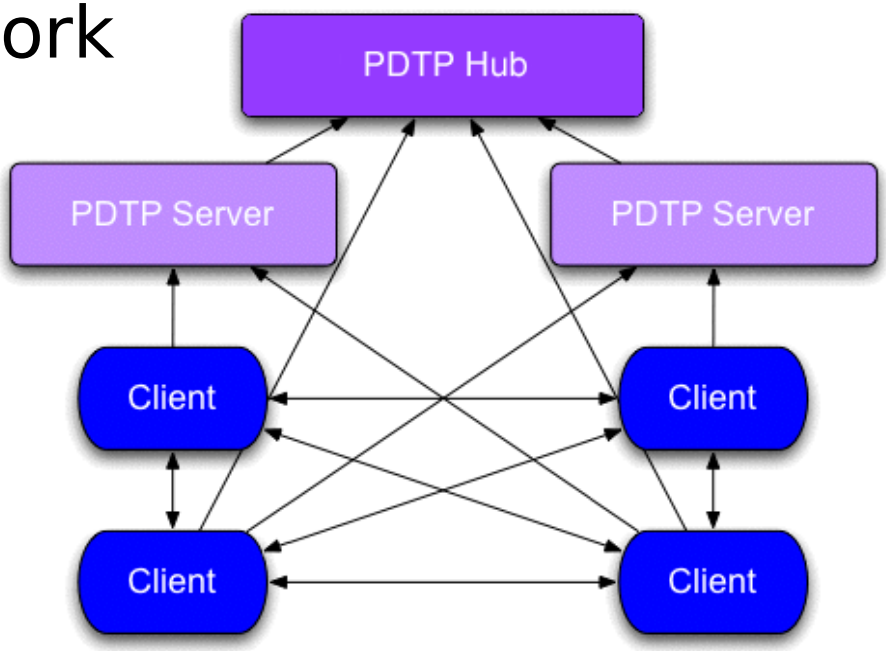
PDTP Networks

- Simplest configuration: One server, one hub
- Server manages transfers on network
- Hub “seeds” clients with initial pieces



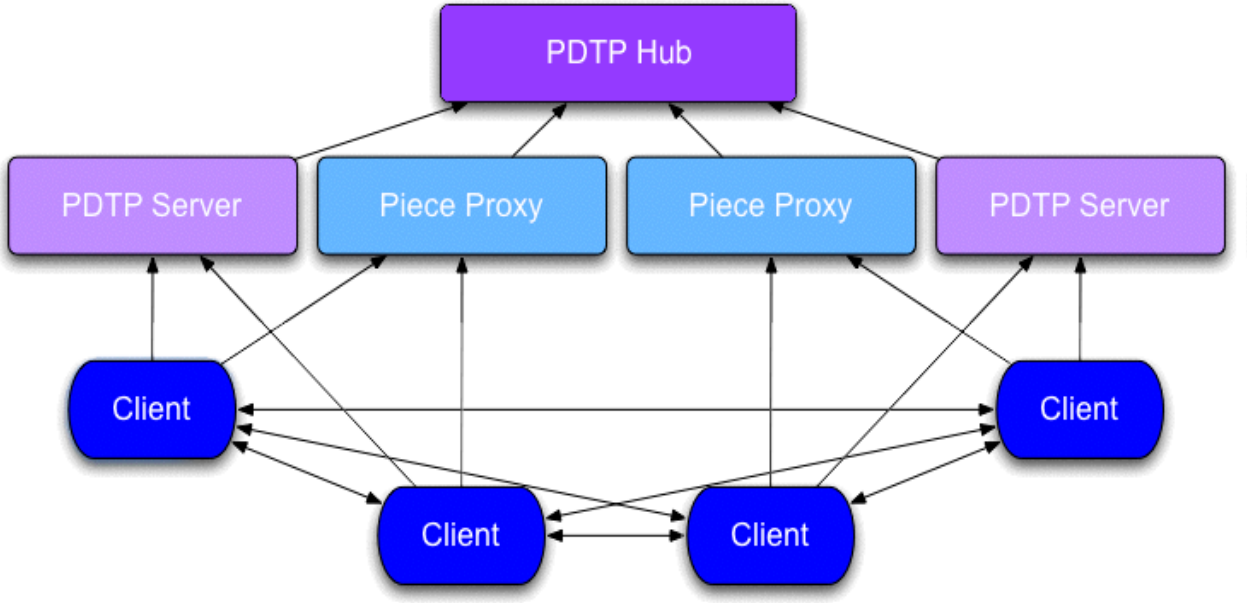
More Scalable Networks

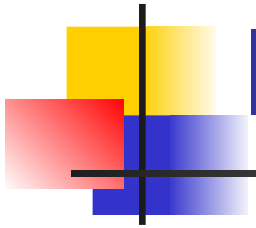
- Additional servers can be added for increased network scalability
- Hub still needs to “seed” entire client network



Maximum Scalability

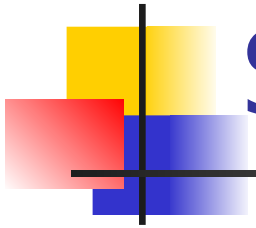
- Piece proxies can be added to decrease the hub's bandwidth consumption
- This configuration shields the address of the hub from the rest of the network





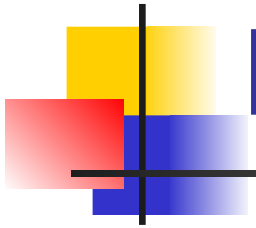
Interesting Features...

- With servers and piece proxies in place, the address of the network hub is hidden from the rest of the network.
- As long as the hub and one server remain active, the network will continue transferring files.
- A possible headache for Johnny Law?



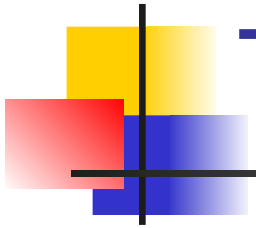
Self-Optimizing Networks

- While BitTorrent relies on some interesting properties of random networks, PDTP servers use a weighted scoring algorithm to select peers for transfers.
- PDTP calculates a weighted score for each peer every time a transfer completes, and factors in the transfer rate and respective scores of all peers which a given peer has ever transferred pieces to/from.



Protocol Design

- PDTP uses a lightweight binary transactional format.
- The protocol is fully bidirectional and supports both synchronous and asynchronous operational modes.
- PDTP does not use ASN.1 due to input validation complexity issues.

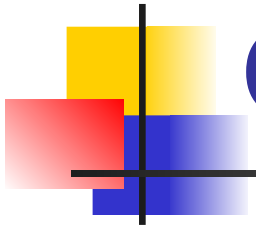


Transaction Format

PDTP transactions are structured as follows:

- `uint32 - Length`
- `uint32 - Serial number`
- `uint16 - Opcode`
- `uint16 - Object count`
- `Arbitrary number of objects`

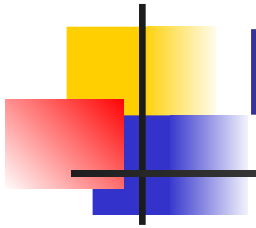
The opcode is a bitfield whose highest bit indicates a request (0) vs reply (1). The lower 15 bits comprise an integer ID for the operation.



Object Format

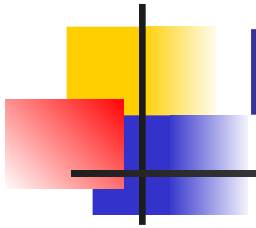
Transactions take a variable number of “objects” as arguments. These are structured as follows:

- `Uint32 – Length of object`
- `Uint16 – Integral type identifier`
- `String – Object payload`



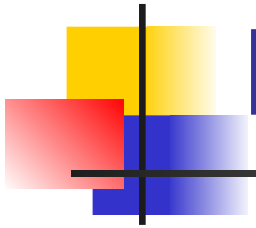
Protocol Security Concerns

- PDTP’s transaction format has been designed with all lengths explicit, centralizing much of the input validation and eliminating assumptions.
- Connection authorization is handled through existing, secure standards such as RFC2104 HMAC hashing.



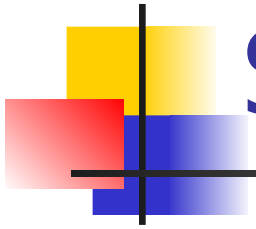
File Integrity Validation

- PDTP clients check for a signed X.509 certificate containing a DSA key.
- Clients will compute a DSS signature for a file and compare it against the server-provided one.
- A signature mismatch indicates server-side file tampering, possibly trojaning.



PDTP Search System

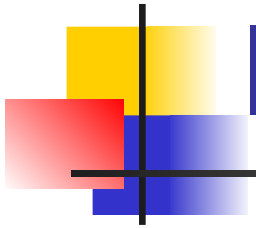
- Decentralized – Clients query servers directly for searches by sending a large number of UDP datagrams.
- PDTP Trackers store lists of active server addresses.
- http://search.pdtp.org/ will store a list of active trackers.



Server Query Process

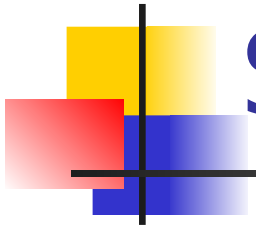
- Clients send a UDP search request to the server.
- Server responds with a UDP acknowledgement containing a results retrieval key.
- Clients connect via TCP to fetch search results.

PDTP Development – libpdtp



- Client library designed for use in single threaded and multithreaded applications, written in C.
- Portable across all Win32 operating systems and POSIX.
- Available on SourceForge at: http://libpdtp.sf.net/

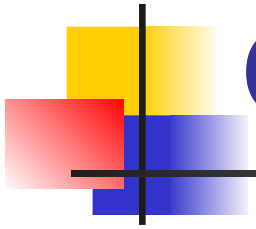
PDTP Development - Squall



- Squall is a development project for all PDTP server/proxy components.
- Squall utilizes advanced event and filesystem monitoring features to improve scalability.
- Squall is portable across Windows NT/2k/XP/2k3 and POSIX.
- Available on Sourceforge at: http://squall.sf.net/

Future PDTP Projects

- Skyfire – GUI client application written with Qt 4. Will allow PDTP searching, server browsing, and file download. On SourceForge at: http://skyfire.sf.net/
- USRI - APR for Squall. Provides a synchronous interface to a number of host specific features across multiple Win32 and POSIX systems. Development page: http://usri.pdtp.org/



Getting Involved

- Join the developers mailing list: developers-subscribe@pdtp.org
- Visit the project forums at: http://forum.pdtp.org/
- Visit the project web page at: http://www.pdtp.org/