# 关于防范游戏加速问题

# 的解决方案

编 写 人 ： 马 喆

编写时间 ： 2010/7/6

版 本 ： 1.0

修订历史记录

|  |  |  |  |
| --- | --- | --- | --- |
| 日 期 | 版 本 | 说 明 | 作 者 |
| 2010-07-06 | 1.0 | 创建文档。 | 马喆 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

## 问题描述：

现存游戏中存在用户使用某些种方法或使用某些工具达到提高游戏中移动速度的问题，这样就造成了游戏系统极大的不平衡性。为了维护游戏系统的公平性，特提出如下方案来解决此类问题。

注：此方案只针对现有游戏系统中的移动数据来进行防范游戏加速的问题。

## 主要问题分析：

在通过了解与实验初步认定导致游戏产生加速的情形主要由以下两个方面造成：

1、使用类似加速齿轮等修改客户端时间工具，修改客户端时间来欺骗客户端，以达到加速发送数据包的目的。

2、用户通过某种方法获取游戏客户端内部API,或者利用某些手段来提高发送移动数据包的频率来达到游戏加速的目的。

## 主要问题解决方案

### 类加速齿轮工具防范方案

#### 类加速齿轮工具工作原理分析：

类加速齿轮工具的主要工作原理是通过修改客户端程序时间类函数来达到欺骗客户端程序目的，使客户端程序认为以达到发送数据所需的时间，从而达到加速的目的。

#### 防类加速齿轮工具策略的防范基本原理：



如图Figue 1 所示，在正常情况下，服务器在时间点B收到客户端在时间点A发送的数据包。那么服务器将时间点B1收到从客户端发送的另一个数据包（时间点A1）。服务器收包的时间间隔（B1-B）与客户端发包的时间间隔（A1-A）基本上应该是相等的或者存在于一个可容忍的误差值之中（当然在实际情况中由于网络的延迟或其他种种原因会造成两个时间间隔不相等的情况，对于这些情况我们将在后面进行讨论并解决）。

若用户在使用类加速齿轮工具的情况下（如图Figure 2所示），由于类加速齿轮工具改变了客户端时间，这就使得服务器本应在时间点B1点收到的数据包实际上在在时间点B1’上就收到了。这样就造成了服务器的实际收包的时间间隔（B1’-B）小于客户端的发包时间间隔（A1-A）的情况，并且这种时间间隔会随着时间的推移将逐步增大。

本解决方案正是利用的客户端在使用类加速齿轮工具后产生的这种特性来验证用户是否在使用类加速齿轮工具。其具体判断标准为：

1、服务器收包间隔（B1-B）持续小于客户端发包时间间隔（A1-A）

2、服务器收包间隔（B1-B）与客户端发包时间间隔（A1-A）的差值n2持续增大。

#### 关于此策略一些细节问题的处理方法

#### 服务器与客户端时钟频率不统一的问题

#### 问题描述：

由于服务器与客户端存在着物理硬件差异，因此可能产生在获取时间的时候存在差异。以在测试中观察到的情况为例，测试客户端的时间会略快于服务器时间，因此造成客户端发包时间差（A1-A）略大于服务器的时间差（B1-B）。这样若随着时间累积则将会达到类似使用类加速齿轮工具的效果，从而使得防类加速齿轮策略造成误杀的情况。

#### 解决方法

由于在此中情况下客户端与服务器时钟频率虽存在差异，但其误差相对较小。因此在服务器中计算客户端发包时间与服务器收包时间差的时候，加入定期同步概念。即服务器在收到若干个数据包后（测试中暂定为2000个数据包，此值为可配置值）对所有计数进行清零，重新开始计数。这就使得在客户端与服务器时钟频率误差达到可影响判断效果前对其进行修正。保证防类加速齿轮策略的准确性。

#### 由于客户端角色若长时间不移动造成服务器与客户端时钟频率误差累计。

#### 问题描述

此问题的产生愿因与问题3.1.2.1类似，也是由于客户端与服务器之间物理硬件的差异造成的时间差导致的。若角色长时间不移动（测试中发现其时间间隔在30分钟以上），在下一次客户端移动数据包到达服务器时会累计在两个数据包之间的时间误差，从而导致防加速齿轮策略对客户端数据包的误判。

#### 解决方法

此问题由于此类情况只在角色长时间不移动的情况下产生，因此添加对最后一次收到客户端移动数据包时间的记录，若服务器收到两次客户端移动消息的时间差大于某个特定的值（测试中设置为5分钟，此值为可设置值）时，则对此策略中的计数进行重置，以达到修正客户端与服务器时钟频率误差。

#### 提高在存在网络延迟的情况下策略精度的一些做法

为了减少防加速齿轮策略的可靠性，避免不必要的误杀情况。采用了对加速条件进行了严格限制的策略，只有在服务器收包间隔（B1-B）与客户端发包间隔（A1-A）的差值需要持续大于1000毫秒（可配置值），且当其差值在持续增大的情况下才会被判定为是使用类加速齿轮工具。但是正是由于这种严格限制策略也导致用户在某些特定的操作下会不是那么精确的被查出，特别是在存在网络延迟，以及使用低倍数的加速齿轮，且用户操作上走走停停等多种情况相叠加的时候会显得不那么灵敏。因此添加了最大可容忍误差值的概念（在测试中配置为4000毫秒，此值为可配置值），若客户端与服务器之间的误差达到此值就一定会被认为是使用了类加速齿轮工具。但考虑到网络环境的复杂性保留了需要持续性的条件，以避免因网络或其他考虑不周全的地方造成误判。

#### 关于容错的处理

为防止其他一些异常问题导致的误判情况，因此添加了对容错处理。

1、在服务器收包间隔（B1-B）与客户端发包间隔（A1-A）的差值设置1000毫秒（可配置值）的容错空间。

2、在服务器连续收到10个（可配置值）符合限制条件客户端数据包时，才认定客户端为使用了类加速齿轮工具。

#### 综述防范条件

1、服务器收包间隔（B1-B）与客户端发包间隔（A1-A）的差值大于1000毫秒（可配置值）

2、服务器收包间隔（B1-B）与客户端发包时间间隔（A1-A）的差值n2将持续增大。

3、服务器收包间隔（B1-B）与客户端发包间隔（A1-A）大于4000毫秒（可配置值）。

3、客户端两次发送移动数据包的时间间隔小于5分钟（可配置值）

4、在服务器收到的2000个（可配置值）客户端移动数据包中有连续10次（可配置值）符合限制条件1，2或符合限制条件3。

### 防加速发送移动数据包的防范方案

#### 加速发送移动数据包的工作原理分析：

此问题主要是由于用户通过某种方法获取客户端API或者通过其他方法在较短的时间内将移动数据包发送到服务器。即如图Figure 1所示，是通过缩短N的值来达到加速的目的。

#### 防加速发送移动数据包策略的基本原理：



如图Figure 3所示，由于角色在从点A移动到点C的过程中需要服务器发送两个数据包（从点A移动到点B，再从点B移动到点C），且其发包的时间间隔是与角色的速度有关的。即假设角色速度为350，那么理论上来讲客户端发送从点A移动到点B的数据包与从点B移动到点C的数据包时间间隔应为350毫秒。由于角色的速度对于服务器是一个已知的值，则理论上可以根据客户端发送两次移动数据包的时间间隔与服务器上已知的角色速度进行比对来判断客户端是否有加速发送移动数据包的嫌疑。

#### 关于此策略一些细节问题的处理方法

#### 对于角色在受到加速，减速魔法等情况下产生客户端与服务器角色速度不同情况的处理

#### 问题描述

当用户角色受到减速魔法或骑乘的时候，存在着客户端与服务器在一定时间内角色速度不同步的情况。例如，某个角色在受到某个减速魔法，其服务器上角色的速度值已经改变，但在一定时间内客户端向服务器发送移动数据包的频率还是按照原有的数据来发送，这就有可能导致防加速发送移动数据包策略误判的问题。

#### 解决方法

对于此类问题，在服务器中添加对前一次角色速度的记录，若新的角色速度在变大时（即客户端减速时）。依旧在一定时间内按照原有的速度对客户端的发包间隔进行检测。在此这个时间间隔为一个可设置的经验值。在测试中体验设置值为100数据包，即在服务器中角色减速后，100的数据包内都按照没减速前的速度做为判断依据。关于加速的处理，由于角色在加速时是速度减小的，因此会立即被设置为新速度，并按照此速度进行处理。

#### 关于容错的处理

为了防止误判情况的发生，采用了如下容错处理。

1、在收到客户端的30个（可配置值）数据包中累计有15次（可配置值），两个数据包的时间间隔小于服务器上角色的移动速度。即50%的失败率

2、客户端两次数据包的时间间隔与服务器上角色速度之间允许有15毫秒（可配置值）的误差值。

#### 综述防范条件

客户端在发送给服务器的30个（可配置值）数据包中累计有15次（可配置值）两个数据包的时间间隔小于服务器上角色的移动数据（15毫秒的可配置误差值）

## 其他问题的讨论与解决办法：

### 对于移动数据包中时间戳可能被修改的情况分析



由于防类加速齿轮策略与防加速发送移动数据包的策略都是依赖于客户端在发送移动数据包中的时间戳，因此存在着通信协议被破解，并被修改的风险。

但考虑到用户修改协议的目的是加快移动数据包发送的速度来判断，因此可分为两种情况：

第一种情况，如图所示将移动数据包的时间戳（A1’）修改的小于正常值（A1）。在这种情况下，若修改的时间值过小则将符合防加速发送移动数据包的策略方案，因此会被此策略屏蔽到。

第二种情况，是将移动数据包的时间戳（A1’’）修改的大于或等于正常值（A1）。在这种情况下，若加快数据报的发送速度，则会与防类加速齿轮工具策略的条件相符。因此也会被此策略屏蔽。