编号 ：

多 路 径 编 码 传 输 管 理 技 术 研 究 报 告

拟制单位：

拟制人： 陈泽山

审　核：

会　签：

标准化：

批　准：

公司名称

2025 年 4 月

目 录

1 概述

1.1 摘要

卫星互联网作为一种新兴的全球通信基础设施，在为传统地面网络难以覆盖的偏远地区和欠发达地区提供互联网接入方面发挥着越来越重要的作用。随着用户对高带宽和低延迟应用需求的不断增长，例如视频流、在线游戏和实时通信，卫星互联网面临着提升传输性能与可靠性的挑战。然而，卫星系统固有的特性（如较长的传播时延，特别是地球同步轨道卫星，以及轨道运动和环境因素引起的链路不稳定）限制了其服务质量。

天空地一体化网络的提出在一定程度上弥补了卫星互联网的不足，但由于其链路异构性强、动态变化频繁、传输环境复杂，传统的单路径传输机制在可靠性、时延和带宽利用率方面仍存在显著不足。多路径编码传输技术通过在多个路径间对数据进行冗余编码与分发，有效提升了抗链路波动与中断的能力，增强了传输的鲁棒性与可靠性；同时能够优化带宽利用率，实现并行传输和智能路径调度，降低整体时延，减少对传统重传机制的依赖。该技术尤其适用于卫星通信、空地协同、无人系统集群等高动态网络环境，是实现高效、稳定通信的重要手段。

多路径传输的实现固然能带来诸多益处，但也面临着若干显著的挑战：（1）数据包乱序问题。由于数据流被分割成多个数据包，并通过具有不同延迟和传输特性的多条路径进行传输，到达接收端的这些数据包的顺序可能与它们在发送端发出的顺序不一致。这种乱序到达会对上层协议的可靠传输机制带来挑战，需要接收端进行复杂的重排序处理。（2）网络管理复杂性。实现多路径传输需要在终端设备（发送端和接收端）上部署更为复杂的协议栈。这包括对多个网络接口的管理、数据流量在不同路径上的智能分流、最优路径的选择以及接收端对乱序到达的数据包进行高效重组等功能。这些额外的复杂性增加了终端设备的实现难度和管理负担。（3）中间网络兼容性。现有的网络基础设施中部署了大量的中间设备（例如防火墙、NAT、负载均衡器等），它们可能对多路径传输的数据流进行检查和处理，并可能引入兼容性问题。例如，中间设备可能会错误地丢弃来

自同一连接但在不同路径上到达的数据包。如何确保多路径传输能够与现有的网络基础设施良好地协同工作是一个重要的挑战。

针对上述挑战，本文旨在研究基于天空地一体网络的多路径编码传输技术，具体而言，研究内容包括：为应对异构网络环境下数据编码的复杂性以及由此导致的数据包乱序问题，提出了一种编码协议，通过生成冗余数据包来增强传输的鲁棒性，其核心优势在于，接收端无需关注单个数据包的到达顺序，仅需接收到足够数量的线性无关编码包即可成功解码恢复原。针对多路径传输中存在的网络异构性以及由此带来的管理复杂性，以及网络基础设施大量部署的中间设备造成的兼容性问题，提出一种网络传输协议，能够根据实时网络条件动态调整数据在多路径上的分配，优化吞吐量并增强传输的稳定性，从而在空天地一体网络的异构路径中实现高效的数据分发，另外协议基于 UDP 协议设计，有助于降低被中间设备（如状态防火墙）错误拦截或干扰的可能性，从而增强了在复杂网络环境下的兼容性。

1.2 术语和缩略语

术语	缩略语	释义
Multipath Transmission	-	多路径传输
Network Coding	NFV	网络编码
Forward Error Correction	FEC	前向纠错
Automatic Repeat Request	ARQ	自动重传请求
Coding Coefficient	-	编码系数
Load Balancing	-	负载均衡
Coding/Decoding	-	编码/解码
Encoding Vector	-	编码向量

2 研究依据和参考材料

《信息技术 传输协议多路径通信规范》（GB/T 38634-2020）

《IP 网络技术要求——网络传输性能》（YD/T 1171-2015）

《软件系统验收规范》（GB/T 28035-2011）

《云计算服务安全指南》（GB/T 31167-2014）

《基于天空地一体网络的云网融合技术研究及应用可行性研究报告》

3 国内外研究现状

异构网络多路径编码传输是通信技术的重要领域，旨在通过多种网络路径实现高效数据传输。中国和国际的研究现状显示，双方都在积极探索，但各有侧重。

3.1 国内研究现状

国内在多路径编码传输领域的研究以算法和协议创新为主，特别是在跨层优化和网络编码方面。例如 DBCCA 和 SmartCC 等算法旨在适应动态网络环境，优化异构网络的传输效率。

李贺武、吴建平等人[1]在《互联网端到端多路径传输跨层优化研究综述》论文中详细分析了多路径传输技术的发展，重点讨论了跨层优化在异构网络中的应用。论文指出，传统的传输层协议（如 TCP）在异构网络中面临链路动态变化、路径重叠等问题，传统的传输层信息不足以有效发挥多路径传输的优势，因此需要引入物理层、链路层、网络层和应用层的相关信息进行优化，端到端多路径传输跨层优化研究分类如图 1 所示。该综述还总结了近年来利用跨层信息进行多路径传输优化的研究，并对未来的研究趋势进行了展望。

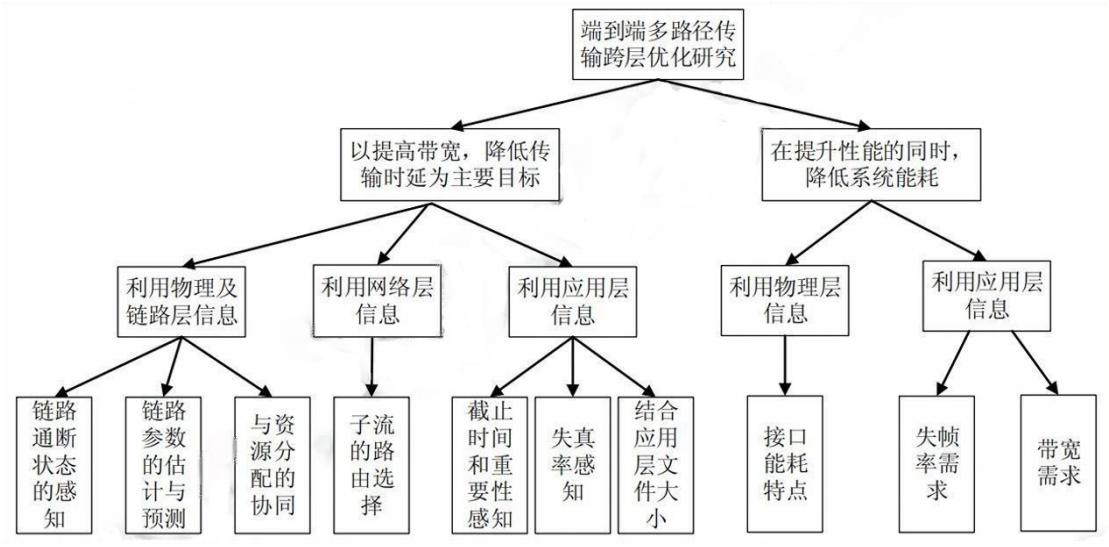

图 1 端到端多路径传输跨层优化研究分类

另外，我国研究人员提出了多种针对异构网络多路径传输的算法和协议。例如：

Li H 等人[2]提出了一种基于延迟的 MPTCP 拥塞控制算法（DBCCA）用于优化 MPTCP 的传输性能，总体架构如图 2 所示。所提出的算法主要包括两个步骤：

首先，构建一个约束优化问题，旨在最小化不同路径的延迟差异，并通过遗传算法获得一个近似最优的速率分配向量。其次，基于获得的速率分配向量和往返时间调整拥塞窗口，以便在每条路径上分配流量。DBCCA 联合考虑吞吐量和延迟，在异构网络环境中有效优化了移动流量的传输。

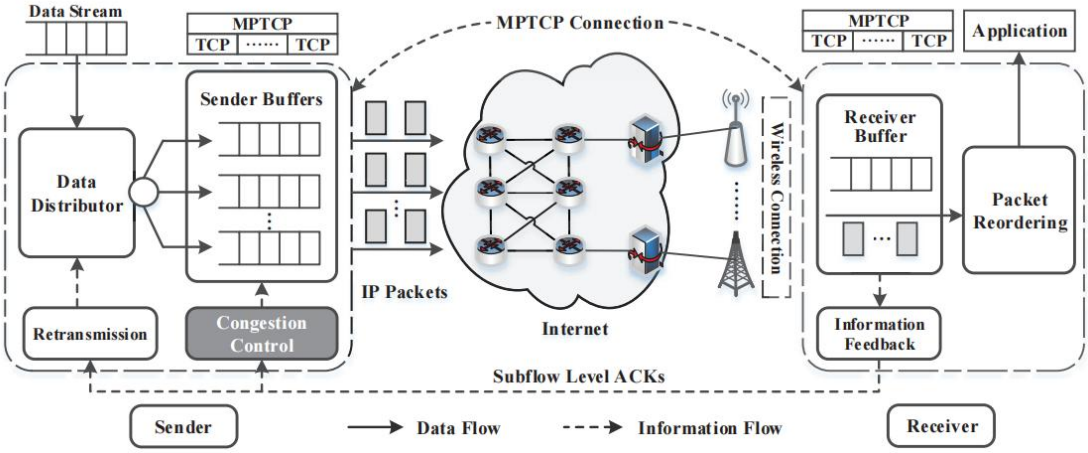

图 2 基于 MPTCP 的 DBCCA 传输协议

Li W 等人[3]提出了一种基于学习的多路径拥塞控制方法，称为 SmartCC，用于处理异构网络中多个通信路径的多样性。SmartCC 采用异步强化学习框架来学习一套拥塞控制规则，使得发送端能够观察网络环境并采取相应的行动，适应性地调整子流的拥塞窗口，以适应不同的网络情况。为了解决高维空间中状态无限的问题，SmartCC 提出了一种分层瓦片编码算法用于状态聚合，以及一种 Q 学习的函数估计方法，能够高效地推导出最优策略。由于 SmartCC 的异步设计，模型训练和执行过程是解耦的，学习过程不会对 MPTCP 拥塞控制中的决策过程引入额外的延迟和开销。

在具体应用领域，我国的研究也取得了进展。例如，在在车云多路径传输中，如何根据链路质量的变化动态调整编码速率，避免不必要的带宽资源消耗并提高网络吞吐量，成为了一个巨大的挑战。Yin C 等人[4]提出了一种自适应网络编码（ANC）方案，如图 3 所示，该方案通过将隐马尔可夫模型（HMM）与网络编码方案结合，能够根据估算的数据包丢失率（PLR）有效地调整编码速率。ANC 方案克服了无线链路质量的快速变化，最大化了吞吐量并减少了传输中的数据包丢失。在吞吐量性能方面，仿真和实际实验结果表明，ANC 方案在车载物联网系统中的车载无线多路径传输中，优于现有的最新网络编码方案。

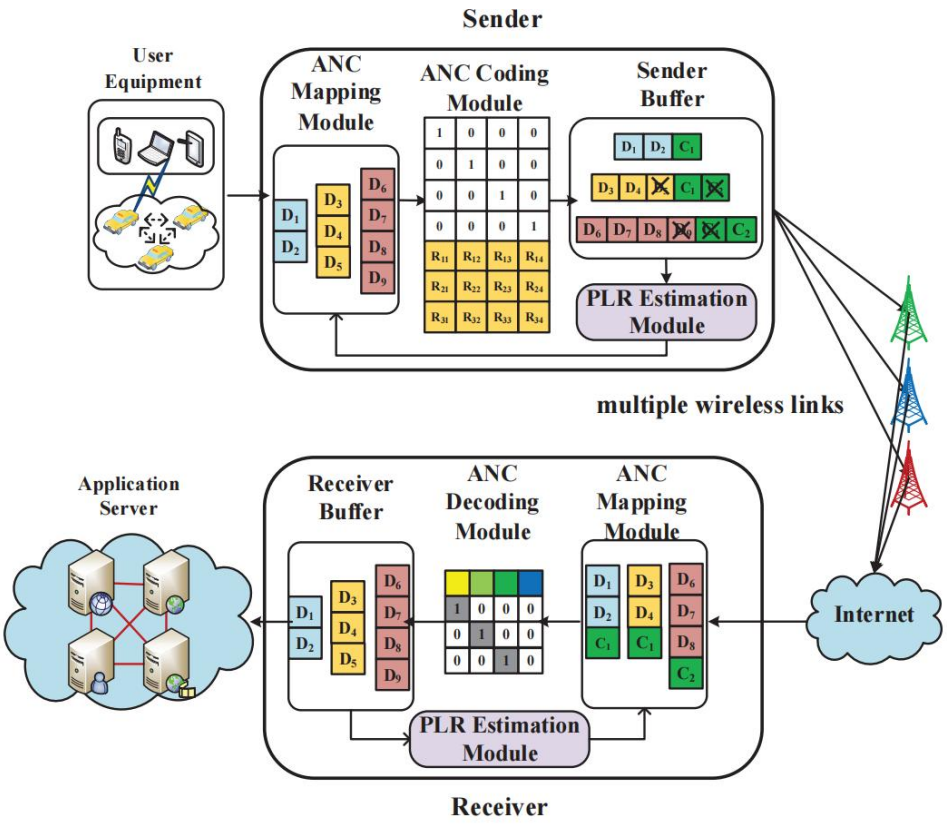

图 3 ANC 编码架构

此外，我国的研究还涉及到 SDN（软件定义网络）下的多路径路由算法，余翔等人[5]提出了一种结合蚁群算法的多路径路由方案，显著降低了数据流的时延和分组丢失率。在 5G 和 6G 网络中，中国的研究还关注异构网络融合共生，如 Ford A 等人[6]在《异构网络融合共生的需求、挑战与架构》中指出，在 5G 网络中，3GPP 引入的 ATSSS（访问流量引导、切换和分割）功能基于多路径 TCP（MPTCP），管理 4G/5G 和 Wi-Fi 网络的流量分流，体现了多路径在异构网络中的重要作用。

3.2 国外研究现状

国际上，IETF（互联网工程任务组）和 3GPP（第三代合作伙伴计划）等组织在多路径传输技术的标准化方面起着重要的推动作用。

3.2.1 IETF（互联网工程任务组）多路径传输标准化

IETF 是全球网络协议和标准制定的核心组织之一，它在多路径传输技术的标准化方面做了大量工作。特别是在多路径 TCP（MPTCP）[7]和 QUIC 协议[8]等方面，IETF 的研究和标准化工作为多路径传输技术的发展奠定了基础。

(1) MPTCP (Multipath TCP)

MPTCP 是一种在传统的 TCP 协议基础上扩展出来的协议，它允许在多个网络路径上同时发送数据，从而提高了带宽利用率、网络的容错性和延迟性能。

MPTCP 通过在多个路径上同时发送数据来提升传输性能，减少延迟并增强网络的鲁棒性。例如，用户设备可以同时使用 Wi-Fi 和 4G 连接，通过 MPTCP 将数据分布到这两条路径上，从而提高总体吞吐量和可靠性。

IETF 通过其 Multipath TCP Working Group (MPTCP WG) 对 MPTCP 协议的设计和规范进行了深入讨论。这个工作组的目标是创建一个稳定、可靠且具有互操作性的多路径传输标准。MPTCP 的标准化帮助了各大运营商和设备制造商在实际应用中实现多路径连接。IETF 发布了多个 RFC（请求评论）文档来规范 MPTCP，包括 RFC 6824（介绍 MPTCP 的基本原理）和 RFC 8684（描述 MPTCP 的流量调度与优化方法）。

MPTCP 特别适用于移动通信（例如 4G/5G 网络）、无线局域网（Wi-Fi）和其他异构网络环境。它可以在不同网络之间切换，提高用户体验，尤其是在网络状况不稳定或存在拥塞时。

(2) QUIC (Quick UDP Internet Connections)

QUIC 是一种由 Google 提出的基于 UDP 的传输协议，旨在提高 Web 应用的速度和安全性。IETF 在标准化 QUIC 方面的工作也涉及了多路径传输的应用。

QUIC 的多路径支持：QUIC 支持多路径连接，允许一个应用通过多个路径与服务器进行数据传输。这对于高带宽低延迟的应用（如视频流和游戏）非常有用，可以在网络出现拥塞或路径失效时，自动切换到其他路径，从而保证数据传输的稳定性和高效性。

QUIC 协议的多路径特性是 IETF 在其工作组内的一个重要研究方向，目的是提高 QUIC 协议在现代网络环境中的适应性和可靠性。IETF 的标准化工作为 QUIC 的部署和广泛应用提供了一个稳定的框架。

3.2.2 3GPP

3GPP 是全球主要的通信标准化组织之一，负责制定移动通信技术（如 LTE、5G 等）的标准。在多路径传输方面，3GPP[9]也在推进其标准化工作，特别是在 5G 网络和网络切片等先进通信技术中。

(1)5G 网络中的多路径支持

5G 网络是下一代移动通信网络，旨在提供更高的带宽、更低的延迟和更大的连接密度。多路径传输在 5G 网络中的应用尤为重要，尤其是在网络切片和边缘计算等技术的背景下。

5G 网络要求实现极低的延迟和高可靠性，这需要支持多个数据传输路径的技术。通过多路径传输，数据可以在多个路径上同时传输，这有助于避免单路径的瓶颈，提高网络的可靠性和带宽利用率。

3GPP 在制定 5G 标准时，特别强调了多路径传输在用户面（UP）和控制面（CP）中的应用。多路径传输不仅提升了终端的带宽，还能在不同网络环境下保持传输的稳定性。特别是在移动终端上，5G 网络允许终端通过同时使用 Wi-Fi、LTE 和 5G 等多个网络进行数据传输。

对于 NB-IoT（窄带物联网）等低功耗广域网技术，3GPP 也在标准化过程中考虑了多路径传输的优化，以提高物联网设备的连接质量和延迟性能。

(2)多路径在网络切片中的应用

网络切片是 5G 中的一项重要技术，允许在同一物理网络上创建多个虚拟网络，以满足不同的服务需求。通过多路径传输，网络切片能够在不同的路径上分发流量，提高网络的灵活性和效率。

在多路径传输的帮助下，网络切片可以在多个不同的传输路径上分布流量，确保即使部分路径失败，数据流仍然能够通过其他路径传输，保证服务的可靠性。

3.3 总结

国内外在多路径编码传输领域的研究均有进展。国内研究侧重于算法和协议创新，如 DBCCA 和 SmartCC 通过跨层优化和网络编码提升异构网络传输效率。研究人员还针对特定应用场景提出了自适应网络编码（ANC）和基于蚁群算法的多路径路由方案。此外，国内也关注 5G/6G 网络中异构网络融合共生，并认识到多路径传输的重要性。

国际上，IETF 和 3GPP 等组织积极推动多路径传输技术的标准化。IETF 通过 MPTCP 和 QUIC 协议的标准化，提升了带宽利用率、容错性和延迟性能。3GPP 则在 5G 网络和网络切片等技术中强调多路径传输的应用，以实现低延迟、高可靠的网络连接和灵活高效的资源分配。

4　技术设计

4.1　总体技术架构

多路径编码传输的核心目的是通过结合多路径传输与网络编码技术，解决传统单路径或简单多路径传输中的固有缺陷，从而在复杂网络环境中实现高可靠性、高效率、低延迟的数据传输。多路径编码传输管理总体技术架构如图 5 所示。

数据编码协议负责将应用层原始数据转换为适合网络传输的编码形式，增强抗丢包能力，其中利用弹性编码窗口，动态调整参与编码的源符号数量，根据网络状态（如丢包率）增减冗余，平衡效率与可靠性。

多路径传输协议负责管理多条传输路径（子流）的协同工作，优化资源利用，其中流量控制模块根据接收端处理能力调整发送速率，防止缓冲区溢出；丢包检测与重传模块通过 ACK/NACK 机制识别丢包，仅在必要时重传关键数据（如未达到解码阈值的编码符号）；拥塞控制模块监测路径拥塞程度（如延迟、丢包率），动态分配负载，避免单一链路过载；数据包调度模块智能分配编码符号到不同子流，优先使用高质量路径（如低延迟、高带宽）。

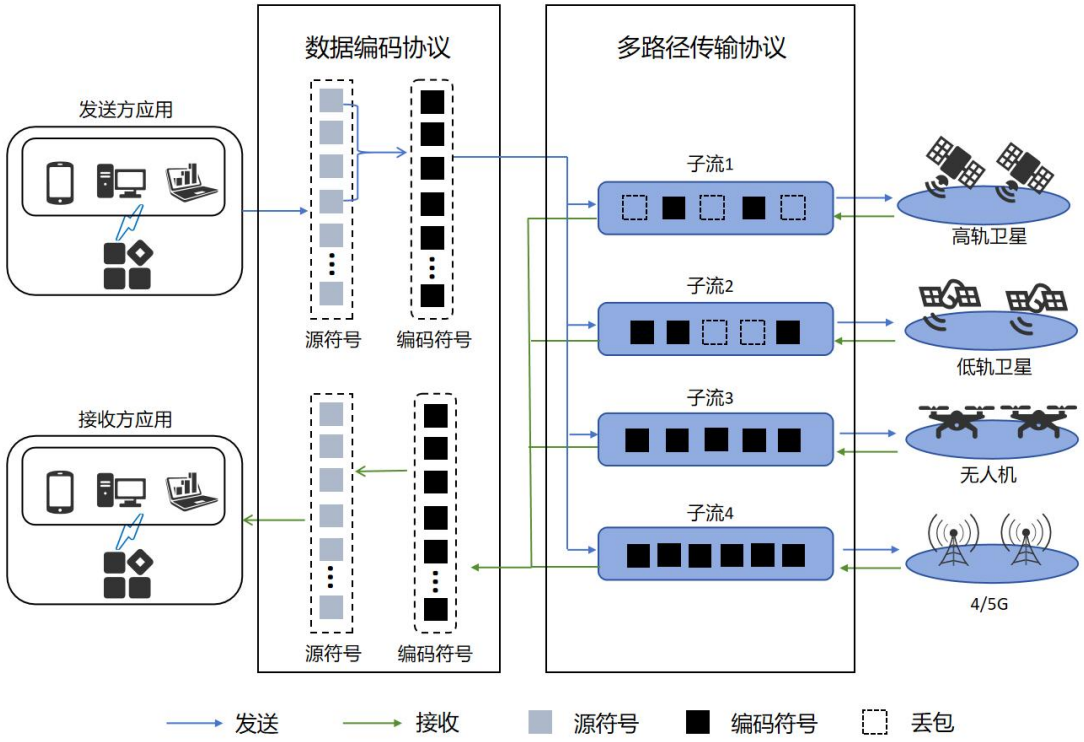

图 5 多路径编码传输管理总体技术架构

多路径编码传输工作流程大致可以分为三个阶段：

（1）编码阶段

应用数据被分割为源符号，通过弹性编码窗口动态生成编码符号，编码符号附带编码向量系数，供接收端解码使用。

（2）多路径传输

多路径传输协议根据路径质量（如带宽、延迟）和编码需求，将编码符号分配到不同子流（如子流 1~4）。通过拥塞控制和数据包调度，避免劣质路径成为瓶颈。

（3）接收端处理

收集来自多条路径的编码符号，利用编码向量构建线性方程组。当收到足够数量的线性无关编码符号时，解码恢复原始源符号。

4.2 数据编码协议

不同路径可能延迟不同、丢包不同、带宽不同，编码能有效缓解带来的乱序和不同步问题。数据编码协议基于 Tetrys 网络编码技术设计，具备高效利用多路径资源、卓越的容错恢复能力以及动态适应性等突出优势，特别适用于卫星互联网的多路径传输环境。

该编码方式的核心功能包括以下几个关键方面，这些功能协同作用，确保协议具备高度的稳健性：（1）实时编码：通过动态生成编码符号，适应网络条件的实时变化；（2）解码：在接收端有效恢复丢失的源符号；（3）信令：传输编码窗口中的符号 ID 及相关编码系数（在适用时）；（4）反馈管理：处理解码器反馈信息，以优化传输效率；（5）弹性窗口管理：动态调整编码窗口，提升整体性能。

编码协议的逻辑框架如图 4 所示，应用（发送方）将源数据分块形成源符号（source symbols），并将其输入到编码器（Encoder）。编码器基于弹性编码窗口内的源符号，生成编码符号（coded symbols）以及对应的编码向量（encoding vector），其中编码向量包含源符号 ID 和用于线性组合的系数。编码器将这些编码符号及编码向量打包为输出数据包（output packets），通过网络（包括卫星互联网与地面网络多路径）发送至接收端。接收端将接收到的编码数据包输入解码

器（Decoder），解码器利用编码向量中的信息，通过线性代数方法（如高斯消元法）恢复源符号。解码成功后，解码器向编码器发送反馈数据包（feedback packets），通知已解码的源符号 ID，以帮助编码器更新其弹性编码窗口并优化后续编码策略。同时，解码器将恢复的源数据传递给接收端的应用（接收方），完成数据传输过程。

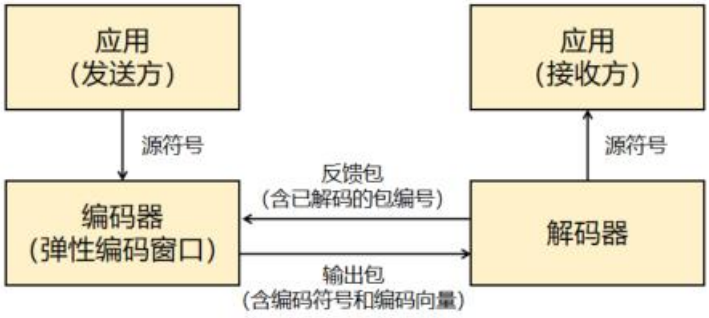

图 4 编码协议架构图

4.2.1 编码

在编码协议中，编码是数据传输的核心过程，旨在通过添加冗余信息来提高数据的可靠性和抗丢包能力。编码过程由编码器执行，基于源数据生成编码符号，并通过网络发送给接收端。具体编码采用线性网络编码思想，结合弹性编码窗口机制，允许动态调整编码策略以适应网络变化。

4.2.1.1 编码步骤

（1）源符号的准备

输入数据：编码器从应用层接收源数据（例如视频或音频流），这些数据被分割成固定或可变大小的单元，称为源符号（source symbols）。在本编码协议中，源符号大小可以是可变的，这为多媒体流提供了灵活性。

标识符分配：每个源符号被分配一个唯一的源符号 ID（source symbol ID），通常是一个序列号，用于在编码和解码过程中追踪数据。

初始窗口填充：源符号被存入弹性编码窗口（elastic encoding window），它是一个动态缓冲区，用于存储待编码的源符号集。窗口大小由编码器设置上限，以控制计算复杂度和内存使用。

（2）初始码率选择与动态调整

在传输开始时，编码器无法得知信道的丢包率，因此会选择一个初始码率（code rate），即编码符号与源符号的比例（例如 1/2，表示每生成两个编码符

号对应一个源符号）。较高的初始码率引入更多冗余，以提高抗丢包能力。

传输一定数量包后，编码器会根据解码反馈或信道估计动态调整码率，确保在不同丢包率下达到最佳传输效率和可靠性。

（3）编码向量的构造

编码向量（Encoding Vector）是定义线性组合规则的核心元数据，其本质是一个由编码系数构成的序列，用于明确编码符号与源符号之间的数学关系。构造过程如下：

编码符号是通过源符号的线性组合生成的。公式如下：

$$S_c = \sum_{i=1}^{n} c_i \cdot S_i \tag{1}$$

其中：

S_c 是生成的编码符号（coded symbol）。

S_i 是弹性编码窗口中的第 i 个源符号。

C_i 是对应的编码系数（coding coefficient）。系数由编码系数生成器（CCGI 以确定性方式生成，基于有限域（如 GF(2^8)）中的运算。协议支持基于 Vandermonde 矩阵的系数生成，例如：

$$c_i = \alpha^{(source_symbol_id \oplus coded_symbol_id) \bmod 256} \tag{2}$$

其中α是有限域的原根，⊕表示模运算，确保系数唯一且可预测。

n 是窗口中当前使用的源符号数量。

所有系数 $(c_1, c_2, ..., c_n)$ 构成编码向量（encoding vector），用于定义线性组合的规则。该编码向量与对应的编码符号一同发送至接收端，解码器据此信息恢复原始源数据。

（4）编码符号生成

编码器从弹性编码窗口中选择一组未被接收端确认的源符号，应用编码向量中的系数进行加权求和。假设窗口中有源符号 S_1, S_2, S_3，系数为 c_1, c_2, c_3，则编码符号计算为：

$$S_c = c_1 \cdot S_1 + c_2 \cdot S_2 + c_3 \cdot S_3 \tag{3}$$

在有限域中，加法通常是异或（XOR）运算，乘法遵循域的规则。

生成编码符号后，如果窗口达到最大限制，编码器移除最旧的源符号（通常是已确认的符号），并添加新的源符号，确保窗口保持动态平衡。

（5）反馈与优化

接收端的解码器会定期发送反馈信息，报告已成功解码的源符号 ID。发送端根据这些反馈数据从编码窗口中移除相应的符号。如果反馈表明丢包率较高，编码器会适当增加码率（生成更多编码符号）以增强容错能力；若网络状况有所改善，则降低码率以提升传输效率。

4.2.2 解码

解码过程旨在从接收到的编码符号中恢复丢失的源符号。解码部分也是是协议的核心环节，依赖于线性代数的求解方法，并结合协议特有的弹性编码窗口和反馈机制来提升效率和适应性。

4.2.2.1 解码步骤

（1）接收编码符号

解码器从网络接收到的每个编码包包含以下关键信息：

编码符号（Coded Symbol）：源符号经过线性组合后的结果。

编码向量（Encoding Vector）：一个向量，记录了生成该编码符号时所使用的源符号 ID 及其对应的系数。

编码符号 ID（Coded Symbol ID）：用于标识该编码符号的唯一编号。

解码器通过分析编码向量，确定每个编码符号是由哪些源符号组合而成的，以及每个源符号的系数是多少。例如，一个编码向量可能是[1, 0, 2]，表示该编码符号由源符号为$C_1 = 1 \cdot S_1 + 0 \cdot S_2 + 2 \cdot S_3$。

（2）构建线性方程组

每个接收到的编码符号可以写成一个线性方程。例如，假设接收到一个编码符号S_{c_j}和其编码向量$[c_{j,1}, c_{j,2}, ..., c_{j,n}]$，则：

$$S_{c_j} = c_{j,1} \cdot S_1 + c_{j,2} \cdot S_2 + \cdots + c_{j,n} \cdot S_n \tag{4}$$

其中：S_{c_j}是接收到的编码符号；$c_{j,1}$是编码向量中第 i 个源符号的系数；S_i是第个源符号（待求解的未知数）。

对接收到的多个编码符号，进一步用矩阵形式表示。设接收到 m 个编码符号，涉及 n 个源符号，解码器会将所有方程组织成矩阵形式：

$$A \cdot \mathbf{s} = \mathbf{b} \tag{5}$$

其中，A 是一个 $m \times n$ 的系数矩阵，由所有编码向量组成；**s** 是一个 n 维向量，表示 n 个待求解源符；b 是一个 m 维向量，表示接收到的 m 个编码符号 $[S_{c_1}, S_{c_2}, ..., S_{c_m}]$。

（3）检查解码条件

要恢复所有 n 个源符号，系数矩阵 A 的秩（rank）必须等于 n，即：

$$\text{rank}(A) = n \tag{6}$$

这意味着解码器需要接收至少 n 个线性无关的编码符号。如果 $m < n$ 或接收到的编码符号之间存在线性相关，则无法解码所有源符号;如果 $\text{rank}(A) < n$，解码器只能恢复部分源符号，或者需要等待更多编码符号到达以补充信息。

(4) 执行解码计算

解码器使用高斯消元法（Gaussian Elimination）求解线性方程组。具体步骤如下：首先，通过行变换（如加减乘除操作），将系数矩阵 A 转化为上三角矩阵或行阶梯形式。随后，从矩阵的最后一行开始，逐步回代计算每个源符号的值。

高斯消元法的复杂度为 $O(n^3)$，因此源符号数量 n（即编码窗口大小）对性能有较大影响。

编码协议支持弹性编码窗口，解码器可以随着编码符号的逐步到达，动态更新矩 A 和向量 b，不必等待所有数据到达再解码。

（5）反馈管理

解码成功后，解码器会向编码器发送反馈信息，通知已成功恢复的源符号（通过其 ID 标识），编码器根据反馈调整弹性编码窗口，移除已确认的源符号，仅对尚未解码的源符号继续生成编码符号。反馈机制有效减少了冗余数据传输，提升了传输效率。

4.2.3 弹性编码窗口

弹性编码窗口是编码协议中用于存储和管理待编码源符号的动态缓冲区，相较于传统的固定大小编码窗口，弹性编码窗口的大小和内容可根据网络状况、接

收端反馈及传输需求动态调整，其“弹性”表现为窗口能够扩展或收缩源符号数量，以优化编码过程。该窗口的运作基于动态反馈机制，其具体工作步骤如下：

（1）窗口的初始化

数据传输启动时，编码器将初始源符号（如S_1, S_2, S_3）存入弹性编码窗口。初始窗口大小可根据应用需求或网络条件预设（如 3 个符号），以平衡计算复杂度和传输效率。

(2)窗口动态更新

添加新符号：应用层生成新源符号（如S_4）时，编码器将其加入窗口，扩展窗口内容（如S_1, S_2, S_3变为S_1, S_2, S_3, S_4）。

移除已确认符号：接收端（解码器）会定期发送反馈，通知编码器哪些源符号已成功接收或解码（例如 S1）。编码器根据反馈从窗口中移除这些符号，释放空间（如更新为S_2, S_3, S_4）。

（3）窗口大小控制

为避免计算复杂度过高，编码器会设置窗口的最大容量（例如最多存储 10 个符号），具体大小取决于系统的计算能力和内存资源。

窗口大小会根据网络状况动态变化。在高丢包率场景下，编码器可能扩大窗口，增加更多源符号以生成更多编码符号，从而提高冗余度和可靠性；在网络稳定时，窗口可缩小，减少冗余以提升传输效率。

4.3 多路径传输协议

多路径编码协议对数据进行冗余编码，增强了数据在传输过程中的鲁棒性与可靠性。编码后的数据包如何在多条异构路径之间高效地调度和分发编码数据，以进一步提升传输性能，则成为设计中的另一核心挑战。本节提出多路径传输协议，协议基于 MPQUIC 协议，提供的多路径调度与拥塞控制机制能够灵活适配路径质量，按流粒度管理各子路径，可以解决路径异质性带来的数据包乱序积压与接收端阻塞问题，实现跨路径的编码包调度与灵活接收，有效提升整体吞吐量与实时性。

4.3.1 子流建立

在多路径传输协议中，**子流（Subflow）**是实现多路径传输的基本单位。协议允许连接在多个路径（例如 Wi-Fi 和蜂窝网络）上并发地建立多个子流，从而提升整体吞吐量、可靠性与抗干扰能力。

子流建立通过基于公告包（announcement packet）的机制，该机制通过设置特定标志位，引导接收端识别新子流，具体步骤为：

（1）当一端（例如客户端）希望将一个新的 IP 地址和端口添加至连接中时，会从该地址发送一个设置了 MultipathFlag 的公告包；

（2）接收方通过包中的 ConnectionID 将其与已存在的子流进行匹配：若该包来源地址未匹配、但 ConnectionID 有效匹配且未设置 MultipathFlag，则将被视为一次连接迁移尝试，而不是多路径扩展；若 MultipathFlag 设置正确，接收方则将该路径识别为新的子流，并允许立即使用该子流发送数据。

子流的建立并非完全对称。协议遵循以下原则：

发起方（发送公告包的一方）：在未收到来自该子流的数据前，不得使用该子流发送数据，确保路径的有效性与连通性；

接收方：可以在接收到公告包之后，立即使用该子流进行数据传输，也就是握手完成后即可发送数据的机制保持一致；

图 6 展示了多路径传输协议中子流建立（Subflow Establishment）的时序过程，该过程是整个多路径连接建立机制的重要部分。可以将其分为两个阶段：左侧是初始主连接的建立，右侧是新子流的添加。

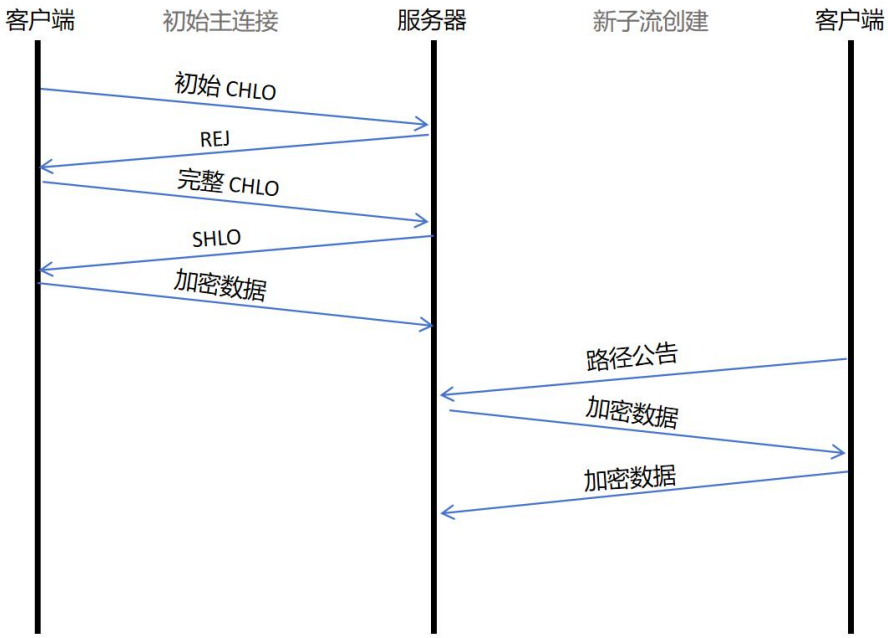

图 6 子流建立时序过程

初始主连接建立过程如下：

（1）初始 CHLO（Client Hello）:客户端向服务器发起连接请求，发送一个初始的 CHLO 消息；此时客户端可能缺少服务器的配置信息（如证书、公钥等），因此为“未完成”。

（2）REJ（Reject）：服务器拒绝该请求，并返回包含其配置、证书链和 Diffie-Hellman 参数的 REJ 消息。

（3）完整 CHLO：客户端收到 REJ 后，基于服务器信息构造一个完整的 CHLO 消息，并重新发出；该消息中包含协商出的密钥信息。

（4）SHLO（Server Hello）：服务器确认并返回 SHLO，表示握手完成，双方此时就拥有了共享密钥。

（5）加密数据：握手完成后，即可进行加密数据的传输。

额外新子流的创建过程如下：

（1）Path Announcement（路径通告）：客户端或服务器发起路径通告（表示发现了新路径），它通过主连接的信令机制传输。

（2）加密数据：一旦路径被确认有效，新子流可被正式启用，接下来就可以在该子流上传输加密数据。

如图 7 所示，新子流建立需经历以下状态变化，确保路径双向可用性。

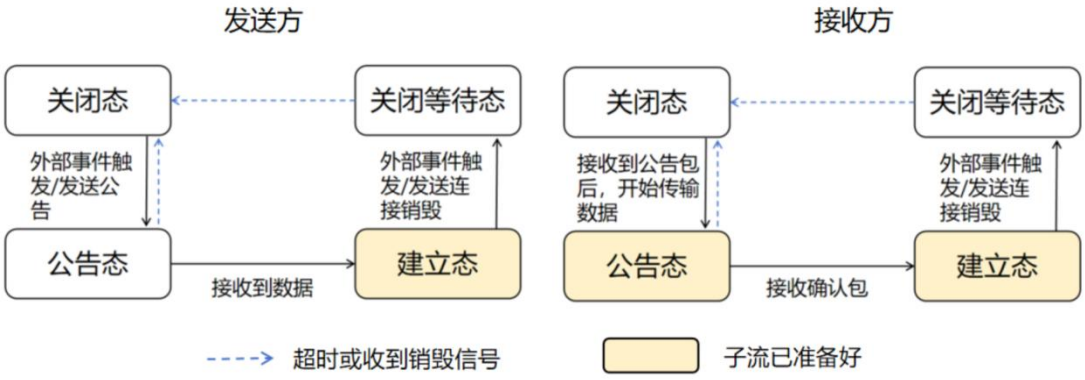

图 7 子流建立状态编号

关闭态：初始状态，子流尚未建立，连接方未采取任何操作。

发起方关闭态→公告态：外部事件触发，例如检测到新网络接口，发起方向目标地址发送**公告包**。

发起方公告态→建立态：发起方只有在收到来自该子流的数据后，才会确认此子流建立成功，进入建立状态，才可正式用于数据传输。

接收方关闭态→公告态：接收到公告包后，可以立刻使用这个子流发送数据，**无需等待确认**。

接收方公告态→建立态：接收到来自发起方的确认或应答包，子流正式建立，被确认可用。

4.3.2 数据包编号与流偏移

在多路径传输协议中，数据包通过不同路径传输，路径之间的延迟和丢包率差异会导致数据包乱序到达。而数据包编号可以帮助多路径传输协议区分不同路径上的包，检测丢失，进而依据丢包信号辅助流量控制，拥塞控制及数据包调度，协调多路径的传输。

数据包编号用于标识每一个被发送的数据包，它在每条连接中单调递增，有助于区分新发送的数据包与重传的数据包，并为接收方的 ACK 确认机制和发送方的丢包检测提供依据。

流偏移用于标识某段数据在其所属的逻辑流（stream）中的具体位置，从而使得接收方即使在收到乱序或重传的数据包时，也能够正确地将其插入到原始数据流中，保证数据的顺序和完整性。

多路径传输协议在继承上述机制的基础上，进一步进行了扩展，以适应多路径环境下的传输需求和复杂性：

（1）在多路径环境中，不同子流之间可能具有显著不同的 RTT、带宽或丢包率，为了防止数据包编号混淆，多路径传输协议为每条子流分别维护独立的 Packet Number 空间。使得：每条子流可以独立地进行拥塞控制和丢包检测；避免了在高 RTT 路径上发送的旧包被误判为重传包，保证了在多子流环境下丢包检测的准确性与高效性，从而提升整体传输的可靠性。

（2）尽管每条子流的数据包编号是独立的，多路径传输协议仍然使用全局唯一的流偏移值来标识流数据的位置。也就是说，同一个逻辑流上的数据，即便分散在不同子流中传输，其偏移值仍然连续并可拼接。该设计带来了以下优势：

（1）支持跨路径的数据包调度与灵活重传，丢失的数据可在任意子流上重新发送；

（2）保证接收端能够统一管理逻辑流的数据拼装，简化了流重组逻辑；

（3）避免了路径差异导致的数据乱序引起的队头阻塞（Head-of-Line Blocking）问题。

综上所述，通过在不同子流中引入独立的数据包编号空间，并保持全局一致的流偏移机制，实现了灵活可靠的数据调度与重传策略，显著提升了在多路径环境下的传输效率与鲁棒性。

4.3.3 控制数据包注入

在多路径传输协议中，数据包的注入（从用户空间将数据调度并传输到不同子流上）受到多项机制的控制与协调，主要包括以下三个核心组成部分：（1）流量控制；（2）拥塞控制；（3）数据包调度器。

4.3.3.1 流量控制

多路径传输协议基于 QUIC 的双层流量控制机制，防止接收端缓冲区溢出：（1）流级流量控：限制每个逻辑流在未收到接收方窗口更新前，最多能发送的数据量；（2）连接级别流量控：限制整个 QUIC 连接上在途（in- flight）数据的总量。

为了在多路径环境下提高利用率并保持低延迟，对这两种流量控制参数都按子流维护，并动态根据每条子流的 RTT 统计来调整：

窗口更新和发送间均基于该子流的最小平滑 RT 来计算，即 RTT 越小，计算出的窗口偏移量越小、更新间隔越短，意味着更频繁地发送窗口更新，从而在快

速路径上更及时地释放接收方缓冲区并提升带宽利用；对于 RTT 较大的路径，则以较大的偏移量和较长的间隔进行更新，避免过早耗尽接收端缓冲区。

通过这种 RTT 感知的动态调整，多路径传输协议能在异构子流间平衡响应速度与缓冲安全，既发挥低延迟路径的优势，也防止慢路径造成意外拥塞或缓冲溢出。

4.3.3.2 拥塞控制

为了适应不同路径的网络特性（如带宽、RTT、丢包率等），多路径传输协议在每条子流上均实施独立的拥塞控制：

独立拥塞窗口：每个子流维护自己专属的拥塞窗口（Congestion Window），并根据该路径的实时网络反馈（RTT、丢包信号、带宽估计等）动态地增减窗口大小，从而灵活地控制数据注入速率；

路径感知速率调整：对于 RTT 较小或丢包率较低的子流，拥塞控制算法会更积极地扩大窗口，以充分利用快速路径带宽；而对于慢速或不稳定路径，则会相应地收缩窗口，以避免引发过度重传或排队延迟。

4.3.3.3 数据包调度器

数据包调度器是整个数据注入控制的核心模块：负责从所有逻辑流中提取待发送数据，组合成数据包，具体工作流程如下：

（1）数据提取与打包

调度器从各逻辑的发送队列中，按优先级和可用数据量提取待发送的，将这些帧组合成 UDP 数据报包，同时为每个包分配一个子流标识。

（2）子流选择

在子流层面，调度器会检查每个子流的拥塞窗口和流量控制窗是否有剩余空间，结合子流当前的 RTT、丢包率和带宽估计，判断该子流是否符合该包的发送要求，仅在满足“窗口未被耗尽、路径性能稳定”条件时，才将包注入对应子流。

（3）统一全局调度

采用单一的调度实而非分散的子流调度，使调度策略可以在全连接范围内做决策，可根据流的延迟敏感度（如控制信令 VS 大文件传输）调整调度优先级。

优先选用 RTT 最小的子流，提供接口可插入其它调度算法，如带宽感知、流分类（视频、语音、后台同步等）或成本优化（如节省移动数据流量），支持

根据应用场景或网络条件热插拔不同的策略模块，保证异构在多变环境下的灵活适应性。

（4）灵活的跨子流重传

丢包重传时，调度器不局限于原始丢包所在的子流，可将需重传的数据重新打包，与新数据一起分配到当前性能最佳或拥塞最轻的子流上，减少空洞等待和不必要的延迟。

4.3.4 加密与认证

除了握手和重置包之外，协议始终为其通信提供加密和认证。这一设计确保了数据隐私和完整性，防止数据在传输过程中被篡改，同时也对抗了中间设备（如防火墙和代理）对协议的干扰。通过对包头（header）进行认证，能有效防止中间设备对连接状态进行非授权的修改，从而避免了传统协议中的协议固化问题，即协议在长期使用中因不允许变化而无法适应新需求。

加密与认证的具体流程：

（1）当客户端首次连接到服务器时，双方通过握手交换加密密钥，并利用对称加密算法加密数据包中的负载与头部信息。

（2）在握手过程中，客户端与服务器会交换加密密钥，这使得后续的连接可以在零往返时间内使用加密数据进行传输。

（3）对于后续的连接，客户端可以使用先前获取的密钥信息建立连接，在连接建立时直接发送 0-RTT 数据。

5 验证方案

本方案验证所提出的多路径编码传输协议特别是在丢包恢复、延迟、吞吐量和鲁棒性等方面的性能提升。

5.1 多路径传输性能测试

通过设置不同网络路径的质量，测试多路径编码传输协议在吞吐量、延迟、丢包率等方面的对比。

步骤：

（1）配置两个节点，分别作为发送端和接收端，并在中间使用两个独立路径（路径 A 和路径 B），每条路径的带宽、丢包率和延迟可独立配置。

（2）使用 iperf3 进行吞吐量测试，同时使用 ping 测量延迟。

评价指标：

吞吐量：计算每个协议的总吞吐量。

延迟：测量不同协议的端到端延迟。

丢包率：统计丢包数据，验证丢包恢复效果。

连接的建立和关闭时间：测量连接的时延和稳定性。

5.2 不同时延和丢包率下的恢复性能

模拟不同的网络环境（高丢包、高延迟）下，验证多路径传输协议对丢包的恢复能力。

步骤：

（1）设定两个路径，分别使用高延迟路径（如 200ms）和高丢包路径（如 20%丢包）。

（2）配置每个路径的网络条件，并通过仿真工具 ns-3 或 Mininet 设置不同的网络负载。

（3）在每个路径上分别使用多路径传输协议进行数据传输，并记录每个协议下的传输延迟、丢包率和重传情况。

评价指标：

丢包恢复时间：使用 Wireshark 观察多路径编码传输协议在丢包情况下的恢复时间。

吞吐量变化：记录协议在不同丢包和延迟情况下的吞吐量波动。

稳定性：观察协议在恶劣网络环境下的传输稳定性（例如，在路径切换、丢包发生时的行为）。

5.3 实际应用场景测试

模拟视频流传输或文件传输场景，验证多路径传输协议在实时数据传输中的优势。

步骤：

（1）在发送端播放视频流或传输大文件。

（2）配置网络条件为实时视频传输环境，保证低延迟和高吞吐量。

（3）使用 FFmpeg 或 VLC 播放视频流，或通过 iperf3 传输大文件。

性能指标：

视频质量：通过 PSNR（峰值信噪比）或 MOS（Mean Opinion Score）测量视频质量。

延迟：实时视频或文件传输的端到端延迟。

丢包率：数据传输过程中丢包情况。

6.应用前景展望

多路径编码传输在多个领域展现出了巨大的应用潜力，以下是几个主要应用场景的详细分析：

在车联网和自动驾驶方面，由于车联网需要处理大量的实时数据，如车辆位置、速度和周围环境信息，以支持自动驾驶和车辆协作。多路径编码传输通过提高数据传输的可靠性和速度，能够满足这些高要求场景的需求。

在视频传输和流媒体方面，由于高清视频传输对带宽和稳定性要求极高，尤其是在无线网络环境中。多路径编码传输通过将视频数据分发到多个路径上，可以有效应对网络波动，确保视频流的连续性和高质量。

在无线传感器网络方面，由于节点能量有限且网络拓扑动态变化，数据传输的可靠性和实时性至关重要。多路径编码传输通过结合网络编码技术，可以显著提高数据收集的成功率和能量效率。

综上所述，多路径编码传输作为一种前沿的网络通信技术，通过结合多路径传输和编码技术，为现代通信需求提供了高效、可靠和安全的解决方案。其在无线传感器网络、物联网、视频流媒体、车联网以及卫星通信等领域的应用潜力巨大。尽管面临数据包重排序、编码优化和标准化等挑战，但随着 5G、边缘计算和人工智能等技术的发展，多路径编码传输有望成为下一代网络通信的核心技术之一，为智能化、实时化和高带宽的通信需求提供强有力的支持。

8.参考文献

[1] 江卓, 吴茜, 李贺武等. 互联网端到端多路径传输跨层优化研究综述[J]. 软件学报, 2019, 30(2): 302-322.

[2] Li H, Wang Y, Sun R, et al. Delay-based congestion control for multipath TCP in heterogeneous wireless networks[C]//2019 IEEE Wireless Communications and Networking Conference Workshop (WCNCW). IEEE, 2019: 1-6.

[3] Li W, Zhang H, Gao S, et al. SmartCC: A reinforcement learning approach for multipath TCP congestion control in heterogeneous networks[J]. IEEE Journal on Selected Areas in Communications, 2019, 37(11): 2621-2633.

[4] Yin C, Dong P, Du X, et al. An adaptive network coding scheme for multipath transmission in cellular-based vehicular networks[J]. Sensors, 2020, 20(20): 5902.

[5] 余翔, 易明敏, 杨路. 一种 SDN 架构下业务属性相关的多径路由算法[J]. 电信科学, 2016, 32(11): 10-15.

[6] 罗洪斌, 张珊, 王志远. 异构网络融合共生的需求, 挑战与架构[J]. 电信科学, 2022, 38(6): 18-30.

[7] Ford A, Raiciu C, Handley M, et al. TCP extensions for multipath operation with multiple addresses[R]. 2013.

[8] Iyengar J, Thomson M. QUIC: A UDP-based multiplexed and secure transport[M]//RFC 9000. 2021.

[9] Ghosh A, Maeder A, Baker M, et al. 5G evolution: A view on 5G cellular technology beyond 3GPP release 15[J]. IEEE access, 2019, 7: 127639-127651.

变更历史

版本	变更内容	创建者/变更者	参与者	变更日期

格式说明

多级标题规范：

标题：

1　大标题字体为 1 号黑体，不加粗。

2　一级标题用 3 号黑体字，不加粗，用“1”、“2”、“3”……标识。

3　二级标题用 4 号黑体，用阿拉伯数字（Times New Roman）“1.1”“2.1”“3.1”……标识。

4　三级标题用小 4 号黑体，用阿拉伯数字“1.1.1”、“2.1.1”、“3.1.1”……标识。

5　四级标题及以下用小 4 号黑体，用阿拉伯数字“1.1.1.1”、“2.1.1.1”、“3.1.1.1”……标识。

正文：

6　正文中文用小四号宋体、英文用小四号 Times New Roman 字体，不加粗。

7　正文用 1.5 倍行间距。

图表：

8　图、表要有标题，标题用 5 号黑体，编号从 1 开始，例如表 1 XXX、表 2 XXX、图 1 XXXX。

9　表格的表头填充色用浅灰色，表头字体用加粗 5 号宋体，表格内容用 5 号宋体。

10 标题使用建议：标题与标题之间要有文字概述等，不建议几级标题直接连续使用，中间无任何文字。

目录：

11 目录：“目录”两个字中间空格 2 个汉字，目录正文用 5 号宋体。

12 其他：建议通篇语言描述风格一致。