一种基于控制理论的动态自适应视频流方法

尹、阿布舍克·金达尔、维亚斯·塞卡尔、布鲁诺·西诺波利

卡耐基梅隆大学

{vsekar,brunos}@andrew.cmu.edu abhishekjindal93}@gmail.com银小旗522

# 摘要

用户感知的体验质量(QoE)在内部视频应用中至关重要，因为它会影响内容提供商和交付系统的收入。考虑到网络中几乎没有优化这些措施的支持，瓶颈可能出现在输送系统的任何地方。因此，客户端播放器中强大的比特率适配算法对于确保良好的用户体验至关重要。先前的研究显示了最先进的商业解决方案的主要局限性，并提出了一系列启发式修复方案。尽管出现了几个提议，但在以下方面仍然明显缺乏共识:(1)如何最好地设计客户端比特率适配逻辑(例如，使用率估计与缓冲区占用率)；(2)特定类别的方法在不同的操作条件下(如高通量可变性)表现如何；或者(3)他们实际上如何平衡不同的QoE目标(例如，启动延迟与重新缓冲)。为此，本文做出了三个关键的技术贡献。首先，为了给这个空间带来一些严谨性，我们开发了一个原则性的控制理论模型来推理一系列策略。第二，我们提出了一种新的模型预测控制算法，它可以最优地将吞吐量和缓冲区占用信息结合起来，从而优于传统方法。第三，我们在参考视频播放器中展示了一个实际的实现，使用真实的跟踪驱动仿真来验证我们的方法。

# CCS概念

* **信息系统→多媒体流；网络**

***→网络协议设计；应用层协议；***

# 关键词

互联网视频；比特率适配；DASH模型预测控制

允许免费制作本作品的全部或部分数字或硬拷贝供个人或课堂使用，前提是拷贝的制作或分发不是为了盈利或商业利益，并且拷贝带有本通知和第一页的完整引文。必须尊重除ACM之外的其他人拥有的本作品组件的版权。用信用抽象是被认可的。以其他方式复制或重新发布，在服务器上发布或重新发布到列表，需要事先获得特定许可和/或费用。向请求权限[permissions@acm.org。](mailto:permissions@acm.org)

*2015年8月17日至21日在英国伦敦举行的第15届全球通信大会*

◯c 2015 ACM。ISBN 978-1-4503-3542-3/15/08。。。$15.00

DOI:<http://dx.doi.org/10.1145/2785956.2787486>

# 介绍

最近的许多研究都强调了用户感知体验质量(QoE)在互联网视频应用中的关键作用，因为它最终会影响内容提供商的收入流[24，35]。具体而言，重新缓冲的持续时间(即，播放器的播放缓冲区没有要呈现的内容)、启动延迟(即，用户点击与开始呈现的时间之间的延迟)、平均回放比特率和所传送的比特率的可变性等指标已经成为关键因素。

鉴于复杂的互联网视频交付生态系统和各种瓶颈的存在，客户端视频播放器中的比特率适配逻辑对于优化用户体验变得至关重要[16]。在今天占主导地位的基于HTTP的传输模型[44]中，视频通常以不同的比特率级别进行分块和编码。自适应视频播放器的目标是为未来的块选择比特率级别，以提供尽可能高的QoE例如最大化比特率，同时最小化再缓冲的可能性并避免过多的比特率切换。

最近的许多努力指出了设计这种适应逻辑的关键挑战(例如[46，17，32，34])，并且出现了一些尝试和解决这些挑战的建议(例如[34，17，33])。然而，尽管nu- merous算法大量出现，但这些解决方案在几个方面似乎缺乏明确性和一致性；例如，一些人主张更好的吞吐量估计[47]，而另一些人建议改进块调度[34]。一些再搜索者甚至反对基于速率的方法，这种方法依赖于对先前块下载的吞吐量估计，并支持基于缓冲区占用的算法，这种算法纯粹基于缓冲区占用做出决策[33]。

为了理解在不同操作状态(例如，低对高吞吐量可变性)下不同类别算法(例如，基于速率对基于缓冲器)之间的基本权衡，我们首先将视频比特率调整公式化为随机最优控制问题。我们大致定义了视频适配问题中涉及的关键动态变量和一个具体目标。这个框架允许我们为这个问题勾勒出更广阔的控制算法设计空间。我们发现了现有方法中的一个主要缺点，这种方法完全依赖纯粹的基于速率或缓冲区的策略，这可能会错过结合两种信号的策略。

基于对控制理论公式的理解，我们认为模型预测控制[22]是

一类合适的算法，可以优化组合基于速率和基于缓冲器的反馈信号。在高层次上，MPC试图预测移动前瞻地平线上的关键环境变量，并基于预测解决精确的优化问题。多点控制是众多现实控制问题中的首选技术[22]。除了直观的公式之外，它还可以直观地处理复杂的控制目标和约束，并具有一组众所周知的调整参数，如预测范围。此外，与先进的控制方法相比，多点控制具有其他质量上的优势，因为它的开发时间要短得多，而且更容易掌握，因为改变模型参数不需要完全重新设计。

在我们的上下文中，MPC方法需要预测接下来几个块的预期吞吐量，并使用它来为QoE最大化做出最佳比特率决策。实际上，我们的仿真结果证实，如果我们能够运行一个最优的最大功率控制算法，并且预测误差较低，那么最大功率控制方案可以优于传统的基于速率和基于缓冲区的策略。

然而，在实践中，运行基于MPC的算法是有挑战性的，因为它需要在每个时间步骤解决一个非平凡的离散优化问题。即使忽略计算开销，也存在实际困难，因为我们可能需要将此求解器逻辑与每个视频播放器捆绑在一起，或者要求用户下载并安装额外的软件。为了应对这些挑战，我们开发了一种简单而高效的FastMPC机制。从概念上讲，FastMPC本质上遵循一种表枚举方法，在这种方法中，我们描述问题状态空间，以最佳方式离线求解特定实例，并存储最佳控制决策以备将来在线使用。然而，如果天真地实现，这个表的大小会导致视频播放器的大量内存开销和启动延迟(例如，需要加载额外的JavaScript)。幸运的是，我们展示了使用简单的值绑定和压缩策略，我们可以通过可管理的表大小实现接近最优的性能。

我们已经在一个名为dash.js [1]的开源动态自适应流媒体播放器中原型化了我们的FastMPC比特率自适应算法。我们选择的平台是实用的——它是基于HTML5规范的MPEG-DASH标准的参考开源实现，并得到了领先行业参与者的积极支持[7]。我们展示了我们的实现给基线dash.js播放器增加了可忽略的开销。我们还在演示页面中展示了基于FastMPC的播放器[14]。

我们使用实测[9，10]和合成吞吐量可变性轨迹上的真实仿真实验来评估我们的算法和原型实现。我们还用基于仿真的灵敏度分析实验来验证这些结果，以分析关键操作参数对不同类别算法性能的影响。我们的主要发现是:

1. 我们提出的MPC方法在宽带(FCC)数据集中始终比最先进的自适应算法强15%，在蜂窝(HSDPA)数据集中强10%

就生活质量中位数而言。与工业级参考玩家dash.js相比，它还实现了显著的改进(QoE中位数提高了60%)；

1. 与其他算法相比，我们快速且低开销的实现FastMPC需要相似的CPU使用量和仅60 kB的额外内存使用量。

贡献和路线图:总之，本文做出了以下主要贡献:1

比特率适配问题的形式控制理论模型的发展(第3节)；

*•*

设计一种包含现有基于速率和缓冲的策略的边际成本控制方法(第4节)；

*•*

一种实用而快速的基于表枚举的快速多点控制算法，它接近于精确多点控制方法的性能(第5节)；

*•*

基于开源参考视频播放器dash.js的低开销实现(第6节)；

*•*

在广泛的操作参数和真实轨迹范围内对不同类别的算法进行系统评估(第7节)

*•*

在下一节中，我们首先讨论DASH的背景和相关工作。

# 背景和相关工作

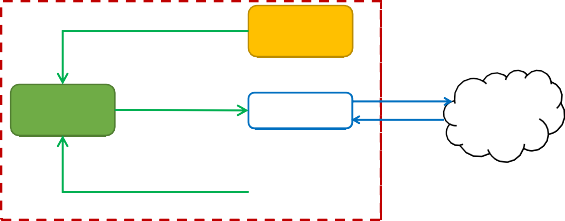
我们从基于HTTP的自适应视频流工作原理的高级概述开始，然后描述当今最先进解决方案的主要缺点。

微软的Smooth- Streaming [13]、苹果的HLS [5]和Adobe的HDS [2]等互联网视频技术都依赖于基于HTTP的自适应流媒体。这类协议正在通过HTTP或DASH的动态自适应流的保护下进行标准化[16]。在DASH系统中，每个视频由多个片段或“块”(对应于几秒钟的播放时间)组成，每个块以多个离散的比特率编码。来自不同比特率流的块被对齐，使得视频播放器可以在必要时在块边界切换到不同的比特率。与实时消息协议(RTMP)等流媒体协议相比，这种方法有几个实用优势。HTTP的使用使提供者能够无缝地绕过中间盒子。此外，它可以使用现有的商品CDN服务器，而不需要定制修改。最后，通过使服务器无状态化，可以使用多个服务器和cdn实现更好的应用层弹性[41，40]。

图1显示了自适应视频播放器的抽象模型。播放器在其决策逻辑中使用一些输入(例如，缓冲器占用率或网络吞吐量的估计)来选择要下载的下一个块的比特率级别。在做出这一决定时，玩家必须考虑许多潜在的相互冲突的QoE因素:

(1)最小化再缓冲事件，其中回放缓冲器

1该论文的早期研讨会版本证明了基于MPC的方法的合理性[50]。然而，它没有提供一个具体的算法，一个实际的实现，以及使用真实吞吐量跟踪的评估。



吞吐量预测

吞吐量视频播放器

预言者

得到

比特率控制器

比特率

超文本传送协议

互联网

矮胖的人或物

缓冲区占用

QoE

**最终用户**

缓冲器

### 图DASH玩家的抽象模型

为空，无法呈现视频；(2)在吞吐量限制内传送尽可能高的回放比特率；(3)最小化启动延迟，使得用户在等待视频加载时不会退出；以及(4)通过避免频繁或大的比特率跳跃来保持回放尽可能平滑[24，35]。

为了理解为什么这些目标是冲突的，让我们考虑两个极端的解决方案。最小化再缓冲和启动延迟的一个简单的解决方案是总是选择最低比特率，但是它与提供高比特率的目标相冲突。相反，选择最高可用比特率可能会导致许多重新缓冲事件。类似地，如果同时最小化再缓冲和最大化平均比特率的最佳选择是切换每个块的比特率，则保持平滑回放的目标也可能冲突。

本文的重点是客户端适配解决方案。其他补充工作包括使用服务器端比特率切换(例如[37，18])、避免突发的TCP更改(例如[27])，以及网络内吞吐量管理和缓存(例如[31，45，42])。我们关注客户端问题有两个关键原因。首先，与需要网络内支持(例如[31，45，42])、服务器端软件更改(例如[37，18])或修改的解决方案相比，客户端解决方案提供了最快速部署的替代方案

对低层传输协议的批准(例如[27，28，39，29，36])。其次，客户端通常处于快速检测性能问题和响应动态的最佳位置。也就是说，我们相信我们开发的正式基础和算法可以同等地应用于这些其他部署场景。

许多测量研究表明，最先进的视频播放器在这些QoE测量(例如[46，34，32])方面表现不佳。这些研究表明，大多数问题不是特定玩家的作品，而是在所有最先进的玩家中表现出来，如流畅流[13]、网飞[11]、Adobe OSMF [3]和Akamai HD [4]。为简洁起见，我们不在此重复这些结果，但请感兴趣的读者参考以前的工作(例如[46，34，32])。

为了缓解这些问题，最近在研究文献中有几个提议(例如[47，34，18，33，38])。在高层次上，这些解决方案可以大致分为两类:(1)基于速率的算法和(2)基于缓冲区的算法。采用基于速率的方法的视频播放器基本上根据es-选择最高可能的比特率

估计的可用吞吐量。然而，如先前的工作所示，基于HTTP的吞吐量估计存在明显的偏差[32]，这导致了传统的基于速率的方法的问题。一些解决方案试图通过平滑吞吐量估计值[47]或选择更好的调度策略[34]来克服这些偏差。另一方面，最近的工作支持基于缓冲区的算法[33]。这类算法不是使用吞吐量估计，而是使用缓冲区占用率作为反馈信号，并设计机制将缓冲区占用率保持在所需水平，基本上丢弃任何可用的吞吐量信息。

尽管对这个主题有广泛的兴趣，但今天严重缺乏的是对比特率自适应算法的原则性理解。上述每个解决方案都提供了在特定(和隐含)环境假设下工作的点启发。虽然孤立地看，每种方法的表现都优于商业玩家，但很少有人系统地比较不同类别的算法是如何相互叠加的，或者这些技术组件中的哪一个是关键的，或者这些算法在不同的操作状态(例如，吞吐量稳定性、缓冲区大小、比特率级别数)下有多稳健。此外，这些算法中的许多甚至没有正式说明它们寻求优化的目标，这使得进行有意义的比较变得更加困难。

我们在这项工作中的首要目标是给这个空间带来一些清晰度。我们不是设计另一个单点解决方案，而是从通过控制理论开发第一性原理方法开始，以开发一个通用框架来推理算法类。在下一节中，我们使用这个控制理论“镜头”来正式定义视频比特率自适应算法试图解决的随机优化。

# 控制理论模型

在本节中，我们开发了一个HTTP视频流过程的数学模型，并正式定义了策略适配问题。该模型为我们提供了一个比较和评估现有算法的框架，并作为潜在改进的基础。

# 视频流模型

我们将视频建模为一组连续的视频片段或块，= 1，2，，K，每个片段或块包含L秒的视频。每个块以不同的比特率编码。假设是所有可用比特率级别的集合。视频播放器可以选择以比特率Rk下载块k。设dk(Rk)是以比特率Rk编码的块k的大小。在恒定比特率(CBR)的情况下，dk(Rk) = L Rk，而在可变比特率(VBR)的情况下，dk Rk关系在块之间可能不同。

*∼*

**

*∈ *

**

** ** *  * **

选择的比特率越高，用户感受到的视频质量就越高。假设q( ) : R+是一个非递减函数，它将选定的比特率Rk映射到用户q(Rk)感知的视频质量。请注意，q()可能取决于

**

** * →*

视频播放设备以及视频内容。例如，在高清电视上，3兆位/秒和1兆位/秒可能会导致用户体验的显著差异

𝐵𝑘

下载并等待

𝑑𝑘(𝑅𝑘) + Δ𝑡

𝐶𝑘

𝑘

缓冲区占用

𝐵𝑘+1

0

再缓冲

时间

等待时间tk的确定，也称为块调度问题，在提高多玩家视频流的公平性方面是一个同样有趣和重要的问题[34]。然而，在本文中，我们假设一旦下载了块k，播放器就立即开始下载块k + 1。一个例外是当缓冲区满了，玩家等待缓冲区减少到一个允许附加块k的水平。形式上，

𝑡𝑘

开始块k

𝑡𝑘+1

开始区块k+1

⇼t =

*dk(Rk)*

*b*

+Lb！

(4)

### 图2:缓冲器动态图

+

3Mbps和1Mbps的视频质量在阿莫可能相似-

*k k Ck*

*最大*

+

胆汁装置；此外，提高“动态”块的比特率将导致比提高“静态”块更多的QoE增益。视频片段被下载到回放缓冲区，该缓冲区包含已下载但尚未观看的视频。设B(t) [0，Bmax]为t时刻的缓存占用率，即缓存中剩余视频的播放时间。缓冲区大小Bmax取决于服务提供商的策略，以及播放器的存储限制。典型的玩家缓冲区

*∈*

可能包含几十秒的视频片段。

图2有助于说明视频播放器的概念操作。在时间tk，视频播放器开始下载块k。这个块的下载时间将是dk(Rk)/Ck；即，它取决于所选块的大小

比特率为，平均下载速度为

从；外；离；除；超过；以前的

*k*

*k*

# 生命质量最大化问题

比特率适配的最终目标是提高用户的生活质量，以实现更高的长期用户参与度[24]。我们的目标是提供一个灵活的生活质量模型，而不是固定的生活质量概念，因为这是一个活跃的研究领域[19]。虽然用户的具体QoE功能可能有所不同，但我们可以将视频QoE的关键元素列举如下:

1. *平均视频质量:所有块的平均每块质量:1k q(Rk)；*

*K*

*k=1*

Σ

1. *平均质量变化:这跟踪幅度*

从一个块到另一个块的质量变化:

1σk-1 q(rk+1)q(rk)；

*k1*

*k=1*

3.再缓冲:对于每个块k，如果向下

在此下载过程中遇到。一旦块k被

完全下载后，视频播放器等待∏tk

加载时间dk(Rk)/Ck高于块下载开始时的播放缓冲级别(即Bk)。因此

在时间tk + 1开始下载下一个块k+1。我们作为-

假设等待时间很短，不会导致

总再缓冲时间3是σK

dk(Rk)Bk。

重新缓冲事件。如果我们用Ct表示时间t的网络吞吐量，那么我们有:

*k=1*

*Ck*

+

*t = t+dk(Rk)+t(1)*

*C*

*k+1 k k*

*k*

4.启动延迟Ts，假设Ts Bmax。

由于用户可能对四个分量中哪一个更重要有不同的偏好，我们通过上述分量的加权和来定义视频片段1到K的QoE:

*Ck =*

1∫tk+1∫tk

*Ct dt。(2)*

*K k1*

*tk+1 tk≈tk tk*

*qoek =σq(rk)λσq(rk+1)q(rk)*

缓冲区占用率随着块的下载和视频的播放而变化。具体来说，

一

*K*

块k之后，缓冲区占用率增加1秒

下载并随着用户观看视频而减少。2

*k=1*

Σ

*k=1*

*—*

*k=1*

发展研究

*k*

*Ck*

*k*

*— Bk*

*— sTs* (5)

+

让Bk = B(tk)表示当播放器开始下载块k时的缓冲区占用率。缓冲区动态可以表示为:

这里，λ、s分别是对应于视频质量变化、再缓冲时间和启动延迟的非负加权参数。相对较小的λ表示用户不是特别关心视频质量

*Bk+1*

= Bk

*dk(Rk) Ck*

+L≈tk

+

−

+

！

(3)

可变性；λ越大，为实现更平滑的视频质量变化所做的努力就越多。相对于其他参数，较大的值表示用户被深深地欺骗了

这里，符号(x)+ = max x，0确保该项永远不会为负。请注意，如果Bk < dk(Rk)/Ck，则当视频播放器仍在下载块k时，缓冲区会变为空，从而导致重新缓冲事件，如图-

** **

图2。

2“启动”阶段会因玩家而略有不同

关心再融资。在用户喜欢低启动延迟的情况下，我们采用大s。

总的来说，这种对生活质量的定义是非常普遍的，因为它允许我们根据不同的分配因素来模拟不同的用户偏好。

3另外，还可以考虑再缓冲的数量-

等待一定量的缓冲积累后再排出

缓冲区。

*k=1*

*Ck*

*k*

*用σK表示的ing事件*

1 dk(Rk) > B。

比特率

最大

*R1，RK，Ts*

*QoEK*

一

(6)

*s.t. tk+1 = tk +*

*dk(Rk)*

*Ck*

+∑tk，

(7)

*Ck =*

一

*B = B*

*tk+1 tk≈tk*

∫

*t*

*k+1 k*

t

*Ct dt，*

(8)

*坦克*

*发展研究*

*k+1*

*k k*

*k Ck*

+L≈tk

！

*, (9)*

+

+

*B1 = Ts，Bk∏0，Bmax]*

*rk∈∀k= 1，k*

(10)

(11)

所有算法的设计空间

基于缓冲区



**A3？**

基于费率

**一流的**

吞吐量预测

**主动脉第二声**

缓冲区占用

### 图3:受缓冲区和吞吐量动态影响的QoE最大化公式。

一

QoE最大化问题:我们现在准备形成QoE最大化的比特率适配问题，如图3所示，表示为QOE

一

把《追踪c 》, t∞，t,《追踪]》

### 图4:视频适配问题算法的设计空间:大多数当前方法选择的策略只是一个变量的函数；例如，A1是基于速率的(RB)，而A2是基于缓冲区的(BB)。

不同的算法在各种可变性条件下使用最先进的预测器执行。

现在，不同的自适应算法本质上采用不同的函数f()。具体来说，文献中出现了两大类算法:基于速率的算法和基于缓冲区的算法。RB策略本质上选择双向

**

*t 1 K+1*

作为输入，优化程序

仅基于吞吐量预测，即，

提供以下输出:1)比特率决定，R1，

*RK，和2)启动时间Ts。*

请注意，问题QOE \_MAX K被公式化为-苏明在此优化时视频回放尚未开始，因此启动延迟Ts是一个决策变量。

一

然而，当下一个要下载的块是k0并且当前缓冲区占用率是Bk0时，这种QoE最大化也可以在时间tk0的视频回放期间发生。在这种情况下，

我们可以去掉变量Ts，表示相应的

*稳态问题为QOE最大稳态。*

*k0*

# 算法的分类

在本节中，我们描述了问题QOE最大值，并描述了这个框架内现有的比特率自适应算法，以了解它们之间的关系。

一

图3中的问题是一个有限时域随机最优控制问题。随机性的来源在于可用吞吐量Ct。在时间tk，当玩家选择比特率Rk时，只有过去的吞吐量Ct，t tk是可用的，而未来的值Ct，t > tk是未知的。

** **

** *≤* **

然而，吞吐量预测器可用于获得定义为c ' t，t > tk的预测。基于这样的预测和缓冲器占用率信息(取而代之的是精确已知的)，比特率控制器选择下一个块k的比特率:

** **

*R = f B，Cˆ，t > t ，R，i < k。* (12)

*k*

*k*

*t*

*k*

*我*

*Rk = f Cˆt，t > tk，Ri，i < k。* (13)

例如，典型的RB策略是选择低于预测吞吐量的最大可能比特率。

另一方面，BB策略提倡仅基于缓冲区占用率进行决策，即:

*Rk = f (Bk，Ri，i < k)，* (14)

同时将吞吐量变化视为未建模的干扰。例如，黄等人举例说明了设计BB算法的一个路线图[33]。

然而，请注意，这两类算法都是可能有用的信息，如图4所示。因此，两者原则上都是次优的。理想情况下，我们希望同时使用缓冲区占用率和吞吐量预测，从而考虑比特率自适应策略的更大设计空间，如图4中的等式(12)和算法A3所示。

# 最优比特率自适应的模型预测控制方法

在本节中，我们介绍了一种用于比特率适配的模型预测控制(MPC)方法，并描述了一种基于MPC的具体工作流，它可以将吞吐量预测和缓冲区占用率最佳地结合起来。我们也去-

开发一种稳健的MPC方法，可以更好地处理高度可变网络条件下的吞吐量预测错误

有效吞吐量预测器的设计本身就是一个有趣的研究方向。在这篇文章中，我们只关注比特率自适应算法，并假设预测给我们，并根据它们的预期预测误差来表征。也就是说，我们着重于f()的设计和预测误差对比较控制算法性能的影响。在接下来的章节中，我们将系统地评估dif-

**

条件。

# 为什么是MPC？

首先，我们提供了在我们的环境中选择MPC背后的直觉。注意，在所有可能的控制算法中，我们不能说最大功率控制是必要的或最优的选择。我们的目标仅仅是论证MPC是比特率自适应问题的自然适应。

正如我们之前看到的，比特率自适应本质上是一个随机最优控制问题。在这方面，有两种候选的众所周知的控制算法:(1)比例积分微分控制[25]

算法1使用MPC 1的视频适配工作流程:初始化

**2:对于k = 1到K do**

3:如果玩家处于启动阶段，那么

和(2)基于马尔可夫决策过程(MDP)的控制[21]。4:

*k*

*s*

*多媒体个人电脑*

*k1*

*k*

[传统知识，传统知识+N ]

ˆ

[传统知识，传统知识+N ]

*C*

=吞吐量

[t1，tk])

虽然与MPC相比，PID在计算上更简单，

它只能稳定系统，不能解释-

5: [R，T ] = fst R *，B，c*

这将优化我们的QoE目标。此外，PID控制被设计成在连续的时间和状态空间中工作

6:在Ts秒后开始回放7:否则如果回放已经开始

在像我们这样高度分散的系统中使用它可能会产生8:

9: *R = f 稀有 ，B，Ck 多媒体个人电脑 k1* *k* ]k k+N

ˆ

[传统知识，传统知识+N ]

*C*

=吞吐量

[t1，tk])

性能下降或不稳定[25]。或者，对于MDP，我们可以考虑将吞吐量和缓冲区状态转换公式化为马尔可夫过程，并使用标准算法(如值迭代或策略迭代)找到最优控制策略[21]。然而，这有一个强有力的假设，即吞吐量动态遵循马尔可夫过程，不清楚这是否在实践中成立。我们将MDP的潜在用途和吞吐量动态分析视为未来的工作(见第8节)。

MPC的情况:理想情况下，给定视频[t1，tK+1]整个范围内未来吞吐量的完美知识，通过解决整个视频的优化问题，可以一次性计算出最佳比特率R1、RK和启动延迟Ts。实际上，如此完美

一

*  *

ˆ

[t，t

10:结束if

11:下载比特率为Rk的块k，等待找到

**12:结束于**

论文不是设计一个预测机制，而是依靠现有的方法。自然，提高这种预测的准确性将提高通过最大功率控制获得的增益。也就是说，正如我们下面讨论的那样，MPC可以扩展为对错误具有鲁棒性。

1. *优化:这是MPC算法的核心:给定*

当前缓冲器占用率Bk，先前比特率Rk 1

信息不可用，因此很难找到这样的信息

使用离线优化的最佳解决方案。

虽然完美的信息可能不可用于en-

和吞吐量预测

*r。在稳定状态下，R = f*

[tk，t ]k+N

*k*

*k*

*多媒体个人电脑*

，找到最佳比特率

稀有 *，B，c* ,

*k1*

*k*

[传统知识，传统知识+N ]

轮胎的未来，有可能合理准确的吞吐量

通过求解QOE最大稳态k+n1来实现。

相反，预测可以针对

在启动阶段

*k*

*T as*

，它还优化了启动时间

*多媒体个人电脑*

[t，t

*k k+N ]*

未来[tk，tk+N ]。这里的直觉是，网络条件在短时间尺度上相当稳定，并且通常在短时间范围(几十秒)内不会发生剧烈变化

onds) [51]。基于这一认识，我们可以使用该水平的预测运行QoE优化，应用第一比特率Rk，并将水平向前移动到[tk+1，tk+N+1]。这种方案被称为模型预测控制(MPC)或滚动时域控制[22]。MPC算法应用广泛

用于从工业控制到导航的不同领域。最大功率控制的一般好处在于，它可以利用预测在线优化受约束的动态系统中的复杂控制目标。

# 基本最大功率控制算法

算法1展示了用于比特率适配的MPC工作流程的高级概述。在我们的上下文中，该算法本质上是通过向前看N步(即移动地平线)来选择比特率Rk，并解决特定的QoE最大化问题(这取决于玩家是否在

稳定或启动阶段),具有吞吐量预测Cˆt，t∞。

[Rk，Ts]= FST Rk 1，Bk，C\ ，通过求解QOE MAX k+n1实现。如果我们忽略关于计算开销的实际细节，我们可以简单地使用

现成的解决方案，如CPLEX，来解决这些具体的优化问题。正如我们将在第5节中看到的，我们不需要在实践中明确解决视频播放器中的优化问题。

*k*

1. *应用:开始用Rk下载k块，并向前移动地平线。如果播放器处于启动阶段，请等待Ts后再开始播放。*

与我们讨论的基于缓冲区、基于速率的工作流相比，这种工作流有几个质量上的优势。首先，该MPC算法以有原则的方式使用吞吐量预测和缓冲区信息。第二，与纯RB方法相比，MPC平滑了每一步的预测误差，对预测误差更加稳健。具体来说，通过在移动范围内优化几个块，一个特定块的大预测误差

直接优化正式定义的QoE目标，同时

[t，t

*k*

*钾+氮*

[传统知识，传统知识+N ]

*k*

]

。第一比特率

应用者

会对性能产生较低的影响。第三，货币政策委员会

使用反馈信息，优化过程在每个步骤k迭代。

在迭代k，玩家保持从块k到k+N ^ 1的移动地平线，并执行以下三个关键步骤，如算法1所示。

*−*

在RB和BB中，不同QoE因素之间的权衡没有明确定义，因此只能以特定的定性方式解决。

# 鲁棒最大功率控制

1. *预测:预测吞吐量*

[tk，t ]k+N

下一个N

基本的最大功率控制算法假设交流电流的存在

使用吞吐量预测器的块。我们的目标是精确的吞吐量预测。然而，在某些严重的网络中-

工作条件，例如在蜂窝网络中或者在互联网拥塞的黄金时间，这种精确的预测器可能不可用。例如，如果预测器始终高估吞吐量，它可能会导致高再缓冲。为了抵消预测误差，我们开发了一种鲁棒的多预测控制算法。

|  |  |
| --- | --- |
| 缓冲级别 | 1s |
| 预比特率 | 350kbps |
| 吞吐量 | 350kbps |

鲁棒MPC本质上优化了最坏情况下的QoE as-苏明，即实际吞吐量可以取一个范围[c7t，c7t]内的任何值，与点估计c7t相反。鲁棒MPC需要解决以下优化问题

**…**

**离线枚举**

|  |  |
| --- | --- |
| 缓冲级别 | 2s |
| 预比特率 | 350kbps |
| 吞吐量 | 600kbps |

|  |  |
| --- | --- |
| 缓冲级别 | 20多岁 |
| 预比特率 | 3000kbps |
| 吞吐量 | 3000kbps |

**…**

CPLEX

CPLEX

CPLEX

时间t

为了获得比特率

(右 *，B，[C，c]):*

*k k robustmpc*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 方案 | 缓冲级别 | 预比特率 | 吞吐量 | 最佳比特率 |
| 一 | 1s | 350kbps | 350kbps | 350kbps |
| 2 | 2s | 350kbps | 600kbps | 600kbps |
| **…** | **…** | **…** | **…** | **…** |
| 50,000 | 20多岁 | 3000kbps | 3000kbps | 3000kbps |

*k 1k t t*

最大

最小Qoek+n1

(15)

*Rk，，Rk+n1 C∈[C，C]*

*t*

*t*

*t*

*k*

查询查找

*特别条款限制(7)至(11) (16)*



一般来说，解决这样一个最大-最小鲁棒优化问题可能不是小事。然而，在我们的具体案例中，我们可以证明最坏的情况发生在

**在线比特率自适应**

### 图5:“快速决策”思想:我们列举了可能的scenario，并为每个场景创建了一个索引最优决策的表。

吞吐量处于其下限Ct = Cˆ。因此

*•*

鲁棒MPC的实现是简单的。我们使用最低可能的Ct作为常规MPC QoE最大化问题的输入，而不是Ct。形式上，

定理1。鲁棒多点控制控制器相当于以吞吐量下限为输入的常规多点控制，即，

*•*

*计算开销:首先，MPC的高计算开销对于低端移动设备来说尤其成问题，这些设备预计将成为未来的主流视频消费者。由于比特率自适应决策逻辑是在播放器开始下载每个块之前调用的，比特率自适应逻辑中的过度延迟将对播放器的QoE产生负面影响。*

*Rk = frobustmpc*

(Rk 1，Bk，[Ct，Ct])

*部署:由于我们没有针对QoE最大化问题的封闭式或组合式解决方案，*

= fmpc(Rk 1，Bk，Ct)

验证草图。从概念上讲，QoE函数QoE(R，C)可以写成3项之和(g1:总视频质量，g2:总质量变化，g3:再缓冲时间)，其中只有再缓冲时间项取决于吞吐量C。因此，

我们将需要使用求解器(例如，CPLEX或Gurobi)。但是，视频播放器可能无法与这种解算器功能捆绑在一起；例如，许可问题可能会阻止分发此类软件，或者它可能需要额外的插件或软件安装，这对采用造成重大障碍[26]。

最大

*稀有*

部

*碳[碳，碳]*

∈

*生活质量*

从上面的讨论中，很明显我们开发的解决方案应该是轻量级的和组合的(即，

！不在线解决LP或ILP)。因此，在本节中，我们

最大

*≡*

*稀有*

*g1(R) λ g2(R)最大值*

*C∏C，C]*

*—* ** *−*

* g3(R，c)*

通过开发一个快速的

和不需要任何开销的低开销FastMPC设计

最大生活质量

*≡*

*稀有*

由于吞吐量C的任何降低都会导致更长的重新缓冲时间，因此在C = C时会达到最小QoE

一个潜在的缺点是，稳健的最大功率控制比常规的最大功率控制更保守，总是假设吞吐量最低。这里的保守程度完全取决于下限有多宽松/多紧。实际上，我们在实现中使用过去几个块的最大预测误差作为界限，并发现它在实践中运行良好(在第7节中讨论)。

# 在实践中使用最大功率控制——快速最大功率控制

虽然基于速率和基于缓冲区的算法需要相对较小的计算，但MPC的挑战是我们需要在每个时间步长解决离散优化问题。这里有两个实际问题:

视频播放器中的显式解算器功能[48]。

# 快速多点控制的高层思想

在高层次上，FastMPC算法本质上遵循表枚举方法。在这里，我们执行一个离线步骤，枚举状态空间并求解每个特定的实例。然后，在联机步骤中，我们只使用这些存储的映射到当前操作条件的最优控制决策。也就是说，该算法将简化为一个简单的表查找，由最接近当前状态的键值进行索引，并且查找的输出是所选配置的最佳解决方案。

在我们的设置中(图5)，状态空间由以下维度决定:(1)当前缓冲级别，(2)选择的先前比特率，以及(3)下一个N个块的预测吞吐量(即计划范围)。因此，FastMPC将需要列举潜在的场景来捕获不同的

输入每个维度的值，并离线解决优化问题。

不幸的是，直接使用这个想法将是非常不方便的，因为我们有一个高维的状态空间。例如，如果我们有100个可能的缓冲级别值、10个可能的比特率、5个大小的范围和1000个可能的吞吐量值，表中将有1018行！四

这种大的状态空间有两个明显的后果。

首先，在内存中显式存储整个表可能不切实际。请注意，这不仅仅是一个假设。如果我们需要在dash.js player [1]中实际实现这种表查找，这将意味着非常高的内存占用和很大的启动延迟，因为表需要下载到播放器模块。第二，它会产生不小的离线计算成本，可能需要随着操作条件的变化而重新运行。

# 优化FastMPC性能

接下来，我们提出两个关键的优化，使表枚举方法易于处理。

通过宁滨压缩:首先，为了解决离线扩展成本，我们的见解是，我们可能不需要非常细粒度的缓冲区和吞吐量级别值。因此，这些值可以适当地粗化成集合箱。此外，对于宁滨，我们不需要显式存储行键，因为这些键是直接从bin行索引中计算出来的。这里的挑战是平衡宁滨粒度和实践中最优性的损失。在实践中，我们发现使用100个箱进行缓冲级别和100个箱进行吞吐量预测效果很好，并产生接近最优的性能。

表压缩:我们的第二个见解是，离线计算学习的决策表将具有显著的结构。具体来说，几个类似场景的最佳解决方案可能是相同的。因此，我们可以将这种结构与宁滨策略相结合，探索一种使用游程编码来存储决策向量的简单无损压缩策略。然后可以使用二分搜索法在线检索最优决策。在实践中，我们看到在压缩的情况下，表占用不到60 kB，其中100个箱用于缓冲级别，100个箱用于吞吐量预测，5个比特率级别。

# 履行

在这一节中，我们描述了在dash.js框架中实现MPC的方法。我们的实现基于dash.js主分支(v1.2.0版本)，因为它是开发时的稳定版本。我们相信我们的实现可以很容易地适应未来的版本，因为我们只需要很少的修改(800行JavaScript)。有关源代码和演示的更多信息，请访问我们的演示页面[14]。

*≈*

播放器的选择:许多以前的自适应比特率播放器是使用Adobe OSMF框架原型[34，3，12]

4100缓冲级别 10位速率 1000吞吐量1值

*1000吞吐量5个值= 1018个条目。*

这似乎是一个自然的选择。然而，我们与行业人员的交谈显示，几乎所有内容提供商都在转向基于MPEG-DASH标准的HTML5播放器[16]，因此OSMF(基于Flash，市场份额不断下降)不太可能成为一个具有现实影响的平台。选择了DASH播放器后，我们定性地评估了DASH标准的几个实现(例如[23，8，43])。不幸的是，这些依赖于定制客户端或利基视频播放器平台。考虑到这些因素，我们选择了dash.js框架，因为它是MPEG-DASH标准的参考开源实现，并得到了领先行业参与者的积极支持[7]。我们相信我们的原型工作也将为这些标准化工作的发展提供信息。例如，任何控制算法的一个关键要求是知道每个视频块的大小(以字节为单位)，但是该标准没有要求清单报告块大小，这可能是当前规范的一个关键缺点。

dash.js概述:为了理解我们的实现和修改，我们从dash.js播放器架构的一些简单背景开始。关键组件在图6中突出显示。

在高级别上，dash.js实现将高级别的视频流功能与低级别的特定dash标准相关组件分开。因为我们对特定于标准的实现不特别感兴趣，所以我们不修改代码，只关注自适应流相关的功能。

比特率适配和视频流逻辑的关键类别和功能如下:

BufferController:这个类提供了通过请求新片段和做出比特率改变决定来管理播放器缓冲级别的功能。具体来说，会定期调用函数validate，并调用AbrController类中的getPlaybackQuality函数来找到最佳比特率。它还维护了一个变量bufferLevel来记录播放器的当前缓冲区占用率，这可以用于比特率决策。

*•*

这个类包含核心比特率适配逻辑。在最初的dash.js实现中，使用基于规则的决策逻辑来找到比特率。具体来说，下载比率规则根据“下载比率”(最后一个块的播放时间除以其下载时间)选择策略；另一方面，BufferRule根据缓冲级别最近是否达到下限来选择比特率，以避免重新缓冲。为每个规则分配优先级，以解决冲突并做出最终比特率决定。

*•*

修改和扩展:我们在dash.js中观察到两个有问题的实现细节。首先，代码定期调用validate函数来检查缓冲区的状态，并调用AbrController中的函数来决定当前比特率是否应该更改。请注意，这意味着比特率决定并不总是在块边界做出，这可能会导致比特率决定的执行延迟，甚至重新加载以前的块。

|  |
| --- |
| 基于规则的决策逻辑 |
| FastMPC |
| RB、BB、喜庆 |

附加类/函数



缓冲控制器

AbrController

吞吐量预测器

getPlaybackQuality

使生效

记录

### 图6: dash.js代码结构和我们的修改

第二，dash.js并行下载多个块，即使视频流中较早的块应该优先。

为了解决这些问题，我们改变了比特率

3Mbps，以避免选择最大比特率总是最佳解决方案的琐碎情况。

1. *移动数据集(HSDPA)[10]:HSDPA数据集由挪威电信3G/HSDPA移动无线网络中移动设备的视频流吞吐量的30分钟连续1秒测量组成。我们经常从完整的数据集中挑选1000个吞吐量跟踪。*
2. *合成数据集:最后我们还使用合成数据集来补充前面提到的数据集。通过是基于一些隐藏的状态St* 对共享瓶颈链路的用户数量进行建模。实际吞吐量Ct遵循高斯分布

*∈ *

给定隐藏状态的值，均值ms和方差σ2

dash.js代码中的cision和chunk下载过程，方法是对BufferController类进行两个关键更改:

*St = s*

*s*

。我们改变状态转移概率ma-

1)比特率决定是在每个块开始时做出的，2)块下载是完全顺序的，即不允许多个块的同时下载。这允许与我们的模型和其他提议的算法一致的基本实现框架。

通过这些修复，我们实现了不同的比特率适配算法(例如，FastMPC、BB、RB)，通过我们自己的实现替换了最初的基于规则的比特率适配逻辑。FastMPC实现有一个静态表，用于索引控制决策。我们还在先前工作的基础上实现了一个基于调和平均值的吞吐量预测方案[34]，以及BufferController类中的附加日志功能，以记录播放器状态的完整日志，包括缓冲级别、比特率、再缓冲时间、预测/实际吞吐量。

# 估价

在这一部分，我们使用真实玩家和模拟实验的组合，将我们的方法与现有的基于速率和缓冲区的方法进行比较。我们还提供了关于我们的FastMPC实现的CPU和内存开销的微基准测试。

# 设置

我们从描述关键参数开始:(1)吞吐量变化轨迹；(2)视频特定参数；(3)各种自适应算法的配置；以及(4)我们在本节中使用的标准化QoE度量的定义。

## 输入参数

吞吐量跟踪:我们的目标是使用真实的网络可变性条件来评估各种比特率自适应方法。然而，考虑到在几十秒内大规模持续输出测量的缺乏，我们使用了现有数据集和合成模型的组合:

1.宽带数据集(FCC)[9]:FCC数据集由100多万组吞吐量测量组成，其中每组包含六个数据点，每个数据点代表5s间隔内的平均吞吐量。我们提取相同服务器和客户端IP地址的吞吐量跟踪，并将它们连接起来以匹配视频的长度。对于实验，我们随机挑选1000个平均吞吐量在0到之间的串联轨迹

trix以及产生走线的参数ms、σ2。

图7显示了三者的吞吐量特性

*s*

数据集。在三个数据集中，吞吐量在宽带网络中最稳定，在移动网络中变化最大。换句话说，HSPDA数据集是对我们的MPC方法的一个很好的压力测试，它假设吞吐量在短时间尺度上是可预测的。

视频参数:我们使用的是DASH- 264 JavaScript参考客户端测试页[6]中的“Envivio”视频，时长260s，由65个4s组块组成。视频由H.264/MPEG-4 AVC编解码器以以下比特率级别编码:

= 350kbps、600kbps、1000kbps、2000kbps、3000kbps。这与YouTube视频比特率级别分别为240p、360p、480p、720p和1080p的要求一致[15]。我们将缓冲区大小设置为Bmax = 30s。We as- sume q()是一个恒等式函数。作为默认的QoE函数，我们使用权重λ = 1，= s = 3000，这意味着1秒的再缓冲/启动时间所受到的惩罚与将块的比特率降低3000 kbps相同。我们还进行了改变QoE权重的敏感性实验。

**

** ** **

## 算法和度量

自适应算法:确定每个类中的最优算法是困难的，因为它涉及在无限维的函数空间上进行优化。为此，我们从以前的工作中为每类算法选择了一种广泛采用的函数形式，并通过基于包含100条吞吐量轨迹的训练数据集的经验模拟来优化自由参数。我们评估以下算法:

1. *RB:比特率被选为最大可用比特率，该比特率小于使用过去5个块的调和平均值的吞吐量预测的p = 1倍；*
2. *BB:我们采用黄等[33]建议的函数，其中比特率Rk被选择为最大可用比特率，当储层r = 5s和缓冲层c = 10s时，该比特率小于rk = f (Bk)。*
3. *FastMPC:我们使用前瞻水平h = 5，吞吐量预测使用过去5个区块的调和平均值；我们使用100个容器进行吞吐量预测，100个容器用于缓冲级别。我们还评估了精确的最大功率控制，并对接下来的5个块进行了完美的吞吐量预测。*

1 1 1

HSDPA联邦通信委员会

合成的

HSDPA联邦通信委员会

合成的

HSDPA联邦通信委员会

合成的

0.8

0.8

0.8

0.6

累积分布函数（Cumulative Distribution Function的缩写）

0.4

0.2

0.6

0.4

累积分布函数（Cumulative Distribution Function的缩写）

0.2

0.6

0.4

累积分布函数（Cumulative Distribution Function的缩写）

0.2

0

0 1000 2000 3000 4000 5000

平均吞吐量(千位/秒)

0

0 500 1000 1500 2000

吞吐量的标准偏差(kbps)

### 图7:数据集的特征

0

−0.1 0 0.1 0.2 0.3 0.4

平均百分比预测误差

一

0.8

0.6

累积分布函数（Cumulative Distribution Function的缩写）

0.4

RB BB

FastMPC RobustMPC dash . js FESTIVAL

一

0.8

0.6

累积分布函数（Cumulative Distribution Function的缩写）

0.4

RB BB

FastMPC RobustMPC dash . js FESTIVAL

一

0.8

RB BB

FastMPC RobustMPC dash . js FESTIVAL

0.6

累积分布函数（Cumulative Distribution Function的缩写）

0.4

0.2

0.2

0.2

0

−0.5 0 0.5 1

标准化QoE

1. 催化裂化数据集

0

−0.5 0 0.5 1

标准化QoE

1. HSDPA数据集

0

−0.5 0 0.5 1

标准化QoE

1. 合成数据集

### 图8:不同吞吐量轨迹的真实实验结果

1. *RobustMPC:我们假设吞吐量下限为cˇt = cˇt/(1+err)，其中cˇt是使用过去5个组块的har- monic均值获得的，而预测误差err是过去5个组块的最大绝对百分比误差。*
2. dash.js:最初的实现采用了基于规则的比特率决定逻辑，如第6节所示。我们保持原始比特率自适应逻辑不变，但禁用多块下载，并允许策略仅在块边界切换。5
3. *喜庆[34]:这种基于速率的算法平衡了效率和稳定性，并在*

标准化生活质量指标:我们定义了一个标准化生活质量指标

里克如下。对于给定的吞吐量轨迹Ct，t[t1，tK+1]，离线最优QoE，用QoE(OPT)表示，是在完全了解以下内容的情况下可以实现的最大-最小QoE

** *∈* **

整个地平线上的未来吞吐量。它可以通过求解问题QOE MAX K6来计算，并给出了可达QOE的理论上界。另一边

一

一方面，真正的在线算法A基于当前吞吐量预测Ct，t > tk选择比特率Rk，而不知道整个未来。我们用QoE(A)来表示算法A实现的在线QoE，并将算法A的A (n- QoE(A))的归一化QoE定义为:n-QoE(A) = QoE(A)。

** **

*生活质量*

玩家，但这不是本文关注的问题。我们假设

在连续的块下载之间没有等待时间，并且在没有随机化块调度的情况下实现FESTIVE。请注意，这不会对单人游戏的玩家QoE产生负面影响。具体来说，FESTIVE使用过去5个块的谐波平均值以及作为过去5个块中比特率切换的函数的稳定性分数，根据p = 1倍吞吐量预测来计算效率分数。选择比特率是为了最小化稳定性分数加上α = 12倍效率分数。

吞吐量预测器:注意RB、\* MPC和FES-active需要一个好的吞吐量预测器。为不同的场景开发好的预测器超出了本文的范围。基于先前工作的见解，我们使用最近5个数据块的观测吞吐量的平均值，因为它对每个数据块估计中的异常值是稳健的[34]。我们将在第8节重新讨论这个问题。

5这样可以对算法进行一致的比较

# 真实玩家评价

首先，我们给出了真实玩家设置的仿真，将我们的快速MPC方法与之前的几种方法进行了比较。我们的基本实验装置由两台计算机组成(Ubuntu

12.04 LTS)与100Mbps直接网络连接，连接视频客户端和服务器。视频客户端是一个带有V8 JavaScript引擎的适用于Linux(39版)的Google- Chrome网络浏览器，而视频服务器则是一个基于node . js(0 . 10 . 32版)的简单HTTP服务器。我们使用linux tc工具根据所采用的吞吐量跟踪来限制两台计算机之间链路的吞吐量。我们使用Emu- lab [49]并行进行了几个这样的实验。

图8显示了三组吞吐量轨迹上归一化QoE的CDF。首先，我们看到现有算法仅实现了60-70%的最佳QoE，这证实了视频QoE仍有很大的提升空间。其次，Ro- bustMPC在所有数据集上都优于非MPC算法，在FCC、HSDPA和合成数据集上的中值归一化QoE分别提高了15%、10%和5%

而不是把它和其他文物混为一谈

lel下载。我们还测试了原始的dash.js，没有任何修改，但是它的性能比我们的修改版本(未显示)更差。

6为了便于离线计算这个最优值，我们假设它可以从一个连续的范围内选择比特率[Rmin，Rmax]。

1 1 1

0.8

0.6

累积分布函数（Cumulative Distribution Function的缩写）

0.4

0.2

0

0 500 1000

1500 2000 2500 3000

0.8

0.6

累积分布函数（Cumulative Distribution Function的缩写）

0.4

0.2

0

0

500 1000 1500

0.8

0.6

累积分布函数（Cumulative Distribution Function的缩写）

0.4

0.2

0

0

RB BB

FastMPC RobustMPC dash . js FESTIVAL

5 10 15 20 25 30

平均比特率(kbps)

平均比特率变化(kbps/块)

### 图9:面心立方数据集的详细性能

总再缓冲时间

1 1 1

0.8

0.6

累积分布函数（Cumulative Distribution Function的缩写）

0.4

0.2

0

0 500 1000 1500 2000

2500 3000

0.8

0.6

累积分布函数（Cumulative Distribution Function的缩写）

0.4

0.2

0

0

500 1000 1500

0.8

0.6

累积分布函数（Cumulative Distribution Function的缩写）

0.4

0.2

0

0 5 10

15 20

RB BB

FastMPC RobustMPC dash . js FESTIVAL

25 30

平均比特率(kbps)

平均比特率变化(kbps/块)

### 图10:HSDPA数据集的详细性能

总再缓冲时间

分别是。第三，与原始dash.js播放器相比，我们看到了显著的改进(60+%的中值归一化QoE)。最后，我们发现基本快速预测控制比鲁棒预测控制对预测误差更敏感。虽然在联邦通信委员会和合成结果上，快速和鲁棒预测控制之间没有区别，但这种区别在HSPDA结果中尤其明显，在该结果中，常规快速预测控制受到影响，与RB和BB相比没有增益。

为了更好地理解预测误差的影响，图7显示了数据集的每次会话平均预测误差百分比的CDF。在FCC数据集中，我们的谐波平均吞吐量预测器的平均误差小于5%，而在HSPDA数据集中，最坏情况的预测误差可高达40%。我们还观察到，在HSPDA数据集中，预测器在超过20%的时间内高估了真实吞吐量，这导致了显著的再缓冲。因此，不准确的预测会破坏常规快速预测的决策制定，而鲁棒预测的影响较小，因为它包含了预测误差，以避免在预测不准确时过于激进地选择比特率。

早期的标准化生活质量结果显示了不同生活质量因素的综合组合。接下来，我们放大单个质量因素来解释图9和10中的QoE改善。在FCC数据集中，由于吞吐量是可预测的，所有算法都实现了类似的低再缓冲时间。性能差异本质上源于减少不必要的比特率切换。RobustMPC、FastMPC和BB实现了相似的平均比特率，但Ro- bustMPC使用的比特率开关更少。在HSPDA结果中，再缓冲时间成为一个更重要的问题。虽然与BB相比，FastMPC实现了类似的平均比特率和更少的开关，但它遭受了大量的再缓冲时间。另一方面，RobustMPC显著缩短了再缓冲时间，但平均比特率略低:65%的情况下零再缓冲，而BB和FastMPC为40%。因此，RobustMPC在整体QoE方面仍然优于其他算法。

做一个跨数据集分析，我们看到总体QoE的尾部分布呈现不同的特征。在联邦通信委员会的结果中，只有1%的用户体验到正常的生活质量

< 0而在HSPDA，这种情况占所有病例的10%。7同样，主要原因是移动网络的高可变性导致长时间的再缓冲，从而影响整体QoE。

最后，尽管FESTIVE是一种基于速率的算法，但在我们的数据集中，它的性能比常规RB稍差，因为它更重视稳定性，即使在可用吞吐量增加的情况下，它也能缓慢地切换比特率。8另一方面，dash.js基于启发式规则的自适应实现了较低的再缓冲时间，但会导致许多不必要的切换。因此，它的整体QoE明显比所有算法差。

# 灵敏度分析

对于灵敏度分析，我们使用定制的模拟框架评估不同的算法。像以前一样，模拟将吞吐量跟踪作为输入，并对视频下载/回放过程和缓冲动态进行建模。在时间tk，当需要块k的比特率时，模拟调用嵌入不同算法的比特率控制器来获得Rk。使用这个框架，我们研究了这些方法对关键因素的敏感性，例如:(1)预测误差，(QoE函数的选择，(3)播放缓冲区大小，(4)比特率级别的数量，以及(5)启动延迟。

吞吐量预测:在这里，我们希望研究一般预测器的预测误差的影响，而不是分析特定的预测器(例如调和平均)。为此，我们使用平均误差水平来表征吞吐量预测器的性能，并根据平均误差水平将预测输出建模为真实吞吐量与添加的随机噪声的组合。图11a显示

7当再缓冲时间过长或开关过多时，QoE可能为负值。

8这不是一个缺陷，而是实现多方公平的一个深思熟虑的选择[34]。

一

0.95

0.9

0.85

QoE

0.8

0.75

0.7

0.65

MPC BB RB

一

0.9

MPC-OPT

FastMPC BB

铷

0.8

0.7

0.6

QoE

0.5

0.4

0.3

0.2

0.1

0

一

0.8

0.6

QoE

0.4

0.2

0

MPC-OPT

FastMPC BB

铷

一

0.9

0.8

QoE

0.7

0.6

0.5

MPC-OPT

FastMPC BB

铷

0.1 0.2 0.3 0.4 0.5

预计误差

1. 预计误差

平衡避免不稳定避免再缓冲

生活质量偏好

1. QoE首选项

10 20 30 40 50

缓冲区大小

1. 缓冲区大小

2 4 6 8 10

启动时间

1. 启动时间

### 图11:灵敏度分析与工作条件的关系

吞吐量预测误差如何影响比特率自适应算法的性能。不出所料，BB不受影响，因为它不使用任何吞吐量信息。当吞吐量预测准确时，MPC比BB算法具有更大的优势。当预测误差超过25%时，最大功率控制甚至可能比最大功率控制更差。这意味着，如果实际预测误差很大，那么视频播放器应该放弃RB或MPC，使用纯BB算法。与常规预测控制相比，鲁棒预测控制受预测误差的影响较小，因为它考虑了可能的误差，并使最坏情况下的QoE最大化。

用户的QoE偏好:我们比较了3组QoE权重下算法的性能，“平衡”(λ = 1，= s = 3000)、“避免不稳定”(λ = 3，= s = 3000)、“避免再缓冲”(λ = 1，= s = 6000)。如图11b所示，随着用户对比特率不稳定性给予更多的惩罚权重，MPC算法比RB和BB更具优势。这是因为MPC算法明确地模拟了QoE函数中比特率与比特率不稳定性的权衡，而RB和BB是以特定的方式进行的。然而，当再缓冲时间是一个更重要的因素时，由于两个关键原因，BB算法的性能与FastMPC算法相似。首先，BB算法保持一个最小的缓冲级别，这样玩家就有更好的机会以更少的/没有再缓冲的时间度过低吞吐量。第二，尽管MPC算法在完美的吞吐量预测方面做得很好，但由于谐波均值预测器不完美，它们可能会遭受较长的再缓冲时间。因此，可以通过保持最小缓冲水平和采用更精确的预测器来改善最大功率控制。

缓冲区大小和启动延迟:图11c分析了播放缓冲区大小的影响。首先，当缓冲区大小较小时(<播放时间25s)，增加缓冲区大小可以提高所有算法的性能。更大的缓冲区保护玩家免受再缓冲事件的影响，还提供了更多的自由度来优化性能。当缓冲区大小达到一定水平(播放时间为25s)时，即使缓冲区大小进一步增加，所有算法的性能也保持不变。最后，RB受缓冲区大小的影响最小，因为它在决策逻辑中没有考虑缓冲区级别。

虽然我们的方法会自动优化启动延迟，但我们会分析如果启动延迟固定，整体QoE(启动延迟项除外)会受到什么影响。如图11d所示，随着启动时间的增加，随着播放器积累更多的视频，所有算法的性能都会提高

在启动阶段存储在缓冲器中，使得管理再缓冲事件更加容易。

比特率水平:我们还研究了比特率水平的数量如何影响性能(未显示)。通过BB和MPC，我们可以使用更细粒度的比特率级别集来实现更好的性能。然而，对于RB，当我们添加更多比特率级别时，RB的性能首先会提高，但是当比特率级别过多时，性能会降低。原因是RB开始更频繁地改变比特率，导致比特率不稳定性增加。MPC的一个警告是，对于快速MPC实现，更细粒度的比特率级别也需要更多的离散化级别。理解这种权衡是未来工作的一个有趣方向。

# MPC配置和开销

开销:如前所述，相对于BB和RB风格的算法，FastMPC可能会增加玩家开销。我们比较了我们实现的FastMPC、BB和RB算法与默认dash.js播放器的CPU和内存使用情况。我们发现，FastMPC、BB和RB都消耗了相似数量的CPU，而FastMPC只多使用了60 kB的内存(未显示)。

**FastMPC离散化:回想一下，离散化级别的数量是FastMPC的一个重要设计参数。更多的离散化级别会提高FastMPC的性能，但需要更多的播放器内存，还可能会增加启动延迟。我们研究了这种性能与开销的权衡**

在图12a和表1中。从图12a中，我们看到更多的离散化级别意味着FastMPC有更大的性能提升，但这种提升显示出收益递减；例如，FastMPC在100级时达到90%的最佳QoE，而如果只有5级，则降至70%。其次，增益与离散化水平的关系也与吞吐量预测值有一定的相关性，尤其是在非常粗略的离散化情况下。表1显示，虽然内存开销

额外的JavaScript代码大小

|  |  |  |
| --- | --- | --- |
| 离散化级别 | 满桌 | 游程编码 |
| 50 | 25.0 kB | 19.1 kB |
| 100 | 100 kB | 56.4 kB |
| 200 | 400 kB | 141 kB |
| 500 | 2.50兆字节 | 451 kB |

### 表1: FastMPC表大小

一

0.8

0.6

QoE

0.4

0.2

0

快速预测+完美预测快速预测+调和平均

0.95

0.94

0.93

0.92

QoE

0.91

0.9

0.89

0.88

最大功率控制，误差= 10%最大功率控制，误差= 15%最大功率控制，误差= 20%

吞吐量预测:正如其他研究人员观察到的，更好的吞吐量预测可以提高蜂窝网络中的视频性能[52]。我们工作的一个关键限制是，我们没有准确的吞吐量预测算法，而且文献出奇的稀少和过时[30，51，20]。未来工作的两个有趣方向是基于其他客户测量结果的众包方法[41]和更好地理解

5 10 50 100 500

快速离散化级别

1. [数]离散化

2 3 4 5 6 7 8 9

前瞻地平线(区块)

1. 前瞻地平线

野外吞吐量的可预测性和稳定性。

# 9结论

### 图12: MPC配置参数

随着层数的增加，我们前面讨论的简单压缩方案可以减少内存开销，尤其是当层数较多时。例如，100级的压缩率为0.5，而500级的压缩率可以将表的大小减少82%。即使有500个级别，表的大小也相当低。

前瞻视野:图12b显示了水平zon规划如何影响多点控制算法的性能。随着前瞻视野的增加，MPC性能增长并保持稳定，因为考虑了未来吞吐量的更多信息。然而，随着我们对未来的进一步研究，预测精度可能会降低。如果地平线太大，MPC的性能甚至会下降。

# 结果摘要

我们的主要发现总结如下:

1. RobustMPC在宽带(FCC)和蜂窝(HSDPA)数据集上都优于现有算法，而常规的FastMPC由于高吞吐量的不稳定性在蜂窝网络中没有表现出优势；
2. 我们对FastMPC算法的实现产生了非常低的开销:与原始dash.js相比，CPU开销接近于零，内存使用增加了60kb；
3. 灵敏度分析表明，在较宽的参数范围内，FastMPC优于BB和RB。然而，通过增加FastMPC离散化粒度和采用更精确的吞吐量预测器，仍有改进的空间。

# 讨论

在结束之前，我们再次讨论两个悬而未决的问题。

多玩家效果:在本文中，我们纯粹专注于改进单个视频播放器的设计。一个自然的问题是将这些见解扩展到多个参与者以及与交叉流量的交互[34，32]。为了充分考虑多方交互和公平性，我们可以扩展我们的控制理论模型，在QoE函数中明确考虑公平性项，并模拟TCP对吞吐量分配的影响。例如，在这种情况下，我们可以从博弈论或分布式控制理论的角度来推理公平。这是未来研究的一个有趣方向。

我们的论文是由最近围绕动态自适应HTTP流(DASH)算法设计的争论所推动的。为了给这个空间带来一些严谨性，我们开发了一个控制理论问题公式，使我们能够系统地探索设计空间，并通过明确定义的QoE指标评估不同类别的解决方案。与现有解决方案相比，我们拥有更广阔的设计空间，凭借这一重要见解，我们设计并实施了一种模型预测控制方法，以优化结合缓冲区占用率和吞吐量预测，从而最大限度地提高用户的QoE。我们恶魔-

用dash.js参考视频播放器演示了一个MPC的实际实现。我们使用真实吞吐量可变性跟踪的跟踪驱动仿真证实了在各种操作条件下，与现有解决方案相比的优势，并且计算和内存需求的增加可以忽略不计。作为未来的工作，我们计划引入更精确的吞吐量预测，并明确捕捉多方交互。

# 感谢

这项工作得到了美国国家科学基金会ECCS-0925964和CNS-1345305的部分支持。我们感谢我们的牧羊人基思·温斯坦帮助我们改进了最终版本。我们感谢Aditya Ganjam和David Oran关于行业玩家平台的有益讨论，这些讨论为我们的实施提供了信息。

# 10参考文献

1. dash-行业-论坛，dash.js。https://github.com/Dash-Industry-Forum/dash.js/wiki.
2. Adobe HTTP动态流。[www.adobe.com/products/hds-dynamic-streaming.html.](http://www.adobe.com/products/hds-dynamic-streaming.html)
3. Adobe OSMF播放器。[http://www.osmf.org。](http://www.osmf.org/)
4. Akamai高清网络。[www.akamai.com/hdnetwork.](http://www.akamai.com/hdnetwork)
5. 苹果的HTTP直播。https://developer.apple.com/streaming/.
6. DASH-264 JavaScript参考客户端登陆页面1.4.0。<http://dashif.org/reference/players/javascript/1.4.0/samples/>dash-if-reference-player/index . html
7. DASH行业论坛成员。[http://dashif.org/members/.](http://dashif.org/members/)
8. VLC插件。

[http://www-itec.uni-klu.ac.at/dash/?page\_id=10.](http://www-itec.uni-klu.ac.at/dash/?page_id=10)

1. FCC数据集。

[https://www](http://www.fcc.gov/measuring-broadband-america). fcc.go[v/测量-宽带-美国。](http://www.fcc.gov/measuring-broadband-america)查阅日期:2014-12-01。

1. HSDPA数据集。[http://home.ifi.uio.no/paalh/dataset/hsdpa-tcp-logs.](http://home.ifi.uio.no/paalh/dataset/hsdpa-tcp-logs)查阅日期:2014-12-01。
2. 网飞。[http://www.netflix.com/.](http://www.netflix.com/)
3. OSMF 2.0版本代码。http:

//source forge . net/project/osmf . adobe/files/latest/download。

1. 平滑流协议。[http://go.microsoft.com/?linkid=9682896.](http://go.microsoft.com/?linkid=9682896)
2. 演示页面为我们的基于MPC的比特率适配。[http://users.ece.cmu.edu/~vsekar/mpcdash.html.](http://users.ece.cmu.edu/~vsekar/mpcdash.html)
3. YouTube直播编码器设置、比特率和分辨率。https://support.google.com/youtube/answer/2853702?hl=en.
4. 一、索达加尔。因特网多媒体流的MPEG-DASH标准。IEEE多媒体，2011。
5. 南Akhshabi，L. Anantakrishnan，C. Dovrolis和A. C. Begen。当HTTP自适应流媒体播放器争夺带宽时会发生什么？在Proc中。NOSSDAV，2012年。
6. 南Akhshabi，L. Ananthakrishnan，A. Begen，和

C.多弗波利斯。稳定振荡自适应流媒体播放器的基于服务器的流量整形。在Proc中。ACM SIGMM NOSSDAV，2013年。

1. A.巴拉尚德兰、塞卡尔、阿凯拉、塞山、和张。开发网络视频体验质量预测模型。在Proc中。ACM SIGCOMM，2013年。
2. H.Balakrishnan，M. Stemm，S. Seshan和R. H. Katz。广域网络性能的稳定性分析。在Proc中。美国心理学会，1997年。
3. D.P. Bertsekas，D. P. Bertsekas，D. P. Bertsekas，和D. P. Bertsekas。动态规划和最佳控制，第1卷。雅典娜科学贝尔蒙特，马萨诸塞州，1995年。
4. E.卡马乔和阿尔巴。模型预测控制。斯普林格，2013年。
5. 长度D. Cicco，V. Caldaralo，V. Palmisano和S. Mascolo。TAPAS:自适应流算法快速原型开发工具。在Proc中。下一次研讨会，2014年。
6. F.多布良、塞卡尔、阿万、斯托伊察、约瑟夫，

A.甘杰姆，詹，张。了解视频质量对用户参与度的影响。在Proc中。ACM SIGCOMM，2011年。

1. G.富兰克林、鲍威尔和沃克曼。动态系统的数字控制，第3卷。爱迪生-韦斯利门洛公园，1998年。
2. A.甘杰姆，西迪奎，詹杰，刘晓星，，姜，

动词 （verb的缩写）塞卡尔和张。C3:互联网规模的视频质量优化控制平面。在Proc中。NSDI，2015年。

1. 米（meter的缩写））戈巴迪、程永安、贾恩和马西斯。涓流:限制YouTube视频流的速率。在Proc中。USENIX空管，2012年。
2. 南水粉，G. Bichot，A. Bsila，C. Howson。分布式和自适应HTTP流。在Proc中。ICME，2011年。
3. D.哈韦、切尔托夫和阿尔梅罗。无线多媒体应用的接收机驱动速率适配。在Proc中。MMSys，2012年。
4. 问:何、多弗波利斯和阿马尔。大传输TCP吞吐量的可预测性。在Proc中。ACM SIGCOMM，2005年。
5. R.胡黛尔和水粉。塑造HTTP自适应流以获得更好的用户体验。在Proc中。MMSys，2012年。
6. T.-黄、汉迪戈、海勒、麦肯恩和

R.乔哈里。困惑、胆怯和不稳定:选择视频流速率很难。在Proc中。国际管理委员会，2012年。

1. T.黄、乔哈里、麦肯恩、特朗内尔和

米（meter的缩写））华生。基于缓冲区的速率自适应方法:

来自大型视频流服务的证据。在Proc中。ACM SIGCOMM，2014年。

1. J.姜、塞卡尔和张。在基于HTTP的自适应视频流中使用FESTIVE提高公平性、效率和稳定性。在Proc中。CoNext，2012年。
2. 南克里希南和西塔拉曼。视频流质量影响观众行为:使用推断因果关系

准实验设计。在Proc中。国际管理委员会，2012年。

1. R.库什宁、屈夫勒和赫尔格纳。互联网视频流中基于HTTP的请求-响应流的评估。多媒体系统，第245-256页，2011年。
2. 长度德·西科、马斯科洛和帕米萨诺。自适应实时视频流的反馈控制。在Proc中。2011年美国计算机学会多媒体系统会议。
3. Z.李、朱晓明、贾汉姆、潘若尔、胡海峰、贝根和

D.奥兰。探测和适配:大规模HTTP视频流的速率适配。通信领域精选，IEEE杂志，32(4):719–733，2014。

1. C.刘、布瓦齐兹和加博伊。并行自适应HTTP媒体流。在Proc中。ICCCN，2011年。
2. H.刘，王永年，杨永年，田阿甲，王洪浩。优化内容多宿主的成本和性能。在Proc中。ACM SIGCOMM，2012年。
3. X.刘、杜布良、米尔纳、江、塞卡尔、、张。一个协调的互联网视频控制平面的案例。在Proc中。ACM SIGCOMM，2012年。
4. R.莫国平、罗世雄、陈东伟和张瑞昌。QDASH:一个感知QoE的DASH系统。在Proc中。MMSys，2012年。
5. C.穆勒、莱德雷尔、波切和蒂姆尔。Libdash

-MPEG-DASH标准的开源软件库。在Proc中。ICME，2013年。

1. 长度波帕、古德西和斯托伊察。HTTP作为未来互联网的窄腰。在Proc中。HotNets，2010年。
2. R.Rejaie和J. Kangasharju。摩卡:一种用于互联网流的质量自适应多媒体代理缓存。在Proc中。NOSSDAV，2001年。
3. 南Akhshabi，A. Begen，C. Dovrolis。HTTP自适应流中速率自适应算法的实验评估。在Proc中。MMSys，2011年。
4. G.田和李亚男。动态HTTP流中灵活流畅的视频适配。在Proc中。CoNext，2012年。
5. Y.王和博伊德。使用在线优化的快速模型预测控制。控制系统技术，IEEE交易，18(2):267–278，2010。
6. B.怀特，勒普罗，斯托尔勒，瑞西，古鲁普拉萨德，

米（meter的缩写））纽博尔德，m .希布勒，c .巴伯和a .乔格勒卡。分布式系统和网络的集成实验环境。继续。OSDI，2002年。

1. X.尹、塞卡尔和西诺波利。设计动态自适应流算法的原则框架。在Proc中。ACM SIGCOMM HotNets，2014年。
2. Y.张和杜菲尔德。论互联网路径属性的恒定性。2001年在IMW。
3. X.邹、埃尔曼、戈帕拉克里希南、哈勒波维奇、

R.贾娜，金，雷克斯福德和辛哈。准确的预测能改善蜂窝网络中的视频流吗？在Proc中。ACM HotMobile，2015。