0.0.1版

なぜなにトレント はっじまるよ!!

Lorem Ipsum Dolor Facilisis

はじめに

1

はじめに

1. Torrent は悪いイメージもあるが貢献している

2. P2Pの学習に最適

はじめに

Torrentクライアントを開発しています。この本は、その時に P2P について学習した事をまとめたものです。

今でこそ、仕様書に書かれている内容に満足していますが、当初は上手く理解できない所が、かなりありました。

実際に作成していくうちに、理解できる範囲が広がっていきました。そして、そろそろ完成というところまで進みました。

この本では、この経験にならい、実際に Torrent クローンを作成しながら、P2Pについて解説して行きたいと思います。

ネガティブなイメージ

ちまたでは、Torrent は違法なファイル共有アプリという認識が強いように思います。 Twitter の検索機能で、Torrent と検索してみてください。 Torrentについてネガティブなイメージを持つことでしょう。違法な利用を助長するようなツイートを発見する事ができるからです。

Google で、Torrentと検索してみてください。「 アメリカ合衆国のデジタルミレニアム著作権法に基づいたクレームに応じ、このページからxx件の検索結果を除外しました。」 といった文言が表示されます。また、Twitterで検索した結果と同じように

違法な利用を助長するようなサイトを発見することができるでしょう。

世の中に貢献している

しかし、こういったネガティブなイメージは Torrent の一面でしかありません。まっとうな使い方も多くされています。

例えば、大規模なネットワークシステムのデプロイといった事があげられます。デプロイとは、ユーザーへサービスを提供するための、準備作業の事をいいます。

Google であれば、データを即座に検索できる用にサーバーを立ち上げる。 Twitter ならば、ツイートの送信、表示、などをできるようにするといった事です。

大手のネットワークでは数千、数万のコンピュータが動作しています。

これらのコンピュータへ変更を加える必要が出てきた場合には、この変更を数千、数万のコンピュータへ反映する必要があります。こういったデータの配信の Torrent の技術が利用されています。

また、大量にデータを配信する環境が必要ななのは企業だけではありません。個人でも活用されています。例えば、OSのイメージの配信に利用されています。

OSというと、WindowsやMACなどメーカーが CD や DVD といった記憶媒体を通して配布される事をイメージするかも知れません。

しかし、独自にOSをパッケージングする事は、大手の企業だけがする仕事ではありません。今では個人が趣味でOSをパッケージングして配信する事も可能です。可能なだけではなく、ありふれた行為になりつつあります。

しかし、個人で配信する場合、当然ながら数千、数万のサーバーを用意する事はできません。また、 CD や DVD の記憶媒体を大量に配るにしても限界があります。そういった、個人がデータを配布するのにTorrentの技術が利用されています。

学習に最適

また、Torrent を学ぶことは、P2Pを学ぶ上で最適な教材でないかと考えられます。

まず Torrent は、もっとも普及した P2P の通信方法であります。 それだけではなく、 最新の技術を取り入れ進化し続けている技術でもあります。

Torrent を学習すると、ネットワークアプリの作成のノウハウ、 分散ハッシュテーブル、 ゲーム理論を応用した柔軟なネットワークなどなど、 基本から応用まで扱う事になります。

なによりも、おおくのアプリや仕様がオーブンに公開されており、P2Pを実例をもって学ぶことができるのです。

Torrent 以外でメジャーな P2P アプリはあります。 しかし Torrent ほど、オッピロゲなはないでしょう。日本で流行した Winny のソースは公開されていません。 Winny について知りたければ、 ハックする必要があります。海外で流行したWinmxもそうです。

Torrentはその仕様が公開されています。どのような通信プロトコルが利用されているのか文章化されています。おおくの実装例がオープンソースとして公開されています。公開されているだけではありません。さまざまな言語で書かれています。Python、 Ruby、Java、JavaScript、C++ などなど、ありとあらゆる言語で書かれています。

もしも、煮詰まった時は、これらの実装を読む事で保管する事も可能でしょう。あなたの得意な言語で読む事ができるのです。

さいごに

Torrent のネガティブなイメージは一面にすぎない事。多くの史実用的なプロダクトを利用されていること。 また、Torrent が

P2P の教材として優れている事ほ理解して頂けたでしょうか？
少しでも興味を持たれた方、本書を通してTorrentの理解の助けになれば幸いです。

Torrentって何

1. データの配信はコストが高い

2. Torrentは、低コストでデータは配信を可能にした

Torrentって何

Torrentを違法なファイル共有ソフトという認識をもたれているかも知れません。しかし、それは間違いです。 Torrent は違法な利用を助長するような機能は含まれていません。たとえば、Torrent には匿名性はほとんどありません。 また、Torrent クライアントはネットワークから特定のファイルを探し出し共有する機能を持っていません。

つまり、 Torrent クライアントを利用してデータをダウンロードする事は、Webブラウザーを利用して、Webサーバーからデータをダウンロードするのと差がありません。

Google などの検索エンジンを使用して欲しいデータを探し、データをダウンロードします。ただ違うのは、 Torrent はどの通信方法よりも効率良くデータを配信することができることです。

データの配信はとてもコストが高い

インターネットでデータ配信するとしましょう。 例えば生放送で動画を配信したい場合、どのくらいのコストがかかるでしょうか？

例えば、320×240 の画面サイズの動画だと、1秒間に50kb程度の帯域を使用します。任期のある生放送などは、一度に2000人以上の人が視聴します。この場合、2000×50kb=100mbのデーター転送が必要になります。

ご家庭にある通信回線は10MB程度ですから、100MBという値は既に個人で配信できる量ではありません。

低コストでデータ配信が可能な優れもの

Torrentはこれらのコストをきわめて最小化することができます。Torrentの仕組みを利用すれば、こういったサービスを個人が提供する事ができます。

Torrent はデータをダウンロードするユーザーもデータの配信に加わるようにする事でこの問題を解決しました。

当然のことですが、配信に加わるコンピュータが増えれば増えるほど配信できるデータ量は増えます。

例えば、2.5MBのデータ配信を配信するとしましょう。 これは、同時に50人くらいに配信できます。しかし、これが限界です。もしも、この50人が2.5MBの帯域をデータの配信に利用してくれたら、50×2.5mb= 150mbの配信が可能になります。3000人程度にデータを配信できます。

さらに、この3000人がデータの2.5MBの帯域を利用してくれたら、3000×2.5mb=7500mbの配信が可能です。これは、15万人に配信が可能な規模です。

Torrentはこのような仕組みを実際に形にしました。Torrentによって、データを欲しがっている全ての人に素早くデータを配信する事ができるようになったのです。

序文

Lorem Ipsum

1. **本書の目的は、P2Pが難解な問題であるという誤解をとく事**

序文

本書は、Torrentプロトコルについてまとめたものです。しかし、作者(kyorohiro)はBittorrentの作成者ではないので、その仕組みや、その仕組みにした意図をすべて組み取ることはできません。本書に記載されている内容は、kyorohiroかせ解釈し解決した事が書かれています。オリジナルではありません。オリジナルのように思案して、思考できるようになるべく、Torrentクローンを作成しました。

その仕組みを理解して実現しました。そして、独自のP2Pネットワークを思案して検討するに至ります。しかし偽物です。

色々書いていますが、偽物のの発言ですから「そのように解釈したのか？」「それは違うのではないか？」「この人もしかして理解できていないのでは？」的な視点で見てもらうのが、丁度良いと思います。

本書が目指すゴール

本書を読む事で

誰もがP2Pアプリを作る事ができるようになります。P2Pで動画配信、P2Pを応用した柔軟なネットワークについて思案し、実現できる事ができるようになります。

P2Pと聞いて「何か難しい事をしているのではないか？」「しDHTと聞いて、高度な数学的難解な問題？」といった誤解を解きます。

本書を読んだ後もP2Pはなんら難しい事ではなくて、小学生が算数の問題を解くような簡単な問題なのだと理解していることでしょう。そして、あなたは、最新の論文を読み。それも足らず、新たな仕組みを提案する側にたっているに違い有りません。

おおまかな仕組み

2

Lorem ipsum dolor rutur amet. Integer id dui sed odio imperd feugiat et nec ipsum. Ut rutrum massa non ligula facilisis in ullamcorper purus dapibus.

おおまかな仕組み

Lorem Ipsum

1. トラッカーサーバーでPeerを管理
2. 協力してデータを配信

おおまかな仕組み

本章では、Torrentクライアントを利用してデータをダウンロードする時のおおまかな処理の流れを解説します。

- データを配信している端末をさがす
- ネットワークに加わり、データをダウンロードする

最初に宣言した通り、本書はTorrentクライアントを実そうする仮定を通して、P2Pの仕組みを解説していきます。なので、直にTorrentの実装に入りたいのですが、ワンクッションおくことにしました。

この章では、これから対峙する相手を紹介します。対峙する相手の姿、形を明らかにします。

実装を行っていくと「行き詰まるポイント」があります。そのほとどは、Torrent で扱われている技術がシンプルな事に由来します。シンプルな問題と理解しつつも、そのシンプルさゆえに、確信がも持てづ行き詰まります。

本章ではTorrentが難解でない事を解説します。本章を読み終えた頃には、得た確信を疑うことなく実装できるようになっていることでしょう。

配信している端末を探す

Lorem Ipsum

1. サーバークライアント方式がある
2. DHTを利用する方式がある

配信している端末を探す

Torrent ネットワークでは、ひとつのデータに対して配信する端末は多数存在します。以前に説明した通り、データをダウンロードする端末がデータの配信に加わる訳ですから当然の事でしょう。

データをダウンロードするには、広大なネットワークからデータを配信している端末を探しだす必要があります。本パートでは、データを持っている端末を探し出す方法について解説します。

配信している端末を探すのは大変

インターネット上には数えられないほどの端末が存在しています。そして、どの端末がどのテーターを持っているか知るすべはありません。

例えば、インターネットは、IPv4アドレスで管理されています。IPv4の数だけインターネット上に端末が存在できます。IPv4は24bitの数字で表現されていますから、2の24乗=42億以上の端末を扱う事ができます。

さらに、現時点では、このアドレスは足りなくなりつつあり。IPv6というよりたくさん端末を扱う事ができる方法が利用されるようになりました。つまり、42億で収まらない端末が既にネットワーク上に存在しているのです。

この42億以上の端末から任意のデータを配信している端末を探すのは困難です。例えば、ひとつの端末へ確認するのに、10ミリ秒必要だとしましょう。すべての端末へ確認するとしたら、420億秒必要になります。これは、一年以上かかる事を意味します。

Torrent でお互いに位置を知る方法は2つある

Torrent では2つの方法が提供されています。ひとつは、サーバー・クライアントの仕組みを利用したものです。データを配信したい端末を管理するサーバーを用意する方法です。そのサーバーに聞く事で、配信している端末を発見する事ができます。

もうひとつは、DHTを利用したものです。データの配信/ダウンロードをしたい端末が協力してデータを配信したい端末を管理します。DHTに参加している端末に、欲しいデータの在処を聞く事で、配信しているデータを発見する事ができます。

サーバー・クライアント方式

データを配信したい場合、「データを配信する端末を管理するサーバー」を用意します。このサーバーの事を Tracker と呼びます。Tracker はGoogleやYahooといったサーバーと同じ仕組みで動作しており、「www.exsample.com」 や「xx.xx.xx.xx」といったアドレスを指定する事でアクセスできます。

「xxxx.torrent」と拡張子をもつデータTorrentファイル に、このアドレスを記載しておいて、データをダウンロードしたい端末に渡します。データをダウンロードしたい端末は、Trackerにアクセスして、データを配信しいてる端末を知ることができます。

Tracker を用意する必要があります。すこし敷居が高いです。公開されたサーバーをレンタルするなりして公開されたIPを取得する必要があります。

しかし、その瞬間データを配信している端末のアドレスを管理するだけの、超軽量なサーバーアプリを用意すめだけで、数千、数万の端末へデータを配信できるようになる訳ですから、メリットは大きいです。

また、自身で用意しなくても、公開されている Tracker サービス を利用する事もできます。

DHT(kademulia)を利用する方式

サーバー・クライアント方式と同じように、データをダウンローとしたい端末へTorrentファイルを渡します。

サーバー・クライアント方式と異なり、データを配信しいる端末を管理しているサーバーはTorrentファイルに記載されていません。

Torrentファイルから機械的に生成されるID (Hash値) を、元に、DHTネッワーク上から、データーを配信している端末を管理している端末を探し出します。

六次の隔たりを利用する

では、どのようにして、IDから配信している端末を探すのでしょうか？。その仕組みは簡単です。周知のTorrentクライアントへ、そのデータを配信していそうな端末を紹介してもらいます。これを繰り返す事で、データを配信している端末へたどりつくことができます。

例えば、自分は欲しいデータを配信している端末の事は知らなくても、友人は欲しいデータを配信している端末を知っているかも知れません。

友人は知らなくても、友人の友人は知っているかもしれません。現実世界では、「六次の隔たり」と言われており、6人もたどれば、世界中の誰とでもつながっているそうです。

Torrent ネットワークも同様の仕組みを利用することで、データを配信している端末を探し当てます。

距離を定義する

「六次の隔たり」は6回聞くだけで欲しいデータを持っている端末を探し出すというものです。これを実現するには、「誰に聞くと良いのか？」を知らなくてはなりません。聞く人を間違えると、6人で到達いる事はないでしょう。聞くだけではだめなのです。

誰に聞けば良いのかを知る尺度としてDHTでは距離を定義します。欲しいデータと距離が近い端末に聞ようにする事で、まったく関係の無い人に聞くリスクを軽減するのです。

全てのデジタルデータに値を割り振るアルゴリズムが存在します。DHTでは、このアルゴリズムを使用してデータに値を振ります。端末にも、同様に値を振ります。

値どうしの差分が距離です。　「へちまたん」と「ありすたん」にこのアルゴリズムをかけた結果、「100」、「123」と値を振られたとしましょう。この場合、「へちまたん」と「ありすたん」の距離は123-100で23と定義できます。

自分に近い距離の事はより詳しい

距離が定義できましたが、聞いた人が自分よりもそのデータについて詳しくないとダメです。

DHTネットワークでは、「自分と距離が近い人の情報はたくさん知っている。遠い距離の人もすこしは知っている」といった状態を維持するようにします。

これで、聞く相手は自分よりもより多くの情報を知っている可能性があがりましたね。無事「六次の隔たり」ほ実現できそうです。

データをダウンロードする

Lorem Ipsum

1. データ配信する
2. ブロックに分割する
3. 常に速度が早い回線を探す

ネットワークに加わりデータをダウンロードする

欲しいデータをもっている端末が見つかると、Torrent クライアントはデータのダウンロード処理を開始します。前章で説明した通り、Torrentクライアントま特徴のひとつとして、データのダウンロードすると同時に、ダウンロードしたデータを配信します。

単純に配信するサーバーが増える訳ですから、配信できるデータ量は増えます。しかし、注意深くその仕組みを考えなと、非効率になる恐れもあります。例えば、極度にデータの配信効率の悪い端末からデータをダウンロードしてしまう。一部の端末に負荷が集中してしまう。データの配信に加わらなくても、データのダウンロードが出来てしまう。といった事があげられます。

Tor-
rentは、これらの問題について考慮しています、多数の端末から、 効率的にデータを配信するダウンロードする仕組みを提供しています。

ブロック単位で、複数からダウンロード

Torrent クライアントはデーターを任意のブロックに分割して管理しています。データをダウンロードするのも、配信するのも、このブロック単位で配信します。

このブロック単位のデータを複数の端末から、戦略的にデータを配信してもらいます。これによって、効率的にデータを配信する事に成功しています。

- 転送速度が安定する

負荷が集中した場合、データの配信する量は減ります。このような場合、負荷が高い状態の端末からデータを配信してもらうべきではあれません。データの配信を一度止めて、別の端末からデータを配信してもらう事を検討すべきです。

Torrentでは、常にデータを配信する速度は計測されており、その瞬間最も安定してデータを配信できると思われる端末へデータの配信を依頼します。

- 上がり速度、下がり速度の差を平均かできる

我々が利用している回線はダウンロードする速度の方が、アップロードする速度よりも圧倒的に早い場合がおいです。2013年のWimaxはダウンロード速度が10Mbps、アップロード速度は2Mbpsとなっており。5倍くら差があります。

ひとつの端末からダウンロードするだけだと、相手がアップロードす速度分しかデータを配信してもらえません。つまり、Wimax上だと2Mbpsです。しかし、複数の端末からデータをダウンロードすれば、10Mbpsを一杯を利用する事ができます。

- フリーライドを許さない

また、データを配信してくれた端末へ、優先的にデータを配信します。データをよりおおく配信してくれた端末とは通信が良好なはずです。また、Torrentネットワークに貢献していない、まったくデータを配信しない端末は冷遇される事を意味しています。つまり、他にデータを配信要求がある協力的な端末ほどにはデータが配信されないのです。

これは、データの配信に加わった方が、より早くデータをダウンロードできる事を意味します。合理的に考えれば、データのダウンロードが完了するまでは、データの配信に協力する選択を選ぶことでしょう。それがもっとも早く、データをダウンロードする方法だからです。

Torrentファイルを

Lorem ipsum dolor rutur amet. Integer id dui sed odio imperd feugiat et nec ipsum. Ut rutrum massa non ligula facilisis in ullamcorper purus dapibus.

Torrentファイル

Lorem Ipsum

1. ト

2. し

Torrent ファイルを読みこんいでみる

Torrentでのデータのダウンロード処理は、Torrentファイルを読み込む所からはじまります。

それにならい、実際に Torrent ファイルを読み込み、必要な必要な情報を取得するところからはじめて見ましょう。

Bencoding

Lorem Ipsum

1. **Torrentファイルはbencodingて書かれている。**
2. **Bencodingは、Integer、String、List、Dictionary を扱える。**

Torrentファイルは Bencoding

Torrentファイルは、bencoding という形式で書かれています。Torrentファイルに記載されている事を読み解くためには、bencding/bencode を解釈できるようにならなくてはなりません。

まずは、Bencosingのパーサーを書いていきましょう。Bencodingは、文字列、整数、辞書、リストの4つのデータを扱うことができます。

beninteger : “i” [0-9]* “e”

benstring : <string length> “:” <bytes array string>

string length : [0-9]*

bendiction : “d” <dictelements> “e”

benlist : “l” <listelements> “e”

benobject : beninteger | benstring | bendiction | benlist

listelements : benobject (benobject)*

dictelements : benstring benobject (benstring benobject)*

そして、上記のようなフォーマットで書かれています。

- **文字列**

Bencode で文字列は、「<文字の長さ> “:” <文字>」という形式で書かれています。例えば、「torrentという文字列は、bencodeでは、「7:torrent」と書く事ができます。

もうひとつ、bencode の文字列は、バイトデータとして扱われる事もあります。IPアドレスのバイト表示や、Hash値などの非アスキーな範囲のデータなども、本形式で扱います。

例えば、日本語で「アイ」はSJISで表現すると「0x83, 0x41, 0x83, 0x43」の4バイトで表現できます。この場合、Bencodeでは、「4:アイ」と表記できます。

oden

    4:oden

オデン SJIS

    6:オデン

SHA1Hashデータ

    20:<SHA1 Hashデータ>

- 整数

整数は0より大きな値を表現するデータです。「“i” *[0-9] “e”」という形式で表すことができます。例えば、1024は「i1024e」と書くことができます。

ファイルのサイズ、ポート番号、といった、数字で表現できるものに利用します。

2

 i2e

1024

 i1024e

65526

 i65526e

- リスト

リストはデータ構造のひとつです。順序ありのデータを保持する事ができます。例えば、IPアドレスの一覧、ファイルの一覧、といったものを表現するのに使用します。

あいうえお、かきくけこ

　l12:あいうえお12かきくけこe

128、 100、500

　li128ei100ei500ee

- 辞書

辞書はデータ構造のひとつです。キーワードとデータを関連づけて保持する事ができます。

例えば、RPGゲームの主人公のパラメータとして、名前、性別、レベル、得意な魔法、といったものが設定されているとしましょう。辞書はこの、バラメータ名とその値を関連づけます。「名前」というキーワードで、主人公の名前を記録したりできます。

例えばlevelが13で、nameが山田、を辞書で表現すると、
di5:leveli13e4:name4山田e

- 今後の表記について

今後、データ構造を表す場合は、以下の表記を利用します。

リスト

「li512e4:teste」は、[512,test]

辞書

「d5:leveli13e5magic6halitoe」は {level:13,magic:halito}

Bencoding実装

Lorem Ipsum

1. **Lorem ipsum dolor sit amet, consectetur.**
2. **Nulla et urna convallis nec quis blandit odio mollis.**
3. **Sed metus libero cing elit, lorem ipsum. Adip inscing nulla mollis urna libero blandit dolor.**
4. **Lorem ipsum dolor sit amet, consectetur.**
5. **Sed metus libero cing elit, lorem ipsum. Quis que euismod bibendum sag ittis.**
6. **Sed metus libero cing elit, lorem ipsum.**
7. **Quis que euismod bibendum sag ittis.**

見慣れたデータ構造に落とす

Dart言語には、Bencoding のデータ構造をもっています。今回は、Bencdoingのデータを読み込み、Dart言語のデータ構造に落とこみます。

Bencodeの文字列は、Dart言語ではcore.Stringで表せます。Bencodeの数字は、Dart言語では、core.int で表せます。 Bencodeのリストは Dart言語のcore.List、Bencodeの辞書は、Dart言語ではMapで表現できます。

Bencodeはパースしやすい構造

Bencode はパースが容易な構造になっています。なぜならば、どのデータなのかが、最初の一文字目で判別する事ができるからです。

“i” ならば、整数。 “0-9” ならば、文字列、”l” ならばリスト、”d” ならば辞書 といった感じです。

これを、コードに直すと、

```
Object decodeBenObject(data.Uint8List buffer)  {
  if( 0x30 <= buffer[index] && buffer[index]<=0x39) {//0-9
    return decodeBytes(buffer);
  } else if(0x69 == buffer[index]) {// i
    return decodeNumber(buffer);
  } else if(0x6c == buffer[index]) {// l
    return decodeList(buffer);
  } else if(0x64 == buffer[index]) {// d
    return decodeDiction(buffer);
  }
  throw new ParseError("benobject", buffer, index);
}
```

書けます。見ての通り、先頭の値に応じて処理が分岐しているだけです。後は、おのおのデータ構造とみなして、変換してあげれば完成です。

例えば、リストは、以下のように書けます。

```
List decodeList(data.Uint8List buffer) {
  index++;
  List ret = new List();
  while(index<buffer.length && buffer[index] != 0x65) {
    ret.add(decodeBenObject(buffer));
  }
  index++;
  return ret;
}
```

とし