手游《魂斗罗：归来》状态同步技术分享

聂 鹏

2019.5.16

分享简介

《魂斗罗.归来》是一款用Unity4制作的横版动作射击类手游，于2015年初立项，2017年6月正式上线，第一周即进入免费榜第一，畅销榜前五。

介绍魂斗罗中使用的状态同步技术，以及为应对横版射击游戏里，丰富的各种关卡元素，复杂的角色行为带来的联机同步技术挑战，所做出的努力尝试和优化过程。

内容大纲

一．同步方案
二．状态同步的基本框架
- 移动包
- 事件RPC
- 属性下发
- 计算一致性
- 拉扯校正

三．击飞击退同步效果优化
四．局部空间同步方案
五．拉扯问题排查
六．移动插值和预测
- ds移动包缓存
- ds预测
- 航位推测 vs 影子跟随

七．网络延迟/丢包
- 弱网络模拟

八．断线重连
九．反外挂
一〇．优化(流量/性能)

魂斗罗采用状态同步

相对于帧同步来说，状态同步的特点

优点：

- 客户端手感等同于单机
- 对网络延迟和抖动适应性强
- 能方便地断线重连
- 安全性高，易于进行反外挂

缺点：

- 开发较复杂
- 不容易单局录像
- 服务器负载高
- 流量消耗大

魂斗罗的状态同步方案

- 方案
  - 服务器上跑Unity
  - 客户端和服务器单局共用同一套代码。
- 优势
  - 便于保证同步计算一致性
  - 提高开发效率，方便联机联调。

状态同步的特点

- 客户端本地先行，本地操作不受网络延迟影响
- 服务器运行完整的单局逻辑
- 一切以服务器为准

状态同步的特点

- 计算一致性
  - 客户端和服务器执行的是同样的代码逻辑，计算结果应该是一样的。同样的代码逻辑应该有同样的计算结果，这是服务器对客户端进行验证的基础

状态同步技术要求

- 围绕计算一致性的技术要求
  1. 逻辑代码执行顺序一致
  2. 中间步骤的数值完全一致
  3. 撇除对本地的一切依赖（如本地时间类）
  4. 随机数序列（随机数使用相同的种子）
  5. 碰撞系统/物理引擎需要支持一致性
  6. …

状态同步的基本框架

- 事件RPC
  - 分可靠和不可靠，是客户端和DS之间的各种事件通知，如移动操作，技能播放，命中，死亡等
- 属性同步
  - 从DS到客户端，低频率的固定数据下发
  - 可以使用DeltaCompress对数据进行压缩[4]

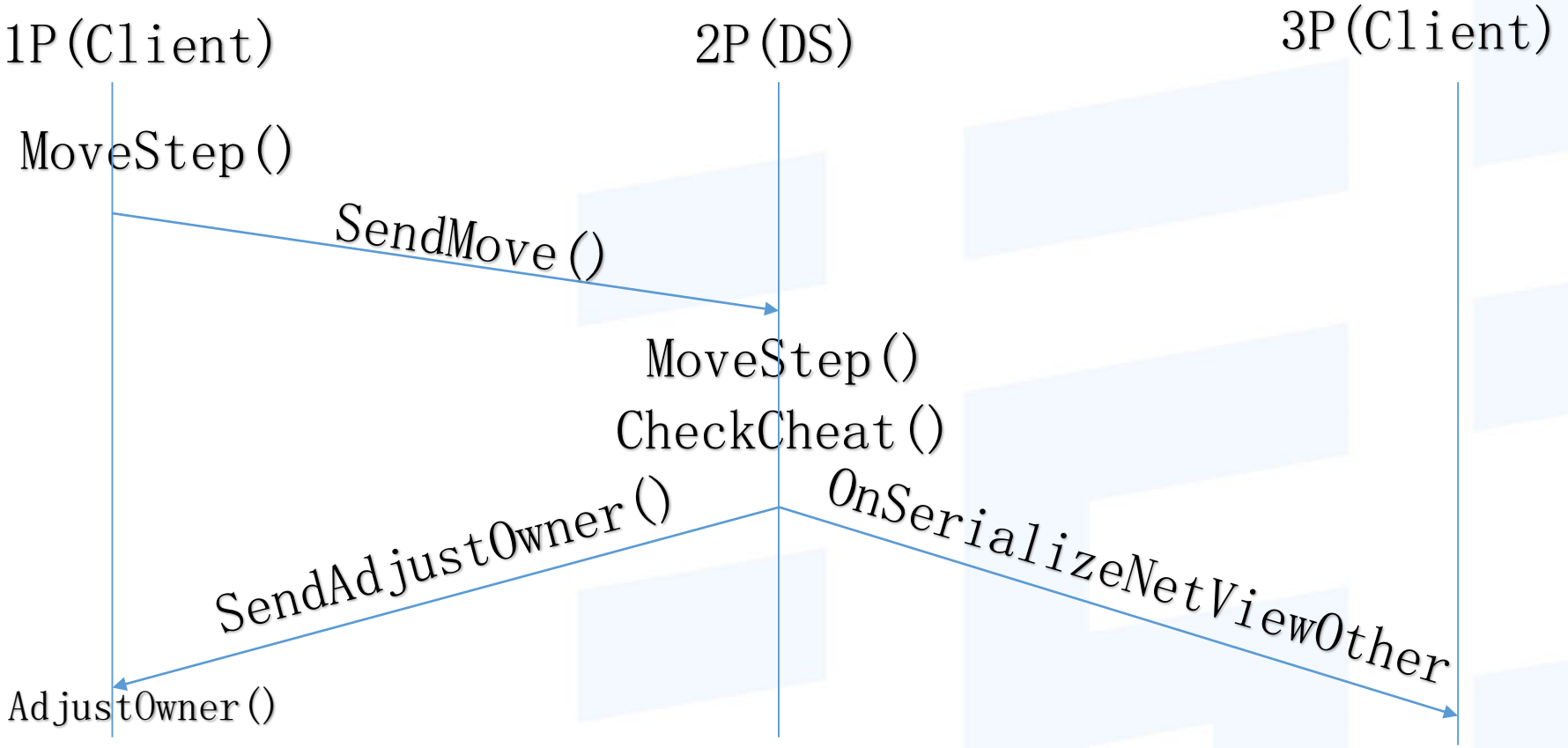
状态同步的基本框架

- 移动包
  - 1P玩家发送给DS的。
  - 每帧发送，包括玩家的各种操作信息。
- 校正包
  - DS发送给1P玩家的。
  - 当DS验证一致性失败时发送，对1P玩家进行纠正。

预期存在的几种合理拉扯

```
BePushed,           // 被推动，（组队关，坦克车推动主角移动）
EnterLeaveElevator, // 上下电梯
EnterLeaveCarrier,  // 上下载具
BeDropPVE,          // 组队PVE里主角被直接拉到队友旁边
DSOnlyMove,         // ds接管时间，自己模拟结束时，导致的拉扯
UnlockChunk,        // 组队PVE里，ds上箱庭先解锁了，不再对主角构成阻挡
ActorCol,           // 由于带碰撞阻挡的角色死亡回收，导致的拉扯
NextLevel,          // 组队切关卡，先前的关卡销毁了，新的关卡加载了，主角重置到新关卡出生点
ResetPos,           // 在ds上被重置位置 （掉入死亡面等）
Transfer,           // 主角瞬移技能
SkillFailed,        // 包含位移操作的那个技能释放失败了
```

状态同步的基本框架



MoveStep(): 执行一次移动更新
SendMove(): 发送移动包
CheckCheat(): 位置校验
SendAdjustOwner(): DS下发校正包
AdjustOwner(): 1P位置被纠正，回滚操作并重新执行
OnSerializeNetViewOther: 属性同步下发到3P

移动包的设计

- 使用固定的时间间隔来进行移动更新
  - 客户端和DS都使用相同的更新时间间隔
  - 每个更新后，客户端发送移动包给DS
- 使用序号来标识第几次更新
  - 移动包里带上序号和客户端的计算结果
- 客户端把每一个发送的移动包在本地加入到移动包缓存队列。

校正包的设计

- 校正包包括了DS上2P角色当前的所有信息
  - 包括更新序号（表示DS更新到哪一步了）
  - 2P最新位置
  - …
- 客户端用收到的校正包里的信息进行本地纠正

客户端校正步骤

1. 客户端先根据校正包里的更新序号把移动包缓存队列里过时的移动包移除。
2. 把当前位置以及各种状态值回滚到校正包里的值。
3. 将移动包缓存队列里剩余的移动包再按序号顺序执行一遍。

移动更新流程梳理

- 为了保证客户端和DS计算结果的一致性需要对逻辑代码的执行顺序进行梳理。

移动更新流程梳理

- 所有逻辑都是客户端先行，避免任何的反例。
- 使用统一的位置更新入口(MoveStep)
  - 入口唯一
  - 更新顺序固定
- 不能直接使用Unity的刚体移动
  - 没法自己控制刚体更新时机
- 所有移动相关的逻辑状态也要纳入MoveStep

击飞击退同步效果优化

- 无拉扯高延迟版本
  - 击飞事件从ds到受击者客户端，再到ds，再到攻击者客户端
- 有拉扯低延迟版本

- 无拉扯低延迟版本
  - 3P进行客户端预表现

- 视频1 vs 视频2

局部空间同步方案

- 联机电梯
  - 联机关卡里存在的移动中的碰撞板，电梯移动不受玩家控制

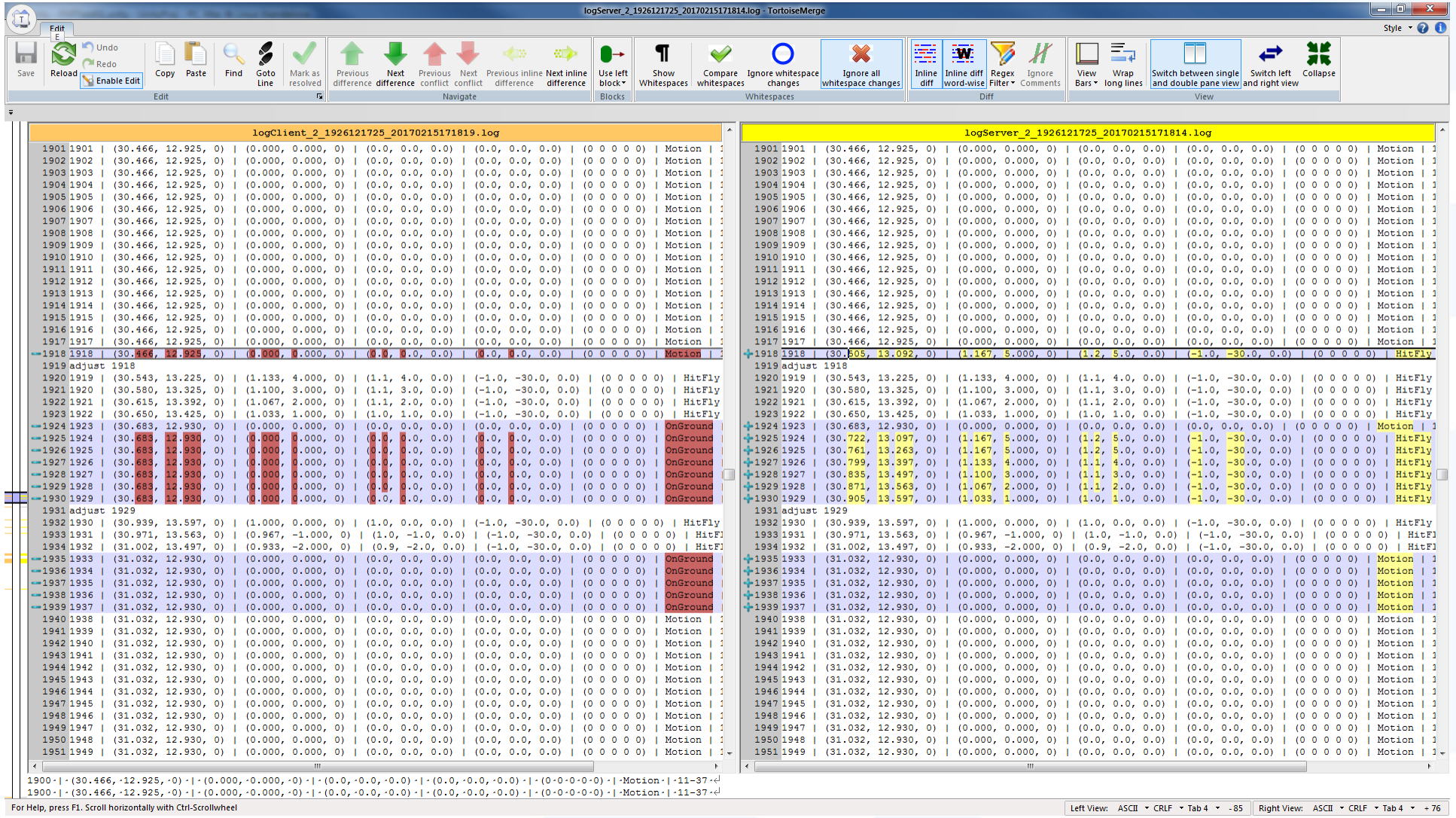
- 玩家+电梯，如何同步？
  - 处在电梯上时，玩家位置使用电梯的局部空间坐标进行上报和校验。
  - 进入/离开电梯，切换玩家坐标空间

如何排查一致性问题

- 计算出现不一致时会导致不同程度的拉扯，影响体验
  - 拉扯具有偶发性，不好重现
  - 受网络条件影响

拉扯问题如何排查

- 日志记录
  - 在客户端和DS上同时打日志。
  - 日志里记录移动更新序号，以及各种中间计算结果。
  - 基于日志记录对比客户端和DS的中间计算结果，找到是在哪一步不一致了，进而分析导致不一致的具体原因。


logServer_2_1926121725_20170215171814.log - TortoiseMerge
Edit
Save
Reload
Undo
Redo
Enable Edit
Copy
Paste
Find
Goto Line
Mark as resolved
Previous difference
Next difference
Previous conflict
Next conflict
Previous inline difference
Next inline difference
Use left block
Show Whitespaces
Compare whitespaces
Ignore whitespace changes
Ignore all whitespace changes
Inline diff
Inline diff word-wise
Regex Filter
Ignore Comments
View Bars
Wrap long lines
Switch between single and double pane view
Switch left and right view
Collapse
Style
Edit
Navigate
Blocks
Whitespaces
Diff
View
logClient_2_1926121725_20170215171819.log
logServer_2_1926121725_20170215171814.log
For Help, press F1. Scroll horizontally with Ctrl-Scrollwheel
Left View: ASCII CRLF Tab 4 - 85
Right View: ASCII CRLF Tab 4 + 76

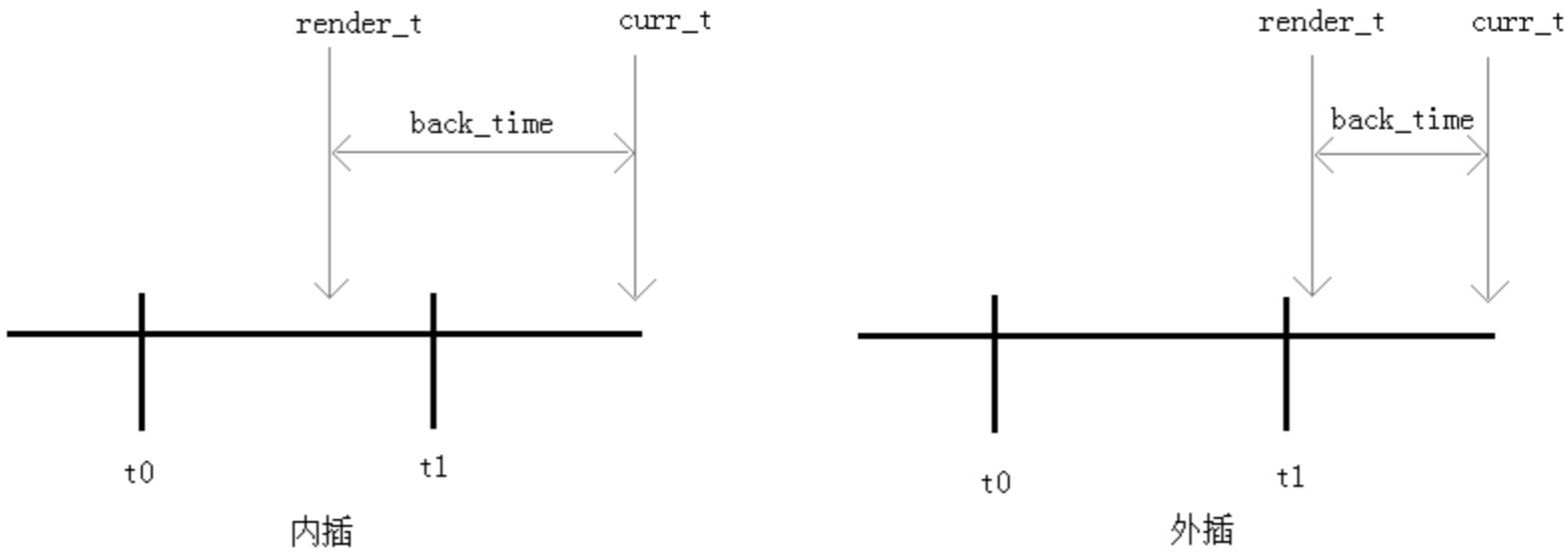
3P角色的移动同步

- 3P角色的移动，及时性和平滑性
  - 3P玩家相对于1P玩家的位置滞后，这导致射击命中判定的矛盾
  - 影子跟随[1] VS 航位推测[6]

影子跟随

- 使用收到的历史位置进行插值（内插/外插）
- 从当前时间curr_t往前回退一段时间back_time，得到render_t，如果render_t在最后一个包之前，则进行的是内插；如果render_t在最后一个包之后，则进行外插。

影子跟随

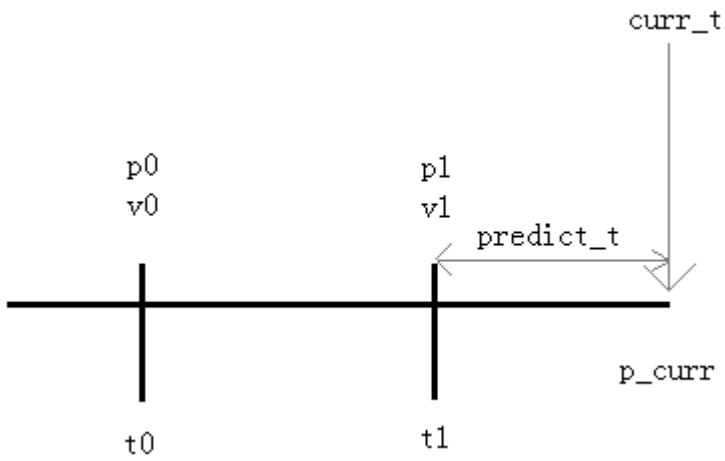


- 如果back_time太小，那么外插就比较多，容易导致不平滑
- 如果back_time太大，那么就不够及时，延迟大
- 低延迟和平滑性难以兼顾

航位推测

- 航位推测法（英语：Dead reckoning，缩写：DR）是一种利用现在物体位置及速度推定未来位置方向的航海技术。英语中“Dead”是从“deduced（推导）”转化而来。

航位推测



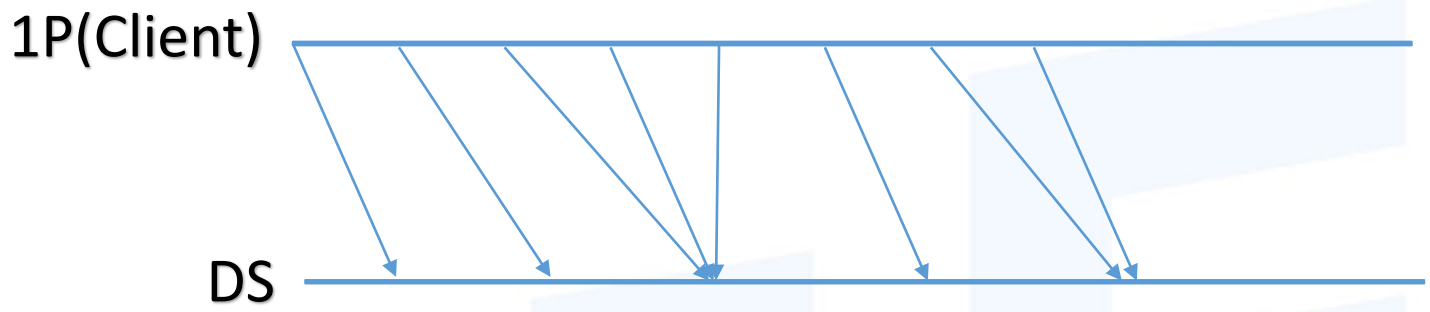
- 影子跟随方法仅仅用到了位置，没有利用到速度。
- 航位推测方法，ds按固定间隔下发位置和速度。
- 速度比较平滑的情况下比较适用。
- 用最后一次的速度来预测当前位置p_curr 。

```
p_curr = p1 + v1 * predict_t
```

3P玩家移动延迟优化

3P玩家移动平滑性优化

网络原因导致的不平滑



问题:

1P向DS发送固定间隔的MoveStep，由于网络拥塞等原因，DS可能收包间隔并不均匀，这样就导致2P的移动不平滑。同时也带来DS性能的波动。

DS移动包缓存

- 解决:

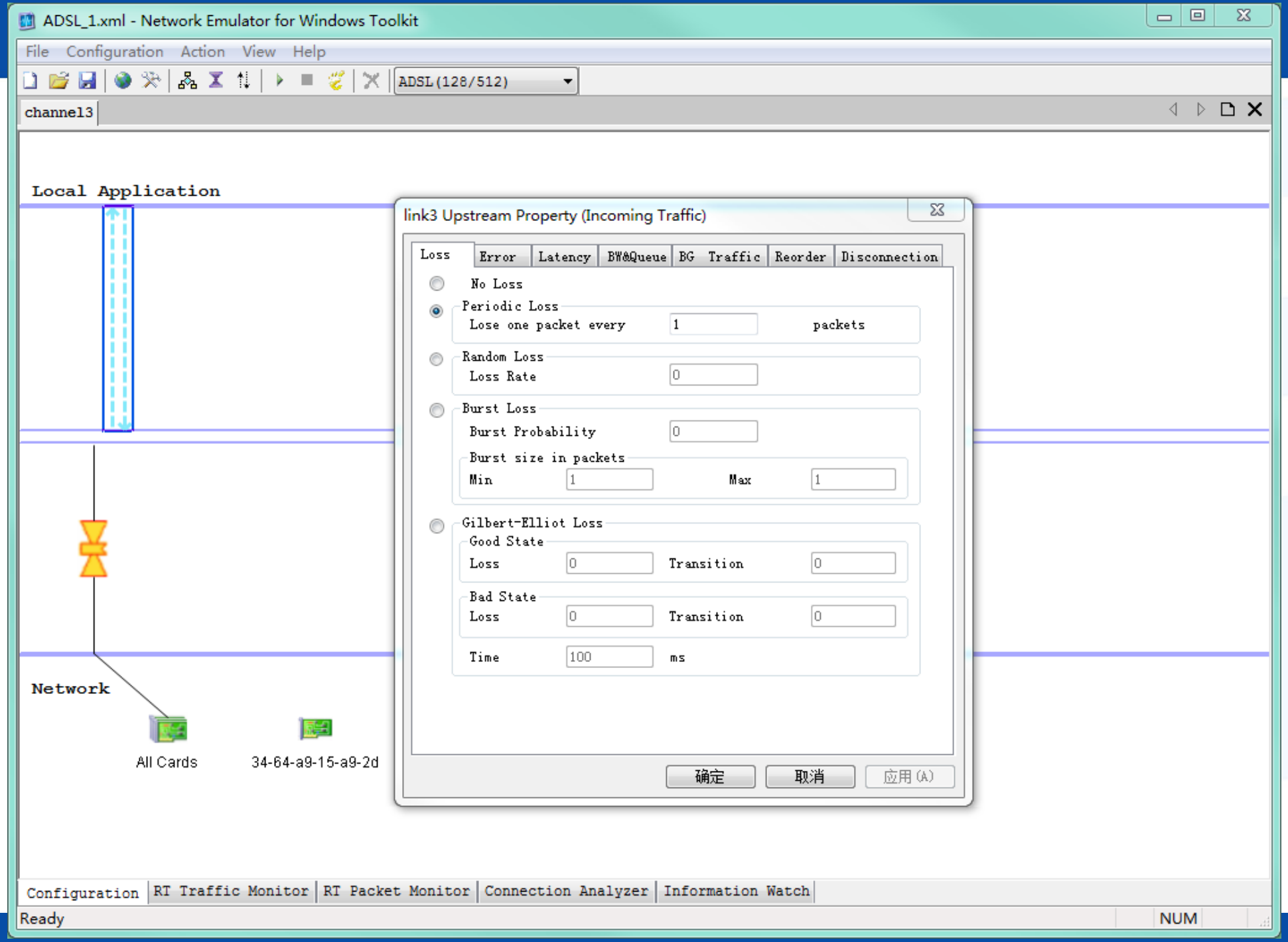
DS收包后不是立即执行MoveStep，而是缓存到队列，而后从队列里取包执行，避免一帧里执行太多MoveStep，解决了2P的移动平滑性。

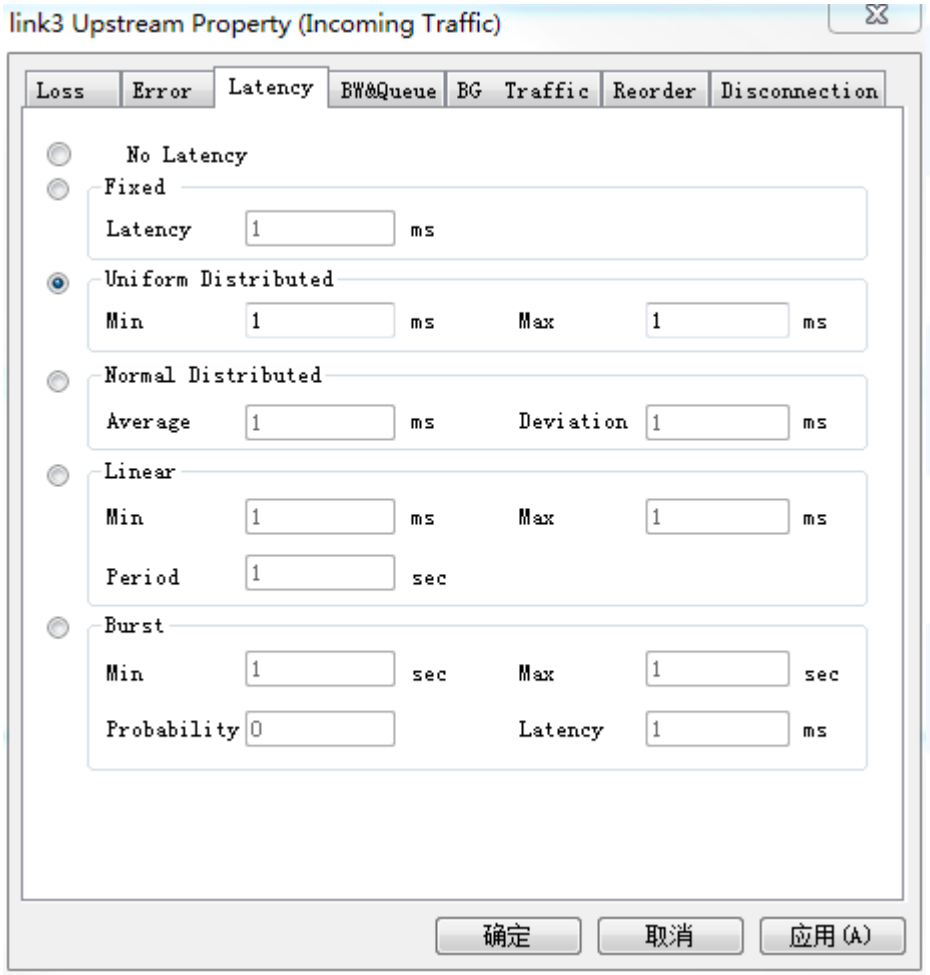
DS移动预测

- 问题：
  - ds侧收包时间间隔并不均匀，可能会相邻两个包间隔时间很大，在这个间隔时间里，ds没有执行MoveStep，没有位置更新。
- 分析：
  - 大量的时间里，其实是没有操作的，移动包仅仅是一个位置上报和MoveStep更新驱动。
- 解决：
  - ds在正常的延迟内没有收到包，就进行自行预测，等收到包后再确认，预测失败即回滚，再按收到的包进行更新。

弱网络测试

- 高延迟（ping>350）时主要是会感觉到其他玩家移动滞后，以及一定的不平滑感
- 弱网络模拟工具"Network Emulator Client"介绍


ADSL_1.xml - Network Emulator for Windows Toolkit
File
Configuration
Action
View
Help
ADSL(128/512)
channel3
Local Application
Network
All Cards
34-64-a9-15-a9-2d
link3 Upstream Property (Incoming Traffic)
Loss
Error
Latency
BW&Queue
BG Traffic
Reorder
Disconnection
No Loss
Periodic Loss
Lose one packet every
1
packets
Random Loss
Loss Rate
0
Burst Loss
Burst Probability
0
Burst size in packets
Min
1
Max
1
Gilbert-Elliot Loss
Good State
Loss
0
Transition
0
Bad State
Loss
0
Transition
0
Time
100
ms
确定
取消
应用(A)
Configuration
RT Traffic Monitor
RT Packet Monitor
Connection Analyzer
Information Watch
Ready
NUM


link3 Upstream Property (Incoming Traffic)
Loss
Error
Latency
BW&Queue
BG Traffic
Reorder
Disconnection
No Latency
Fixed
Latency
1
ms
Uniform Distributed
Min
1
ms
Max
1
ms
Normal Distributed
Average
1
ms
Deviation
1
ms
Linear
Min
1
ms
Max
1
ms
Period
1
sec
Burst
Min
1
sec
Max
1
sec
Probability
0
Latency
1
ms
确定
取消
应用(A)

断线重连

- 支持网络掉线重连和杀进程重连。
- 掉线后，服务器上2P玩家仍会继续更新，只是没有操作了。
- 重连时，DS会把2P玩家的数据下发给客户端，以供恢复状态。

反外挂

- 命中欺骗外挂
  - 上报命中检测数据（子弹出射点坐标，命中点坐标，子弹发射方向等），ds进行物理射线检测，校验是否命中
- 时间加速挂
- 移动加速挂

流量优化

- 流量优化
  - 降低属性下发频率
  - RPC里避免字符串参数，尽量使用id查表方式。
  - 使用DeltaCompress数据压缩
  - 合并同类的属性下发包，减少不必要的包头数据

单局DS承载

- 每个DS起一个服务器进程
- DS性能优化
  - DS进程预拉起，资源预加载，DS单局复用。
  - DS避免跑骨骼动画，除开少数特殊例子里骨骼上绑了攻击盒的。
  - 除开渲染和音效外，跟客户端单局性能优化类似。
- M1单机承载大约

游戏模式	房间人数	CPU	内存	M1单机承载
PVP	6	3.49%	133MB	2003人
组队PVE	2	2.41%	134MB	956人

参考资料

[1] Source Multiplayer Networking

[2] UnrealEngine/Networking

[3] Networking and Online Games, Grenville Armitage, Mark Claypool, Philip Branch 2006 John Wiley & Sons, Ltd

[4] https://en.wikipedia.org/wiki/Delta_encoding

[5] https://gulu-dev.com/post/2016-07-24-id-network-model-evolution

[6] https://en.wikipedia.org/wiki/Dead_reckoning

[7] http://mrelusive.com/publications/papers/The-DOOM-III-Network-Architecture.pdf

腾讯游戏学院
TENCENT INSTITUTE OF GAMES

THANK YOU