HTTPでリアルタイムサーバを扱ったオンライン対戦タイピングソフト

『vs Typing』

情報処理学科2年

K198017

若林 宏樹

目次

第１章　はじめに

1-1 きっかけ

第２章　作品概要

2-1 開発環境

2-2 作品説明

第３章　使用技術(何を以ってそれを選んだか、客観的視点で)

3-1 通信形式

HTTP, WebSocket, P2P(Peer to Peer)

3-2 オンラインゲームにおける快適さ

直接通信型、サーバ集中型、クライアント分散型の処理の差

3-3 マッチングシステム

第４章　制作を終えて

第１章　はじめに

1-1 きっかけ

第２章　作品概要

2-1 開発環境

【サーバ】

AWSEC2 AmazonLinux

PHP7.1

【クライアント】

Unity2019.4.7f1

2-2 作品概要

【作品名】

競技用タイピングソフト

【作品説明】

タイピングのプレイ形式は、ソロモードとマルチモードを用意。

ソロモードはクライアントのみで完結させており、オフライン時でも遊べるようになっている。ただし、オフライン時の記録はデータベースに反映されない。

マルチモードではHTTP通信でサーバを経由しプレイヤーデータのやり取りを行い、リアルタイムでの対戦を行う。

第３章　使用技術

3-1 通信形式

【通信形式概要】

リアルタイム通信を行うオンラインゲームを制作するに当たってサーバとの通信は必須であると考えた。今回はHTTPを使用したが、他にもWebSocketやP2P(Peer to Peer)等、偏にオンライン対戦を実現すると言っても様々な通信規約がある。

ここでは何故HTTPを選択したのかを、他の通信規約の解説も交えて説明していく。

【各通信規約についての説明】

・HTTP

🡨---- ここにHTTP通信のイメージ画像 ----🡪

・WebSocket

　WebSocketプロトコルは双方向通信を実現させるために作られた通信プロトコル。最初はHTML5の仕様として策定されていたが、現在は単独のプロトコルとして存在している。

　大雑把な流れとしては初めにHTTPのようにしてコネクションを確立し、以降TCP/IPのように振る舞うといったもの。

　この方式では一度コネクションを確立した後は、サーバとクライアントの双方から通信を行う事が可能になっている。これが双方向通信と呼ばれる所以である。

　また、一度コネクションを確立した後はそのコネクション上で通信を行う他、ヘッダのサイズが小さくなっており通信量の削減にもかっている。

　このように言うと良い事ずくめのようだが、前述のメリットはコネクションを同時に一つしか持てないサーバでは複数の通信用のサーバを確立する必要があり相性が悪い物だとデメリットと表裏一体となる可能性もある。また、HTTPプロキシでは緩和出来たDoS攻撃がTCPプロキシでは対処できない等の問題もあるとの事。

　今後HTTPに統合される可能性があるという話もあり、伝統的な通信規約であり私自身が使っていく事が増えるであろうHTTPを今回は選択した。

🡨---- ここにWebSocket通信のイメージ画像 ----🡪

・P2P(Peer to Peer)

P2Pはそもそもクライアント-サーバ方式ではない通信規約である。ネットワークに接続されたコンピューター同士が対等な立場、機能で直接通信を行う物である。

デメリットとしては、データを共有する全てのコンピューター同士が通信をするという仕様上、接続するコンピューターが増加するほどに通信経路が複雑化していってしまうという物がある。例えば2台だと通信経路は1つ、3台だと通信経路は3つだが、4台になると6つ5台になると11つと指数関数的に複雑になっていく事が分かる。

メリットとしては、間に介する物が純粋に少ない(お互いのコンピューターとネットワークのみ)為、少人数での通信であれば最小限のラグで実装出来るという物がある。またクライアント-サーバ方式では大人数のアクセスによってサーバ及び回線に負荷がかかる事もあるが、P2P方式では通信が複雑になるという部分に目を瞑れば接続数が膨大になっても特定の機器にアクセス集中が起きにくくなっている。

今回の作品では1vs1の対戦方式であり、観戦者も含めても大人数での接続という事にはならないのでこの方式での実装も考えた。しかし、今回のリアルタイムサーバを扱うというテーマから逸脱していたためこの方式での実装は見送った。

🡨---- ここにP2P通信のイメージ画像 ----🡪

3-2 オンラインゲームにおける快適さ

【オンラインゲームにおける快適さに対しての考え】

　私はオンラインゲームにおける快適さは所謂『重さ』に依存していると考えた。

　この『重さ』がゲームのストレスに影響して来る場面は起動時、シーン切り替え時、ゲーム中といった物が主だったものだろう。ここでは特にゲーム中のリアルタイム通信に関する物に着目していこうと思う。

　『重さ』は通信、計算、描画等が影響して来るのだが、それをサーバとクライアントにどの程度分散させるのかで調整が可能ではないかと予測し、処理の分散について注目した。

【処理分散の方式】

・クライアント分散型

　今回採用したのはこの形式。

・サーバ集中型

サーバ側で全ての処理を行うのでゲーム世界の動作自体は同期しているが、通信遅延などの影響でクライアント側に描画される映像には差が出る事があるという物。FPSやMMORPGなどで多く採用されている。

クライアント側で描画される映像に差異が出るとなるとFPSには向いていないように感じるかもしれないが、クライアント側の描画に僅かな遅延を入れたり、サーバ側の判定を遡って行うなどの工夫によって成り立てている。他にもこの方式では帯域が渋滞する問題が出る事が多いので通信の頻度を下げたり、それで問題があるものについてはデータを圧縮する等の工夫が凝らされている。

今回はサーバが資金の関係上AWSの無料枠という事で、どこまでの負荷に耐えうるかの試験を行う事が出来ず不安が残ったのでこの形式での実装を見送った。

・直接通信型

　これは前述したP2Pの事。互いの処理を互いに行い直接行うので所謂ラグが少なくなっている。正確に言うと、ラグに影響する要素の数が少なくなっている。

　この方式はマッチングのみをサーバを介して行い、ゲーム中はリアルタイムサーバ(ゲーム中の処理を行うサーバ)を介さずに通信・対戦を行うという物。

　主に1フレームのラグも許せない格闘ゲーム等で採用されている事が多く、私自身も採用を検討したがP2Pの項目で前述したような理由で今回の採用は見送った。