**Source Engine Multiplayer Matchmaking**

Online multiplayer presents a unique challenge for game developers regarding how to select and gather groups of players for a session. Normally, players don’t put much thought into what server to join because the process of searching for an active server has been streamlined through the game itself through matchmaking. Many games featuring online matchmaking solve the problem by simply directing the player to the nearest server relative to his/her location akin to anycast. However, this challenge becomes more complicated for online games which need to take user preference into account. The company known as Valve Software has incorporated user preference as a part of their own matchmaking system into their game engine, the Source Engine, where users would be presented with a list of various servers and decide which one to join. Valve’s networking system was created using a modified set of TCP and UDP protocols such that it can allow any Source Engine games to easily establish an online multiplayer network and advertise themselves without any prior set up needed.

**Steam’s Network Architecture**

Because Valve Software’s platform serves as both a gaming platform and a hosting service for their games, the company also acts as the main platform which bridges the connection between a client and a game’s servers. The goal of Source Engine’s networking is to allow anybody to create and host a server which can be advertised by Valve’s platform, Steam. To achieve this, they use an architectural infrastructure similar to the Master/Slave architecture used in DNS and other multiplayer matchmaking platforms. When a server wants to be advertised, it will connect to the master server using a set of modified UDP packets using Steam’s headers and takes place on port 27015[[1]](#footnote-1).

The master server is responsible for managing and keeping track of all servers of a specified game similar to how DNS keeps track of the relationships between IP Addresses and domain names[[2]](#footnote-2). These master servers are owned by and managed by Valve, giving them access to player and server statistics, and the master server keeps track of all available servers, the IP Addresses being used, and the port number needed to connect to them. Any players who want to find a game will first connect to the master server and request a query of all available servers found.

**Obtaining a query of all available servers**

When a player starts a game, the first thing he/she must do is find a server to play in. The user client will send a query request to what is known as the master server, a host that is responsible for maintaining a list of all known servers similar to a root DNS server. As such, clients would send a UDP Request message which can be seen as an abbreviated and modified version of a DNS request packet for a list of servers. This process is known as the Master Server Query Protocol[[3]](#footnote-3). For these messages, the UDP header will contain FF FF FF FF 54 53 to identify its packet as the Source Engine Query protocol[[4]](#footnote-4).

An example of the packet would be the hex packet shown below. This is an old version of the Master Server Query Protocol which uses a slightly different header label for a request message[[5]](#footnote-5).

31 ff 30 2e 30 2e 30 2e 30 3a 30 00 5c 6e 61 70 70 5c 35 30 30 00

|  |  |  |
| --- | --- | --- |
| **Type** | **Hexadecimal code** | **Meaning** |
| Message Type | 0x31 | Query request for all servers |
| Region Code | 0xFF | Return a list of servers from all regions |
| IP PORT | 0x30 2E 30 2E 30 2E 30 3a 30 00 | Translates to IP Port: 0.0.0.0:48  Establish a socket with the master server for response and to establish future connections. Like DNS, must start with 0.0.0.0 as your IP address. |
| Filter | 0x00 | No filter restrictions |

The client will establish a socket connection and send a UDP packet to Steam’s master servers. The packet shown above would be the default query packet used to get a list of all available servers from the master server[[6]](#footnote-6). Like other UDP packets sent through a socket connection, the packet sent and converted between hexadecimal and int32. The master server will process the packet and extract the byte array to read the request message[[7]](#footnote-7). The master server will reply with its own message in the form of a UDP Reply packet. Originally, this response was much simpler in that it simply responds with a UDP header of FF FF FF FF 66 0A with the rest of the message being a series of IP Addresses and their respective ports being used. However, the protocol was changed into an undocumented version[[8]](#footnote-8).

As of December 2016, Steam has migrated the Master Server Query Protocol and integrated it into their Steam client. This means that the Steam client must send its request directly to the Steam platform to receive a list of server IPs. This new process is not documented, and the UDP headers used weren’t found. As a result, any attempts to detect the new version of the Master Server Protocol have been unsuccessful. In addition, any attempts to use the legacy protocol even through Steam’s Python API[[9]](#footnote-9) have proven to be unsuccessful because all master servers have been updated to reject all UDP queries from clients[[10]](#footnote-10), and all legacy games have been updated to use the new Master Server Query format[[11]](#footnote-11). This was part of Valve’s on-going updates to prevent malicious users from abusing Steam’s services to initiate DDoS reflection attacks and to prevent packet spoofing[[12]](#footnote-12). Currently, this protocol can be initiated only through the Steam client, Steam’s publisher directory, or through the web client which requires an API key. This made prevented me from recording and documenting the current version and the legacy version of the Master Server Query through Wireshark.

A picture containing letter

Description automatically generated

This is the Master Server Query response output received when sending a request via Web API: <https://api.steampowered.com/IGameServersService/GetServerList/v1/?access_token=XXX>

Regardless of the method used, the Master Server Query Protocol will provide the client with a list of server IP Addresses, the port used based on any filters for a specific game. However, this may not be enough information to assist a user in choosing a server[[13]](#footnote-13). The client may want to request more information such as latency, game type, custom server settings, and what other players might be on the server. Since our client now has a list of IP Addresses, the client can simply ask each servers directly for more information, leading to an additional set of server query protocols.

**Getting more details about every server**

In order to acquire further details regarding a server, the client will forge what the UDP header labels as the Source Engine Query packet. While this packet is labelled as such, the protocol uses 3-5 different UDP packets, each with a different header used to request a specific set of information from the server. Out of the 5 query packets, 2 have been documented as obsolete and been deprecated from all Source Engine games.

* **A2S\_INFO**
  + A UDP packet sent to request basic server information and directed at a specific server of an IP Address.
  + This packet is sent to every server found when a game client requests a list of all available servers.
  + Requested details include player count, server name, game in progress, and the player count of that server.
  + Starts with FF FF FF FF 54 53 to mark itself as an A2S\_INFO Request packet
  + Includes a 4-byte challenge number to prevent packet spoofing and reflection attacks[[14]](#footnote-14).

Graphical user interface, text

Description automatically generated

Whenever a Source Engine game or the Steam client has to display a list of server, it will send an A2S\_INFO Request packet to every available server. The image shows a Wireshark capture shortly after requesting an unfiltered list of all available servers.

* **A2S\_Player**
  + Sends a request to the server to obtain information about the players currently in that server’s game session
  + This packet is sent whenever the game client requests more information regarding a specific server.
  + Starts with 0xFF FF FF FF 55 and includes the string “Source Engine Query\0” as part of its header.
* **A2S\_Rules (Couldn’t be found, possibly integrated into another protocol)**
  + Sends a request to the server about any special settings that a server might have[[15]](#footnote-15)
  + These settings would include descriptions such as if the server is officially hosted by Valve Software Co., is the server using a custom game mode, is the server marked as being under Valve’s VAC Anti-Cheat software, and any other special settings that could have been implemented by the server owner
  + Starts with 0x56 in its header
* **A2A\_PING (Obsolete, deprecated)**
  + Merely a ping request used to check how much latency is between the user and the server host. This information is acquired when sending an A2S\_INFO request or during the Master Server Query protocol
  + When trying to test for A2A\_PING messages on Wireshark, only A2S\_INFO packets were sent. This means that the ping function has been integrated with A2S\_INFO.
  + Declared obsolete by Valve employees as of December 27, 2012, and is no longer used.
* **A2S\_SERVERQUERY\_GETCHALLENGE (Obsolete for most games)**
  + While we’re using UDP to send and receive request and response packets, these messages are tailored towards specific clients and each user needs to have their conversation labelled to identify the client machine.
  + Whenever a user first connects to a server, the user would have to request a challenge number from the server[[16]](#footnote-16).
  + That challenge number is how the user identifies himself when sending requests to the server[[17]](#footnote-17).
  + This UDP packet is how the client would request a challenge number when speaking to a specific server to identify itself.
  + This format is no longer used for most games. Instead, clients will send an A2S\_PLAYER or an A2S\_RULES packet with an empty challenge number.

The server will receive the request and send what is labelled as the Source Multi-packet Response Format. Because we’re sending it to one client and this response packet is also written in UDP which can be delayed or sent out of order, the packet header must be modified such that it can provide details and information regarding the server’s connection to that specific client. This means that the UDP header must include an ID number assigned to that specific user and label to identify the packet order. This ID number is labelled as the Challenge Number. The server will randomly assign a Challenge Number to the client to identify that particular user’s conversation[[18]](#footnote-18). This challenge number is attached at the end of all server reply packets which are sent based on the type of request packet received.

* If the request packet was an A2S\_INFO packet, it returns an A2S\_INFO Response packet providing a list of server details and settings established by the server.

Table

Description automatically generatedGraphical user interface, timeline

Description automatically generated

**Label for each part of the packet:**

|  |  |
| --- | --- |
| **Hex Code** | **Meaning** |
| 0xff ff ff ff 49 | 49 labels this as a response packet |
| 0x11 | 0001 0001 This is a public server, and it has VAC anti-cheat enabled |
| 0x 53 75 6e 72 75 73 74 20 2d 20 5a 6f 6d 62 69 65 20 53 75 72 76 69 76 61 6c 00 | Server Name: Sunrust – Zombie Survival |
| 0x7a 73 5f 74 68 65 5f 70 75 62 5f 65 64 69 74 32 00 | Map Name: zs\_the\_pub\_edit2 |
| 0x67 61 72 72 79 73 6d 6f 64 00 | Game Name: garrysmod |
| 0x 5a 6f 6d 62 69 65 20 53 75 72 76 69 76 61 6c 00 | Game Mode: Zombie Survival (This is what the server browser displays when looking at the server) |
| 0x67 6d 3a 7a 6f 6d 62 69 65 73 75 72 76 69  76 61 6c 20 | Server Tag gamemode is set to zombiesurvival (This is how the browser labels and sees the server’s gamemode) |
| 0x76 | This server supports a maximum of 118 players (+ 9 bots) |
| 0x65 | There are 101 human players present in the game |
| 0x80 00 | This server supports up to a maximum of 128 players |
| 0x 01 32 30 32 31 2e 30 36 2e 30 39 00 | Date the server was created: 06/09/2021 |
| 0x 20 67 6d 63 3a 70 76 70 | This game type is PVP |
| 0x20 76 65 72 3a 32 31 30 37 31 32 00 | Version Number: 210712 |
| 0x6c 6f 63 3a 75 73 20 76 65 72 3a 32 31 30 37 31 32 00 | This server is located in the U.S. |
| 0x00 00 00 00 | End of packet |

The packet response recorded during this game is different from the default format found on Valve’s wiki page. The creator of this game appears to have changed the network protocol to better suit his game. Compared to the page describing the server response format, the following types were either not found or taken out in this version of the Source Engine:

* ID: Gives the Steam App ID used to identify the type of game
  + Possibly removed because the game name is already included as part of the header. The ID was probably excess information.
* Folder – Identifies the folder where the game is located.
  + Information that isn’t used or necessary when learning about the list of servers to join.
* Server Type – Identifies if the server is a dedicated server or a peer-to-peer server
  + Server browser has no setting or makes any distinction between connecting to a dedicated server or peer-to-peer server. They’re all treated the same.
* Environment – Identifies the operating system this server is using.
  + Unnecessary information to learn about when connecting to a server.

It is not mentioned why these parts of the header were removed, but it could be that the headers mentioned above were considered unnecessary and were removed in later versions of the Source Engine or from this specific game.

**A2S\_Player Request packet:**

When the game client presents a list of servers, it automatically sends a set of A2S\_INFO queries to every available server to get the basic server information. A player might want to request more information for a specific server which might pertain to what settings a server has and what players are participating in the session. If a user wants to obtain more information regarding the players, the game client will send another UDP packet requesting the server to provide more regarding the other players. To observe this exchange, I joined an empty server and used Wireshark to record myself requesting the player count of the server I was in.

**A2S\_Player Request Packet:**

Graphical user interface, application, Word

Description automatically generated

**A2S\_Player Response Packet:**

Table

Description automatically generated

The Source A2S\_PLAYER request packet sent when requesting more server information. Note that the data after 0x55 represents the Challenge Number.

Table

Description automatically generatedGraphical user interface

Description automatically generated

|  |  |
| --- | --- |
| **Hexadecimal Code** | **Translated Meaning** |
| 0x44 | A2S\_Player Reponse packet |
| 0x 01 00 | There is 1 player on the server (Only me) |
| 0x 4e 45 54 57 4f 52 4b 49 4e 47 50 52 4f 4a 45 43 54 00 | User on the server: NETWORKINGPROJECT  (My in-game name) |
| 0x00 00 | This player is at index 0 |
| 0x00 00 | This player has 0 points |
| 0x4c f6 46 44 | Unknown: Presumably tied to the duration the player has spent on the server |

* As shown in the two images above,
  + The response packet I received provided my client with details regarding my name, my score, and possibly my current time spent recording on the server
  + Asking for server info and refreshing server info would have my client send an A2S\_PLAYER Request packet and an A2S\_INFO packet for server information and player information
  + Both packets were sent separately and received their own respective response packet.

**A2S\_PING & A2S\_RULES:**

With A2S\_PING being obsolete, no UDP packets containing the matching header were found despite using ping commands in-game and sending ping requests directly to a server. However, while tracking packets, messages containing the A2S\_SERVERQUERY\_GETCHALLENGE header were found. This was in spite that no request packets were sent out of or found on Wireshark. This is because the A2S\_INFO and A2S\_RULES query may also initiate a challenge request number if needed. However, this is unusual as this feature was supposed to be deprecated, and the game I was testing my Wireshark capture on was not on the list of games that still supported this feature[[19]](#footnote-19).

Table

Description automatically generated

|  |  |
| --- | --- |
| **Data captured** | **Translated Meaning** |
| 0x41 | This is an A2S\_SERVERQUERY\_GETCHALLENGE packet |
| 0x 0d 3a ae 8b 00 00 00 00 00 00 00 00 00 | This represents the challenge number used to identify the specific conversation that my game client has with the server |

While the Source Challenge Response packet was still obtainable, A2S\_RULES was an unknown packet that couldn’t be found despite any attempts to send a query a server. Using Steam’s Python API to directly send an A2S\_RULES request yields no response, this protocol wasn’t found in any Source Engine games being tested, and there is no publicly documented method of acquiring an A2S\_RULES packet, so it is uncertain if the protocol has been retired or has a more obscure method needed to request for server rules.

However, in the figure showcasing the master server query protocol via web API, every server displayed a set of server configurations such as game, game mode, security settings, and map name. These additional settings fit the description of what A2S\_RULES was supposed to provide as the Master Server Query protocol itself was initially designed to provide only the list of IP Addresses and port numbers used by every server. It is likely that A2S\_RULES was incorporated with the Master Server Query protocol. Regardless, A2S\_INFO and A2S\_PLAYER are the two main packets sent from a client to a server, and these two packets are most often used to acquire the necessary info displayed the game. When presented with this information, the user will have the necessary information to decide which server to join in, fulfilling the goals of Steam’s Source Engine Networking protocols.

**References and discrepancies**

<https://steamcommunity.com/discussions/forum/14/2989789048633291344/>

<https://steamcommunity.com/discussions/forum/14/2974028351344359625/>

* These were community posts released by one of Valve’s Software Developers, Fletcher Dunn who is responsible for managing Steam’s networking software.
* This post describes changes made to the A2S\_INFO protocol to improve its anti-spoofing resistance.
* These posts describe the changes made to the packet header which are used today along with a description for the purpose for some of the packet changes and explained some of the packet’s anti-spoofing and DDoSing countermeasures.

<https://developer.valvesoftware.com/wiki/Master_Server_Query_Protocol>

<https://developer.valvesoftware.com/wiki/Server_queries>

* This is the main source of information for this research paper as it has the most information regarding the
* These wiki sites were created and curated by Valve Software
* However, the site is managed and edited by community members which may lead to certain inaccuracies and discretion regarding reliability of the information provided.
  + This is further exacerbated by the issue that not all articles were up to date with posts and discussions taking place over 5 years ago.
* Regardless, the site is referenced multiple times by Valve’s employees on their GitHub site for reference, and the employees also manage the site and answer any discrepancies on the discussion page as well which can verify the site’s credibility.

<https://steamapi.xpaw.me/#IGameServersService/GetServerList>

* Used to initiate a Master Server Query protocol via web API.
* A token was acquired from Steam’s web developer request site which was used to send the API request.
* However, any attempts to trace the requests packets on the web yielded no results because there was not enough information regarding what packet to be looking for.
* This site only provides a list of what each web command does and what it provides, but there is no documentation regarding how it works so this site wasn’t very useful for information gathering.

<https://github.com/ValveSoftware/GameNetworkingSockets>

* Because Valve has their engine open to the public, they also made their networking software open-sourced on Github.
* The Github page was used mainly to read the additional notes left behind by the developers explaining what some of their protocols do.
* However, any attempts to use their protocol manually send request packets on Python were unsuccessful because Valve’s servers were updated to no longer accept packets that didn’t originate from the Steam client.
  + This is possibly related to their anti-spoofing and reflection prevention measures.

1. https://help.steampowered.com/en/faqs/view/2EA8-4D75-DA21-31EB [↑](#footnote-ref-1)
2. https://docs.linuxgsm.com/steamcmd/steam-master-server [↑](#footnote-ref-2)
3. https://developer.valvesoftware.com/wiki/Master\_Server\_Query\_Protocol [↑](#footnote-ref-3)
4. https://steamcommunity.com/discussions/forum/14/2989789048633291344/ [↑](#footnote-ref-4)
5. https://steamcommunity.com/discussions/forum/14/2989789048633291344/ [↑](#footnote-ref-5)
6. https://gaming.stackexchange.com/questions/298348/the-server-list-from-source-engine-games-only-show-very-few-servers [↑](#footnote-ref-6)
7. https://developer.valvesoftware.com/wiki/Master\_Server\_Query\_Protocol/Parsing\_packet [↑](#footnote-ref-7)
8. https://developer.valvesoftware.com/wiki/Master\_Server\_Query\_Protocol#Reply\_format [↑](#footnote-ref-8)
9. https://github.com/ValvePython/steam/blob/7ddb7f3eb9d951692a1a01fcdc73638a3034fbc0/steam/game\_servers.py [↑](#footnote-ref-9)
10. https://www.mail-archive.com/hlds\_linux@list.valvesoftware.com/msg67998.html [↑](#footnote-ref-10)
11. https://www.mail-archive.com/hlds\_apps@list.valvesoftware.com/msg00978.html [↑](#footnote-ref-11)
12. https://steamcommunity.com/discussions/forum/14/2989789048633291344/ [↑](#footnote-ref-12)
13. https://steam.readthedocs.io/en/latest/api/steam.game\_servers.html [↑](#footnote-ref-13)
14. https://steamcommunity.com/discussions/forum/14/2989789048633291344/ [↑](#footnote-ref-14)
15. https://developer.valvesoftware.com/wiki/ConVar [↑](#footnote-ref-15)
16. https://store.steampowered.com/oldnews/422?l=english [↑](#footnote-ref-16)
17. https://developer.valvesoftware.com/wiki/Master\_Server\_Query\_Protocol#Challenge\_response [↑](#footnote-ref-17)
18. https://steamcommunity.com/discussions/forum/14/2989789048633291344/ [↑](#footnote-ref-18)
19. https://developer.valvesoftware.com/wiki/Server\_queries#A2S\_SERVERQUERY\_GETCHALLENGE [↑](#footnote-ref-19)