Lab 15:

1. In the context of IP restricted NAT devices, what does the hole-punching technique do? Describe this technique, using a diagram to aid your explanation if appropriate.

Hole punching is a technique used to establish a direct communication channel between two devices located behind different NATs (Network Address Translators). NATs are often used to allow more than one device to share a common public IP address.

Typically multiple devices behind NATs cannot communicate with one another directly as Nats block incoming connections. However hole punching hole punching solves this by allowing the devices to exploit the behaviour of NATs and establish a direct connection.

It works as follows:

1. Device A and Device B want to connect directly but NAT is preventing this.
2. Both devices independently sent a message to an online server.
3. The server receives the messages and notes the port and public IP of each device.
4. The server sends the public IP and port of each device to the other device.
5. Device A and Device B begin sending packets to each others public IP and port.
6. Having received outgoing packets the NATs on both devices create temporary mappings in their NAT tables.
7. The NATs allow incoming packets from the IP and Port they received packets from.
8. Device A and B can now communicate directly by sending packets to each others Public Ip and ports.

A screen shot of a computer

Description automatically generated with medium confidence

2. In multiplayer online games, interest management cuts down the bandwidth usage by filtering irrelevant updates. Describe two common techniques of interest management.

* Spatial Partitioning: This technique divides the game world into spatial regions or cells. Each player is assigned to a specific cell or set of cells based on their position in the game world. The server only sends updates to players for the objects within their assigned cells or neighbouring cells. This way, players are only aware of and receive updates for nearby game objects, reducing bandwidth usage.
* Entity Component System (ECS): In an ECS architecture, game objects are composed of components that represent specific functionalities. Each player's client maintains a local view of the game world by subscribing to specific components of interest. The server only sends updates for the subscribed components to each player. This allows players to receive updates relevant to their interests while filtering out unnecessary data.

3. Interest management is important for good network performance in massively multiplayer games. What is a potentially visible set, and how does this approach differ from static zones? How do these interest management approaches benefit the game?

* Potentially Visible Set (PVS) is a dynamic approach where each player maintains a list of game objects that are potentially visible from their current position. This set is continuously updated based on the player's movement and the game world's visibility rules. When updates are sent, they are filtered to include only the game objects in the player's potentially visible set. This approach optimizes network traffic by reducing the number of updates sent to each player.
* Static Zones is a static approach where the game world is divided into predefined zones or areas. Each player is associated with a specific zone. Updates are only sent to players who are present in or near the same zone as the game object. This approach simplifies interest management but may result in sending updates to players who are not actually interested in those objects.

The PVS approach differs from static zones in the following ways:

1. Dynamic Visibility: The PVS approach takes into account the actual visibility of game objects from the player's current position. It considers factors such as line of sight, occlusion, distance, and other visibility calculations to determine which objects are potentially visible to the player. The visibility information is dynamically updated as the player moves within the game world.
2. Object-Driven: The PVS approach is driven by the objects themselves rather than predefined zones. Instead of dividing the game world into fixed zones or areas, the PVS approach considers the visibility of individual game objects and dynamically determines which objects should be included in the player's potentially visible set. This allows for more fine-grained control over the updates sent to each player.

These interest management approaches benefit the game by reducing network bandwidth usage and server processing load. By filtering and sending only relevant updates to players, the overall network performance is improved, and the game can scale to support a larger number of players. Additionally, these techniques help optimize the player experience by ensuring that players receive updates for objects that are relevant to their current location and interactions, enhancing realism and immersion.

1. Describe one example for client-side attack and server-side attack, respectively. Please include details of how this attack works and a mechanism to prevent it.

Client-Side Attack: One example of a client-side attack is a cheat or exploit that modifies the game client to gain unfair advantages in gameplay. For example a speed hack in an online racing game. In this attack, the player modifies the game client to increase the speed of their vehicle beyond the normal limits. This gives them an unfair advantage over other players by allowing them to move faster and win races more easily.

To prevent client-side attacks, game developers employ several security measures:

1. Client-Server Validation: The game server constantly validates the actions and inputs received from the client. It performs checks to ensure that the client's actions are valid and within the expected limits. For example, in the case of a racing game, the server would verify that the speed of the player's vehicle does not exceed the maximum allowed speed.
2. Data Verification: Game clients can use various techniques to verify the integrity of the game files. This can involve using cryptographic checksums or digital signatures to ensure that the game client has not been tampered with. If any modifications are detected, the client can be prevented from connecting to the game server.
3. Server-Side Simulation: Critical game mechanics and calculations are performed on the server rather than the client. For example, in the case of speed in a racing game, the server would calculate the speed and position of each player's vehicle and distribute this information to all clients. By keeping the authoritative control on the server, cheating attempts by manipulating the client become ineffective.

Server-Side Attack: One example of a server-side attack is a Distributed Denial of Service (DDoS) attack targeting the game server. In a DDoS attack, the attacker overwhelms the server with a large volume of malicious traffic, causing it to become unresponsive or crash. This disrupts the gameplay experience for legitimate players.

To prevent server-side attacks like DDoS attacks, game developers could implement the following measures:

1. Network Filtering: Game servers employ network filtering techniques to identify and block malicious traffic. This can involve setting up firewalls, Intrusion Detection Systems (IDS), or traffic analysis tools that can detect and mitigate DDoS attacks by blocking or rate-limiting suspicious traffic.
2. Load Balancing and Redundancy: Game servers can distribute the incoming traffic across multiple server instances using load balancing techniques. By spreading the load, it becomes harder for attackers to overwhelm a single server. Additionally, redundant server infrastructure can be set up to ensure that if one server is compromised, the game can continue to operate on other servers.
3. Traffic Monitoring and Anomaly Detection: Game servers can employ monitoring systems to analyse network traffic patterns and detect anomalies that might indicate a potential attack. These systems can automatically trigger protective measures or alert administrators to take appropriate action.
4. Content Delivery Networks (CDNs): CDNs can be utilized to distribute game content and handle incoming traffic. CDNs have robust infrastructure and distributed servers that can absorb and mitigate DDoS attacks, protecting the game server from direct attacks.
5. Cheating in online games is the action of pretending to comply with the rules of the game, while secretly subverting them to gain an unfair advantage over an opponent. Describe two ways of cheating. Please include details of how each cheating works and a mechanism to prevent it.
6. Aimbot: An aimbot is a cheat that assists players in aiming and shooting their opponents with high precision. It typically works by automatically tracking the positions of other players and adjusting the cheater's aim to ensure accurate shots. This gives the cheater an unfair advantage by eliminating the need for skillful aiming.

Mechanism to Prevent Aimbot Cheating:

* Anti-Cheat Systems: Game developers often implement sophisticated anti-cheat systems that continuously monitor player behaviour and detect abnormal patterns. These systems can detect the use of aimbots by analysing the accuracy and consistency of player shots. When detected, the cheating player can be flagged or banned from the game.
* Behaviour Analysis: Games can employ behaviour analysis techniques to identify players who consistently display unnatural aiming behaviour. By monitoring statistics such as hit rates, kill-to-death ratios, and player movement patterns, suspicious activities can be identified and investigated.

1. Wall Hacks: Wallhacks, also known as ESP (Extra Sensory Perception), allow players to see through walls and other solid objects. This cheat gives players the ability to track the positions of opponents hidden from regular view, providing a significant advantage in terms of planning attacks and avoiding detection.

Mechanism to Prevent Wall Hack Cheating:

* Game Server-Side Validation: Game servers can validate the visibility of objects and players to ensure that only the information that should be visible is transmitted to the clients. By maintaining control over the visibility of game elements, the server can prevent wall hacks by not revealing the positions of hidden players or objects to cheaters.
* Encrypted Data Transmission: To prevent wall hacks, game developers can employ encryption techniques to secure the communication between the game client and server. This helps prevent cheaters from intercepting and manipulating the game data to gain unfair visibility advantages.