Use of Server Mesh in the Context of Games

*Miguel Ferreira (1230199)*

*Instituto Superior de Engenharia do Porto, Portugal*

***Abstract*— The current architecture of online multiplayer games faces significant limitations regarding seamless data transfer, often forcing players to endure loading screens between transitions and server lag. Additionally, issues such as limited server capacity and poor scalability in distributed systems hinder the overall gaming experience. Addressing these challenges has become a focal point of technological innovation in the gaming industry, with several companies competing to develop solutions. One of the most promising approaches is server meshing, which enables dynamic load distribution and horizontal scalability across multiple servers. Cloud Imperium Games (CIG) is at the forefront of this innovation, demonstrating the potential of server meshing to revolutionize gaming. By enabling near-limitless scalability and reducing server load bottlenecks, this technology represents a significant leap toward creating vast, persistent game worlds with seamless player experiences. This paper explores the technological foundations of server meshing, its potential impact on game design, and its implications for the future of distributed systems in the gaming industry.**

***Index Terms*—Distributed systems, Dynamic server loading, Server mesh, Static server mesh, Dynamic Server Mesh, Static Server mesh, Start Citizen, Ashes of Creation, Usefulness.**

# I. INTRODUCTION

The exponential growth of online multiplayer games has exposed fundamental limitations in traditional server architectures, particularly in their ability to support massive concurrent player interactions within persistent virtual worlds. Contemporary multiplayer games face significant technical constraints in managing seamless data transfer, server capacity, and distributed system scalability, often resulting in compromised player experiences through loading screens, server segmentation, and performance degradation under high load conditions.

Server meshing emerges as a transformative solution to these architectural challenges, representing a paradigm shift in how game worlds are hosted and managed across distributed systems. This technology enables dynamic load distribution and horizontal scalability across multiple servers, effectively creating a unified, coherent game world without traditional server-imposed boundaries. Companies like Cloud Imperium Games (CIG) and Intrepid Studios are pioneering the implementation of server meshing technology, demonstrating its potential to revolutionize multiplayer game architecture.

.

## A. Abbreviations and Acronyms

CIG: Cloud Imperium Games

MMO: Massively Multiplayer Online

TiDi: Time Dilation

NPC: Non-Player Character

AWS: Amazon Web Services

AI: Artificial Intelligence

# II. Current Distributed software architecture

The current scenery of distributed systems offers solutions that are highly scalable and reliable, allowing for mass exchange of data, tools like Kubernetes allow for easy horizontal scalability and as distributed architectures become more prevalent it is a must-use tool[1], there are already corroborated solutions for making systems fault tolerant like using redundancy between servers[2].

## A. Differences

Even if modern software architecture has been improving, even being able to work with millions of data, our games server architecture still has problems in proving these capabilities. The main problem is that games are full of state, and state management is still one of the biggest hurdles to solve, this problem is accentuated on game servers due to its innate need for high state management [3]. As such, we have seen these kinds of limitations in display, for example, with Eve Online when the biggest online battle happened [4], it was needed to reinforce the servers and use a special technology created by the formers, called TiDi[5]which slows down game time so servers do not crash.

# III. Server Meshing: A Technological Overview

Server meshing is the process of dynamically splitting the load of a game world across multiple servers, effectively treating them as a unified entity. This contrasts with the more traditional sharded server model, like the mega servers of Guild Wars 2 [6], where game worlds are divided into static instances. In server meshing, a game world can be hosted across multiple servers that coordinate dynamically based on real-time load and player activity.

Two primary types of server meshing have been proposed:

1. **Static Server Mesh**: In this configuration, the game world is partitioned into predefined areas, with each area assigned to a specific server. Players seamlessly transition between these areas without needing to experience loading screens. However, the static mesh does not allow for dynamic reallocation of server resources [7].
2. **Dynamic Server Mesh**: This more advanced configuration allows for real-time load distribution. The game world is not divided into static zones but rather allocated based on player density and server capacity. Servers can dynamically adjust to handle more players in high-traffic areas or offload areas with fewer players, optimising performance and scalability[7] .

## A. Load Balancing in Server Meshing

Load balancing is a pivotal component of server meshing,

ensuring that computational workloads are efficiently

distributed across multiple servers. This optimizes resource

usage, prevents overloads on specific servers, and ensures

seamless player experiences, especially in large-scale

online games where player density fluctuates. Various

algorithms are employed to manage load distribution, each

with distinct advantages and trade-offs, in the following cases, it is shown 3 different static Load balancing algorithms:

1. **Round-Robin Load Balancing** is a Static load balancing algorithm that distributes incoming requests across servers in a cyclic manner. While simple and effective in ensuring an even distribution, it fails to consider individual servers' capacity or specific task requirements[8], [9], [10], [11].

* *Advantages*: Simple implementation, even workload distribution.
* *Drawbacks*: Ignores real-time server load and capacity, potentially leading to inefficiencies in dynamic environments like MMOs.

1. **IP Hash** uses the client's IP address to determine which server should handle the request. It applies a hash function to the source IP address, then maps the result to one of the available servers [10], [11]. This ensures that the same client (with the same IP) is consistently directed to the same server, which can be useful for maintaining session persistence in online games.

* *Advantages*: Ensures client requests are consistently directed to the same server, helping with session management and reducing overhead.
* *Drawbacks*: Doesn’t account for load distribution across servers, so some servers could become overloaded if many clients share similar IP hash results.

1. **Weighted Round-Robin Load Balancing** is an enhancement of the round-robin method, this approach assigns weights to servers based on their processing capabilities[10], [11]. More powerful servers handle a larger share of the workload, making it particularly useful in server meshing, where different regions may have varying player densities.

* *Advantages*: Considers server capacity, improving load distribution in heterogeneous environments.
* *Drawbacks*: Requires continuous monitoring and updates to server weights, adding complexity to the system.

There are also dynamic load balancing algorithms, which might be used in the implementations from the different companies that want to use server meshing, they can continually evaluate servers load and distribute tasks in real-time to adapt to changing conditions. This is critical for an MMO environment, where server loads can fluctuate due to player activity, world events and interactions, the following algorithms are the most common:

* 1. **Weighted Least Connection** is an algorithm that distributes incoming tasks based on the number of active connections to each server and factoring in server capacities (weight)[11]. Servers with fewer active connections relative to their weight receive more traffic. For example, more powerful servers can be assigned a higher weight, allowing them to handle more players or tasks than weaker servers.
* ***Advantages*:** Considers both the current load and the capability of each server, ensuring efficient distribution.
* ***Drawbacks*:** The weighting system must be carefully tuned to prevent overload on high-capacity servers.
  1. **Resource-based Algorithm** distributes tasks based on the available system resources, such as CPU, memory, and network bandwidth, on each server. It dynamically monitors the resource utilization of each server and sends tasks to the server with the most available resources[11]. This approach is particularly effective in MMOs where resource usage can vary significantly depending on player activity or in-game events.
* ***Advantages*:** Ensures that servers with more available resources take on more tasks, optimizing server performance and avoiding overloading.
* ***Drawbacks*:** Requires continuous monitoring of multiple system metrics, adding complexity to the system.
  1. **Weighted Response Time** allows to direct tasks to the server with the fastest response time, while also considering a weight assigned to each server’s capability. The Response time is a good indicator of how busy a server is, and weighting ensures that more powerful servers can still take on additional load, even if their response times are slightly longer[11]. This is particularly effective in MMOs for maintaining player experience during peak activity.
* ***Advantages*:** Balances real-time performance with server capabilities, optimizing both response time and resource allocation.
* ***Drawbacks*:** Monitoring response times and weights adds complexity, and the system must be continuously adjusted as server performance changes.

**Implications for Server Meshing**

In server meshing, where multiple servers manage different regions of a game world, load balancing algorithms are crucial for ensuring smooth transitions, real-time responsiveness, and optimal resource allocation. The choice of load-balancing strategy depends on the game’s architecture, player activity, and hardware constraints. Dynamic approaches should be preferred, as they offer real-time adaptability and handle fluctuating loads without disrupting gameplay.

## B. Server Meshing in Depth

It is important to note that discussing server meshing in detail is challenging, as only a few companies have publicly disclosed information about this technology, and none of them have fully implemented it for public scrutiny. Consequently, this section will hypothesize how the system might work based on available information, followed by a comparison with traditional software implementations to assess the plausibility of these claims.

Server meshing involves the complex orchestration of multiple servers that handle different parts of a virtual game world. The innovation here is how it allocates resources dynamically, meaning the system adjusts the workload in real time based on player activity and server conditions. In traditional server architectures, game worlds are divided into distinct, isolated instances or “shards”, with each server hosting a specific area [6], in contrast, dynamic server meshing treats the game as a continuous whole, meaning that the world is divided into smaller zones or regions, which are then allocated to different servers. In static server meshes, these zones are predetermined and fixed[7], [12]. However, in dynamic server meshes, the system continuously monitors the load in each zone and dynamically reassigns them to different servers based on real-time needs. This could involve splitting a heavily populated zone across multiple servers if the player count increases, this is the partitioning part.

The core feature of the dynamic server meshing which is its ability to adjust server allocations based on real-time player density, load, and activity could be defined in the following steps:

* **Monitoring**: The system consistently monitors metrics like player count, NPC activity, bandwidth usage and server performance.
* **Load Distribution**: When a region becomes overloaded due to a high number of players, the server meshing system splits the workload, and this can be done in two ways:
  1. **Dynamic Zone Creation**: The system divides the game world dynamically by creating smaller zones within the crowded area.
  2. **Parallel Server allocation**: Multiple servers can work together to handle the same region, parallelizing tasks like physics calculation, NPC movements, and player interactions.

Managing complex interactions implies the need to have a specialized orchestration middleware capable of:

* Deciding which servers handle which areas.
* Implementing mechanics for fault tolerance in case a server fails
* Handling latency-sensitive tasks like collision detection or other game components between servers

Finally, this system must be capable of elastic scalability, allowing it to scale up or down smoothly depending on player demand.

## C. Benefits of Server Meshing

1. **Horizontal Scalability**: Server meshing enables scaling horizontally by adding more servers to the network. This approach contrasts with traditional vertical scaling, where upgrading server hardware becomes cost-prohibitive as player numbers grow.[13]
2. **Seamless Player Experience**: By reducing or eliminating loading screens between server transitions, server meshing allows for a more immersive experience. Players can traverse vast game worlds without interruptions, making the game more persistent and alive[14].
3. **Reduced Bottlenecks**: Dynamic server meshing addresses the issue of overcrowding in high-traffic areas by allocating more servers to those areas, reducing lag and enhancing performance [13], [14].

# IV. Where Is It Being Used

There are two major games where this technology is being developed, the two being:

1. **Ashes Of Creation**
2. **Star Citizen**

Though their implementations differ slightly, both aim to optimize load distribution and create seamless player experiences across vast game worlds

## A. Ashes Of Creation

In Ashes of Creation, server meshing ensures seamless transitions between servers without loading screens. Key features include:

* **Authority Representation**: Each player is managed by an authoritative server. When moving between servers, the player’s state is transferred without creating new instances, reducing resource costs.[14]
* **Proxy System**: Players are represented by proxy actors on neighbouring servers, which are promoted to authoritative actors when needed. This system minimizes the need to recreate players during transitions[14].
* **Data Reuse and Event Communication**: Data is efficiently reused across servers, and event information is decentralized, ensuring smooth transitions and collaboration between servers[14].
* **Fault Tolerance**: If a server fails, adjacent servers maintain gameplay stability, ensuring the system continues without interruption[14], [15].

## B. Star Citizen

Star Citizen employs a similar approach but with an added replication layer that stores and synchronizes game state data between servers and clients. Key components include:

* **Replication Layer**: This layer manages the game state and ensures consistent synchronization between server nodes, facilitating smooth player transitions[16].
* **Streaming Bubbles**: As players move, the system dynamically loads and hands off control of entities between servers, allowing seamless transitions without interruptions[17].

In both games, server meshing allows for dynamic load distribution and near-limitless scalability, enhancing the player experience and reducing performance bottlenecks[17].

## C. What does this allow in the two games?

In both Ashes of Creation and Star Citizen, server meshing transforms how players interact with the game world:

1. **Seamless Exploration**: Players can move freely across the game world without being interrupted by loading screens.
2. **Massive Player Interactions**: Allows thousands of players to share the same space, meaning huge battles, and large-scale events.
3. **Dynamic World Events**: The game world adapts in real time. Players can participate in events that bring together large numbers of people, like kingdom wars in Ashes of Creation.
4. **Persistent Universe**: Players will experience a world that feels alive and always active. You can leave a mark on the world, like affecting the economy, territories, or factions.
5. **Large, Connected Worlds**: Instead of being restricted to one server or isolated area, players can seamlessly interact with others across vast distances.
6. **Uninterrupted Gameplay**: Players won’t experience sudden disconnects or crashes. The world continues without interruptions

# V. Future Developments

One of the most promising directions for future server meshing development would be the integration of artificial intelligence (AI) to predict and manage server load during high-intensity events. AI systems can be leveraged to preemptively identify scenarios, such as large-scale battles or player gatherings, where the number of participants is expected to exceed the server's capacity. By analyzing player movement patterns, historical data, and real-time behaviors, AI can anticipate these surges and proactively allocate additional resources or initiate the division of regions into smaller server nodes before the event occurs. This approach could significantly reduce downtime and improve player experience during critical moments.

Additionally, the implementation of dynamic gridding, which is currently in development by Intrepid studio could also mean an enhancement in server meshing, this technology dynamically subdivides servers into smaller grids based on player density, allowing for more precise load distribution[14].

# VI. Conclusion

Server meshing represents a significant evolution in online multiplayer game architecture, offering a solution to longstanding challenges in scalability, performance, and player experience. By enabling dynamic load distribution across multiple servers, it facilitates the creation of vast, persistent game worlds that can accommodate thousands of concurrent players without the performance bottlenecks typically associated with traditional server architectures. This technology's ability to scale horizontally, reduce server lag, and eliminate loading screens provides a more seamless, immersive gaming experience.

The potential of server meshing, as demonstrated by games like Star Citizen and Ashes of Creation, underscores its transformative impact on the gaming industry. These implementations showcase how dynamic server allocation can support massive player interactions and persistent world environments while maintaining high performance and fault tolerance.

However, despite its promise, server meshing is still in its early stages of development, with only a few companies actively working on full implementations. The ongoing evolution of this technology will depend on overcoming several technical hurdles, including real-time state synchronization and fault tolerance across a distributed system. The future of server meshing holds significant promise, for the gaming industry and cloud computing, where dynamic load management and real-time scalability are essential. As server meshing continues to evolve, it is poised to redefine the boundaries of what is possible in multiplayer game design, enabling richer, more expansive, and truly persistent virtual worlds.

References

[1] “Overview | Kubernetes.” Accessed: Oct. 15, 2024. [Online]. Available: https://kubernetes.io/docs/concepts/overview/

[2] “Fault Tolerance in Distributed Systems: Strategies and Case Studies - DEV Community.” Accessed: Sep. 27, 2024. [Online]. Available: https://dev.to/nekto0n/fault-tolerance-in-distributed-systems-strategies-and-case-studies-29d2

[3] “Distributed State — Challenges and Options | by Koshy | Medium.” Accessed: Oct. 01, 2024. [Online]. Available: https://nittikkin.medium.com/distributed-state-management-80c8100bb563

[4] “Fury at FWST-8 Battle Report | EVE Online.” Accessed: Sep. 27, 2024. [Online]. Available: https://www.eveonline.com/news/view/fury-at-fwst-8-battle-report

[5] “Introducing Time Dilation (TiDi) | EVE Online.” Accessed: Oct. 01, 2024. [Online]. Available: https://www.eveonline.com/news/view/introducing-time-dilation-tidi

[6] “Introducing the Megaserver System – GuildWars2.com.” Accessed: Oct. 01, 2024. [Online]. Available: https://www.guildwars2.com/en/news/introducing-the-megaserver-system/

[7] “The Nosy Gamer: The Server Meshing Wars Have Begun.” Accessed: Sep. 25, 2024. [Online]. Available: https://nosygamer.blogspot.com/2024/07/the-server-meshing-wars-have-begun.html

[8] “Load Balancing Algorithms - Simplified Learning.” Accessed: Oct. 23, 2024. [Online]. Available: https://waytoeasylearn.com/learn/load-balancing-algorithms/

[9] “Load Balancing Algorithms Explained with Code (and Visuals).” Accessed: Oct. 23, 2024. [Online]. Available: https://blog.algomaster.io/p/load-balancing-algorithms-explained-with-code

[10] “Load Balancing Algorithms - GeeksforGeeks.” Accessed: Oct. 23, 2024. [Online]. Available: https://www.geeksforgeeks.org/load-balancing-algorithms/

[11] “Types of load balancing algorithms | Cloudflare.” Accessed: Oct. 23, 2024. [Online]. Available: https://www.cloudflare.com/learning/performance/types-of-load-balancing-algorithms/

[12] “Server meshing - Star Citizen Wiki.” Accessed: Sep. 28, 2024. [Online]. Available: https://starcitizen.tools/Server\_meshing

[13] “Horizontal scaling vs vertical scaling: Choosing your strategy | DigitalOcean.” Accessed: Oct. 01, 2024. [Online]. Available: https://www.digitalocean.com/resources/articles/horizontal-scaling-vs-vertical-scaling

[14] “(42) Ashes of Creation Alpha Two Server Meshing Technology Preview - YouTube.” Accessed: Sep. 27, 2024. [Online]. Available: https://www.youtube.com/watch?v=pdav0as54mU&t=360s

[15] “Server meshing - Ashes of Creation Wiki.” Accessed: Sep. 27, 2024. [Online]. Available: https://pt.ashesofcreation.wiki/Server\_meshing

[16] “Server Meshing and Persistent Streaming Q&A - Roberts Space Industries | Follow the development of Star Citizen and Squadron 42.” Accessed: Sep. 25, 2024. [Online]. Available: https://robertsspaceindustries.com/comm-link/transmission/18397-Server-Meshing-And-Persistent-Streaming-Q-A

[17] “What is Star Citizens Serving Meshing and Persistent Entity Streaming | by Marvas McCladdie | Medium.” Accessed: Sep. 25, 2024. [Online]. Available: https://medium.com/@MarvasMcCladdie/star-citizen-is-one-of-the-most-ambitious-and-highly-anticipated-video-games-of-all-time-486abb0a02a9