Design and Development of a Real-Time 3v3 Multiplayer Shooter Game

[1]Divineson JG, [2]Aju VR, [3]Aljin Beni J, [4]Dinesh P, [5]Mrs. Jino Bani

[1,2,3,4]Student, Dept. of Computer Science and Engineering
[5]Project Guide, M.E. Department of AI & DS
[1,2,3,4,5]Good Shepherd College of Engineering and Technology, Tamil Nadu, India
[1]divinesonjg@gmail.com

***Abstract*—**Online multiplayer competitive games often face issues such as limited synchronization in gameplay physics, unforeseen latency, poor hit registration, and misuse of client memory. This is due to the decentralized systems that trust client applications, physics and reduce accountability. This paper proposes a server-centric networking system that provides real-time validation to detailed gameplay information, regular state updates, including vectors, and damage registration reports (Hitboxes). The system mainly focuses on state integrity through authoritative server handling like headless engine system, preventing unauthorized modifications even local clients can't able to change spatial details. It also enables servers to reject inputs about the movement and anyone can receive snapshots about the combat mechanics, improving accountability and server participation. And the prediction-based module using (CSP) Client-Side Prediction is integrated to simplify highly networked rendering of physics and provide access to gameplay data for high-latency players. The prototype is developed using Unity3D and Netcode framework, demonstrating improved synchronization and demonstrating efficient cheat prevention. Although currently at a prototype stage, the system provides a scalable foundation for authoritative networking.

***Index Terms*—**Synchronization, E-Sports, Public Networking, Headless Server, Client-Side Prediction, Server Participation, Strict Authoritative Systems.

I. Introduction

Online competitive multiplayer games are basically the backbone of digital entertainment and e-sports things like shooters, arenas, sports, simulations, combat platforms and other essential genres players rely on every day. Since these projects are mostly funded through monetization models by players of the community, there's a pretty strong need to make sure that networks are used fairly and efficiently. That said, the reality isn't always ideal. Many of these projects run into common issues lack of synchronization, weak hit detection, delays, and sometimes even misuse of memory. A big reason for this is that there isn't really a system where the server can easily see what's happening. So, developers who are actually hosting these matches don't get much visibility into how their physics are being tracked or not. Traditional networking methods don't help much either. They're usually limited to decentralized peer-to-peer processes, with little to no direct involvement from central servers. Lately, there's been a clear push toward using headless tools to close this gap.

The idea is to make sure the physics information more visible in real time and give servers authority and a way to actually participate in networking. That's where this proposed system comes in. The proposed system is designed to improve synchronization via exposing coordinate details and hit monitoring by giving servers access to detailed information about both ongoing and completed combat matches. This includes things like spatial vectors, hitbox details, movement updated by client and renderer, and official damage reports.

But it doesn't stop at this point, servers can also reject inputs found by their own from the validation layer, share snapshots, and rollback positions, which makes the whole process more authoritative. From a technical side, the system is built using Unity3D for the frontend development and CSharp's Netcode framework for the backend development. This combination keeps it simple and straight to use, with rendering friendly interface while still being scalable and efficient in handling physics. Overall, the goal here is to move away from the usual peer-to-peer networking approach and shift toward something stricter and more authoritative.

II. Related Works

Recent research in multiplayer game networking keeps highlighting a familiar set of issues all over the world -lack of synchronization, inefficiencies, cheating risks, and very limited server participation in peer-to-peer networks. Traditional matchmaking and monitoring systems don't really solve these problems either. They're usually decentralized all the time, which makes them more vulnerable to data manipulation and delays because of this, quite a few studies have looked into using emerging technologies to tackle these limitations and improve accountability in multiplayer operations. Headless servers, for instance, have been widely explored as a way to improve synchronization and trust in gaming systems. The idea is pretty simple: it uses a centralized and immutable data structure, meaning once something is calculated, it can't just be changed unless there's consensus. This makes auditing and tracking much more reliable.

Research shows that there exists a chance to reduce fraud and cheating in combat networking processes [1], [2]. In particular, headless-based frameworks have shown promise in making rendering more transparent, keeping records safe and secure while ensuring fair evaluation of inputs.

Centralized networking models used to take things a step further by removing dependence on a single client authority. Instead, data is calculated across dedicated nodes. This setup improves reliability and avoids single points of failure. Studies suggest that such systems can strengthen trust between players and game developers, while also enabling real-time monitoring of match activities [3], [4]. On top of that, using centralized server technologies helps with secure data sharing and smoother communication among different endpoints. Another key development in this area is the use of prediction algorithms to automate movement workflows.

These algorithms basically run on a set of predefined rules and execute on their own once certain conditions are met, which shows that there will be only minimized amount of visual lag. Studies suggest that this kind of automation boosts efficiency, cuts down on rendering errors, and keeps processes consistent whether it's hit evaluation, physics execution, or packet handling [5], [6]. It also helps reduce network workload and speeds up overall game operations.

Even with all these improvements, there's a clear gap. Most existing systems are heavily focused on backend features like logic and automation, but they don't really address direct renderer interaction or real-time lag compensation. In a lot of cases, there isn't an easy or latency-friendly way for the player to actually engage with these platforms like any any other offline game, that's exactly where the proposed system fits in. It builds on these existing ideas but shifts the focus more toward the rendering client. Along with synchronization and automation, it brings in a prediction-based interpolation module to make the platform smoother and more accessible for all type of player even high-ping players can access game states. The goal is to combine synchronization, efficiency, and server participation into something that feels more practical and usable in real-world scenarios.

III. Proposed System

To address the limitations of traditional peer-to-peer matchmaking systems, a synchronized and authoritative networking platform is proposed. The system is designed as a server-centric digital solution that enables strict access to physics data like hitboxes, vector details, real-time tracking, and active server validation. Unlike traditional decentralized systems, this proposed system approach combat accountability, emphasizes synchronization, and efficient dataflow among components. The system has architecture roles: Server, Network, Physics, and Local Client. Each role has its responsibilities. Match-related information such as vectors, coordinate updates, assigned endpoints, and damage reports is made accessible to servers, ensuring visibility at every stage of the gameplay lifecycle. Clients are required to submit continuous input updates, which are maintained as predictive records to preserve data integrity and prevent unauthorized modifications which was the main goal. This approach aligns with existing research that highlights the importance of immutable and auditable physics in improving synchronization and reducing cheating in gaming sector systems [1], [2]. And the key feature of the proposed system is the integration of a server-side and validation-reporting mechanism. Servers can track combat-related issues or malpractices, which are made visible to all endpoints. Clients are required to render with authoritative updates or corrective rollbacks, creating synchronized feedback about reported vectors. This mechanism enhances accountability and ensures that match activities are continuously monitored, addressing limitations identified in prior studies that lack server engagement and real-time validation [3].

In addition, the system can handle prediction-based logic with dedicated interpolation that allows clients to render with physics data using smooth transitions. This feature improves visuality for players to understand state details even if they are high-ping players and simplifies the process of receiving snapshots. The integration of predictive interaction mechanisms extends beyond traditional networking systems, which primarily focus on server-side logic without providing latency-friendly renderers. The proposed system can helps manage data through a way of storing and retrieving it from a datastore. It can also ensure memory management through structured variable storage and retrieval mechanisms. All the gameplay activities such as updates, math logic, and damage are recorded for traceability and prediction reference. This platform further helps in synchronization among the components through the means of transmission and updates provisioned through the same. This logic-based implementation ensures that only authoritative servers can perform specific inputs, thereby maintaining system integrity. Moreover, clients are also restricted; such actions can only be done by a central-server. These features all work to create a solution that is reliable and can scale for orchestrating multiplayer matches.

A. System Overview

The proposed platform is aimed at orchestrating multiplayer matches as well; however, it differs from traditional platforms due to its focus on synchronization, validation, and server participation. In contrast to traditional systems that provide trusted calculations only for local clients, the proposed platform is focused on the authority of this information: servers will be able to obtain data about the match and participate in the validation process. Thus, the platform provides several system tiers: servers, physics, and clients. These tiers interact in accordance with their roles: for example, local clients will be allowed to render information about gameplay details, predict its movement, and provide inputs about possible vectors; physics will have to calculate the status of the match according to the provided vectors; servers will have control over packet data.

Furthermore, it is expected that all communication will take place in terms of constant data flow between clients and server authorities. It means that there will be an automatic creation of snapshots in case of state updates and inputs and that clients will be able to communicate with server authorities about positional problems and receive adequate rollbacks from them. Finally, one of the important innovations of the proposed system is connected with the introduction of the predictive interpolation feature, which will facilitate renderer interaction with the platform since clients will be able to access the necessary frames without any stutter, as if playing a local offline game.

As a result, the system departs from the traditional approach to the networking of multiplayer projects in terms of focusing on the synchronization of packet exchange, structuring, and server verification. In other words, this platform provides an efficient framework for orchestrating matches.

B. System Architecture

The proposed system's architecture follows a client and server model, integrating a lightweight renderer using Unity3D with a scalable headless to ensure efficient performance and security. The renderer is developed using Unity3D, providing an interactive and user-friendly visual for different categories of player. It enables functionalities such as player authentication logging function, 3D visualization mechanics, input submission module, and interaction with the prediction module. This smoothness of the interface ensures accessibility even for common high-ping players, supporting broader gameplay participation and let them to be a part of the match.

The headless is designed using Netcode framework, which handles UDP requests, physics logic, data processing and all other states. The system makes sure that it works well and it helps the different parts of the system talk to each other in real time so they can do certain things. The headless of the system talks to the datastore to store state information and get it back when it is needed which helps keep everything synchronized and this is very important, for the physics data. A lightweight prototype database is used since it's still under prototype phase to manage structured arrays, including player details, state updates, match information, and collision reports. This design can be extended to scalable cloud systems for real-world deployment.

The player interacts with the system through the renderer and the renderer sends inputs to the headless UDPs. The headless processes these requests, this way of doing things makes sure that physics is handled in a way and that everything stays up to date in real time. The system also has a prediction module. This prediction module looks at what the client's pressing for and generates frames that make sense based on the state data.

The prediction module makes the game easier to play. It lets the client interact with it in a smooth way and makes it easy to get to the positions they need in their own framerate using CSP. This component enhances visuality and represents an advancement over traditional networks by enabling seamless interaction and lag-compensated data access.

C. Methodology

To achieve maximum efficiency when processing and analyzing the available data, as well as providing synchronized interaction between the clients and system components, the methodology should be used. According to the selected methodology, the client's actions and inputs are performed using specific UDPs. This means that each packet is handled in the same way.

First of all, the player should undergo the stage of authentication and once if the connection granted client can access specific servers. After that, the client will be able to interact with the rendering module and other elements of the engine. In particular, servers will have access to tracking state records and calculating physics, while clients will update information about input status from time to time. Public players will be allowed to view visual information and send inputs or commands concerning the ongoing gameplay. When performing certain actions, clients interact with the system by contacting the corresponding modules. All interactions occur in a controlled way through headless services, meaning that each request should meet strict math criteria. Logic-based access control system is employed in order to limit the range of possible exploits for each client.

Data transmitted by clients in terms of their vectors and inputs are systematically recorded in the appropriate arrays in order to guarantee their traceability and predictability. At the same time, the process of calculating out and predicting the frames is followed by a state broadcasting cycle during which the math is validated and authorized by the headless servers. In total, the use of this methodology allows for effective operation of all engine components and smooth synchronization between clients.

D. Implementation

To prove the feasibility of the proposed networking system, a functional prototype has been developed. In terms of the system architecture, this prototype incorporates several key elements: renderer, headless, and datastore.

The renderer is the interface provided to clients allowing them to utilize some basic functions of the engine, such as movement and viewing, visualization of projectiles, making an input and communicating with different modules of the engine. The headless is designed to process requests from the clients, execute various physics based on the math defined for a particular state and communicate with the datastore.

As noted above, the system is divided into several modules, each of them performs different calculations and is associated with one another. Thus, the damage module enables servers to make a hitbox about a particular collision from the physics raycast, while the headless records this hitbox and ties it to the relevant match along with client ID and tickrate. All types of data generated in the system (related to clients, matches, physics updates, and hitboxes) are stored in a structured array. The choice of such solution is justified by the ability to store vectors in an efficient and consistent way and provide easy-to-use access to it for different parts of the engine.

It should be noted that the current prototype involves the use of a lightweight memory sufficient for prototyping but not adequate to support the engine in its final version. However, the design of the prototype makes it possible to scale the engine easily and integrate any type of datastore to make it more stable and reliable. Different kinds of tests have been applied to check the networking of the system. They included unit testing, integration testing, and client-level testing.

IV. PREDICTION MODULE DESIGN

The proposed system consisted of prediction-based module to enhance visual interaction to the gameplay rendering details and improve smoothness to other physics-related information. The primary objective of this prediction module is to make clients to interact with the system's states using local frames, eliminating the need for waiting knowledge or complex lag through network ping or any other transport delay. This module utilizes CSP techniques in processing client inputs and provides a visual based on the state data available in the framerate being rendered.

Clients can press any kind movement or combat buttons about the character it can be running, shooting updates, aiming, and jumping mechanics in a seamless manner just like any other offline game. The system processes client inputs and extracts key vectors from the input to retrieve appropriate visuals from the engine about the character to provide smooth animations using the state provided about the physics.

One of the key advantages of the prediction module is its ability to simplify delayed network data into easily renderable visuals to players. This makes it easier for people to play those who are high-ping players like the general internet. The game is also easy to control with which means more players can participate and see what is going on with the combat. This prediction module also helps clients get positions right away so they do not have to wait for the server to confirm the information. This reduces dependency on manual server waiting and allows clients to quickly render critical frames.

The prediction module is still in the prototype-level CSP system stages but it shows that we can use interpolation technology to make networking platforms better. In the future we might add features, such, may include the use of machine learning models,

server-side rollback, and lag-compensation interaction and letting servers rewind the state to make it even fairer for players to hit targets and to make the hit-registration work even better.

V. Results and Discussion

The proposed system was developed as a prototype using Unity3D for the frontend and Netcode for the backend, tested the system with some sample clients from a multiplayer session to see how well it works. Also, effectiveness in improving synchronization, security, and client interaction. The results obtained from the implementation highlight several improvements over traditional peer-to-peer matchmaking systems. One key outcome of the system is that it makes data more centralized. All physics information like velocities, coordinate updates and damage reports is available to servers based on their authority. Unlike conventional systems where data is trusted to local clients, this platform enables headless servers to validate and calculate match activities in real time. This aligns with prior research that emphasizes the role of synchronized and authoritative systems in reducing hacking and improving fairness in gaming sector operations [1], [2], [7].

Another important result is the improvement in state integrity and validation. The system makes sure that once spatial updates are validated, they cannot be spoofed. This keeps the math permanent and unalterable. It reduces the risk of memory being altered or injected. This also makes it easier to track and review the hitboxes. Similar findings have been discussed in centralized-based networking models, where authoritative records play a role in ensuring reliable hit-registration.

The implementation of the prediction and input-sanitization mechanism in our project significantly improves client participation and provide expected performance like reporting the pressed buttons from the local keyboard. Clients can report inputs related to movement vectors or shooting triggers, and these packets are visible to the server. While the servers are required to respond with verified snapshots or corrective rollbacks to clients, creating a synchronized feedback loop. This directly addresses the gap identified in existing systems, where server involvement is minimal or absent [8], [3]. In terms of network efficiency, the integration of structured packet management and logic-based access control (LBAC) ensures smooth transmission among endpoints. Each client is permitted with specific bandwidths; it helps reduce unauthorized spoofing and improving network security. The use of automated calculations and organized state storage also contributes to better lag compensation and management, which is consistent with studies on prediction-based interpolation in public gaming systems [5], [6].

The prediction-based interpolation module makes the game easier to play. Players can control the character in their frames to get visual about the physics status, projectiles or reported hits. The prediction-based visual module is really helpful for clients because they can use rendering to hide latency and get smoothness, about spatial status, combat or reported movement. And this reduces dependency on server tickrates and makes the game more playable via CSP, especially for high-ping players.

Apart from these positive outcomes, some limitations were also observed. The system currently relies on a prototype-level datastore (Memory), which may not be suitable for large-scale production level deployment. Additionally, the prediction module is still in its early stages and may require further training and optimization for more accurate and latency-aware responses however we are using CSP to provide physics details as context to the rendering module. Future improvements may include the integration of advanced anti-cheat frameworks to increase the immutability, scalable cloud databases, and more sophisticated prediction models. The results show that the proposed system is a way to deal with the problems of traditional peer-to-peer matchmaking systems. The system combines being authoritative using validation and letting clients render with it. The proposed system provides a solution that can work for many players and can be used to oversee multiplayer matchmaking in a modern way. The proposed system is a solution, for multiplayer matchmaking environments because it uses synchronization, validation and client interpolation.

VI. Limitations

The proposed system is currently developed as a functional prototype and has certain limitations that can be addressed and developed as production level application in future work. The system uses a local instance using memory arrays, which may not be suitable for large-scale deployment involving high volumes of matches and concurrent players. This makes it easier for people to play those who are high-ping players like the general internet. The game is also easy to control with which means more players can participate and see what is going on with the combat. This prediction module also helps clients get positions right away so they do not have to wait for the server to confirm the information. This reduces dependency on manual server waiting and allows clients to quickly render critical frames.

The prediction module is still in the prototype-level CSP system stages but it shows that we can use interpolation technology to make networking platforms better. In the future we might add features, such, may include the use of machine learning models, server-side rollback, and lag-compensation interaction and letting servers rewind the state to make it even fairer for players to hit targets and to make the hit-registration work even better.

VII. Conclusion and Future Work

This paper presented a synchronized and authoritative networking system for multiplayer competitive games, this proposed system was primarily designed to overcome challenges such as lack of synchronization, hacked physics, cheat concerns, and limited server participation. The proposed system introduces a server-centric platform that enables real-time access to combat data, prediction module to get smoothness about rendering, structured math, and an interactive validation mechanism. By incorporating logic-based access control and immutable math handling concept in order to protect engine details, the system improves fairness and ensures that match activities remain tamper-proof and traceable.

The contribution of prediction-based interpolation module allows clients to render gameplay information using local frames where players can use their own monitors. This enhances visuality and makes the game accessible to high-ping players Since most raw server data is only updated at low tickrates. The prototype implementation demonstrates the feasibility of combining authoritative mechanisms with predictive interaction to improve matchmaking systems. Future work will focus on enhancing server scalability through cloud-based deployment, creating a high security mechanism via Anti-Cheat validation and Tracks client

connection and packet activities with precise timestamps, and integrating advanced machine learning models for better aimbot detection.

Furthermore, the elimination of host advantage creates a genuinely level playing field that is absolutely critical for competitive tournaments. By completely removing the client's ability to dictate match outcomes, developers can maintain the integrity of their digital economies and global leaderboards. This architectural shift significantly reduces the administrative burden of manually reviewing reported cheaters and verifying combat logs. Ultimately, adopting such strict server-authoritative frameworks will foster healthier, more sustainable gaming communities. Overall, the proposed system provides a strong foundation for modernizing digital gaming, synchronization, and setting a new technical benchmark for the rapidly expanding e-sports industry.

REFERENCES

[1] G. Fiedler, J. Glazer, S. Madhav, and A. Chen, "Networked Physics in Multiplayer Games: A Systematic Literature Review," IEEE Access, vol. 9, pp. 13904-13921, 2021.

[2] J. Peterson, K. Williams, O. Haass, T. Maqsood, and I. Ahmed, "Headless Technology: Potential Applications for Public Sector E-Sports and Project Management," Sustainability, 2022.

[3] M. Janssen, S. A. Chun, and V. Weerakkody, "Server-Authoritative Technology as Infrastructure in Public Sector," ACM, 2018.

[4] M. Xu, X. Chen, and G. Kou, "A Systematic Review of Headless Architectures," Financial Innovation, 2019.

[5] Y.-H. Chen, S.-H. Chen, and L.-C. Lin, "Server-Based Smart Netcode for Matchmaking System," IEEE, 2018.

[6] F. S. Hardwick, R. N. Akram, and K. Markantonakis, "Transparent Server-Based Networking Framework," arXiv, 2018.

[7] E. Kademeteme and S. Bvuma, "Using Client-Side Prediction for Networked Games in the Public Sector," IEEE, 2019.

[8] A. Chen, M. Rossi, and J. Gregory, "Latency Compensation and Prediction Techniques in Fast-Paced Multiplayer Shooters," IEEE Access, 2022.