The Mammoth Dedicated Server Guidebook

Why dedicated servers are important in multiplayer games, and how to make dedicated server software that provides the best support possible for your games.

Andrew Armstrong
Mammoth Media
http://www.mammothmedia.com.au/
Phone: +61-7-3271-5000
Fax: +61-7-3721-5001
email: info@mammothmedia.com.au

COPYRIGHT AND LICENSING

Please feel free to share this document – particularly with developers of multiplayer games! The goal of this document is to increase awareness into the key issues involved in creating solid dedicated server software to improve the online gaming experience for as many people as possible.

The latest version of this document will always be available from the following URL: http://www.mammothmedia.com.au/resources/.

CONTENTS

1. ABSTRACT

This document aims to provide some information to the game development industry on suggested best practices in both developing and distributing dedicated game server software for current and future games that have a multiplayer gameplay component.

Many newly released games continue to lack robust dedicated server support; hampers the game's overall online community growth and multiplayer success considerably - as with there being no where to play, there is no community to be a part of.

Games that have provided solid dedicated server support from the outset have grown into some of the biggest and most recognisable brands in the world – Counter-Strike, Battlefield, Quake, Call of Duty, Unreal, and Tribes are just some of those that are now household names when it comes to multiplayer gaming. Their work in creating a robust dedicated server product was integral to the success of these games online.

In an effort to eliminate the uncertainty in how to provide a solid dedicated game server application that will be widely accepted by the game service provider community, several best practices and points to consider have been raised and discussed in this article. Several common pitfalls to avoid are also explained.

These points are based on Mammoth Media's many years of experience with working amongst a wide range of game server software and online communities.

2. HOW DO MULTIPLAYER GAMES WORK?

Multiplayer games allow two or more players (or "clients") to interact with each another in the same virtual world powered by the game.

First-person shooter (FPS) games often allow players to move around the same world and shoot at each other in order to score points, while real-time strategy (RTS) games allow players to control an army of units and structures in order to attack and defend against an opponent.

For these virtual worlds to be possible; each player must have their world synchronized with one another so that the movement or actions performed by one player are reflected in every other player's perception of the same world, and is performed using networking protocols over the Internet or a Local Area Network (LAN).

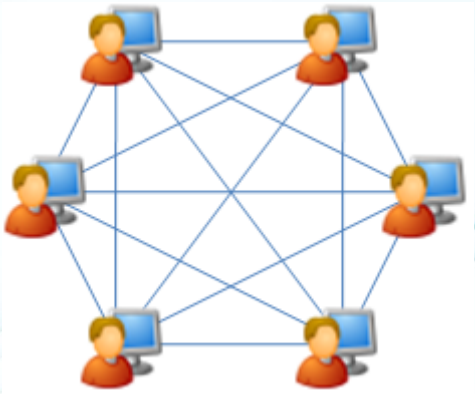
There are currently two major networking models used to keep each player's world in a synchronized state:

2.1. PEER-TO-PEER

Peer-to-peer networking for multiplayer games utilizes a distributed architecture, with each player responsible for keeping every other player within the world informed of their current position and other relevant information.

Real-time strategy games often use this model to allow ad-hoc multiplayer games to be formed between small numbers of players.

Figure 1: Peer-to-peer Topology



While peer-to-peer is generally fine for small groups of players in certain types of games, it doesn't scale very well – once you reach a certain amount of players, the amount of bandwidth you need to send between each player becomes significant. As

most home broadband systems have much less upstream/upload capability (when compared to their download capability), peer-to-peer games can only scale so far before you run into network congestion issues that will quickly render the game unplayable.

2.2. CLIENT/SERVER

The client/server multiplayer networking model features an authoritative server that is responsible for relaying information between each individual client.

Unlike the peer-to-peer model that requires each player to directly communicate with every other player, the client/server model only request each player communicate directly with the server, as the server will relay information between clients as appropriate.

Figure 2: Client/Server Topology

The client/server model is the focus of this document and targets dedicated game servers.

Unlike peer-to-peer games, the limitations on bandwidth are much less of an issue. Instead of having to communicate with all the other players, each client is only sending data to one point – the server. This greatly simplifies the flow of traffic and means it can scale much more effectively.

2.3. WHAT IS A DEDICATED GAME SERVER?

In the context of the client/server model, a game server (or "server" for short) is the centralized authority responsible for relaying network messages between each player (or client) that participates in the same virtual world.

Unlike the peer-to-peer networking model, where each player within the world communicates with every other player, the server within the client/server model

does not necessarily participate within the virtual world itself, and only acts as a communication facilitator between each player.

A player may sometimes be able to "host" a game and become a server, while also participating in the game themself. This is also known as a "listen" server, as the player is in the game world and is "listening" for other players to connect to them. The virtual world is managed by this player's computer and they become responsible for relaying network messages between other players, but must also allow the player to be playing the game at the same time.

A dedicated server, on the other hand, does not participate within the world; and instead operates independently from any player within the game and sit waiting to serve players that join.

Figure 3: Client/Server Topology with a Dedicated Server

Figure 4: Client/Server Topology with a Listen Server

Players often need powerful graphics accelerators, sound cards, and a suitable broadband Internet connection to be able to participate in the latest multiplayer games and to host a "listen" server (as they too are in the game).

Dedicated servers are only interested in managing the virtual world for other players (as the dedicated server itself is not a player within the game), and should not require powerful graphics accelerators in order to run as there is no complex game interface to render, unlike a player who is participating in the game.

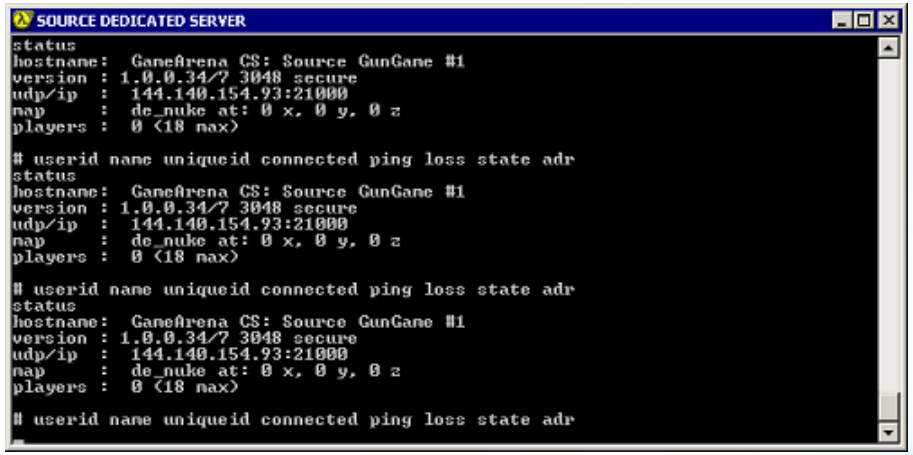
Two common dedicated servers and gaming clients are presented below. The Counter-Strike: Source game server is a traditional console application, while the FEAR: Combat server is a basic Windows application.

You will notice that neither of these two dedicated game server are rendering the graphical game interface, or participating in the world itself. An example for each game server's game client is also provided for comparison.

Figure 5: Player version of Counter-Strike: Source

Figure 6: Player version of FEAR Combat

Figure 7: Counter-Strike: Source Dedicated Server



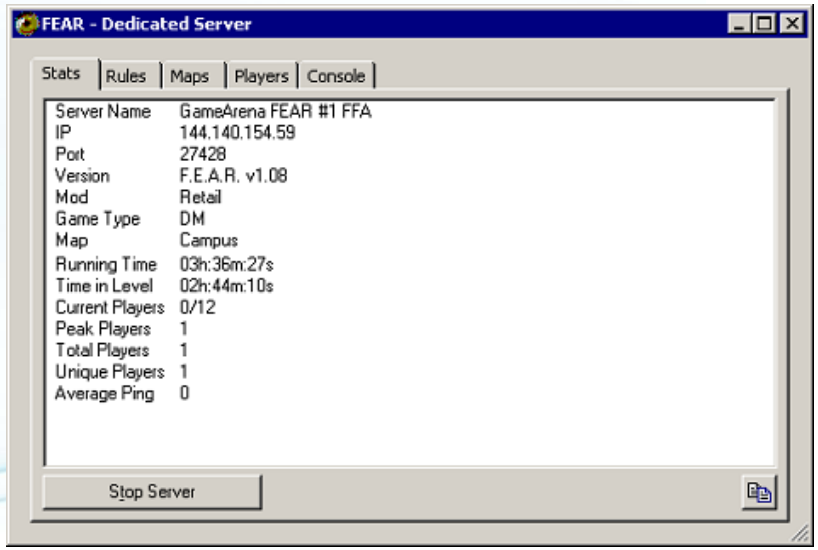
```
status
hostname:  GameArena CS: Source GunGame #1
version : 1.0.0.34/7 3048 secure
udp/ip  :  144.140.154.93:21000
map     :  de_nuke at: 0 x, 0 y, 0 z
players :  0 (18 max)

# userid name uniqueid connected ping loss state adr
status
hostname:  GameArena CS: Source GunGame #1
version : 1.0.0.34/7 3048 secure
udp/ip  :  144.140.154.93:21000
map     :  de_nuke at: 0 x, 0 y, 0 z
players :  0 (18 max)

# userid name uniqueid connected ping loss state adr
status
hostname:  GameArena CS: Source GunGame #1
version : 1.0.0.34/7 3048 secure
udp/ip  :  144.140.154.93:21000
map     :  de_nuke at: 0 x, 0 y, 0 z
players :  0 (18 max)

# userid name uniqueid connected ping loss state adr
```

Figure 8: FEAR Combat Dedicated Server



Server Name	GameArena FEAR #1 FFA
IP	144.140.154.59
Port	27428
Version	F.E.A.R. v1.08
Mod	Retail
Game Type	DM
Map	Campus
Running Time	03h:36m:27s
Time in Level	02h:44m:10s
Current Players	0/12
Peak Players	1
Total Players	1
Unique Players	1
Average Ping	0

Traditionally, only First Person Shooter (FPS) genre games make use of dedicated servers, as there are a high number of players playing together at any one time (8-32+) and an arbitrary amount of players may join or leave the server at any time. One participant is not held responsible for everyone else being able to play.

Other multiplayer game genres such as Real Time Strategy (RTS) games make use of the peer-to-peer networking model, often involving only a small amount of players (8 players or less).

3. WHY ARE DEDICATED GAME SERVERS NEEDED?

Providing a consistently high-quality multiplayer experience that does not suffer from lag (or disruptions in the game experience) traditionally becomes more resource intensive as the amount of players participating within the world increases, which is often the case for First Person Shooter (FPS) games.

In order to support and scale to the large number of players interacting within the same virtual world, a dedicated server can be utilized to manage the burden of synchronizing each player with one another and maintaining the world state.

Resource requirements which include memory, processing power and bandwidth traditionally increase as each player joins the same world, as the server must now be able to process more information quickly and ensure it reaches each player in the fastest possible manner.

The computer systems which run the dedicated game server applications are typically very powerful and come with multiple processor cores, several gigabytes of memory, and are often connected to an Internet backbone to provide very fast connectivity to players in order to reduce latency between network messages, are online 24 hours per day, 7 days per week, all in an effort to ensure the best multiplayer experience for players and provide high availability. The more powerful the physical server, the more dedicated servers you can run on it – thus allowing more players to play online.

Unfortunately, many new multiplayer-enabled games offer a very poor - or non-existent - multiplayer experience to customers because the game lacks a dedicated server application, or the application fails to meet several important but often unrecognized guidelines in its implementation (which are discussed below).

Without dedicated game servers being a possibility, players would be forced to utilize "listen" servers (if supported) to participate in multiplayer games and would be at the mercy of consumer-grade hardware and Internet links, as well as the hosting player's desire to continue playing, in order to interact online.

4. WHO RUNS THE DEDICATED SERVERS?

The vast majority of dedicated servers are run by hobbyists, fans of a game, and professional Game Service Providers (GSPs). A small number of "official" dedicated servers are sometimes run by the game developer or publisher.

Internet Service Providers (ISPs) often maintain a gaming division that operate hundreds of game servers for their customers (free of charge to both the customer and game developers), as well as hobbyists that maintain a handful of gaming servers for their online teams and friends, as well as Game Server Rental (GSR) companies that provide customized dedicated game servers (utilizing powerful resources) to customers for a fee.

In practice, this means that game developers can create a game and have third parties providing almost all the infrastructure support required for gamers to actually play their game online. **This can significantly reduce the costs and management overheads of a multiplayer game.**

5. DEDICATED SERVER IMPLEMENTATION: BEST PRACTICES

While the vast majority of game developer's release appropriate dedicated servers (as they have either had previous experience in the field or have researched the issue in some depth) there are still some games which continue to stumble in providing a suitable dedicated server; much to the distress of the game's online community and Gaming Service Providers (GSPs) who would be more than happy to run many servers for the game.

Mammoth Media (http://www.mammoth.com.au) has been responsible for managing thousands of game and voice servers for several years from both a Game Service Provider (GSP) and Server Rental Provider (GSR) perspective, and has experience with hundreds of different games and how they offer dedicated server support, as well as experience from speaking with other game server providers.

Providing a solid dedicated server implementation for a game can be difficult when the requirements for dedicated server software from the view of server providers is not always clear to developers.

In an effort to assist game developers in their design, implementation and deployment of their game's dedicated server software, several guidelines have been created that are the result of Mammoth Media's years of experience in working with thousands of game servers covering hundreds of games over many years.

These concerns are not only held by Mammoth Media, but also other server providers and hobbyists that use both automated systems and manual interaction to manage their gaming servers.

Several terms are used within each guideline mentioned below. For clarity, the same terminology used in GameCreate (http://www.gamecreate.com), a game server management panel, will be used.

Server: A dedicated game server process. This is an instance of the dedicated game server software application.

Host: A physical computer which runs one or more Server instances.

5.1. IMPLEMENTATION GUIDELINES: ADVANTAGES - THINGS TO DO

Several points to consider when developing a dedicated server are provided below.

5.1.1. SUPPORT MULTIPLE CONCURRENT SERVER INSTANCES PER HOST

Dedicated server software simply runs as a regular process on a computer system. If the software is well-written and optimised for performance, they will rarely consume all the resources (processor/CPU time, memory) of a Host.

Because of this, almost every game server provider will want to run more than one instance of your game Server software on the same physical Host. For example, a Deathmatch and Capture the Flag server may both be created and run separately to provide more options to players in what they would like to play.

In terms of development, this means the game server software needs to be created in such a way that multiple instances are easily supportable by people that wish to run servers.

The biggest technical issue here is that the server needs support to listen for network connections on arbitrarily specified ports. These should be specifiable either on the command line (via a parameter such as "—port=27015") in a configuration file.

Some software also checks to make sure that no other instances of it are running before it will start – obviously such a restriction should not be in place in this case.

5.1.2. SUPPORT TEXT CONFIGURATION FILES AND COMMAND LINE ARGUMENTS

Game server settings are often chosen by editing a plain text configuration file, and are sometimes specified through command line arguments provided to the dedicated server application.

The simplest and most common way to configure a game server and provide all the support necessary to easily manage it is to provide a simple text file which allow all the various configuration options to be set (such as the game server name, what network ports to use, how many players it should support, etc).

An additional benefit of this approach is that the game server software only needs to be installed once; and multiple game servers may run from the same game installation (both reducing disk space consumption and management overhead).

Example': To launch a dedicated server of 'BZFlag' (www.bzflag.org) with a specific configuration, the following command line can be used:

```
bzfs.exe -conf data\server1.conf
```

This will start a new instance of the 'BZFlag' game server and will instruct it to read its configuration information from the file located at 'data\server1.conf'.

The configuration file would contain the server's game and program settings, such as its port number, map rotation, server name, and any other settings used to configure the server.

If the game server software did not support this option (for example, it was hard-coded to read 'server.conf' during start-up), separate installations for each server would be required.

5.1.3. PROVIDE AN EXAMPLE AND SUITABLE CONFIGURATION FILE

Some new games are often released with no accompanying documentation or example configuration files for a dedicated server, making it difficult to both set up a server quickly for players at launch, and to ensure suitable game settings that players will enjoy have been chosen.

Including a simple ''ReadMe.txt'' file or sample, default configuration file with the dedicated server software that describes or demonstrates how to configure and start a basic server with popular game settings, while highlighting important configuration variables (such as how to set the server name, connection port and map rotation), will allow game server's to be set up quickly after acquiring the server software with suitable settings that players will enjoy.

It is important to note that many game servers are often setup by administrators that may not necessarily have the game installed, and they may not have any interest in the game. Players often then communicate back with server providers to tweak game settings.

Game server providers often set up server's using a variety of settings for newly released games on release day, so providing a sample configuration file that indicates which settings players will enjoy most is always welcomed.

5.1.4. DISTRIBUTE FREE SERVER SOFTWARE

While not a requirement (many service providers will simply purchase the game to setup dedicated servers if required) there are many significant advantages offered by releasing standalone dedicated server software for free. This allows more servers to be set up for the developer's game (since it costs nothing to do and is instantly available via the Internet).

Several game developers have released their server software a day or two prior to the retail launch of the game, allowing server providers to prepare for the game's release ahead of schedule and ensure there are enough server's up and running the moment player's visit the Multiplayer menu in-game.

5.2. IMPLEMENTATION DISADVANTAGES: THINGS NOT TO DO

Many service providers have encountered a variety of pitfalls in dedicated server software that has been released over the years that make it difficult or impossible to provide multiplayer game servers, even when the server software has been made available. These hurdles should be avoided.

5.2.1. DON’T REQUIRE A CD/DVD IN THE HOST'S CD/DVD DRIVE

Dedicated server software that requires a physical CD/DVD of the game to be in an optical drive to run almost guarantees that the majority of server provider's will not be able to run even one server for the game.

Unlike a player's computer that includes a CD/DVD drive as a standard commodity, the powerful computers located in network data centres that are responsible for hosting the majority of dedicated game servers are mostly always slimmed down, raw grunt machines that do not contain CD/DVD drives, powerful graphics cards or even sound cards. Instead, these machines focus on raw processing power and memory while keeping their physical dimensions as small as possible in order to accommodate more physical hardware within the data center.

5.2.2. DON’T REQUIRE A GRAPHICS ACCELERATOR TO RUN

As previously mentioned, the majority of the hardware that are dedicated to running game servers do not even have the physical space for a graphics accelerator (as they

should not need one), and only contain enough graphical power to render the standard Windows or Linux interface. If the dedicated game server software requires Pixel Shaders or Dynamic Lighting affects provided only by graphics accelerator cards (for example), it will not be possible to actually start a server for the game.

As most servers are operated and controlled remotely (using Remote Desktop to access the Host from over the network, for example), the simpler the interface, the more accessible it will be. A simple text-based console is probably the most efficient.

5.2.3. DON'T REQUIRE UNIQUE CD-KEYS FOR EACH GAME SERVER

When each game server requires a unique CD-Key per server instance, that may not be shared, it will impact on the amount of server's that may be provided by each provider.

Many server providers have automated systems that can automatically start game server instances on demand. For example, a gaming team may want to temporarily use a server for a few hours for a casual game, and that server can be created and shutdown for that time period. This would not always be possible if the amount of CD-Key's available to the provider is limited.

It's important to ensure this restriction is definitely not present for retail games that do not offer free dedicated server software. Requiring separate purchases of the game to run more than one dedicated game server would severely restricted the amount of servers available to the game's community, as it would be expensive to create additional servers for community members.

This extends to other authentication systems, such as login accounts or third party software required on each server. The dedicated server software should be as self-encapsulated as possible, bearing in mind the goal is to have it run on as many systems as possible, as widely as possible – giving your audience as many options as possible.

5.2.4. DON'T STORE CONFIGURATION FILES OUTSIDE OF THE INSTALL DIRECTORY

Reading and writing to game server configuration files that exist outside of the game's installation directory (which may be for example C:\servers\mygame) can be problematic.

Game servers are usually not run with Administrator or root privileges; and are restricted in what directories they may access. It also becomes confusing when game servers are making changes outside of their installation directory and prevents server providers from only having to manage a single directory related to the game.

5.2.5. DON'T CHANGE PROCESS ID AFTER STARTING

Game server management panels and other monitoring software often keep track of game server instances by monitoring the server processes they originally started, by remembering the Process ID returned by the operating system when the server software was launched by the monitoring application.

In some games, the Process ID will change after the initial process was originally started, because either the launched process has spawned the 'real' server process after it (as the initial process was simply a wrapper or shell script), or with each map change, the game server software restarts itself.

This can make it difficult for automated monitoring software to know what is happening – a monitor that is just watching the processes will see that the first process has exited and might try to start another instance – which will of course lead to many more servers starting in a cascade effect.

In the event that your software must change processes, it is important to keep track of the new process ID. A common method of doing this is by writing a pidfile (process ID) file into the game server directory.

NOTE: The vast majority of game server software does not change or launch a separate process after starting the initial application, and so this feature would not be necessary for these applications as the PID located in any pidfile would match the PID originally captured when the process was initially started by the monitoring software.

However, as an example, the multiplayer game Soldat will start a new process (and terminate the old one) whenever the map changes, causing a new Process Identifier (PID) to be generated. The game server monitoring software should be informed that the Process ID it should be looking at has now changed in order to keep track of the correct process.

Most game server management panels (and other monitoring software) support what is known as a Process Scoreboard file (often seen with a .pid file extension), in order to track the correct server process if the game server software reports it has changed.

The Process Id File (or "scoreboard" file) is a simple plain text file whose only content is the launched server processes Process Id:

```
1234
```

It is important to note that if the Process ID does change, then the game server software must update this Process ID file with the new value Process ID value, providing any monitoring software with the new ID of the process to track.

In order to support this feature, and keeping in mind multiple server instances should be supported, a configuration option for the server to create a PID Scoreboard should be provided.

'Example': To create a server for 'Soldat' which has a server executable named 'soldatserver.exe' using the configuration file 'server1.cfg' and storing a Process Id in a file named 'server1.pid' the syntax would be:

```
soldatserver.exe -c soldat1.cfg -pid server1.pid
```

This allows multiple servers to be started, each with a separate Process ID file (along with its individual configuration). Of course, this option could be set within the configuration file if that is more suitable.

5.2.6. WRITING SETTINGS TO THE WINDOWS REGISTRY

Game server software should not attempt to read or write configuration settings to the Windows Registry in order to manage the server's configuration.

Server software may be completely removed from its original machine and moved to another for backup or re-location purposes at any time.

If the software is reading server configuration information's from the registry, then this information will be lost if the server is ever moved to a new machine (for example, if the computer the server is running on requires maintenance).

It is standard practice to only configure game server software through configuration files that are written in plain text. Attempting to store configuration information in the Windows Registry is not desirable.

5.2.7. PROMPTING FOR USER INPUT

Asking a question of the server provider during server start-up or later on throughout the server's operation is not desirable, as game servers are often started by an automated process and left unattended with no one monitoring the computer's desktop interface.

For example, if a server process was not shutdown cleanly and upon starting again it had detected the previous failure, it should not ask whether it should enable "Safe Mode" and require a person to click Yes or No to a dialog box or type in to a console in order to continue loading the game server server application.

Most dedicated servers are started by a game server management panel, via a Windows shortcut, or Linux crontab during start-up.

The server application should never prompt for a user to interact with the application in order to proceed, as no one will be watching.

If the error is unrecoverable (such as the specified map could not be found or no configuration file was loaded), then a dialog (or console error) that indicates the reason for failure would be acceptable; as the error does really need operator intervention to correct.

The traditional approach taken for unrecoverable errors is to write an error to the server's log file and console window followed by the application terminating.

5.3. IMPLEMENTATION GUIDELINES: USEFUL INFORMATION AND TIPS

Provided below are other useful tips and information that should be useful to consider when developing game server software. These tips have been derived from interacting with hundreds of different game's dedicated server software.

5.3.1. REVIEW CPU AND MEMORY USAGE

More servers for a game may be created on the same physical computer depending on how well the server software copes while running.

Traditionally most game servers are quite idle on CPU (0-5%) when the server is empty, and may scale up to 80%+ CPU usage during high load of 32 or more players, depending on the game, number of players and the map in play.

Generally speaking, a server operator will want to try to make sure that the average maximum CPU usage doesn’t exceed around the 80-90% mark, which will ensure the best performance possible.

There are a handful of games whose server software idle at near-maximum CPU usage (90%+) which restricts the amount of servers that may be run for that particular game, as even when the server is empty, the amount of resources consumed can be large.

There are many technical reasons that game servers use this much CPU. Generally speaking it is simply the game server software not being correctly coded for the shift from being a game – where the goal is to run as fast as possible all the time and use all the performance available – to being a server, where the goal is to use as little CPU as possible to do all the required tasks.

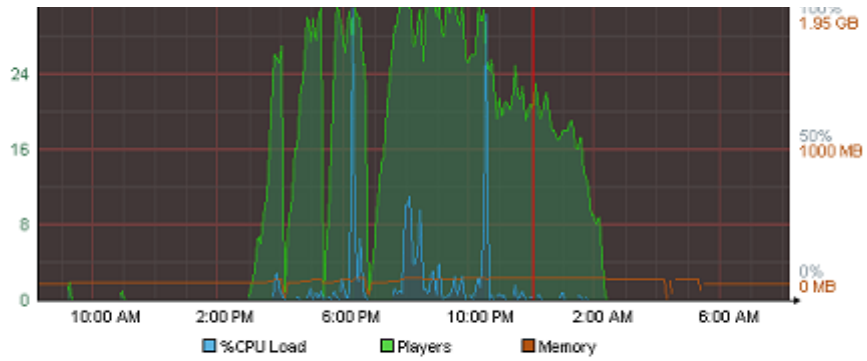
Memory usage between games varies, however it is traditionally constant at 10 - 200MB per instance from start-up. Memory is less of a concern when compared to CPU usage, although a common problem is memory leaks.

Memory leaks can cause performance problems and issues for people running multiple servers on the one Host. Leaks are more typically a problem in newly released software that hasn't been fully tested in a production environment (another advantage of widely spread dedicated servers – many more testers!). They are often easily identified by server management software like GameCreate which provides graphs that clearly show memory usage continuing to grow.

Below are some performance graphs for several different multiplayer game servers captured from real-world data, showing the overlay between the amount of players on the server, CPU and memory load.

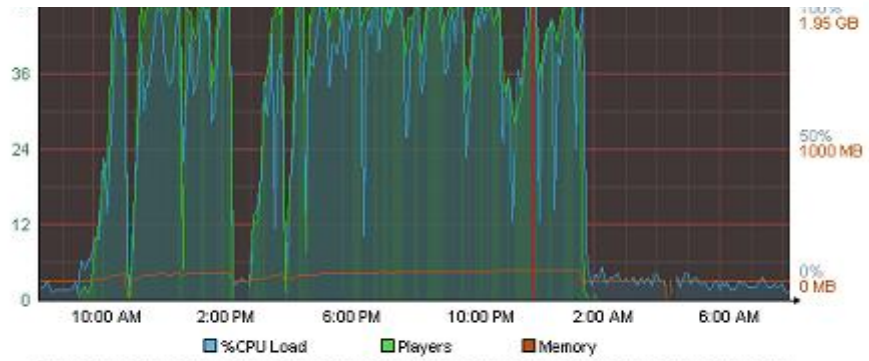
Different games respond differently to the load, as well as the type of map and other variables in play.

Figure 9: Performance Graph of a 32 Player Battlefield 2142 Conquest Server



%CPU Load	Average:	2.17%	Maximum:	100.00%	Current:	0.00%
Players	Average:	10.12	Maximum:	32.00	Current:	0.00
Memory	Average:	123.47 MB	Maximum:	147.00 MB	Current:	108.89 MB

Figure 10: Performance Graph of a 48 Player Battlefield 2142 Titan Server



%CPU Load	Average:	51.22%	Maximum:	100.00%	Current:	4.70%
Players	Average:	25.64	Maximum:	48.00	Current:	0.00
Memory	Average:	156.20 MB	Maximum:	200.00 MB	Current:	127.64 MB

Figure 11: Performance Graph of a 20 Player Counter-Strike: Source Server

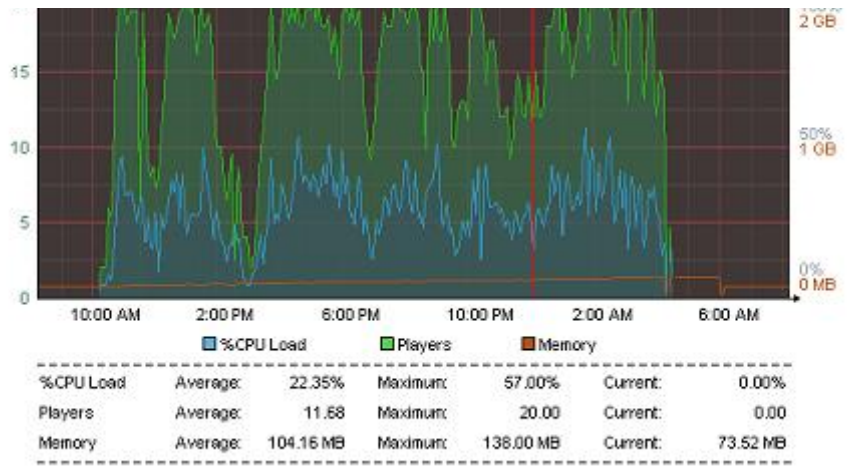


Figure 12: Performance Graph of a 64 Player Battlefield 2 Server

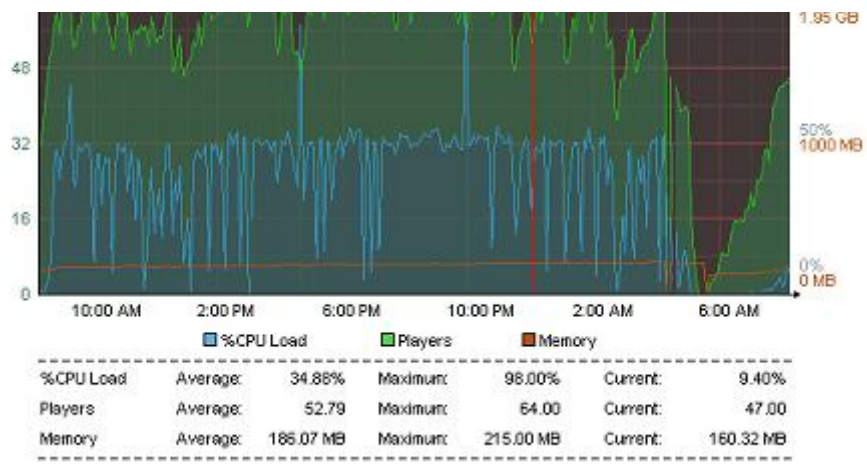


Figure 13: Performance Graph of a 48 Player Battlefield: Vietnam Server

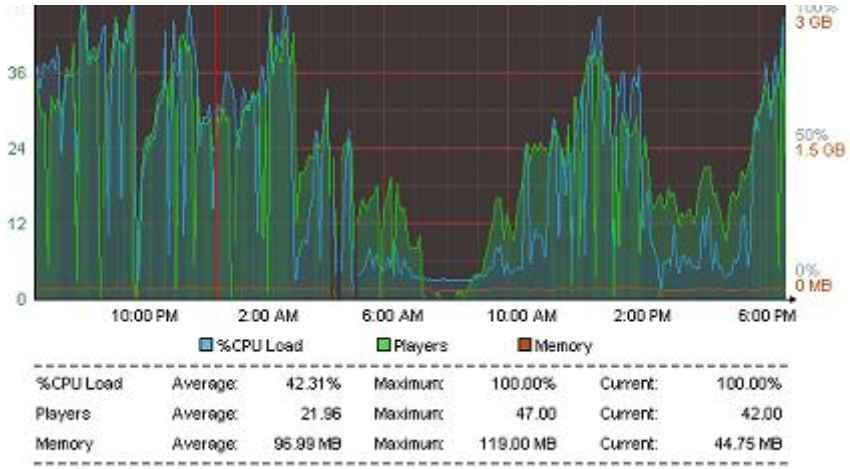
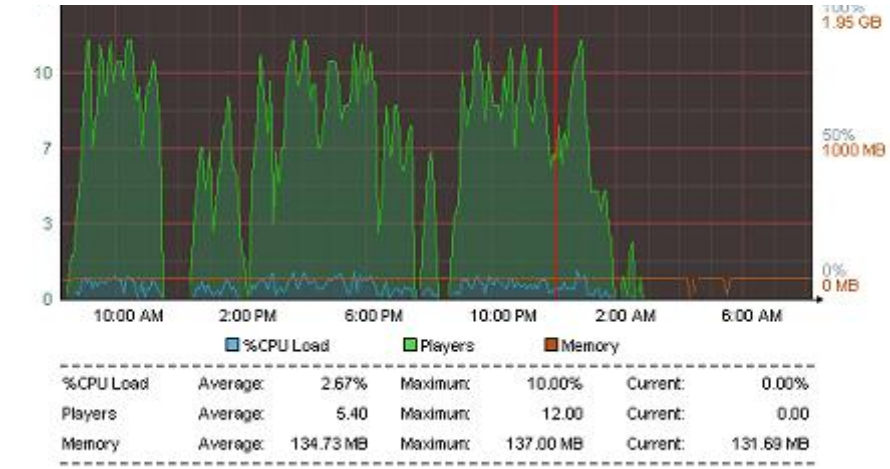


Figure 14: Performance Graph of a 14 Player Call of Duty 2 Server



As demonstrated, some games like Battlefield: Vietnam have performance problems, one the player numbers exceed 24 players, the CPU usage skyrockets.

Battlefield 2142 also has a comparison shot between a Titan based game server (which uses a large amount of CPU power) compared to a Conquest server, demonstrating how a difference in game modes within the same game software can result in significant differences in resource consumption.

Call of Duty 2 uses a very small amount of CPU power. It is an older title and thus is a "simpler" game, requiring less computational power.

While there are no specific constraints surrounding CPU and memory usage for game servers, it is something to keep in mind, as the resource requirements for each game server will impact the overall capacity that may be provided. The more optimisation time spent on dedicated server code, the better it will run – meaning server operators can fit more players per Host, which translates directly into better coverage for your player base.

Note that each server shown has been automatically restarted at 5:00am, which explains the sudden drop in statistics.

5.3.2. EXTERNAL SERVER QUERY PROTOCOL

Query protocols are used to request some public information from a running game server, such as its current map, name of the server, list of players, and any other meta-data information relevant to the game. They are usually returned as a key/value pair (for example, "players=8").

Nearly every multiplayer game supports an external status query protocol.

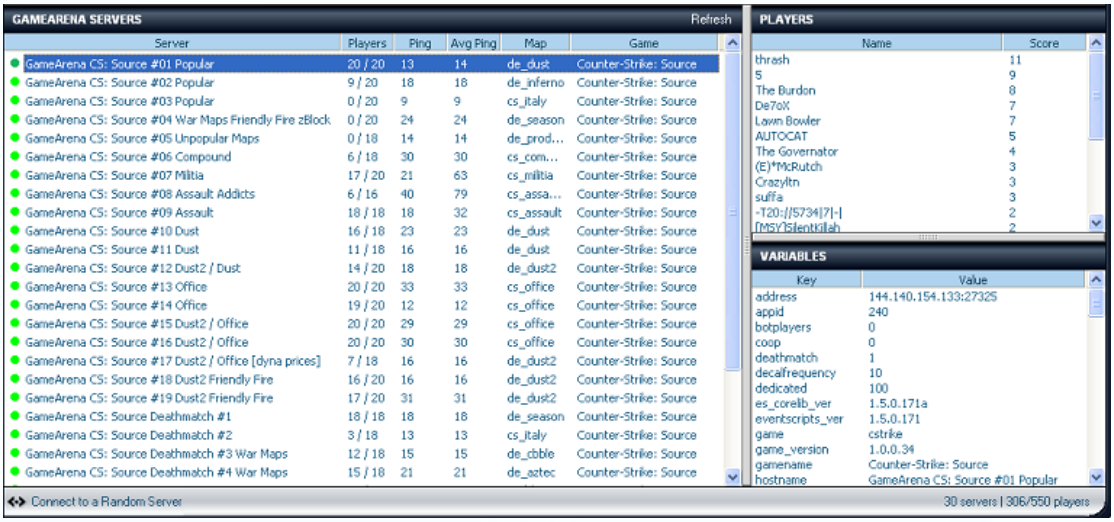
Players often see this information when browsing for servers using the in-game server list or through game server browser tools such as HLSW (http://www.hlsw.de). Game Service Providers and community websites often display this public server information automatically on their websites.

Query information allows players to quickly and easily see what game servers are active and to check whether their friends are online without having to launch the game first.

There are a large variety of game server query protocols that have been developed, and many games re-use existing protocols (such as the All Seeing Eye (ASE) or GameSpy protocol), or end up creating their own.

However, an advantage to implementing an existing query protocol is that existing server browser tools and query frameworks such as Qstat (www.qstat.org) that already support the majority of these protocols can talk to your game server without any modifications.

Figure 15: Partial screenshot of COGS (Complete Online Gaming System) In Action



Mammoth's COGS (and many other server browsers such as HLSW and XFire) are able to view the list of players and server details in real time, externally from the game, through the user of the query protocol.

This allows gamers to find a server that is populated and has the desired settings prior to starting the game itself. They can also browse servers looking for the names of their friends (or enemies, as the case may be) so they can join and play together.

Many game server management panels automatically query every game server they are managing to ensure it is responding (to ensure uptime) as well as to record the utilisation statistics previously shown.

5.3.3. REMOTE ADMINISTRATION TOOL

Many games (such as Counter-Strike and Battlefield 2) provide the ability to remotely administer the game server software in some fashion without having direct access to the Host computer directly.

Often referred to as 'Remote Console' or RCON, players enter a secret administration password (defined previously in a server configuration file) to the server in order to authenticate themselves, and are then entitled to remotely execute game-server console commands as if they were at the server console directly, or perform administrative functions in-game, such as kicking troublesome players or changing the level without a vote being called.

The ability for these trusted users and server operator's to remotely interact with each dedicated game server helps to ensure the game play experience on the server is enjoyable for players by removing trouble makers, switching to a new map in a tournament environment without needing to call a vote, or simply placing a password on the server to restrict who may enter the server.

5.3.4. UNIQUE PLAYER IDS

Managing players is an integral part of running a game server, including the ability to identify them, remove them from the server in case they are misbehaving, and to ban them (permanently or semi-permanently block their access to the server).

The most common method of providing this sort of functionality is to generate a Unique ID for each player. These Unique IDs are usually based off a CD key or product key which is provided with the game, unique for each purchase – the player enters them when installing them and they are mathematically altered to create an ID that is visible to server administrators and players.

Without such a system, it is almost impossible to ban troublemakers. This can greatly affect the community, as serial pests will go out of their way to ruin the game for others (just because they can).

A tangible benefit for developers and publishers of such a system is that a player that is widely banned off many servers simply has no other recourse to buy another copy of the game (assuming he wants to keep playing).

5.3.5. SOFTWARE INSTALLATION/UPDATE ISSUES

Installation and updates of game server software is often distributed using three main delivery methods:

5.3.5.1. AUTOMATED SERVER UPDATE TOOLS

Some software provides automatic package management to download, install, and update game server software.

HldsUpdateTool is an example of such a system. Developed by Valve for selected Steam-based multiplayer games (http://www.steampowered.com), this tool automatically downloads and updates the game server software installation when ran from the command line, providing a painless software management experience for server providers. No Administrator access or Windows Registry access is required for this tool to run.

Providing such a system for your software does make maintaining easier for both you and the server administrators, but there is obviously more effort required to create the required tools and systems to manage such a system.

5.3.5.2. ZIP/ARCHIVE FILES

Many game servers are simply Zipped up or installed as part of a self-extracting executable (with no real Setup Wizard), and requires no Administrator or Windows Registry access to unpack its contents.

This method is extremely flexible for server operators.

5.3.5.3. WINDOWS INSTALLER PACKAGE (OR SIMILAR)

Several game server installs are often distributed within a self-extracting Windows Installer package, with a Setup Wizard being used to install the software to the computer, and often requires Administrator privileges to install while also writing configuration information to the Windows Registry.

While the HldsUpdateTool and ZIP/Archive Files approaches to software delivery are the easiest to use, the Windows Installer Packages are also suitable, however the following points should be kept in mind when creating the Windows Installer (or similar) installation and update packages:

1. If possible, an option to NOT include any Desktop Shortcut, Start Menu entries, etc, associated with the software that will be installed, as these are the options that generally require Administrator access to configure.
2. Many Setup Wizards attempt to locate the original software installation when trying to perform an upgrade. At times some game installations may need to be patched that have been directly copied from the original computer the software was installed on. Installer programs often search the Windows Registry to find the installed location, only to fail with an error similar to "Cannot find the original installation" and promptly exit. If possible, providing an option to simply search for the on-disk location to the original install to patch would be fantastic.

6. SUMMARY

This document has attempted to provide some useful information with underlying reasons to best-practices related to game server software design, implementation and deployment from the view of server providers to help ensure future game releases receive all of the server support that providers have to offer and support the game's underlying community.

7. OTHER RESOURCES

GameCreate - http://www.gamecreate.com

GameCreate is a game server control panel allowing remote control management of game servers. It is an enterprise-level tool targeted at large GSPs and ISPs, but can be used to manage as little as a single server efficiently.

Mammoth Media - http://www.mammothmedia.com.au

Mammoth Media have provided commercial game server management services since 2001, and for several years prior to that in a non-commercial environment.

8. DOCUMENT HISTORY

Date	Author	Changes
2009-11-13	David Harrison	Formatting, minor additions, CC licensing.
2009-11-05	Andrew Armstrong	Wrote document.