时间敏感网络中的可变长整形队列调整算法

蔡嫦娟, 庄雷, 杨思锦, 王家兴, 阳鑫宇

引用本文

蔡嫦娟, 庄雷, 杨思锦, 王家兴, 阳鑫宇. 时间敏感网络中的可变长整形队列调整算法[J]. 计算机科学, 2024, 51(8): 354-363.

CAI Changjuan, ZHUANG Lei, YANG Sijin, WANG Jiaxing, YANG Xinyu. Variable-length Shaping Queue Adjustment Algorithm in Time-sensitive Networks [J]. Computer Science, 2024, 51(8): 354-363.

计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.230500214

时间敏感网络中的可变长整形队列调整算法

蔡嫦娟[1]　庄　雷[2]　杨思锦[2]　王家兴[1]　阳鑫宇[1]
1 郑州大学网络空间安全学院　郑州 450002
2 郑州大学计算机与人工智能学院　郑州 450001
(changjuancai@163.com)

摘　要　针对异步整形器(ATS)采用固定长度整形队列实现流量整形存在缓存资源利用率低、可调度流平均时延高等问题,提出了一种基于改进磷虾群算法与流量预测的可变长整形队列调整算法。综合考虑流的队列分配规则、有界时延需求及有限缓存资源,定义时间敏感网络中可调度流传输约束。引入混沌映射、反向学习与精英策略并设计自适应位置更新策略以提升传统磷虾群算法的求解能力,利用改进磷虾群算法寻找整形队列可调整上限。基于卷积神经网络与长短期记忆模型(CNN-LSTM)预测流量,根据预测值计算队列长度调整步幅。仿真结果表明,与采用固定长度整形队列的方法相比,所提算法能有效提高可调度流数量,降低调度流(ST)平均时延,并提升网络缓存资源利用率。

关键词:时间敏感网络;异步整形器;改进磷虾群算法;流量预测;可变长队列
中图分类号　TP393

Variable-length Shaping Queue Adjustment Algorithm in Time-sensitive Networks

CAI Changjuan[1],ZHUANG Lei[2],YANG Sijin[2],WANG Jiaxing[1] and YANG Xinyu[1]
1 School of Cyber Science and Engineering,Zhengzhou University,Zhengzhou 450002,China
2 School of Computer and Artificial Intelligence,Zhengzhou University,Zhengzhou 450001,China

Abstract　A variable length shaping queue adjustment algorithm based on an improved krill herd algorithm and traffic prediction is proposed to address the issues of low buffer resource utilization and high average delay of schedulable streams using fixed length shaping queues for traffic shaping in asynchronous traffic shaper(ATS). Considering the queue allocation rules of flows, bounded delay requirements,and limited buffer resources,transmission constraints for schedulable flows are defined in time-sensitive networks. The improved krill herd algorithm is used to find the maximum adjustable upper limit of the shaping queue,using a combination of chaos mapping,opposition-based learning,elite policy,and adaptive location update strategy to enhance the algorithm's solving ability. The traffic is predicted based on convolutional neural network and long short-term memory model(CNN-LSTM),and the queue length is calculated according to the predicted value to adjust the step. Simulation results show that compared with the method of using fixed-length shaping queues,the proposed algorithm can effectively increase the number of schedulable flows,reduce the average delay of scheduled traffic(ST),and improve the utilization rate of network buffer resources.

Keywords　Time-sensitive network, Asynchronous traffic shaper, Improved krill herd algorithm, Traffic prediction, Variable length queue

1　引言

随着全息投影、远程医疗、增强现实、无人驾驶等新兴时延敏感业务的出现,当前互联网需要满足大流量、低时延、高可靠、零丢包等服务质量需求[1-2]。传统以太网只提供尽力而为的服务,无法满足不断发展的互联网业务需求。因此,IEEE 802.1 工作组提出了时间敏感网络(Time Sensitive Network,TSN)[3]以弥补传统以太网的不足。通过流量整形、资源预留、帧抢占等链路层增强机制,TSN 具有强大的互联互通、高质量实时传输的能力,成为满足当前互联网需求的有效途径[4]。

TSN 在保留传统以太网的混合流量传输特点的同时,引入时间同步技术和流量整形机制,以满足实时性要求。其中,IEEE 802.1 Qcr 标准提出的异步整形器(Asynchronous Traffic Shaping,ATS)无需全局时钟同步,避免了时间感知整形器(Time Aware Shaper,TAS)单点时序失准造成的流量传输不可靠的问题;同时,ATS 无需计算复杂的门控列表,也不必保证数据的同步入队和出队,相较于 TAS 与循环排队转发

到稿日期:2023-05-29　返修日期:2023-10-10
基金项目:河南省重大科技专项(221100210900-03)
This work has supported by the Major Science and Technology Program of Henan Province(221100210900-03).
通信作者:庄雷(ielzhuang@zzu.edu.cn)

(Cyclic Queuing and Forwarding，CQF)机制具有更高的灵活性。ATS具有的上述特点使其在非周期性流量与周期性流量共存的TSN应用场景中能够有效利用网络资源，表现出优于TAS的性能[5]。

ATS通过在每个交换节点的出端口部署异步整形结构实现流量的可靠传输。从交换节点入端口进入的流量经过分类过滤后由流门控分配得到内部优先级，然后进入整形队列进行缓存，以便按序进入整形器开始交错整形。交错整形实则是为该流所包含的数据帧计算合格时间(Eligible Time，ET)，即期望的数据帧发送时间，用于后续的传输选择算法。通过整形器之后，数据帧根据内部优先级进入对应的共享队列排队以待转发。接下来，传输选择算法严格按照由高到低的队列优先级执行，若共享队列队首数据帧的合格时间小于当前时间，则直接转发队首数据，否则从下一优先级共享队列检查当前时间是否满足数据帧的合格时间需求。

综上，为了避免不同业务流之间互相影响，使流量传输满足时延需求，ATS利用整形队列存储经分类过滤后的流量，其数量随着接入交换机端口的终端数量的增加而增加[6]。然而，在现实应用中，端口的缓存资源大小通常是在满足性能需求的前提下尽可能小，以节约硬件成本。因此，在有限的缓存资源条件下，对ATS中的整形队列进行合理的资源分配，以最大化缓存资源利用率进而提升网络中可调度流数量，是当前TSN中需要解决的一个问题。

TSN中已有一部分工作针对流对应的队列分配问题进行了研究，如文献[7-8]。文献[7]指出队列资源的管理对实时流的可调度性具有重大影响，研究了TSN交换机中队列管理的理论基础，并基于此提出了一种实时流的队列分配算法。文献[8]阐述了ATS中流的队列分配规则并分析了流量的端到端时延，提出了一种异步流量调度方法。该文中使用的时延界限分析方法对后续ATS方面的研究具有重大参考意义。此外，文献[9-12]指出队列资源对网络性能的影响，研究了在固定长度整形队列的背景下提升网络性能的方法。文献[9]提出了基于网络演算的队列性能分析方法，寻找合适的队列长度值以优化网络系统设计性能和成本。该方法的局限性在于其以静态的方式部署整形队列资源，在运行中不再改变。文献[10]提出了一种基于多级队列的TSN技术，提高了网络中具有不同要求的时间敏感业务的传输能力。文献[11]基于观察数据包注入网络的开始时间对队列的利用率有重要影响资源，提出了一种注入时间规划(Injection Time Planning，ITP)机制来优化时间敏感流的网络吞吐量。Wang等[12]提出了基于时间敏感型软件定义网络(Time-Sensitive Software Defined Network，TSSDN)架构的TSN交换机队列长度预测模型，利用该模型实现数据流路由的动态规划以及网络负载均衡。他们通过预测队列长度来避免拥塞，在整体运行中不考虑对队列总容量进行修改。上述两类研究均是在固定整形队列长度的背景下展开进一步的研究，而文献[13]提出了一种弹性队列引擎策略，通过在硬件层面增加动态队列、动态门控列表和动态内部优先级的弹性设计，优化可用资源，满足队列动态运行时自适应网络状态，使TSN中的服务质量和资源可用性最大化。但该方法在硬件方面的成本将随着网络规模的扩大而增加。

综上所述，当前对TSN中队列研究的不足主要包括两方面：一方面，现有研究大多将队列资源作为实验参数设置[9-12]，通过设置不同的队列参数值验证队列资源对流调度的影响，缺乏动态性；另一方面，在异步时间敏感网络中，当前研究侧重于流的队列分配以及队列优先级分配的问题[7-8,14-16]，未考虑使用固定长度整形队列存在的资源利用率不高、可调度流的平均时延高等问题。因此，本文设计了一种基于改进磷虾群算法与流量预测的可变长队列调整算法，旨在提高可调度流数量，降低调度流量(Scheduled Traffic，ST)的平均时延并最大化缓冲资源利用率。该算法首先使用改进磷虾群算法计算整形队列可调整上限，以避免队列长度不加限制地增加导致队内排队时延增加，出现流不可调度的情况；然后使用CNN-LSTM模型预测ST流的流量值，使算法根据流量预测值计算队列长度调整的步幅值；最后根据队列可调整上限与队列长度调整步幅动态改变ATS中的整形队列长度。

2 背景与动机

2.1 异步整形器

ATS的排队结构延续了基于紧急度的调度器(Urgency-Based Scheduler，UBS)的设计[17]，其两级排队结构包含整形队列和共享队列。其中，整形队列用于存储待交错整形的流；共享队列则用于存储经整形器整形后，拥有相同出端口和内部优先级且来自不同整形队列的流，并对队内数据采用“先来先服务”的原则传输。多个共享队列严格按照优先级的方式进行传输。在ATS中，流对应的整形队列的分配遵循以下规则[18]：QAR1，每个整形队列只与一个入口端口关联；QAR2，每个整形队列只与前一跳的ATS中的一个优先级关联；QAR3，每个整形队列只与一个内部优先级关联。

对于一个N端口的交换机，至少需要$N-1$个整形队列来保证QAR1约束，该下限数量是一个针对设备的常量，随端口的数量而变化[8]。若实现具有P个内部优先级，则从N个入口端口接收单一优先级流量所需的整形队列数至少是$N \cdot P$个[19]，该数量与接入交换机的终端设备数及整形结构内部优先级数成正比。考虑到入口端口能以多个优先级发送流量，网络中所需的整形队列数量将快速增加。

本文采用的ATS排队结构是基于文献[8]和文献[16]的，即整形队列仅仅对ST流进行整形，BE流不参与整形，直接进入隐式队列排队。整体结构如图1所示。

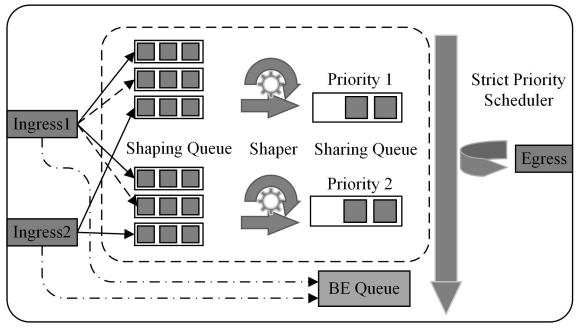


图1 多优先级发送ST流

Fig. 1 Send ST streams in multiple priorities

图1表明交换机入口端口Ingress1以两种优先级发送

ST 流，入口端口 Ingress2 以单一优先级发送 ST 流，出口端口 Egress 的整形结构有两个内部优先级所需要的整形队列数。

2.2 动机

先前的研究[6,16]表明，随着接入交换机的终端数量的增加，ATS 所需要的整形队列数量将快速增加。为大量的整形队列分配有限的缓存资源成为了 ATS 中急需解决的问题。此外，为了避免队列长度过大而引起的网络高延迟现象，或队列长度过小导致的大量丢包情况，应该为每个整形队列设置合适的整形队列长度，从而实现整体缓存资源利用率与可调度流数量的提升。考虑图 2 所示场景，图中左侧由于设置的整形队列容量过大，数据包排队时延增加，因此流不可调度（图 2 左侧黑色的方块）。图中右侧由于设置的整形队列容量过小，大量数据包被丢弃。因此，为了使整形队列能够根据网络情况调整所需的缓存资源，本文设计了一种基于改进磷虾群算法与流量预测的可变长队列的调整算法。该算法根据预测的流量值与队列可调整上限为整形队列动态分配缓存资源，以提升网络资源利用率与可调度流数量，并减少 ST 流的平均时延。

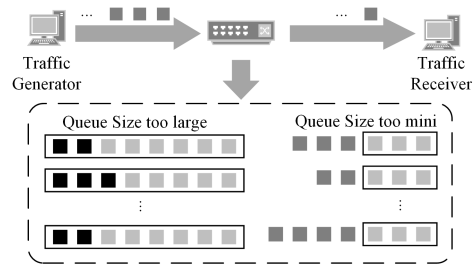


图 2 不合适的队列分配场景图

Fig. 2 Inappropriate queue allocation scenario

3 问题描述

本文考虑的网络结构是由终端发送流量，交换机转发流量，且流量的接收方不与该终端挂载在同一交换机上。

定义 TSN 网络为有向图 $G=(V,E)$，V 为网络中的终端（End Station，ES）和交换机（Switch，SW）集合，E 为网络中终端到交换机或交换机到交换机之间的链路集合。对于任意交换机，将其定义为六元组，$\theta_{sw}=(IN,OUT,Qnum,IPV_{\text{num}},P_{\text{level}},Qsize_{\text{num}})$。其参数含义如表 1 所列。

表 1 交换节点参数

Table 1 Parameters of switch node

参数	含义
IN	接入 sw 入口端口的终端集合
OUT	接入 sw 出口端口的终端集合
Q_{num}	sw 出口端口的整形队列的数量
IPV_{num}	sw 出口端口的整形结构的内部优先级数量
P_{level}	sw 入口端口可发送的流的不同优先级数量
$\boldsymbol{Qsize}_{\text{num}}$	sw 中的整形队列长度矩阵

为了满足 ATS 中流的队列分配规则，对于任意交换机，其出口端口内的整形队列数应满足式(1)。

$$Q_{\text{num}}[j]=\sum_{i=1}^{|IN|}P_{\text{level}}[i]\cdot IPV_{\text{num}}[j] \tag{1}$$

其中，$Q_{\text{num}}[j]$表示第 j 个出口端口的整形队列数量，$P_{\text{level}}[i]$表示入口端口 i 能以多少个不同优先级发送流，$IPV_{\text{num}}[j]$表示出口端口 j 的整形结构中的内部优先级数。

定义网络中的流量集合为 F，其包括了非周期性确定性流集合 F_{ST} 和尽力而为流集合 F_{BE}。对任意的 ST 流，将其定义为六元组，$f=(stime_v,etime_v,pnum,D_f,priority_s,id)\in F_{\text{ST}}$，其参数含义如表 2 所列。

表 2 ST 流参数

Table 2 Parameter of ST streams

参数	含义
$stime_v$	流在节点 $v\in V$ 开始传输的时间
$etime_v$	流在节点 $v\in V$ 结束传输的时间
$pnum$	流的发包数量
D_f	流的最大端到端可容忍延迟
$priority_s$	流在端口 s 分配的内部优先级
id	流的标识

对任意终端发送的 BE 流，将其定义为四元组，$f=(stime_v,etime_v,pnum,id)\in F_{\text{BE}}$，其参数含义与 ST 流参数类似，但不为 BE 流单独设置端到端最大可容忍时延及优先级。任意数据包 $p_k(0<k\leqslant f.pnum,f\in F)$由二元组构成，$p_k=(f.id,size)$。其中，$size$ 代表数据包的大小，$f.id$ 标识了数据包所属流。

假设每个终端发送的流 $f_i\in F$ 所含数据包流量满足泊松分布，并受漏桶模型约束上限，则对于指定的时间间隔 d，该模型保证流 f_i 进入网络的累计输出数据包 $W_i(d)$满足式(2)。

$$W_i(d)\leqslant\hat{b}_i+d*\hat{r}_i \tag{2}$$

其中，漏桶率$\hat{r}_i$和突发度$\hat{b}_i$可以根据终端 ES 输出流的持续时间、流包含的数据包数量及数据包大小确定，如式(3)、式(4)所示：

$$\hat{r}_i=\frac{\sum_{k=1}^{f_i.pnum}p_k.size}{f_i.etime_{\text{ES}}-f_i.stime_{\text{ES}}} \tag{3}$$

$$\hat{b}_i=\left(1-\frac{\hat{r}_i}{r(s)}\right)*\left(\sum_{k=1}^{f_i.pnum}p_k.size\right) \tag{4}$$

其中，$r(s)$为端口转发速率。

针对上述网络结构和本文使用的 ATS 排队结构，可调度流的约束和本文的目标函数如下：

1)队列分配约束

为了隔离混合流量并实现服务质量保障，对于任意终端发出的不同 ST 流 f_i 和 f_j，其所分配的对应队列满足上文所述的 ATS 流的队列分配规则。该约束定义为：

$$\begin{aligned}&\text{QAR1: if } s_-(f_i)\neq s_-(f_j)\\&\qquad\text{then } q(s,f_i)\neq q(s,f_j)\\&\text{QAR2: if } s_-(f_i)=s_-(f_j)\\&\qquad\text{but } f_i.priority_{s_-(f_i)}\neq f_j.priority_{s_-(f_j)}\\&\qquad\text{then } q(s,f_i)\neq q(s,f_j)\\&\text{QAR3: if } f_i.priority_s\neq f_j.priority_s\\&\qquad\text{then } q_s(s,f_i)\neq q_s(s,f_j)\end{aligned} \tag{5}$$

其中，$s_-(f)$为流 f 相对于当前端口的上游端口，$s_+(f)$为流 f 相对于当前端口的下游端口，$q(s,f)$为流 f 在端口 s 内分配的整形队列，$q_s(s,f)$为流 f 在端口 s 分配的共享队列。

2）时延约束

本文的时延约束主要考虑 ATS 中流的队列分配及队列优先级分配所经历的最大时延与传输路径的链路时延。对于每一条输入流，为了保证可调度，上述两类时延之和应小于或等于端到端的最大可容忍时延。该约束定义为：

$$D_{ST}(x,f,y)\leqslant f.D_f \tag{6}$$

$$D_{BE}(x,f,y)\leqslant d_{BE} \tag{7}$$

式(6)保证 ST 流的可调度时延不超过最大可容忍时延；式(7)保证 BE 流的可调度时延不超过 d_{BE}。其中，$D_{ST}(x,f,y)$与 $D_{BE}(x,f,y)$分别表示 ST 流与 BE 流从源 x 到达目的接收器 y 所经历的端到端延迟。$D_{ST}(x,f,y)$由 4 部分构成，如式(8)所示：

$$D_{ST}(x,f,y)=d_{ST}(x,f,s_1)+d_{ST}(s_n,f,y)+\sum_{k=1}^{n-1}d_{ST}(x,f,s_k,s_{k+1})+\sum_{h=1}^{n-1}d_{link}^{h} \tag{8}$$

其变量含义分别如下：

$d_{ST}(x,f,s_1)$：表示流 f 从流量发送方 x 到达第一个交换节点中端口 s_1 的延迟，该延迟取决于终端站的实际物理实现，不受流到队列和队列到优先级的影响，可以认为 $d_{ST}(x,f,s_1)=0$。

$d_{ST}(s_n,f,y)$：表示流 f 从最后一个交换节点的端口 s_n 到达流量接受方 y 的延迟。该延迟受 3 种类型流量的影响，分别是 F_H、F_E，以及在高优先级 ST 流传输前正在传输的 BE 流或低优先级 ST 流，其定义如式(9)所示：

$$d_{ST}(s_n,f,y)=\frac{\sum_{k=1}^{f.pmum}p_k.size}{r(s_n)}+\frac{\sum_{g\in F_H\cup F_E\setminus\{f\}}\sum_{k=1}^{g.pmum}p_k.size+\hat{l}_{F_L}}{r(s_n)-\sum_{g\in F_H}\hat{r}(g)} \tag{9}$$

其中，F_H 表示与流 $f\in F$ 处于同一出端口，共享队列优先级高于流 f 所处共享队列优先级的所有流集合；F_E 表示与流 f 在同一出端口，共享队列优先级低于流 f 所处共享队列优先级的所有流的集合；F_L 表示与流 f 在同一个共享队列且排队在其之前的所有流集合；$\hat{l}_{F_L}$ 表示低优先级队列中长度最大的一个数据流长度；$r(s_n)$代表端口 s_n 转发速率，$\hat{y}(g)$代表流 g 的漏桶率。

$d_{ST}(x,f,s_k,s_{k+1})$：表示相邻近交换节点的端口之间的转发延迟，定义如式(10)所示：

$$d_{ST}(x,f,s_k,s_{k+1})=\max_{g\in F_q(s_{k+1},f)}(d_{ST}(s_k,g,y)) \tag{10}$$

其中，$F_{q(s_{k+1},f)}$表示与流 f 在下游端口有相同的整形队列的流集合。

d_{link}^{h}：表示传输路径的链路时延，该时延由传输路径所包含的跳数、链路带宽、链路路径长度共同决定。

$D_{BE}(x,f,y)$的构成与 $D_{ST}(x,f,y)$相似，仅 $d_{BE}(s_n,f,y)$有所不同，定义为：

$$d_{BE}(s_n,f,y)=\frac{\sum_{k=1}^{f.pmum}p_k.size}{r(s_n)}+\frac{\sum_{g\in F_H\cup F_E\setminus\{f\}}\sum_{k=1}^{g.pmum}p_k.size}{r(s_n)-\sum_{g\in F_H}\hat{y}(g)} \tag{11}$$

由于 BE 流所处的队列优先级最低，在 BE 流传输前，不存在比 BE 流优先级还低的流正在传输。因此，在式(11)中不考虑 $\hat{l}_{F_L}$ 参数。

3）队列资源约束

对于网络中任意交换机的任意端口，其可用缓存资源有限。该约束保证对于任意 ST 流和 BE 流，其占用的缓存资源不超过该端口总的可用缓冲资源，如式(12)所示：

$$\sum_{q=1}^{Qnum[s]}Q_{s,ST,q}+Q_{s,BE}\leqslant Q_{s,max} \tag{12}$$

其中，

$$Q_{s,ST,q}=\sum_{f_j\in F_{q(s,f)=q}}\sum_{k=1}^{f_j.pmum}p_k.size$$

$$Q_{s,BE}=\sum_{f_j\in F_{BE}}\sum_{k=1}^{f_j.pmum}p_k.size$$

$Q_{s,ST,q}$定义为端口 s 的整形队列 q 占用的缓存资源，$Q_{s,BE}$表示端口隐式队列占用的缓存资源，$Q_{s,max}$为端口总的缓存资源。$F_{q(s,f)=q}$表示在当前端口所分配的整形队列为 q 的流集合。

在满足上述约束的前提下，本文的目标是使网络中可调度流数量增加，ST 流可调度时延减小，缓存资源利用率提升。首先，定义端口 s 的缓存资源利用率为端口所有整形队列占用的缓存资源与隐式队列占用的缓存资源之和与端口总缓存资源之比，即

$$Utilization(s)=\frac{\sum_{q=1}^{Qnum[s]}Q_{s,ST,q}+Q_{s,BE}}{Q_{s,max}} \tag{13}$$

然后，使用 δ_i 和 η_j 两个布尔变量分别表示 ST 流与 BE 流的调度结果，定义为

$$\delta_i=\begin{cases}1, & D_{ST}(x,f_i,y)\leqslant f_i.D_f\\ 0, & \text{other}\end{cases} \tag{14}$$

$$\eta_j=\begin{cases}1, & D_{BE}(x,f_j,y)\leqslant d_{BE}\\ 0, & \text{other}\end{cases} \tag{15}$$

式(14)与式(15)分别表示 ST 流与 BE 流的端到端时延在满足约束(2)时，标记布尔变量 δ_i 和 η_j 为 1，表示该流可调度；否则标记为 0，表示该流不可调度。

综合上述约束及分析，为了实现本文的目标，定义目标函数 M 为：

$$M=\max(\sigma_1\cdot M_1+\sigma_2\cdot M_2+\sigma_3\cdot M_3) \tag{16}$$

$$M_1=\sum_{f_i\in F_{ST}}a\cdot\delta_i+\sum_{f_j\in F_{BE}}b\cdot\eta_j \tag{17}$$

$$M_2=\sum_{f_i\in F_{ST,\sigma}}\frac{1}{D_{ST}(x,f_i,y)}+\sum_{f_j\in F_{BE,\eta}}\frac{1}{D_{BE}(x,f_j,y)} \tag{18}$$

$$M_3=\sum_{sw\in SW}\sum_{s=1}^{|sw.OUT|}\sum_{q=1}^{sw.Qnum[s]}Utilization(s) \tag{19}$$

其中，σ_1，σ_2，σ_3 为权重系数，根据需求调整。本文的首要目标是调度更多的流，其次是减少可调度流的平均时延，最后是提高资源利用率，因此在目标函数 M 中设置 $\sigma_1>\sigma_2>\sigma_3$。目标函数分量 M_1 表示可调度流数量，M_2 表示可调度流时延，M_3 表示全网缓存资源利用率。在目标函数分量 M_1 中设置权重

系数 $a>b$，即尽可能多地调度 ST 流。$F_{ST,\sigma}$ 为可调度的 ST 流集合，$F_{BE,\eta}$ 为可调度的 BE 流集合。

4 算法设计

本章由两部分构成，第一部分使用改进的磷虾群算法计算整形队列的可调整上限，以避免队列长度调整过大，出现网络整体传输时延过高的现象，即 BufferBloat[20] 现象。第二部分使用 CNN-LSTM 模型对 ST 流进行流量预测，并使用流量预测值计算队列长度的调整步幅；然后根据队列可调整上限与单次调整步幅实现可变长队列调整算法。

4.1 改进磷虾群算法 COEAKH

磷虾群算法(Krill Herd，KH)在解决优化问题方面优于一些著名的群智能算法[21]，如遗传算法、粒子群算法、差分进化算法、人工蜂群算法、蚁群算法等。KH 擅长探索解空间，定位有可能具有全局最优值的区域，但不擅长开发新区域，易陷入局部最优。因此，本文提出一种改进的磷虾群算法 COEAKH，改进思路为：初始化种群时，使用均匀分布代替随机分布，以提高种群质量；位置更新过程中，使用精英策略结合自适应的位置更新策略，跳出局部最优并获得更优解。算法首先使用 Tent 映射结合反向学习初始化种群，提高寻优前期种群的质量。然后在磷虾个体位置更新时使用精英策略，保留精英个体，对非精英个体进行交叉变异以获得更优秀的个体。为了提高算法跳出局部最优解的能力和局部搜索能力，本文增加了自适应的位置优劣判断因子，根据判断结果选择使用莱维飞行策略或高斯游走方法更新位置，以扩大搜索区域范围，提高种群局部搜索能力。

算法通过不断更新个体位置，优化目标函数，求出该整形队列针对应用场景的队列长度上限。算法具体步骤如下：

1)初始化。使用 Tent 映射与反向学习初始化种群，提升寻优前期的整体种群质量。X_i 为初始化第 i 只磷虾的位置，UB 和 LB 分别代表解的上下限。初始化步骤如下：

步骤 1 随机生成初始值 X_0'

步骤 2 利用 Tent 映射函数生成混沌序列数

$$X'_{i+1}=\begin{cases}2X_i', & 0\leqslant X_i'<1/2\\ 2(1-X_i'), & 1/2\leqslant X_i'\leqslant 1\end{cases} \tag{20}$$

步骤 3 将生成的混沌序列数映射到真实解空间

$$X_i=LB+(UB-LB)*X_i' \tag{21}$$

步骤 4 生成反向解

$$X_i^*=UB+LB-X_i \tag{22}$$

步骤 5 将适应度好的个体组成初始种群

$$X_i=\{X_i^*\mid Fitness(X_i^*)<Fitness(X_i),i\in N\}\bigcup\{X_i\mid Fitness(X_i^*)>Fitness(X_i),i\in N\} \tag{23}$$

2)适应度函数。本文综合考虑流的可调度时延及流的丢包量，定义适应度函数为：

$$Fitness=\sum_{f\in F_{x(f)=q}}\varepsilon\cdot(D_{ST}(x,f,y)-f.D_f)+\zeta\cdot Drop_f \tag{24}$$

其中，ε,ζ 为比例系数，设 $\varepsilon=\zeta$，即选择既不会导致过度缓存也不会导致大量丢包的合适长度作为队列值上限。$F_{q(s,f)=q}$ 表示在当前端口所分配的整形队列为 q 的流集合，$Drop_f$ 为流 f 的丢包数。

3)速度及位置更新。计算适应度后，磷虾个体按照式(25)更新速度，然后对磷虾群使用精英策略，将非精英个体分别按照式(26)和式(27)进行交叉变异。

$$\frac{dX_i}{dt}=N_i+F_i+D_i \tag{25}$$

$$X_i=\begin{cases}X_r, & rand()<0.2*\hat{K}_{i,\text{best}}\\ X_i, & \text{else}\end{cases} \tag{26}$$

$$X_i=\begin{cases}X_{\text{best}}+\chi(X_p-X_q), & rand()<\dfrac{0.05}{\hat{K}_{i,\text{best}}}\\ X_i, & \text{else}\end{cases} \tag{27}$$

式(25)表明磷虾个体速度的更新受 3 部分影响，分别是邻域内的磷虾的引导 N_i、食物位置的引导 F_i，以及随机的物理扩散 D_i。式(26)与式(27)均是交叉变异算子，其中 $\hat{K}_{i,\text{best}}$ 表示第 i 只磷虾的适应度与当前具有最佳位置的磷虾的适应度之差在最坏适应度值与最佳适应度值之间的映射，X_{best} 为当前适应度最好的个体，$rand()$ 函数为随机数生成函数，χ 为权重系数，r,p,q 为随机数。

为了使磷虾个体分布接近最优解，改进的算法设计了自适应的位置优劣判断因子，如式(28)所示：

$$Fitness_i^t-Fitness_{\text{best}}^t>\omega\cdot|Fitness_{\text{avg}}^t-Fitness_{\text{best}}^t| \tag{28}$$

其中，$Fitness_i^t$，$Fitness_{\text{avg}}^t$，$Fitness_{\text{best}}^t$ 分别表示磷虾个体在第 t 次迭代中的适应度值、种群平均适应度值，以及具有最佳位置的磷虾个体的适应度值。ω 为自适应调节参数，定义为 $\omega=\log_{0.5}(I/I_{\max})$，即随着迭代次数的增加而减少，目的是在搜索前期，尽可能地探索未知区域；到搜索后期，对邻域进行仔细搜索。I 和 $I_{\max}$ 分别对应当前迭代次数和最大迭代次数。

对于满足式(28)的磷虾个体，按照适应度值降序排列，将前 K 个的磷虾位置按照式(29)更新位置，其余的则按照传统位置更新公式，即式(30)更新位置。不满足上述公式的磷虾个体，其位置更新使用式(31)。

$$X_i(t+\Delta t)=X_i(t)+\Delta t\frac{dX_i}{dt}+\frac{A}{I^2}\otimes Levy(\lambda) \tag{29}$$

$$X_i(t+\Delta t)=X_i(t)+\Delta t\frac{dX_i}{dt} \tag{30}$$

$$X_i(t+\Delta t)=X_i(t)+\Delta t\frac{dX_i}{dt}+normrnd() \tag{31}$$

式(29)与式(31)在保留传统磷虾群位置更新特点的同时，分别加入了莱维飞行因子与高斯游走因子，以增强跳出局部最优解的能力和对局部更加细致搜索的能力。其中，式(29)中的 A 为莱维飞行最大步长控制量，$Levy(\lambda)$ 为莱维随机搜索路径。式(31)中 $normrnd()$ 为服从高斯分布的随机数生成函数。

计算位置更新后的适应度值 $Fitness_i'$ 与原适应度值 $Fitness_i$ 的优劣，若优于原值，则确定新位置，否则磷虾位置不变更。

若磷虾位置迭代相应次数后仍未更新，则根据概率选择采用莱维飞行更新位置，选择标准为：

$$\psi=1-\exp(-|Fitness_i^t-Fitness_{best}^t|) \tag{32}$$

若生成的随机数小于 ψ，则使用式(31)更新，否则使用式(29)更新，并比较更新位置后的适应度值与原值，优于则更新位置，否则不更新。每迭代一次，将适应度最差的 K 个磷虾个体的位置用适应度最优的 K 个磷虾个体的位置替换，以保证下一代种群的适应度优于当前种群的适应度。算法伪代码如算法 1 所示。

算法 1 COEAKH 算法

输入：解空间下限 LB，解空间上限 UB，精英个数 K，种群规模 N，最多允许的未更新次数 M，最大迭代次数 I_{max}

输出：整形队列可调整的上限值 q_{limit}

1. 生成混沌序列的初始随机数，$X_0'=$ rand()
2. 根据式(20)－式(23)初始化
3. 使用式(24)计算初始种群的个体适应度
4. While I<I_{max} do
5. 根据式(25)更新所有磷虾速度
6. 使用式(24)计算速度更新后的所有磷虾适应，并降序排列
7. for j=0 to N－K do
8. 对非精英个体按照式(26)和式(27)进行交叉变异
9. 使用式(24)计算经过交叉变异后的磷虾适应度
10. end for
11. 将磷虾群重新按照适应度降序排列，记录最佳适应度的个体位置为 X_{best}，种群的平均位置为 X_{avg}
12. for i=0 to N do
13. if Fitness(X_i)－Fitness(X_{best}) > $\omega L_{N,s}^i=[l_1,l_2,l_3,\cdots,l_n]$| Fitness($X_{avg}$)－Fitness($X_{best}$) |
14. 对前 K 个磷虾按照式(29)更新位置
15. 将剩下的 N－K 个磷虾按照式(30)更新位置
16. else
17. 按照式(31)更新位置
18. end if
19. 若新的适应度优于原适应度，则更新位置；否则不更新
20. if 磷虾 i 位置未更新的次数大于 M
21. if rand() > 1－exp(－| Fitness(X_i)－Fitness(X_{best}) |)
22. 按照式(29)更新位置
23. else
24. 按照式(31)更新位置
25. end if
26. end if
27. 将当前适应度最差的 K 个磷虾位置用适应度最优的 K 个磷虾位置替换
28. end while

通过 COEAKH 算法，可以计算出贴合应用场景的队列可调整上限，使整形队列长度在满足流量传输需求的前提下尽可能小，从而避免队列资源动态调整过大而导致流量不可调度。

4.2 流量预测结构与可变长队列调整算法

本文提出的整形队列调整算法根据 ST 的流量预测值计算队列长度调整的步幅，然后结合队列可调整上限实现整形队列长度的动态调整。因此，在计算可调整上限之后，需要建立多因素的 CNN-LSTM 模型以预测流量值。

定义一个根据时间戳排序的待提取特征的数据为：

$$L_{N,s}^i=[l_1,l_2,l_3,\cdots,l_n]$$

它表示第 N 个交换机中第 s 个端口中第 i 个整形队列在 T 时间内的数据向量，l_k($1\leqslant k\leqslant n$)表示第 k 条 ST 流的相关数据，每一条数据是一个 m 维的向量，使用 m 维向量表示。

$$l_k=[e_1,e_2,e_3,\cdots,e_m]$$

该向量表示到达某一队列的 ST 流可能受这 m 个因素影响。将该信息矩阵作为 CNN 结构的输入，经历多个隐藏层进行特征提取和映射后，得到关键影响因素作为输出。其中，隐藏层利用卷积层和稀疏连接的线性变换实现特性提取，通过激活函数的非线性变化实现特征映射，使用池化层和 DropOut 层减少模型参数，防止过拟合。卷积层和池化层的数量可根据需求定制。

LSTM 将 CNN 输出的特征序列作为输入，通过在细胞中添加门结构实现对历史信息的遗忘和添加新的信息。其中，遗忘门决定哪些信息应该丢弃，输入门用于更新细胞状态，输出门将状态值经过衰减后作为下一时刻的输出，结构如图 3 所示。

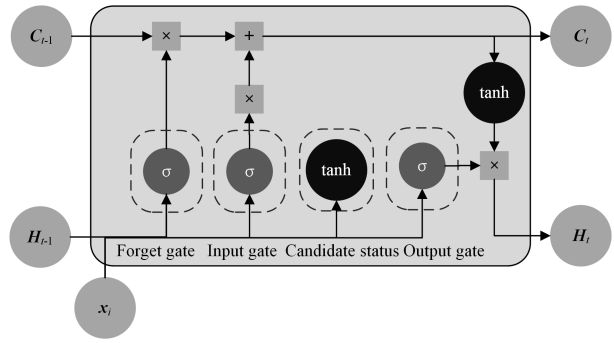


图 3 LSTM 单元结构图

Fig. 3 LSTM cell structure

记每个时间步的输入和输出为 x_t 和 H_t，状态为 C_t。当前记忆细胞的状态 C_t 根据上一时刻的输出 H_{t-1}、当前输入 x_t 及上一时刻的状态 C_{t-1} 更新，如式(33)所示：

$$C_t=F_t\odot C_{t-1}\oplus I_t\odot\widetilde{C}_t \tag{33}$$

其中，F_t 与 I_t 分别为遗忘门函数及输入门函数对 x_t 和 H_{t-1} 作用后的结果；$\widetilde{C}_t$ 为当前时刻产生的备选状态。t 时刻的输出如式(34)所示，其中 O_t 为输出门函数对 x_t 和 H_{t-1} 作用后的结果。

$$H_t=O_t\odot\tanh(C_t) \tag{34}$$

在模型训练时，使用固定的时间步长 t，并定义数据$[x_1,x_2,x_3,\cdots,x_t]$与$[x_{t+1},x_{t+2},x_{t+3},\cdots,x_{2t}]$分别表示输入数据与真实数据。根据时间步长为 t 的流量序列提供的历史信息预测下一时间段内涌入某交换机中某队列的流量总量。在测试数据集时，使用均方误差(MSE)作为模型的损失函数，其数学表达式如式(35)所示：

$$MSE=\frac{1}{n}\sum_{i=1}^{n}(y_i-\hat{y}_i)^2 \tag{35}$$

利用 MSE 值的计算得到梯度下降解，并使用 Adam 进行梯度下降优化，最后通过学习得到网络各个参数的值。

通过流量预测模型得到下一时间段内到达某整形队列的ST流数据量 D_{next}。然后，利用预测的流量值与可调整上限，实现基于改进磷虾群算法与流量预测的可变队列调整算法。算法定义 $\zeta=Max_{data}/(T\cdot B_w)$，$\theta=D_{next}/B_w$，其中偏差值 ζ 表示在 T 时间内发送 Max_{data} bit 数据需要占用的带宽资源比例，度量值 θ 表示下一时间段内 ST 流的流入流出比。当 $\theta>1+\zeta$ 时，表示下一时间段 ST 流到达的数据量过多，为了避免队列长度不足而导致的不必要丢包，使用式(36)计算队列长度增加的幅度；当 $\theta<1-\zeta$ 时，表示下一个时间段 ST 流到达的数据量过少，资源的利用率较低，使用式(37)计算队列长度减少的幅度；当 $1-\zeta<\theta<1+\zeta$ 时，到达的流量合适，无需调整队列大小。

$$Q_{up}=\min(q_{limit}-Q_{current},D_{next}-Q_{current}) \tag{36}$$

$$Q_{down}=\max(1500,\frac{(Q_{current}-D_{left}-D_{next})}{2}) \tag{37}$$

所有调整后的队列长度的最小值不低于 1500B，最大值不超过队列可调整上限 q_{limit}。$Q_{current}$ 代表当前队列长度，D_{left} 代表截止到队列长度修改时还滞留在队列里的数据量。

算法 2 可变长队列调整算法

输入：当前队列长度 $Q_{current}$，当前队内数据量 $D_{current}$，时间间隔 T，带宽 B_w，网络负载 ρ_L，流量预测值 D_{next}，队列可调整上限 q_{limit}=COEAKH()

输出：下一时间段的队列长度 Q_{next}

1. if $t_{now}\%T=0$ do
2. $D_{left}=D_{current}-T\cdot B_w$
3. 计算 $\zeta=\frac{\rho_L\cdot B_w}{T}$，$\theta=\frac{D_{next}}{B_w}$
4. 分别按照式(36)、式(37)计算 Q_{up}，Q_{down}
5. if $\theta>1+\zeta$ do
6. $Q_{next}=Q_{current}+Q_{up}$
7. else if $\theta<1-\zeta$ do
8. $Q_{next}=Q_{current}-Q_{down}$
9. else
10. $Q_{next}=Q_{current}$
11. end if
12. end if

经过流量预测后，预测结果作为算法输入，根据当前队列长度、时间间隔和带宽、计算出截止到队列调整时还滞留在队内的数据量(第 2 行)。然后，分别计算度量值、偏差值以及上调幅值与下调幅值(第 3－4 行)。接着，判断流量对当前队列长度而言的大小，若流量相对较大，则按照式(第 5－6 行)调整队列长度；若流量相对较小，则按照式(第 7－8 行)调整队列长度；否则不调整(第 9－10 行)。

5 实验分析

5.1 实验设置

本文采用的仿真实验平台是 OMNeT++[22]，在该平台上使用 INET 框架[23]构建本文的仿真环境，结合 Python 实现本文所提算法。OMNeT++是一个模块化的、可扩展的、基于组件的 C++仿真库和框架，允许快速网络原型制作并提供多种用于状态检查和结果评估的工具。INET 是 OMNeT++仿真环境的开源模型库，它除了提供互联网堆栈协议、有线和无线链路层协议，还实现了 TSN 中多种整形器功能及多种队列模型。

1)网络拓扑

本文利用 INET 框架内包含的 TsnSwitch 组件在仿真平台搭建了环形拓扑作为实验拓扑，如图 4 所示。每个 TsnSwitch 连接着一个终端节点，该节点包含一个流量产生器 Talker，由 INET 框架中的 ActivePacketSource 模块以及一个利用 PassivePacketSink 模块实现的流量接收模块 Listener 实现。由节点的流量产生模块产生满足泊松分布的流量，该流量的目的节点不与该节点挂载在同一交换机上。TsnSwitch 之间的链路为全双工以太网链路，设置带宽 B_w 为 1 Gbit/s，链路延迟 d^h_{link} 设为 0.5 μs。每个 TsnSwitch 的出端口的总缓存大小设置为 1024 kB。利用 INET 的 Queue 模块在 TsnSwitch 中构建多个初始容量相同的加权轮询(Weighted Round-Robin，WRR)队列作为整形队列，严格优先级(Strict Priority，SP)队列作为共享队列。其中，出端口的整形队列所分配的缓存资源之和为 256 kB，隐式队列所分配的缓存资源为 256 kB，共享队列所分配的缓存资源之和为 512 kB[5]。

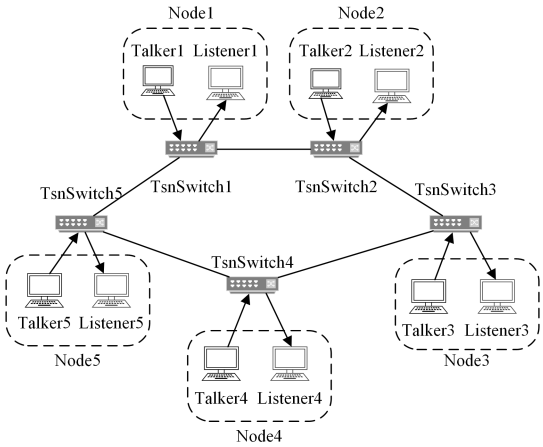


图 4 实验拓扑图

Fig. 4 Experimental topology

2)流量特性

从主机中随机选择流量发送方和接收方，采用最短路径算法传输数据包。ST 流和 BE 流根据独立的泊松过程产生，单个数据包长度在[64 Byte，1500 Byte]之间随机产生[5]。在本文实验中，仅考虑 ST 流和 BE 流，其相关参数如表 3 所列。流量强度通过相对流量负载来表征，即相对于链路传输的比特率。例如，强度 $\rho_L=0.2$ 对应于通过 5 个源节点注入网络的总比特率为 0.2×1 Gbps，由此，每个源节点分别贡献总负载的 1/5。

表 3 流量参数

Table 3 Traffic parameters

流量类型	单流数据量/Byte	最大可容忍时延/ms	流量占比/%
ST 流	[1500,4500]	2	30
BE 流	[400,4500]	10	70

3）群智能算法参数

为了验证本文 COEAKH 算法的有效性，选择与 KH[24]，IKH[25]，LKH[26]进行比较。4 种算法的共同参数设置相同，除了 COEAKH 采用混沌映射结合反向学习初始化种群外，其余算法均以相同的方式生成初始种群。KH，IKH，LKH 分别根据文献[24-26]设置除共同参数外的其余相关参数。COEAKH 设置最大 Levy 步长 $A=10$，精英个数 $K=10$。

4）神经网络模型参数

模型使用 CNN-LSTM 架构，卷积层使用 32 个过滤器和 relu 激活函数，Dropout 参数为 0.3；LSTM 层的记忆元分别为 10 和 20，它们均采用 tanh 激活函数；最后的全连接层使用 relu 激活函数。模型的编译采用 Adam 优化器，并选择 MSE 作为损失函数。

5）队列参数

本文设置交换机每个出口端口的内部优先级有两个，入口端口以两个不同的优先级发送 ST 流。在可变长队列的情况下，隐式队列长度固定为端口总有缓存资源减去该端口所有整形队列上限之和后的值；在固定长度队列的情况下，隐式队列长度为端口总有缓存资源减去该端口整形队列长度之和后的值。实验中的固定队列长度指交换机中所有整形队列长度一致且不随着网络的运行动态改变。Q_1，Q_2，Q_3 分别为长度等于 $\sum q_{\text{limit}}/2n$，$\sum q_{\text{limit}}/n$，$\sum 2q_{\text{limit}}/n$ 的固定长度队列，n 为端口的整形队列数；Q_{dynamic} 表示基于流量预测的可变长队列，初始部署为 $q_{\text{limit}}/2$，剩余资源充入 Buffer 中以便共享。

5.2 对比实验分析

首先，对比了 COEAKH 与 KH，LKH，IKH 的收敛及寻优能力，结果如图 5 所示。与其他算法相比，COEAKH 由于利用 Tent 映射和反向学习初始化种群，减少了算法受初始种群分布的影响，提升了收敛速度；自适应位置更新策略的引入，使算法具有跳出局部最优解及增强局部搜索的能力，提高了寻优精度。利用 COEAKH 能够寻找到更加满足网络情况的队列长度，将该值作为整形队列可调整上限能够保证流传输需求，同时避免排队时延过大或大量丢包，为后续的队列调整算法提供了基础。

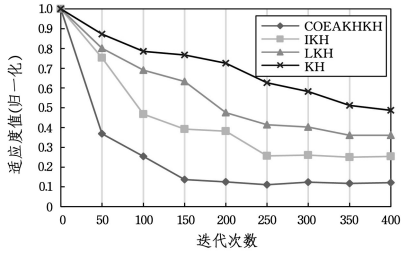


图 5 不同算法的收敛能力与最优适应值

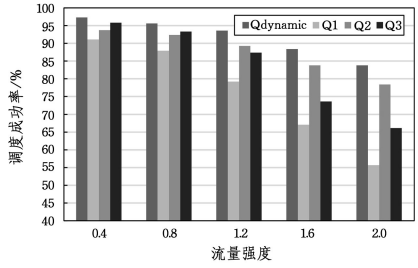
Fig. 5 Convergence capability and optimal fitness value of different algorithms

其次，比较了采用本文所提的队列长度调整算法的可变长整形队列与固定长度整形队列在可调度流数量、ST 流平均时延以及网络缓存资源利用率方面的性能。为了便于表述，下文统称为可变长队列与固定长度队列。

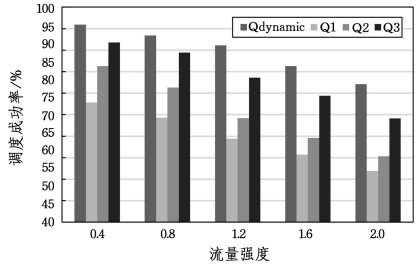
1）可调度流数量

在不同流量强度的情况下，比较使用可变长队列与固定长度队列成功调度的 ST 流与 BE 流数量，结果如图 6 所示。

图 6(a)和图 6(b)表明，采用可变长队列将直接影响 ST 流可调度数量，间接提升 BE 流可调度数量。这是因为当预测出到达流量不大时，利用本文的可变长整形队列调整算法可以动态减小队列长度，实现 ST 流的快速转发，从而调度更多的 ST 流并为 BE 流留出相对较多的调度时间；在预测下一时段到达流量大时，可变长队列在可调整范围内适当增加队列长度，以存储更多原本可以调度却因为队列长度不足而丢弃的流数据包。由于使用 COEAKH 算法计算了更加贴合场景的可调整范围的上限，因此队列长度不会增至过大，流仍然可以在可容忍时延范围内尽快处理。



(a)ST streams



(b)BE streams

图 6 不同队列分配方式下 ST 流与 BE 流调度成功率

Fig. 6 Scheduled success rate of ST streams and BE streams in different queue length allocation methods

2）ST 流平均时延

在流量强度 $\rho_L=1.0$ 的情况下，比较使用可变长度队列与固定长度队列调度在 ST 流数量相同时，流的平均时延，结果如图 7 所示。

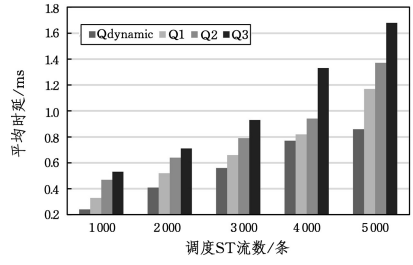


图 7 ST 流平均时延

Fig. 7 Average delay of ST streams

图 7 表明，调度相同数量的流，采用可变长队列可调度流的平均时延小于采用固定长度队列时的平均时延。在可变长队列的情况下，利用本文的队列调整算法可以使队列长度在

满足流量传输的情况下尽可能地小。因此，在调度相同数量流的情况下，采用固定长度队列所需要的平均队列长度大于采用可变长队列平均所需长度，导致队内排队时延增加，ST流的平均时延增加。

3)资源利用率

利用式(13)计算资源利用率，在流量强度 $\rho_L=1.0$ 的情况下，比较使用可变长队列与固定长度队列的网络缓存资源利用率，结果如图8所示。

图8表明，随着仿真时间的增加，可变长队列与固定队列的资源利用率均增加。在固定队列长度的情况下，当大量数据流导向同一个整形队列时，由于无法动用其他空闲的缓存资源，因此无法缓存更多的包；而利用可变长队列，可以根据预测流量动态调整队列长度，提升缓存资源利用率。

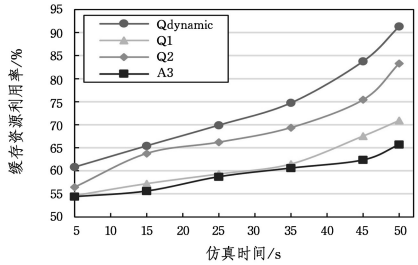


图8 缓存资源利用率

Fig. 8 Utilization rate of cache resource

结束语 本文针对TSN中ATS使用固定长度整形队列实现流量整形与调度存在的资源利用率不高、可调度流的平均时延高等问题进行研究，设计了一种基于改进磷虾群算法与流量预测的可变长队列调整算法。首先使用改进磷虾群算法COEAKH计算出针对应用场景的队列可调整上限；然后使用CNN-LSTM结构根据历史信息预测下一时间段到达某整形队列的ST流数量，通过预测值计算队列长度调整的步幅大小；最后根据步幅值与队列可调整上限，动态改变整形队列容量。仿真结果表明，与固定长度整形队列相比，所提算法能有效提升可调度流数量，减少ST流平均时延，并提高缓存资源利用率。本文通过动态调整整形队列的长度，在避免数据包过度缓存的前提下使数据包尽可能入队，以达到网络性能优化的目的。然而，丢包率是反映网络状态的重要指标，后续工作可以考虑从队列的丢弃策略方面入手，设计既能反映网络状态又能提升网络性能的队列长度管理算法。

参 考 文 献

[1] YANG S J, ZHUANG L, SONG Y, et al. Intelligent scheduling mechanism of the time-sensitive network model in polymorphic network[J]. Chinese Journal of Journal on Communications, 2022, 41(2): 85-93.

[2] MA D, FEI X, MU X W. A VNF-aware Virtualization Layer Constructing Algorithm Based on Adjustable Hop Count[J]. Journal of Zhengzhou University (Engineering Science), 2021, 42(5): 50-55.

[3] NASRALLAH A, THYAGATURU A S, ALHARBI Z, et al. Ultra-low latency (ULL) networks: the IEEE TSN and IETF DetNet standards and related 5G ULL research[J]. IEEE Communications Surveys & Tutorials, 2019, 21(1): 88-145.

[4] WANG J X, YANG S J, ZHUANG L, et al. Multi-objective Online Hybrid Traffic Scheduling Algorithm in Time-sensitive Networks[J]. Computer Science, 2023, 50(7): 286-292.

[5] NASRALLAH A, THYAGATURU A S, ALHARBI Z, et al. Performance Comparison of IEEE 802.1 TSN Time Aware Shaper (TAS) and Asynchronous Traffic Shaper (ATS) [J]. IEEE Access, 2019(7): 44165-44181.

[6] PRADOS-GARZON J, TALEB T, BAGAA M. Optimization of Flow Allocation in Asynchronous Deterministic 5G Transport Networks by Leveraging Data Analytics [J]. IEEE Transactions on Mobile Computing, 2023, 22(3): 1672-1687.

[7] LIN Y, JIN X, ZHANG T, et al. Queue assignment for Fixed-priority real-time flows in time-sensitive networks: Hardness and Algorithm [J]. Journal of Systems Architecture, 2021 (116): 102141.

[8] SPECHT J, SAMII S. Urgency-Based Scheduler for Time-Sensitive Switched Ethernet Networks [C]// 2016 28th Euromicro Conference on Real-Time Systems (ECRTS). 2016: 75-85.

[9] YIN S W, WANG S, HUANG T. Analysis and Optimization of Queue Based on Network Calculus in Time-Sensitive Networking[J]. ZTE Technology Journal, 2022, 28(1): 21-28.

[10] MA X, LI S, GUAN Z, et al. Time-Sensitive Networking Mechanism Aided by Multilevel Cyclic Queues in LEO Satellite Networks [J]. Electronics, 2023, 12(6): 1357-1369.

[11] YAN J, QUAN W, JIANG X, et al. Injection Time Planning: Making CQF Practical in Time-Sensitive Networking [C]// IEEE INFOCOM 2020 - IEEE Conference on Computer Communications. 2020: 616-625.

[12] WANG X, SHANG Z, XIA C, et al. TSN Switch Queue Length Prediction Based on an Improved LSTM Network [J/OL]. https://doi.org/10.1155/2021/5130888.

[13] MARINO A G, FONS F, GHARBA A, et al. Elastic Queueing Engine for Time Sensitive Networking [C]// 2021 IEEE 93rd Vehicular Technology Conference (VTC2021-Spring). 2021: 1-7.

[14] ZHOU Z, BERGER M S, RUEPP S R, et al. Insight into the IEEE 802.1 Qcr Asynchronous Traffic Shaping in Time Sensitive Network [J]. Advances in Science, Technology and Engineering Systems Journal, 2019, 4(1): 292-301.

[15] SPECHT J, SAMII S. Synthesis of Queue and Priority Assignment for Asynchronous Traffic Shaping in Switched Ethernet [C]// 2017 IEEE Real-Time Systems Symposium (RTSS). 2017: 178-187.

[16] PRADOS-GARZON J, TALEB T, BAGAA M. LEARNET: Reinforcement Learning Based Flow Scheduling for Asynchronous Deterministic Networks [C]// 2020 IEEE International Conference on Communications ICC. 2020: 1-6.

[17] IEEE Draft Standard for Local and metropolitan area networks-Bridges and Bridged Networks Amendment: Asynchronous Traffic Shaping[S]. Institute of Electrical and Electronics Engineers (IEEE), 2020.

[18] PRADOS-GARZON J,TALEB T. Asynchronous Time-Sensitive Networking for 5G Backhauling [J]. IEEE Network,2021,35(2):144-151.

[19] MATE M,SIMON C,MALIOSZ M. Asynchronous Time-Aware Shaper for Time-Sensitive Networking[C]// 2021 17th International Conference on Network and Service Management (CNSM). 2021.

[20] YE J C,LEUNG K C,LOW S H. Combating Bufferbloat in Multi-Bottleneck Networks:Theory and Algorithms [J]. IEEE/ACM Transactions on Networking,2021,29(4):1477-1493.

[21] HE L,HUANG S. An efficient krill herd algorithm for color image multilevel thresholding segmentation Problem [J]. Applied Soft Computing,2020(89):106063.

[22] VARGA A,HORNIG R. An overview of the OMNeT++ simulation environment [C]// Proceedings of the 1st International Conference on Simulation Tools and Techniques for Communications,Networks and Systems & Workshops. 2008.

[23] INET Framework[EB/OL]. https://inet.omnetpp.org/.

[24] GANDOMI A H,ALAVI A H. Krill herd:A new bio-inspired optimization Algorithm [J]. Communications in Nonlinear Science and Numerical Simulation,2012,17(12):4831-4845.

[25] WANG G,GUO L,GANDOMI A H,et al. Lévy-Flight Krill Herd Algorithm [J]. Mathematical Problems in Engineering,2013(2013):1-14.

[26] GUO L,WANG G G,GANDOMI A H,et al. A new improved krill herd algorithm for global numerical Optimization [J]. Neurocomputing,2014(138):392-402.

CAI Changjuan, born in 1999, postgraduate. Her main research interests include time-sensitive networks and next-generation Internet.

ZHUANG Lei, born in 1963, Ph.D supervisor. Her main research interests include time sensitive networking, future network architecture, and network virtualization.

(责任编辑:何杨)