2017 年清华大学“华罗庚杯”数学建模竞赛

承诺书

我们仔细阅读了清华大学数学建模竞赛的竞赛规则.

我们完全明白，在竞赛开始后参赛队员不能以任何方式（包括电话、电子邮件、网上咨询等）与队外的任何人（包括指导教师）研究、讨论与赛题有关的问题。

我们知道，抄袭别人的成果是违反竞赛规则的, 如果引用别人的成果或其他公开的资料（包括网上查到的资料），必须按照规定的参考文献的表述方式在正文引用处和参考文献中明确列出。

我们郑重承诺，严格遵守竞赛规则，以保证竞赛的公正、公平性。如有违反竞赛规则的行为，我们将受到严肃处理。

我们参赛的题目是：A 题

参赛队员(打印并签名) ：1.张锦苏

2.谢倩

3.陈经基

日期：2017 年 4 月 25 日

网络侧变量估计用户视频体验建模

摘要

随着网络应用的发展，人们对互联网用户体验（QoE）的需求逐渐提升。本题（网络侧估计终端用户体验建模）就是一个很好的例子。越来越多的用户选择在移动智能终端上用应用客户端 APP 观看网络视频，这是一种基于 TCP 的视频传输及播放。为了更充分地理解网络视频传输的原理，进而通过提供的实验数据建立用户体验评价变量（初始缓冲时延，卡顿时长占比）与网络侧变量（初始缓冲峰值速率，播放阶段平均下载速率，E2E RTT)之间的函数关系，我们对这个过程展开了若干方面的分析和建模，分别如下：

初始缓冲加载阶段：

- 数据方面，分析首先将所有 RTT 相同的数据分为一组，然后通过对于具有相同 RTT 的数据点的观察，得到在 RTT 为定值的情况下初始缓冲时延和初始缓冲峰值速率之间的函数关系，并且通过观察该函数关系里所涉及的参数关于 RTT 变化趋势得到参数关于 RTT 的函数关系，从而总结出一个描述初始缓冲时延和 RTT 以及初始缓冲峰值速率之间函数关系的经验公式。
- 从计算机网络理论方面，我们依据 TCP 原理对初始缓冲时延从简至繁进行建模。我们首先了固定拥塞窗口数量，对窗口大小是否足以充分利用带宽进行了分类讨论，推导出简单模型，得到了关于初始缓冲峰值速率、端到端环回时间（E2E RTT）、最大报文段（MSS）、初始缓冲下载数据量的分段函数关系式。从绘制出的图形发现没有考虑丢包率，于是又将丢包因素加入到模型中。接着依据 TCP 慢启动原理推导一个复杂模型，同样得到了分段函数关系式，也类似地考虑了丢包因素。最后将拥塞避免机制也加入到模型中，依据拥塞控制算法编写了模拟程序，将模拟初始缓冲时延和实际初始缓冲时延进行比较。

在视频播放过程中，卡顿严重影响了用户体验（QoE）。我们着重分析视频卡顿占比部分，首先为了方便起见，将所有出现卡顿的三千余条数据单独取出进行处理。

- 数据方面，通过比对、观察数据特性，总结出了一个卡顿时长占比关于播放阶段平均下载速率的经验公式。
- 理论方面，我们从 TCP 启动机制和视频播放物理过程出发，视频卡顿现象进行建模描述，所建立模型分为了平均下载速率快和平均下载速率慢两个部分，其中前者说明了在较快的下载速度下视频卡顿主要归咎于基于加载停滞的不当缓冲策略，而后者则通过一个简单的追及模型定量了描述了下载速率较慢的情况下卡顿占比和平均下载速率的关系，并且和前面所得到的经验公式的相关部分进行了比较。除此之外，我们对停滞时间、加载过程（比如余项）也进行了更加详细的分析。

背景介绍和题目重述

上个世纪八十年代以来，随着信息技术的发展，网络规模逐渐增大。Internet 的前身是 ARPAnet，军方借此传输数据。转为民用之后，得到了迅速发展。而供给方为了提高通信效率、增大通信可靠性，逐渐在竞争中达成了统一，即以光纤和以太网为基础，以 TCP/IP 协议栈为纽带的网络体系结构。

网络的普遍民用化带来了网络应用的发展。传统媒体（如电视、报纸）逐渐走低，互联网媒体则开启了它的繁荣之路。网络视频是互联网媒体的重要组成部分，其优点非常明显：内容丰富，选择多样，形式交互。根据尼尔森 2013 年报告，18 岁以下的青少年平均每周观看 36 分钟的移动视频，观看网络视频的时间为 21 分钟；18-24 岁的青年平均每周观看移动视频为 33 分钟，观看网络视频为 1 小时 21 分钟。另外，有调查显示，2017 年全世界移动数据流量的 2/3 将为视频。从实际生活出发，在观看视频的过程中，我们都希望没有卡顿，流畅地从头看到尾。但是，卡顿现象并不罕见：在九万余条数据中，或多或少地出现了三千余次卡顿现象。原先，我们只要求“看到视频”，而现在，我们对网络应用性能的要求逐渐提升，不但希望可以看到视频，更希望可以看得流畅、清晰。

我们注意到，对流畅性能的满足实际上表明了人们对时间特性的需要逐渐增大。传统的 TCP 协议已经可以较好地保障顺序性和可靠性，而“怎样看得流畅”则和缓冲时延、卡顿时延等时间因素密切相关。正如题目所说，看网络视频影响用户体验的两个关键指标是初始缓冲等待时间和在视频播放过程中的卡顿缓冲时间，用户体验可以用初始缓冲时延和卡顿时长占比（卡顿时长占比 = 卡顿时长 / 视频播放时长）来定量评价。

常用符号表达

符号名	对应表格列数	含义
V_{init}(kbps)	B	初始缓冲峰值速率
D_{init} (bytes)	N	初始缓冲加载量
$D_{init}^{'}$		初始缓冲加载量 $T_d \times Rate$
T_d (ms)	E	初始缓冲时延
$thre$		阀值
p		丢包率
N		次数
V_{ave}		总阶段视频下载平均速率
V_{buffer}	D	播放阶段视频
$Rate$		视频码率
T_{tot}		视频总时长
T_{play}	J	播放时长
T_{stall}	I	卡顿时长
T_{buffer}		实际加载时长
$T_{bufferstall}$		实际加载卡顿
$D_{totbuffer}$		视频下载总量
$D_{totplay}$		播放总量
D_{tot}		视频总量
D_{re}		重播放下载量
D_{rest}		视频余量

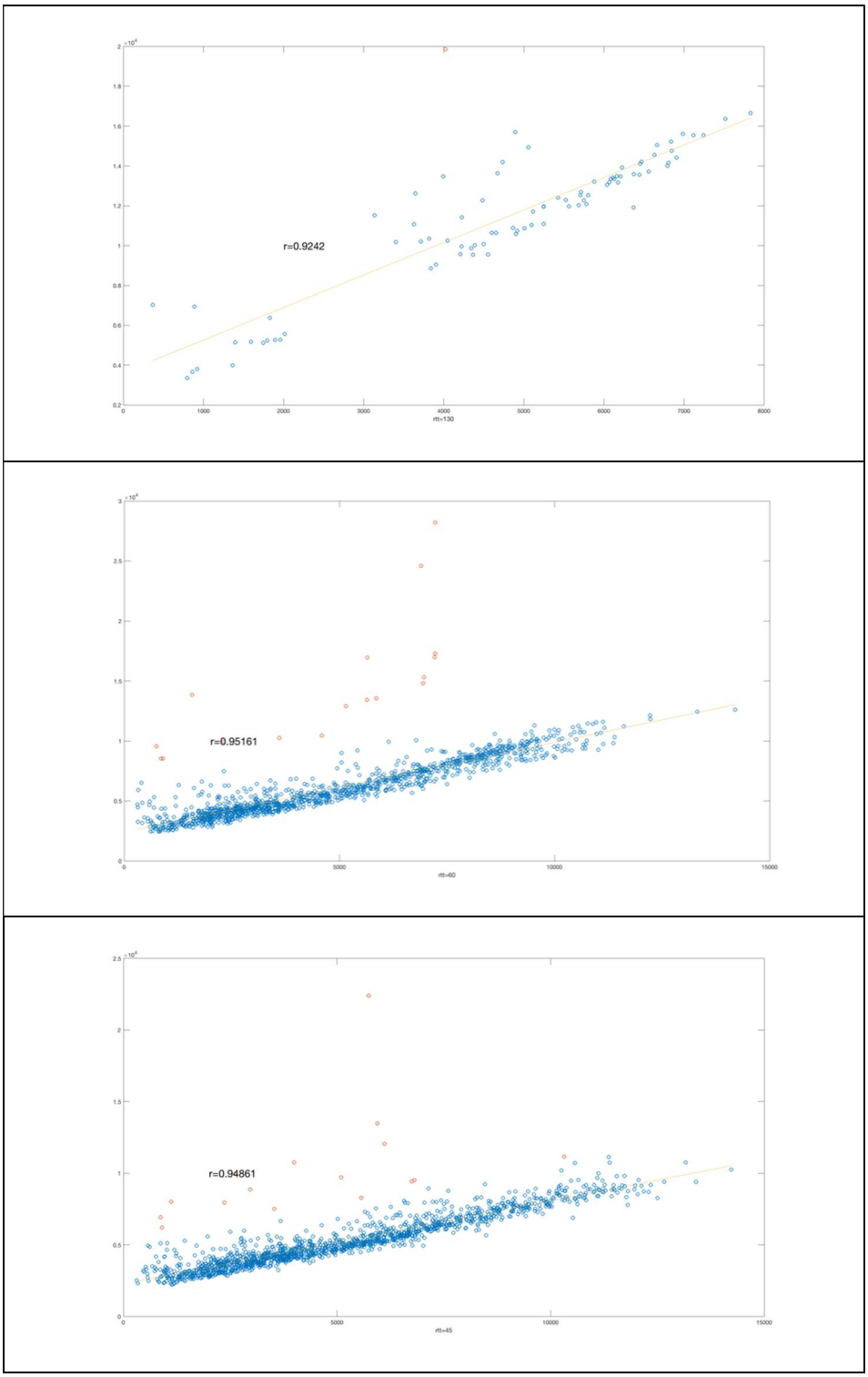
专业名词

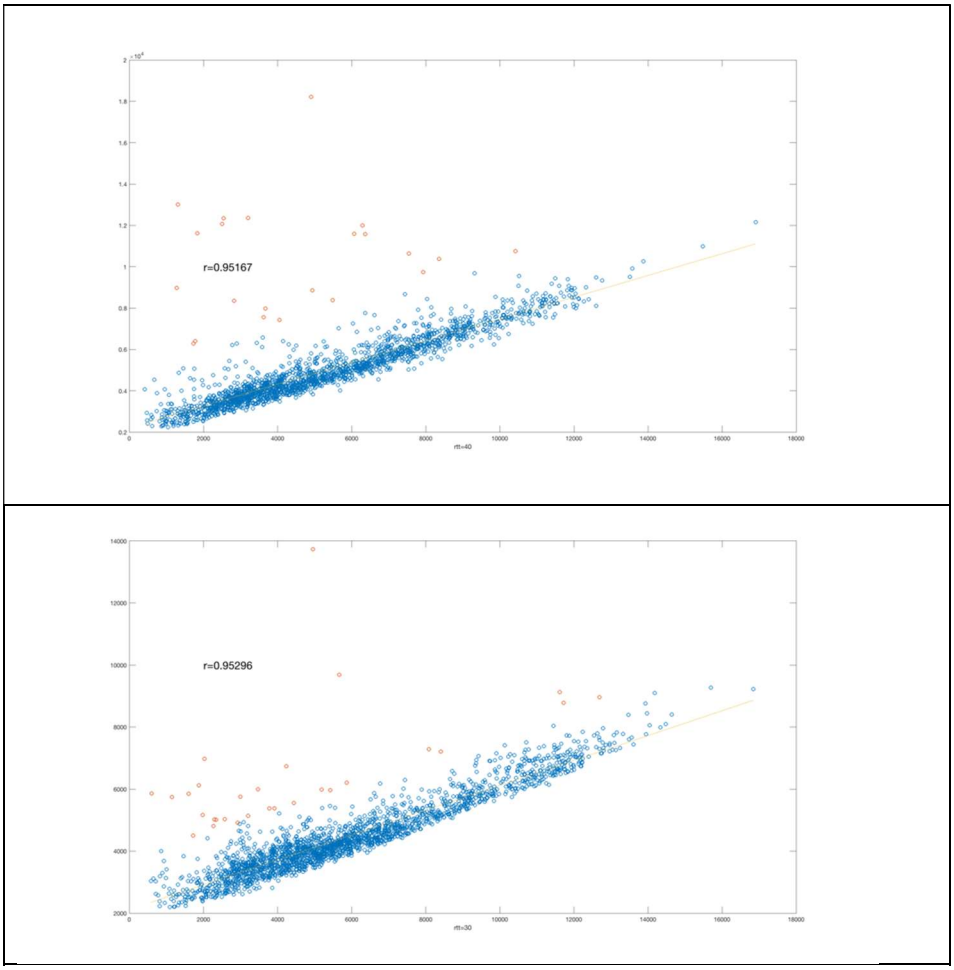
名词	解释
TCP	因特网运输层的面向连接的可靠的运输协议
MSS	最大报文段长度，典型值为 1460byte
RTT	往返时间
ACK	肯定确认
$cwnd$	为拥塞窗口，是发送方维护的一个限制发送速率的变量，发送方的未确认发送数据量不会超过 cwnd

视频初始缓冲阶段：依据数据建模

考虑到 RTT 的取值只有一百余种，因此首先不妨将所有数据按照 RTT 进行分组，将所有 RTT 相同的点分为一组，然后考虑在 RTT 相同的情况下通过寻找数据之间的联系来探讨初始缓冲时延 T_d和初始缓冲峰值速率v_{init}之间的函数关系。

不妨将作为$v_{init} \times T_d$为纵坐标，T_d作为横坐标，将相同 RTT 的点画在同一个坐标系上如下：

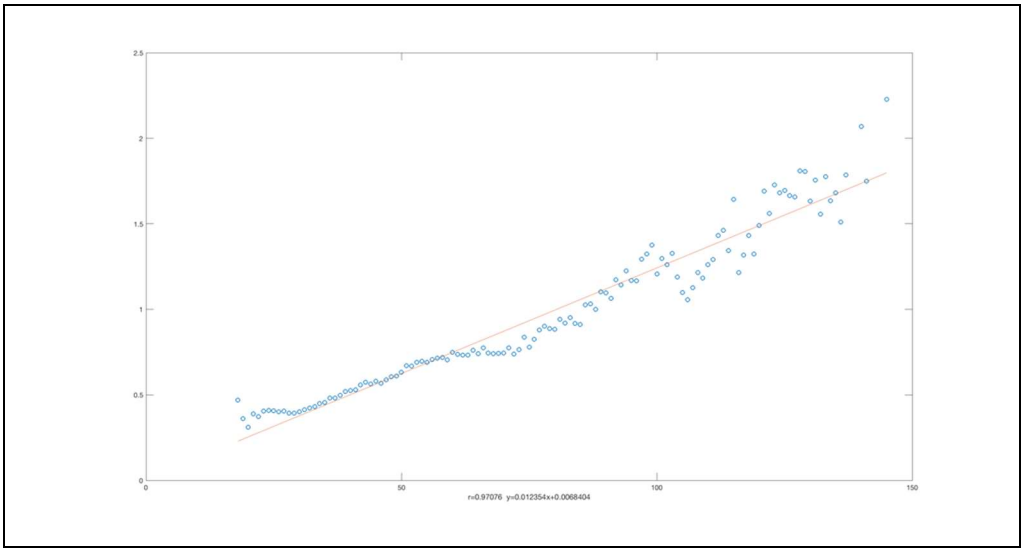


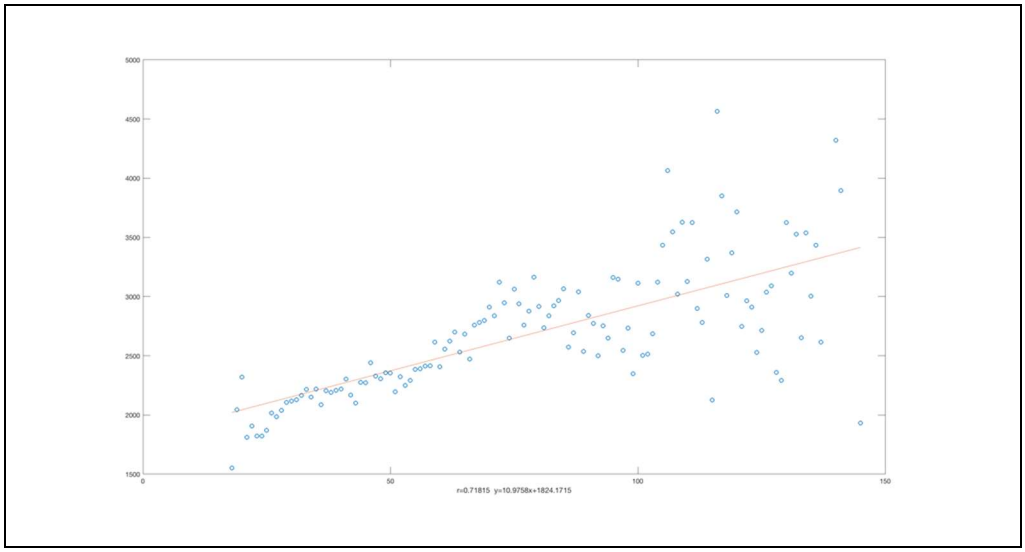


可以发现在 RTT 一定的情况下，和基本呈现线性关系，在筛选掉了极少数的极端数据（占所有数据的 1.3%）之后，$\mathrm{v_{init}} \times T_d$和$\mathrm{v_{init}}$之间的线性相关系数绝大多数在 0.9 以上，因此可以认为$\mathrm{v_{init}} \times T_d$和$\mathrm{v_{init}}$之间的函数关系为：

$$v_{init} \times T_d = a(\mathrm{RTT})\, v_{init} + b(RTT)$$

通过最小二乘拟合可以得到$a(RTT$ 和$b(RTT$ ，其和 RTT 的关系如下图所示（图中横轴为 RTT，纵轴为 a、b，第一张图表示 a 与 RTT 的关系，第二张图表示 b 与 RTT 之间的关系）：





可以发现 a,b 基本和 RTT 呈现线性关系，这样的话就有：

$$\mathrm{a(RTT)} = \mathrm{a}_1 RTT \quad {}_1$$
$$\mathrm{b(RTT)} = \mathrm{a}_2 RTT + b_2$$

则：

$$\mathrm{v_{init}} \times T_d = (a_1 RTT \quad {}_1)\mathrm{v_{init}} + a_2 RTT \quad {}_2$$

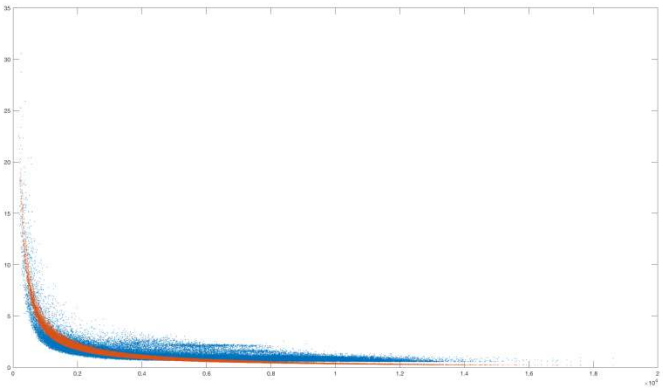
因此：

$$\mathrm{T_d} = a_1 RTT + b_1 + \frac{a_2 RTT}{v_{init}} + \frac{b_2}{v_{init}}$$

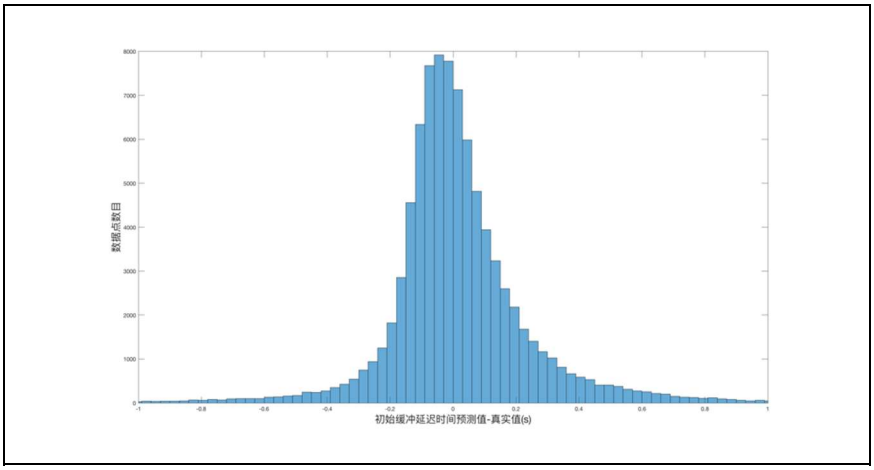
这样就得到了一个$\mathrm{T_d}$关于RTT和$\mathrm{v_{init}}$之间的函数形式，之后使用最小二乘拟合求出其中的参数如下：

$$\mathrm{a}_1 = 0.010349$$
$$\mathrm{b}_1 = -0.04729$$
$$\mathrm{a}_2 = 10.54623$$
$$\mathrm{b}_2 = 2364.455$$

这样就得到了一个使用 RTT 和$\mathrm{v_{init}}$预测$\mathrm{T_d}$的函数，预测的数据点（横轴为最大峰速率，纵轴为初始缓冲延迟时间）（红色数据点）和真实数据点（蓝色数据点）如下图所示：



用其拟合数据得到的残差（预测值-真实值）的分布如下直方图所示：



可以发现误差基本均匀地分布在 0 附近，这说明以上的公式还是较好地反应和T_d和 RTT 与v_{init}之间的关系。

视频初始缓冲部分：依据原理建模

TCP 快速重传机制：如果 TCP 发送方接收到对相同数据的 3 个冗余的肯定确认（Acknowledgement，ACK），说明在这已被确认过 3 次的报文段之后的报文段已经丢失，发生“丢包”现象，立即执行快速重传。依据 RFC 5681，在确认期望序号的报文段后，对下一个按序报文段的到达最多等待 500ms，如果下一个按序报文段在这个时间间隔内没有到达，则发送一个 ACK。

TCP 三次握手机制：依据 TCP“面向连接”的特性，在一个应用进程可以开始向另一个应用进程发送数据之前，这两个进程必须先相互“握手”，它们相互发送某些预备报文段，以建立确保传输数据的参数。报文段的数据量受限于最大报文段长度（Maximum Segment Size，MSS）。

TCP 拥塞控制机制：TCP 发送方确定发送速率，使得既不出现网络拥塞，与此同时又能充分利用所有可用的带宽。Jacobson 在 1988 年提出的拥塞控制算法包括 3 个主要部分：①慢启动；②拥塞避免；③快速恢复。

其中慢启动和拥塞避免是 TCP 的强制部分，快速恢复是推荐部分，对 TCP 发送方并非是必需的。慢启动和拥塞避免的差异在于对收到的 ACK 做出反应时增加拥塞窗口（Congestion Window，cwnd）的长度，拥塞窗口是发送方维护的一个限制发送速率的变量，发送方的未确认发送数据量不会超过 cwnd.

视频初始缓冲阶段分为 3 个子阶段：

（1）视频解析，持续时长和 OTT 平台、终端的设计原理有关，通常为 RTT 的某个倍数，设为 $xRTT$ 。

（2）三次握手，持续时长为 $1.5RTT$ 。

（3）数据下载缓冲。

模型假设：

1. 不考虑 TCP 实际协议中的超时机制。
2. 不考虑 TCP 重传机制结合 GBN 协议和 SR 协议的特性，即只考虑三次冗余确认后快速重传。
3. 不考虑拥塞控制机制中快速恢复的部分。

符号定义：

V_{init}：初始缓冲峰值速率

RTT：端到端环回时延

MSS：最大报文段长度，取推荐值 1.46KB

D_{init}：初始缓冲需要下载的数据量，为初始缓冲量 4s 乘以视频码率 2934kbps 所得 1.46MB，相当于 1000MSS

T_d：初始缓冲时延的拟合值

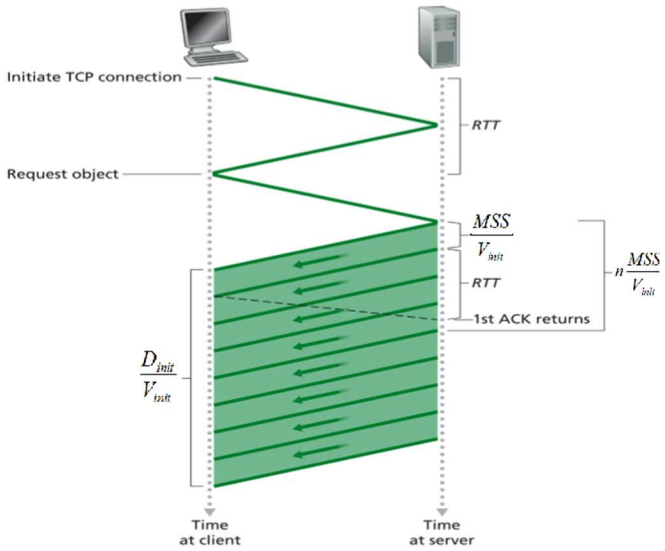
数据预处理：

剔除了 47 条初始缓冲实际下载数据量远小于初始缓冲需要下载数据量 1.46MB 的数据。

简单模型：假设拥塞窗口的大小是固定的，为 n 个报文段（MSS）。

情形 1：窗口大小合适，充分利用带宽，此时 $n\frac{MSS}{V_{init}} > RTT + \frac{MSS}{V_{init}}$，即

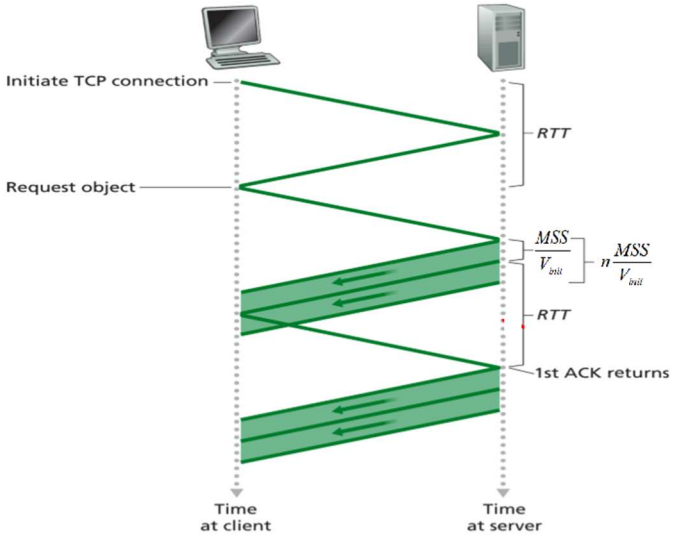
$n > RTT\frac{V_{init}}{MSS}+1$.



由图可知，初始缓冲时延为

$$T_d = xRTT + 2RTT + \frac{D_{init}}{V_{init}}$$
$$= (x+2)RTT + \frac{D_{init}}{V_{init}}$$
$$= (x+2)RTT + 1000\frac{MSS}{V_{init}}$$

情形 2：窗口较小，未能充分利用带宽，此时 $n\frac{MSS}{V_{init}} < RTT + \frac{MSS}{V_{init}}$，即 $n < RTT\frac{V_{init}}{MSS}+1$.



由图可知，初始缓冲时延为

$$T_d = xRTT + 2RTT + \frac{D_{init}}{V_{init}} + (m-1)(\frac{MSS}{V_{init}} + RTT - n\frac{MSS}{V_{init}})$$
$$= (x+m+1)RTT + \frac{D_{init}}{V_{init}} - (m-1)(n-1)\frac{MSS}{V_{init}} \quad 。$$
$$= (x+m+1)RTT + [1000-(m-1)(n-1)]\frac{MSS}{V_{init}}$$

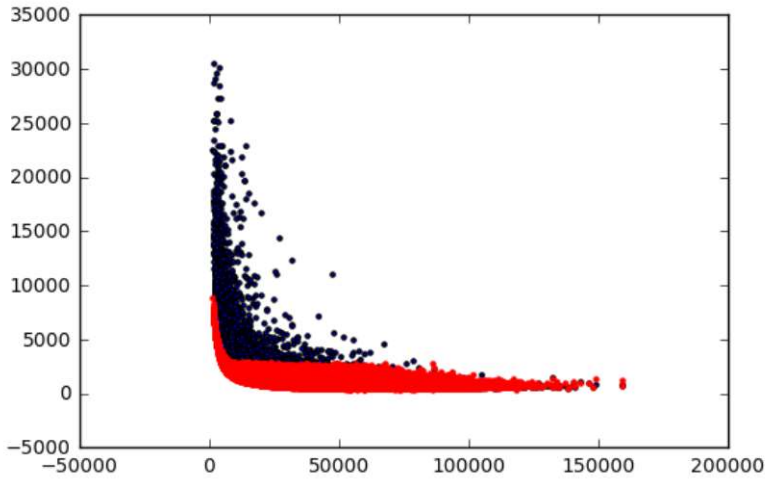
其中 $m = \frac{D_{init}}{nMSS} = \frac{1000}{n}$ 为服务器闲置的次数。

比如图中 D_{init}=4MSS， $n=2$， $m=2$.

结合两种情形可知，

$$T_d = \begin{cases} (x+2)RTT + 1000\frac{MSS}{V_{init}}, & n > RTT\frac{V_{init}}{MSS} + 1 \\ (x+\frac{1000}{n}+1)RTT + (n+\frac{1000}{n}-1)\frac{MSS}{V_{init}}, & n < RTT\frac{V_{init}}{MSS} + 1 \end{cases}$$

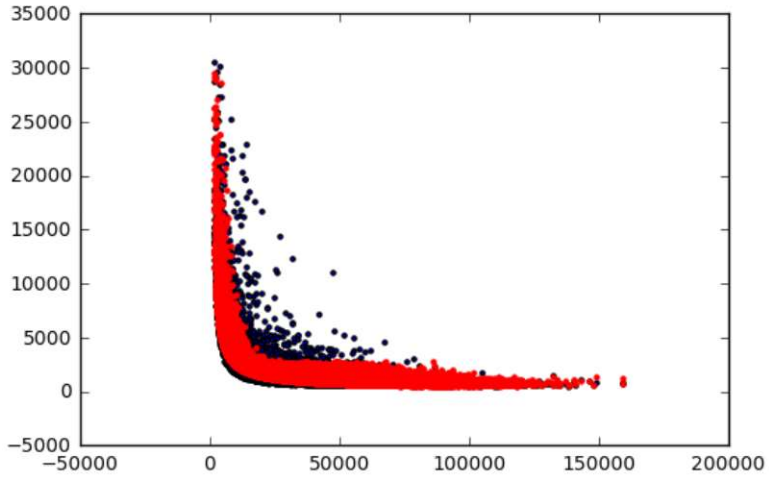
比如 $x=8$， $n=100$ 时对初始缓冲时延的拟合如下图所示（蓝色为原始数据，红色为拟合数据，横轴为初始缓冲峰值速率，纵轴为初始缓冲时延）



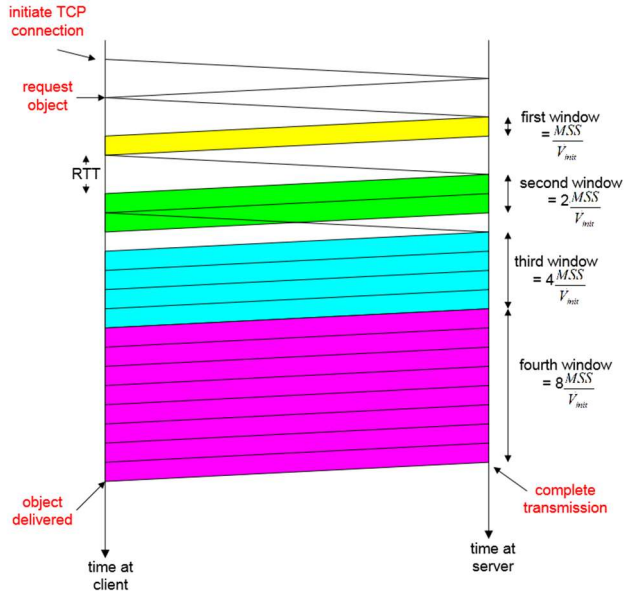
可以观察到当初始峰值速率较低时，拟合值与原始值有很大的误差，原因可能有三点：（1）我们用固定的 1.46MB 取代了第 N 列的实际初始缓冲下载数据量，在现实中往往比需要下载的 4s 视频缓冲量略多；（2）网速不稳定，达不到初始峰值速率；（3）没考虑丢包现象。下面将丢包因素加入模型中，丢包率的计算公式为 $p = 1.5\left(\frac{MSS}{RTT \times V_{init}}\right)^2$，因此可以估计1000 个报文段中有$1000p$ 次丢失，每次产生3个冗余 *ACK*，且等待每个 *ACK* 都花费最多500*ms*，因此重新估计初始缓冲时延为

$$T_d' = T_d + 1500 * 1000p = T_d + \left(\frac{1500MSS}{RTT \times V_{init}}\right)^2.$$

新拟合的结果如下图所示，拟合效果还是不错的，但依然不完美。



复杂模型 1：考虑慢启动过程，拥塞窗口呈指数增长。



视频解析时长 $xRTT$

服务器从发出报文段到接受 ACK 所需的时间为 $\frac{D_{init}}{V_{init}}+RTT$

发送第 k 个窗口所需的时间为 $2^{k-1}\frac{D_{init}}{V_{init}}$

发送第 k 个窗口后服务器的闲置时间为 $\max\left\{\frac{D_{init}}{V_{init}}+RTT-2^{k-1}\frac{D_{init}}{V_{init}},0\right\}$

由图可知，初始缓冲时延为

$$
\begin{aligned}
T_d &= xRTT + 2RTT + \frac{D_{init}}{V_{init}} + \sum_{k=1}^{y}\left(\frac{MSS}{V_{init}} + RTT - 2^{k-1}\frac{MSS}{V_{init}}\right) \\
&= (x+2)RTT + \frac{D_{init}}{V_{init}} + y\left(\frac{MSS}{V_{init}} + RTT\right) - (2^y - 1)\frac{MSS}{V_{init}} \\
&= (x+2+y)RTT + \frac{D_{init}}{V_{init}} + (y - 2^y + 1)\frac{MSS}{V_{init}} \\
&= (x+2+y)RTT + (y - 2^y + 1001)\frac{MSS}{V_{init}}
\end{aligned}
$$

其中 y 为服务器闲置次数，满足 $y = \min\{a-1, b\}$.

a 为达到初始缓冲下载数据量所需的窗口数，满足

$$
\begin{aligned}
a &= min\left\{k : 2^0 S + 2^1 S + \ldots + 2^{k-1} S \geq D_{init}\right\} \\
&= min\left\{k : 2^0 + 2^1 + \ldots + 2^{k-1} \geq \frac{D_{init}}{MSS}\right\} \\
&= min\left\{k : 2^k - 1 \geq \frac{D_{init}}{MSS}\right\} \\
&= min\left\{k : k \geq log\left(\frac{D_{init}}{MSS} + 1\right)\right\} \\
&= \left\lceil log\left(\frac{D_{init}}{MSS} + 1\right)\right\rceil \\
&\approx 10
\end{aligned}
$$

b 为服务器闲置次数的上限，满足

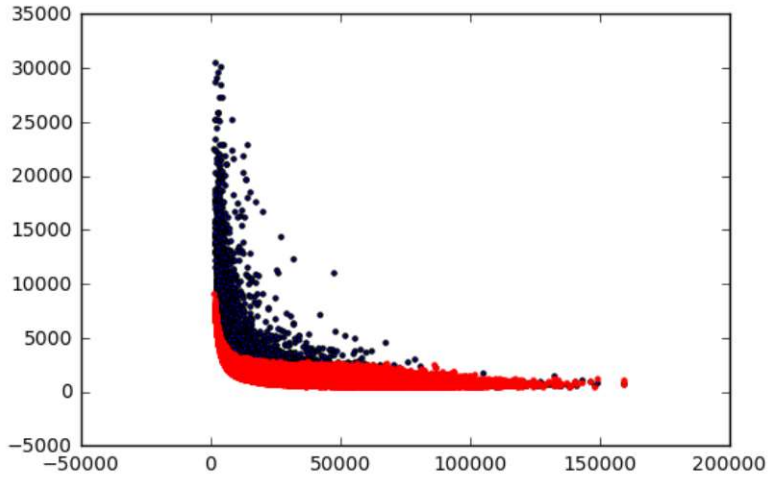
$$
\begin{aligned}
b &= max\left\{k : \frac{V_{init}}{MSS} + RTT - 2^{k-1}\frac{V_{init}}{MSS} \geq 0\right\} \\
&= max\left\{k : \left(2^{k-1} - 1\right)\frac{V_{init}}{MSS} \leq RTT\right\} \\
&= max\left\{k : 2^{k-1} - 1 \leq RTT\,\frac{V_{init}}{MSS}\right\} \\
&= max\left\{k : k - 1 \leq log\left(RTT\,\frac{V_{init}}{MSS} + 1\right)\right\} \\
&= \left\lceil log\left(RTT\,\frac{V_{init}}{MSS} + 1\right) + 1\right\rceil
\end{aligned}
$$

比如图中 $D_{init} = 15MSS$， $y = 4$， $a = 2$， $b = 2$， $T_d = 2RTT + 14\frac{MSS}{V_{init}}$.

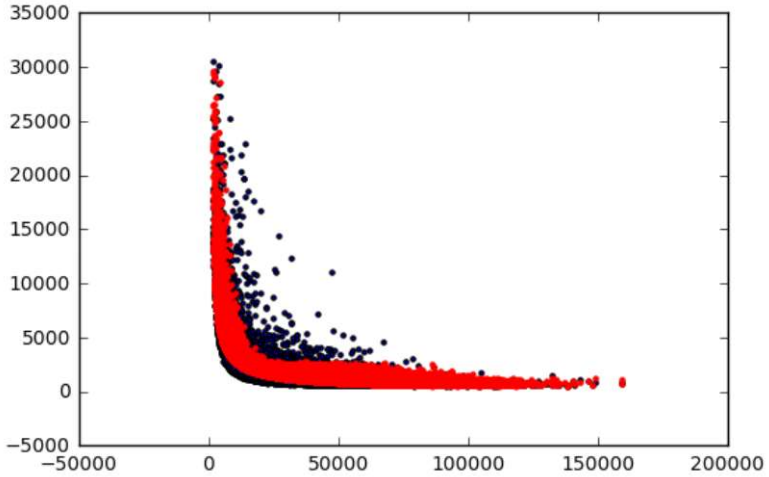
因此

$$T_d = \begin{cases} (x+11)RTT + 498\dfrac{MSS}{V_{init}}, & b > 9 \\ (x+2+b)RTT + (b-2^b+1001)\dfrac{MSS}{V_{init}}, & b \le 9 \end{cases}$$

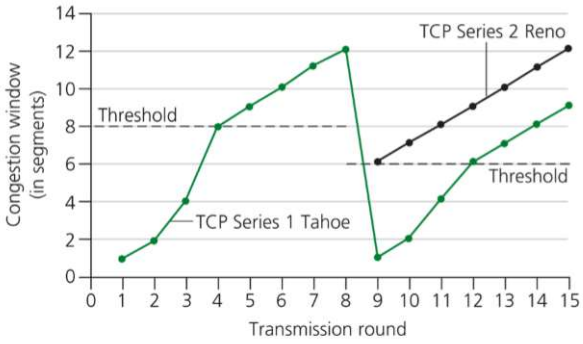
其中 $b = \left\lfloor log\left(RTT\dfrac{V_{init}}{MSS}+1\right)+1 \right\rfloor$.



出现了与简单模型相似的初始峰值速率较低时，拟合值与原始值有很大的误差的现象，类似地加入丢包因素得下图，并没有比简单模型有所改进。



复杂模型 2：考虑慢启动和拥塞避免过程，拥塞窗口呈指数增长，达到阈值后窗口减半，线性上升（如下图绿线部分）。



算法如下：

```
initialize: cwnd = 1
for (each segment ACKed)
  cwnd=cwnd+1
until (loss event OR cwnd>threshold)
Until (loss event)
{ every w segments ACKed:
cwnd++   }
threshold = cwnd/2
cwnd = 1
perform slowstart
```

视频播放卡顿部分：依据数据建模

为了简化问题，做出若干假设如下：

- 在视频播放阶段客户端网络条件稳定，即可以将视频下载的速率认为是一个恒定的值；
- 视频播放阶段不出现用户手动暂停，快进，回退等操作，视频播放全程不存在用户对视频播放过程的影响；

为了方便起见，不妨：

- 将播放阶段的平均下载速率记为V_{buffer}(kbps)；
- 将视频码率记为 $Rate$(kbps)；
- 将播放阶段总时长记为T_{tot}(ms)；
- 将视频播放时长记为T_{play}(ms)；
- 将卡顿时长记为T_{stall}(ms)；
- 将卡顿占比记为$Ratio$；
- 将初始缓冲数据量记为D_{init}(byte)

则显然应该有：

- $T_{tot} = T_{play} + T_{stall}$
- $Ratio = \frac{T_{stall}}{T_{play}}$

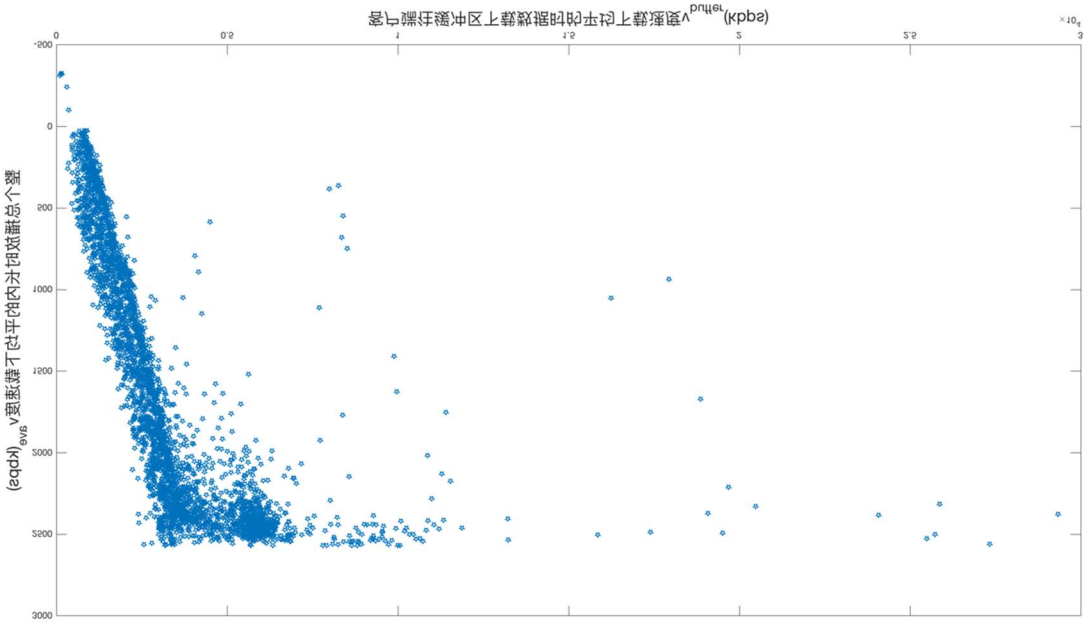
首先所给出的 90000 条数据里面大多数并不存在卡顿现象，为了方便处理，将所有出现卡顿现象的数据，约三千余条取出，以下数据集均指这三千余条出现卡顿的数据；

在处理数据之前不妨先初步探讨出现卡顿现象的原理，如果在线视频的缓冲区低于某一门限值（本题目要求中设为 0），则视频会暂停播放并且等待缓冲区数据量重新恢复到一定水平。[来源]由此看来，如果下载速率大于视频码率，则应该不会出现卡顿；但是数据集中却出现了不少平均下载速率大于视频码率，但是却出现了一定的卡顿（尽管卡顿占比较小），猜想由于某种原因，视频客户端并不会一直往缓冲区里下载数据，这才导致了卡顿的出现；

基于上述原因，应当把数据中的平均下载速度理解为在客户端往缓冲区里下载数据时候的平均下载速度，而在整个播放时长里地平均下载速度，记为v_{ave}，应当是小于v_{buffer}的一个量，计算v_{ave}如下：

- $v_{ave} = \frac{T_{play} \times Rate - D_{init}}{T_{tot}}$

并且将数据点按(v_{buffer}, v_{ave})绘制成图如下：



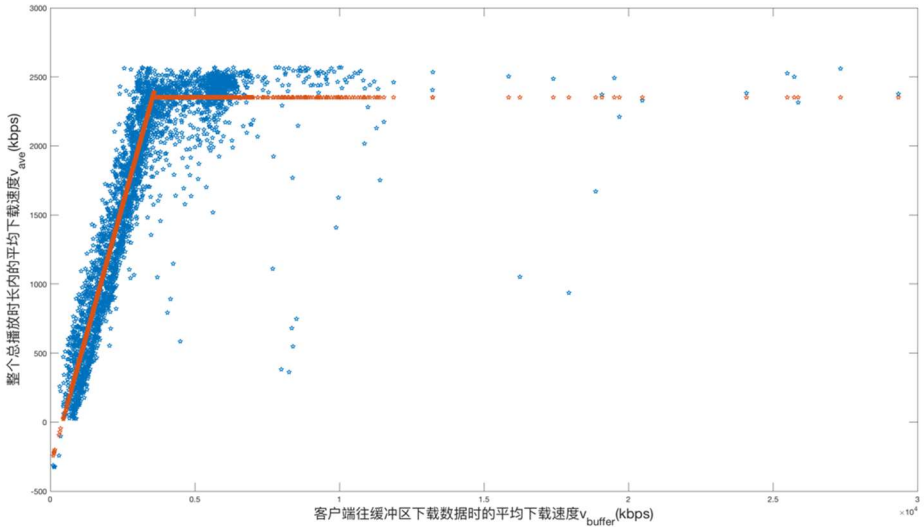
由上图可以观察发现，v和speed之间基本呈现出分段线性的关系，不妨令$v_{ave} = f(v_{buffer})$;
则：

- $f(v_{buffer}) = a \times v_{buffer} + b, \quad 0 \leq v_{buffer} \leq d$
- $f(v_{buffer}) = c, \quad v_{buffer} > d$

在猜测了$f(v_{buffer})$的函数形式之后，只要确定了参数a, b, c, d的值就可以将整个函数完全对应下来；不妨使用最小二乘法来求取这些参数(参数 d 是手动确定下来的)，得到：

- a =0. 7535
- b =-319. 0393
- c =2352. 5885
- d =3600

拟合结果如下图所示（红色为拟合的函数，蓝色为数据点）：



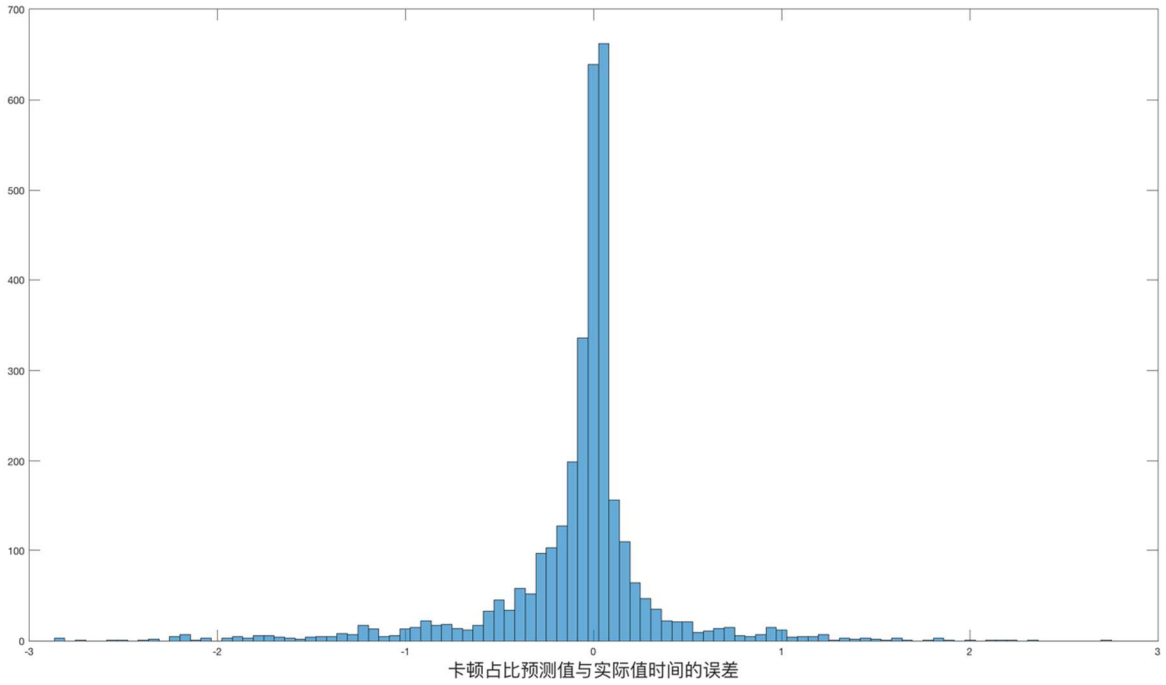
至此我们已经求出了 v 和 speed 之间的函数关系式，则有：

- $\frac{T_{play}\times Rate-D_{init}}{T_{tot}}=f(v_{buffer})$
- $T_{play}\times Rate=f(v_{buffer})T_{tot}+D_{init}$
- $T_{play}=\frac{f(v_{buffer})T_{tot}+D_{init}}{Rate}$

其中v_{buffer}为给定的平均下载速率，T_{tot}为播放阶段总时长（基本上为定值 30s），D_{init}为初始缓冲下载数据量（基本也为定值，为视频码率乘 4s），$Rate$为视频码率，也就是说这时候已经可以预测出t_2的数值了；因此$T_{stall}=T_{tot}-T_{play}$也同样可以求出，这样卡顿占比就是:

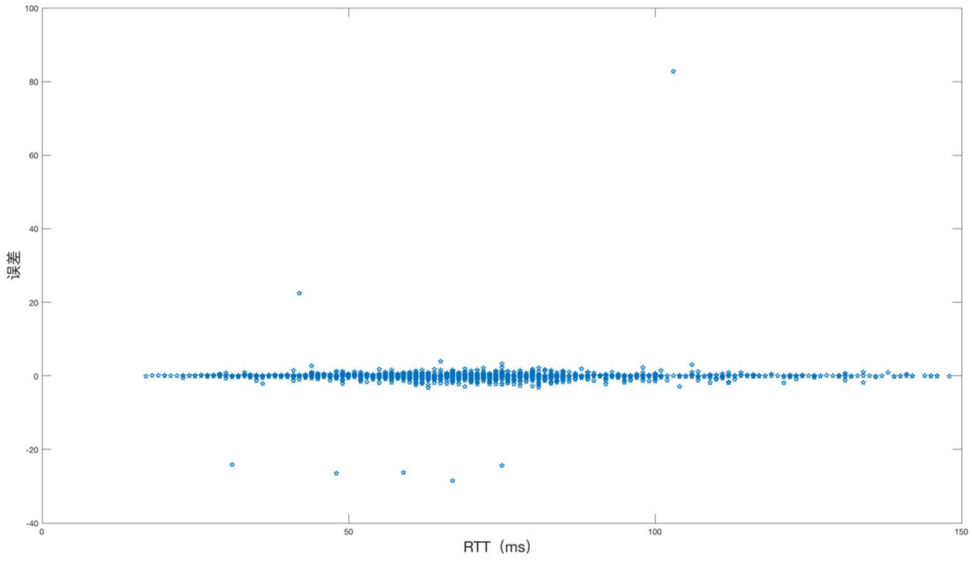
- $Ratio=\frac{T_{stall}}{T_{play}}$

至此已经可以确定下卡顿占比与v_{buffer}之间的函数关系了，接下来进行数据拟合：使用上述函数预测的卡顿占比和数据实际值之间的误差分布如下直方图(区间为[-3,3])所示：

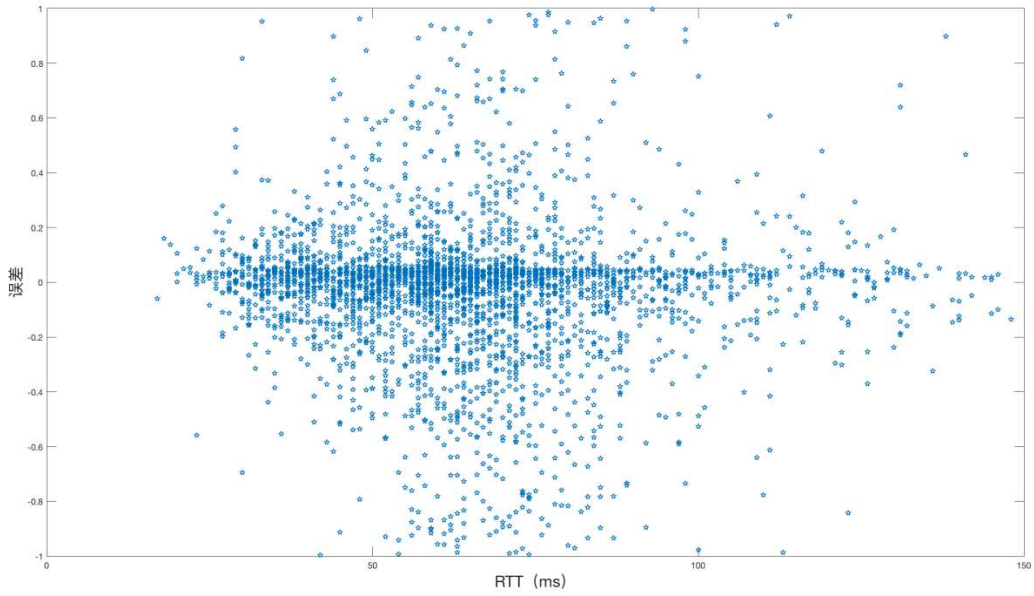


可以发现绝大多数的误差都落在了[-0.1,0.1]的区间内，经统计，总共有 1763 条数据落在预测值正负 0.1 的范围内，占了所有 3271 条卡顿数据的 54%，总共有 2523 条数据落在正负 0.3 的范围内，占比为 77%；并且误差的中位数为 0.0029，非常接近于 0，这也说明了预测的结果较好。

接下来查看 rtt 对误差分布的影响，如下图所示：



放大后得到：



发现误差均随机地分布在 x 轴附近，并且越靠近 x 轴出现的概率越大，这说明 rtt 对于卡顿占比的影响很小，因此在以上的拟合卡顿占比的函数中未出现 RTT 是合理的；

视频播放卡顿阶段：依据原理建模

一、数据处理模型近似

研究数据我们发现，在三千余条卡顿数据中有二十八条数据，其视频观看数据量（播放时长＊视频码率）小于初始缓冲下载数据量。我们假定看视频时即使网络有卡顿，也理应当时先把所有数据量看完以后再卡顿（也就是说不存在还有数据没播放却卡顿了的情况）。所以我们将这二十八条数据剔除。
总的来看，我们可以假定在这 30 秒内视频都在加载，视频最大加载量应该大于等于用户观看的数据量。

视频下载量＝初始缓冲下载数据量＋视频下载平均速度×总时长
视频观看量＝观看时间×视频码率

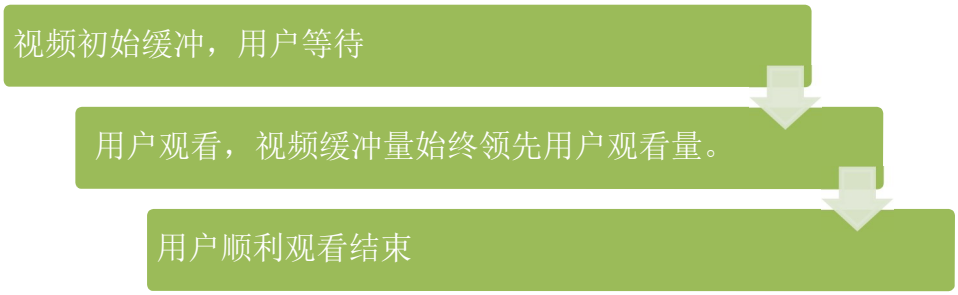
即：

$$D_{totbuffermax} = D_{init} + V_{buffer} \times T_{tot}$$
$$D_{totplay} = Rat \times T_{play}$$
$$D_{totbuffer} \geq D_{totplay}$$

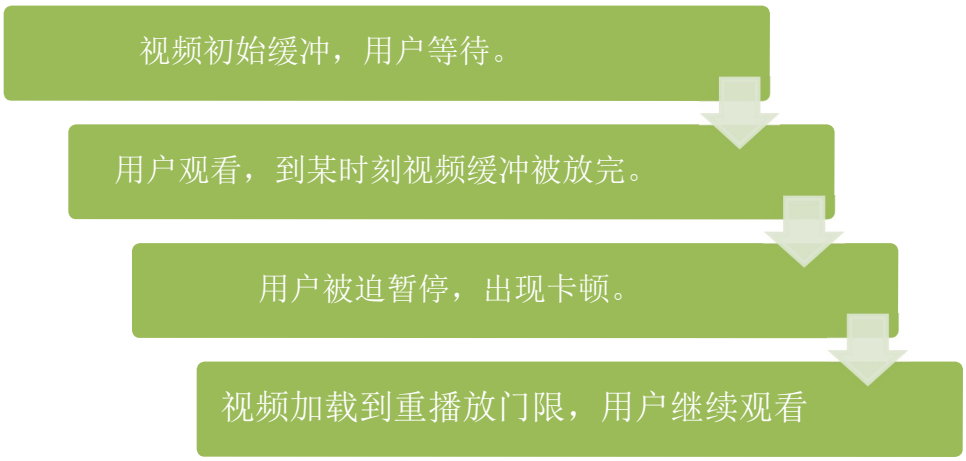
此标准下我们筛选掉了五十余条视频下载量小于视频观看量的部分。由此剩余卡顿数据 3271 条，可做卡顿模型有效的分析数据。

二、过程模型近似

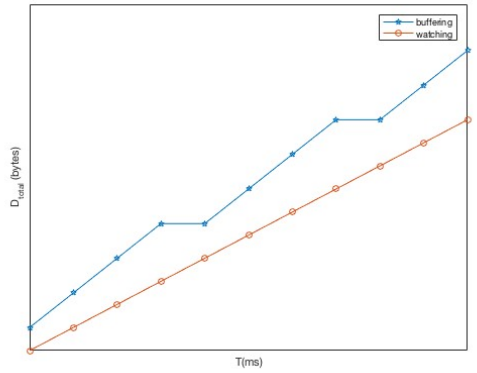
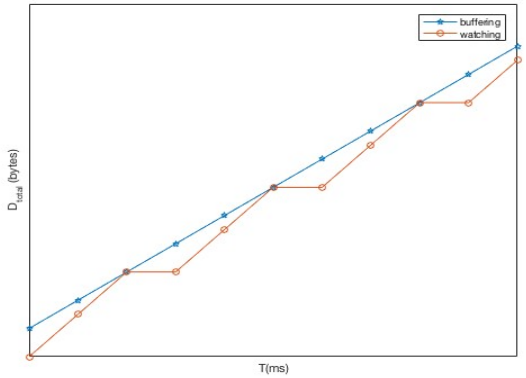
在观看视频部分，我们假设用户没有快进或者倒退行为。
那么一个正常的播放过程将会服从以下流程：



在用户出现卡顿的情形里模型会服从以下流程：



使用加载量～视频播放时间的关系来记录本流程，则两个过程可以用以下两张图表示。横坐标为视频播放时间，纵坐标为视频加载量。蓝色线为视频加载线，红色线为视频播放线。

正常的顺畅观看情形	卡顿情形
	

三、预估视频加载量和视频观看量

我们假设，看视频出现卡顿的数据里面，在整 30 秒内始终都有下载行为。并且，如果网速不好，那么视频加载量和视频观看量应该相差不大，基本上下载部分都可以被观看完。

因此我们得到了

视频下载量＝初始缓冲下载数据量＋视频下载平均速度×总时长

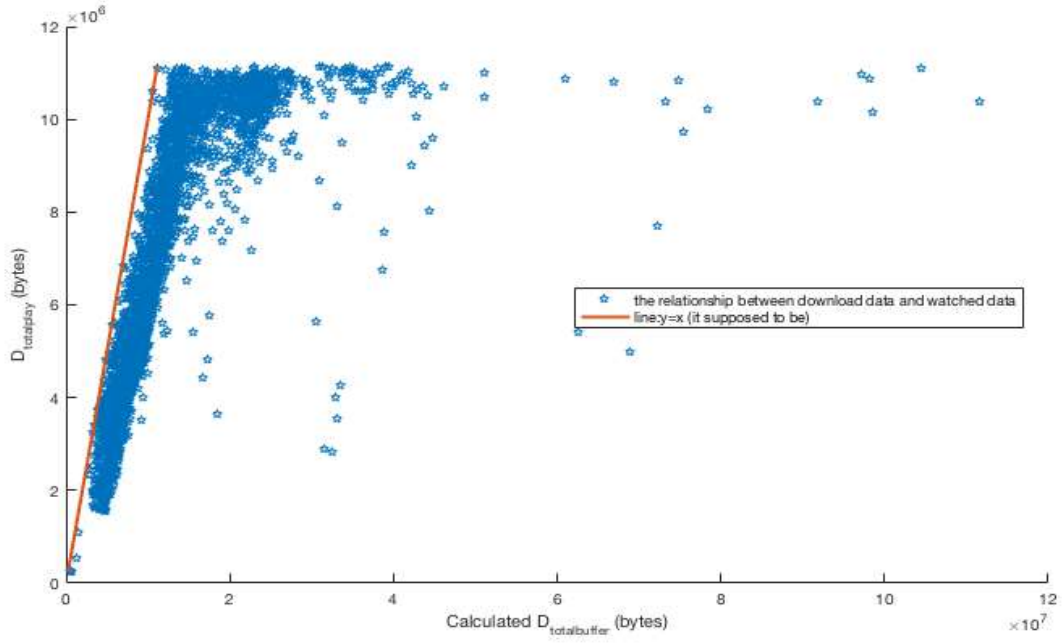
视频观看量＝观看时间×视频码率

即

$$Calculated\ D_{totbuffer} = D_{init} + V_{buffer} \times T_{tot}$$

$$D_{totplay} = Rat \times T_{play}$$

用以上公式处理数据点，我们得到下图。图横轴为预估视频加载量；纵轴为视频播放量。蓝色线为真实数据，红色线为一条$y = x$曲线。该图说明预估的视频加载量普遍大于视频观看量。但是在有些数据上，视频加载量高出观看量一个数量级。



所以我们猜想，可能出现以下情况：

1. **由于测试时长只有 30s 左右，有些视频加载了却没有播放出来。换句话说，没有卡顿视频出现的情况里，视频都被如愿加载完了。凡是有了卡顿，就有视频余量 $\mathbf{D_{rest}}$没有被考虑到。**

2. **假设错误。我们假定在一个卡顿模型中视频在全局播放时间都有被加载。但是，实际上，视频加载可能有停滞的时候。因此我们用全局播放时间来计算，视频下载量就被大大高估。针对这一点，我们对模型进行如下修正：**

视频下载量＝初始缓冲下载数据量＋视频下载平均速度×下载时长
下载时长<总时长

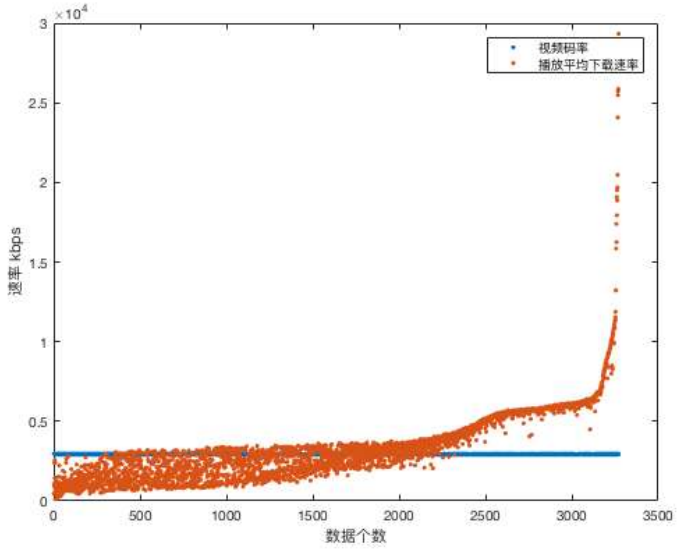
即：

$$Real\ D_{totbuffer} = D_{init} + V_{buffer} \times T_{buffer}$$
$$T_{buffer} \leq T_{tot}$$
$$Real\ D_{totbuffer} \leq Calculated\ D_{totbuffer}$$

视频加载量与观看量的偏差可能是以上因素共同作用的结果。

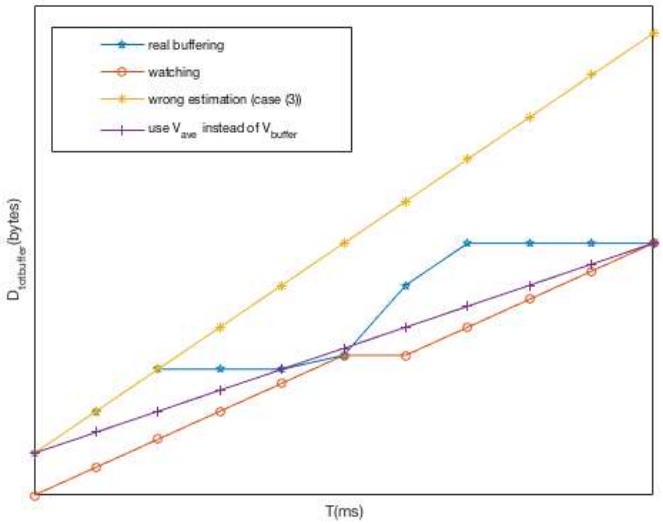
四、下载速率与视频码率

我们的假设错误可以帮助解释一些相对极端的数据。
我们通常认为，在下载速率比视频码率慢的时候会出现卡顿。而从数据集看，在 2781158 号用户的播放阶段平均下载速率是29932kbps，远超过了视频码率2934kbps。然而在这个过程中还是出现了卡顿。在 3217 条数据中，共有 1665 条数据的平均下载速率大于视频码率，见下图。



我们希望可以解释为什么平均下载速率比视频码率高很多的情况下，依然会出现卡顿情形。因此，我们预估；

1. **对于平均下载速率比较高的用户端来说，在视频播放阶段的加载也出现了停滞现象（$T_{bufferstall}$）。**

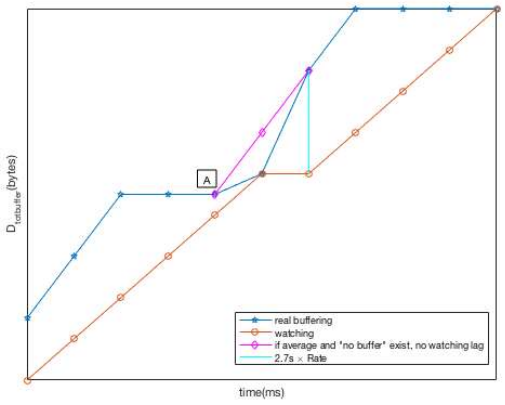


上图预测了一种在下载速率较高情况下的加载停滞。
我们先前的计算中过大地假设了加载量（见黄线）。只有在播放（橙）线要超过假设加载（黄）线时才出现卡顿，因此假设错误；

如果我们使用另一种简化模型，假设速率均一、时刻加载，总播放时长平均速率（V_{ave}）代替播放阶段下载平均速率（V_{buffer}），则会发现很大程度上视频依然不会出现卡顿（见紫色）。

2. **视频播放阶段的下载在停滞后的重启阶段是加速的。**

我们可以通过下图来解释。在视频下载速率始终为定值且大于视频码率的情况里，那么视频播放值逼近了加载值时（见图中 A 点），视频开始加载，然后视频加载应沿着大于视频码率的速率稳定上升（粉色线），并不会导致卡顿。因此，我们判断在重播放加载期间，视频下载速率是由小到大逐渐上升的，可能服从一个指数增长的模式。在某一时刻造成了卡顿（见图中深蓝色线和橙色线交汇处）。



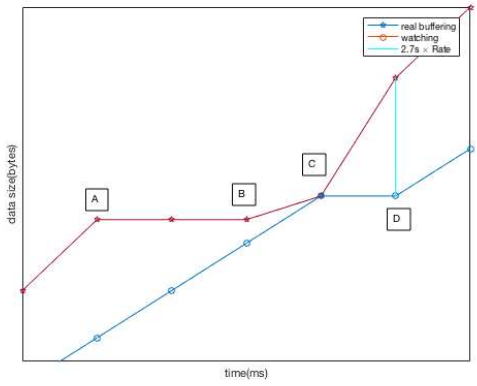
五、快下载速率的烦恼：加载停滞与不当的缓冲策略驱动

出现卡顿的情况大概可以按此描述：

用户观看阶段，视频匀速播放。但因为用户网络条件比较好，视频早早就被加载好。为了避免浪费资源，系统在 A 时刻发现加载量遥遥领先观看量D_{pause}然后停止加载。到达 B 时间点，系统发现如果不启动加载可能会出现卡顿，记录此时差值 $D_{restart}$.

因为启动加载初期 tcp 下传输非稳态。一开始加载量会比较小。当视频观看量等于观看量（c 点）时，视频出现卡顿。到 D 时刻后，视频已经加载$D_{re} = \frac{2.7s \times Rate(kbps)}{8}\ (bytes)$.

用户开始继续观看视频。



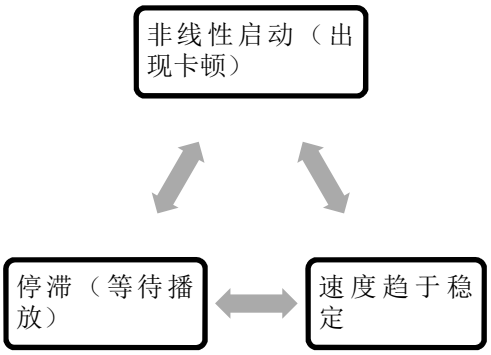
我们发现，在这个过程里，$D_{restart}$、T_{chase}为未知量，从 B 点到 D 点的下载函数也未知。从TCP其可能服从一个指数模型，但不便计算。

因此，我们假定在 B　点到 C 点服从线性关系，C 到 D 服从线性关系。则有：

$$V(dc) = \frac{D_{re}}{T'_{stall}}$$

$$\frac{V(bc) - Rate}{T_{chas}} = -D_{restart}$$

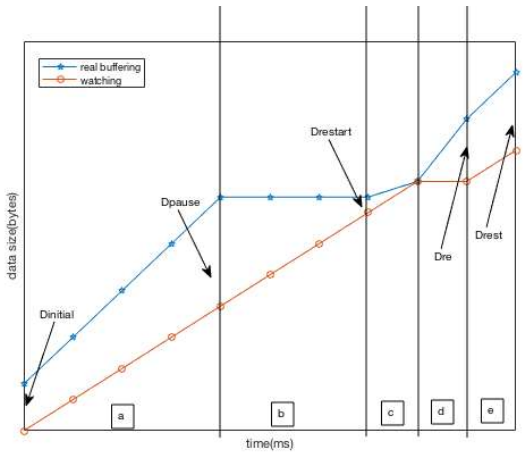
我们把上图叫做一个卡顿单元。对于一个平均加载速率较快的系统来说，卡顿是由加载停滞与不当的缓冲策略推动，每一个卡顿单元时间都大体相似。从 TCP 网络性质来看，我们估测整个播放阶段加载可以如下描述：



对于一个相对稳定的系统来说，以上参数应趋近常量。在视频加载的平均速率比较快的情况下，大体来说单次卡顿时长都差不多。所以

$$T'_{bufferstall} \times n_{buffer} = T_{bufferstall} = T_{tot} - T_{buffer}$$

在模型计算中，考虑最简单的情况：**只卡顿一次，且缓冲加载停滞只有一次。忽略 e 区剩余视频量。**



全程播放曲线：

$$D_{play} = Rate \times T_{play}$$

$$T_a + T_b + T_c = T_{play}; T_d = T_{stall}$$

全程加载曲线：

$$D_{tot} = D_{ini} + T_a \times V_{buffer} + T_b \times 0 + T_c \times V_c + T_d \times V_d$$

$$where\ T_b = T_{buffer};$$

$$Vd = \frac{D_{re}}{T'_{stall}}; T_d \times V_d = D_{re};$$

全程速度恒定：

$$T_c \times V_c + T_d \times V_d = V_{buffer} \times (T_c + T_d)$$

B 区：

$$\text{Rate} \times \text{T}_{\text{buffer}} = D_{pause} - D_{restart}$$

A 区：

$$\text{T}_\text{a} \times (V_{buffer} - Rate) + D_{ini} = D_{pause}$$

假定D_{pause}、$D_{restart}$为定值，则V_{buffer}为定值。

综上，我们得到：

$$T_d = (T_{play} \times Rate - D_{init} + D_{re})/V_{buffer} - T_{play} + \text{T}_{\text{buffer}}$$

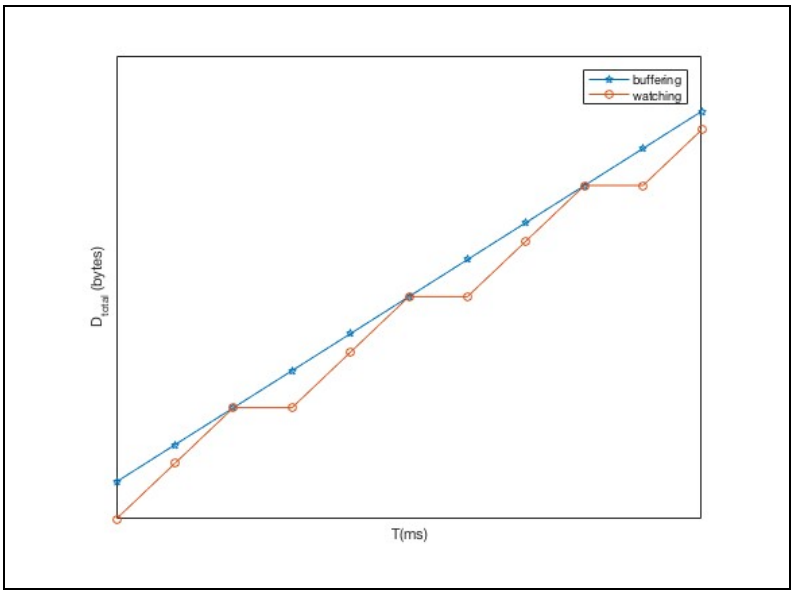
$$where\ T_{buffer} = \frac{D_{totplay} - D_{init}}{V_{buffer}} = \frac{(T_{play} \times Rate) - D_{init}}{V_{buffer}}$$

$$\text{thus}, T_d = (2T_{play} \times Rate - 2D_{init} + D_{re})/ V_{buffer} - T_{play}$$

$$Ratio = \frac{T_d}{T_{play}} = 2 \times \frac{Rate}{V_{buffer}} - \frac{2 \times D_{init} - D_{re}}{V_{buffer} \times T_{play}} - 1$$

因为这个模型估计时只考虑了一个单元、一次卡顿且忽略视频加载余项，故与实际数据差比较大。我们初步认为，这个问题的解决策略是针对$\text{T}_\text{a} \times (V_{buffer} - Rate) + D_{ini} = D_{pause}$，我们认为可以提升$D_{pause}$阀值。我们猜想在同一个视频应用中，对于单个用户而言，从何时停止缓冲到何时继续缓冲是一个定值。不过，较大的D_{pause}可以为恢复缓冲的初期加载提供非常好的时间空当。

六、低播放平均下载速率



考虑在低平均下载速率情况下的模型，在这种情况下，由于网速较低，因此客户端应当是无时不刻都在往缓冲区中下载数据的，而出现卡顿现象是因为视频码率高于下载速率，从而导致缓冲区数据量会随着视频的播放而不断减小，直到缓冲区数据量将为 0 时，视频播放将被暂停，从而出现卡顿现象，直到缓冲区的数据量一直增加到重播放门限$\mathrm{D_{re}}$为止，考虑到整个播放过程比较长，因此不妨将缓冲区数据量达到重播放门限$\mathrm{D_{re}}$到卡顿之后下一次达到重播放门限这段时间内的卡顿占比视为整个播放过程的卡顿占比（忽略了最开始），则可以得到$\mathrm{T_{stall}}$和T_{play}的关系如下：

$$\mathrm{D_{re}} = T_{stall} \times v_{buffer} = t_{play} \times (Rate - v_{buffer})$$

经整理可以得到：

$$\mathrm{ratio} = \frac{\mathrm{T_{stall}}}{\mathrm{T_{play}}} = \frac{Rate}{v_{buffer}} - 1$$

这样就推导出了一个卡顿占比ratio关于平均下载速率$\mathrm{v_{buffer}}$的函数。

重新考虑前文中从数据中总结出来的经验公式在平均下载速率较低的部$\frac{\mathrm{T_{play}} \times \mathrm{Rate} - \mathrm{D_{init}}}{\mathrm{T_{tot}}} = \mathrm{v_{ave}}$，对其进行变形可以得到：

$$\frac{T_{stall}}{T_{play}} = \frac{(Rate - v_{ave}) - \frac{D_{init}}{T_{tot}}}{v_{ave} - \frac{D_{init}}{T_{tot}}} \approx \frac{Rate - v_{ave}}{v_{ave}} = \frac{Rate}{v_{ave}} - 1$$

将上述公式和$\mathrm{ratio} = \frac{\mathrm{T_{stall}}}{\mathrm{T_{play}}} = \frac{Rate}{v_{buffer}} - 1$比较，发现不同之处仅有$\mathrm{v_{ave}}$替代了$\mathrm{v_{buffer}}$的位置，并且由前文总结经验公式的过程之中可以知道，$\frac{\mathrm{v_{ave}}}{\mathrm{v_{buffer}}} \approx 0.7535$，这个系数和前文专业名词部分的 TCP 吞吐量公式$\mathrm{throughput} = \frac{0.75\mathrm{W}}{\mathrm{RTT}}$中的系数 0.75 非常接近，因此猜想是 TCP 传输特性导致在视频播放阶段实际往缓冲区下载数据的速度只有测得下载速率的 0.75 倍。

由上文对于低平均下载速率根据原理的建模可知，在网速较差（平均下载速率小于视频码率）的情况下，卡顿现象难以避免。相关调查显示对于用户而言卡顿较较长时间的初始缓冲更难以容忍（如下图所示）[1]，因此建议在低下载速率的情况下，客户端可以通过增加初始缓冲时间来减少卡顿以提升用户体验。

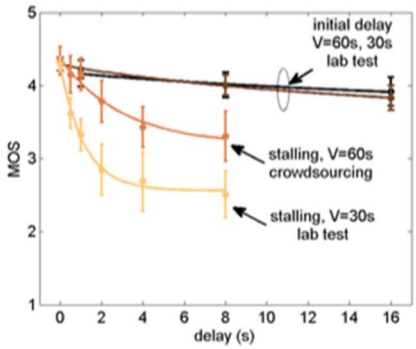


Fig. 4. One stalling vs. initial delay for YouTube QoE for videos of duration $V = 30\,\mathrm{s}$ and $V = 60\,\mathrm{s}$, respectively.

[1] Hoßfeld, T., Seufert, M., Hirth, M., Zinner, T., Tran-Gia, P., & Schatz, R. (2011, December). Quantification of YouTube QoE via crowdsourcing. In Multimedia (ISM), 2011 IEEE International Symposium on (pp. 494-499). IEEE.

七、对加载停滞概念的思考

在上一小节中我们给出了假设的三种情况，现在我们从另一个角度对上述模型假设进行印证。

假设视频下载时有停滞，则有*实际加载时长（未知）＋停滞时长（未知）＝总时长（约30000ms）*

$$T_{tot} = T_{buffer} + T_{bufferstall}$$

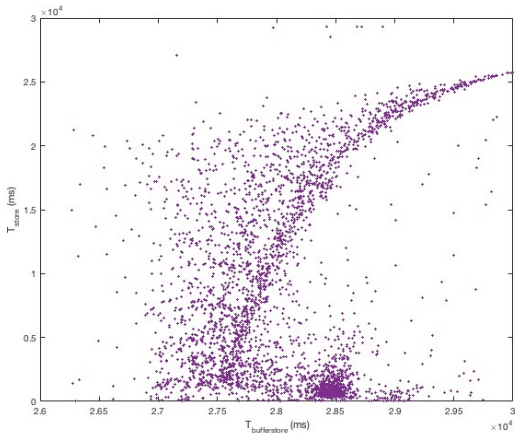
除此之外，*播放阶段平均下载速率∗实际加载时长（未知）＝视频总量－初始加载*

$$V_{buffer} \times T_{buffer} = D_{totbuffer} - D_{init}$$

如果不考虑视频余量D_{rest}的影响，则*总观看时间∗视频码率＝总观看量＝总加载量*

$$T_{play} \times Rate = \ _{totbuffer} = D_{totplay}$$

我们可以通过上述公式计算出视频实际加载时长和加载停滞的时间。加载停滞可以被认为是一种缓冲策略。在这里我们近似将下载量计为观看量，可能会导致加载停滞时间计算偏长。下图体现了加载停滞时间与播放卡顿时间之间的关系。



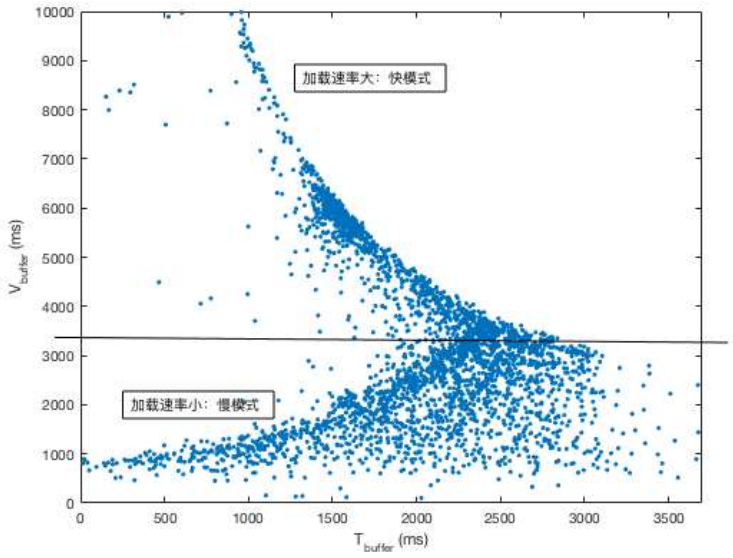
直观可解释为，平均加载速率很大的时候，加载停滞可能是因为都加载完了，因为前文所说的缓冲策略原因；平均加载速率太低时候，所以一直试图加载没有结果。
以上我们求出了加载停滞和卡顿时间之间的关系；从中可见，在 30000ms 的视频播放测试中，有很多数据点集中在$T_{bufferstore} = 28.5\ s$附近。可见这个时候其实网速比较快，这段加载停滞可被认为是一种缓冲策略，即提前加载了很多，所以暂停一段时间下载，如果用户看到了某一段继续加载。这个从另一侧面验证了我们的卡顿之谜。

八、下载时 v—t 关系详解：快模式与慢模式的博弈；网速随机变量的影响

如果我们纵览全局，在不考虑视频余量的情况下把实际缓冲时间和平均下载速率按照如下公式求出

$$T_{buffer} = \frac{D_{totplay} - D_{init}}{V_{buffer}} = \frac{(T_{play} \times Rate) - D_{init}}{V_{buffer}}$$

得到下图：



我们可以发现两条相对明确的趋势曲线和一个明确的分界点。我们把这个位置叫做鞍点，在此上方（“快模式”）即缓冲策略问题占主导地位；在此下方（“慢模式”）占据主导地位。这个位置大概在 3700kbps 左右。

1. 快模式：加载平均速度大于播放速度，但因为加载过程中有停滞，重新加载时速度又是前慢后快，出现卡顿；或者因为加载速度不够稳定，在某一小段时间太慢，导致卡顿。总体来看，加载平均速度越大，需要加载的时间越少。
2. 慢模式：加载平均速度小于播放速度，于是需要很多时间加载。这时视频卡顿比较严重。所以速度越大，可供观看的时间越久，可供加载的时间就越久。

九、播放阶段平均高下载速率（>3700 kbps）下的视频余量模型修正

在正常的视频播放案例里，视频顺利被放完。同样 30000ms 的测试时间下，卡顿模型的视频，或多或少，一定没放完。考虑视频余量的影响，则

$$T_{play} \times Rate = D_{tot} - D_{rest} = D_{totplay}$$

*总观看时间 * 视频码率 = 总量 − 视频余量 = 总观看量*

我们猜想在如此高的网速之下，视频余量全部被加载完了。所以

$$D_{tot} = D_{totbuffer}$$

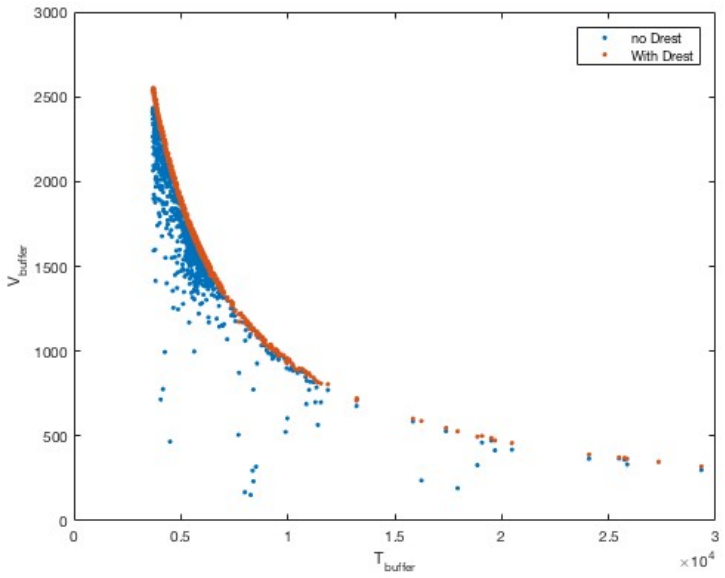
所以

$$T'_{buffer} = \frac{D_{tot} - D_{init}}{V_{buffer}} = \frac{(T_{tot} \times Rate) - D_{init}}{V_{buffer}} \approx \frac{30(s) \times 2934(kbps) - 1540000(byte)}{V_{buffer}}$$

$$As\ mean\left(D_{init}\ when\ V_{buffer} \leq 3700\ kbps\right) \approx 154000\ byte$$

考虑初始缓冲加载量还是比较小，视频总大小波动不大（总时长、码率波动不大），播放阶段视频实际下载时长和视频下载速度应该成反比关系。

我们筛选了播放阶段平均下载速率高于 3700kbps 的结果，正如猜想，对于视频余量的修正非常有意义。下图分别给出了视频余量的修正函数图。我们认为之前计算的 V_buffer 有可能偏小。



模型评价与总结

1. 对于预测初始缓冲时延部分，分别建立了基于数据和基于 TCP 原理的两个模型。其中两者均能够对所给数据进行较好的拟合。但是前者所使用的经验公式要比后者简单得多，而后者所使用的复杂预测时延的公式是基于 TCP 传输数据的原理严格地推导出来的，相比较前者具有着更加明显的实际意义以及拓展价值；
2. 对于预测卡顿时长占比部分，同样分别建立了基于数据和基于对视频卡顿原理的分析的两个模型。前者对于数据的拟合效果要远优于后者，但是由于前者是基于数据建模，得到的经验公式较难直观地看出其实际意义并且难以进行推广；而后者基于原理的建模部分，虽然为了突出原理过分简化了问题而使得拟合结果不佳，但是其较为深入地分析了在平均下载速率较高和较低两种情况下出现卡顿的原因，从而使得依据该模型得以提出若干旨在优化用户在线视频体验、减少视频卡顿时间的建议；
3. 在本次建模竞赛的两个建模任务中，均使用了基于数据和基于原理两种方法进行建模。就总体而言，基于数据的模型拟合数据效果更优，但是其模型难以解释，而基于原理的模型则更加直观，并且具有更多的可拓展性。

参考文献

[1]James F. Kurose, Keith W. Ross, 陈铭（译者）. 计算机网络：自顶向下方法（第六版）[M]. 北京：机械工业出版社，2014

[2]Mok, R. K., Chan, E. W., Luo, X., & Chang, R. K. (2011, August). Inferring the QoE of HTTP video streaming from user-viewing activities. In *Proceedings of the first ACM SIGCOMM workshop on Measurements up the stack* (pp. 31-36). ACM.

[3]Hoßfeld, T., Egger, S., Schatz, R., Fiedler, M., Masuch, K., &Lorentzen, C. (2012, July). Initial delay vs. interruptions: Between the devil and the deep blue sea. In Quality of Multimedia Experience (QoMEX), 2012 Fourth International Workshop on (pp. 1-6). IEEE.

[4]Hoßfeld, T., Seufert, M., Hirth, M., Zinner, T., Tran-Gia, P., & Schatz, R. (2011, December). Quantification of YouTube QoE via crowdsourcing. In Multimedia (ISM), 2011 IEEE International Symposium on (pp. 494-499). IEEE.

[5]Ameigeiras, P., Azcona-Rivas, A., Navarro-Ortiz, J., Ramos-Munoz, J. J., & Lopez-Soler, J. M. (2012). A simple model for predicting the number and duration of rebuffering events for YouTube flows. IEEE Communications Letters, 16(2), 278-280.

[6]郭士琪,王妍. 网络视频广告的前景分析[J]. 现代经济信息,2010,(12):175.

[7]程征,胡启林. 国外短视频新闻机构发展现状与启示[J]. 中国记者, 2015,(02):116-118.

[8]易发胜. 基于服务的网络端系统 QoS 的研究[D].电子科技大学,2008.