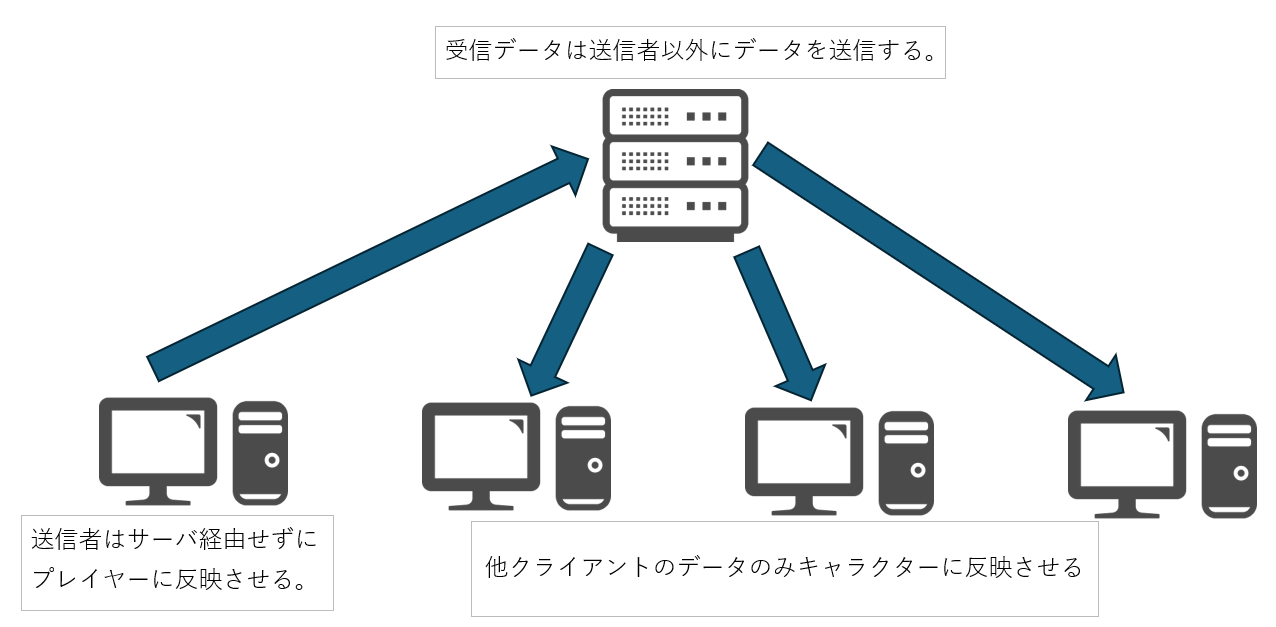
**---今回はクライアントから送られてきた情報を配信して、他のクライアントへ反映させます---**

* **内容**サーバを経由して複数ユーザと通信を行う。
* **目標**サーバ側で複数のクライアントを管理できるようになる。
* **プレイヤーの同期**前期はプレイヤー情報をそのままエコーサーバとして送信者に送り返していました。（送信者側では受け取ってそれで移動。）  
  今回は複数のクライアントをサーバ側で管理し、送信者**以外**にデータ送信を行い、同期を行っていきます。  
  （クライアント側では自プレイヤーは、自分の入力で動き、他プレイヤーは受け取ったデータでキャラを動かすことになります。）

サーバ側はUDP通信の為、誰かを判別する為、受信時にクライアントアドレス情報管理を行う必要があり、アドレス情報を保管、クライアントにアドレスに対応したIDを割り振り、どのキャラクターの同期データなのかを判断するようにします。  


* **データ通信の流れ**Loginの流れ・サーバ

1. クライアントからLoginデータを受信。
2. 送信者含めクライアント全員にIDを設定したLoginデータを送信。
3. 送信者に既存のクライアント情報を送信

・クライアント

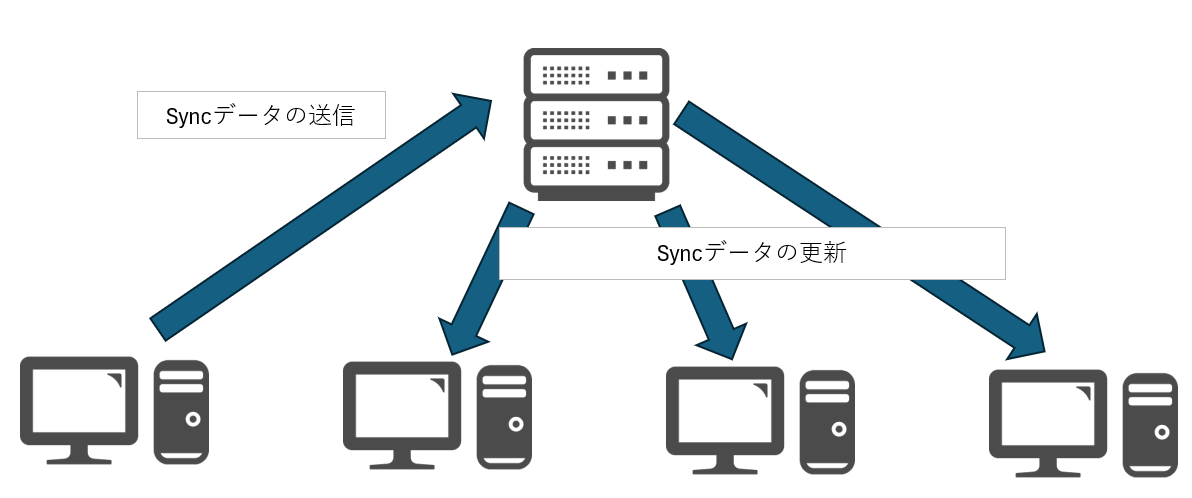
1. サーバにLoginコマンドを送信
2. サーバからIDを付与されたログインコマンドを受信し、プレイヤーを生成
3. ダイアグラム

   自動的に生成された説明サーバから自分より先にログインしているユーザ情報を取得し、各プレイヤーを生成。

Syncの流れ  
・サーバ

1. クライアントから送られてきたデータを送信者以外のクライアントに送信

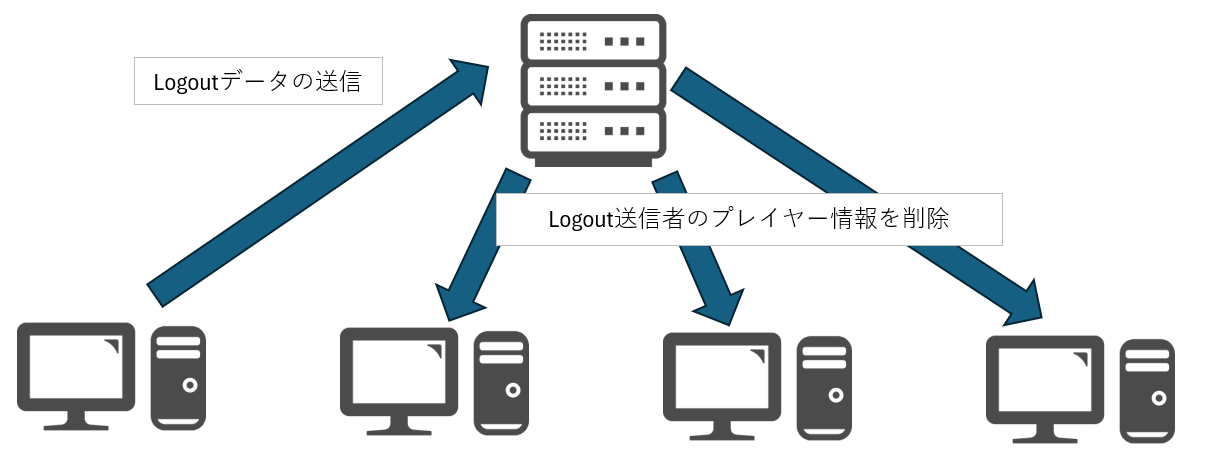
・クライアント

1. 毎フレーム、プレイヤーの情報をまとめたPlayerInformation構造体のデータを送信する。
2. 受信したデータはIDをもとにプレイヤー情報を更新する。

Logoutの流れ  
・サーバ

1. 受信したLogoutデータからIDを参照し、プレイヤー情報を削除。
2. 送信者以外のクライアントにデータ送信。

・クライアント

1. Finalize関数内でサーバにLogoutデータを送信する。
2. Logoutdデータを受信したとき、対象のIDのデータを削除する。

* **課題内容  
  ・サーバ側**クライアントから送られてきたデータのタグを解析し、それぞれの処理を実行する。  
  このコードを読み、プログラムの流れを理解すること。また、送信処理などが未実装となっているため各自で実装すること。  
  (TODO 03\_01～06まで)  
    
  **・クライアント側**毎フレームサーバへプレイヤー情報を送信する処理の完成。  
  また、サーバから受信したデータの処理についてコードを読み、プログラムの流れを理解すること。  
  ログアウト処理の送信など未実装となっている処理を実装すること。  
  (TODO 03\_07～10)  
  前回の課題と同じコードの部分はコピーしても構いません。

**プレイヤー操作**移動：AWSD or ←↑↓→  
攻撃：Ｘキー

テキスト

自動的に生成された説明

送られてきたデータのIDをもとにプレイヤー情報を更新する。

毎フレームプレイヤー情報(ID、position、angle、stateなど)を送信する。