CH Server Architecture

CH 服务器体系

Owner

Development: Peng Tao(XiaoP)

Revision History

Version	Date	Author	Comments
1.0	9/11/04	XiaoP	Initial release

Table 1: Revision History

Table of Contents

1．概述（Overview）

本文描述了 CH 服务器端的设计问题以及服务端的架构。

2．假设与需求（Assumptions & Requirements）

2．1　世界实例（World Instance - WI）

将世界历史在一个游戏实例（WI）中完全演绎，特定的历史事件和历史人物只在一定的历史时期中出现。WI演绎的速度为 600 虚拟年/年（virtual years per year）TBD[1]，则演绎完 3000 年历史需要 5 年左右的系统时间。同时，在系统中存在多个WI独立运行，它们之间相隔一定的演化历史时间。假如WI之间以 200 虚拟年（以下称CHYear）为间隔，则系统中将有 15 个左右的WI同时运行，他们构成了世界实例组（WIGroup）。这样，系统在每 4 个月将新增一个WI从历史起点启动，并将它加入到WIGroup中，同时最前面的一个WI将合并入Super World Instance（SWI）。SWI和另外一个娱乐性实例Future World Instance（FWI）是一直存在于系统中的两个实例，在考虑系统压力的时候将整个系统的在线人数作为它们的实例最大同时在线人数。

2．2　地区（Zone）

整个世界按照地图分成了多个地区（Zone），地区边缘在地图上用特定的地形表示以避免接入者达到边界处。不同的 Zone 之间以特定的入口相连，接入者通过这些特定的入口可以在不同的地区之间来往。另外，系统还提供一些特别的方式使接入者更直接快速地进入不同的地区，这将在内容层次（Content Layer）上得到不同的表现。

单一的世界以及允许在地区之间自由游历使得接入者能够获得统一的交互感受，在世界实例跨越的可能性上，整个系统的接入者能够感受到一个统一的世界体系。将世界分成多个地区，还有利于平衡服务器的负载，有利于根据客户需要扩充服务器规模。而且，如果某个地区服务器系统发生问题，它不会影响到其他地区。但是，跨地区的时候，由于要载入新的地图、地形数据，客户端可能会产生等待时间。另外，地区多实例（见下小节）的实现要保证密切的服务器间通行，设计比较复杂。

Zone 中的地形只会在长时间上有少量变化（总体上看是一种腐蚀效应）。

2．3　地区实例（Zone Instance - ZI）TBD

如果有太多的接入者要进入到某一个地区，系统将为更多的接入者创建另外一个属于该地区的实例（ZI）。这样，多个 ZI 将分担这个地区接入者请求的处理。每个 ZI 一般在单独的一台服务器上运行，这样的服务器称为 ZIServer。

一些特别的 ZI 被用来做系统测试，它们可以具备随机地图。

如下图，当地区 B 的实例 B1 达到了它的人口上限时，系统将产生另外一个属于 B 的实例 B2，这样，我们可以很容易地扩充一个地区的接入规模。当然，在地区实例组中，B1、B2……所在服务器之间要进行密切的通信以及同步来保证这些 ZI 的无缝组合，使得接入者感觉是在一个地区空间中同时进行交互。

[1] TBD means to be decided

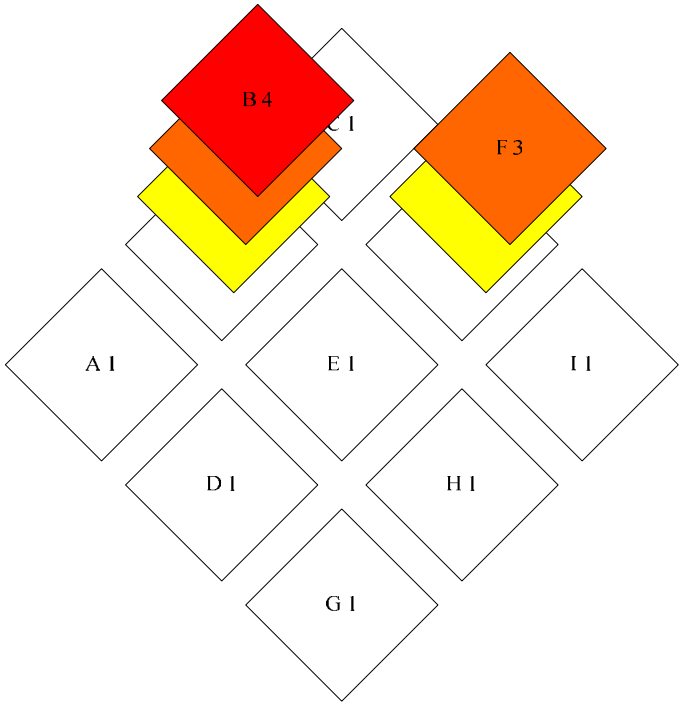

Fig.1 Multiple Instances of Map Zones

2．4　运行时系统操作（In-game Sysop）

服务器支持一些高权限的系统操作，这种系统操作可能由服务器端管理人员进行，也可能来自有权限的客户端。

2．5　客户端更新（Client update）

服务器通过专门的通信信道和协议自动更新客户端的资源。

2．6　安全性（Security）

客户端和服务器端的通信数据经过一定的加密，被封装成数据包以后才能在网络上传输，通信两端都具备封包和解包的接口。

2．7　资源需求（Resource Requirements）

我们假设系统的同时在线人数达到 50,000 人，世界被分成了 100 个地区，每个服务器可以支持 50 个接入者同时交互。另外，一个极端情况是所有接入者都进入同一个 WI。下表列出了一些服务资源的需求情况。

Total		
Concurrent users	50,000	
World Instance		
Concurrent users per instance	1,000-50,000	
Zones per instance	>100	初期比较少
Zones		
Users per zone with acceptable game experience	<50,000	全部集中在一个地区的极端情况
Machine estimates per Instance		
Game servers（Zone Instances）	<1000	初期比较少，非均匀分布

Instance Controller	1	
Zone Controller	>100	每个 Zone 需要一个
SQL Server	<50	初期比较少，每 20 个 GameServer 需要一个

3．架构（Architecture）

3．1　应用架构

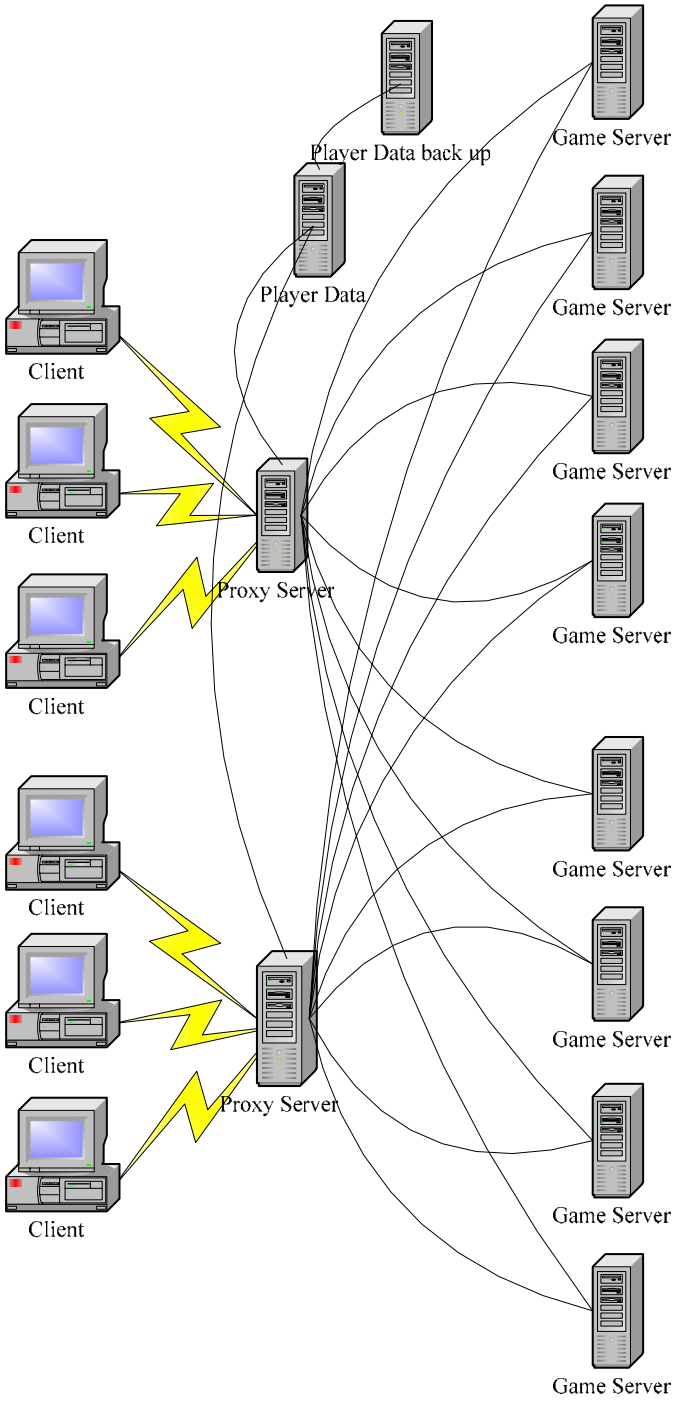

Fig. 2　一种简单的应用架构

3．1．1 代理服务器（Proxy Server）

代理服务器是分布在各处的系统接入点，它负责传递客户端与服务器之间的数据。每一个代理服务器都与所有的服务器相连，它们之间是高速通道，这样才能保证世界数据的中心化、客户端接入的高效统一。从架构上来讲，代理服务器其实仅仅在接入点充当路由器的角色，它具备以下功能：

- 路由客户端与服务器的连接，这样，客户端无论在何处都可以连接上任意的地区实例（ZI）服务器。因为 CH 游戏服务器（Game Server）一般运行着一个 ZI，它属于某个 Zone，路由系统能使客户端进入世界的每一个 Zone。同时，代理服务器和游戏服务器之间是高速通道，在高效路由的情况下（即不作其他处理），还能够确保各处的客户端都高效快速地和游戏服务器交互。
- 能够配合一个独立的聊天信息系统，帮助转发聊天信息或者直接指导客户端聊天连接。这样，尽可能地避免了聊天信息对Origin[2]的影响，减轻游戏服务器的负荷。
- 支持游戏相关通信，比如 BBS，Email 等外部系统，使它们不得影响 Origin。
- 身份验证工作。能够直接访问用户数据库，获取用户信息和静态数据以及一些统计数据。由于一些身份验证工作要在代理服务器上进行，所以能够访问用户数据库是必要的，这也能够降低 Origin 在身份验证上的工作量。
- 代理服务器能够暂时在本地保留一些用户的信息，这样能够降低 SQL 数据库的访问频繁度，提高效率。
- 能够将多个往来同一游戏服务器的连接捆绑集群，以提高网络效率。这要求游戏服务器端支持相应的分离功能。
- 支持多种类型的客户端。

可以形象地说，各个代理服务器就像 Origin 伸向各处的连接管道的入口。

3．2 服务端架构

服务器端采用一系列的服务而不是单一的程序，它的主体称为“**Origin**”。

下图表示了主要的服务以及连接的网络层次。

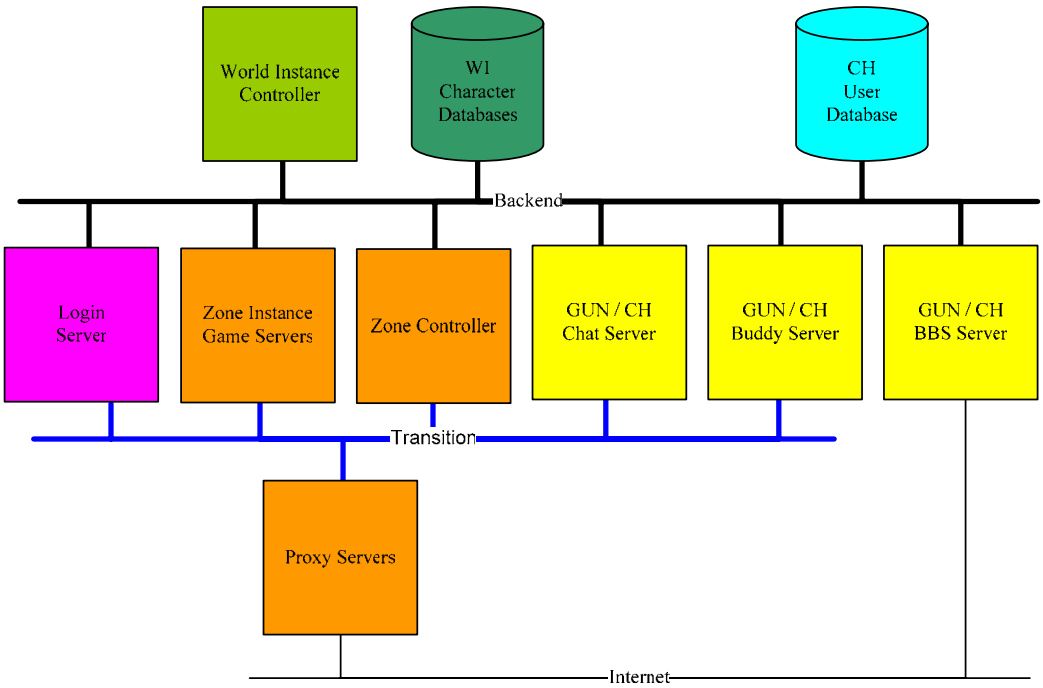

Fig. 3 服务器端实际服务层次

[2] 世界交互系统的服务端主体称为“Origin”

上图中，黄色部分为独立于 Origin 的服务体系。它们可以是来自第三方的系统，但是它们仍然可以访问中心数据库的部分信息。其中，聊天、伙伴服务器也通过代理服务层来提高性能。

下图则表示了 Origin 各个服务之间的逻辑关系。

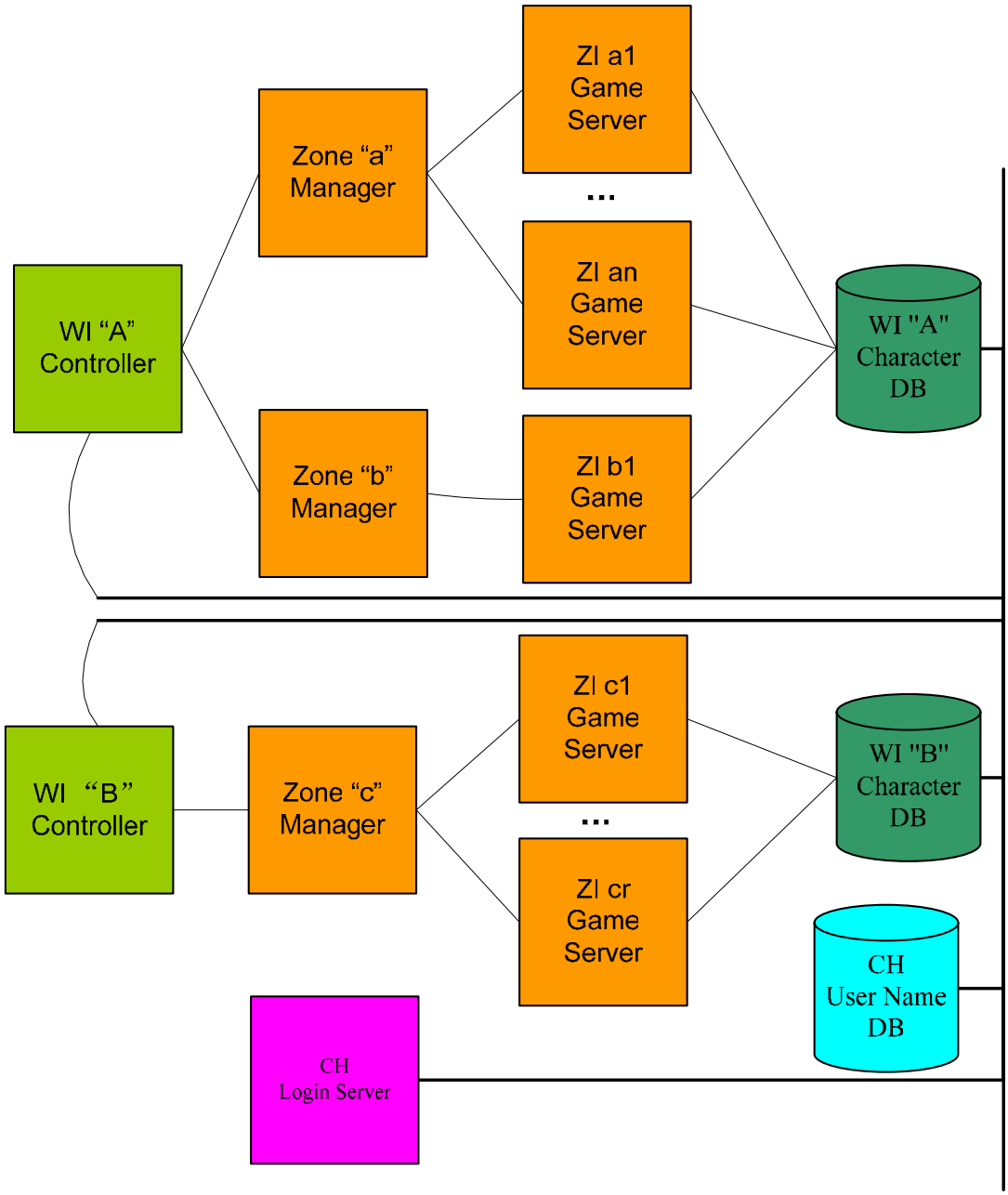

Fig. 4 服务器端服务的逻辑结构（Origin）

3．2．1 游戏服务器（Game Server）

游戏服务器是 Origin 系统中的主要服务器，负责运行各种 ZI，也称为“地区实例服务器（ZI Server）”。它与数据库服务器交互并配合其他 ZI 共同维护一个 Zone 的模拟。游戏服务器具备以下功能：

- 维护与 Zone Controller 的通信
- 支持各种 Zone Instance
- 验证客户端的各种行为、命令等请求
- 执行并响应客户端的各种请求，与世界的模拟模块（Simulation Module）交互。
- 向客户端报告各种来自 Sim Module 的相关事件

- 部分 AI 的模拟
- 验证来自 Proxy 的接入者信息

3．2．1．1 CPU 预算开销

Assumption is quad-proc 1.5 GHz servers:

Zone Game Server

2x Game CPUs (assumes 200ms game ticks)

5ms	Scheduling overhead
15ms	Network read
100ms	Simulation tick (physics, path planning, AI, etc.)
80ms	Network write

2x Utility CPUs

50%	Net
30%	SQL

3．2．1．2 内存预算开销

Zone Game Server

128MB (x2)	Per zone instance
128MB	Operating System
512MB (rounded up)	Total per public zone server

3．2．2 地区管理器（Zone Manager）

负责管理属于同一个 Zone 的 ZI 并且维护基本的用户状态。

- 维护各个 Zone Instance 之间的通信信道。
- 维护 Zone 中 ZI 和用户的列表。
- 处理并响应进入 Zone 的用户请求。
- 在需要的时候创建新的 ZI 加入 ZI Group。

3．2．3 世界实例控制器（World Instance Controller）

管理整个 WI 的各个 Zone，维护 WI 中的基本用户信息。

- 维护各个 Zone 之间的通信信道。
- 配合 ZoneMan 处理跨越地区的请求。
- 处理并响应进入 WI 的用户请求。

3．2．4 系统登陆服务器（CH Login Server）

Login Server 负责管理接入者（用户）列表，同时还负责用户角色的管理。

- 接入者的帐号信息将在连接 Proxy 时被验证，一旦通过验证，Proxy 通知 Login Server 该接入者的

进入，Login Server 则在用户列表中保留一个接入者的信息，Proxy 在验证权限时也要查询这个列表。

- 维护与各个 Proxy 和 WI Controller 的通信
- 维护 WI 的列表，接受查询
- 维护接入者的角色信息列表，接受查询
- 创建角色
- 删除角色

3．2．5 数据库服务器（Database Server）

代理服务器和游戏服务器都需要访问一些数据库服务器，这些数据库服务器有以下几种类型：

- 静态数据服务器，维护一些固定物体的数据、物质库、地区（Zone）等等
- 动态数据服务器，包括接入者数据、统计数据、大多数物体数据等等，可能还包括游戏相关的信息比如 BBS、Email 数据等。
- 动态和静态结合的数据服务器，比如原始地形数据和地形修改信息的组合。

3．2．5．1 客户帐号数据库服务器（Account DB、User DB）

- 帐号许可
- 帐号与角色的关系数据
- 角色基本数据
- 整个系统只有一个 ADB

3．2．5．2 世界实例数据库服务器(WI DB)

- 物质数据
- 物体数据（注意，由于角色只是物体之一，所以角色信息保存在各个 WI DB 上）
- 地区数据
- 尽量优化数据，做到每个 WI 一个 WI DB

3．2．5．3 事务估计 (Transaction Estimates)

下表表示了系统中 SQL 的估计事务流量，这些数据混合了各种 Zone 数据以及测试区的数据，有待进一步考察。

Assumptions		
Simultaneous users		50000
Percentage return after crash		90%
Persistent updates per user per hour		15
Max transactions per sec per SQL server		350
Safety margin		20%
Steady-state connects per sec		5
Crash connects per sec		375
SQL Requests per sec (steady-state)		
Login		5
Game starts		5
Character updates		208.33
	Total:	218
	SQL Servers:	1

SQL Requests per sec (crash)		
Login		375
Game starts		375
Character updates		208.33
	Total:	958
	SQL Servers:	4

3．3　简要流程（Walkthrough）

这里将叙述客户端与服务器端以及服务器之间的简单交互过程，其中不包括错误的情况，例如服务器满、数据库错误等等。

1. 客户端连接代理服务器，请求验证帐号
2. 代理服务器查询 Login Server 帐号是否在它处被使用，如果未被使用，建立使用标志
3. Proxy 进一步查询 Account DB 进行身份验证
4. Proxy 查询 Login Server 中用户的角色信息，返回角色列表给客户端
5. 客户端选择角色后，Proxy 根据用户角色信息，在与角色对应的 Zone（某个 WI 中）管理器之间建立连接，请求一个 ZI 的信道。ZoneMan 加入该角色到 Zone 角色列表并通知 WI Controller。
6. ZoneMan 返回一个最富裕的 ZI 给 Proxy，Proxy 接着在对应 ZI 与客户端之间建立一个信道
7. ZI server 通过该信道激活对应角色的模拟，开始与客户端进行交互
8. 经过数小时的交互，客户请求退出
9. ZoneMan 清除角色标志并通知 WI Controller
10. ZoenMan 关闭信道
11. Proxy 通知 LoginServer 清除使用标志
12. Proxy 断开与客户端的连接

4．网络（Network）

4．1　协议（Protocol）

CH Game Server 周期性的发送状态更新信息到客户端，客户端在这些更新信息之间插值其改变量。不同的延时、丢包、时钟漂移、错误插值以及其他问题都将引起客户端与服务器不同程度的失谐。应此，应该在接入者某些行为操作上注意避免这种偏差，例如，采用指定攻击目标的操作而不是瞄准目标攻击的操作有助于摆脱射击动作时目标在各个客户端位置的偏差。

Game Server 负责整个世界状态的模拟，它决定接入者行为的结果。唯一的例外是，客户端可以直接操作接入者角色的位置，这样使得游戏像是立即对输入进行响应一样。当然，服务器端对这样的信息是不认可的，客户端需要适当地检查它接收到的更新信息。

4．2　筛选（Filtering）

在同一时间，一个 Zone 中的许多物体都需要对外发出更新通知，需要某种方式的筛选。可采用以下策略：

- 只对某个角色更新离它半径 X 以内的物体信息
- 对这些物体按照优先权排序，比如按照离角色的距离排序
- 根据客户端的带宽将更新信息（根据优选权）分为重要的、次要的、可以忽略的部分

5．安全性（Hacking）

下面是一些减小破解攻击对 CH 系统影响的方针和建议。

5．1　规则（Policies）

- 禁止发布没非安全版本的软件
- 在服务器上记录可疑行为，自动中断连接
- 对一些明显弱智的重复行为，虚拟世界不给予行为结果应有的奖励

5．2　客户端攻击（Client hacking）

- 在内存中混合一些数据，以增加攻击者值搜索的难度。比如简单地加减一个随机常数就会明显地降低值搜索的成功率。
- 每次启动使用不同的内存分配。所有重要的数据都应该在堆内存中进行分配，在程序启动的时候分配随机大小的内存块。
- 为了避免反相算法分析，在客户端和服务器端使用不同的随机数发生器。

5．3　网络层攻击（Network）

- 不对客户端发送不必要的信息。比如，如果客户端只需要显示生命条而不需要现实具体的数值，服务器应该仅对客户端发送生命当前的百分比，而不是当前值。
- 系统产生的信息数据在加密时应该采用叫安全的算法以及变化的密钥
- 用户产生的信息数据则**不应该被加密**，比如聊天。如果加密，攻击者可能使用已知明文查找数据，从而破解加密算法。

6．外部相关技术（External Dependencies）

以下技术可能是 CH 系统（不仅仅是服务器端）要用到的：

- **Game OS**——游戏框架程序
- **GOS Script**——script language to implement front-end visuals and functionality
- **High speed networking layer**——Core networking layer that provides communication on a massively multiplayer scale
- **Database layer**——A database layer that encapsulates use of SQL from the game
- **Authentication, Authorization, Billing** – Required technologies for Origin to make sure a user is validated to play in our game and that that user is billed appropriately. These technologies allow us to provide an in-game presentation of our connection of the Zone.
- **Automatic Updating** – This gives us the ability to download new game improvements and fixes to our users after we ship the server
- **Admin/Ops server management tool** – This is a tool that will provide the ability to maintain and manage the server side of things for both administration and operations.
- **Data logging** – This is a system for outputting event data to a log file for further analysis.

6．1　Gun Technologies

CH is relying on GUN to provide several key technologies since we don't have the manpower to write all of these components on our own.

Network Layer

- Reliable and unreliable message delivery services.
- Plug-in authentication architecture.
- Packet encryption to help thwart hackers.
- Packet compression for large messages.
- Game clock syncronization.

Patching

- Support for patching files, distributing new files, deleting old files.
- Should be easy to produce distributions. Ideally there is tool that compares two directories to automatically create a distribution.

Chat

- In game chat system.
- Ability to chat with people across shards.
- Support for groups, e.g. allegiances, party, friends, map, and user-defined.
- Voice chat.
- Supports multiple languages.
- Bad word filtering. Needs to be locali

Buddy List

- In-game system similar to Zone Friends where users can add people as friends without first obtaining permission.
- Account based but UI displays game character names. Users who wish to keep their characters separate can disassociate a character from their account with respect to the buddy system.
- Ability to send application defined messages, e.g. an invitation to a map.
- Persistent messages.

BBS

- Available in game and the web.
- Available when game servers are down.
- Read-only section for official announcements.
- Events calendar for official and user create events.
- Specialty forums where postings are done through application defined forms. For example, a classified section for item trades.
- Semi-private areas for allegiances where only members can post.

- Open forums for general discussions by users.

Server Administration

- Start, stop, and monitoring of services.
- Configuration management.

Community Management

- In game customer support system.
- User management; e.g. message, boot, and gag.

In-Game Web Browser

- Ability to host web browser inside a full-screen D3D application.

6.2 Zone Services

Billing

- Core billing system.
- ZoneAuth for in-game authentication.

Web page creation

- Supporting web pages for the game.
  - Billing
  - Getting Started
  - News & Events
  - Hints & Tips
  - Downloads
  - Other TBD sections?
- Custom billing pages for in-game display.
- Other custom pages for in-game display.

6.3 Areas that need to be addressed

- **Databases**
- **Security**
- **Scalability**
- **Fault Tolerance & Reliability**
- **Testing & Debugging**
- **In-game sysop/community tool**