**Design Manual**

**Full body motion tracking system**

**Group 20**

* Karunarthna D.A.D.T , E/16/173 , [e16173@eng.pdn.ac.lk](mailto:e16173@eng.pdn.ac.lk)
* Wijayasinghe S.D.D.D , E/16/394 , [e16394@eng.pdn.ac.lk](mailto:e16394@eng.pdn.ac.lk)
* Wijekoon W.M.T.S , E/16/396 , [e16396@eng.pdn.ac.lk](mailto:e16396@eng.pdn.ac.lk)

**GitHub repository link**

<https://github.com/cepdnaclk/e16-3yp-full-body-motion-tracking-system>

1. **Introduction**

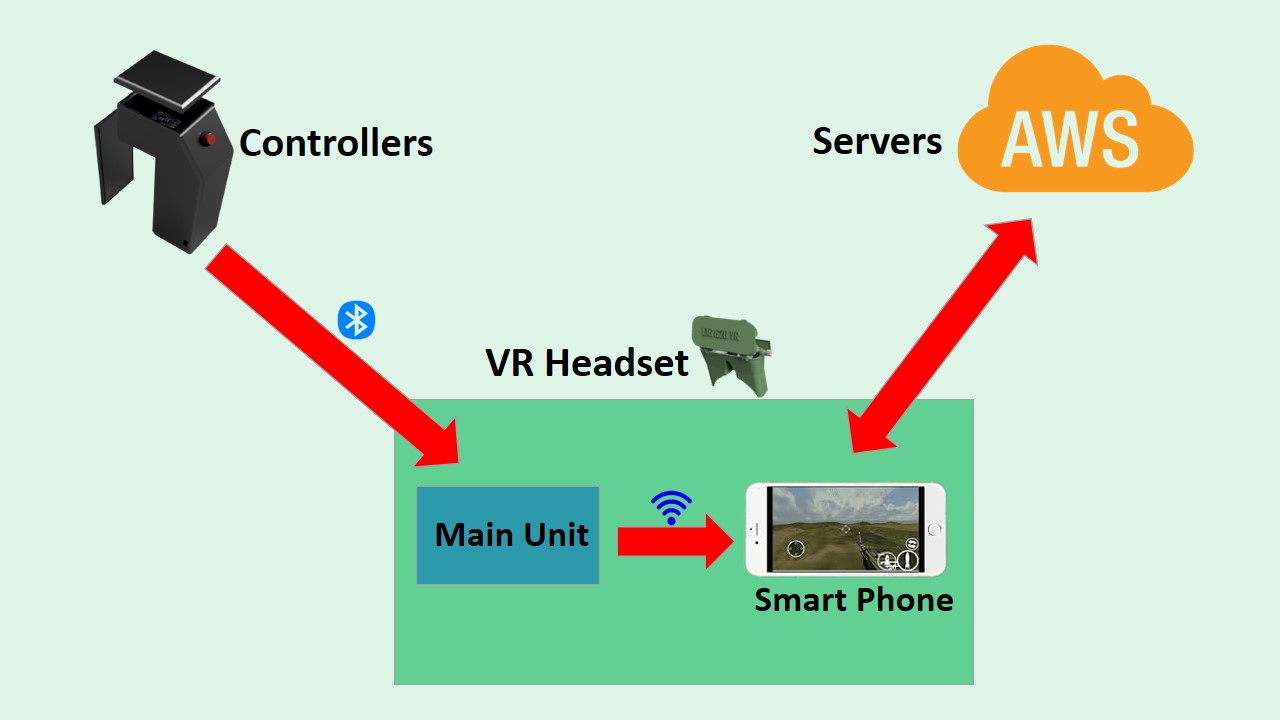
Full body motion tracking system is a multiplayer first-person shooter game based on body motion of the players.

The aim of this this product is to improve gamer's sense of freedom of movement and feelings of immersion within the virtual environment while playing together over a greater distance to achieve a common goal.

This consists of the following components.

* **Controller** is used to provide player inputs.
* **VR Headset** is used provide virtual reality for the wearer.
* **Main Unit** is used to identify player actions by analyzing inputs given by the controllers and this unit is included inside VR Headset.
* **Game**

1. **Hardware**

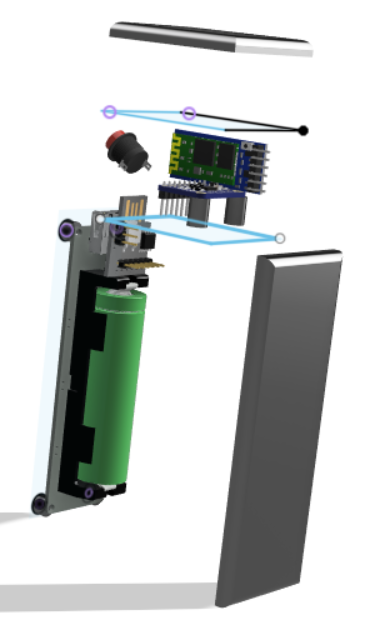
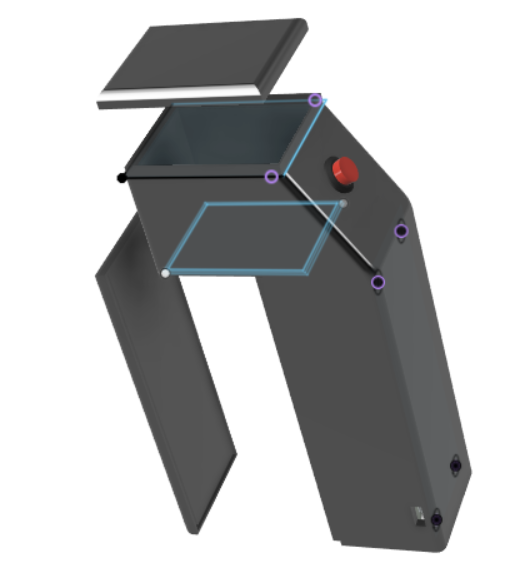
****

Design Architecture

As shown in the above figure, Controllers are used to provide inputs to the game using ADXL 345 sensor and push button. The data gathered by controllers are sent to the main unit which is included inside of the VR Headset, via Bluetooth.

Those data are analyzed in the main unit and the actions performed by the player are identified here. Then those actions are mapped into smart phone using ESP32S Wi-Fi module via Wi-Fi. The smart phone which is used to display game, is connected to AWS servers through internet.

**2.1 Controller**

****

* + 1. **Software**

A controller consists of following components

* ADXL345 Sensor
* ATTiny85V board
* HC05 Bluetooth module
* Push-button
* 18650 Li-Ion battery
* 18650 Battery Shield

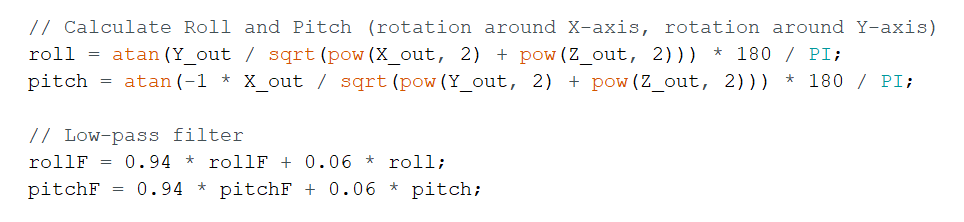
The CAD design of the controller can be found in our GitHub page.

In here ADxl345 is used to measure the acceleration resulting from motion and it is connected to ATTiny85V board. And also a push button is used in controller for providing fire input. HC05 module is used for sending those inputs to the main unit via Bluetooth.

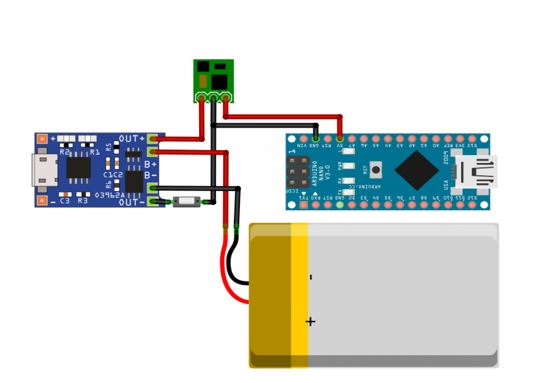
The controller is powered with 18650 Li-Ion battery with a 18650 battery shield.

**Sensor movement mapping**

ADXL345 sensor outputs x, y, z axis acceleration in g (m^2/s). Controller board process them and send roll and pitch values to main unit via bluetooth. (BT serial port protocol).



**2.2 LiPo Battery Charger**



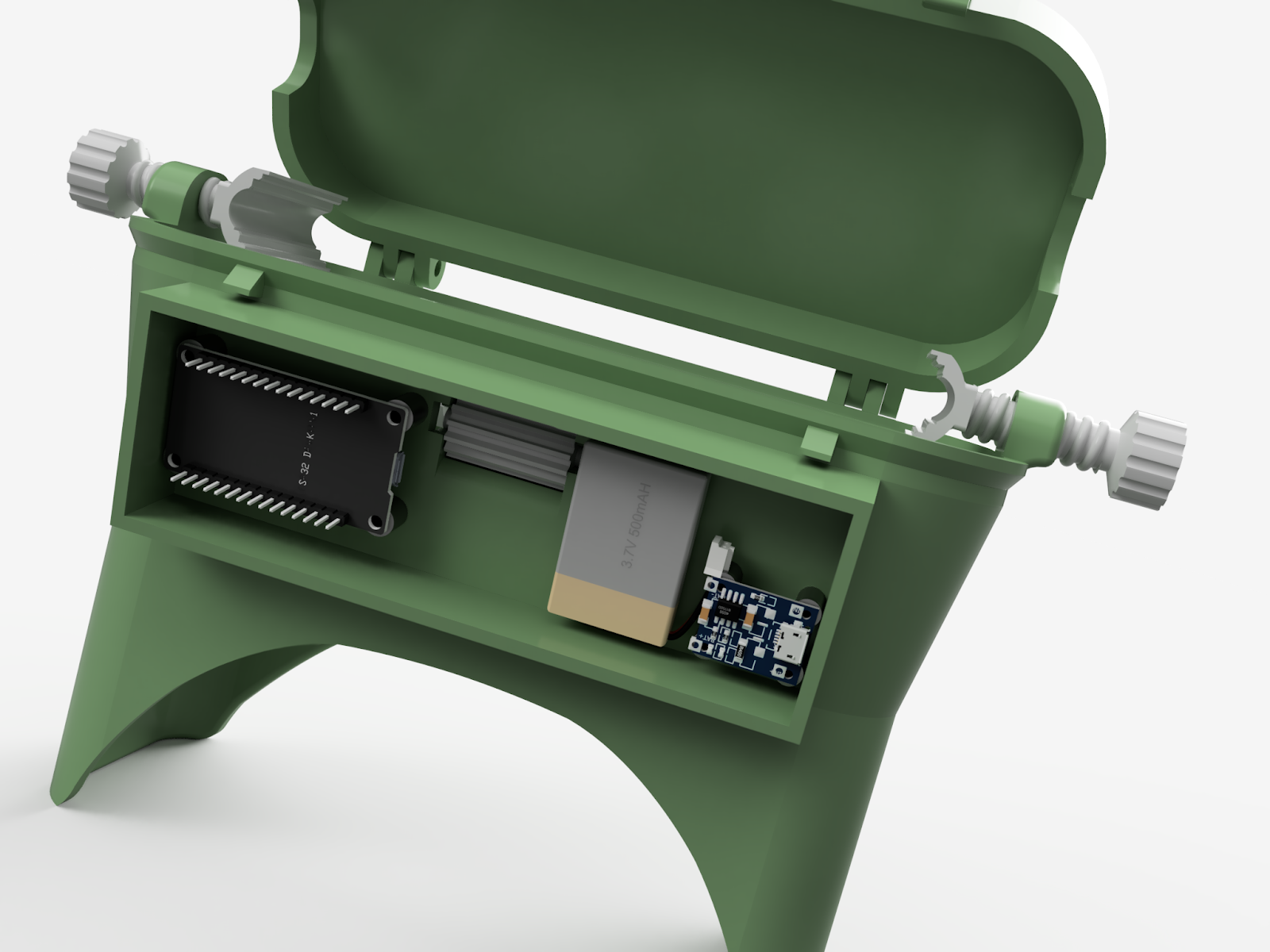
This module is designed for the main unit to power the ESP32S module.

This is modified with R3 = 1.6kΩ

**2.3 VR Headset**

The VR Headset consists of following components

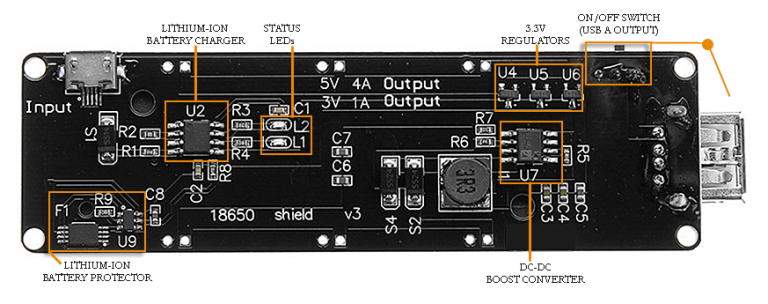
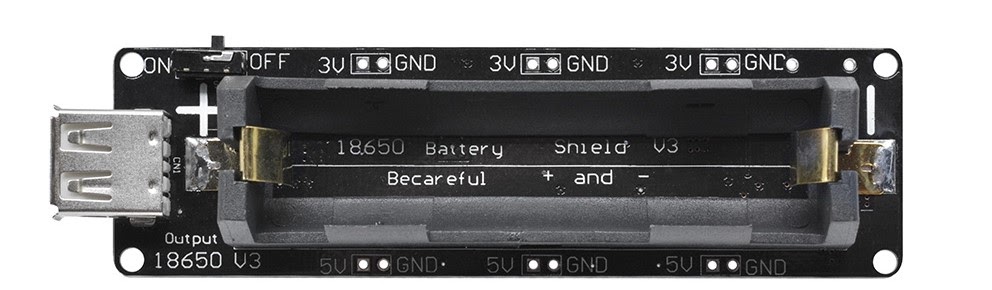
* ESP32S Wi-Fi/Bluetooth Module
* 650mAh LiPo battery
* TP4056 Charger module
* 5V Step-up voltage Regulator(not shown)
* VR lenses



The CAD design of the VR Headset can be found in our GitHub page.

For VR game play, Godot OpenVR plugin is used.

2.4 18650 Battery Shield

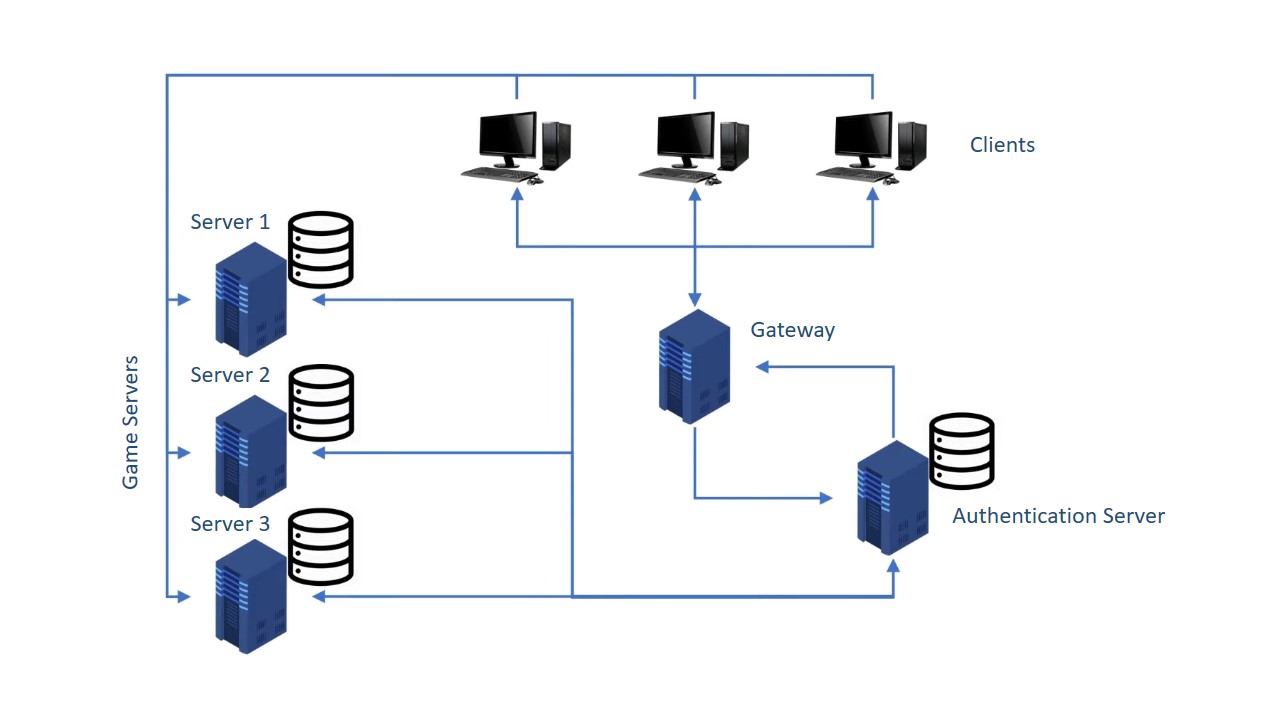
****

This can be used for charging, with a micro USB.

1. **Software**

The game in this project was implemented using Godot engine version 3.2.3.

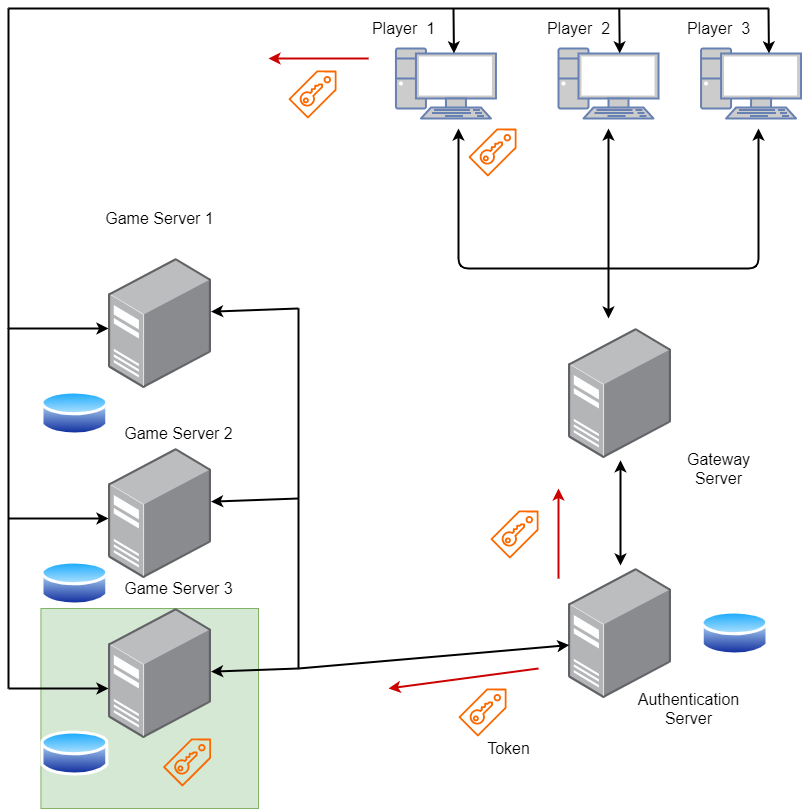
**3.1 Network Architecture.**



The above figure gives an overview of the overall game network architecture.

According to the figure clients (players) are first directed towards the gateway server and the gateway servers is then connected with authentication server for verifying player credentials.

On a successful authentication, authentication server issues a token and it is sent to the game server as well as to the relevant client. Therefore when the client connects to the game, the game serves can compare between the token of the client with the token given by the authentication server to authenticate the client.



Above graph shows how token sharing is happening between Authentication server and client, as well as between Authentication server and Game server.

**3.1 Game client implementation**

**3.1.0 DTLS Certification generation**

DTLS certificate can be used to protect players’ usernames and passwords of when the try to login to the game and this can be implemented using Godot game engine.

* To create a certificate you need to create a new project in Godot.
* Then create a normal node and give it a script.

You can refer to the DTLS\_Certification code under docs in our GitHub page.

* In that code as CN you need to add a string of issuer name (your server/ domain name).
* As O you need to add a string of your organization name.
* As C you need to add a string of ISO 3166 code of the country organization is based on.
* In addition you can add two variables as not\_before and not\_after to specify the validity period of the certificate
* After finishing the code once you play the project, certificate and the key will be generated and by default It will be stored at “C:\Users\User\AppData\Roaming\Godot\app\_userdata\X509Generator\Certificate”

Here the X509Generator is the name of the project

Next you need to add certificate file under Game -> Client\_v0 -> Resources -> Certificate

In addition you need to add both certificate and key file under Game -> **Gateway\_Server\_v0 -> Certificate**

**3.2 User data security**

In a multiplayer game more attention should be paid to securing user data. Following actions are taken for securing data of users.

* Usernames and passwords are only stored at the authentication server which is not directly accessible.
* Usernames and passwords are encrypted with sha256 encryption method.
* Usernames and passwords are sent between client-server after 1 round of encryption.
* Passwords are deleted as soon as sent to the server for verification.

**3.3 Anti cheating mechanism**

One of the main concerns in multiplayer gaming is the cheating methods. Therefore implementation of anti-cheating mechanism is a must.

As previously said, player data is stored only on the authentication server and player actions will be validated when an action is performed, only if valid world state on game server is updated.



Invalid Power → Kick Player

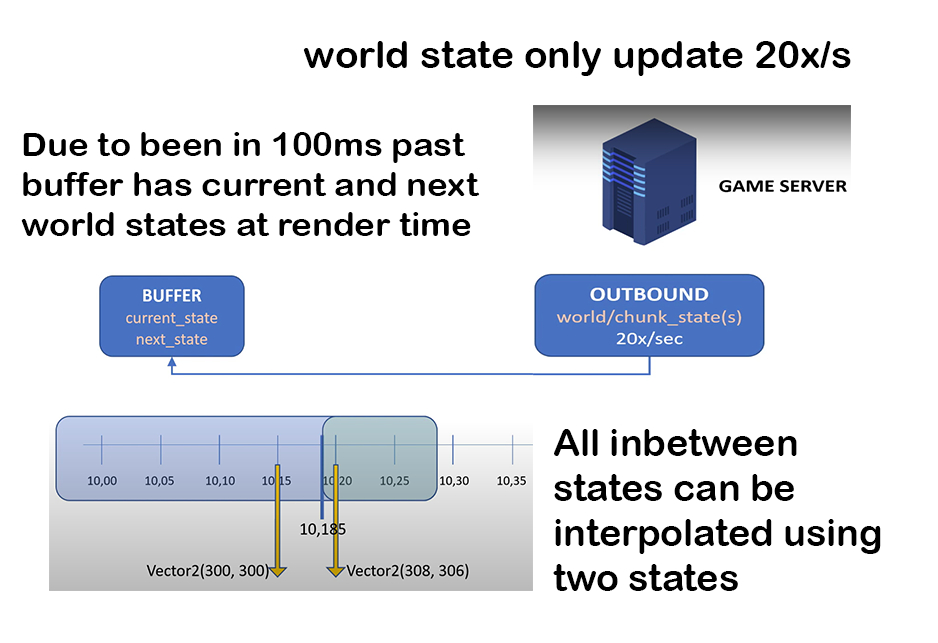


**3.4 Performance**

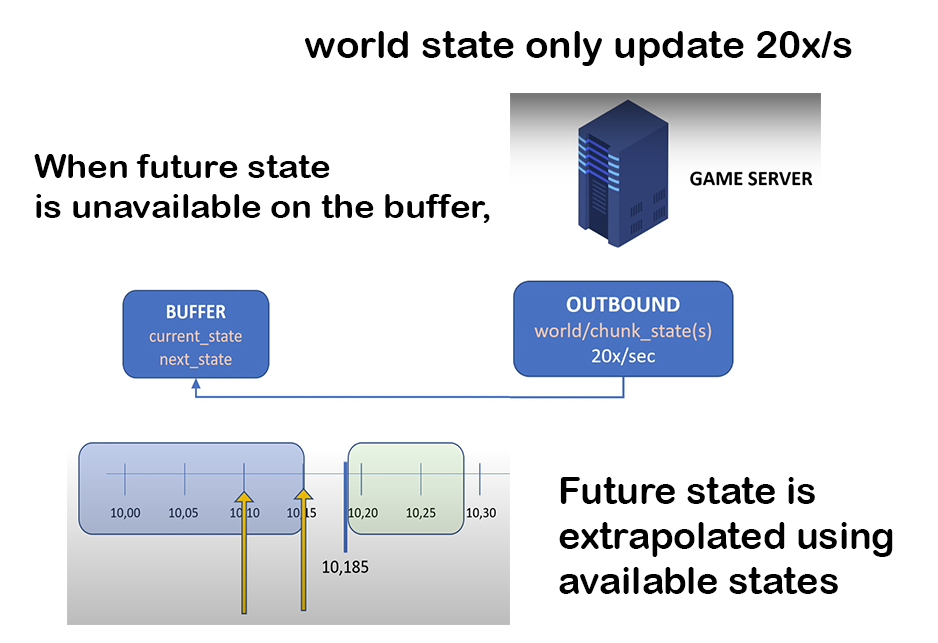
In a multiplayer game, performance is a key factor which defines the quality of the game.

Therefore the following actions are taken for a better performance.

* Even though game server update at a rate of 20x/s, client side world is rendered 60fps for a smooth experience for the user. For fixing that gap, interpolation technique is used.



* Extrapolation technique is used for predicting the future status for avoiding lagging issues.

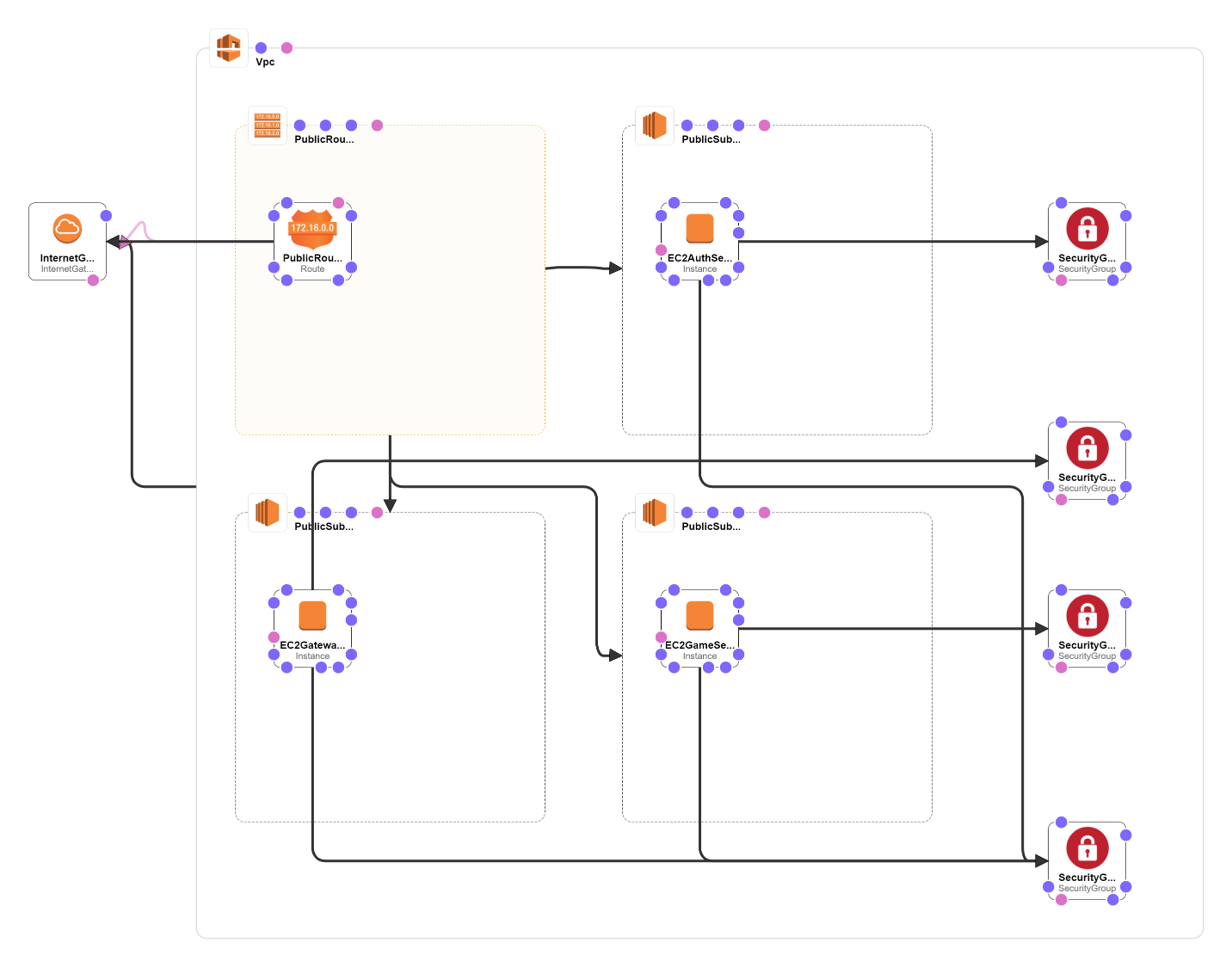


* And also player clock must be in sync with the server clock. Otherwise multiplayer game mechanics would not work properly. Therefore all client clocks are synced with server clocks in game.

**3.5 Servers implementation & running**

**3.3 AWS Cloud formation**

The following figure shows the cloud formation design of the infrastructure.



As shown in figure, gateway is used for connecting to the internet. And also the Amazon virtual private cloud (VPC) is divided into 3 sub nets.

1. Gateway server subnet – For gateway servers
2. Game server subnet – For game servers
3. Authentication server subnet – For authentication server

For each of these subnets, separate security groups are connected as well.

1. **Testing**

Under hardware testing, sensitivity of the sensor, stability and response lag are the main sections to look for.