Mankind Worlds
Technical Document

Revision 0.1.1
04/20/2006

To be rewritten and extensively expanded by the project lead programmer / technical director.

Table of Contents

Section I: Introduction

Mankind Worlds is trying to obtain a different style of server architecture than seen at common MMOG… it's a game of real globality.
One of the key discussions since Mankind 1 got expanded to the Asian market was, if a Separated universe just for Asian players might better fit the different cultures and player expectations. This is open for further discussions but even a 2nd universe just for the Asian market does not change the initial idea behind - a dynamic server clustering methods gets used to organize and manage one universe for all players… everybody is playing at the same playground.

Section II: Game Architecture

Server

The server structure gets divided into 4 basic parts that do cooperate together:
* database
* world environment
* units (any ship, tank or structure) environment
* chat, VoIP (Voice over IP)

All those are defined to be independent server clusters to cooperate together with each other. The aim is to get load better distributed and harder to break the entire performance in case one part gets heavily loaded. Additionally, a shutdown of the chat/VoIP server does not mandatory result in a complete universe closure. The aim is to be as independent as possible with each system to be not only able to maintain each part separately but also do not require to shutdown the entire system (and as such require to kick all players) when maintaining one part of the server system.

database environment
The database environment gets defined as a separated cluster of servers and is based on an open grid solution like from Oracle (just as example).
It's an environment where a database is no longer based on one physical machine and performance no longer gets defined by hardware power only.
Databases get shared on multiple physical computers. Storage gets split - combined with included RAID systems it is very secure. Machines can be taken out while the entire system is running so there is no downtime or interruption of service because of this. Data gets stored on multiple physical hard drives so if one does fail the entire system keeps on running and hardware can be replaced by hot-plug without requiring bringing down the entire cluster.
Moreover, if increasing subscription numbers will require more and more calculation speed, no endless and cost intense high-end hardware upgrade is required mandatory. Mid-end computer hardware is cheap and quickly available.
Anyway, such a dynamic and open-end system might be too cost intense to get started and for licensing. A detailed study on cost, effectiveness and expected demand from the game should get done to evaluate this… additionally, network and communication speed limits must be seriously taken into account.
However, a dynamic and most important, open end database environment seems to be mandatory to fit requirements and possible later highly increasing demand on performance and unexpected maintenance duties.

world environment
It gets defined as an open-end cluster of computers that a special server software is

running on. Load gets balanced automatically and therefore process calculations get shifted to every computer within automatically upon demand. While this cluster does not store any data (except for caching) it does independently calculate the dynamic world environment. This does include all planet movements and all effects the world (gravity, weather effects, gas clouds and so on) has on any unit (player or computer controlled ship, vehicle or structure) inside the world.
So for example if any ship does move into a gas cloud this world cluster does give back the various kinds of effects it does represent and their strength. The units cluster then can calculate possible effects on the ship based on ships equipment and such.
As these required calculations are rather small and simple the required cluster will be likely smaller as well.
Additionally a strict load and speed calculator must be present to make sure there is no action without the environment server replies by time. There can't be any unit action. Without any effect just because the environment server is overloaded and can't give environment data by time

units environment
It gets defined as an open-end cluster of computers that a special server software is running on. Load gets balanced automatically and therefore process calculations get shifted to every computer within automatically upon demand. While this cluster does not store any data (except for caching) it does independently calculate all units (player or computer controlled ship, vehicle or structure) actions. This does range from movement and battle calculations (weapon impacts/damage) up to any calculation any device on any unit might do on any other unit (like scanning, radar, towing, transfers and so on). Expected load is high so an open-end architecture to easily upgrade for increasing performance requirements is necessary.

chat, VoIP environment
It gets defined as probably only one high performance server. The main duty is to handle chat traffic and redirect VoIP requests.
As VoIP does require a specific server to handle channels the system is designed to be client based. This does mean that any client can open a private (password protected) or open VoIP server which does then run on the client's local computer.
For example, a client could open a guild VoIP server where the "chat, VoIP environment" then does only store the restriction list and broadcast to all guild members so they can join.
On software an established system like TeamSpeak or Ventrillo might be enough instead of doing a self development… licences and their price must be investigated.

Client

Dataflow between client and server gets secured as required and if possible encrypted and compressed. Expected amount of units within visible range might be large and a minimum of 56kBps x2 should be aimed. However, the nature of Mankind Worlds as a build up strategy game does make it a complex and critical aspect of the game and 56kBps x 2 might be not fast enough, in which case a study on the gaming market should evaluate possible valid higher minimum speed requirements… based on technology and hardware available at release.
Moreover, several additional services are planned (like a small separate camera window to track unit movement while doing other tasks) to allow players with higher connection speed (and/or broadband) to have more in-depth services within the game.

As another aspect, client and server communication gets fully handled dynamically. There are no "zones" inside the game which does mean that the player doesn't has any

"expected" waiting time while moving through the universe.
To gain this aim a specific packet and request priority handling gets suggested.
This is to deal with expected "high load" situations to define which data to load first. As an example, if a player does move through the universe and stops within a specific area where a battle is raging on, it is essential to get the basic unit data first like unit type and equipped weapons data to let the player select his units as soon as possible to give orders. Other information like appearance (unit color, placed texture or logo and such) can follow later.

With this said, the entire client is based on a window thread system. This does mean that basically no action from the player should break (lag) the client. If you open a new window, a place holder gets displayed until window graphics and objects as well as text strings are loaded. After, a load meter gets displayed until all required data got fetched from the server to fill in the window with the requested data. Additionally, windows can get minimized to some kind of taskbar – further details on such must get defined and discussed.
Additionally, no client error should lead to a complete client crash. An in-game crash and exception handling as well as problem reporting system gets suggested so an error might break a specific action but hopefully not the entire game.

hardware requirements
The aim is to use latest technology on graphic cards on release. A dynamic scaling of display quality is suggested to get good frame rates even if number of displayed objects does drastically change. The aim is to be able to set all game settings to maximum and then can play at an average frame rate of 25 – 30 fps on latest top gamer hardware on release date… and can set all game settings to minimum and can play at an average frame rate of 25 – 30 fps on hardware at least one generation older than the common one the time the game gets released.
Mankind 1 did show massive problems with client load and server request times so the aim is to develop a dynamic polygon reduction system based on different texture packs. As such, the player does select a specific texture quality and polygon count gets adjusted dynamically based on this to keep an average frame rate. Additional options like LOD, shading, lightning and more can be setup as well… as much it is reasonable for the player and does reflect valid improvements on FPS and/or render quality.

Section III: Maintenance

Server maintenance is a critical task for the operating company of Mankind Worlds. Unavailable servers do mean that customers can't play and because of that the company might lose customers and money.
To avoid this, it is common to do weekly (one day every week) server maintenance works and bring down all servers for several hours. This does allow keeping hardware and databases clean and stable and does give time to replace hardware and do small server and/or client updates if required on a regular base which does allow planning and organizing such very well.
As stated at Section II: Game Architecture, the server environment of Mankind Worlds is dynamic and open. This does allow doing server maintenance works on the fly. Servers get shut down one by one on a specific day every week and after maintenance brought back online – then the next server does follow. Doing so does allow keeping the whole universe online and does not require to completely closing it for any customer. Those might just expect a slowdown on the game world or maybe closing sections of the Universe step by step (and reopening it before the next section gets closed) for a few hours but can still play.

Additionally, a complete shutdown for a regular and detailed maintenance is scheduled once every month for example.Section II:

Section IV: Security

Security of any MMOG server and client is a very critical issue and must get a very high priority at any game planning and coding stage.
Customers will try to get access to the game servers directly and manipulate the client in any way that can get imagined. Their intentions and reasons to do so are different but can do high damage to the company and player base as well as to the game economy anyway.

To achieve their goal there are several different ways known that established them self as ways to misuse a MMOG:

- trying to hack into the game servers itself
  firewalls, anti-spy software and regular log studies can prevent a lot of damage

- trying to hack a players computer
  can't be fully avoided as the customer is responsible himself for the security of his own computer; however, it can be made sure the customer can't get hacked through the game client and locally stored data can get secured as good as possible (like encryption) to make it harder to get the plain data even if any attacker does make it into any customers local computer

- client hacking (on hard drive or directly in memory)
  It is known to be a common strategy to hack and manipulate the client either at physical files on hard drive or directly within the computers memory to manipulate them and fake client actions and or responses (client <-> server communication) to fool the server… that's why never any relevant calculation or verification should get done on the client. The client is within enemies hands and can't be controlled… and tracking down client hacking is often very difficult or even impossible.

- client <-> server stream hacking / packet sniffing
  Means to monitor the client <-> server communication while the game is running and trying to alter it to fool the server and fake actions or even just communication packets. Some kind of encryption of the data stream between client and server is highly recommended.

- trying to misuse the game design, coding issues and server behaviour

  - time cheat
    manipulating clients local windows and BIOS clock

  - intentional client and/or server crash
    once a client and/or server crash scenario got found which can get reproduced by the player at will (or even accidentally but does occur often) he might misuse such to get advantage of possible exception situations
    As an example a server crash might force a rollback and data restore from a 3 minutes older state for security reasons where players can give away stuff within those 3 minutes to another player so after the crash and rollback they have doubled their assets (one from backup character and one from the other player they gave their stuff just before the crash).

- macros, botting
  Means to use programs like mouse and keyboard fakers to simulate any mouse move, mouse button or keyboard key press to automate movements and orders. This can't get prevented and also often hardly detected so any game conception should keep in mind that such is possible – barriers to make this harder or useless should be kept in mind.

- Simply using a coding bug or design flaw to get any advantage
  like misusing a distance detection error where no distance check gets calculated at a specific situation so players can stack as many ships to one spot as they like.

All of the above does require carefully planning and coding the client and server software and also prepare for any attacks by making sure as much as possible can be tracked and identified.