# Implementing Virtual Clients in Quake III Arena

Stig Magnus Halvorsen, Kjetil Raaen



# Implementing Virtual Clients in Quake III Arena

Stig Magnus Halvorsen[1,2,3] and Kjetil Raaen[1,2,3]

[1]The Norwegian School of IT (NITH), Oslo, Norway
{halsti,raakje}@westerdals.no
[2]Simula Research Laboratory, Bærum, Norway
[3]University of Oslo, Oslo, Norway

**Abstract.** Performing server and network experiments on video games can be cumbersome because it usually requires a large number of players to generate sufficient server and network load. A solution is automated artificial intelligence-controlled virtual clients that behave as real players. This paper describes an implementation of virtual clients in the open source video game *Quake III Arena*, which converts the game into an open source tool for generating server load with realistic network traffic for investigating game system scalability.

**Keywords:** Virtual Client, Quake III Arena, Quake 3, Load Generation

## 1 Introduction

Network and server load can be generated through the use of live sessions with real people as test candidates, as it was done in [2]. Alternatively, automated artificial intelligence (AI)-controlled virtual clients, or simply "virtual clients", can be used as demonstrated in [3]. The latter is beneficial as it allows experiments to be done by a single scientist requiring little time and hardware. We want to follow the second approach in our research and in [1], we proposed several open source games as potential tools for scientific work. None are ideal for server and network experiments because no tools are provided for server load and network traffic generation. This paper documents the addition of *virtual clients* to *Quake III Arena* (Q3A) to solve this lack of tools. Q3A is an open source game already discussed in [1]. The modified game can be used as a load-generating tool that scientists may use or alter to test and evaluate their concepts.

## 2 Implementing Virtual Clients

We were inspired by Q3A's implementation of the single player game mode. The design solves a similar problem to ours because it requires some server-side logic within the client to function properly. Single-player mode is started by first launching a local and "hidden" server with a specified map, followed by launching the client logic that connects to the local server.

Our solution is designed to launch from a console, using the engine's built-in *console variable* (cvar) system. A virtual client can hence be enabled by setting

the proper cvars through program arguments, followed by connecting to a server. The virtual client retrieves a message containing the configuration of the server on connection. This is used to create a local shadow copy of the game server, with the same loaded map, entities and configuration. The entire procedure utilizes a modified version of the local server initialization process from the original single player mode.

Once the shadow server is initialized, it sleeps approximately a second to synchronize the shadow server with the game server. The received data from the game server contains the initial position and view angles of the client, which is used to add a bot to the shadow server at the exact same position. The bot library enables the interaction of bot entities with the map. The bot's commands are transmitted to the shadow server in the same way as connected regular clients. All commands are processed in a server function that takes a given command as a parameter. The function is modified to enqueue the received command in the client's command queue. The engine notices the new command in the client queue and immediately transmits it to the game server. Updates from the game server are forwarded to the shadow server by copying the player and entity states from each received update into the bot and entity structures of the shadow server.

## 3 Results

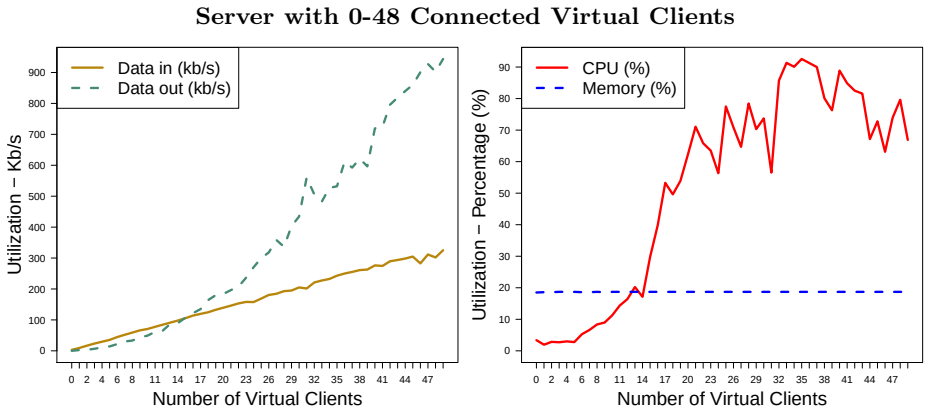


Fig. 1: Network Load

Fig. 2: CPU & Memory Utilized

To evaluate how the virtual clients affect the server, we ran an experiment with 0-48 virtual clients connected to a single, unmodified Q3A server. Figure 1 shows the server's network load. The server's *Data in* graph shows that bandwidth increases proportionally with the number of clients. *Data out* increases quadratically because each client needs updates from all the other clients within its *view area* [4]. Figure 2 illustrates server CPU and memory utilization. CPU utilization increases until it reaches approximately 71% with 21 connected virtual clients. The server starts to struggle since the operating system needs to share its processing capabilities with other processes. Memory usage remains constant as the

game engine always pre-allocates enough memory for the maximum number of supported clients.

Our implementation is capable of running in graphical and console mode. Table 1 shows an average of the various client types' resource usage. The data is the average of ten readings from a host with a quad core 2.6Ghz CPU (max 400%) and 4GB memory. The console version utilizes one thread, while the two others use six. The console client requires significantly less processing power and memory than the others, while network differences are minimal. The console version allows up to 57 instances on this specific host, multiple host machines should be used for large scale testing.

| **Type** | **CPU** | **Memory** | **Bandwidth in** | **Bandwidth out** |
|---|---|---|---|---|
| Regular player | 252.42% | 79 MB | 1.64 Kb/s | 5.80 Kb/s |
| VC Graphical | 234.26% | 94 MB | 2.00 Kb/s | 5.14 Kb/s |
| VC Console | 6.94% | 29 MB | 1.41 Kb/s | 6.00 Kb/s |

Table 1: Various Client Types and their System Load

## 4 Evaluation & Conclusion

Q3A has been modified into a load-generating tool for investigating game system scalability, by implementing virtual clients that produce server load with authentic network traffic. Researchers can implement their concepts into the game and analyze performance differences by utilizing the virtual client support. This can be used to improve the credibility of [3] and similar research.

Q3A is old and may not be advanced enough for some researchers, but is likely a better test platform than small prototypes such as the one utilized in [3]. Virtual clients may be considered complex in comparison to client packet capture and playback. However, packet traces cannot support evaluation of server processing load. The current implementation supports only one virtual client per instance of the software, wasting resources when launching multiple.

## References


1. Halvorsen, S.M., Raaen, K.: Games for Research: A Comparative Study of Open Source Game Projects. In: Euro-Par 2013: Parallel Processing Workshops. LNCS, vol. 8374, pp. 353-362. Springer, Heidelberg (2014)
2. Claypool, M., Claypool, K.: Latency and Player Actions in Online Games. In: Communications of the ACM - Entertainment networking. Commun. ACM, vol. 49, pp. 40-45. ACM, New York (2006)
3. Raaen, K., Espeland, H., Stensland, H.K., Petlund, A., Halvorsen, P., Griwodz, C.: Lears: A lockless, relaxed-atomicity state model for parallel execution of a game server partition. In: ICPPW 2012: Parallel Processing Workshops. ICPPW, vol. 41, pages 382389. IEEE Computer Society, Washington (2012)
4. Stefyn, D., Cricenti, A.L., Branch, P.A.: Quake III Arena game structures. Swinburne University of Technology. In: Technical reports No. 110209A (Feb 2011). Faculty of Information and Communication Technologies. Centre for Advanced Internet Architectures.