Tambur: Efficient loss recovery for videoconferencing via streaming codes

Michael Rudow; Francis Y. Yan; Abhishek Kumar

April 17–19, 2023

摘要

数据包丢失会降低视频会议的用户体验（QoE）。对于因往返延迟过长而无法进行重传的远距离通信，通常使用前向纠错（FEC）来恢复丢失的数据包。然而，传统的实时应用 FEC 方法在应对丢失突发方面效率较低，而突发丢失在实际中常常发生。通过采用一种新型的理论 FEC 方案——“流式编码”，可以用显著更少的冗余来恢复突发丢失。然而，现有的流式编码方案无法满足视频会议的需求，其在提高视频会议 QoE 方面的潜力尚未得到充分验证。本文提出了一个基于流式编码的视频会议新方案——Tambur，解决了上述问题。我们首先在大量来自 Microsoft Teams 的轨迹模拟中评估 Tambur，发现相比基线方法，Tambur 减少了视频帧解码失败的频率（降低了 26%），同时将冗余使用的带宽减少了 35%。随后，我们用 C++ 实现 Tambur，并将其集成到一个视频会议应用中，通过模拟网络进行端到端 QoE 指标的评估，展示了在多个关键指标上的显著改进。例如，Tambur 使视频冻结的频率和累计时长分别减少了 26% 和 29%。

关键词： 视频会议；前向纠错编码；流式编码；

1 引言

视频会议通话的质量直接决定了远程会议的效果 [10]。如今，视频会议已无处不在，可以是点对点通话 [16]，也可以是多方通话 [19]。本研究聚焦于点对点通话。视频质量依赖于多个关键性能指标，例如视频卡顿、带宽、数据包丢失和延迟 [9] [17] [21]。

恢复丢失的数据包对于提供高质量的视频会议至关重要 [25]。即便丢失一个数据包，也可能导致视频帧无法渲染，甚至因压缩视频的帧间依赖性，阻止多个后续帧的渲染（即导致视频卡顿）。因此，视频会议应用程序通常在应用层处理数据包丢失问题。主要的解决方案包括重传和前向纠错（FEC）。这两种方法都需要传输冗余数据，因此存在带宽分配给冗余数据与原始数据之间的权衡。此外，视频会议应用需要在严格的延迟要求内（最好小于 150 毫秒）恢复丢失的数据包，以满足实时播放的要求。

重传方法由于只需重发丢失的数据包，冗余数据最少，因此在可能的情况下是首选 [30]。然而，由于实时延迟的限制，重传仅适用于往返时延较短的场景。对于其他场景，视频会议应用程序依赖于 FEC 在可接受的延迟内恢复丢失的数据包。

目前，块编码是生产系统中最常用的 FEC 形式。使用块编码时，将 k 个“数据包”生成

r 个冗余包（称为“校验包”）。当 $k+r$ 个包中有部分丢失时，仍然可以恢复 k 个数据包。由于有 r 个额外的校验包，因此带宽开销为 $(r/k)\times 100\%$。设计 FEC 方案的主要目标之一是尽量减少带宽开销。常见的块编码包括 Reed-Solomon（RS）块编码 [27] 和喷泉码（即无比率编码）[20]。许多编码（如 RS 编码）在随机丢失（即数据包独立丢失）的情况下是最优的。例如，在上述例子中，若使用 RS 编码，任意 k 个数据包即可完成恢复。因此，块编码在生产的视频会议应用中非常流行。例如，Microsoft Teams 使用 RS 编码。视频会议应用将压缩视频帧的数据分成多个包发送。若一个或多个连续帧的多个包丢失，则称为“突发丢失”。突发丢失可能由多种原因引起，例如持续的 Wi-Fi 干扰或网络拥塞（应用程序溢出路由器缓存引起的相关丢失 [18]）。对来自 Microsoft Teams 的成千上万次视频通话的包轨迹分析表明，视频会议应用面临的实际丢失确实表现为突发性。

在实时延迟要求下，块编码在应对突发丢失时的带宽消耗非常低效。相比之下，一种相对较新的理论 FEC 框架，称为“流式编码” [7] [22] [23]，可以高效地应对突发丢失，同时满足严格的延迟限制。简而言之，流式编码通过各自的播放截止时间逐帧顺序恢复突发丢失的包，而块编码则通过最早的播放截止时间一次性恢复所有丢失的包。使用块编码恢复丢失时，在恢复最后一个丢失包的截止时间之前发送的后续校验包往往是浪费的。大多数关于流式编码的工作集中在理论研究 [7] [11] [12] [15] [22] [23] [28] [29]，如研究边界和编码构造。一些现有工作 [6] [14] 探索了流式编码的实际应用，但仅限于 VoIP（即语音通信，而非视频通信）。

鉴于带宽和丢失恢复的双重重要性，流式编码对于视频会议应用具有吸引力。然而，这种方法面临两个主要挑战。第一，现有的流式编码与视频会议应用之间存在差距。大多数实际的流式编码变体 [6] [14] 仅适用于输入数据大小随时间保持恒定的场景。而在视频会议中，压缩视频帧的大小是变化的。第二，流式编码在提高真实世界视频会议应用 QoE 方面的效果尚未在真实数据上得到验证。

本研究解决了上述挑战，提出了 Tambur，一种新的基于流式编码的视频会议带宽高效丢失恢复通信方案。Tambur 包括两个部分。首先，一种新的流式编码，基于现有理论框架 [29] 开发，同时克服了其在真实世界视频会议应用中的局限性。具体来说，Tambur 允许为每帧指定带宽开销。此外，对于任何给定的带宽开销，Tambur 可以生成数据包和校验包，使其在（a）从仅部分帧丢失的突发中恢复时不显得过于悲观，且（b）对保护空间中的丢失具有鲁棒性。其次，该流式编码集成了一个机器学习（ML）模型，用于预测分配给流式编码的带宽开销。特别地，采用了一种轻量化的方法，只需一个简单的模型和单比特额外反馈。

我们分析了从 Microsoft Teams 采集的成千上万次视频通话的包轨迹，总结了三个关键观察结果：（a）数据包丢失经常以突发的形式出现；（b）丢失通常（但并非总是）会伴随一个无丢失的保护空间，其中有几帧未发生数据丢失；（c）生产环境中使用的代码（如 RS 编码）需要显著的带宽开销才能实时恢复丢失的数据包，这会耗尽用于传输原始数据的带宽。

我们实现了 Tambur，并与多个基线方法（包括“Block-Multi”，一种跨帧块编码方法，以及 Tambur 的几个变体：“Tambur-full-BW” ——与 Block-Within 的带宽开销相匹配）一同集成到我们开发的视频会议评测平台中。然后我们在模拟网络上评估这些方案对 QoE 的影响。Tambur、Tambur-full-BW 与 Block-Within 和 Block-Multi 相比，分别减少了 26% 和 29% 的视频冻结平均频率。已有研究表明 [24] [26] [32] 视频冻结对用户参与度有显著负面影响，这些改进进一步表明 Tambur 提升了 QoE。

综上所述，我们的主要贡献如下：

- 分析了大规模商业视频会议应用中视频通话的成千上万次数据包丢失日志，并表征了这些日志在流式编码使用上的适用性。据我们所知，这是首次利用大规模真实世界轨迹数据评估流式编码的潜力。
- 提出了 Tambur，该方案通过（a）设计一种新型流式编码以适配视频会议场景，以及（b）集成一个轻量化的 ML 模型预测带宽开销，弥补了流式编码理论与视频会议应用之间的差距。
- 实现一个新的基准测试平台，使视频会议的研究具有易于使用的接口，以集成和评估新的 FEC 方案。此外，在 C + + 中实现 Tambur、Block - Intra 和 Block - Multi，并使用该接口将它们集成到基准平台中。
- 通过仿真在大量的生产痕迹库上评估 Tambur，表明它同时降低了不可恢复帧的频率和带宽开销，分别降低了 26.5 % 和 35.1 %。
- 在仿真网络上评估 Tambur，并在与端到端 QoE 相关的关键指标上显示出显著的改进。

总体而言，据我们所知，这是首次验证流式编码能够改善视频会议 QoE 相关的关键指标。这项研究还展示了一种新型 FEC（流式编码结合基于学习的带宽分配）的潜力，可用于带宽高效的视频会议丢失恢复。

2 相关工作

2.1 传统 FEC 及其在视频会议中的挑战

块编码（Block Codes）是最常用的一类 FEC 方法，其基本思想是将 k 个数据包（如 $D[1], ..., D[k]$）编码成 r 个校验包，形成 $k + r$ 个数据包（如 $D[1], ..., D[k], P[1], ..., P[r]$），以便在部分数据包丢失的情况下仍能恢复 k 个原始数据包。当任意 k 个数据包足以完成恢复时，这类块编码被称为 “最大距离分离（MDS）” 编码。最著名的 MDS 编码之一是 Reed-Solomon（RS）块编码 [27]。其他块编码示例包括喷泉码（fountain codes，或无比率编码）[20] 和二维块编码 [33]。

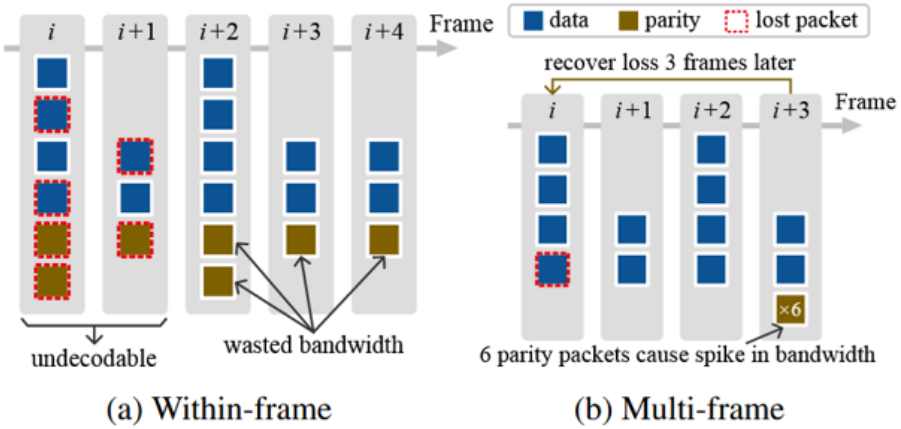


图 1. 使用块编码的两种方法：(a) 在每个帧内 (b) 跨多个帧。

传统上，FEC 应用于单个数据包，但视频会议需要为每个视频帧传输多个数据包。一种自然的解决方案是在每个帧的所有数据包内部应用块编码（见图 1a）。校验包会在每帧最后一个数据包之后立即发送。另一种方法是在多个帧的数据包间应用块编码（见图 1b），校验包在最后一帧的数据包之后发送。我们对 Teams 生产环境中的数据包丢失轨迹的分析表明，数据包丢失通常是突发性的。这两种方法在处理突发丢失时都存在显著局限性。

视频会议使用用块编码的局限性。当丢失以突发形式发生时，“帧内”方法浪费了紧随突发丢失帧之后发送的冗余数据，因为这些数据对恢复丢失帧没有帮助。尽管“跨帧”方法可以克服这一问题，但它有两个主要缺点。第一，恢复丢失的延迟较高，因为需要等待校验包（发送于块最后一帧之后）才能恢复任何数据包。如果块编码的长度过长，恢复延迟可能超过实时播放的时限，从而增加带宽开销并降低对突发丢失的鲁棒性。第二，如果路由器缓存已满，则快速连续发送的数据包可能丢失。当这种情况发生在块的最后一帧时，所有丢失的数据包将无法恢复。

FEC 的校验包消耗的带宽通常显著高于重传，甚至在数据包丢失率较低的情况下也是如此。与重传仅需重发丢失数据包不同，即便是“最优”的 FEC 方案也无法预知哪些数据包会丢失，因此必须发送远多于丢失数据包数量的校验包。例如，为防止视频卡顿，必须每约 150 毫秒发送至少一个校验包，以应对数据包丢失的情况。然而，如果没有数据包丢失，这些校验包是无用的。

2.2 流式编码

流式编码是一类专门应对突发丢失和发送方与接收方之间顺序通信的编码 [7] [22] [23] [28] [29]。从高层次看，流式编码通过每帧的播放截止时间顺序恢复突发丢失的每一帧数据，而非一次性恢复所有丢失帧（如块编码）。我们将详细描述流式编码的理论框架，并提供一个示例，随后讨论它如何成为受视频会议应用限制影响的一种有前景的选择。

流式编码由以下部分组成：（1）一个发送方，按固定时间间隔顺序生成数据包并将其发送给接收方。（2）一个对抗性数据包丢失通道，该通道引入长度为 b 的突发丢失，随后有保护空间内的数据包接收正常。（3）接收方需在指定的时间内恢复丢失的数据包。第 i 个时间点到达的数据包必须在第 $i+\tau$ 个时间点之前恢复完成。我们称参数 τ 为“延迟截止时间”。突发后保护空间的长度必须至少为 τ 个数据包（但更长的保护空间无必要，因为突发丢失应在 τ 数据包内恢复）。**顺序编码**。流式编码的顺序编码特性非常适合视频会议场景，在该场景下，每隔一定时间（例如，对于 30 帧/秒的视频，每 33.3 毫秒）就需要传输一个压缩帧的序列。我们将第 i 帧的视频数据符号表示为 $D[i]$，每个符号可以看作是一个比特向量。这些符号分布在一个或多个数据包中发送到接收方。此外，还会传输一些校验符号，表示为 $P[i]$，这些校验符号是前几帧数据符号的函数（线性组合）。

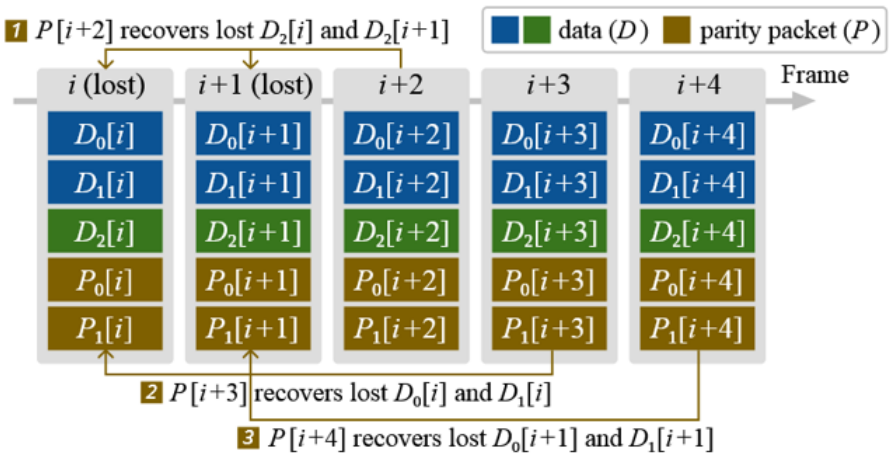


图 2. 使用流代码在 $\tau = 3$ 的延迟期限内恢复从第 i 帧开始的 $b = 2$ 个丢失帧。对于突发中的每个帧，发送 τ 个数据包的所有奇偶校验符号随后恢复其丢失的符号

顺序恢复。在流式编码模型下，延迟截止时间参数 τ 决定了丢失数据包的恢复延迟：如果第 j 个数据包 $D[j]$ 丢失，则必须使用从 $P[j]$ 到 $P[j+\tau]$ 的校验符号来恢复它。每个视频帧必须在严格的延迟限制内恢复，以便实时渲染。延迟截止时间参数 τ 是根据帧速率和单程延迟设置的，以实现适合的最大恢复延迟。例如，如果最大可容忍延迟为 150 毫秒（实时视频通信的标准值），单程传播延迟为 50 毫秒，并且每 33.3 毫秒编码一个帧（即 30 帧/秒），则 τ 可设置为 3（即 $(150-50)/33.3$）。

图 2 展示了使用现有流式编码（例如 [7] [22] [23]）在 $\tau = 3$ 的情况下，顺序恢复从第 i 帧开始的长度为 2 的突发丢失的示例。每帧包含相同数量的数据。首先，突发后立即发送的数据包中的校验符号恢复每个丢失帧的三分之一丢失数据符号（即 $D_2[i]$ 和 $D_2[i+1]$）。然后，第 i 和第 $i+1$ 帧的剩余丢失数据符号（即 $(D_0[i], D_1[i])$ 和 $(D_0[i+1], D_1[i+1])$）分别通过第 $i+3$ 和第 $i+4$ 帧发送的校验符号恢复。

流式编码通过在截止时间前依次恢复每个突发帧，实现了突发丢失的顺序恢复。对于跨多个连续帧（例如从第 i 帧到第 $i+b-1$ 帧）的突发丢失，突发中的数据包 $D[j]$ 是通过从 $P[i+b]$ 到 $P[j+\tau]$ 的校验符号恢复的。这种顺序恢复特性使流式编码可以利用在截止时间内接收到的所有校验符号。例如，$P[j+\tau]$ 被用于在 $D[i]$ 的截止时间（即 $P[i+\tau]$ 的接收时间）之后恢复 $D[j]$，对于 $i < j$。相比之下，块编码在第一个丢失帧的截止时间前（即接收到 $P[i+\tau]$ 符号时）一次性恢复所有丢失数据包，这会浪费后续发送的校验符号。这种关键差异使流式编码可以显著降低带宽开销；突发长度越长，流式编码的优势越大。然而，流式编码要求突发丢失后必须有至少 τ 帧的保护空间，否则某些帧可能无法在延迟截止时间内恢复。

2.3 流式编码在视频会议中的应用挑战

将流式编码用于视频会议主要面临两大挑战。首先，理论模型与实际系统之间存在显著差距，导致现有流式编码无法兼容视频会议应用。其次，流式编码在视频会议中的效果尚未在大规模真实轨迹数据中得到验证。因此，流式编码对提升视频会议应用 QoE 的潜力尚未明朗。这些挑战具体包括以下方面：

现有模型与视频会议应用之间的差距。现有的流式编码实用化研究 [6] [14]，如同其所依赖的理论研究 [6] [22]，仅适用于数据量在每个时间点恒定的场景。然而，视频会议涉及传输

压缩视频帧，其大小是变化的。尽管已有少量流式编码 [28] [29] 可处理这种变异性，但如前文所述，现有流式编码（包括 [28] [29]）的对抗性丢失模型假设突发长度为 b。当应用于视频会议时，该参数 b 转化为连续多帧的所有数据包丢失。然而，视频会议应用通常每帧包含多个数据包，而实际中每帧通常只有部分数据包丢失。现有流式编码过于悲观，因为它们需要从每帧的全部丢失中恢复，这种要求带来显著的带宽开销，从而抵消了流式编码的潜在带宽节省。此外，流式编码对突发丢失后的保护空间中的丢失较为脆弱，但实际中许多突发并未伴随这样的保护空间。

流式编码在真实场景中的适用性。流式编码对 VoIP 应用的研究主要通过理论丢失模型（例如 Gilbert-Elliott 通道 [13]）和模拟轨迹 [6] [14] 展开，在这些场景中，每帧被放入一个数据包，且所有帧/数据包大小恒定。然而，这些结果并不适用于视频会议应用，因为视频会议涉及：（a）每帧多个数据包；（b）每帧数据量变化。流式编码在突发跨多帧发生且伴随较长保护空间时表现最佳。一个自然的问题是，这种丢失模式是否存在于视频会议中，且是否能被流式编码利用。迄今为止，尚无大规模真实数据包丢失研究表明流式编码的适用性。此外，证明流式编码可以提升 QoE 的关键在于改进多个 QoE 相关指标。然而，现有文献中缺乏对流式编码对这些指标影响的分析。最后，尽管帧间依赖性在视频会议中普遍存在，但其对流式编码效果的影响尚未得到评估。

3 本文方法

3.1 本文方法概述

我们提出了 Tambur，通过以下两个核心创新实现：（1）使用机器学习（ML）模型预测决策来分配带宽开销；（2）设计了一种新的流式编码，使其能够在给定任何带宽开销设置的情况下适用于视频会议场景。

首先，ML 模型对每帧分配的校验符号数量进行预测决策，从而帮助动态调整带宽开销以匹配网络状况。其次，Tambur 定义的校验符号支持以下特性：（a）能够通过顺序恢复多帧中的突发丢失，同时利用帧内的部分丢失数据；（b）在单帧内发生偶然丢失时能够立即恢复；（c）对突发后的保护空间中少量数据包丢失具有鲁棒性。第三，Tambur 采用一种新方法将每帧的数据和校验符号分散到多个数据包中。Tambur 的流式编码由校验符号的设计及其在数据包中的分布组成。在丢失恢复过程中，Tambur 利用突发帧中接收到的数据包，从而实现比现有流式编码更低的带宽开销。

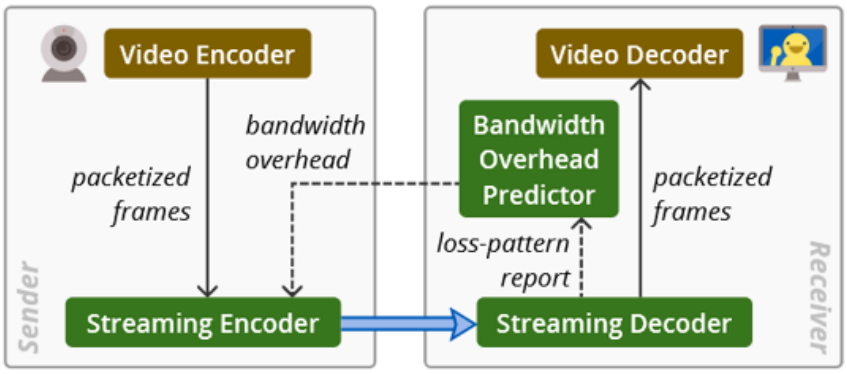


图 3. Tambur 架构图

图 3 展示了 Tambur 在视频会议应用中的结构。流式编码器将压缩帧的数据编码为数据包和校验包。带宽开销预测器基于解码器观察到的丢失情况，使用预测（ML）模型周期性地选择每帧的带宽开销，并将结果发送给编码器。流式解码器利用校验包恢复丢失的数据包。以下将详细描述这些组件。

3.2 Tambur 的流式编码

我们将代码分为两部分：编码和解码。

编码。我们将说明 Tambur 如何编码第 i 帧。图 4 展示了一个 $\tau = 3$ 的一个编码例子。该帧的数据符号，$D[i]$ 被装在数据包中。奇偶校验符号，$P[i]$ 被装在奇偶校验数据包。带宽开销预测器的先前值确定为帧 i 分配多少奇偶校验符号。这些奇偶校验符号会在 τ 个帧后发送（见图 4 中的 “reserved space”）。为第 i 帧发送的奇偶校验符号数量由帧 $(i-\tau)$ 的大小决定。

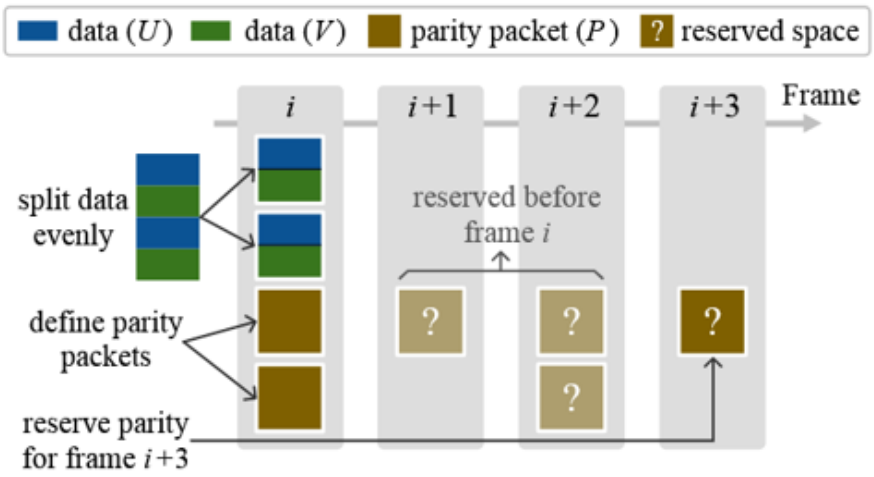


图 4. 编码在 $\tau = 3$ 时的示意图

接下来，我们描述奇偶校验符号是如何形成的。$P[i]$ 的符号是 $(\tau+1)$ 帧 $\{D[i], \ldots D[i-\tau]\}$。为了定义奇偶校验符号，有助于将相关 $(\tau+1)$ 帧的数据符号视为均匀地分为两部分，即 $D[j] = (V[j], U[j])$，其中 $j \in \{i, \ldots, i-\tau\}$。图 4 分别以蓝色和绿色显示了这些组件。

$P[i]$ 的符号是由 $V[i]$，…，$V[i-\tau]$，$U[i]$ 和 $U[i-\tau]$ 精心设计的线性组合。具体来说，$P[i]$

是三个量的和：$P[i] := P_1[i] + P_2[i] + P_3[i]$。$P_1[i]$ 的符号是 $V[i-\tau]$，...，$V[i-1]$ 的符号的线性组合。$P_2[i]$ 的符号是 $U[i-\tau]$ 的符号的线性组合。$P_3[i]$ 的符号是 $U[i]$ 和 $V[i]$ 的符号的线性组合。所有线性组合都经过精心选择，成为线性独立的线性方程。

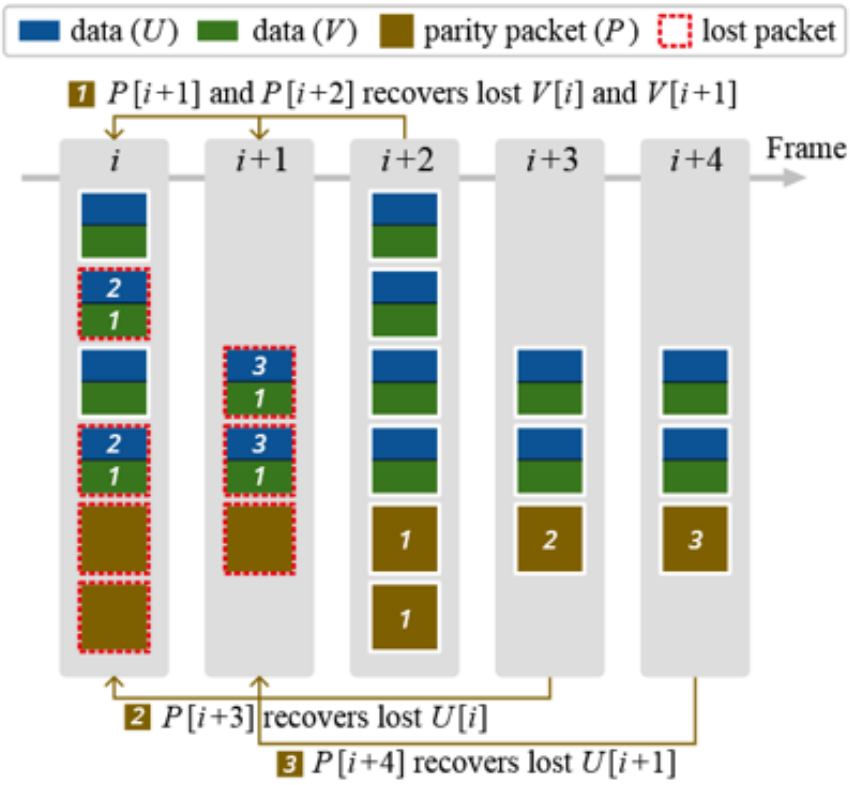


图 5. 使用 Tambur 的流式编码在 $\tau = 3$ 帧延迟内解码跨 2 帧的突发丢包。

解码。将解码过程分为两部分：（1）偶然丢包，和（2）突发丢包。让丢失之前的所有帧都被解码。首先，假设数据包丢失很少，并且 $P[i]$ 的大小超过帧 i 丢失的符号数。那么 $P[i]$ 足以解码第 i 帧（具体来说，通过求解线性方程组）。

其次，考虑 $= 3$ 时两个连续帧之间的突发数据包丢失，如图 5 所示。数据包丢失（红色虚线边框）跨越帧 i 和 $(i+1)$。对于每个帧 i，蓝色、绿色和棕色部分分别代表 $U[i]$、$V[i]$ 和 $P[i]$。首先，$V[i]$ 和 $V[i+1]$ 都使用 $P[i+2]$ 进行解码，$P[i+2]$ 由 (a)$V[i]$ 和 $V[i+1]$ 的符号的独立线性组合组成，并且 (b) $V[i-1]$、$U[i-1]$、$U[i+2]$ 和 $V[i+2]$ 的（接收到的）符号。其次，对于 $j \in \{i, i+1\}$，$U[j]$ 使用 $P[j+3]$ 进行解码，$P[j+3]$ 由 (a) $U[j]$ 的符号和 (b) 独立线性组合组成 $V[j-3]$ ，...，$V[j]$ 和 $U[j]$ 的（可用）符号。该方法的关键是 $U[i+1]$ 不会在 $D[i]$ 的延迟期限（即 $(i+3)$）之前恢复。这使得能够使用额外的奇偶校验符号（即 $P[i+4]$）来恢复 $U[i+1]$，同时仍然解码 $\tau = 3$ 帧内的每个数据包。如果解码失败，接收器会询问发送器生成新的关键帧（即自给自足的帧）来处理帧间依赖性。

与现有的视频会议流编码存在三个主要区别：(1) 帧的数据符号和奇偶校验符号通过多个数据包而不是单个数据包发送。(2) 每个帧的奇偶校验数据包的设计使其可用于恢复丢失的数据包（除了可用于恢复先前发送的帧之外）。(3) 该代码足够灵活，允许使用带宽开销预测器设置每帧带宽开销。

3.3 带宽开销预测器

在 Tambur 中，带宽开销是通过一个机器学习（ML）模型动态预测的，目的是在每一帧中分配校验符号的数量。这种带宽开销分配方案基于接收端观察到的数据包丢失模式，并通过一个预测模型来适配当前网络的状况。通过这种方法，Tambur 能够根据实时的网络条件调整带宽开销，使得冗余数据的发送更加灵活，从而在不增加解码失败风险的前提下尽可能减少带宽开销。例如，50% 的带宽开销意味着如果帧 i 包括 1000 字节，则分配 500 字节的奇偶校验符号。

丢失模式报告预测模型的输入是一个 “丢失模式报告”，由接收端定期生成（例如每两秒一次）。丢失模式报告包括 13 个计算得出的量度，这些量度可以在一次遍历丢失轨迹时线性计算得到。具体而言，丢失模式报告包含以下内容：

- 多帧突发性（Multi-frame burstiness）：跨多帧丢失的比率，衡量突发丢失的严重程度。
- 保护空间充分性（Guard space sufficiency）：在突发丢失后，丢失的帧是否被充足的无丢失帧所保护，以保证数据的恢复。
- 丢失比率：数据包丢失的比例，分别为帧级和包级。
- 连续丢失的平均数量：衡量突发丢失的长度，表示连续丢失的数据包数量。
- 保护空间的平均长度：衡量丢失后的保护空间的平均长度。
- 突发密度（Burst density）和间隙密度（Gap density）：分别衡量丢失发生时连续丢失的频率和丢失与恢复之间的空隙。

基于这些信息，带宽开销预测器将输出每帧应分配多少带宽开销。通过该机制，Tambur 可以灵活调整每帧的冗余数据量，从而在网络带宽紧张时减少冗余数据的传输，在网络条件较好的时候增加冗余数据的传输以提高丢失恢复的成功率。

4 复现细节

4.1 与已有开源代码对比

使用 Ringmaster [3] 作为仿真模拟平台，这是一个新的视频会议平台，可模拟一对一视频通话以对 FEC 方案进行基准测试。使用该论文的源代码 [4]，实现了 streaimg codes 方法，同时实现了传统的 Block-Within 和 Block-Multi 的 FEC 方法。使用 FFmpeg 库 [1]，实现基于 H.264 的视频编解码功能。

由于 Ringmaster 中使用的编解码器是 VP9，而在我未来的工作中主要涉及到 H.264 和 H.265 编解码器，因此首先使用了 ffmpeg 来修改编解码器部分，即通过使用 H.264 实现视频编解码功能。此外，由于该仿真模拟平台处理由丢包导致无法恢复的错误帧的逻辑是，将错误帧及其后续所有接收到的 P 帧舍弃，并从下一个正确接收的 I 帧开始解码播放，即错误帧及其之后所有接收到的 P 帧都不会被播放。这样不仅浪费了带宽，也会造成视频卡顿多个帧的时间。因此为了更好的在实际场景中运行，在这里实现了错误隐藏技术，通过时间和空间上的信息，对错误帧画面进行修复，使得后续 P 帧可以参考该帧继续解码播放。实现的错误隐藏技术是 [31]。

4.2 实验环境搭建

使用 C++ 实现了 Tambur，任何视频会议应用程序都可以使用该库。在发送方，Tambur 将连续的压缩帧作为输入并输出数据包和奇偶校验包。在接收方，Tambur 通过使用接收到的数据包的符号求解线性方程组来解码丢失的数据包。当数据包丢失时，我们将流代码的属性与开源最小割/最大流算法 [8] 相结合，以确定可以在可忽略的时间内使用哪些奇偶校验符号来解码哪些数据符号。然后通过求解最小满秩线性方程组来解码数据。

与视频会议 Ringmaster 集成。为了验证 Tambur 在现实世界中的有效性，将其与 Ringmaster 集成。其视频发送器以精确的帧速率（例如 30 fps）从磁盘上的输入 Y4M 视频文件读取原始帧，并使用 libvpx [2] 库中的 VP9 编码器使用与 WebRTC [5] 中类似的编解码器配置来压缩它们。用户提供的 FEC 方案为编码帧提供奇偶校验数据，该数据在打包后通过 UDP 发送到视频接收器。接收到帧后，视频接收器依次应用 FEC 解码器和 VP9 解码器来解码并渲染原始视频帧。此外，Ringmaster 允许请求新的关键帧，例如，当接收器由于数据包丢失过多而无法恢复视频帧时，从而请求发送器编码新的关键帧以恢复视频。在自动呼叫结束时，QoE 指标是通过聚合来自两个端点的日志来计算的，这些日志记录每个帧编码或解码时的时间戳，以及其帧 ID、大小、FEC 带宽开销等。

4.3 创新点

添加了通过 H.264 技术进行视频的编解码功能。实现了错误隐藏技术来恢复丢包导致无法正确解码的错误帧，使得错误帧后续接收到的帧也可以被解码播放，即使视频质量可能远不如实际视频。

5 实验结果分析

使用 mahimahi 来模拟真实的网络环境，分别测试了 Block-Within，Block-Multi，Tambur 和 Tambur-full-BW 四种方法。每种方法都测试了 20 个会议视频，最终的实验结果如图 6、7、8 所示。

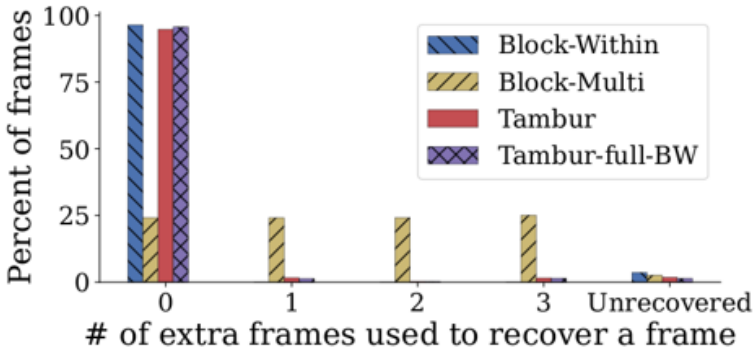
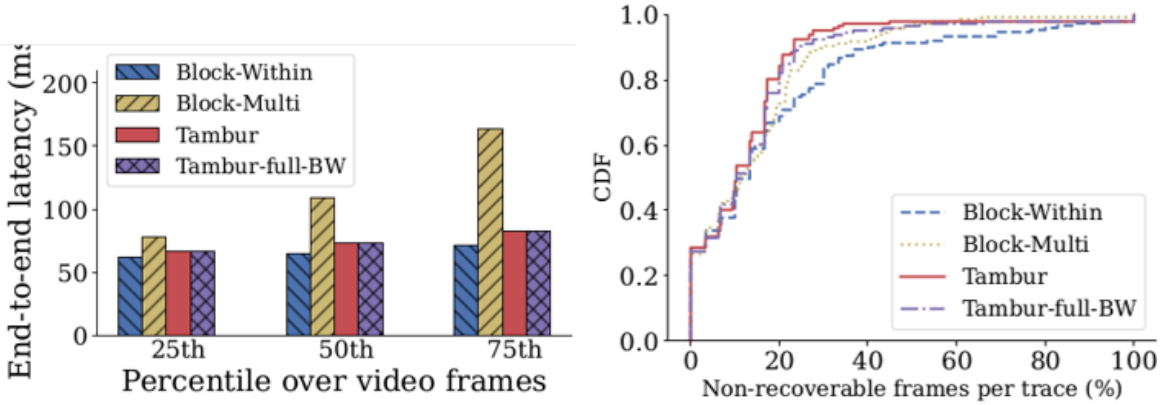


图 6. Tambur 在不使用额外帧的情况下恢复了几乎与 Block-Within 一样多的帧。

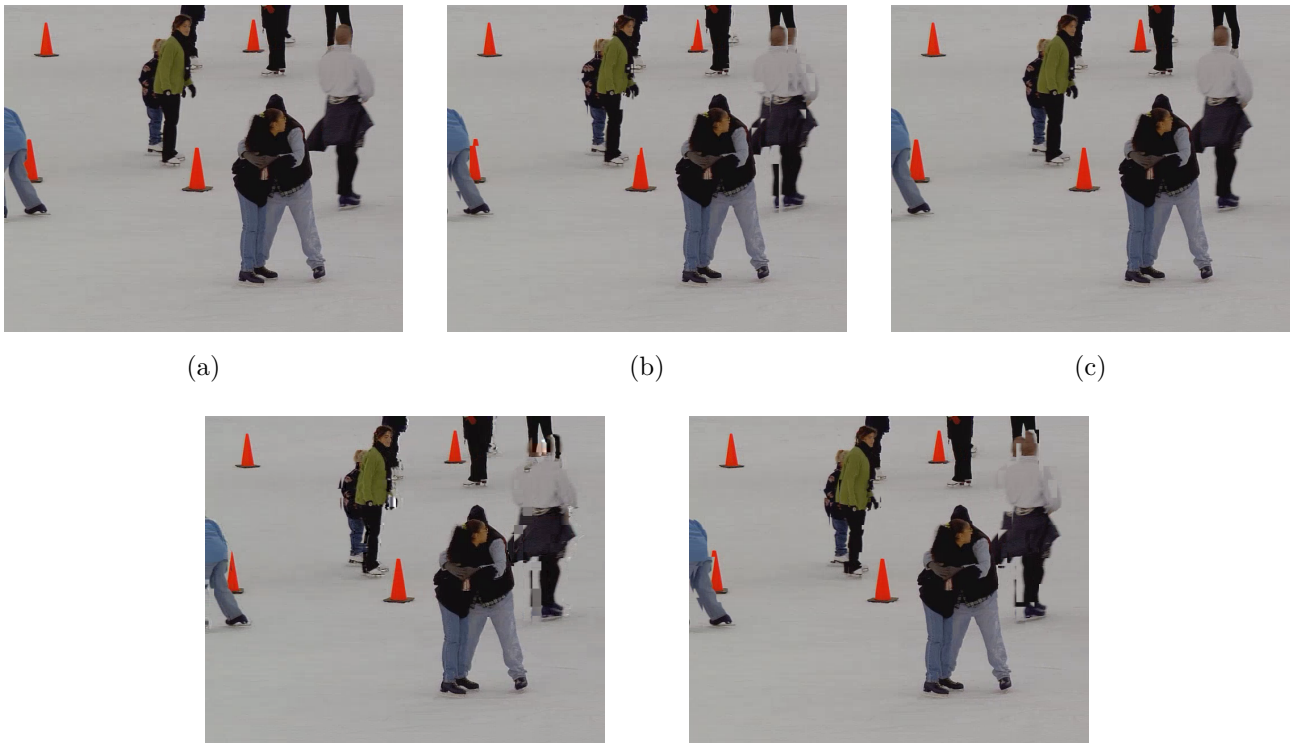
图 6 显示 Tambur 恢复的帧数几乎与 Block-Within 一样多，没有使用额外的帧，并且总体上恢复得更多。Block-Multi 使用 0、1、2 和 3 个额外帧恢复大致相同数量的帧。图 7（a）显示 Tambur 恢复时延比 Block-Within 多一点，显著低于 Block-Multi。且所有方案的端到端



(a) 渲染帧的延迟　　(b) 不可恢复的帧

图 7. Tambur 比 Block-Multi 渲染更多的帧，并且延迟更低。

延迟都在大约 150ms 的上限左右。由于编码/解码时间较短，并且始终使用同一帧的奇偶校验数据来恢复渲染帧，因此 Block-Within 的延迟略低。而在图 7（b）中，可以发现 Tambur 恢复的帧数比 Block-Within 和 Block-Multi 方法都多，因此我们认为 Tambur 的小成本（例如中位数额外花费 1.7 毫秒进行编码）是值得的，因为其余 QoE 指标都得到了实质性改进。



(a)　(b)　(c)

(d)　(e)

图 8. 恢复第 48 帧错误帧。(a) 原始帧；(b) 运动补偿帧 (38.37dB)；(c) 复制上一帧修复 (36.59dB)；(d)FFmpeg 默认修复方法 (37.13dB);(e) 使用 HMVE 修复 (37.90dB)

对于无法恢复的视频帧，采用 HMVE 方法进行错误隐藏。图 8 中 (a), (b), (c), (d), (e) 分别是原始图像，运动补偿后的图像，使用前一帧、ffmpeg 默认的错误隐藏算法以及 HMVE 错误隐藏方法得到的图像。其中运动补偿后的图像是准确获得运动矢量信息构造出的图像，认

为是错误隐藏算法的性能上限。比较 PSNR 值可以发现，使用 HMVE 方法可以更好的提高视频质量。

6 总结与展望

这项工作介绍了 Tambur——一种新的通信方案，用于视频会议的带宽高效丢失恢复，由两个组件组成。首先，一种新的流代码弥合了理论流代码和视频会议应用程序之间的差距，它将任何给定的带宽开销作为输入。其次，基于学习的预测模型来设置带宽开销。在我们对商业视频会议应用程序的大规模真实世界跟踪进行的评估中，Tambur 同时将不可恢复帧的频率和带宽开销分别降低了 26.5% 和 35.1%。我们还设计了第一个用于实施和评估 FEC 方案的视频会议框架。该框架通过提供一个简单的界面来合并（a）新的 FEC 方案和（b）新的基于学习的预测模型，从而可以轻松评估新通信方案的 QoE 优势。使用该框架，我们评估 Tambur 并显示 QoE 指标的改进，包括冻结减少 26%，非渲染帧减少 28%。这些好处使流代码成为恢复视频会议应用丢失数据包的可行解决方案。因此，结果也显示了流代码对于云游戏等其他直播应用程序的前景。

参考文献

[1] *FFmpeg. https://ffmpeg.org/.*

[2] *libvpx. https://chromium.googlesource.com/ webm/libvpx/.*

[3] *Ringmaster. https://github.com/microsoft/ringmaster.*

[4] *Tambur. https://github.com/Thesys-lab/tambur.*

[5] *WebRTC. https://webrtc.org/.*

[6] Ahmed Badr, Ashish Khisti, Wai-tian Tan, Xiaoqing Zhu, and John Apostolopoulos. Fec for voip using dual-delay streaming codes. In *IEEE INFOCOM 2017-IEEE Conference on Computer Communications*, pages 1–9. IEEE, 2017.

[7] Ahmed Badr, Pratik Patil, Ashish Khisti, Wai-Tian Tan, and John Apostolopoulos. Layered constructions for low-delay streaming codes. *IEEE Transactions on Information Theory*, 63(1):111–141, 2016.

[8] Yuri Boykov and Vladimir Kolmogorov. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *IEEE transactions on pattern analysis and machine intelligence*, 26(9):1124–1137, 2004.

[9] Hyunseok Chang, Matteo Varvello, Fang Hao, and Sarit Mukherjee. Can you see me now? a measurement study of zoom, webex, and meet. In *Proceedings of the 21st ACM Internet Measurement Conference*, pages 216–228, 2021.

[10] Ross Cutler, Yasaman Hosseinkashi, Jamie Pool, Senja Filipi, Robert Aichner, Yuan Tu, and Johannes Gehrke. Meeting effectiveness and inclusiveness in remote collaboration. *Proceedings of the ACM on Human-Computer Interaction*, 5(CSCW1):1–29, 2021.

[11] Elad Domanovitz, Silas L Fong, and Ashish Khisti. An explicit rate-optimal streaming code for channels with burst and arbitrary erasures. *IEEE Transactions on Information Theory*, 68(1):47–65, 2021.

[12] Damian Dudzicz, Silas L Fong, and Ashish Khisti. An explicit construction of optimal streaming codes for channels with burst and arbitrary erasures. *IEEE Transactions on Communications*, 68(1):12–25, 2019.

[13] Edwin O Elliott. Estimates of error rates for codes on burst-noise channels. *The Bell System Technical Journal*, 42(5):1977–1997, 1963.

[14] Salma Emara, Silas L Fong, Baochun Li, Ashish Khisti, Wai-Tian Tan, Xiaoqing Zhu, and John Apostolopoulos. Low-latency network-adaptive error control for interactive streaming. *IEEE Transactions on Multimedia*, 24:1691–1706, 2021.

[15] Silas L Fong, Ashish Khisti, Baochun Li, Wai-Tian Tan, Xiaoqing Zhu, and John Apostolopoulos. Optimal streaming codes for channels with burst and arbitrary erasures. *IEEE Transactions on Information Theory*, 65(7):4274–4292, 2019.

[16] Sadjad Fouladi, John Emmons, Emre Orbay, Catherine Wu, Riad S Wahby, and Keith Winstein. Salsify:{Low-Latency} network video through tighter integration between a video codec and a transport protocol. In *15th USENIX Symposium on Networked Systems Design and Implementation (NSDI 18)*, pages 267–282, 2018.

[17] Boni García, Micael Gallego, Francisco Gortázar, and Antonia Bertolino. Understanding and estimating quality of experience in webrtc applications. *Computing*, 101:1585–1607, 2019.

[18] Jim Gettys and Kathleen Nichols. Bufferbloat: dark buffers in the internet. *Communications of the ACM*, 55(1):57–65, 2012.

[19] Xianshang Lin, Yunfei Ma, Junshao Zhang, Yao Cui, Jing Li, Shi Bai, Ziyue Zhang, Dennis Cai, Hongqiang Harry Liu, and Ming Zhang. Gso-simulcast: global stream orchestration in simulcast video conferencing systems. In *Proceedings of the ACM SIGCOMM 2022 Conference*, pages 826–839, 2022.

[20] David JC MacKay. Fountain codes. *IEE Proceedings-Communications*, 152(6):1062–1068, 2005.

[21] Kyle MacMillan, Tarun Mangla, James Saxon, and Nick Feamster. Measuring the performance and network utilization of popular video conferencing applications. In *Proceedings of the 21st ACM Internet Measurement Conference*, pages 229–244, 2021.

[22] Emin Martinian and C-EW Sundberg. Burst erasure correction codes with low decoding delay. *IEEE Transactions on Information theory*, 50(10):2494–2502, 2004.

[23] Emin Martinian and Mitchell Trott. Delay-optimal burst erasure code construction. In *2007 IEEE International Symposium on Information Theory*, pages 1006–1010. IEEE, 2007.

[24] Anush Krishna Moorthy, Lark Kwon Choi, Alan Conrad Bovik, and Gustavo De Veciana. Video quality assessment on mobile devices: Subjective, behavioral and objective studies. *IEEE Journal of Selected Topics in Signal Processing*, 6(6):652–671, 2012.

[25] Péter Orosz, Tamás Skopkó, Zoltán Nagy, Pál Varga, and László Gyimóthi. A case study on correlating video qos and qoe. In *2014 IEEE Network Operations and Management Symposium (NOMS)*, pages 1–5. IEEE, 2014.

[26] Yining Qi and Mingyuan Dai. The effect of frame freezing and frame skipping on video quality. In *2006 international conference on intelligent information hiding and multimedia*, pages 423–426. IEEE, 2006.

[27] Irving S Reed and Gustave Solomon. Polynomial codes over certain finite fields. *Journal of the society for industrial and applied mathematics*, 8(2):300–304, 1960.

[28] Michael Rudow and KV Rashmi. Learning-augmented streaming codes are approximately optimal for variable-size messages. In *2022 IEEE International Symposium on Information Theory (ISIT)*, pages 474–479. IEEE, 2022.

[29] Michael Rudow and KV Rashmi. Streaming codes for variable-size messages. *IEEE Transactions on Information Theory*, 68(9):5823–5849, 2022.

[30] Justin Uberti. Webrtc forward error correction requirements. *Internet Engineering Task Force (IETF), document RFC*, 8854:10, 2021.

[31] Bo Yan and Hamid Gharavi. A hybrid frame concealment algorithm for h.264/avc. *IEEE Transactions on Image Processing*, 19(1):98–107, 2010.

[32] Francis Y Yan, Hudson Ayers, Chenzhi Zhu, Sadjad Fouladi, James Hong, Keyi Zhang, Philip Levis, and Keith Winstein. Learning in situ: a randomized experiment in video streaming. In *17th USENIX Symposium on Networked Systems Design and Implementation (NSDI 20)*, pages 495–511, 2020.

[33] Mo Zanaty, Varun Singh, A Begen, and Giridhar Mandyam. Rtp payload format for flexible forward error correction (fec). Technical report, 2019.