BitTorrent

Filip Rachwalak
Paweł Jurczenko

Wstęp (1)

Nazwa **BitTorrent** odnosi się do trzech elementów:

1. Protokółu wymiany i dystrybucji plików w sieciach P2P.
2. Pierwszej aplikacji klienckiej umożliwiającej korzystanie z protokołu BitTorrent.
3. Amerykańskiego przedsiębiorstwa odpowiedzialnego za utrzymywanie protokołu BitTorrent oraz aplikacji BitTorrent i uTorrent.

Wstęp (2)

Protokół BitTorrent:

- autor: Bram Cohen,
- zaprojektowany i opublikowany w 2001 roku,
- ponad 150 milionów aktywnych użytkowników miesięcznie (stan na grudzień 2011),
- nieustandaryzowany

Przedsiębiorstwo BitTorrent:

- założone w 2004 roku przez Brama Cohena i Ashwina Navina

Terminologia

- *Peer* - węzeł w sieci biorący udział w wymianie plików,
- *Seeder* - węzeł posiadający pełną kopię wymienianych plików,
- *Torrent* - nazwa pliku lub grupy plików, pobieranych przez użytkowników,
- *Tracker* - scentralizowany serwer przechowujący informacje o użytkownikach danego torrenta,
- *Plik z metadanymi* - plik tekstowy zawierający informacje o konkretnym torrencie,
- *Peer ID* - 20-bajtowy string identyfikujący danego peera,
- *Info hash* - hash SHA1 unikalnie identyfikujący dany torrent. Obliczany na podstawie pliku z metadanymi.

Elementy systemu (1)

Wymagane:

- tracker protokołu BitTorrent*
- plik .torrent zawierający metadane
- osoba udostepniająca pliki zdefiniowane w metadanych
- użytkownicy końcowi (i ich klienty)

Opcjonalne:

- serwer webowy
- aplikacja webowa

* istnieją rozwiązania bez trackera, tzw. trackerless

Elementy systemu (2)

Wymagane:

- tracker protokołu BitTorrent*
- plik .torrent zawierający metadane
- osoba udostepniająca pliki zdefiniowane w metadanych
- użytkownicy końcowi (i ich klienty)

Opcjonalne:

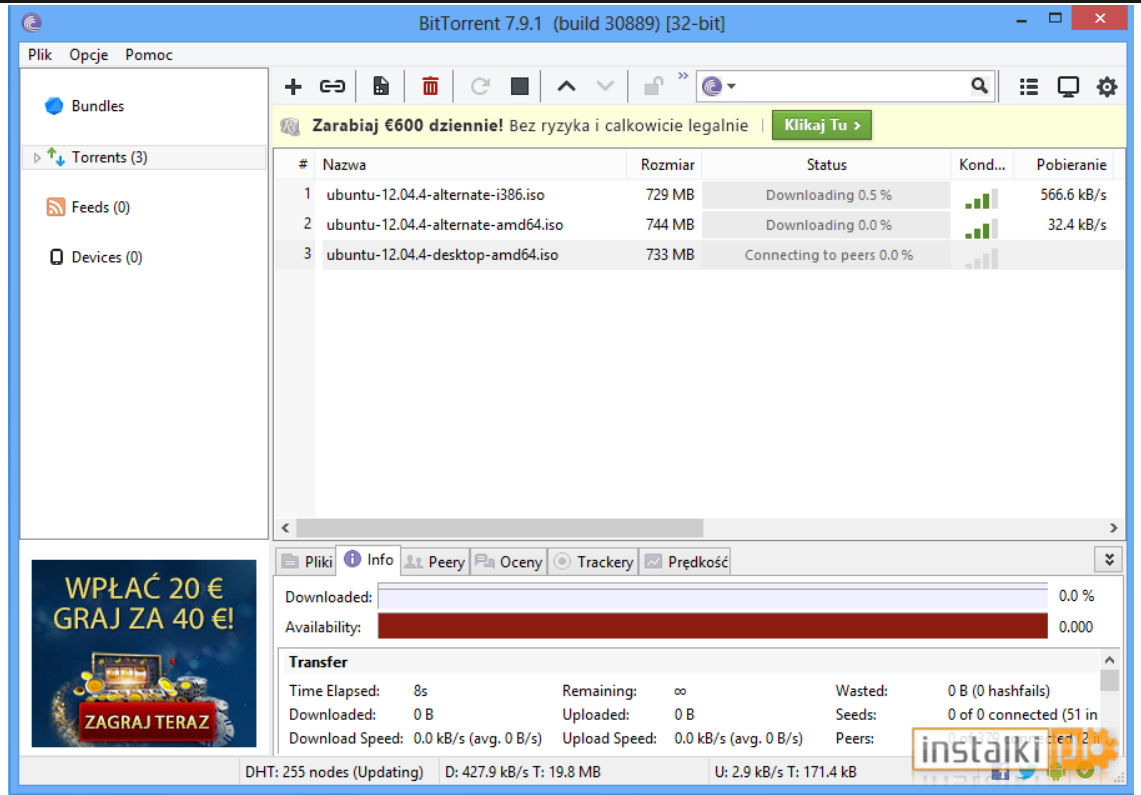
- serwer webowy
- aplikacja webowa

* istnieją rozwiązania bez trackera, tzw. trackerless

Klienty - BitTorrent (1)

- pierwsze wydanie - 2 lipca 2001,
- do wersji 5.3 pisany w Pythonie i udostępniany na zasadach open source,
- od wersji 6.0 jest to zamknięte oprogramowanie pisane w C++, oparte na uTorrent’cie,
- dostępny na: Windows, Mac OS X

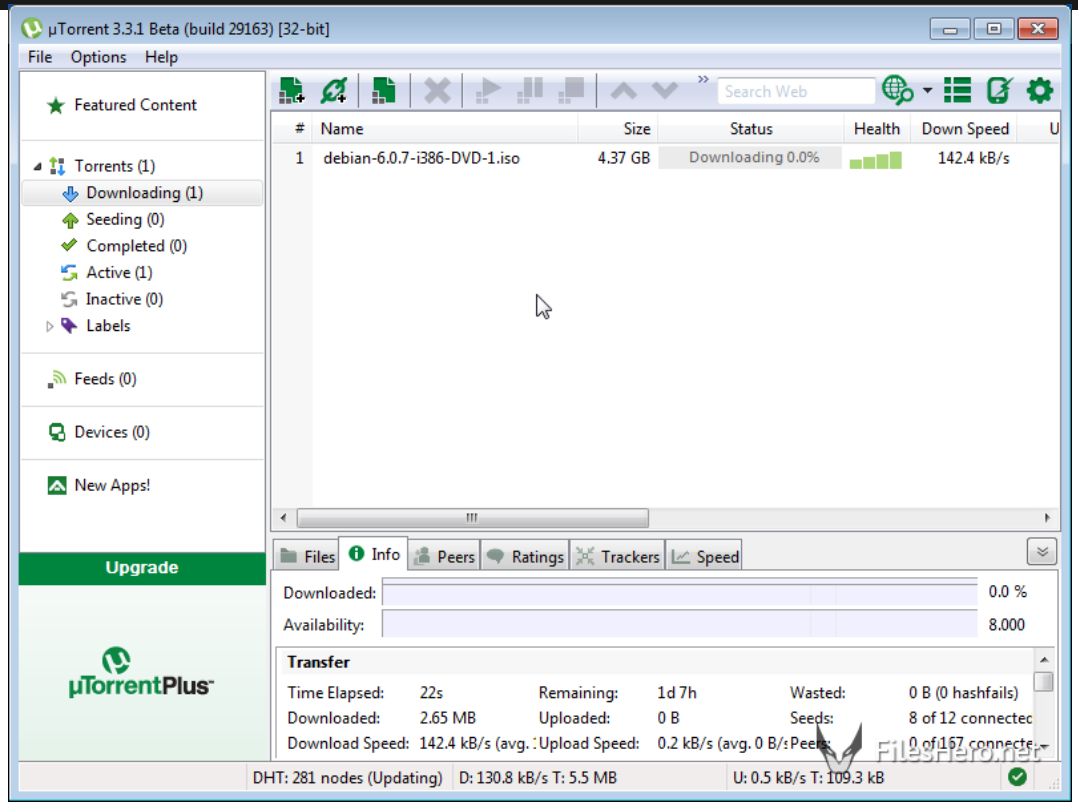
Klienty - BitTorrent (2)



Klienty - µTorrent (1)

- najpopularniejszy klient protokołu BitTorrent na świecie (za wyjątkiem Chin, gdzie dominuje Xunlei),
- w 2006 roku wykupiony przez firmę BitTorrent,
- przedrostek µ (micro) oznacza “lekkość” aplikacj: uTorrent zużywa bardzo mało zasobów komputera,
- jego lekkość wynika z zastosowania języka C++, zrezygnowania ze standardowych bibliotek oraz kompresji za pomocą “executable packera” UPX,
- dostępny na: Windows, Mac OS X, Android, Linux (wersja wspierana przez WINE lub natywna wersja Server z interfejsem konsolowym),
- licencja: adware :(

Klienty - µTorrent (2)



Klienty - inne

- ***qBittorrent***, C++
  FreeBSD, Linux, Mac OS X, Windows, Android
- ***Transmission***, *C*
  BSDs, Linux, Mac OS X, Windows, Solaris
- ***Deluge****,* Python
  FreeBSD, Linux, Mac OS X, Windows
- ***Azureus/Vuze****,* Java

Elementy systemu (3)

Wymagane:

- tracker protokołu BitTorrent*
- plik .torrent zawierający metadane
- osoba udostepniająca pliki zdefiniowane w metadanych
- użytkownicy końcowi (i ich klienty)

Opcjonalne:

- serwer webowy
- aplikacja webowa

* istnieją rozwiązania bez trackera, tzw. trackerless

Elementy systemu (4)

Wymagane:

- tracker protokołu BitTorrent*
- plik .torrent zawierający metadane
- osoba udostepniająca pliki zdefiniowane w metadanych
- użytkownicy końcowi (i ich klienty)

Opcjonalne:

- serwer webowy
- aplikacja webowa

* istnieją rozwiązania bez trackera, tzw. trackerless

Elementy systemu (5)

Wymagane:

- tracker protokołu BitTorrent*
- plik .torrent zawierający metadane
- osoba udostepniająca pliki zdefiniowane w metadanych
- użytkownicy końcowi (i ich klienty)

Opcjonalne:

- serwer webowy
- aplikacja webowa

* istnieją rozwiązania bez trackera, tzw. trackerless

Torrent - organizacja

- wszystkie pliki wchodzące w skład torrenta są traktowane jako jeden ciągły strumień bajtów (konkatenacja plików),
- kolejność plików jest zdefiniowana w metadanych,
- ciągły strumień danych dzielony jest następnie na równe kawałki (ang. pieces), które z kolei dzielą się na bloki (ang. blocks),
- z każdego kawałka wyliczany jest skrót **SHA1** i umieszczany jest on w pliku z metadanymi,
- rozmiar kawałków zależy od całkowitego rozmiaru plików: powinien być wielokrotnością 2 i powinien być tak dobrany, żeby plik .torrent zajmował ok. 50-75 kB,
- typowe rozmiary kawałków: 256 kB, 512 kB, 1 MB

Plik z metadanymi - wstęp

- dostarcza klientowi informacje o adresie trackera,
- zawiera informacje o wszystkich plikach wchodzących w skład torrenta,
- zawiera informację o wszystkich kawałkach (ich skróty SHA1),
- powinien mieć rozszerzenie **.torrent**,
- powinien mieć typ MIME **application/x-bittorrent**,
- jest binarnie zakodowanym słownikiem wartości

Plik z metadanymi - struktura (1)

- *announce* - zawiera adres URL trackera,
- *comment* - opcjonalny komentarz,
- *created by* - opcjonalna informacja o wersji i programie wykorzystanym do stworzenia pliku . torrent,
- *creation date* - opcjonalna data stworzenia pliku,
- *info* - wskazuje na słownik zawierający informacje o plikach do pobrania

Plik z metadanymi - struktura (2)

Struktura słownika **info** dla torrentów z jednym plikiem:

- *length* - rozmiar pliku w bajtach,
- *name* - nazwa pliku,
- *piece length* - rozmiar pojedynczego kawałka w bajtach,
- *pieces* - konkatenacja wszystkich skrótow SHA1 (wyliczonych z poszczególnych kawałków)

Plik z metadanymi - struktura (3)

Struktura słownika **info** dla torrentów z wieloma plikami:

- *files* - lista słowników, z których każdy przechowuje informacje o jednym pliku:
  - *length* - rozmiar pliku w bajtach,
  - *path* - ścieżka pliku w strukturze katalogowej
- *name* - nazwa głównego katalogu w strukturze katalogowej,
- *piece length* - rozmiar pojedynczego kawałka w bajtach,
- *pieces* - konkatenacja wszystkich skrótow SHA1 (wyliczonych z poszczególnych kawałków)

Elementy systemu (6)

Wymagane:

- tracker protokołu BitTorrent*
- plik .torrent zawierający metadane
- osoba udostepniająca pliki zdefiniowane w metadanych
- użytkownicy końcowi (i ich klienty)

Opcjonalne:

- serwer webowy
- aplikacja webowa

* istnieją rozwiązania bez trackera, tzw. trackerless

Elementy systemu (7)

Wymagane:

- tracker protokołu BitTorrent*
- plik .torrent zawierający metadane
- osoba udostepniająca pliki zdefiniowane w metadanych
- użytkownicy końcowi (i ich klienty)

Opcjonalne:

- serwer webowy
- aplikacja webowa

* istnieją rozwiązania bez trackera, tzw. trackerless

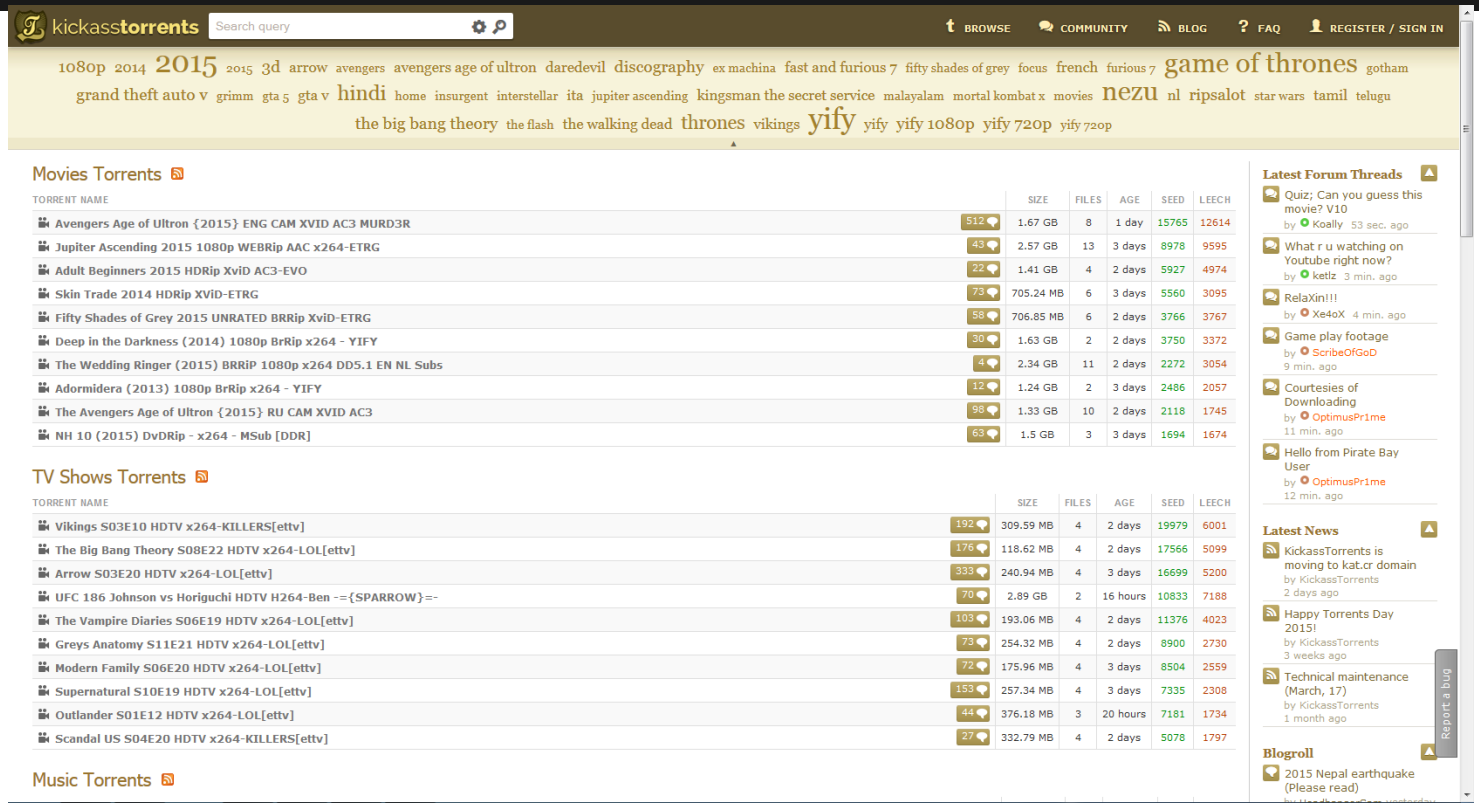
Trackery - wstęp

- usługa HTTP umożliwiająca użytkownikom dołączenie do wymiany plików w ramach danego torrenta,
- jedyny centralny element w ramach całego systemu,
- sam w sobie nie udostępnia żadnych danych będących fragmentami plików,
- swoje działanie opiera na komunikacji z peerami (może na przykład uznać że jakiś węzeł uległ awarii i nie należy już do sieci),
- każdy peer okresowo (np. co 30 minut) kontaktuje się z trackerem w celu zaktualizowania informacji,
- z punktu widzenia dostępności dzielą się na publiczne i prywatne

Trackery publiczne (1)

- nie wymagają posiadania konta użytkownika,
- nie premiują efektywnych węzłów,
- nie potępiają nieefektywnych węzłów,
- nie przeciwdziałają free-riderom,
- nie przeciwdziałają oszustom,
- przykłady trackerów: 1337x, EZTV, RARBG,
- przykłady stron oferujących pliki .torrent: KickassTorrents, Isohunt, Torrentz (meta-search)

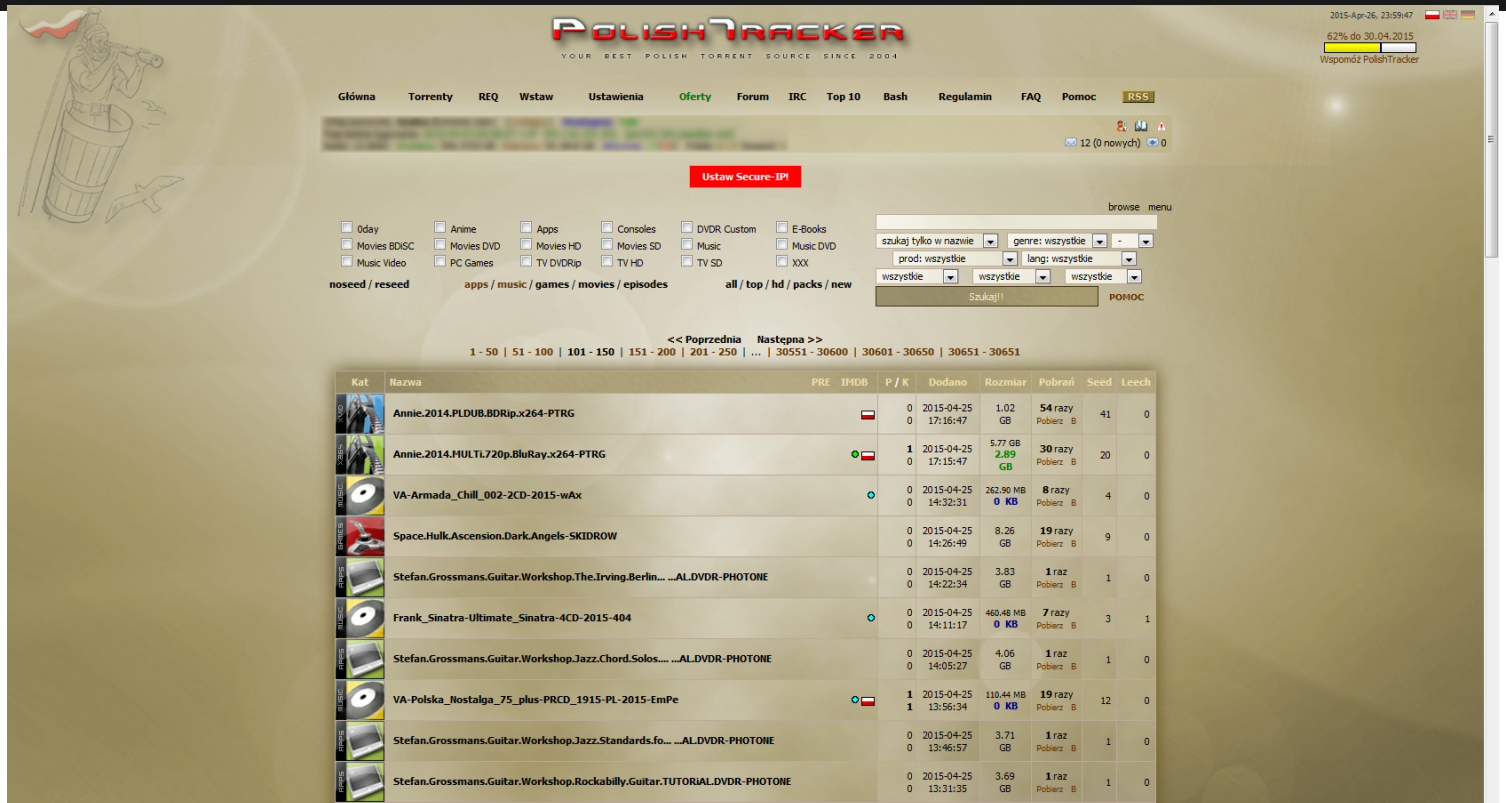
Trackery publiczne (2)



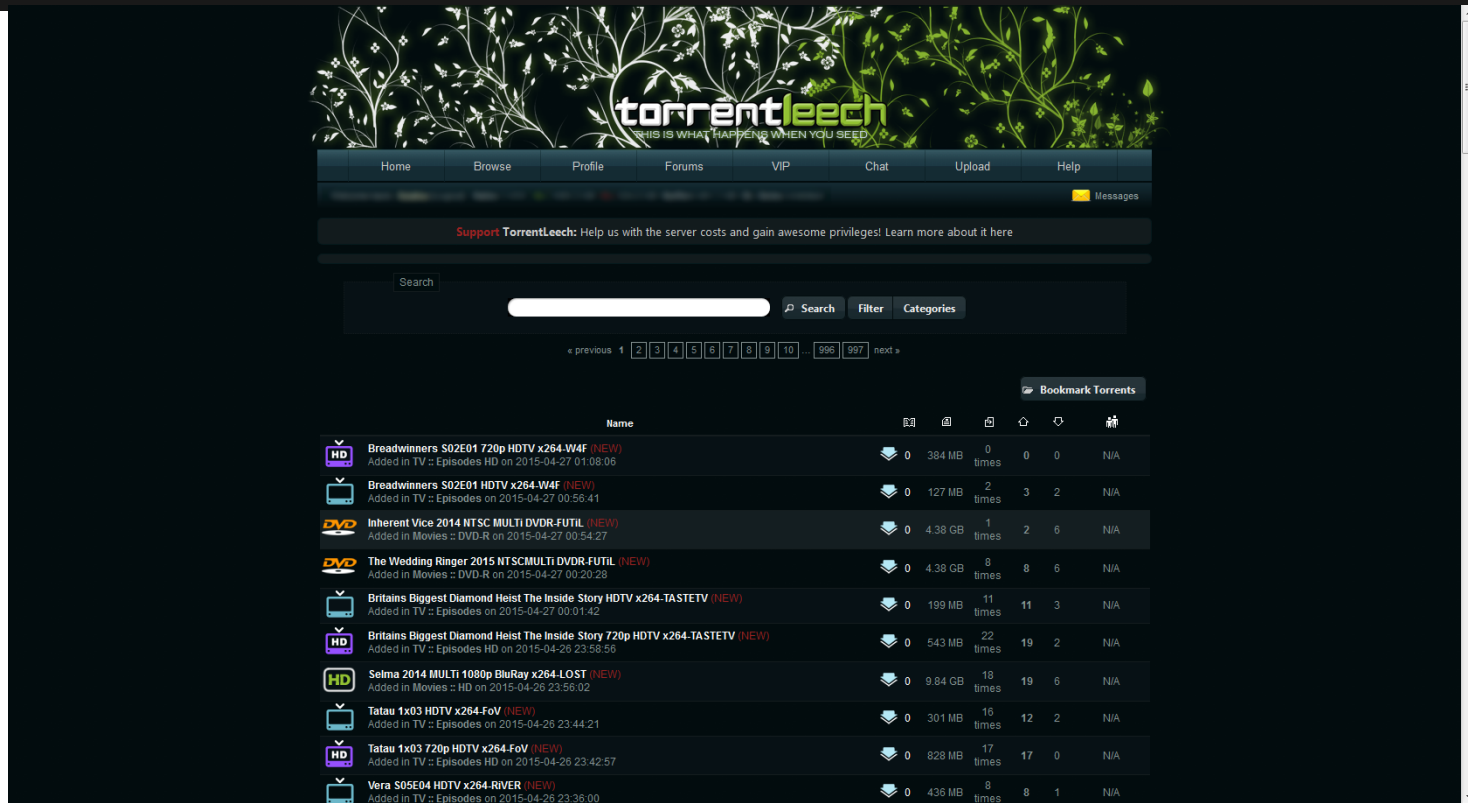
Trackery prywatne (1)

- wymagają posiadania konta użytkownika,
- czasami wymagają zaproszenia od istniejącego już użytkownika,
- rejestrują rzeczywisty wkład użytkownika do sieci wymiany plików,
- premiują efektywnych użytkowników (np. poprzez zniesienie limitów na liczbę pobieranych torrentów),
- potępiają nieefektywnych użytkowników (nakładają ograniczenia, nakładają ostrzeżenia lub w ostateczności - banują),
- przeciwdziałają free-riderom (rejestrują liczbę tzw. “hit-and-run”),
- częściowo przeciwdziałają oszustom (na tyle na ile skuteczny jest algorytm wykrywania zachowania sugerującego bycie oszustem),
- przykłady: PolishTracker, TorrentLeech, TorrentBytes

Trackery prywatne (2)



Trackery prywatne (3)



Trackery - sposób działania

- The Tracker HTTP Protocol - prosty mechanizm umożliwiający wzajemne poznawanie się peerów,
- konieczny podczas dołączania nowego węzła do sieci,
- nie posiada dostępu do wymienianych danych,
- swoją wiedzę opiera na (okresowych) żądaniach otrzymywanych od węzłów

Tracker HTTP Protocol (1)

Aby wysłać REQUEST do trackera, peer odczytuje jego adres URL z pola **announce** umieszczonego w pliku zawierającym metadane.

Wewnątrz REQUESTa musi znajdować się wiele parametrów...

Tracker HTTP Protocol Request (1)

- “info_hash” (required) - 20-bajtowy hash (SHA1) obliczony przez peera z wartości klucza **info** z pliku .torrent,
- “peer_id” (required) - 20-bajtowy łańcuch zawierający ID węzła wysyłającego REQUEST,
- “port” (required) - numer portu nasłuchiwania połączeń przychodzących od innych peerów,
- “ip” (optional) - powinien zawierać faktyczny publiczny adres peera (IPv4/IPv6/DNS name),
- “numwant” (optional) - oznacza liczbę peerów jaką chcielibyśmy otrzymać od trackera

Tracker HTTP Protocol Request (2)

- “event” (optional) - jeżeli został ustawiony, jego dopuszczalnymi wartościami są:
  - “started” - wartość obowiązkowa w przypadku pierwszego żądania HTTP GET do trackera
  - “stopped” - wysyłany podczas “łagodnego” (ang. graceful) odłączania się węzła od sieci,
  - “completed” - wysyłany w momencie zakończenia pobierania pliku

Tracker HTTP Protocol Request (3)

- “uploaded” (required) - ilość bajtów wysłanych w sieci od czasu przekazania wiadomości “started” do trackera
- “downloaded” (required) - ilość bajtów pobranych w sieci od czasu przekazania wiadomości “started”
- “left” (required) - ilość bajtów potrzebnych do zakończenia pobierania

Tracker HTTP Protocol (2)

Po otrzymaniu REQUESTa tracker musi odpowiedzieć dokumentem (o typie MIME **text/plain**), który zawiera słownik z następującymi kluczami:

Tracker HTTP Protocol Response (1)

- “failure_reason” (optional) - zawiera szczegóły błędu w czytelnej formie (string),
- “interval” (required) - określa przedział czasowy pomiędzy kolejnymi REQUESTami,
- “complete” (optional) - liczba seedów,
- “incomplete” (optional) - liczba peerów w trakcie pobierania

Tracker HTTP Protocol Response (2)

- “peers” (required) - zawiera listę peerów, z którymi należy się połączyć w celu wymiany danych. Struktura wewnętrzna:
  - “peer_id” (required) - zawiera ID węzła
  - “ip” (required) - zawiera adres węzła (IPv4/IPv6/DNS)
  - “port” (required) - zawiera port

The Peer Wire Protocol - wstęp

- protokół wykorzystywany do komunikacji między klientami,
- każdy peer nasłuchuje połączeń na jednym ze swoich otwartych portów,
- IP i port peera są raportowane do trackera,
- komunikację między dwoma węzłami musi rozpoczynać **handshake** (zawierający podstawowe informacje, takie jak nazwa protokołu, info hash czy peer id),
- po handshake'u następuje wzajemna wymiana komunikatów typu **bitfield**, będącymi informacją o posiadanych przez węzeł kawałkach

The Peer Wire Protocol - przechowywany stan

- każdy peer musi przechowywać dwie flagi o każdym węźle z którym jest połączony:
  - flagę **choked**, oznaczającą że węzeł jest “zduszony” i nie może ubiegać się o dane zdalnego węzła,
  - flagę **interested**, oznaczającą że zdalny węzeł jest zainteresowany pewnymi danymi
  - stan powyższych flag jest przesyłany w komunikatach typu **choke**, **unchoke**, **interested** i **uninterested**
- żądania danych przesyłane są komunikatem typu **request**,
- odpowiedzi z danymi (kawałkami) są przesyłane w komunikatach typu **piece**

The Peer Wire Protocol - strategia wyboru kawałka

- strategia wyboru następnego kawałka, który należy pobrać, nie jest zdefiniowana przez twórcę protokołu,
- z reguły stosuję się stratęgię **Local Rarest First (LRF)**,
- polega ona na wybieraniu takiego kawałka, który jest posiadany przez najmniejszą liczbę węzłów, z którymi jesteśmy połączeni,
- dzięki temu kawałki których jest mało lub są wręcz unikalne (np. posiada je tylko jeden węzeł), będą jak najszybciej dystrybuowane dalej

The Peer Wire Protocol - strategia wyboru węzła

- sposób wyboru węzła, któremu powinniśmy udostępniać dane, nie jest zdefiniowany przez twórcę protokołu,
- problem ten jest rozstrzygany przez zastosowanie różnych metryk, takich jak:
  - “ochota” zdalnego węzła na udostępnianie nam danych (jak długo jesteśmy dla niego **unchoked**),
  - przepustowość łącza zdalnego węzła,
  - wkład zdalnego węzła w rozpowszechnianie danych
- standardowym rozwiązaniem jest zastosowanie strategii **tit-for-tat** (wet za wet), polegającej na udostępnianiu danych czterem najlepszym (w znaczeniu posiadanej przepustowości łącza) węzłom oraz jednemu węzłowi losowemu
- pozostałe węzły są wówczas w stanie **choked**,
- wykorzystanie losowego węzła pozwala na znalezienie jeszcze lepszego kandydata do wymiany niż obecni oraz umożliwia słabym węzłom udział w wymianie danych

BitTorrent - security

Kilka możliwych ataków na sieć działającą pod protokołem BitTorrent

Atak DoS na trackera

Ataki typu DoS przeciwko trackerom będą skutkować głodzeniem “prawdziwych” peerów.

Możliwość obrony przed tym atakiem to umieszczenie kilku trackerów w pliku .torrent, przez co rozproszymy ruch zmniejszając ich obciążenie.

Peer ID issue

W protokole BitTorrent nie ma silnego uwierzytelnienia podczas komunikacji z trackerem (adres IP oraz peer ID nadane samemu sobie przez węzeł). Atakujący będący na węźle ofiary (ten sam adres IP) mógłby łatwo podszyć się pod nią podając jej peer ID.

Niektóre klienty umieszczają zbyt dużo informacji w stringu tworzonym na potrzeby peer ID

Validating the Integrigty of Data Exchanged Between Peers

Wszystkie dane oferowane przez innych peerów powinny być traktowane jako “podejrzane”. Węzeł powinien zwalidować pliki (hashe) przed ich akceptacją i pobraniem.

Kwestia poprawności plików pobranych to w dużej mierze kwestia poprawności pliku z metadanymi.

File and directory names issue (1)

Plik z metadanymi umożliwia sugerowanie nazwy ściąganego pliku albo głównego folderu dla ściąganych plików.

Dopracowany klient protokołu BitTorrent powinien sprawdzić czy te wartości nie zastępują aktualnych usług systemu.

Dodatkowo należy sprawdzić czy ścieżki plików w wieloplikowych torrentach nie są względne (np. “..”, który umieszczałoby taki plik na zewnątrz głównego folderu).

File and directory names issue (2)

W ten sposób atakujący mógłby nadpisać ważne pliki w systemie (właściwie dowolny plik w zakresie uprawnień użytkownika).

W ogólności, aby tego uniknąć, klient BitTorrenta nie powinien nazywać plików i umieszczać ich w lokalizacji “sugerowanej” przez plik z metadanymi bez weryfikacji użytkownika.

Trackerless systems

Rozwiązania działające bez trackera:

- DHT
- PEX
- Magnet Links

Trackerless systems PEX

- Pytam innych o adresy następnych peerów.
- Zwiększona szybkość dzięki odciążeniu centralnego trackera.
- Pozbywamy się Single Point of Failure.
- Przyłączanie peera do sieci musi odbywać się za pośrednictwem trackera lub tzw. bootstrap node (DHT).

Trackerless systems
Magnet Links (1)

- Część standardu URI
- Nie wskazuje miejsca pobierania!

Trackerless systems
Magnet Links (2)

- Magnet link tylko potwierdza, że istnieje plik pasujący do linku abstrahując od jego lokalizacji.
- Hash wygenerowany w magnet linku nie zawiera żadnych informacji o nim - służy tylko i wyłącznie do jego identyfikacji
- Użytkownik sam musi sobie “poszukać” pliku - najczęściej klient dokonuje tego poprzez DHT

Trackerless systems
Magnet Links (3)

Dodatkowe informacje w magnet linku

magnet:?xt=urn:sha1:2XINZZZYCTWF4O7LN4375KNLELSDXR5F&
dn=Iron-Man-2.avi&as=http://www.freebase.be/g2/dlcount.
php?sha1=2XINZZZYCTWF4O7LN4375KNLELSDXR5F

- **sha1 (secure hash algorithm):** algorytm, który został użyty do zaszyfrowania adresu IP.
- **dn (display name):** za tą informacją kryje się plik z tekstem, mający ułatwić wyszukiwanie.
- **as (accteptable substitute):** wyświetla źródła pobierania, aby klient szybciej znalazł plik; w tym celu serwis sieciowy (tutaj Freebase.be) wyświetla adres IP pliku.
- **adres IP:** zakodowane źródło pobierania, które we Freebase.be pozostaje zapisane przez tydzień.

BitTorrent a legalność

BitTorrent “umywa ręce”:

"Wiele osób nie zdaje sobie sprawy, że mamy w naszym ekosystemie ponad dwa miliony licencjonowanych treści. To prawda, że nasza technologia wchodzi w skład technologii wykorzystywanych do piractwa. Ale przekonasz się, że jako samodzielne narzędzie nie jesteśmy bardzo dobrym narzędziem do piractwa. Nie udostępniamy treści, które łamią prawa autorskie. Nie wskazujemy ich i nie promujemy w żaden sposób. Wszystkie tego typu działania dzieją się poza nami"

Matt Mason, BitTorrent

BitTorrent a legalność

Art. 23 *ustawy o prawie autorskim* mówi, że:

wolno nieodpłatnie korzystać z już rozpowszechnionego utworu, a użytkownik nie ma prawnego obowiązku sprawdzania legalności pochodzenia pobieranego zasobu

ALE! (Art. 278 kodeksu karnego):

Kto zabiera w celu przywłaszczenia cudzą rzecz ruchomą, podlega karze pozbawienia wolności od 3 miesięcy do lat 5.

Tej samej karze podlega, kto bez zgody osoby uprawnionej uzyskuje cudzy program komputerowy w celu osiągnięcia korzyści majątkowej

BitTorrent a legalność

Art. 116:

1. Kto bez uprawnienia albo wbrew jego warunkom rozpowszechnia cudzy utwór w wersji oryginalnej albo w postaci opracowania, artystyczne wykonanie, fonogram, wideogram lub nadanie, podlega grzywnie, karze ograniczenia wolności albo pozbawienia wolności do lat 2.

2. Jeżeli sprawca dopuszcza się czynu określonego w ust. 1 w celu osiągnięcia korzyści majątkowej, podlega karze pozbawienia wolności do lat 3.

4. Jeżeli sprawca czynu określonego w ust. 1 działa nieumyślnie, podlega grzywnie, karze ograniczenia wolności albo pozbawienia wolności do roku.

BitTorrent a legalność

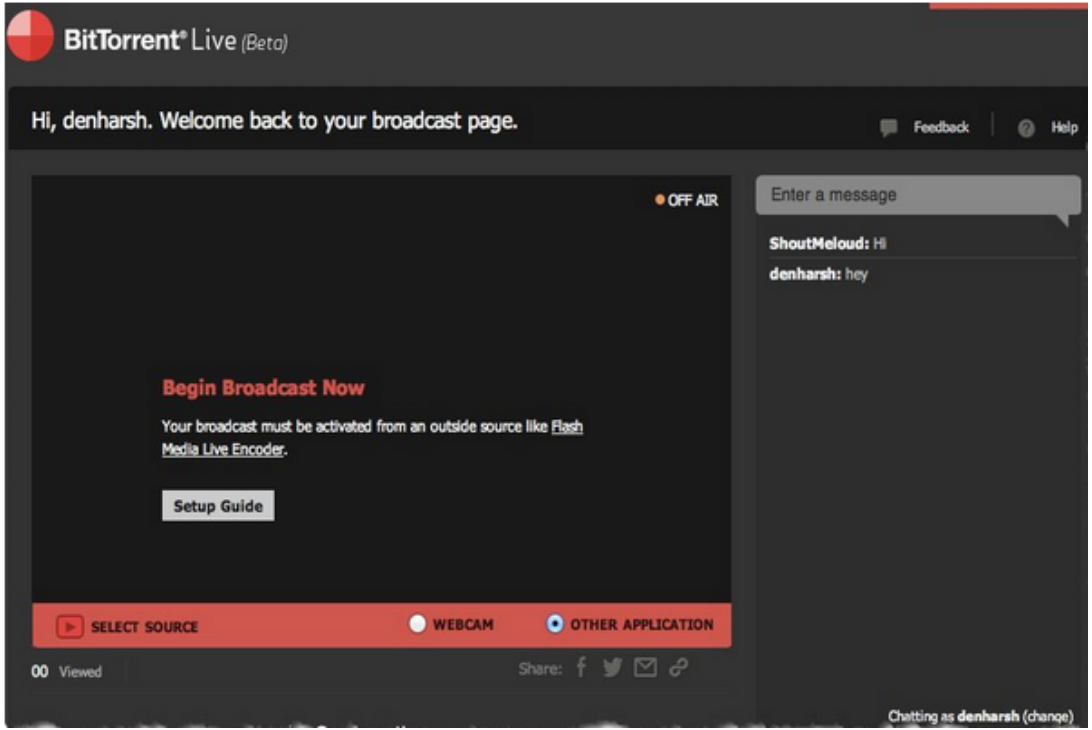
Nielegalne jest udostępnianie, a nie pobieranie, co nie zmienia faktu, że wiele klientów BitTorrent (nawet po ustawieniu limitu upload'u na 0) również rozsyła pliki bez wiedzy użytkownika.

Oprogramowanie oparte o BitTorrent

- BitTorrent Live
- BitTorrent Sync
- BitTorrent Bundle
- BitTorrent DNA
- Micro Transport Protocol (µTP)

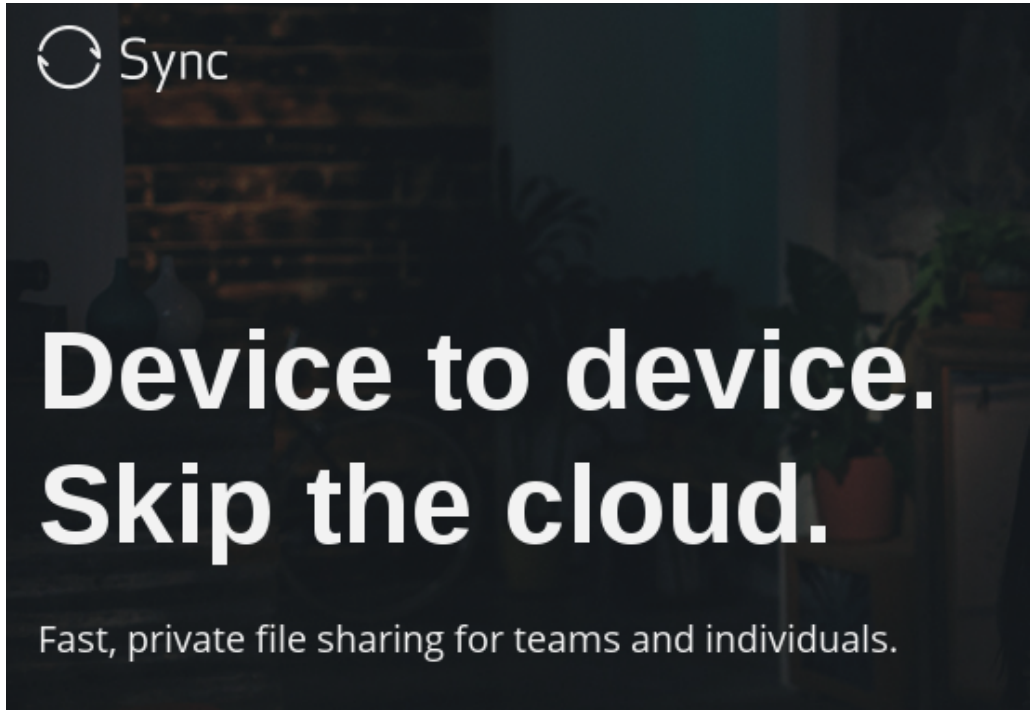
BitTorrent Live

- Platforma do live-streamingu
- Użyteczna dla video-bloggerów oraz internet-marketerów ze słabym łączem
- Oglądający stają się mini broadcasterami
- Wbudowany chat



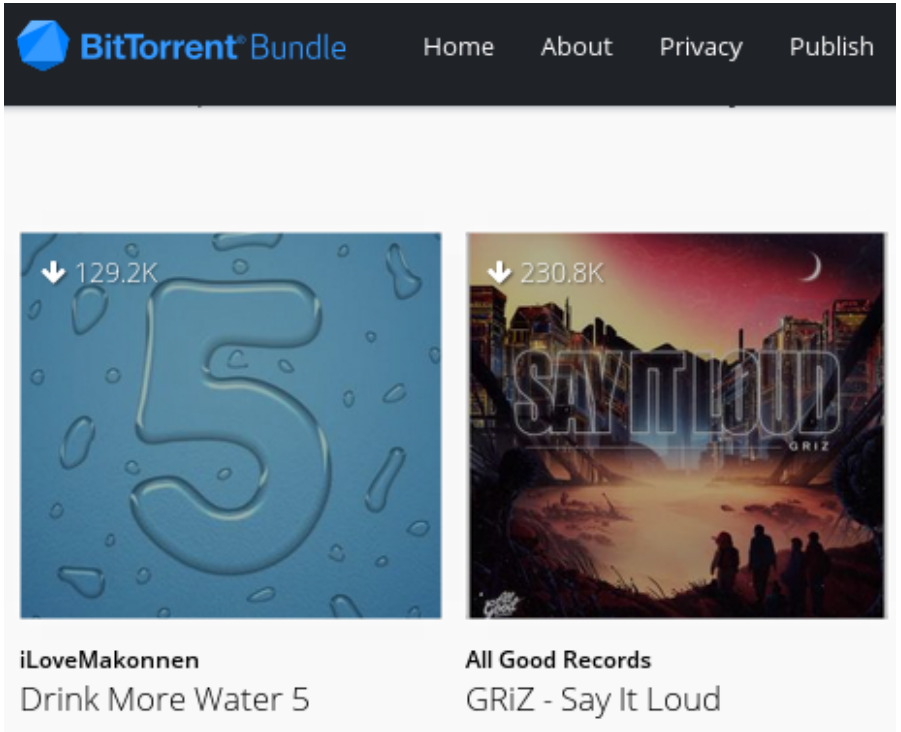
BitTorrent Sync

- BitTorrent między własnymi folderami
- Bez pośrednictwa serwerów
- Przesyłanie zmian w plikach zamiast całych plików
- Synchronizacja wybiórcza (zarówno w sensie plików jak i użytkowników)



BitTorrent Bundle

- Platforma przeznaczona dla twórców filmowych i muzycznych, którzy za jej pośrednictwem mogą dzielić się swoim dorobkiem
- Artyści mają pełną kontrolę nad dystrybucją materiałów (darmowe, płatne “co łaska”)



BitTorrent DNA (Delivery Network Accelerator)

- Program zaprojektowany, by przyspieszyć oglądanie videostreamingów oraz pobieranie oprogramowania. Działa to dzięki temu, że aplikacja dystrybuuje pobrane pliki pomiędzy użytkownikami, dzięki czemu serwery oryginalne nie są obciążone.
- Zanim pobierane są pliki od innych peerów, pobierający węzeł musi najpierw połączyć się z oryginalnym serwerem.
- Instalowana domyślnie (!) z klientem BitTorrent

Micro Transport Protocol (µTP)

- Stworzony przez BitTorrent oraz MIT w celu łagodzenia opóźnień i wykluczenia zatorów sieci.
- Powstał podczas badań nad QoS w celu poprawienia wydajności przesyłania dużych plików.
- Główną zmianą jest oparcie protokołu na UDP.
- Automatycznie zmniejsza zużycie łącza, gdy potrzebuje tego inna aplikacja.

Bibliografia

- http://jonas.nitro.dk/bittorrent/bittorrent-rfc.html
- https://wiki.theory.org/BitTorrentSpecification
- http://en.wikipedia.org/
- Bram Cohen. Incentives Build Robustness in BitTorrent. 2003.
- A. Bharambe and C. Herley. Analyzing and Improving BitTorrent Performance. Technical Report MSR-TR-2005-03, Microsoft Research, 2005.

Koniec