GAMES NETWORKING & SECURITY

WEEK 2

GAME NETWORK ARCHITECTURE

Jack Ingram & Stephen Selwood

LEARNING OUTCOMES

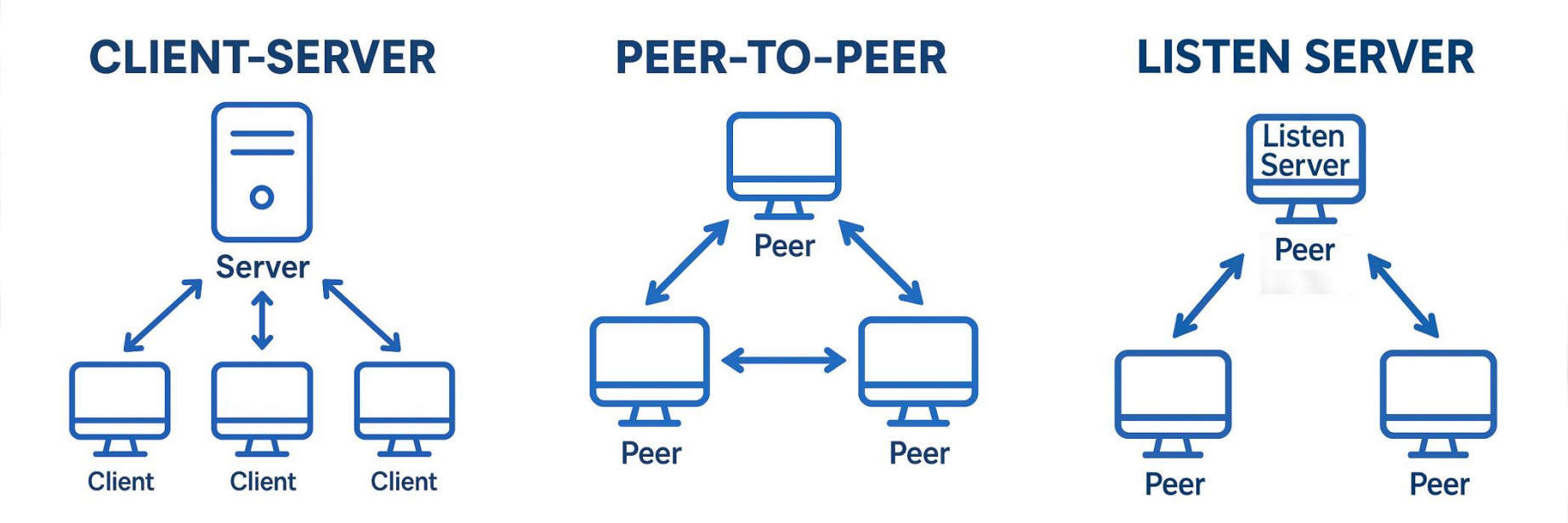
To review different networked games architectures.

To explore the different functions and servers required for a commercial game offering.

To review technologies required to deploy a commercial games offering.

To discuss the different network architecture models for deploying a global networked games.

Game Network Architecture



Different types of network architectures

Game Network Architecture

Network Architecture

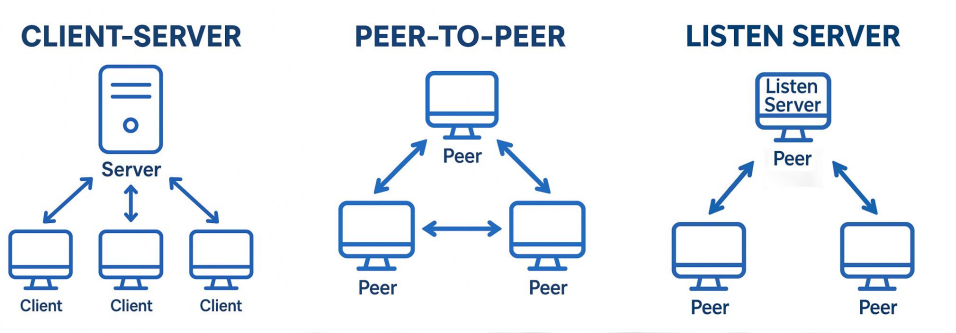
Network architecture impacts performance, fairness, and player experience.

Poorly designed systems can lead to:

- Lag, latency spikes, or desynchronised players.
- Unbalanced gameplay or opportunities for cheating.
- Server overloads and downtime during peak hours.

A well-designed architecture enables:

- Fast and consistent data exchange.
- Secure and authoritative control over the game world.
- Scalable infrastructure to support millions of players.



Different types of network architectures

Game Network Architecture

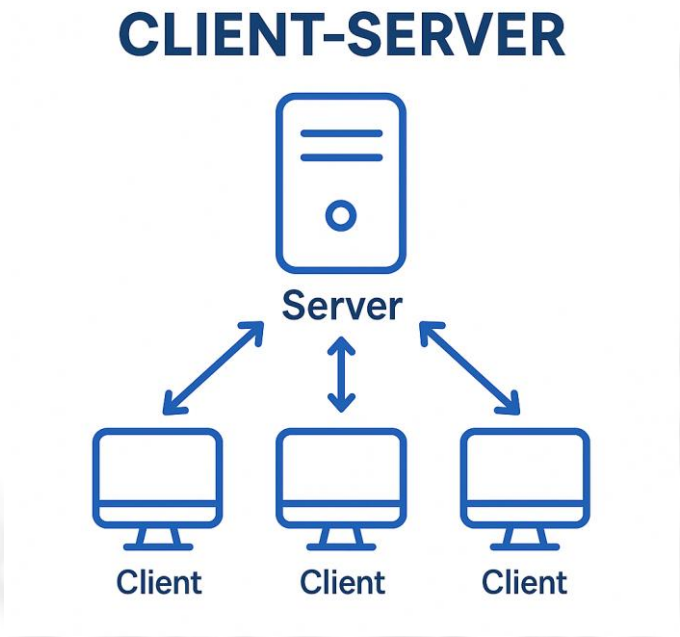
Client/Server Architecture

This is the typical and most common networked game architecture.

All client nodes are attached to a main, dedicated server node and share their updated local game states.

The server node, taking in this data, applies this data to the ‘true’, global game state. This global state is then sent to all client nodes to update their local game states.

This is known as a Dedicated Server.



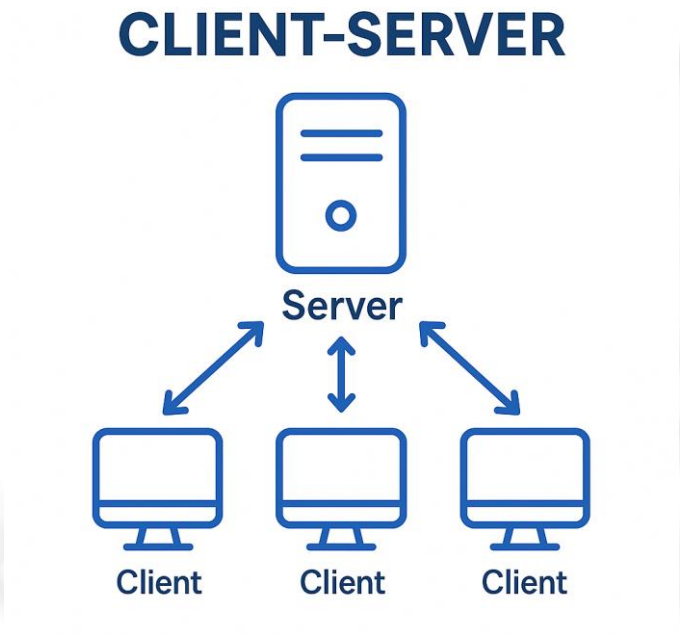
Multiple clients connected to a single server.

Game Network Architecture

Client/Server Architecture | Bandwidth

The bandwidth required for each client node to function as part of the network is constant across all client nodes as data packets are only ever sent to or received from the server.

No single client will have to send more or received more data than any other client, thus requiring no extra bandwidth.



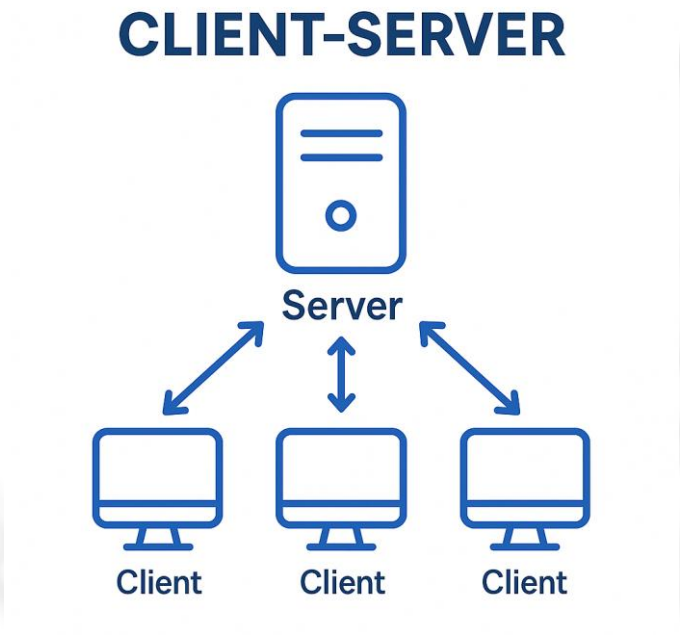
Multiple clients connected to a single server.

Game Network Architecture

Client/Server Architecture | Authoritative Server

The majority of networked games will run an Authoritative server model.

This is where the ‘true’ game state is held in the game server. This state is used to update all clients’ states so their local states all match.



Multiple clients connected to a single server.

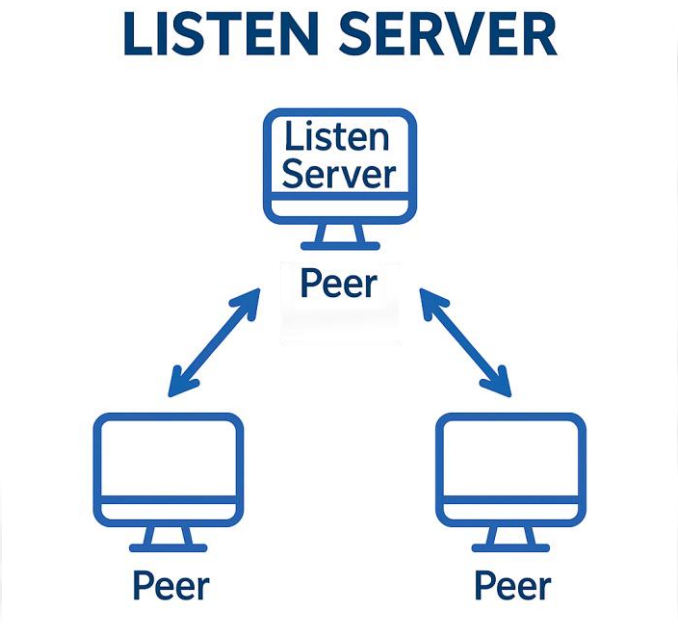
Game Network Architecture

Client/Server Architecture | Listen Server

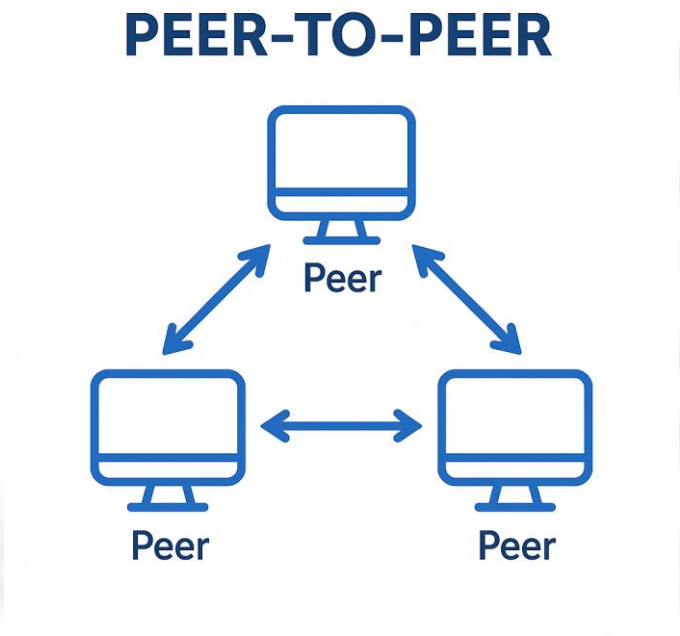
The Listen Server is a variation of the Client/Server model where one of the client nodes acts as the game server node. This model was common with *'LAN Parties'*.

Listen Servers are effectively free, but the client machine that is acting as the server is subject to increased bandwidth, lag and possible performance issues.

If the server node leaves the game, Host Migration will transfer ownership to another client, making that the new server node, so the game can continue.



One client or 'peer' acting as a server.



A peer-to-peer network: no server. All clients are directly connected to each other.

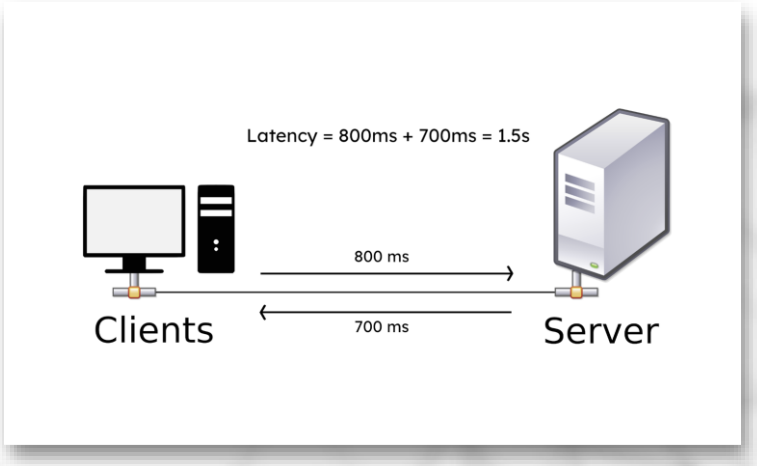
Game Network Architecture

Network Architecture | Latency

Latency is the time required for the data packet to travel from the source to the destination. Many things can affect latency, and latency in turn affects the Round Trip Time (RTT).

Latency is compounded further in the Authoritative Server model as the server node has to collect, correct and distribute the true game state to all nodes at each step (Tickrate).

Latency is an issue for all versions of server architecture.



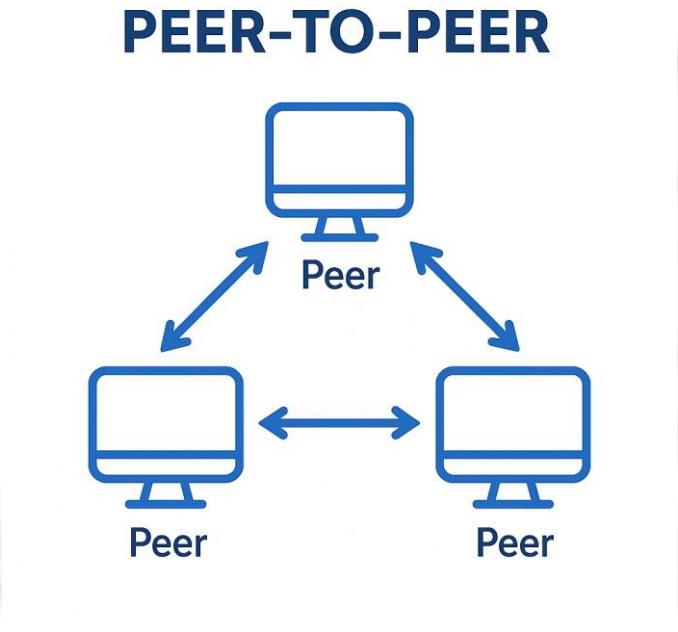
A very simplified example of Latency

Game Network Architecture

Peer-to-Peer (P2P)

In a Peer-to-Peer network, each peer may hold equal authority over the current game state as each peer in the network relies on each other peer to update it as to their game state.

There is no central server, but rather each client node on the network is directly connected to all other client nodes.



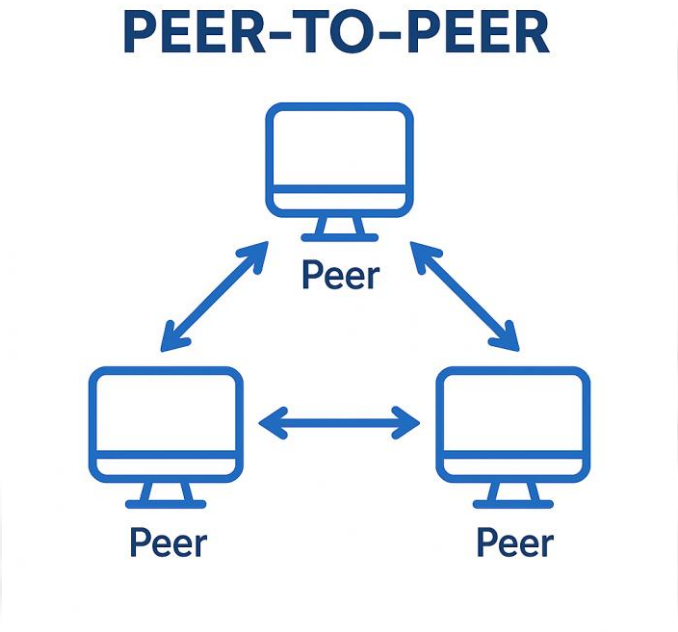
A peer-to-peer network: no server. All clients are directly connected to each other.

Game Network Architecture

Peer-to-Peer (P2P) | The benefits

The benefit of a Peer-to-Peer network is that latency is reduced compared to a Client/Server architecture as packets are multicast directly to each node and not through an intermediate.

P2P networks also benefit from their decentralised nature, meaning that the network isn't reliant on the existence of a server. If one machine goes down, the rest of the network will stay up and running. This also feeds into the bonus scalability of the network: additional nodes can more easily be added.



A peer-to-peer network: no server. All clients are directly connected to each other.

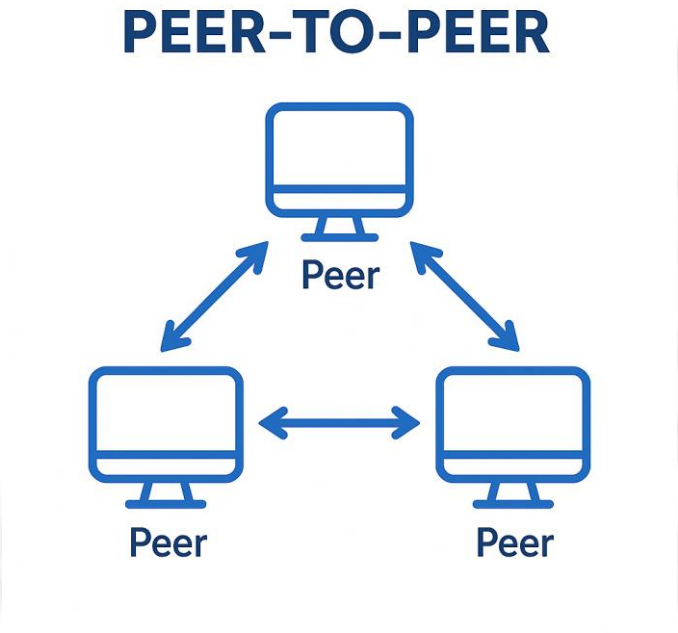
Game Network Architecture

Peer-to-Peer (P2P) | The drawbacks

There are a couple of drawbacks of Peer-to-Peer as well.

P2P networks have little in the way of backups. Backup systems have to be installed each node/machine rather than having one universal and all-encompassing backup system.

Security is also a key issue on P2P networks. Each node on the network is equally responsible for aspects such as virus or intrusion detection. If one machine is lax with this then it becomes a weak point.



A peer-to-peer network: no server. All clients are directly connected to each other.

Game Network Architecture

Commercial game Architecture | Recap

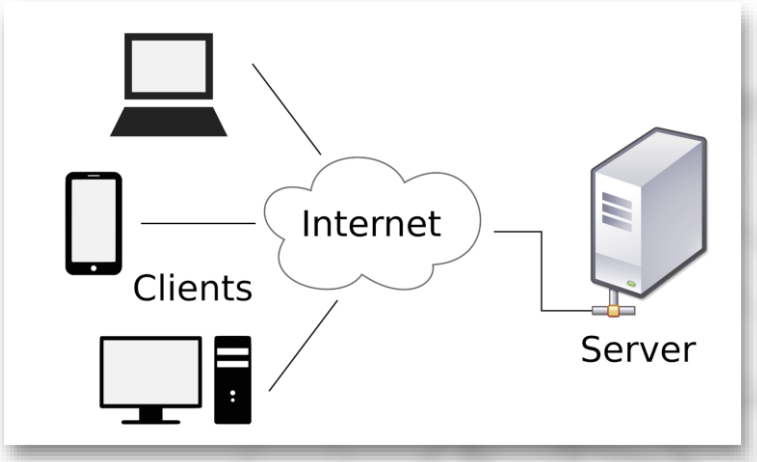
Topology	Central Server	Latency	Scalability	Cost	Typical Use Case
Client/Server	✅ Yes	Medium	Medium	Moderate	Most online games (e.g. Overwatch)
Authoritative Server	✅ Yes	Medium–High (due to validation)	Medium	Moderate	Competitive or MMO games
Listen Server	⚠️ Client-hosted	Medium–High	Low	Low	LAN parties or small co-op games
Peer-to-Peer (P2P)	❌ No	Low	High	Low	Indie or local multiplayer

Game Network Architecture

Commercial game Architecture

The architecture models previously discussed are abstractions of a commercial networked game architecture.

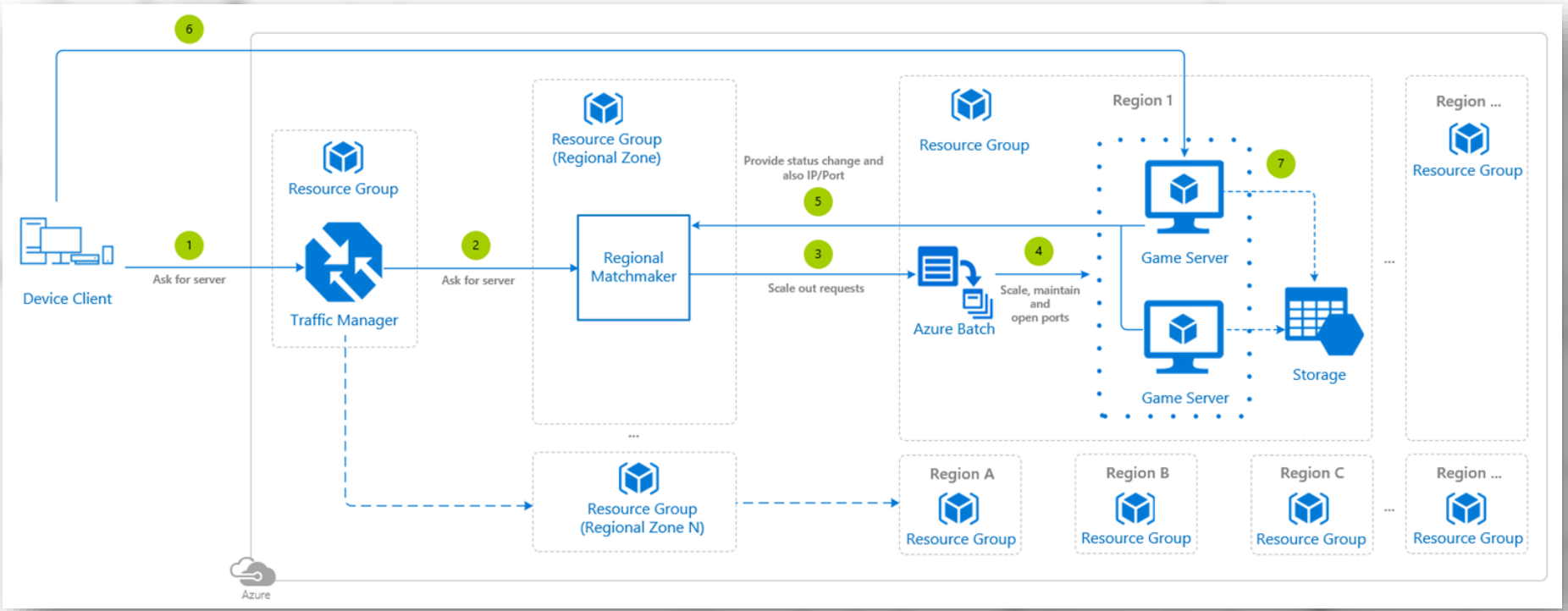
A commercial architecture has to accommodate hundreds of thousands of active players on game servers in different locations across the world.



A dedicated server is the most common type of game network.

Game Network Architecture

Commercial game Architecture

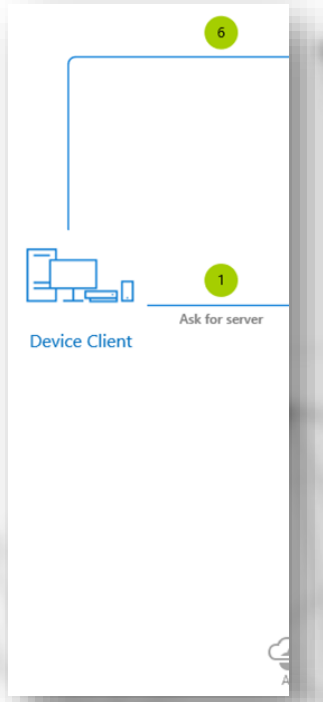


Game Network Architecture

Commercial game Architecture | Game Client

At the Application level (TCP/IP Stack) the game is started and an initial connection and authorisation request is sent from the client node.

As the game servers have a static IP Address and Port the data packets can be routed directly to the server infrastructure.

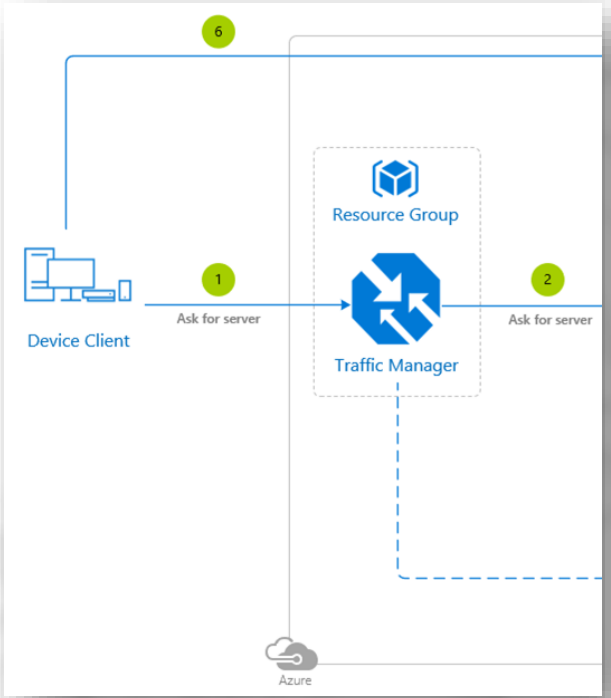


Game Network Architecture

Commercial game Architecture | Traffic Management

A connection request made from a game client to the game server arrives on a recognised Socket (Server Port/ IP Address).

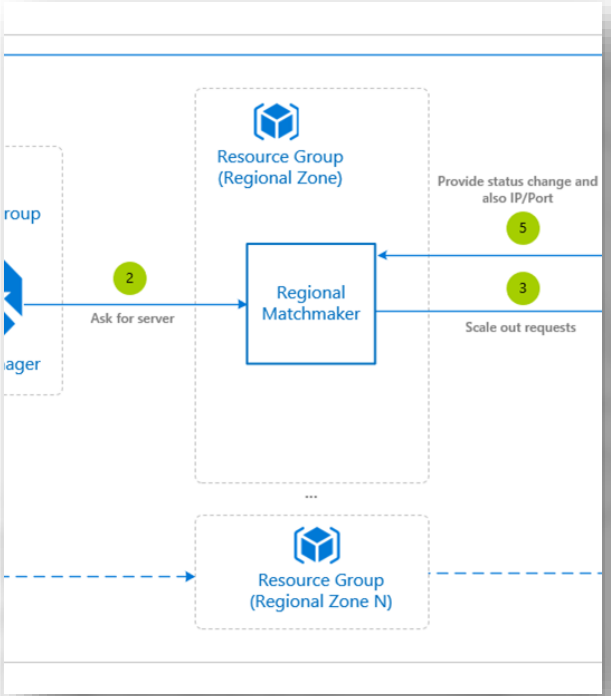
The Traffic Management solution will resolve the connection request by connecting the client to the *best* backend game server based on Location and Load Balancing profile.



Game Network Architecture

Commercial game Architecture | Matchmaking

Based on the IP address of the connection client, the Traffic Manager will resolve the connection request to a regional game server.

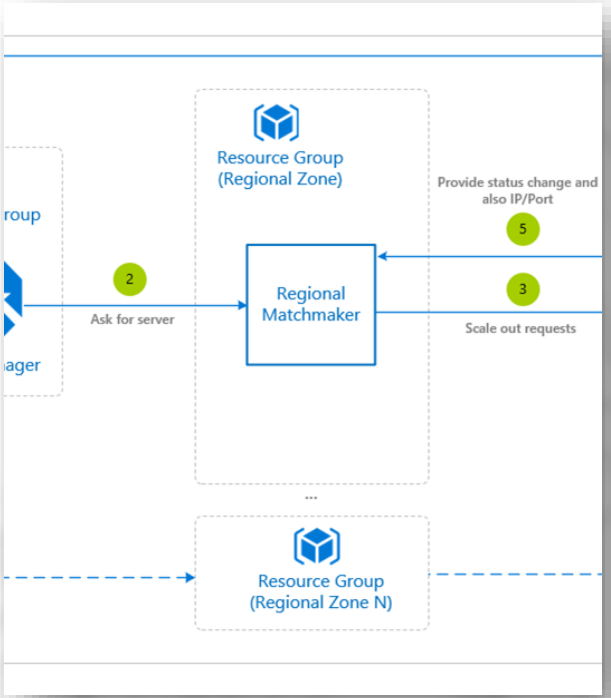


Game Network Architecture

Commercial game Architecture | Matchmaking

The closer the dedicated game server is to the client node, the more likely the player will experience a low RTT.

By connecting clients from the same geographical region, the less likely there will be excessive lag.



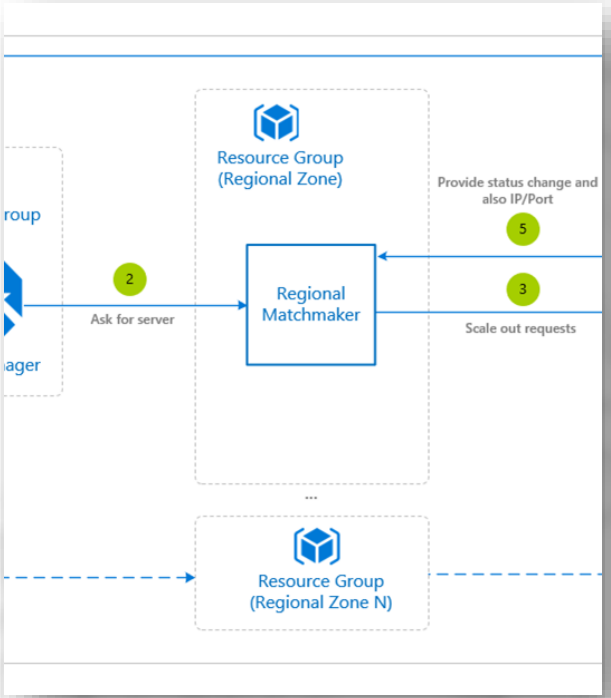
Game Network Architecture

Commercial game Architecture | Matchmaking

Additional benefits from connecting players to their local regional servers include:

Greater likelihood team members and other players will be able to communicate in their local language.

From a scalability perspective clients in the same local region will typically experience the same Peak Time, meaning the servers only have to endure significantly higher usage once a day rather than endure general higher usage all day.



Game Network Architecture

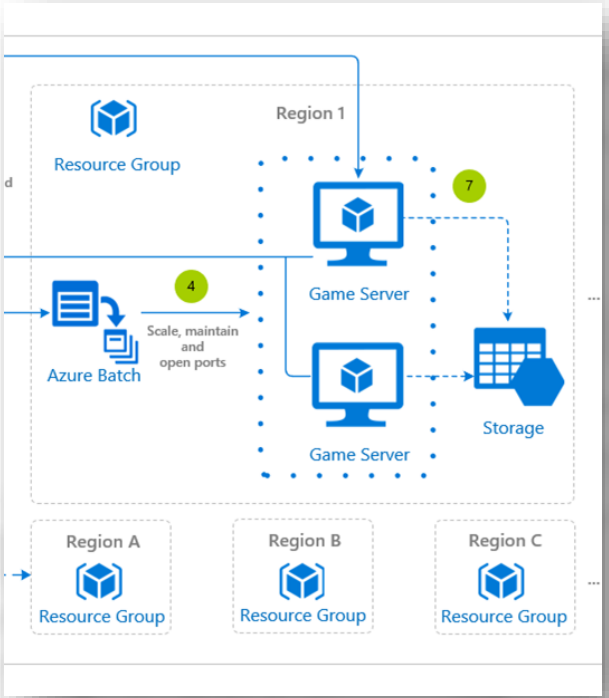
Commercial game Architecture | Game Servers

The Game Servers are a collection of instances that supply the functions required by the game itself.

This could include:

- Game World servers,
- Loot or asset servers,
- Chat servers,
- Login servers,
- Store fronts

Databases will also be associated to the servers to access necessary information.



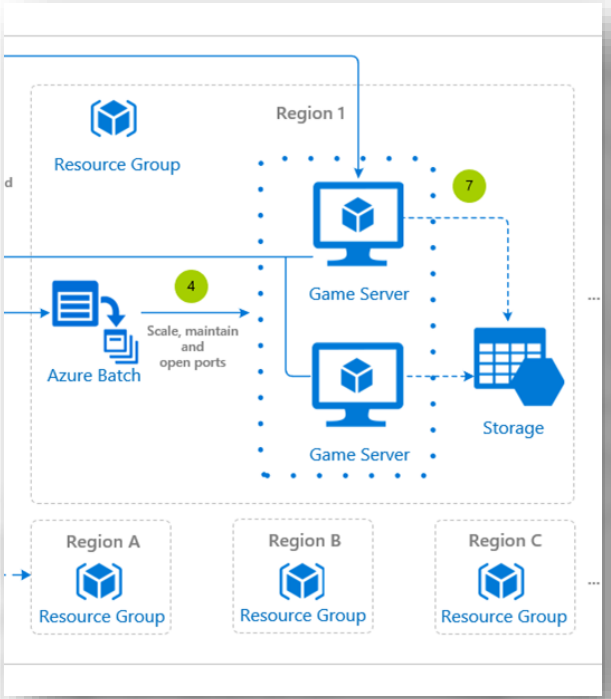
Game Network Architecture

Commercial game Architecture | Game Servers

Popular games with high peak users are likely to have multiple instances of Game World servers to ensure a Game World is balanced and not overcrowded, which would result in lag.

Map sizes and player limitations run on the server would also support this:

>Overwatch - 12 players,
>Team Fortress 2 - 32 players
>PUBG - 100 players



Game Network Architecture

Commercial game Architecture | Deployment

A game such as League of Legends needs multiple servers to deal with demand - 46.5 million peak daily players as of August 2025*.

League of Legends runs 11 localised servers** to handle this global demand:

Brazil, Europe Nordic & East, Europe West, Latin America North, Latin America South, North America, Oceania, Russia, Turkey, Japan and the Republic of Korea

*Source: activeplayer.io
**Source: leagueoflegends.fandom

Game Network Architecture

Commercial game Architecture | Deployment

To deploy the update servers, a number of industry standard solutions are used:

- Virtualization,
- Containers,
- Docker/Kubernetes

Game Network Architecture

Commercial game Architecture | Fortnite Example

- Epic Games runs Fortnite using a hybrid Client/Server and Cloud model.
- Game servers are authoritative to prevent cheating and maintain fairness.
- Matchmaking and region routing handled via AWS and Google Cloud infrastructure.
- Each player is connected to the nearest data centre to minimise latency (e.g. Europe, NA-East, Oceania).
- Game data (assets, progress) synchronised through a centralised back-end.
- Scales dynamically to support millions of concurrent users.

BREAK

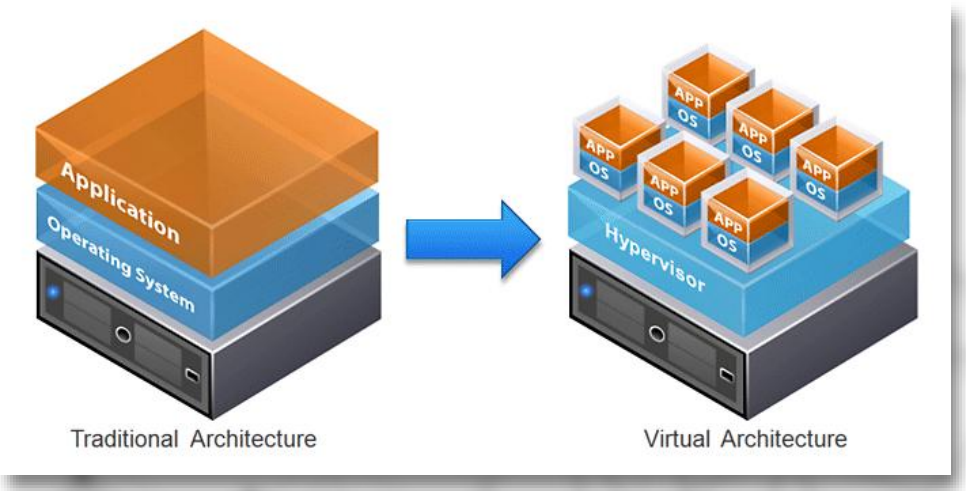
Game Network Architecture

Virtualisation

Virtualisation is the process of taking a single physical server and, on a software level, splitting it into a number of different independent and unique servers that can run different OS and applications.

A Virtualized server can support multiple instances of different game servers at any given time.

Before Virtualisation, one server could only run one OS and one application at any given time.



Game Network Architecture

Dockers/Kubernetes

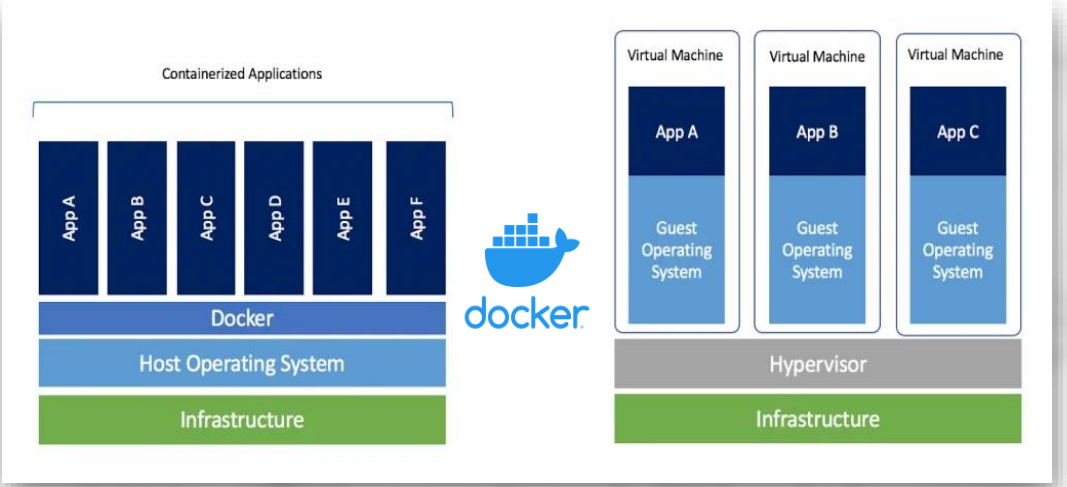
Both Docker and Kubernetes provide solutions for managing the deployment and distribution of the containers to the servers from the host environment.

Game Network Architecture

Dockers | What is it?

Docker is open-source platform for the deployment, shipping and running of applications that can be run either on the cloud or on-premises.

These applications are packaged and ran in a loosely isolated environment called a container.



Game Network Architecture

Containers | What is it?

A container is a standard unit of software that packages up code and all its dependencies and necessary elements into one package. This means that the application runs quickly and reliably from one computing environment to another.

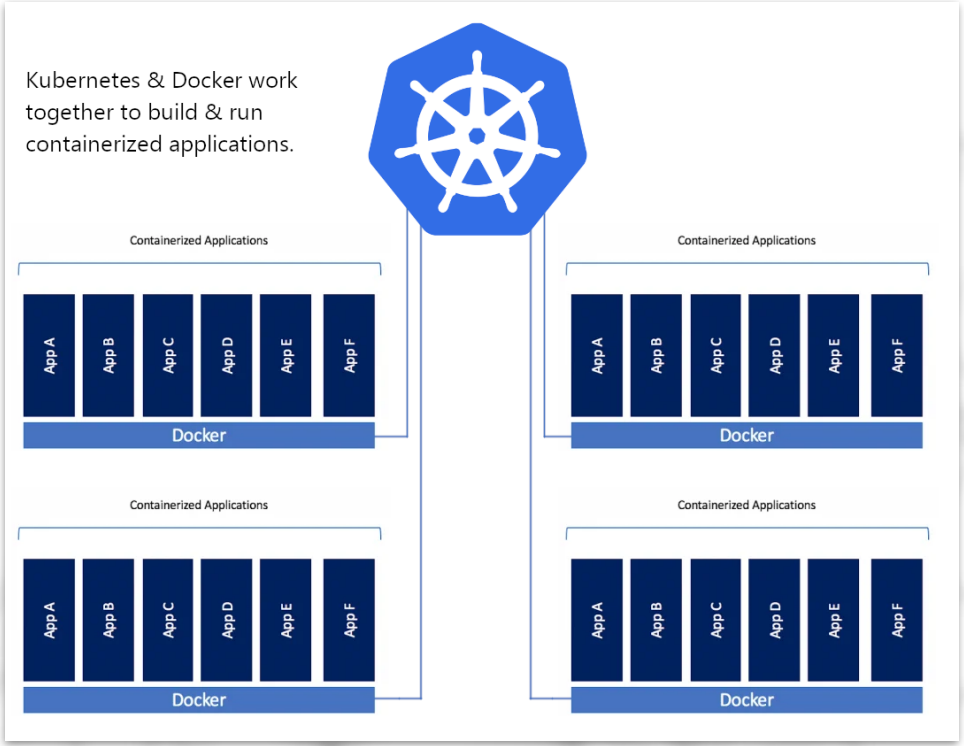
Containers extend the virtualisation process further by allowing multiple images of an application to run on a virtualized server.

Game Network Architecture

Kubernetes | What is it?

Kubernetes is open-source software that provides an API to control how and where application containers will run.

It allows you to run your Docker containers and helps you to tackle some of the operating complexities when moving to scale multiple containers, deployed across multiple servers.



Game Network Architecture

Localisation

The principal benefit of setting up localised game servers is to reduce latency for players across different regions.

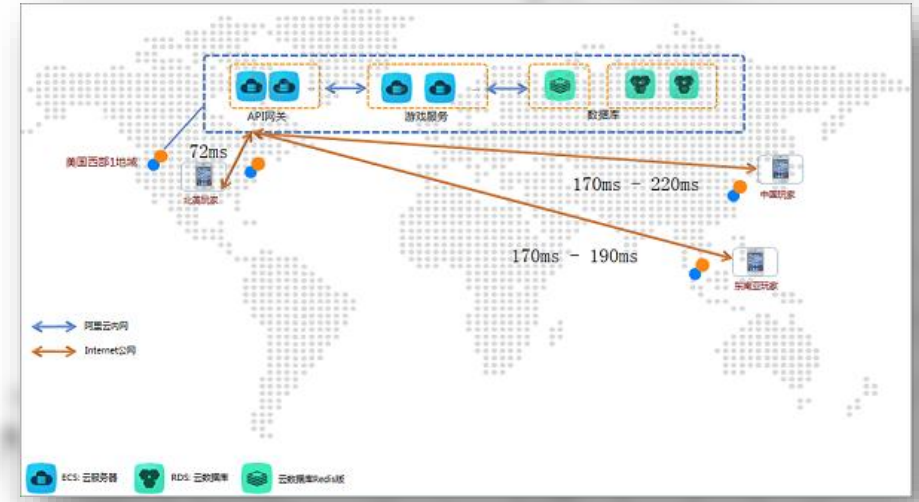
There are a number of different models that can be considered for localizing a game, ranging from fully-centralised to fully-distributed.

Game Network Architecture

Localisation | Fully-Centralised Deployment

A *Fully-Centralised Architecture* would mean that the game servers and services would exist in one central location and all players would connect to that one instance.

This architecture would be suitable for games that are not sensitive to latency.



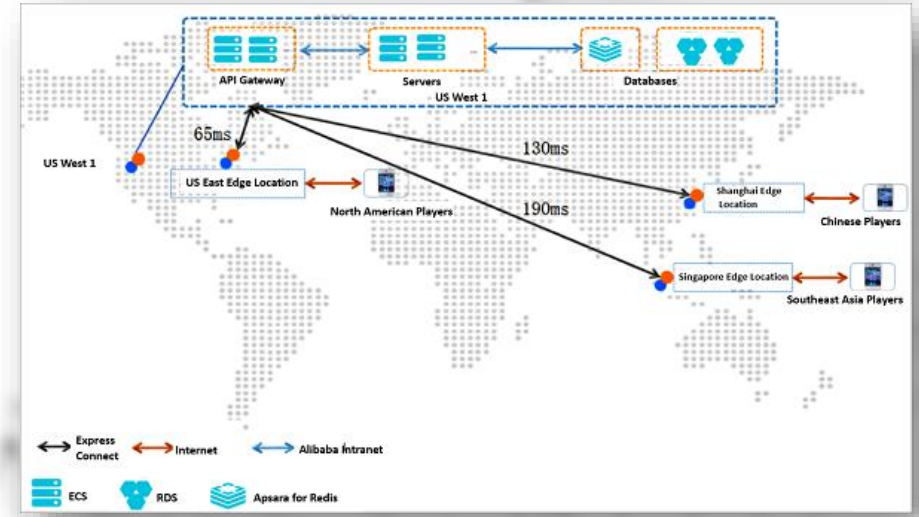
Game Network Architecture

Localisation | Centralised with Network Optimization

Similar to the *Fully-Centralised Architecture*, this model uses *Elastic IP Addresses* to connect distance players to the centralised servers.

An *Elastic IP Address* is a semi-static IP Address that connects the *Edge Location* to the centralised servers.

Unlike a Public IP Address, an *Elastic IP* retains its address once a connection is terminated.



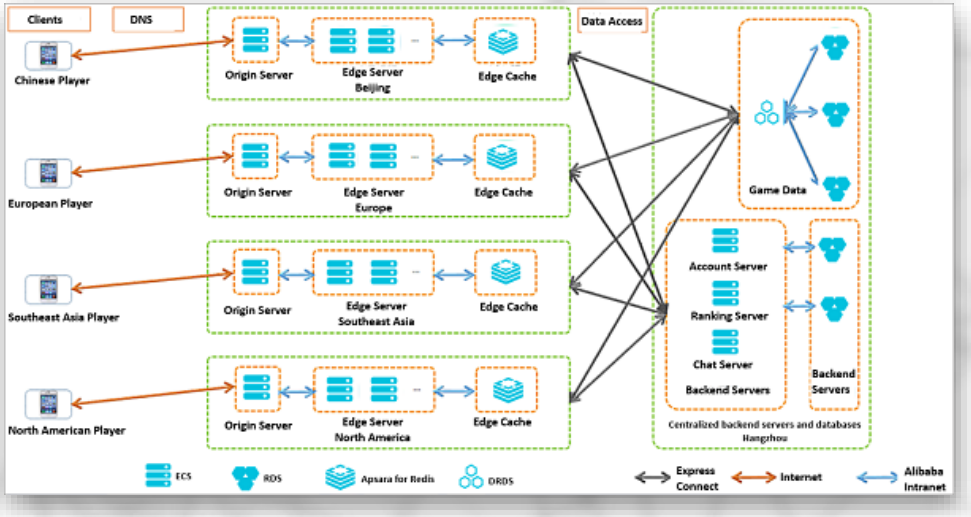
Game Network Architecture

Localisation | Centralised with Distributed Logic

In this model the game data is centralised, while the game logic is moved to localised server node.

The benefit of this is that the game state runs on servers in the players region, ensuring the best real-time player experience, while the data (that may take a few moments to retrieve) is held at centralised locations.

Costs are now being incurred for hosting game servers in multiple regions.

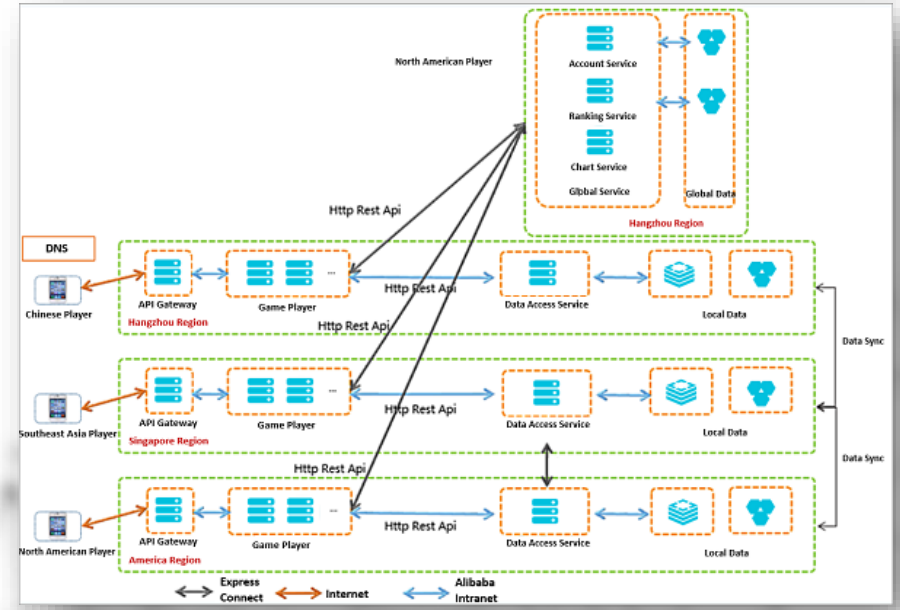


Game Network Architecture

Localisation | Fully-Distributed

In this final model, both the game logic and game data are held at localised regional centres, and the business data is held in the centralised location.

The benefit of this model is quick response time for the player in their region, but the cost of regional hosting increases, plus there is the issues of syncing all localised servers so they are consistent.



REVIEW

There are a number of different networked games architectures, but the Client/Server model is the most favoured.

A commercial networked game is based on the Client/Server model but is a scaled version to meet the player requirements of the game.

There are different localisation architectural models available to a commercial networked game developer.

Cost and complexity rise with more distributed models.

QUESTIONS?

Game Network Architecture

Group Activity

Over the coming weeks, within the module tutorial, you will be building a networked game.

This game will have the standard network functionality like players seeing each other's avatars, being updated with their actions, movement and position etc.

On top of these features, you will also be looking into adding in features such as chat functionality and a leaderboard.

Based on this description, report back on the following:

- Which architecture model do you think should be used and why?
- Which of these services require servers to function?
- Extension: what communication protocols are going to be used and for what data?

Review and group discussion for 10 min.

Take notes with pen and paper or on your phone/laptop.

Informal report back.