AFS Design (draft 2) with feedback from Tim Blokdijk
AF/Tom Nowell 31/07/07

Contents

1 : Introduction

This document describes the basic design of AFS. Its intended for anyone with an interest to developing or using the system beyond the level of the basic user/player.~~Design Goals~~

~~This document describes ... it's intended for people that ...~~
~~AFS is intended to address the original TASServer implementation.~~

~~The general design goal of AFS is to accomplish all lobby related needs in as generic and flexible a way as is possible, in order to future proof the program and simplify its construction.~~

1.1 :

1.2 : The Problems of TASServer

1.2.1 : The Causes

TASServer and tasclient were designed with a utilitarian approach which suited the needs of the time. The C# lobby Jouninkomiko created, was prone to crashes and was incomplete ~~due to his lack of knowledge in the field~~. Betalords tasclient was intended to provide a lobby that simply worked without major errors.

~~-~~However the basic model was not future proof and it wasn't changed. The fear that 2 lobby systems would split the community or that a fork of tasclient could do so prevented early development. It was feared a split could destroy the spring project at such an early stage.

1.2.2 : Inconsistencies

Because tasclient was the only client initially intended for TASServer, numerous inconsistencies exist, aswell as many quirks with regards to how to operate within the protocol that aren't initially clear. For example, team colours are stored in an integer and not in 3 separate rgb values, and they're saved AABBGGRR not

RRBBGGAA. Many other pieces of data are stored in bit fields, with other commands not using bitfields at all. These bit fields have themselves imposed restrictions on lobby and mod development such as limiting the number of factions a mod can have because the number of bits allocated to store the side number isn'tisnt large enough.

1.2.3 : Hard Coded Parameters and OTA Settings

TASServer also assumes an OTA setting. Options such as Limit dgun to starting position, or diminishing metal maker returns make no sense in a mod with no dguns or metal makers, such as expand and exterminate. There is also a lack of support for arbitrary mod specific options that don't apply to the TA setting.

1.2.4 : Limited to the Spring engine

The current lobby design is limited to the Spring engine, AFS would try to support several engines with one lobby. Where lobby modules would be made for each engine/content combination.

1.2.5 : No Generics

TASServer has no generic commands of any kind. New lobby features cannot rely on custom lobby specific protocols, and require the functionality be directly hard coded into the server itself.

1.3 : The Solution

The server itself will simply act as a traffic routing mechanism for the modules and clients connected. The base protocol used by the server will have no context, the context of which will be provided by the sub-protocol used by the end clients handling the data.

2 : Basic Design

AFS will be composed of a basic traffic manager, routing a small group of base protocol commands and managing logins and logging out.

2.1 : Languages and development

AFS will be coded in Java with the option of native module implementations using JNI for the interface. A co-server running somewhere else as a client could run lua based modules using the LuaJava implementation in order to provide smaller custom modules so that the main server is not affected by errors or problems that might arise. Such a co-server would connect to the server and simply extend itself and forward addresses within itself to the main server so the user is presented the addresses as if they were a part of the main server itself, the co-server being a silent partner exchanging data.

It's expected that Sun Java 6 would be running the implementation and as such it would be feasible to suggest eventual ruby python or Javascript support, however that's a long-term suggestion that isn'tisnt a priority,

In order to speed development, the Quick Server library will be used, which will provide stable tested multi threaded TCP connection support. This way work can begin immediately on the servers data handling infrastructure.

2.2 : Data Storage and Tables

All data -will be stored in basic key,value tables with no predefined data. The data these tables store will depend on the tables name and context, and will be directly accessible to the components of the lobby system by name. Tables themselves will be keys in a 'root' table, and can store references to other table keys to save memory. This could be considered primitive type of database. As the tables themselves will be a key in the root table, sub tables will be definable, although really they would simply be individual tables in the root table that're linked to in the parent table using a reference. Sub tables and keys would be referenced using the dot '.' separator, so “table.key” or “table.subtable.key”, the latter of which would be the key “key” in the table “table.subtable”.-

2.3 : Lobby Services and Other Functionality

The lobby service implementations will run in self contained modules, each acting as a plugin in the overall framework, and can be independently updated, restarted, or stopped. Channels, battles, and player properties shall be handled by such modules, with the server itself simply acting as a traffic router. The modules themselves will have addressable names just like any user name, with an 'address' being able to receive messages or send them. This way a battle manager module can present individual addresses to battles rather than complicating its own sub protocol.

2.4 : Networking

The protocol will use newlines for each message like TASServer to separate messages, the separation of parameters within commands however has yet to be decided. TCP is being used, however UDP services may be offered for NAT via server modules.

3 : Multiple Protocol Support

To support multiple connection/lobby protocols, each users traffic will be tagged as belonging to an ID representing a supported protocol, the default protocol simply passing data on to be routed to the necessary components. A module will be able to define a custom 'traffic router', to route protocol specific messages within the server to the right module address when found. If the message isn't routed then it is either dropped or an error message sent back using that protocols standards, otherwise the message is forwarded to the appropriate module.

The final version of AFS should aim to encapsulate IRC for chat purposes, TASServer for backwards compatibility, and the new protocol for future development. TASServer support could be initially implemented by taking TASServer and putting it into a module, removing the connection code, and replacing it with data fed to the module by the server , modifying it in order to support the other modules for chat etc.

4 : Initial Development and Testing

Initially, with TASServer being integrated into a module, the server will have all of TAS Servers features, allowing tasclient and AFLobby to use AFS unmodified. The new modules can then start to replace the TASServer functionality while implementing the new protocol, until the entire functionality of TASServer has been replicated and it can be removed.

5 : Lobby Support

Its not expected that tasclient would be modified to support the new protocol, backwards compatibility should be maintained however to ease the transition of auto hosts and bots to the new protocol. AFLobby should be expected on the new protocol, and Spring Lobby has the necessary multi protocol infrastructure to allow support.

6 : Game Engines

Command Engine needs such a server because of its high flexibility demands for which TASServer does not suffice.

Spring would need such a server in order to implement several long standing requests such as integrated ranking systems and galactic warfare. The proposed design would allow a 3rd party to test and manage such an implementation within a custom module, which after ratification and testing could be introduced to the main server as a part of the plugin design.

GLEvo (Glest evolution) has need of a lobby once they have multi player support completed.

The Populous 3 community also runs on a very old and primitive VB6 matchmaker with an abysmal User interface and could benefit greatly from a spring style server lobby system.

OTA and Supreme commander could be added through the lobby itself, however GPG Net and phoenix worx fill those niches already.

One of the TA3D developers has a lobby system that has yet to be put to good use ready for when TA3D multi

player support is perfected.

7 : Moderation and Access Controls

It has been requested by SwiftSpear that Linux style user access controls be possible. The ability to access controls to a protocol command based on a user or user type. Thus all modules will have to register the commands they support. Users will be put into 3 categories of users; players, moderators and administrators ~~and the third?~~, and all addresses will have a basic table specifying lists of users and user types that cannot send or receive a certain command from their address. Tables will also have such data stored in them. See 2.2 "Data Storage and Tables" for more details about this. ~~like a database.~~

8 : The core protocol commands

There will be 4 basic traffic commands:

MSG – A command or message containing data

GET table.key – retrieve a value

UPD table.key=newvalue – Add/Edit a key/value pair

RMV table.key – remove a value/key

9 : To-do

~~Write this document so that all people involved can understand it~~

- Get feedback from people involved
- Define message formatting
- Define login commands
- Devise a clean method of protocol differentiation between IRC, TASServer, and the new protocol
- Discuss the implementation of the basic set of server modules and their protocols

10 : Document version history

V.	Date	Author	Note
Draft 1	2007-07-31	Tom Nowell	Initial document
Draft 2	2007-07-31	Tom Nowell	Incoperated feedback from "Tim Blokdijk", added more information about the current situation, better document layout.