NoboDie
Nouveau membre
Messages : 16
Inscription : Feb 2014
Réputation : 10 Badges : Aucun haut-fait

Comment fonctionne un serveur Wow

Prélude : Ce sujet n'était pas original, il a été transféré d'un précédent forum d'émulation en date d'Octobre 2013. J'en suis l'auteur, je le transfère donc pour les besoins de ce forum. Bien qu'il soit en partie ré-écrit, il se peut que certains termes soient inadaptés ou ne soient plus d'actualité.

Ce tutoriel est généraliste et non spécialisé dans le C++. Merci aux modérateurs de le déplacer dans un forum adapté une fois celui-ci créé, puis de supprimer ce paragraphe.

Bonjour à tous,

le sujet que je vous propose aujourd'hui n'est pas directement un tutoriel, mais plutôt une explication sur le fonctionnement réel d'un serveur Wow. Le sujet traitera de différents points dont voici le sommaire :

1. Explications succintes
2. Les grandes parties d'un serveur de jeu Wow
3. Que doit-on savoir pour développer sur Wow
4. Définitions principales
5. Quelques chiffres
6. Connexion au serveur de jeu

1) Explications succintes

Avant d'entrer dans le détail, il est bon de donner quelques bases sur ce qu'est réellement un serveur de jeu (en général) et son fonctionnement basique. On part de zéro, mais il est toujours utile de revoir les bases, n'est-ce pas ?

Premièrement, un serveur n'est ni plus ni moins qu'un logiciel. Il s'agit d'une application qui s'exécute en continu sur une machine afin de proposer un service aux joueurs (ici, la possibilité de jouer). Un serveur accepte des connexions de la part de clients (le logiciel (wow) que vous utilisez tous pour jouer) qui vont se connecter à lui et échanger des informations. Ces clients n'ont aucun lien entre eux. Ils passent tous par le serveur pour communiquer. Par la même occasion, le serveur peut faire la police et détecter les mauvais clients (ceux qui ne sont pas censés se connecter, ceux qui trichent, ...). La connexion entre le serveur et le client se fait par le réseau. Il peut s'agir d'Internet si vous vous connectez à un serveur distant (hébergé sur une machine par exemple), il peut s'agir de votre réseau local si vous installez votre serveur chez vous, voir sur l'ordinateur local si vous installez votre serveur directement sur le même ordinateur que votre jeu. Pour communiquer, le serveur tout comme le client partagent le même protocole de communication (c'est l'équivalent de la langue pour un humain). Ils ont évidemment besoin de parler le même langage. Cela se fait à l'aide de ce qu'on appelle les paquets. Ceux-ci peuvent être de deux types :

- Ils vont du serveur vers le client (on appelle cela les paquets CMSG)
- Ils vont du client vers le serveur (on appelle cela les paquets SMSG)

Tout ce qui se passe en jeu (un joueur qui se déplace, une créature qui attaque, une quête qui se termine, une connexion qui commence) est envoyé ou reçu par le serveur via des paquets. Certains sont très simples (uniquement un paquet et un identifiant de cinématique, pour lancer la cinématique correspondante), d'autres abominablement complexe (le paquet envoyé lors du cast d'un spell contient jusqu'à 200 variables).

Le serveur possède ce qu'on appelle du contenu. C'est généralement la seule chose dont les joueurs ont idée car c'est la seule chose qui leur est visible (c'est le haut de l'iceberg). Le contenu peut être de plusieurs types :

- Des spells
- Des world boss
- Des instances/raids
- Des quêtes
- Des zones spawnées (càd des zones sur lesquelles les PNJs sont présents)

Nous verrons plus tard comment ce contenu est géré sur un serveur de jeu ayant pour base TrinityCore.

2) Les grandes parties d'un serveur de jeu Wow

Maintenant que nous savons à peu près comment fonctionne un serveur de jeu générique, il est temps de voir comment fonctionne le serveur de jeu wow.

J'ai délibérement choisi de parler de l'émulateur TrinityCore qui est selon moi la meilleure base de travail. Je ne parlerais pas ici des différences entre chaque émulateur de Wow, le principe est identique pour chacun d'eux et ils sont pressque tous basés sur le même système TrinityCore de toute façon. Ce que vous apprendrez sur TrinityCore, vous pourrez le reproduire sur n'importe quel système.

Voici les différentes briques utilisées par un serveur de jeu de type TrinityCore :

- Une application AuthServer
- Une application WorldServer
- Des fichiers de configuration pour les deux applications citées ci-dessus
- Des bases de données de type SQl (Auth, Char et World)
- Une base de données binaires (DBC) extraite du client
- Un certain nombre de fichiers regroupés sous le nom de MAPS extraits du client
- Un certain nombre de fichiers regroupés sous le nom de VMAPS extraits du client
- Un certain nombre de fichiers regroupés sous le nom de MMAPS extraits du client

Cela fait une grosse quantité de données. Regardons cela de plus près :

- Une application AuthServer
- Une application WorldServer

Ces deux applications sont les applications serveur qui tournent en continu. Pourquoi deux applications alors qu'on ne souhaite faire tourner qu'un serveur Wow ? Tout simplement car TrinityCore a été développé de façon générique et permet donc de gérer plusieurs serveurs. En effet, comme sur officiel, nous nous connectons tout d'abord à l'application AuthServer qui se charge de vous authentifier (en vérifiant les données entrées avec celles présentes dans la DB Auth que nous verrons tout à l'heure). Une fois l'authentification passée, vous choisissez un royaume. C'est ce choix qui déterminera vers quelle application WorldServer vous allez être redirigés. Dans le cas où vous mettez en place un seul serveur, il n'y aura évidemment qu'un seul WorldServer, votre choix sera vite fait.

- Des fichiers de configuration pour les deux applications citées ci-dessus

Les fichiers de configuration sont primordiaux. Ce sont eux qui vont faire le lien entre toutes les données pures (les bases de données, les DBC/Maps/Vmaps/Mmaps, ...) et vos application. Ce sont ces fichiers qui permettent également de configurer bon nombre d'options pour votre serveur.

- Des bases de données de type SQl (Auth, Char et World)

Il en existe 3 comme vous pouvez le constater :

La base de données Auth contient toutes les données liées aux comptes joueurs (identifiants, emails, droits d'accès, ...) ainsi que quelques autres options que nous n'aborderons pas ici.

La base de données Char contient toutes les données dynamiques liées aux personnages (nom, spells, items, monnaies, courriers, ...) et au serveur en lui-même (date du prochain reset de Raid, temps de respawn restant pour les boss, état des instances en cours, ...). On dit qu'elle est dynamique car elle change très souvent au cours du jeu (en fait, dès qu'un joueur est sauvegardé, cette base de données est mise à jour).

La base de données World contient toutes les données statiques liées au serveur (positionnement des PNJs, quêtes, chances de loot de chaque objet, ...). Elle est dite statique car elle ne change pas au fil de l'exécution "normale" du jeu (sauf dans le cas d'un serveur de test sur lequel vous tapez des commandes pour ajouter des PNJs, ...). Cette base donnée est théoriquement chargé en mémoire une seule fois au lancement du serveur, puis n'est plus utilisée (en théorie seulement, en pratique quelques requêtes cherchent encore des données dedans).

- Une base de données binaires (DBC) extraite du client

La base de données binaires est similaire aux DB SQL, sauf qu'elle enregistrée au format binaire. L'avantage principale est qu'elle est chargée directement en mémoire au chargement du programme. L'inconvénient majeur est qu'elle est foncièrement illisible pour le commun des mortels. Il existe différents logiciels pour lire ces informations; Il est également possible de développer vos propres outils de "traduction" de la base vers un format plus lisible comme le format SQL.

- Un certain nombre de fichiers regroupés sous le nom de MAPS extraits du client
- Un certain nombre de fichiers regroupés sous le nom de VMAPS extraits du client
- Un certain nombre de fichiers regroupés sous le nom de MMAPS extraits du client

Ces trois types de fichiers fonctionnent de concert. Il s'agit de données brutes extraites et modifiées à partir du client.

Les maps sont les seuls fichiers indispensables au serveur pour fonctionner. Ce sont ces fichiers qui contiennent la hauteur de chaque point du décor et qui permettent donc de ne pas traverser le sol (en théorie bien sûr, des bugs subsistent toujours). Ils ne servent qu'à effectuer les calculs pour les personnages de type joueur, les PNJs ne sont pas impactés par leur utilisation.

Les vmaps sont facultatives. Elles servent à calculer ce qu'on appelle le Line Of Sight (le Champ De Vision) des créatures. C'est ce système qui empêche les créatures de vous voir à travers les murs. Attention vous parlons bien uniquement de Vision. Une fois que vous êtes ciblé, les créatures vous attaqueront en ligne droite et ne prendront plus les obstacles en compte.

Les mmaps sont facultatives également. Ce sont elles qui empêchent les créatures de traverser les murs ou de voler lorsqu'elles sont censées rester au sol. Elles permettent également aux créatures de contourner les objets pour vous rejoindre (à l'aide d'un système de PathFinding, nous en reparlerons dans un prochain tutoriel).

Voilà l'ensemble des fichiers de base que nécessite un serveur Wow. Vous constaterez que nous n'avons toujours pas parlé des connaissances qu'il faut avoir pour développer sur Wow, ni dans quel langage sont développés les deux applications serveur. C'est ce que nous allons dès à présent.

3) Que doit-on savoir pour développer sur Wow

Les compétences requises pour développer un serveur Wow sont multiples. On peut tenter de tout connaître, mais face à la complexité du système et surtout à son ampleur, il est plus intéressant de commencer par se concentrer sur un point précis du développement. Je vais lister ici les différents "points" qu'il existe, mais il est fort probable que votre travail vous oblige à jongler entre ces différentes catégories. Voici la liste, par ordre de difficulté d'apprentissage (ce n'est pas un dénygrement des premiers niveaux, chacun de ces postes est nécessaire pour le bon fonctionnement d'un serveur. De plus, ces postes sont définis arbitrairement par mes soins après mes expériences personnelles dans l'administration de mes serveurs) :

- L'administrateur du serveur
- Développeur SQL
- Debugueur SQL
- Développeur de contenu
- Debugueur de contenu
- Développeur de fonctionnalités
- Développeur Core

Je commencerais par l'administrateur du serveur. Certains le placent tout en haut de la liste, moi je le place tout en bas. Non pas que son travail soit simple, mais il requiert souvent des connaissances en gestion d'une machine plus que d'une connaissance en développement. En échors de quelques scripts d'exploitation pour relancer le serveur en cas de crash, son but consiste plutôt à faire en sorte que tout tourne correctement. C'est un poste à part qui n'est pas toujours de tout repos.

Un debugueur SQL est généralement du même niveau qu'un développeur SQL, mais sa capacité à comprendre un bug, en identifier l'origine et le corriger le place un plus plus haut de par sa connaissance du système.

Le développeur de contenu est l'équivalent Cpp du développeur SQL. Il est chargé de développer le contenu tel que des quêtes complexes, des bots, des raids, etc. Bien qu'il soit également seul maître de ses scripts, je le place au dessus du debugueur SQL car le Cpp est plus compliqué à appréhender pour un débutant que le SQL.

Le debugueur de contenu est le pendant du debugueur SQL. Sa capacité à naviger dans le code source du serveur lui permet de corriger des spells, des boss contenant des erreurs, etc.

Le développeur de fonctionnalités possède de la même aisance que le debugueur lorsqu'il s'agit de farfouiller dans le code source du serveur, mais son aptitude à imaginer de nouveaux systèmes (nouveau BG, système de transmogrification ou de chambre du vide sur cata, système de marché noir sur Mop, ...) lui permet d'être classé au dessus du développeur de contenu.

Le développeur Core, enfin, est celui qui travaille le plus souvent non pas avec le serveur, mais avec le client. Il est en effet chargé d'identifier le protocole de communication entre le client et le serveur et se base donc sur les connaissances que peuvent lui apporter le client. Il identifie un système, établit un "Proof of Concept" permettant de faire fonctionner un système, et développe les fonctions principales permettant de faire fonctionner le système. Dans le cas d'un nouveau système comme la chambre du vide ou le marché noir, il reprogramme les informations identifiées et les transmet au développeur de fonctionnalités qui se charge alors de coder tout le système. Evidemment, dans de nombreux cas, le développeur Core développe lui-même la nouvelle fonctionnalité. Encore une fois, le travail nécessite souvent de travailler dans plusieurs domaines.

C'est à vous de voir dans quelle catégorie vous placer en fonction de vos objectifs, vos connaissances, votre motivation. Mieux vaut commencer en bas de l'échelle en tant que développeur SQL, puis monter les échelons au sein d'une équipe de développement. Vous pourra vous fournir le support technique lorsque vous aurez un problème ou que vous ne comprendrez pas le fonctionnement de telle ou telle partie du serveur.

N'oubliez pas que toutes les questions intelligentes sont bonnes à prendre.

4) (Annexe)Définitions principales

- **Machine serveur** : La machine serveur est le serveur physique (= l'ordinateur) hébergeant l'ensemble des services. Cela inclue le site internet, le mumble, le serveur de connexion, le serveur de jeu, la base de données, ...
- **Client** : Le client correspond au jeu installé sur votre ordinateur. Il s'agit d'une application dans une certaine versionaccompagné de l'exécutable permettant de se connecter à un serveur via le realmlist.
- **Serveur de connexion** : Le serveur de connexion est un service qui se charge d'authentifier tout joueur qui se connecte. Cela est invisible pour le joueur lambda, mais il passe en réalité à chaque connexion par ce serveur. Si le serveur de connexion n'authentifie pas le joueur, alors celui-ci se voit refuser l'accès au serveur et ne peut donc pas jouer.
- **Serveur de jeu** : Le serveur de jeu est le service principal qui fournit l'ensemble du jeu aux joueurs. Une fois connecté ingame, le client ne fait plus qu'échanger des données avec celui-ci. Si la connexion est rompue, alors le joueur est déconnecté du serveur. C'est ce qui arrive quand on parle de "crash" par exemple. En réalité, la moindre erreur, le moindre oubli dans le code du serveur peut générer un crash et donc la déconnexion du joueur, même si celui-ci n'est pas directement responsable du problème.
- **Base de données** : La base de données et un service invisible aux joueurs, mais pourtant primordial. Il s'agit de la zone de stockage de toutes les informations importantes du serveur. Il existe, pour simplifier, deux bases de données. La première, appelée le "World", contient l'ensemble du contenu du serveur. C'est là dedans que se trouvent toutes les informations de quêtes, de PNJs, de loot, d'objets en vente, etc. Elle contient également les relations qui lient un PNJ à son script (ce script se trouve directement dans le serveur de jeu). La seconde base de données contient l'ensemble des données joueurs. Tout ce qui touche à un personnage (métiers, sorts, items, réputations, ...) se trouve dans cette base de données.
- **Paquet** : Comme expliqué précédemment, le client et le serveur communiquent entre eux plusieurs dizaines, voire centaines de fois par seconde. Cela fonctionne via un système de paquets. On peut assimiler cela à une discussion par mail. On envoie par un mail. Tout comme un mail est composé d'un titre et d'un contenu, un Packet est composé d'un Opcode et d'une suite de données remplie. Imaginons par exemple que vous souhaitez candidater pour un poste dans une entreprise (toutes mes excuses pour l'exemple, la faute aux recherches de stage icon_e_wink.gif). Vous mettez comme titre "Candidature" afin que votre interlocuteur sache que vous souhaitez candidater. Ensuite dans le contenu, vous n'y allez pas n'importe comment. Vous commencez par une partie de présentation, puis vous donnez vos qualités, et enfin vous terminez par une formule de politesse. Bien entendu, vous y joignez votre CV et une lettre de motivation. Et si tout est bien envoyé, alors vous recevrez une réponse, et eut être même qu'elle sera positive icon_e_smile.gif Pour un paquet, c'est exactement la même chose. Pour que le client puisse comprendre ce que vous lui envoyez (et inversement) il faut parler exactement la même langue. Le paquet est être approximatif, avec une faute ou deux dans chaque phrase. Le client attend une structure particulière et vous ne pouvez PAS lui envoyer n'importe quoi. Ainsi, il faut d'abord lui envoyer l'Opcode pour qu'il sache de quoi vous allez parler. Ensuite, il faut lui envoyer exactement la bonne structure avec les bonnes données dedans. Dans le cas contraire, le comportement du système est impossible à prédire. Le client peut tout simplement ne rien faire (par exemple, vous ne voyez bouger aucun pnj), voir crasher (les bien connues Wow Errors).
- **Entité** : Une entité est un objet ingame. Cela peut être n'importe quoi. Un joueur est une entité, un item est une entité, une créature est une entité, un véhicule également. Une chaise, un coffre, un piège, sont tous des entités également.

5) (Annexe)Quelques chiffres

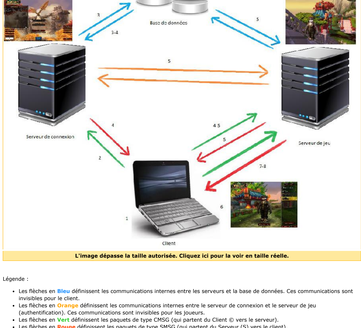
Voici quelques chiffres pouvant faire peur/rassurer/impressionner selon les cas. Ces chiffres sont basés sur ma seule expérience personnelle des serveurs privés que j'ai (eu) l'occasion de gérer.

- Taille du serveur de jeu : Environ **700 000** lignes de code C++
- Taille de la base de données World (contenu du jeu) : environ **2 300 000** enregistrements SQL
- Taille de la base de données Joueurs après un an d'ouverture : près de **150 000 000** enregistrement SQL
- Nombre de paquets différents : Plus de 2500
- Nombre de paquets nécessaires pour avoir un serveur accessible (de type "M" aussi appelé Sandbox "Bac à sable") : environ 30%
- Nombre de paquets nécessaires pour avoir un serveur fonctionnel (de type "Bizzarke") : environ 60%
- Si on transformait tout le contenu du serveur (sources du serveur + base de données World) en un fichier texte, celui-ci ferait **30 000** fois la taille de ce tutoriel.

6) Connexion au serveur de jeu

Maintenant que nous sommes au point sur la définition des principaux termes utilisés et que vous pouvez vous faire une idée de l'ampleur du projet, nous allons pouvoir expliquer le fonctionnement du serveur de jeu.

Voici un schéma mettant en scène l'ensemble des services permettant au joueur d'accéder au jeu.



L'image dépasse la taille autorisée. Cliquez ici pour la voir en taille réelle.

Légende :

- Les flèches en **Bleu** définissent les communications internes entre les serveurs et la base de données. Ces communications sont invisibles pour le client.
- Les flèches en **Orange** définissent les communications internes entre le serveur de connexion et le serveur de jeu (authentification). Ces communications sont invisibles pour les joueurs.
- Les flèches en **Vert** définissent les paquets de type CMSG (qui partent du Client ⇒ vers le serveur).
- Les flèches en **Rouge** définissent les paquets de type SMSG (qui partent du Serveur (S) vers le client).

1. Le joueur entre ses identifiants sur le client.
2. Le message transite vers le serveur de connexion
3. Le serveur de connexion vérifie les informations et valide/invalide la connexion. Il renvoie ces informations au client.
4. Le client affiche la liste des royaumes grace aux informations envoyées par le serveur de connexion. Le joueur choisit son royaume. Le client demande au serveur choisi de lui lister ses personnages.
5. Le serveur de jeu vérifie que le client s'est bien authentifié sur le serveur de connexion. S'il est valide, il renvoie la liste des personnages au client.
6. Le joueur voit sa liste des personnages. Il en sélectionne un et se connecte au jeu.

A. Plusieurs dizaines de paquets sont échangés entre le serveur et le client. Ils contiennent toutes les informations du personnage, puis les informations sur tout l'environnement autour (pnj autour, etc).

B. Le joueur est enfin en jeu et peut se déplacer. A partir de là, plusieurs dizaines de paquets sont échangés chaque seconde, à chaque action du joueur ou d'une entité visible autour de lui (déplacement d'un pnj, lancement d'un sort, invitation par un autre joueur, ...).

Ce petit descriptif a été entièrement rédigé par moi-même. Celui-ci sera mis à jour et amélioré le plus souvent possible. Je reste évidemment disposé à entendre toutes les remarques constructives qui pourront être faites. Merci de votre attention, en espérant que ce sujet soit le premier d'une longue série.

Encore et toujours du travail
Ancien Lead Dev/Admin de **Aurore-Serveur** - Premier serveur Cataclysm
Lead Dev/Admin de **Mist-Eria** - Serveur Cataclysm 4.0.6/**MoP**