高技术通讯 2022 年 第32 卷 第7 期:727-736

doi:10.3772/j.issn.1002-0470.2022.07.007

基于深度强化学习的 MPTCP 动态编码调度系统①

廖彬彬②[*,**] 张广兴[*] 刁祖龙[*] 谢高岗[***]

([*]中国科学院计算技术研究所 北京 100190)

([**]中国科学院大学 北京 100190)

([***]中国科学院计算机网络信息中心 北京 100190)

摘 要 通过在多宿主设备上使用多个网络接口,现有的多路径传输控制协议(MPTCP)能够实现跨物理链路的吞吐量聚合和单路径故障的连通性恢复,并极大地改善了传统单路径传输控制协议(TCP)的网络服务质量(QoS)。然而,当 MPTCP 多条 TCP 子流中的任意一条出现严重的时延抖动、网络拥塞或数据包丢失等性能瓶颈时,这些高延迟或高损耗的子流将会阻塞其他子流的数据传输,使得 MPTCP 的整体传输性能远远低于预期。已有的研究表明,使用数据包调度器或编码器的方法能够有效地缓解这类网络异构性造成的负面影响。但是针对动态多变的异构网络环境,如何设计出高效且自适应的数据包调度程序和编码算法则变得尤为重要。基于已有的 MPTCP 动态前向纠错编码和数据包按比率分配思想,本文提出了使用深度强化神经网络的 MPTCP 动态多路径编码调度器(DMES)。利用 Transformer 神经网络和深度强化学习的智能体,DMES 通过观测当前 MPTCP 网络环境中动态 TCP 子流组成的网络状态空间,并根据实时的多子流状态梯度搜索最佳的动作集合,以最大化反馈函数中定义的 MPTCP 整体性能。实验结果表明,相较于目前最先进的解决办法,DMES 能更加适应动态多变的网络环境。尤其在高丢包和多子流的情况下,DMES 将异构网络导致的 MPTCP 接收端乱序队列(OQS)降低到 24.6% 以上,并且能够在提升 18.3% 的有效吞吐量的同时将 MPTCP 的应用延迟降低 12.2% 左右。

关键词 多路径传输控制协议(MPTCP);动态前向纠错编码;数据包调度;深度强化学习

0 引 言

多路径传输控制协议[1](multi-path transport control protocol, MPTCP)自 2013 年由互联网工程任务组(Internet Engineering Task Force, IETF)标准化以来,已经受到了工业界和学术界的广泛关注和研究。由于其固有的多链路带宽聚合能力和单链路故障恢复能力,MPTCP 已经被应用于加速众多的数据交互场景(如文件传输、Web 浏览和视频推流等)。尤其在移动互联网的背景下,内核装载 MPTCP 的移动设备可以同时利用WiFi和蜂窝无线网络来提高移动应用程序的网络服务健壮性和传输质量保障[2]。

影响 MPTCP 整体性能的一个重要因素是其数据包调度程序的设计和实现,它需要根据特定的策略在有效的传输控制协议(transport control protocol, TCP)子流上分配合适的数据包数量。已有的研究表明,错误的数据包调度决策将会导致 MPTCP 严重的性能问题[3-4]。尤其在多变的无线网络环境中,由于 TCP 子流的状态特别容易遭受网络拥塞和数据包随机丢失的影响[5],MPTCP 连接上的多条 TCP 子流之间的性能差异变得十分巨大。而这种网络的

① 国家重点研发计划(2019YFB1802801)资助项目。

② 男,1995 年生,博士生;研究方向:多路径 TCP,多路径 QUIC;联系人,E-mail: liaobinbin@ict.ac.cn。

(收稿日期:2021-02-24)

异构性主要表现为分发到较快子流上的数据包必须等待较慢子流上的数据包,这便造成了发送端数据的队头阻塞现象(head-of-line blocking, HoL)和接收端的数据包的乱序队列问题(out-of-order queue size, OQS)。因此,如何最小化接收端的数据包 OQS 成为提升 MPTCP 整体性能的关键[6-7]。

实际上,当 MPTCP 所有的子流都相对正常或异构性不太明显时,设计一个合理的数据包调度器便能够实现多子流的正常传输,并缓解 MPTCP 的队头阻塞与乱序问题。然而,如果存在某些 TCP 子流在网络质量上具有很高的多样性时(如剧烈的丢包损耗、网络拥塞或延迟抖动等),仅依靠调度器是无法适应这种异构网络的剧烈变化,并实现数据包的快速恢复的。而使用网络编码的方法可以根据 TCP 子流的网络状态实现超时或丢包的 0-RTT 快速恢复[8]。因此,在进行数据包调度之前增加一个网络编码的过程[9-10],可以使得 MPTCP 不仅能够在相对稳定的网络环境中获得可观的性能提升,而且还能快速地适应网络的剧烈变化。

然而,基于网络编码的数据包调度系统将主要面临如下两个挑战。首先数据包的编码率需要随着 MPTCP 所有 TCP 子流网络状态的相对变化而改变,例如,当所有的子流都比较稳定时使用较低的编码率,而当部分子流出现明显的拥塞或丢包时则需要提升数据包的编码率。其次,数据包调度器也需要通过衡量所有 TCP 子流的相对差异来确定分配编码后的数据包到每条子流的数量。也就是说,无论是网络编码器还是数据包调度器都需要根据所有 TCP 子流的网络状态来决定其采取的动作。然而,决定 TCP 子流状态的因素众多(如丢包率、拥塞窗口、时延抖动等),这将导致 MPTCP 多 TCP 子流组成的状态空间十分巨大。而且这些因素随着时间的变化可能是线性的也可能是非线性的,甚至可能不符合任何数学分布规律。因此基于数学建模的方法来人工拟合一个函数模型以匹配 MPTCP 的子流状态到其对应的全局最优编码和调度动作几乎是不可能的。

近年来,由于不用假设状态空间的分布情况,使用深度神经网络作为函数的估计器来拟合高维度状态空间到连续动作空间的匹配关系已经变得十分常见[11-12]。尤其在众多复杂多变的计算机网络环境中(如数据包分类[13]、自适应比特率流[14]和数据中心流调度[15]等),基于深度神经网络的强化学习方法(deep reinforcement learning, DRL)已经取得了令人惊讶的效果。受到这些实例的启发,本文尝试训练一个深度增强的 MPTCP 动态编码调度系统动态多路径编码调度器(dynamic multi-path encoding scheduler, DMES),来自适应 MPTCP 多子流异构网络环境下的状态变化,并将 MPTCP 的多子流实时状态输入转化为当前状态下的数据包编码比率和分配比率,从而最大化接收端的 OQS 定义的反馈函数。最后通过梯度下降的方法训练 DMES 的深度神经网络直到收敛。实验结果表明,在剧烈异构的动态网络环境中,DMES 将接收端的 OQS 降低到 24.6% 以上,并且能够在平均有效吞吐量提升 18.3% 的同时将 MPTCP 的应用延迟降低 12.2% 左右。

本文共分为 5 节,其中第 1 节为引言部分,对本文研究背景和主要研究内容进行介绍;第 2 节为相关工作,主要介绍已有的 MPTCP 数据包调度方法、网络编码技术以及深度强化学习方法;第 3 节首先分析已有方法的不足之处,然后提出基于深度强化学习的 MPTCP 动态网络编码调度系统;第 4 节介绍了实验布置及说明,并给出了对应的实验结果;第 5 节对本文主要成果和未来工作进行总结和说明。

1 相关工作

1.1 MPTCP 数据包调度器

在现有的多路径传输系统中,网络的异构性是其数据包调度程序设计的难点。因为在性能差异较大的多条路径之间传输数据将不可避免地导致 OQS 问题。而这种 OQS 乱序排队将直接导致整个 MPTCP 连接的时延上升和吞吐量下降[2,4]。作为 MPTCP 的默认调度器,MinRTT 仅考虑多条路径间的往返时间(round-trip time, RTT)异构性,并向 RTT 最小的子流中发送更多的数据包。而当前的 BLEST[16]、STMS[6] 和 ECF[17] 调度器不仅考虑 RTT 的差异,同时将不同路径间的拥塞窗口、吞吐量、in-flight 缓存以及发送队列等 TCP 层的属性差异放到

一个数学模型中,以向质量更好的 TCP 子流中预先分配定量计算的数据包数量,从而缓解网络异构性导致的乱序问题。

但是已有的研究表明,这类数学的方法为了简化建模的过程,通常假设了特定的网络环境,如 MPTCP 只建立 2 条子流、不存在主机或网络缓存的限制、子流的丢包率忽略不计等,以大幅缩小 TCP 子流的状态空间,使得调度器只需要简单地使用 TCP 层的几个属性就能拟合函数来计算每条子流需要分配的数据包数量[7,18],但这也导致了基于数学建模的方法往往只能在有限类型的网络环境中优化特定的 MPTCP 性能指标,却无法自适应更加复杂的网络环境。

1.2 MPTCP 网络编码

在高丢包或高拥塞的动态网络中[19],由于 TCP 子流频繁的数据包重传和重排序,将导致剧烈的网络异构性。这使得仅优化数据包调度器已经无法实现 MPTCP 的带宽聚合和故障恢复能力[20]。有一些解决方案尝试通过优化 MPTCP 的拥塞控制算法,以弥补由于拥塞丢包而导致的性能下降[21-22]。但是,高丢包和高延迟的网络环境很多时候并不是由于网络拥塞导致的。尤其在无线网络中,信号衰减、噪声干扰、链路切换等情况将导致大量的无线随机丢包[2,20]。在无线网络上引入网络编码,已被证实能够极大地降低丢包概率和减少重传,从而提高 TCP 子流的吞吐量[8-10]。

为了实现多路径的容错传输,MPLOT[23]采用固定比率的编码方案来减少数据包的恢复延迟,以提高无线自组织网络的质量。但是当多条路径的质量急剧变化时,固定速率编码的性能将变得不佳[9]。因为编码的过程需要时间代价,而编码的冗余程度也需要消耗额外的网络带宽。不同的网络环境显然需要不同的编码率,例如在网络完全没有丢包时可以不使用编码,而在所有的路径都存在严重的丢包时使用全冗余编码能够极大地保障网络的可用性。FMPTCP[10]率先提出了使用喷泉码来实现多路径 TCP 的动态编码机制,而 MPTCP-dFEC[9]则基于前向纠错码(FEC)实现 MPTCP 数据传输的动态前向纠错。但与之前的数据包调度器设计类似,这两种编码机制都仅基于 TCP 子流的几个属性来建立一个启发式的数学模型以适应一些特定的网络场景。更准确地说,FMCTP 仅使用拥塞窗口、丢包率和 RTT 来调整码率,而 MPTCP-dFEC 则使用丢包率和 RTT 来调整 FEC 的编码率。相较于喷泉码,FEC 编解码的复杂度更低,而且数据恢复能力也相差无几。因此 FEC 更常用于延迟敏感的网络编码调度中。

1.3 深度增强神经网络

增强学习的变种有很多,其中主要包括 Q-learning[24]、Sarsa[25]、Deep Q network(DQN)[11]、以及 Actor-Critic[26]等。在任意的时间步骤 t,一个标准的增强学习系统需要训练 Agent 智能体与未知的环境之间进行交互,智能体根据环境当前的状态 s_t 执行动作 a_t,以最大化自定义的反馈函数 r_t。对于状态空间较小的问题,Q-learning 和 Sarsa 可以通过映射表的方式,将每个状态 s_t 采用动作 a_t 时得到的反馈 r_t 的值存储到 Q 值表中,在训练的过程中不断更新 Q 值表以最大化 r_t 直到收敛。考虑到很多问题的复杂性,使用简单的 Q 表已经无法存储巨大的状态空间。因此使用深度神经网络作为函数估计器映射状态空间 s_t 和动作空间 a_t 的 Q 值被引入到 DQN 中。

但是,DQN 只能处理离散和低维的动作空间,而很多任务的动作空间,如 MPTCP 数据包的编码率和分配率,都是连续的值。考虑到 Policy Gradient(PG[27])的方法常被用到处理连续的控制动作,因此基于 Actor-Critic 的深度确定性策略梯度下降(deep deterministic policy gradient,DDPG[12])算法通过整合 DQN 和 PG,被广泛应用于处理高维状态空间下的连续动作控制。考虑到 MPTCP 的多子流网络状态空间的复杂性和编码调度动作的连续性,为了建立适应各种异构环境下的动态编码调度模型,只有使用 DDPG 的 Actor-Critic 深度神经网络才能够将所有可能的 TCP 子流状态 s_t 估计到其对应的数据包编码率和分配率 a_t 的函数模型,并最大化 MPTCP 整体性能定义的反馈函数 r_t。

2 MPTCP 动态网络编码调度系统

2.1 MPTCP 编码与调度分析

由于 MPTCP 的编码器与调度器用于解决不同

的传输问题,因此现有 MPTCP 系统中的编码和调度模块是分开实现的。但是数据包在经过编码后一定会经过调度器模块的分配,因此编码器和调度器又是强耦合在一起的。如图 1 所示,在设计 DMES 动态网络编码调度系统之前,本文首先基于多路径协同滑动调度器(slide together multi-path scheduler, STMS[6])使用带 TC 模块[28]的无线接入设备构建了一个网络可控的 MPTCP 实验平台,并对已有的编码器、调度器以及两者的组合进行测试分析,以评估当前技术在动态异构网络环境下的性能瓶颈。为了模拟动态复杂的网络环境,每条 MPTCP 连接将建立 $n=3$ 条 TCP 子流,并根据表 1 中每个 Case 在 RTT、丢包率和可用带宽的值域范围内,使用 TC 模块随机地设置这 3 条 TCP 子流的网络质量。而且通过对每个 Case 进行 10 组多路径传输实验,分析 MPTCP 已有的编码器 MPTCP-dFEC(dFEC)、FMPTCP(FMP),目前性能最佳的调度器 ECF 以及它们的组合在各种网络环境设置下的平均 OQS 情况。

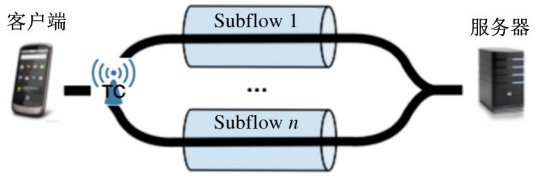

图 1　实验台拓扑图

表 1　子流数不变时的网络质量参数

测试组	RTT/ms	带宽/Mbps	丢包
Case 1	(10,80)	(60,70)	1% ~5%
Case 2	(100,200)	(40,60)	1% ~5%
Case 3	(200,400)	(10,40)	1% ~5%
Case 4	(10,80)	(60,70)	10% ~20%
Case 5	(100,400)	(40,60)	10% ~20%
Case 6	(200,400)	(10,40)	10% ~20%

由图 2 可以看出,相比于现有的编码器,调度器 ECF 能够更好地解决 Case 1、2 和 3 中 RTT 和可用带宽异构性导致的 OQS。而编码器 dFEC 和 FMP 能够更好地解决 Case 4、5 和 6 中高丢包网络导致的异构性。FMP 在高丢包的 Case 6 中表现得比 dFEC 更好,但是在丢包相对较小的 Case 2 中,dFEC 却有更好的性能,这是因为喷泉码的高丢包恢复能力是以编解码开销为代价的。对于 Case 5 中既高延迟抖动又高丢包率的环境,编码器组合调度器的性能有一定的提升,但是仅比单独使用编码器时的平均 OQS 降低 3% 左右。这是因为现有的编码器和调度器都只是根据一部分的 TCP 状态属性各自建立线性数学模型,并不能适应非线性变化的多路径网络环境。而这种简单的组合更不能很好地适应动态复杂的网络环境。

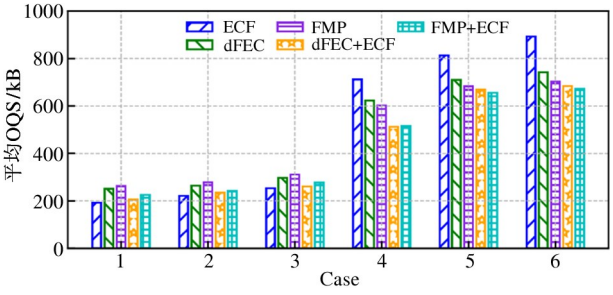

图 2　不同方法在异构网络环境下的平均 OQS

2.2　DMES 网络编码调度系统设计

基于数学建模的方法估计 MPTCP 的数据包编码率和分配率是积累了人类有限的测量经验在特定网络环境下的应用,因此缺乏对各种网络异构性的泛化能力和自适应性。它们不仅只选取部分的 TCP 状态属性作为输入,而且还会假设 MPTCP 不会同时动态维持太多的 TCP 子流。但在真实场景中,MPTCP 建立的 TCP 子流数目却是动态变化的,子流的状态空间也由很多属性参数共同决定,而且参数之间还存在各种联系。因此使用深度增强神经网络(DNN)作为函数模型的估计器,能够在不需要假设 MPTCP 网络环境的前提下,将 TCP 子流的全部属性组成状态空间输入到 DNN 中,以训练出一个将所有可能出现的“状态到动作”的匹配关系都考虑进去的泛化模型。

因此在本节中,DMES 首先使用 TCP 层的属性参数设计 MPTCP 的状态空间,并通过 Transformmer 神经网络解决这些参数之间的相关性和 TCP 子流数目的动态性,以输出一个定长矩阵作为 DNN 的输入。而 DMES 作用在 TCP 子流上的动作空间则默

认使用 MPTCP 数据包的编码率与分配率。然后根据 MPTCP 的设计原理,引入一个新的 flag-bit 位来返回 DMES 系统的反馈值。最后根据这些定义详细地说明 DMES 的流程。

2.2.1 MPTCP 状态空间设计

状态空间:每个时刻 DMES 系统输入的状态空间,实际上是 MPTCP 当前网络环境的一次快照。对于任意的时刻 t,系统智能体需要输入的状态空间可以表示为 $s_t=(s_{t1}, s_{t2},\cdots s_{ti}\cdots s_{tn})$,$s_{ti}(1\leqslant i\leqslant n)$ 是第 i 条子流 TCP 层的全部属性参数,可以用一个元组表示为 $s_{ti}=(d_{ti}, c_{ti}, b_{ti}, l_{ti}, w_{ti}, k_{ti})$,其中,$d_{ti}$表示 TCP 子流 i 的往返时延(RTT)、c_{ti}表示 TCP 子流 i 的拥塞窗口(congestion window, CWND)、b_{ti}表示 TCP 子流 i 的数据包交付速率(packet delivery rate, PDR)、l_{ti}表示 TCP 子流 i 的数据包丢失率(PLR)、w_{ti}表示 TCP 子流 i 的接收窗口(RWND)、k_{ti}表示 TCP 子流 i 的 MPTCP 连接级别的数据包确认数量(Data ACKed)。

使用 MPTCP 的子流加入选项(MP _ JOIN option)和子流关闭选项(RST option),MPTCP 的客户端和服务器间能够动态地维持 n 条存活的 TCP 子流。而每个时刻的状态 s_t 由 MPTCP 当前存活的 TCP 子流构成,这将会导致状态 s_t 的长度是随机变化的。但矛盾的是神经网络的输入层却是固定长度的。考虑到基于多头注意力模型的 Transformer 神经网络[29]可以将变长的输入转换为定长的 Matrix,同时还能将输入属性间的相关性附加到输出的 Matrix 中。因此可以将变长的状态空间 s_t 经过 Transformer 神经网络转换为定长的 Matrix 后喂给 DMES 的神经网络输入层。这不仅能够解决状态空间的动态性问题,而且还能解决属性间的相关性对神经网络模型复杂度的影响。

2.2.2 动作空间与反馈函数设计

动作空间:在任意的时刻,DMES 根据 TCP 子流的网络状态空间对即将发送的数据包首先需要进行编码动作然后进行调度动作。现有的编码算法中,基于异或的前向纠错码(FEC)不仅复杂度低而且数据恢复能力强,因此被广泛应用于 TCP-IR[8]、QUIC[30]和 MPQUIC[31]等网络协议的数据包编码恢复中。2.1 节中的对照实验也表明,在大多数情况下,动态 FEC 能够在 MPTCP 中取得更好的性能。如图 3 所示,对于任意一个动态的 FEC 编码模块(m,k),其中 k 表示应用层的源数据包数量,而 m 是编码后的总数据包数量,$m-k$ 则是冗余的奇偶校验数据包数量。这些奇偶校验数据包使得 MPTCP 的接收端只需要接收任意 k 个编码数据包,就能恢复应用层的 k 个源数据包。因此,在任意的时刻 t 根据所有 TCP 子流组成的状态空间 s_t,数据包的编码率可以表示为 $e_t=(m-k)/k$。而对于编码后的 m 个数据包,MPTCP 的调度器需要确定分配到每条 TCP 子流 i 上的数据包数量为 x_{ti}。如果 MPTCP 总共维持了 n 条子流,则分配到子流 i 的比率表示为 $p_{ti}=x_{ti}/m$,而且 $p_{t1}+p_{t2}+\cdots p_{ti}\cdots+p_{tn}=1$。由于编码和调度强耦合在一起,因此用一个集合将 DMES 的动作空间表示为 $a_t=(e_t, p_{t1}, p_{t2}, \cdots, p_{ti}, \cdots, p_{tn})$。

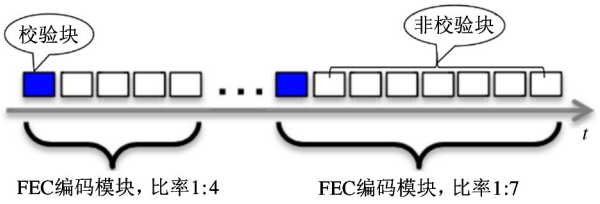

图 3 基于异或的 FEC 的编码率

反馈函数:根据 MPTCP 的设计原则,应用层的延迟和有效吞吐量是当前 MPTCP 连接的主要 QoS 指标。越来越多的研究表明,影响 QoS 的根本原因是 MPTCP 接收端的 OQS 大小,以及其衍生出来的发送端的 HoL 问题。因此,在设计 DMES 的反馈函数 r_t 时需要最小化接收端的 OQS。由于 DMES 运行在 MPTCP 的发送端队列与 TCP 子流之间,这使得接收端的 OQS 大小需要随着连接级别的 Data ACK数据包从接收端返回到发送端。如图4所示,

类型	长度	子类型	保留位	Q	F	M/m	A/a
级别级别数据包确认号							
连接级别数据序列号							
子流级别序列号							
数据级别长度				乱序队列大小			

图 4 DATA ACK 的 Q-bit 标志位

在每个 Data ACK 数据包的数据序列信号字段(data sequence signal option, DSS)新增一个 Q-bit 信号位[7]来返回接收端 OQS。为了使得 OQS 越小越好,DMES 的反馈函数定义为 $r_t = 1/OQS$。

2.2.3 DMES 系统设计

根据上述定义,可以发现 MPTCP 动态编码调度系统的状态空间 s_t 与动作空间 a_t 都是高维度且连续变化的。因此如图 5 所示,本文使用基于 Actor-Critic 的 DDPG 深度神经网络来估计 s_t、a_t 以及 r_t 之间的函数关系模型。DDPG 共维持了 4 个深度神经网络(DNNs)作为函数模型的估计器。其中,2 个 DNNs 是 Actor 深度神经网络和 Critic 深度神经网络,分别用 θ^{μ} 和 θ^{Q} 表示其神经网络的参数。Actor 网络也被称为策略函数网络,用来估计当前观测到的状态 s_t 与需要采取的最佳动作 a_t 间的函数关系为 $a_t = \mu(s_t; \theta^{\mu})$。Critic 网络也被称为值函数网络,用来估计对于任意的“状态-动作”对(s_t, a_t)的反馈函数的值为 $r_t = Q(s_t, a_t; \theta^{Q})$。如图 5 所示,深度强化学习通过 min-batches 的方式从“回放缓存”[32]中找到某个时刻 t 到下一时刻 $t+1$ 的状态转换元组$(s_t, at, s_{t+1}, a_{t+1})$,并根据偏导数式(1)和式(2),以 $SGD(\theta^{\mu})$ 和 $SGD(\theta^{Q})$ 的随机梯度下降方式训练 Actor 和 Critic 表示的神经网络:

$$\nabla_{\theta^{Q}} L(\theta^{Q}) = E_{s_t,\, a_t \sim \rho}[\, y_t - Q(s_t, a_t; \theta^{Q}))\times \nabla_{\theta^{Q}} Q(s_t, a_t; \theta^{Q})\,] \tag{1}$$

$$\nabla_{\theta^{\mu}} J \approx E_{s_t \sim \rho}[\, \nabla_a Q(s_t, a_t; \theta^{Q}) \times \nabla_{\theta^{\mu}} \mu(s_t; \theta^{\mu})\,] \tag{2}$$

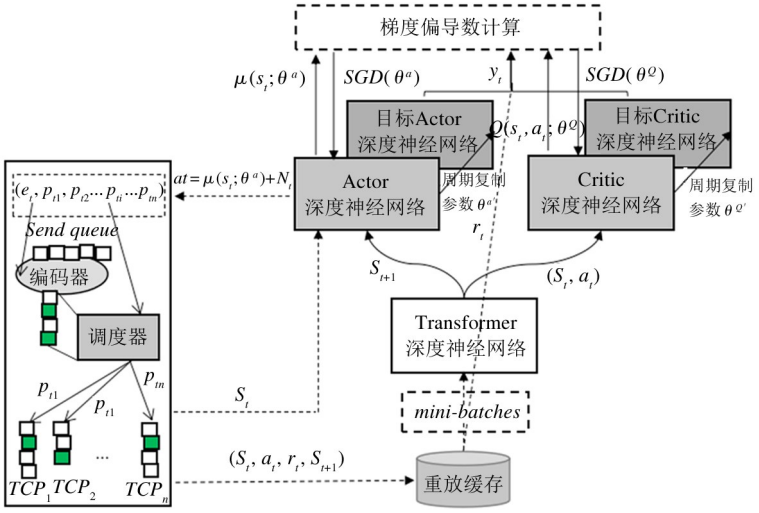

图 5 基于深度强化学习的 MPTCP 编码调度系统

而另外 2 个神经网络为目标 Actor 网络 $\theta^{\mu'}$ 和目标 Critic 网络 $\theta^{Q'}$,主要用来输出每个时刻用来计算损失函数的目标值 y_i:

$$y_i = r_i + \gamma Q'(s_{i+1}, \mu'(s_{i+1}; \theta^{\mu'}) \mid \theta^{Q'}) \tag{3}$$

这 2 个目标网络的参数 $\theta^{\mu'}$ 和 $\theta^{Q'}$ 则是使用跟踪参数 $\tau \ll 1$ 从 θ^{μ} 和 θ^{Q} 缓慢更新而来:

$$\theta^{Q'} \leftarrow \tau\theta^{Q} + (1-\tau)\theta^{Q'} \tag{4}$$

$$\theta^{\mu'} \leftarrow \tau\theta^{\mu} + (1-\tau)\theta^{\mu'} \tag{5}$$

最终的 DMES 训练算法如算法 1 所示,将主要由式(1)、(2)、(4)、(5)更新完成。在算法训练开始时,需要随机初始化 Actor-Critic 网络的 θ^{μ} 和 θ^{Q} 值。目标神经网络的参数 $\theta^{\mu'}$ 和 $\theta^{Q'}$ 也从初始化的参数复制而来。为了对状态转换进行最小采样(mini-batches)采样,因此初始化“重放缓存”为 R。为了实现对连续的动作空间的探索过程,需要为每个动作的探索初始化一个噪声参数 N[12]。最后将 MPTCP 的子流初始状态空间 s' 放入 R 中。初始化完成之后,DMES 将进入一个无限循环直到收敛。对于每个 Data ACK 的返回时刻 t,DMES 的智能体将“探索噪声 N”和 Actor 神经网络的输出组成状态

s_t 的匹配动作,即 $a_t = \mu(s_t;\theta^{\mu}) + N_t$。然后,$a_t$ 中的码率 e_t 和比率 p_t 将会被应用到 MPTCP 发送端的编码器和调度器上。而在 t 时刻应用 a_t 导致的反馈值 r_t(即 OQS),将在 $t+1$ 时刻由 Data ACK 返回,并与状态 s_{t+1} 组成(s_t, a_t, r_t, s_{t+1})存储到回放缓存 R 中。而 Actor-Critic 神经网络的更新,则是通过从 R 中 mini-batches N 个状态转换元组(s_t, a_t, r_t, s_{t+1}),并使用式(1)和(2)进行随机梯度下降更新参数 θ^{μ} 和 θ^{Q}。而目标 Actor-Critic 神经网络 $\theta^{\mu'}$ 和 $\theta^{Q'}$ 的更新则是通过式(4)和(5)完成。

算法 1 DMES 训练算法

1. 随机初始化权重分别为 θ^{Q} 和 θ^{μ} 的 Critic 值神经网络 $Q(s_t, a_t, \theta^{Q})$ 和 Actor 策略神经网络 $\mu(s;\theta^{\mu})$;
2. 初始化目标神经网络 Q' 和 μ',通过 $\theta^{Q'} \leftarrow \theta^{Q}$ 和 $\theta^{\mu'} \leftarrow \theta^{\mu}$ 分别进行权重复制;
3. 初始化回放缓存 $R \leftarrow \phi$;
4. 对每个探索动作,初始化一个随机的噪声 N;
5. 从 Transformer 神经网络中接收初始化状态 s';
6. **For** 每个接收到的 Data ACK **do**
7. 选择动作 $a_t = \mu(s_t;\theta^{\mu}) + N_t$;
8. 执行动作 a_t 并观测反馈 r_t 和下一个状态 s_{t+1};
9. 将状态转移元组(s_t, a_t, r_t, s_{t+1}) 存入 R;
10. 从 R 中采样转移元组(s_i, a_i, r_i, s_{i+1});
11. 计算 $y_i = r_i + \gamma Q'(s_{i+1}, \mu(s_{i+1};\theta^{\mu'})|\theta^{Q'})$;
12. 更新 Critic 神经网络通过更新目标损失函数:
$$L = \frac{1}{N}\sum_{0}^{i}(y_i - Q(s_i, a_i, \theta^{Q}))^2$$
13. 更新 Actor 策略神经网络通过梯度:
$$\nabla_{\theta^{\mu}} J \approx \frac{1}{N}\sum_{0}^{i} \nabla_{a} Q(s_i, a_i, \theta^{Q}) \nabla_{\theta^{\mu}} \mu(a_i, \theta^{\mu}))$$
14. 更新目标神经网络:
$\theta^{Q'} \leftarrow \tau\theta^{Q} + (1-\tau)\theta^{Q'}$ 和 $\theta^{\mu'} \leftarrow \tau\theta^{\mu} + (1-\tau)\theta^{\mu'}$
15. **End For**

3 实验评估

3.1 实验布置及说明

为了让 MPTCP 的对端都支持动态的 FEC 编码,需要在 MPTCP 的握手报文中新增 *FEC _ CABLE* 选项进行对端的 *FEC-enabled* 协商。由于 MPTCP 是内核空间的程序,而基于 Tensorflow[33] 的 Actor-Critc 深度神经网络是用户空间的程序,因此 DMES 使用系统调用函数 *getsockopt*( ) 从内核空间中周期性地获取 MPTCP 的 TCP 子流状态空间 s_t,并通过系统调用函数 *setsockopt*( ) 来执行 Actor-Critic 神经网络在不同的状态 s_t 下输出的 MPTCP 动作空间 a_t,以最小化 r_t 中 MPTCP 接收端的 OQS 队列。

在训练 DMES 时,需要使用 6 层叠加的多头注意力(multi-head attention)模型组成 Transformer 神经网络,每个注意力模型的输入是一个 512 维的矩阵。与此同时,Actor 神经网络和 Critic 神经网络由 2 个全连接的 48 ×48 的神经网络隐藏层和神经网络输出层组成,并使用 rectified linear 函数作为隐藏层的激励函数,以及 hyperbolic tangent 函数作为输出层的激励函数[34]。根据 Adam 方法[35],将更新 Actor 网络和 Critic 网络的学习率分别设置为 10^{-4} 和 10^{-5}。为了模拟动态复杂的网络环境,本节继续使用第 2.1 节中的可控实验床,具体的参数设置如表 2 所示,且每种 Case 进行 20 组多路径传输实验。

表 2 子流数变化时的网络质量参数

测试组	RTT/ms	带宽/Mbps	丢包率	子流数
Case 1	(10,80)	(60,80)	5% ~10%	3 -5
Case 2	(100,200)	(30,50)	5% ~10%	6 -8
Case 3	(10,80)	(60,80)	10% ~15%	3 -5
Case 4	(100,200)	(30,50)	10% ~15%	6 -8
Case 5	(10,80)	(60,80)	15% ~20%	3 -5
Case 6	(100,200)	(30,50)	15% ~20%	6 -8

3.2 性能分析

(1)乱序队列

如图 6 所示,本小节首先分析了各种异构网络环境下的 MPTCP 乱序队列分布情况。实验结果表明,相比于其他的方法 DMES 在所有的 Case 中的 OQS 都是最小的。且在多子流高丢包的环境中,将 MPTCP 的平均 OQS 最高降低 20% 以上。因为深度神经网络将编码器和调度器完美融合到一起,不仅能够基于调度的方式自适应轻微网络异构性而且能够通过编码恢复的方式极大地降低剧烈网络异构性导致的接收端乱序队列数量。

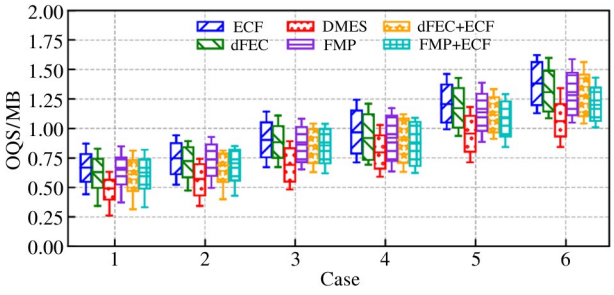

图 6　不同方法在动态异构网络下的 OQS 分布情况

为了更加清楚地描述实验结果，本小节分析了在不同的丢包率和 TCP 子流数量下的 MPTCP 接收端的平均 OQS 情况。如图 7(a)所示，与其他的设计方法相比，在 15% ~20% 的高丢包环境中，DMES 最高能够将平均 OQS 降低 17.4% 左右，若此时 MPTCP 同时建立了 8 条 TCP 子流，如图 7(b)所示，DMES 将平均 OQS 降低到 24.6% 以上。

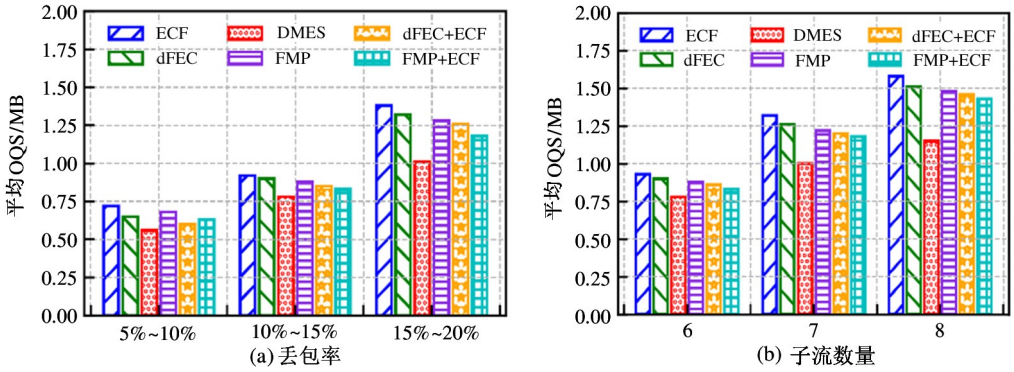

图 7　不同方法在不同丢包和子流下的平均 OQS

(2)应用延迟

由于接收端的乱序直接导致了数据包的延迟交付。因此传输实验中测量了 MPTCP 连接级别的平均应用延迟。如图 8 所示，在子流数目较少且丢包不高的 Case 1 中进行多路径传输实验时，由于 ECF + dFEC 和 DMES 几乎没有编码开销而且能够通过调度器适应网络的异构性，因此它们的应用延迟明显低于存在编码开销的 dFEC 和 FMP。当在Case 3 和 Case 4 中的丢包率达到 10% ~15% 时，相比于 ECF 调度器，DMES 却能够自适应地编码，因此将应用延迟降低大约 6.3% 左右。最后在子流的数量为 6 ~ 8 条且丢包率达到 15% ~20% 的 Case 6 中，DMES 能够将平均应用延迟最高降低 12.2% 以上。

(3)有效吞吐量

MPTCP 连接的吞吐量是 TCP 子流共同作用的结果。但由乱序到达的报文必须按序交付给应用层，使得 MPTCP 的有效吞吐量并不是 TCP 子流吞吐量的简单叠加。因此，通过测量应用程序的实时吞吐量作为 MPTCP 的有效吞吐量。如图 9 所示，在丢包率较低的 Case 1 和 Case 2 中，相较于 ECF 调度器，DMES 的有效吞吐量提升了 7.4% 左右。而在丢包率较高的 Case 6 中，相比于 ECF、FMP 和 dFEC 的设计方法，DMES 的有效吞吐量提升能够分别达到 18.3%、14.7% 和 15.2% 左右。与此同时，当子流的数目较多时 MPTCP 的网络虽然变得更加复杂，但 DMES 的有效吞吐量提升却更加明显。

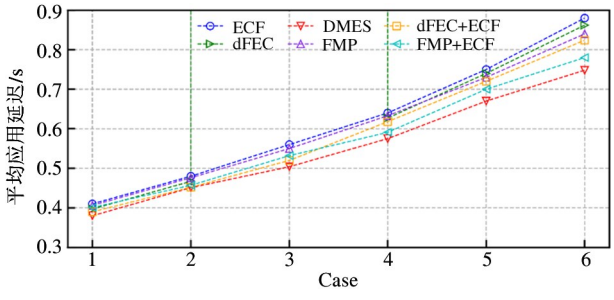

图 8　不同方法在动态异构网络下的平均应用延迟

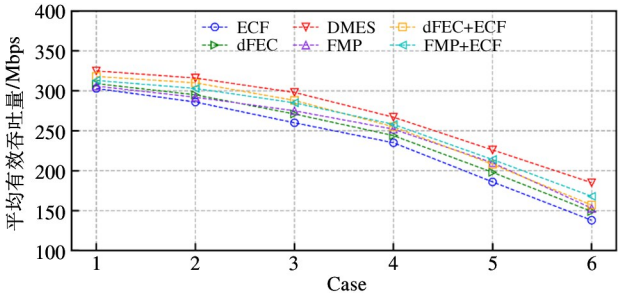

图 9　不同方法在动态异构网络下的平均有效吞吐量

4 结 论

本文通过分析现有的编码器、调度器以及两者的组合在动态多变的异构网络环境中的表现情况,发现了编码器组合调度器在剧烈的异构网络中的性能提升空间。并通过使用深度强化神经网络作为函数模型的估计器,利用 Transformer 神经网络处理 MPTCP 连接动态 TCP 子流的全部属性,以匹配出编码器和调度器当前最佳的编码动作和调度动作,从而最大化 MPTCP 整体传输性能。实验结果表明,本文提出的方法能够实现编码器和调度器能力的完美融合,从而更加适应动态复杂的多路径网络环境。

未来工作将尝试基于深度强化学习的多路径动态编码调度思想应用到用户态的多路径传输协议中,以减少内核态与用户态的交互开销,同时采用更多真实环境下的多路径数据传输实验以增加系统的稳定性。

参考文献

[1] RAICIU C, PAASCH C, BARRE S, et al. How hard can it be? designing and implementing a deployable multipath TCP[C]//Proceedings of the 9th USENIX Symposium on Networked Systems Design and Implementation, San Jose, USA, 2012: 399-412

[2] PAASCH C, DETAL G, DUCHENE F, et al. Exploring mobile/WiFi handover with multipath TCP[C]//Proceedings of the 43th ACM SIGCOMM Workshop on Cellular Networks: Operations, Challenges, and Future Design, Helsinki, Finland, 2012: 31-36

[3] 冯露葶. RTT 不对称路径的 MPTCP 调度策略研究[D]. 南宁: 广西大学信息科学与技术学院, 2019: 15-23

[4] PAASCH C, FERLIN S, ALAY O, et al. Experimental evaluation of multipath TCP schedulers[C]//Proceedings of the 45th ACM SIGCOMM Workshop on Capacity Sharing Workshop, New York, USA, 2014: 27-32

[5] 刘启发. 多路径 TCP 的流量分配与数据调度算法研究[D]. 杭州: 浙江大学信息与电子工程学院, 2017: 21-33

[6] SHI H, CUI Y, WANG X, et al. STMS: improving MPTCP throughput under heterogeneous networks[C]//Proceedings of the 10th USENIX Annual Technical Conference, Renton, USA, 2018: 719-730

[7] LIAO B, ZHANG G, DIAO Z, et al. Precise and adaptable: leveraging deep reinforcement learning for GAP-based multipath scheduler[C]//Proceedings of the 19th IFIP Networking Conference, Paris, France, 2020: 154-162

[8] FLACH T, DUKKIPATI N, CHENG Y, et al. TCP instant recovery: incorporating forward error correction in TCP[J]. *Working Draft, IETF Secretariat*, 2013, 32: 102-113

[9] FERLIN S, KUCERA S, CLAUSSEN H, et al. MPTCP meets FEC: supporting latency-sensitive applications over heterogeneous networks[J]. *IEEE/ACM Transactions on Networking*, 2018, 26(5): 2005-2018

[10] CUI Y, WANG L, WANG X, et al. FMTCP: a fountain code-based multipath transmission control protocol[J]. *IEEE/ACM Transactions on Networking*, 2014, 23(2): 465-478

[11] MNIH V, KAVUKCVUOGLU K, SILVER D, et al. Playing atari with deep reinforcement learning[EB/OL]. http://arxiv.org/pdf/1312.5602.pdf: arXiv, (2013-12-19),[2021-01-12]

[12] LILLICRAP T P, HUNT J J, PRITZEL A, et al. Continuous control with deep reinforcement learning[P]. US patent: US10776692B2, 2020

[13] 冯秀芳, 肖文炳. 神经网络的数据分类算法在物联网中的应用[J]. 计算机技术与发展, 2012, 22(8): 245-248

[14] 罗志强, 王伟, 朱晓荣. 基于 A3C 的无线异构网络自适应视频流传输控制方法[J]. 电信科学, 2020, 36(12):69-80

[15] CHEN L, LINGYS J, CHEN K, et al. Auto: scaling deep reinforcement learning for datacenter-scale automatic traffic optimization[C]//Proceedings of the 49th Conference of the ACM Special Interest Group on Data Communication, Budapest, Hungary, 2018: 191-205

[16] FERLIN S, ALAY Q, MEHANI O, et al. BLEST: blocking estimation-based MPTCP scheduler for heterogeneous networks[C]//Proceedings of the 15th IFIP Networking Conference and Workshops, Vienna, Austria, 2016: 431-439

[17] LIM Y, NAHUM E M, TOWSLEY D, et al. ECF: an MPTCP path scheduler to manage heterogeneous paths[C]//Proceedings of the 13th International Conference on Emerging Networking Experiments and Technologies, Seoul/Incheon, Korea, 2017: 147-159

[18] ZHANG H, LI W, GAO S, et al. Reles: a neural adaptive multipath scheduler based on deep reinforcement learning[C]//Proceedings of the 38th International Conference on Computer Communications, Paris, France, 2019: 1648-1656

[19] LI L, XU K, WANG D, et al. A measurement study on TCP behaviors in HSPA networks on high-speed rails[C]//Proceedings of the 34th International Conference on Computer Communications, Hong Kong, China, 2015: 2731-2739

[20] LI L, XU K, LI T, et al. A measurement study on multipath tcp with multiple cellular carriers on high speed rails[C]//Proceedings of the 49th Conference of the ACM Special Interest Group on Data Communication, Budapest, Hungary, 2018: 161-175

[21] 吴桦, 王凌, 程光. 基于 DASH 流媒体的 TCP 拥塞控制算法优化[J]. 计算机研究与发展, 2019, 56(9): 165-176

[22] SINKY H, HAMDAOUI B, GUIZANI M. Proactive multipath TCP for seamless handoff in heterogeneous wireless access networks[J]. *IEEE Transactions on Wireless Com-*

munications, 2016, 15(7): 4754-4764
[23] SHARMA V, KALYANARAMAN S, KAR K, et al. MPLOT: a transport protocol exploiting multipath diversity using erasure codes[C]// Proceedings of the 37th International Conference on Computer Communications, Phoenix, USA, 2008: 121-125
[24] WATKINS C J C H, DAYAN P. Q-learning[J]. *Machine Learning*, 1992, 8(3-4): 279-292
[25] RANDLOV J, ALSTROM P. Learning to drive a bicycle using reinforcement learning and shaping[C]// Proceedings of the 15th International Conference on Machine Learning, Madison, USA, 1998: 24-27
[26] KONDA V R, TSITSIKLIS J N. Actor-critic algorithms [C]// Proceedings of the 23th Advances in Neural Information Processing Systems, San Jose, USA, 2000: 1008-1014
[27] SILVER D, LEVER G, HEESS N, et al. Deterministic policy gradient algorithms[C]// Proceedings of the 31th International Conference on Machine Learning, Renton, USA, 2014: 387-395
[28] HUBERT B. Linux advanced routing and traffic control HOWTO[J]. *Netherlabs BV*, 2002, 13(2): 22-31
[29] VASWANI A, SHAZEER N, PARMAR N, et al. Attention is all you need[EB/OL]. http://arxiv.org/pdf/1706.03762.pdf:arXiv, (2017-12-06),[2021-01-12]
[30] LANGLEY A, RIDDOCH A, WILK A, et al. The QUIC transport protocol: design and Internet-scale deployment [C]// Proceedings of the 48th ACM Special Interest Group on Data Communication, Los Angeles, USA, 2017: 183-196
[31] DE CONINCK Q, BONAVENTURE O. Multipath quic: design and evaluation[C]// Proceedings of the 13th International Conference on Emerging Networking Experiments and Technologies, Seoul/Incheon, Korea, 2017:160-166
[32] SCHAUL T, QUAN J, ANTONOGLOU I, et al. Prioritized experience replay [EB/OL]. http://arxiv/pdf/1511.05952.pdf:arXiv, (2016-02-25),[2021-01-12]
[33] ABADI M, BARHAM P, CHEN J, et al. Tensorflow: a system for large-scale machine learning[C]// Proceedings of the 12th USENIXsymposium on Operating Systems Design and Implementation, Savannah, USA, 2016: 265-283
[34] NAIR V, HINTON G E. Rectified linear units improve restricted Boltzmann machines[C]// Proceedings of the 27th International Conference on Machine Learning, Haifa, Israel, 2010: 21-24
[35] KINGMA D P, BA J. Adam: a method for stochastic optimization[C]// The 3rd International Conference for Learning Representations, San Diego, USA, 2015: 1-15

A dynamic coding and scheduling system of MPTCP based on deep reinforcement learning

LIAO Binbin[*,**], ZHANG Guangxing[*], DIAO Zulong[*], XIE Gaogang[***]
([*] Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190)
([**] University of Chinese Academy of Sciences, Beijing 100190)
([***] Computer Network Information Center, Chinese Academy of Sciences, Beijing 100190)

Abstract

By using multiple network interfaces on multi-hosted devices, the existing multi-path transport control protocol (MPTCP) transport protocol can achieve throughput aggregation across physical links and connectivity recovery from the single-path failures, and greatly improve the network quality of service (QoS) of the traditional single-path transport control protocol (TCP). However, when any one of MPTCP's multiple TCP subflows suffers serious performance bottlenecks such as delay jitter, network congestion or packet loss, these high delay or high loss subflows will block the data transmission of other subflows, which makes the overall performance of MPTCP far lower than expected. Existing studies have shown that using packet scheduler or encoder can effectively alleviate the negative effects caused by the heterogeneity of such networks. However, it becomes important to design an efficient and adaptive packet scheduler and coding algorithm in the dynamic and changeable heterogeneous network environment. Based on the existing MPTCP dynamic coding and splitting ratio, a dynamic multi-path encoding schecluler (DMES) is proposed by using deep reinforcement learning. By using the transformer network and deep reinforcement learning agent, DMES observes the state space of current MPTCP network environment and searches the best action, so as to maximize the overall performance of MPTCP. The experimental results show that compared with the most advanced solutions, DMES can better adapt to the dynamic and changeable network environment. Especially in the case of high packet loss and multiple subflows, DMES can reduce the out-of-order queues (OQS) by up to 24.6%, and increase the goodput by 18.3% while reducing the application delay by about 12.2%.

Key words: multi-path transport control protocol (MPTCP), dynamic forward error correction coding, packet scheduling, deep reinforcement learning