



时间敏感网络中基于 EDWF-MTTF 的启发式调度算法

李传煌^{1,2}, 廖君虎¹, 宣家栋¹, 徐琪², 张慧峰², 朱俊², 邹涛², 张汝云²

(1. 浙江工商大学信息与电子工程学院 (萨塞克斯人工智能学院), 浙江 杭州 310018;

2. 之江实验室, 浙江 杭州 311121)

摘要: 随着工业控制和信息网络的快速融合发展, 基于以太网的时间敏感网络受到了广泛关注。时间敏感网络采用时间触发通信调度来保证数据传输的确定性。然而, 现有调度算法难以快速计算大规模周期性流量的调度表。因此, 设计了基于带权重的最早截止时间-最大传输时间优先的启发式调度算法, 通过灵活适配流量排序权重以及回溯增强等方法, 可以在较短时间内解决工业自动化场景中大规模流量的调度问题。实验结果表明, 相比经典整数线性规划方法, 带权重的最早截止时间-最大传输时间优先的启发式调度算法能更快地计算出大规模周期性流量调度表的较优解。

关键词: 时间敏感网络; 周期性流量; 流量调度; 启发式算法; 回溯法

中图分类号: TP393

文献标志码: A

doi: 10.11959/j.issn.1000-0801.2024009

EDWF-MTTF-based heuristic scheduling algorithm in time-sensitive networks

LI Chuanhuang^{1,2}, LIAO Junhu¹, XUAN Jiadong¹, XU Qi², ZHANG Huifeng²,

ZHU Jun², ZOU Tao², ZHANG Ruyun²

1. School of Information and Electronic Engineering (Sussex Artificial Intelligence Institute),
Zhejiang Gongshang University, Hangzhou 310018, China

2. Zhejiang Lab, Hangzhou 311121, China

Abstract: With the rapid integration and development of industrial control and information networks, time-sensitive networks based on ethernet have received widespread attention. In these networks, time-triggered communication scheduling is adopted to ensure the determinacy of data transmission. However, existing scheduling algorithms have struggled to rapidly calculate the scheduling tables for large-scale periodic traffic. Therefore, a heuristic scheduling

收稿日期: 2023-09-26; 修回日期: 2024-01-11

通信作者: 徐琪, xuqi@zhejianglab.com

基金项目: 国家自然科学基金资助项目 (No.U22A2005, No.61871468, No.62111540270); 之江实验室科研攻关项目 (No.2021LE0AC02); “慧眼行动” 资助项目 (No.D4DAC0D6); 浙江省自然科学基金资助项目 (No.LZ23F010003, No.LQ23F010009); 浙江省新型网络标准与应用技术重点实验室资助项目 (No.2013E10012)

Foundation Items: The National Natural Science Foundation of China (No.U22A2005, No.61871468, No.62111540270), The Key Research Project of Zhejiang Lab (No.2021LE0AC02), “Operation Wise Eyes” Project (No.D4DAC0D6), The Zhejiang Provincial Natural Science Foundation (No.LZ23F010003, No.LQ23F010009), Zhejiang Provincial Key Laboratory of New Network Standards and Technologies (NNST) (No.2013E10012)

algorithm based on the earliest deadline with weight first-maximum transfer time first was designed. This algorithm, by flexibly adapting traffic sorting weights and enhancing backtracking methods, could resolve the scheduling issues of large-scale traffic in industrial automation scenarios in a relatively short time. Experimental results show that compared to the classic integer linear programming method, the earliest deadline with weight first-maximum transfer time first heuristic scheduling algorithm is able to calculate a more optimal solution for the scheduling table of large-scale periodic traffic more swiftly.

Key words: time-sensitive network, periodic traffic, traffic scheduling, heuristic algorithm, backtracking algorithm

0 引言

随着工业控制和信息网络融合以及新兴业务的发展,现代网络物理系统对通信的确定性和实时性提出了更严格的要求,保证传输信息的有限时延对实现网络的确定性至关重要。以太网虽然作为各个领域的标准通信机制,但由于缺乏实时性,不适用于具有严格时延要求的应用。因此,IEEE 802.1 工作组中的时间敏感网络(time-sensitive networking, TSN)任务组开发了一套协议标准:TSN 定义了以太网数据传输的时间敏感机制。该标准在以太网的基础上加入了时间同步、流量整形、流量调度和网络配置等新机制,使时间敏感任务的服务质量得到有效保证。TSN 将以太网高兼容性及高质量的数据传输相结合,受到广泛关注^[1],是工业互联网未来的发展趋势^[2]。

时间敏感网络中数据的确定性传输主要通过标准中定义的调度机制来保证,调度机制的有效运行依赖于高效的调度算法。工业自动化场景对数据传输的确定性有较高的要求,因此,本文选取了适合该场景的时间感知整形机制进行深入研究。

时间感知整形机制在 IEEE 802.1Qbv 标准中定义,交换机的出端口基于静态生成的门控列表(gate control list, GCL)定时开关传输特定流量的队列,实现实时流量和其他流量的暂时隔离。虽然 IEEE 802.1Qbv 标准定义了详细的调度机制,但如何计算 GCL 并不在标准的范围内。通过流量调度表的计算可以确定交换机出端口的

GCL,然而计算流量调度表是一个 NP 难(NP-hard)问题,很难提出一种通用的适用于不同网络拓扑且可扩展到大规模流量的算法。一方面,常见的解决方法是基于可满足性模理论(satisfiability modulo theories, SMT)对 TSN 流量调度问题进行数学建模,并使用最优化方法数学求解器计算可行的调度方案,但需要大量的计算时间,仅适用于流量规模较小的应用场景;另一方面,为提高计算效率,可采用启发式算法求解此类流量调度问题,但启发式算法存在扩展性问题,难以高效调度所有流量等。

为此,本文针对工业自动化场景中时间敏感型流量调度问题,设计了基于带权重的最早截止时间-最大传输时间优先(earliest deadline with weight first-maximum transfer time first, EDWF-MTTF)的启发式调度算法。该算法在第一阶段对流量调度产生较大影响的参数设计了基于流量优先级的 EDWF-MTTF 排序机制。第二阶段根据 EDWF-MTTF 排序机制设计了基于 EDWF-MTTF 排序的启发式方法,并在第三阶段提出了以第二阶段算法为基础的基于启发式的回溯增强方法以提高算法在解决大规模流量问题时的成功率。对新算法进行仿真实验,并与经典的整数线性规划(integer linear programming, ILP)算法进行比较,实验结果证明,新算法能够在较短时间内计算出大规模周期性流量调度表的较优解,解决了经典的 ILP 算法在大规模流量下扩展性问题,并可以针对不同的网络拓扑灵活改变流排序标准,实现高效、快速的大规模 TSN 流量调度。



1 相关工作

1.1 时间感知整形机制

IEEE 802.1Qbv 工作组规定了基于时间感知整形器的流量调度机制,采用类似时分复用的方式保证时间敏感流的端到端确定性时延。

时间感知整形器的流量调度机制如图1所示,交换机端口有数条队列,每条队列传输一种类别的流量,类别与流量优先级的对应关系需要预先定义,例如,需要规划的周期性时间敏感流优先级较高,在类别7的队列中进行传输。每个队列都配有一个门控开关,只有当门控开关处于打开状态时,才能传输队列中的数据包,而门控开关由 GCL 控制,它规定在特定时刻打开哪几个队列的门控及打开多长时间,例如,在 T2 时刻关闭了所有队列的门控,而在 T3 时刻仅仅打开传输类别7流量队列的门控,这样可以消除低优先级流量对时间敏感流的影响。

1.2 流量调度算法

IEEE 802.1Qbv 标准规定了 TSN 中的各个流量按照预先计算好的调度表配置 GCL 实现传输的机制,但是计算调度表的算法并没有明确指出。目前计算 TSN 调度表的方法分为两类,一类是利用 SMT 或 ILP 理论,通过第三方求解器对调度问题的约束条件转化而来的表达式进行精确求解,

另一类是采用启发式算法来求解此类流量调度问题以提高求解效率。

Craciunas 等^[3]解释了 IEEE 802.1Qbv 标准和相关的问题,并给出了一个 SMT 模型,该方案能在 5 h 内解决由 5 个交换机、7 个终端和最多 100 条流组成的问题。Pop 等^[4]根据上述的模型,利用 ILP 求解器 CPLEX 实现了调度表的计算。随后 Craciunas 等^[5]又运用非线性算法作为理论背景提出了另一个 SMT 模型。Dos santos^[6]提出了第一个在 SMT 中建模并利用 Z3 求解器来解决模型的开源调度程序。后续的研究大多数是针对特定的问题来优化 SMT 模型,如减少产生的时间表中的时隙条目数量^[7]、提高低优先级流量的服务质量^[8]。

在联合路由和调度方面,主要使用 ILP 算法^[9-10],然而,当前基于 ILP 的调度器^[11]仍需 10 min 解决由 4 个交换机、16 个终端节点和少于 250 条流组成的问题。

可以看出,SMT 和 ILP 算法被广泛运用于通信网络的调度计算中,但其仍有显著的缺点,如得到最优的调度方案,需要花费大量的计算时间等。为提高效率,通常采用启发式算法来求解类似的调度问题。目前解决 TSN 调度问题的启发式方法的研究并不多,Durr 等^[12]将 TSN 调度视为一个无等待数据包调度问题,提出了基于禁忌搜索的启发式算法,并使用贪婪算法为每个流分

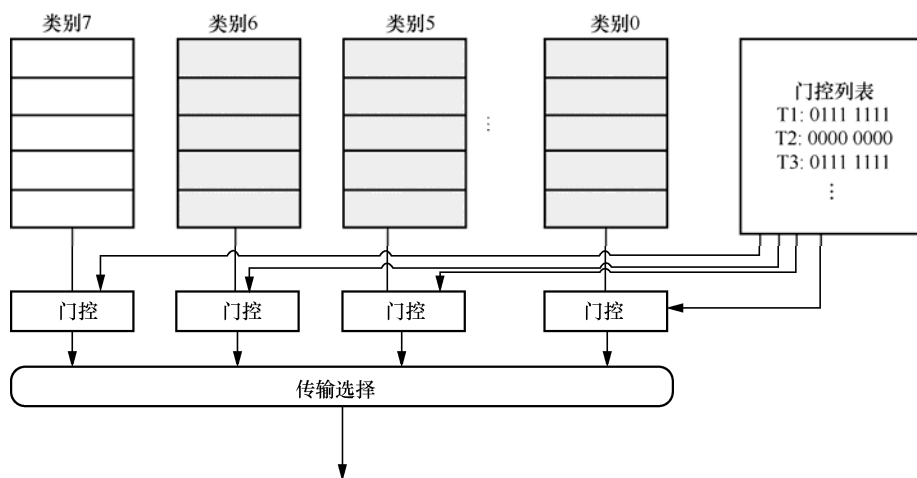


图1 时间感知整形器的流量调度机制

配相应的时间表。针对基于 TSN 的云计算平台新添加的流调度问题, Raagaard 和 Pop 等^[13]提出了单程多项式时间启发式算法, 尽量在最早的时间点分配每个帧, 这样就不会违反任何约束, 然而这种方法不能保证所有流量均被成功调度, 因此, 需要运用精确的调度求解方案。Pahlevan 等^[14-15]提出了一种启发式列表调度器和一种遗传算法, 通过分析流的多个潜在的路由, 缩短调度的完成时间, 但是文中没有给出具体的编码方法。根据文献[14-15]的工作, Arestova 等^[16]详细解释了应用于 TSN 调度的遗传算法编码方案。曹志鹏等^[17]基于改进的遗传算法^[18], 对选择算子和交叉变异概率加以改进, 并结合基于 K 最短路径的负载均衡路由算法的结果进行调度计算。Wang 等^[19]将改进的蚁群优化算法来调度 TSN 中的时间触发流, 在收敛速度和最优性方面均优于传统的蚁群算法。Gavrilut 等^[20-21]还研究了 TSN 中多种类型流量的调度方案, 使用 K 最短路径算法^[22]进行路由选择, 从中选取利用率最低的路径, 通过贪婪随机自适应搜索程序 (greedy randomized adaptive search procedure, GRASP)^[23]进行调度。

由于 TSN 标准定义的特殊机制, 有门控和队列的概念, 一些基于以太网来提高实时性的专有解决方案调度方法不能直接用到大规模流量的 TSN 调度中来, 同时现有的一些精确求解方法 (如 ILP 算法) 计算时间较长, 而比较高效的启发式算法扩展性较低, 且不能成功调度所有流量。因此需要开发一种高效的、可拓展的、端到端的总时延较小且能成功对较大规模流量进行调度的算法。基于此, 本文先对整个调度问题进行建模, 并定义了时间敏感流量传输过程中的约束, 最终提出了一种基于 EDWF-MTTF 的启发式调度算法。该算法在开始阶段可根据不同的网络拓扑结构灵活适配流量排序权重, 提高了扩展性。在后续阶段针对在计算大规模流量调度表时成功率低的问题利用回溯增强方法进行优化, 提高了整个

算法的成功率。此外, 本文研究的时间敏感网络调度算法仅需要解决数据传输过程中排队时延的非确定性问题。所以本文解决的调度问题是建立在路由已知的情况下, 即通过广度优先搜索算法来找到流的最短路径并将最短路径设置为流的传输路径。相较于曹旭等^[24]提出时间敏感网络流量调度方案, 他们虽然能够很好地解决兼容性, 但是当面对流量较大的情况下却无法得出可行解。Berisa 等^[25]提出一种具有帧抢占特征的启发式算法和一种旨在最大化音/视频桥 (audio video bridging, AVB) 流可调度性的新型路由机制可以最小化端到端时延。但是依然没有相应的机制去应对流量增大时的调度情况。

2 问题描述

问题来源于工业 TSN 中的通信, 具体表述如下: 在网络链路上调度满足周期性传输的流量, 其中假设每个流的路由确定, 并且每条链路的传输速率已知。为了模拟现实场景中的传输和交换时延, 引入节点和链路上的传播时延。通过计算满足多个约束条件下每个流注入其传输链路中每个节点的精确时间, 实现这些流的确定性传输。

2.1 网络与流量模型

以图 2 所示的一个工业典型场景为例。网络中有 9 个节点、8 条链路、4 条流, 每条链路的传输速率确定, 同时每条链路和每个交换节点都有固定的时延。

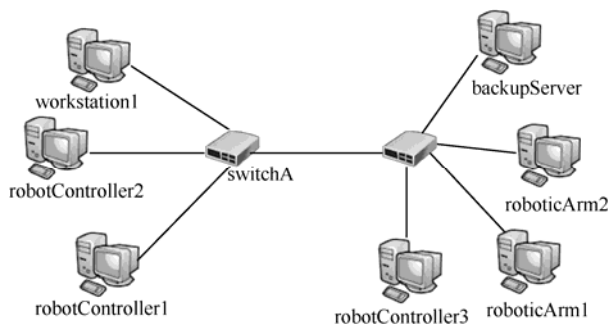


图 2 一个工业典型场景



本文用 robotController 模拟工业中发出控制指令的机器人控制器, roboticArm 模拟机器人的机械臂, 而 workstation 和 backupServer 模拟网络中低优先级数据的传输。robotController1 和 robotController2 发送的控制数据都要通过 (switchA, switchB) 链路, 在不做任何处理的情况下, 某些时刻, 会在 switchA 的出端口产生排队时延, 从而导致控制信息的端到端时延变化很大, 因此需要根据确定的链路传输速率和节点时延精确规划这两条流注入其传输链路中每个节点的时间, 并通过配置交换机队列的 GCL 在指定时刻打开传输该流量的队列, 同时关闭传输其他低优先级流量的队列, 这样控制信息传输的端到端时延在理论上是一个确定值。

根据上述实例, 本文将网络建模为一个简单互连的有向图 $G=(V, \mathcal{E})$, 其中 $v_i \in V$ 代表网络节点, 包括支持 TSN 特性的终端和交换机, $\mathcal{E}$ 表示节点之间双向链路的集合。 $a, b \in V$, a, b 是拓扑中任意两个相邻节点。将交换时延定义为 D_{v_a} 。 $e_{a,b}=(v_a, v_b) \in \mathcal{E}$ 表示节点之间的链路, $C_{e_{a,b}}$ 是该链路的传输速率, $D_{e_{a,b}}$ 代表链路时延。为了实现时间敏感流的调度, 整个网络中的设备通过 802.1AS 协议定义的机制实现全局时间同步。

在整个网络中从源节点到目标节点周期性传输的数据称为流, F 代表所有流的集合, 对于任意流 $i \in F$, 它的主要参数包括流的源节点 org_i 、目标节点 trg_i 、传输周期 T_i 、发送流的大小 p_i (以字节为单位), 其最大长度为以太网最大传输单元 (maximum transmission unit, MTU), 流从源节点最早的发送时间为 r_i , 流在目标节点最迟的接收时间为 $\tilde{d}_i$ 。每个流可能具有不同的周期, 把超周期定义为所有流周期的最小公倍数, 即 $\text{HP} = \text{lcm}\{T_i | i \in F\}$ 。

同时本文将通过链路 $e_{a,b}=(v_a, v_b) \in \mathcal{E}$ 的所有流量集合定义为 $F_{e_{a,b}}$, 链路上的某个流 i 在该链路

上的传输时间表示为 $t_i^{e_{a,b}} = 8 \frac{p_i}{C_{e_{a,b}}}$, (每个字节 8 bit)

并将流 $i \in F$ 在链路 $(v_a, v_b) \in \mathcal{E}$ 上传输的第 k 个帧定义为 $f_{i,k}^{e_{a,b}}$ (无下标 k 则默认是流 i 在此链路上传输的第一个帧), 其传输开始时间为 $s_{i,k}^{e_{a,b}}$, 流的传输路径可以表示为 R_i , $R_i = \{e_{\text{first}, \text{first}+1}, \dots, e_{\text{last}-1, \text{last}}\}$, 其中, $(\text{first}, \text{first}+1)$ 和 $(\text{last}-1, \text{last})$ 分别为流 i 传输路径上的第一条链路和最后一条链路。模型中的参数定义见表 1。

表 1 模型中的参数定义

参数	含义
V	节点集合
$\mathcal{E}$	链路集合
D_{v_a}	交换节点时延
$D_{e_{a,b}}$	链路时延
$C_{e_{a,b}}$	链路传输速率
F	流量集合
org_i	流 i 的发送节点
trg_i	流 i 的目标节点
T_i	流 i 的传输周期
p_i	流 i 的大小
r_i	流 i 在源节点最早的发送时间
$\tilde{d}_i$	流 i 在目标节点最迟的接收时间
HP	超周期, 所有流周期的最小公倍数
$t_i^{e_{a,b}}$	流 i 在链路 (v_a, v_b) 上的传输时间
$F_{e_{a,b}}$	通过链路 (v_a, v_b) 的所有流量集合
$f_{i,k}^{e_{a,b}}$	流 i 在链路 (v_a, v_b) 上传输的第 k 个帧
$s_{i,k}^{e_{a,b}}$	流 i 在链路 (v_a, v_b) 上传输的第 k 个帧的传输开始时间
R_i	流 i 的传输路径集合

2.2 流量传输约束

在一些典型 TSN 场景中, 为了保证时间敏感流的端到端传输时延, 需要对其在每个链路上的传输开始时间施加一定的约束^[5,26]。

约束 1: TSN 中的流量是严格周期性发送的,

所以在—个传输周期内，每条链路上属于同—个流的帧周期性出现，所以同—条流的不同帧之间没有抖动。

$$\forall i \in F, \forall k \in \left\{1, \dots, \frac{HP}{T_i} - 1\right\}, \quad (1)$$

$$\forall e_{a,b} \in \mathcal{E} : s_{i,k}^{e_{a,b}} = s_{i,k-1}^{e_{a,b}} + T_i$$

即每条链路上同—个流的第 $k-1$ 和第 k 帧的传输开始时间之间的差距是一个定值 T_i 。

约束 2: 为了保证到达交换节点的 TSN 流量无等待传输，任何链路都不能在同一时刻被不同流占用。—个帧的传输必须在另—个帧的传输开始之前完成。

$$\forall e_{a,b} \in \mathcal{E}, \forall s_{i,k}^{e_{a,b}}, s_{j,h}^{e_{a,b}}, (i,k) \neq (j,h) : s_{i,k}^{e_{a,b}} + t_i^{e_{a,b}} \leq s_{j,h}^{e_{a,b}} \vee s_{j,h}^{e_{a,b}} + t_j^{e_{a,b}} \leq s_{i,k}^{e_{a,b}} \quad (2)$$

约束 3: 在帧完全到达交换节点并进行处理之前，无法从该节点发送该帧。相邻两条链路上同—个流的帧必须在连接它们的交换节点处理完之后才能发送到下一条链路中。

$$\forall i \in F, \forall e_{x,a}, e_{a,b} \in \mathcal{E}, i \in F_{e_{x,a}} \cap F_{e_{a,b}} : s_i^{e_{a,b}} \geq s_i^{e_{x,a}} + t_i^{e_{x,a}} + D_{e_{x,a}} + D_{v_a} \quad (3)$$

约束 4: TSN 流量必须及时到达目标节点，因此每个流的帧的传输间隔必须在 $[r_i, \tilde{d}_i]$ 范围内，所以只需检查每个流的第—个帧的传输开始时间和最后—个帧是否能在截止日期前传输完成即可，约束 1 只需检查每个流第—个周期内的帧。

$$\forall i \in F : s_i^{e_{\text{first}, \text{first}+1}} \geq r_i$$

$$\forall i \in F : s_i^{e_{\text{last}-1, \text{last}}} + t_i^{e_{\text{last}-1, \text{last}}} + D_{e_{\text{last}-1, \text{last}}} \geq \tilde{d}_i \quad (4)$$

优化目标：所有流的端到端时延之和。

$$\min \sum_{i \in F} s_i^{e_{\text{last}-1, \text{last}}} + t_i^{e_{\text{last}-1, \text{last}}} + D_{e_{\text{last}-1, \text{last}}} - s_i^{e_{\text{first}, \text{first}+1}} \quad (5)$$

3 算法设计

基于 EDWF-MTTF 的启发式调度算法先根据

不同的流量信息对流量进行排序并创建—个流量队列，基于队列索引从低到高依次为每条流量的所有帧分配传输开始时间。然后基于设计好的流量排序机制设计—个启发式算法对满足约束的流量进行调度。最后使用回溯增强方法优化大规模流量下的调度问题。该算法包含 3 个阶段，分别是 EDWF-MTTF 流量排序机制、基于 EDWF-MTTF 排序的启发式方法、基于启发式的回溯增强方法。

3.1 阶段 1: EDWF-MTTF 流量排序

网络中每条流有最早传输开始时间、链路传输时间、截止接收时间等不同参数，如果某条流在其传输链路上所有可分配的传输开始时间都无法满足传输约束，算法就无法得到可行解，所以良好的流量优先级排序机制对创建正确的调度表会产生积极的影响。

通常流的截止接收时间会对调度过程产生比较大的影响，因此把截止接收时间作为第—个流量优先级排序的方法，为了能适应改变截止接收时间所占的权重，其可以通过 $[\tilde{d}_i / w]$ 决定 (w 代表截止接收时间所占的权重，针对不同的网络拓扑可以灵活设置权重，改变流排序方法，提高了扩展性)，本文将此命名为带权重的最早截止时间优先 (earliest deadline with weight first, EDWF)。

为了防止两条流的第—个优先级值相同，提出了第—个排序方法，即流的总传输时间，可通过流在每条链路上的传输时间相加并乘以一个超周期内传输次数式 (6) 得到。通常需要保证传输时间较长的流被优先分配，所以最终的排序方法需要用—个定值减去总传输时间，这个定值设置为 HP，将第—个排序方法命名为最大传输时间优先 (maximum transfer time first, MTTF)。

$$HP / T_i \sum_{e_{a,b} \in R_i} ((t_i^{e_{a,b}} + D_{e_{a,b}} + D_{v_b}) - D_{e_{\text{last}-1, \text{last}}}) \quad (6)$$

本文根据以上提出的两个排序方法进行整合，提出—个流量优先级排序机制并命名为 EDWF-MTTF。



3.2 阶段 2: 基于 EDWF-MTTF 排序的启发式方法

基于上述流量优先级排序机制, 设计了基于 EDWF-MTTF 排序的启发式方法, 该方法的目的是为所有排好序的帧分配一个传输开始时间, 满足所有约束要求。基于 EDWF-MTTF 排序的启发式方法流程如图 3 所示。

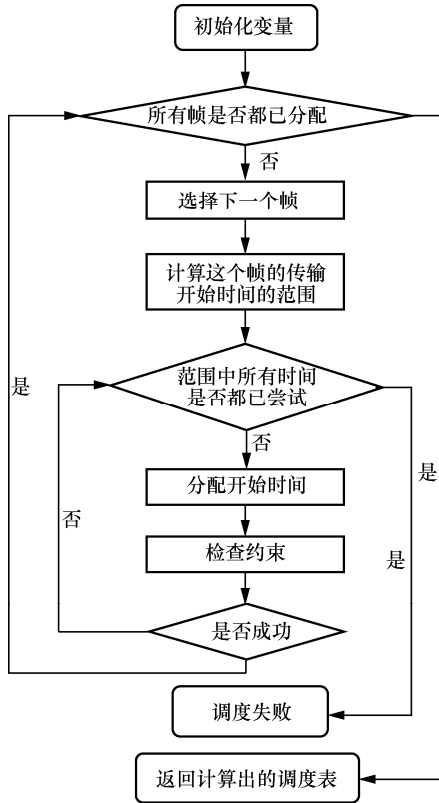


图 3 基于 EDWF-MTTF 排序的启发式方法流程

为了降低调度过程中约束检查复杂度, 需要为每条流的每个帧计算满足一些约束条件的最早传输开始时间和最迟传输开始时间, 这样在为其分配可用时间段时可以跳过一些约束。

$\forall i \in F$, $est_i^{e,a,b}$ 代表链路 (v_a, v_b) 上属于流 i 帧的最早传输开始时间, 通过流从源节点的最早发送时间 (若该帧为第一条链路上传输的帧) 或者前一个帧的最早传输开始时间确定, $lst_i^{e,a,b}$ 代表链路 (v_a, v_b) 上流 i 帧的最迟传输开始时间, 通过流在目标节点的最迟接收时间 (若该帧为最后一条

链路上传输的帧) 或者后一个帧的最迟传输开始时间确定。这样, 每个帧的传输开始时间范围可以由 $[est_i^{e,a,b}, lst_i^{e,a,b}]$ 确定, 其表达式如下。

$$\begin{cases} est_i^{e_{first, first+1}} \leftarrow r_i \\ est_i^{e_{a,b}} \leftarrow est_i^{e_{x,a}} + t_i^{e_{x,a}} + D_{e_{x,a}} + D_{V_a} \end{cases} \quad (7)$$

$$\begin{cases} lst_i^{e_{last-1, last}} \leftarrow \tilde{d}_i - t_i^{e_{last-1, last}} - D_{e_{last-1, last}} \\ lst_i^{e_{a,b}} \leftarrow lst_i^{e_{b,y}} - t_i^{e_{a,b}} - D_{e_{a,b}} - D_{V_b} \end{cases} \quad (8)$$

因为后续重复出现帧的传输开始时间根据第一次定期出现帧的传输开始时间来规划, 所以约束式 (1) 一定满足, 而约束式 (4) 已经集成在计算每个帧的传输开始时间范围的式 (7)、式 (8) 中, 同时算法一次性依次规划一个流的所有帧, 约束式 (3) 也满足, 因此仅需检查约束式 (2)。

基于 EDWF-MTTF 排序的启发式方法如算法 1 所示, 约束检查算法如算法 2 所示。算法 1 步骤如下: 第一步接收一个已经按照优先级排好序的帧队列 $allFrames$ 作为输入。第二步初始化用于存储已分配帧的信息的列表和所有未分配帧的列表。第三步当帧队列中仍有帧未被处理时进入一个循环, 从 $unassigned\ frames$ 中取出优先级最高的帧放入 $assigned\ frames$ 表示已分配。调用 $getbounds$ 函数计算帧 n 的传输开始时间范围, 并将第一个值赋给 $startTime$ 初始化变量 $success$ 为 $false$ 用于表示分配是否成功。第四步当 $success$ 为 $false$ 且 $startTime$ 小于链路 (V_a, V_b) 上流 n 帧的最迟传输开始时间时进入一个循环, 调用 $assign$ 函数来分配传输开始时间和链路上的传输时间, 调用 $CHECK$ 检查分配的帧是否满足约束条件, 如果满足则将 $success$ 设置为 $true$ 。第五步如果分配成功, 则递增计数器, 表示已经规划了下一帧的传输, 回到步骤 3 再继续处理下一帧; 如果分配不成功, 则返回一个标志, 表示无法满足约束条件的调度方案。第六步算法结束, 返回存储已分配帧信息的列表 $assigned\ frames$ 。 $unassigned\ frames$ 代表根据优

优先级标准在列表中有序排列的未分配帧(同一条流的每个帧依次排列), assigned frames 代表正在分配传输开始时间的帧, $\text{startTime}(n)$ 表示当前为帧 n 分配的传输开始时间。

算法 1 基于 EDWF-MTTF 排序的启发式方法

输入: 根据优先级排好序的帧队列

输出: 每条链路的调度表

程序 CREATESCHEDULE(allFrames)

assigned frames $\leftarrow []$

unassigned frames $\leftarrow$ allFrames

$n \leftarrow 0$

while $n < |\text{allFrames}|$ do

 assigned frames[n] $\leftarrow$ get(unassigned frames)

 startTime $\leftarrow$ getbounds(n)

 success $\leftarrow$ false

 while $\neg \text{success} \wedge \text{startTime} < \text{lst}_n^{e,a,b}$ do

 assign(startTime, assigned frames[n], t_n)

 success, startTime $\leftarrow$ CHECK (assigned

frames, n , startTime)

 if success then

$n \leftarrow n + 1$

 else

 return infeasible

return assigned frames

算法 2 约束检查算法

输入: 帧 n 的传输开始时间

输出: 约束检查结果和新的传输开始时间

程序 CHECK(assigned frames, n , startTime)

success $\leftarrow$ true

for 每个和帧 n 在同一链路上的帧 n' do

 if startTime $< \text{lst}_n^{e,a,b}$ then

$x_1 = \text{startTime}(n') - \text{startTime}(n)$

$x_2 = \text{startTime}(n) - \text{startTime}(n')$

 if $x_1 < t_n \vee x_2 < t_{n'}$ then

 startTime++

 success $\leftarrow$ false

if $\neg \text{success}$ then

 return false, startTime

return true, startTime++

约束检查算法采用的主要方法是依次将当前帧分配的传输开始时间与此链路上已分配帧的传输开始时间做差, 并与当前帧在此链路上的传输时间做比较, 若小于该帧的传输时间, 则约束检查未通过。

由于向队列中添加元素的复杂度是 $O(\log n)$, 从队列中检索元素的时间复杂度是 $O(1)$ 。定义拓扑中任意两个节点之间的最长路径长度为 d , 网络中的流总数为 N , 超周期为 HP, 任何流的最大周期为 T , 网络中最大的帧数量表达式为 Nd , 将此表达式替换为 NI。

优先级队列初始化的时间复杂度为 $\text{NI} \cdot \log \text{NI} + \text{NI} \approx O(\text{NI} \cdot \log \text{NI})$ 。其中, $\text{NI} \log \text{NI}$ 对应将所有流的帧插入优先级队列, NI 对应计算每个帧标准值的时间复杂度。

遍历所有帧的时间复杂度为: NI。为每个帧分配空闲的传输开始时间以及约束检查的时间复杂度为 $(T/2)(\text{HP}/2) \approx O(T \cdot \text{HP})$ 。

更新优先级队列的时间复杂度为 $2(d-1) + (d-1) \log \text{NI} \approx O(d \cdot \log \text{NI})$ 。即先从优先级队列中删除当前规划帧的所有前置或后继帧, 然后更新这些帧的标准值。

综上, 整个算法的时间复杂度可以简化为 $O(Nd(T \cdot \text{HP} + d \cdot \log(dN)))$ 。

3.3 阶段 3: 基于启发式的回溯增强方法

针对大规模流量的问题, 基于 EDWF-MTTF 排序的启发式方法尝试为某个帧分配的所有可用传输开始时间都未通过约束检查, 无法得到可行解。为此本文在基于 EDWF-MTTF 排序的启发式方法的基础上提出了一种回溯增强方法, 首先依次为队列中的帧分配传输开始时间, 如果所有可分配的时间都与先前已分配好的帧的传输时间发生冲突, 那么算法回溯到先前分配的帧, 重新为其分配传输开始时间。



基于启发式的回溯增强方法主要是为当前正在分配的帧 `assigned frames[n]` 设置一个冲突集 `conf[n]`，在检查约束时，若发现当前分配的时间已被先前分配的帧占用，那么将先前分配的帧加入当前正在分配帧的冲突集中，冲突集在约束检查的过程中不断更新。当可用的传输开始时间全部被尝试且都未满足约束时，就需要进行回溯，选择重新分配距离当前帧最近的帧，即冲突集中序号最大的帧，这样会跳过导致冗余搜索的步骤，但同时没有跳过任何可行的解，由于不一定是冲突集中序号最大的帧导致无法找到解决方案，需要把帧 n 的冲突集合并到序号最大帧的冲突集中，最后依次清空当前帧和回溯到的帧之间已分配帧的信息。

为了节省回溯时算法的运行时间，引入了一个新变量 `step`，在算法回溯到先前分配的帧时，尝试为其分配的下一个传输开始时间是原来的 `startTime` 上加 `step`，这样可能会跳过一些可行解，但是能够更快地搜索解空间，如何设定 `step` 的值会对算法造成一定的影响，本文将 `step` 设置为 $\lceil t_i^{e,a,b} / 100 \rceil$ ，与帧在链路上传输时间相关。

基于启发式的回溯增强方法伪代码见算法 3，其中 `newstartTime` 表示回溯到该帧时尝试为其分配的下一个传输开始时间。

算法 3 基于启发式的回溯增强方法

输入： 根据优先级排好序的帧队列

输出： 每条链路的调度表

程序 `CREATESCHEDULE(allFrames)`

`assigned frames` $\leftarrow$ []

`unassigned frames` $\leftarrow$ `allFrames`

$n \leftarrow 0$

while $n < |\text{allFrames}|$ do

 if $n = |\text{assigned frames}|$

`assigned frames[n]` $\leftarrow$ get (`unassigned frames`)

`startTime` $\leftarrow$ getbounds(n)

 else

`startTime` $\leftarrow$ `newstartTime`

`success` $\leftarrow$ false

 while $\neg \text{success} \wedge \text{startTime} < \text{lst}_n^{e,a,b}$ do

 assign(`startTime`, `assigned frames [n]`, t_n)

`success`, `startTime` $\leftarrow$ CHECK (`assigned frames, n, startTime`)

 if `success` then

`newstartTime` $\leftarrow$ `startTime`

$n \leftarrow n + 1$

 else

 if `conf[n]` = $\emptyset$ then return infeasible

 else

$\text{max} \leftarrow \max(\text{conf}[n])$

`conf[max]` $\leftarrow$ `conf[max]`

$\cup \text{conf}[n] / \{m\}$

 while $n > \text{max}$ do

`unassigned frames` $\leftarrow$ `unassigned frames` $\cup \{\text{assigned frames}[n]\}$

`assigned frames` $\leftarrow$ `assigned frames` $\setminus \{\text{assigned frames}[n]\}$

`newStartTime` $\leftarrow$ 0

`conf[n]` $\leftarrow$ $\emptyset$

$n \leftarrow n - 1$

 return `assigned frames`

约束检查算法中唯一不同的是在进行回溯时，如果未通过约束检查，将与当前分配帧的传输开始时间冲突的帧放入其冲突集中，同时为了缩短求解时间，将尝试的下一个 `startTime` 的值设置为 `startTim + step`。

4 实验分析

4.1 实验设置

为证明新算法的可行性与高效性，将基于 EDWF-MTTF 的启发式调度算法与传统的 ILP 算

法和随机 (Random) 排序的启发式方法做对比分析。ILP 算法是完整且最优的, 若没有时间限制, 它能保持解的最优性, 因此用该算法来衡量本文的算法。随机排序的启发式方法是用于评估其他启发式算法性能的第二种基线方法, 它为每条流赋一个随机值, 然后依据这个值对所有流进行排序。本文相关算法的程序使用 Java 8 进行编写, ILP 算法是使用 Gurobi 实现的, 它是一种数学规划求解器, 拥有良好的性能和易于理解的应用程序接口 (application program interface, API)。硬件环境为 Intel Core i7-8700 和 16 GB RAM。所采用的 IDE 为 IntelliJ IDEA 2021.1 版本。

为了使得算法解决的问题场景能够涵盖比较广的范围, 根据文献[27]的方法定义了不同类型的实验拓扑, 该文献为分布式系统中的时间触发调度设计了工业规模的拓扑。基于此, 本文在图 4~图 6 所示的实验拓扑中进行实验。

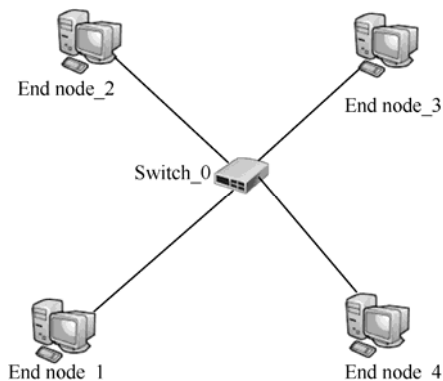


图 4 实验拓扑 1: 星形拓扑

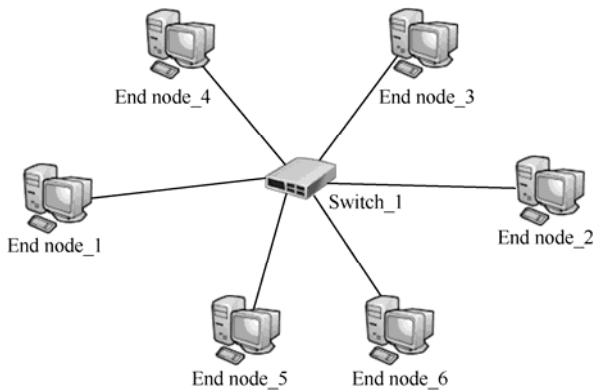


图 5 实验拓扑 2: 星形拓扑

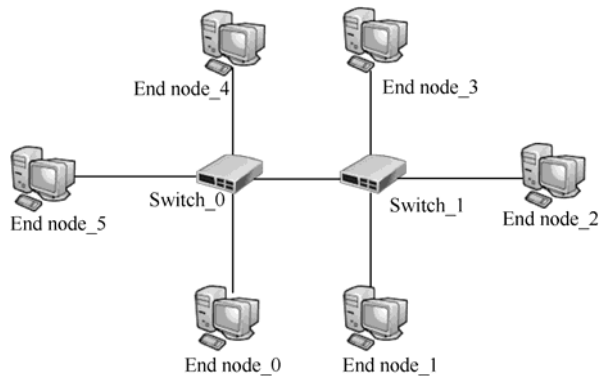


图 6 实验拓扑 3: 线型拓扑

拓扑中的终端节点和交换机之间的链路传输速率为 100 Mbit/s, 两个交换机之间的链路传输速率为 1 Gbit/s。所有链路传输时延为 $1\ \mu\text{s}$, 节点的时延为 $10\ \mu\text{s}$ 。星形拓扑中, 流量可以在任意两个终端节点之间发送, 考虑不同的网络拓扑结构存在着交换节点数、终端节点数等变量, 为了更好地衡量算法在不同拓扑中的性能表现, 针对同一拓扑结构不同终端节点数量的情况开展实验验证。线型拓扑中, 流量在任意一对固定的终端节点之间通信, 例如, node_5 向 node_2 请求通信或 node_1 向 node_5 请求通信等。所有流的大小从 125 byte 到 1 500 byte 之间随机选取, 周期从 $1\ 000\ \mu\text{s}$ 、 $2\ 500\ \mu\text{s}$ 、 $5\ 000\ \mu\text{s}$ 、 $10\ 000\ \mu\text{s}$ 中选择。流的最早发送时间设置一定的上限, 并且最迟接收时间和最早发送时间之间的差值必须覆盖流传输周期的 20%~50%。本文实验拓扑规模见表 2。以实验拓扑 2 为例, 网络拓扑中流的数量统计见表 3。

如表 2 所示, 在流数量为 100 的情况下, node_1 作为客户端向 node_2、node_3、node_4、node_5、node_6 等服务端分别发送 3、3、3、4、3 条业务流, 其他终端中流的流向同理。

为了确保启发式算法和 ILP 算法的运行环境相同, 将 Gurobi 允许使用的线程数设置为 1, 并将每个求解方法的求解时间上限设置为 60 s, 即 ILP 算法会在 60 s 时输出当前最优解, 而非全局最优解。



表 2 实验拓扑规模

拓扑编号	拓扑结构	交换节点数	终端节点数	流数量
1	星形	1	4	{80,160,240}
2	星形	1	6	{100,200,300}
3	线型	2	6	{100,200,300}

表 3 实验拓扑 2 中流数量统计

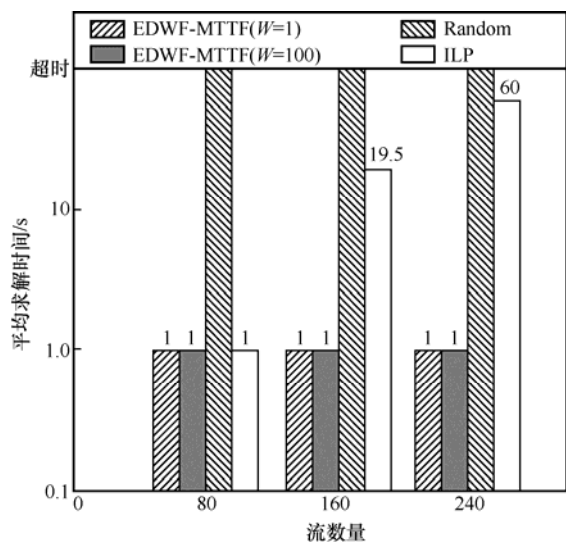
流数量	客户端	服务端					
		node_1	node_2	node_3	node_4	node_5	node_6
100	node_1	0	3	3	3	4	3
	node_2	5	0	3	3	3	3
	node_3	3	4	0	3	4	3
	node_4	4	3	3	0	3	3
	node_5	3	4	3	3	0	3
	node_6	3	4	4	3	4	0
200	node_1	0	6	7	6	6	7
	node_2	7	0	7	7	7	6
	node_3	7	7	0	6	6	6
	node_4	7	6	7	0	6	7
	node_5	7	7	7	7	0	7
	node_6	6	7	7	7	7	0
300	node_1	0	10	10	10	11	10
	node_2	10	0	11	10	10	10
	node_3	10	10	0	10	9	10
	node_4	11	11	10	0	10	10
	node_5	10	8	10	10	0	10
	node_6	10	9	10	10	10	0

4.2 基于 EDWF-MTTF 排序的启发式方法性能分析

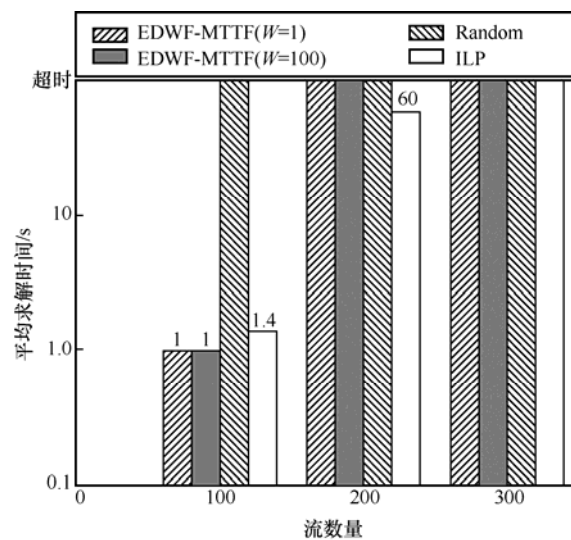
为了对提出的基于 EDWF-MTTF 排序的启发式方法的性能进行分析, 本实验将排序标准中的截止接收时间的权重分别设置为 1 和 100, 将此两种状态下的求解方法与传统的 ILP 算法和随机排序的启发式方法作为对比。

在每个拓扑下通过算法求解出的目标值(若未得到解则计算规划的流条数)和求解时间来衡量算法的性能, 其中横坐标代表拓扑中的流数量。各个拓扑下的实验结果如图 7~图 9 所示。

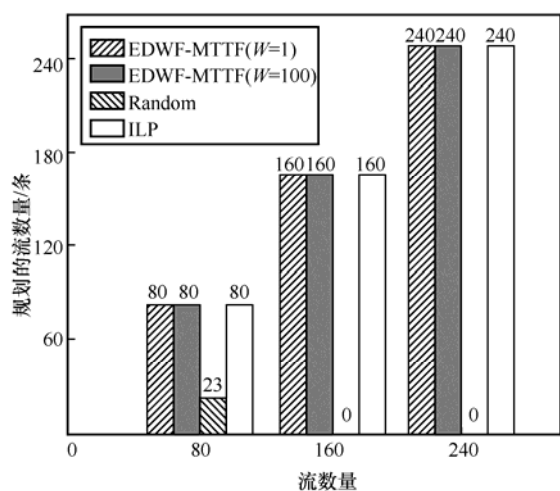
经过多轮实验结果表明, 在不同的拓扑且流数量小于 100 条的情况下, 基于 EDWF-MTTF 排序的启发式方法和 ILP 算法都能在 1 s 内求解调度表, 但是基于 EDWF-MTTF 排序的启发式方法解的最优性远不如 ILP 算法, 这种情况下采用 ILP 算法以获得调度表的最优解, 而对于随机排序的启发式方法, 则在求解过程中超出求解时间的上限(图 7~图 9 中的超时都代表该方法在计算过程中超时), 即在规定时间内无法求解, 说明本文提出的排序机制对求解过程产生了积极影响。



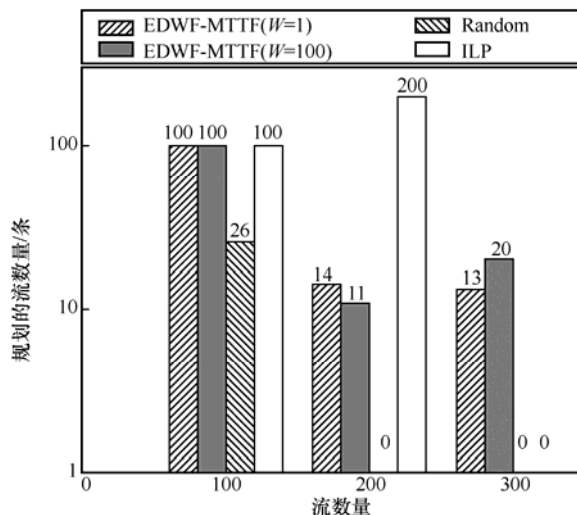
(a) 平均求解时间



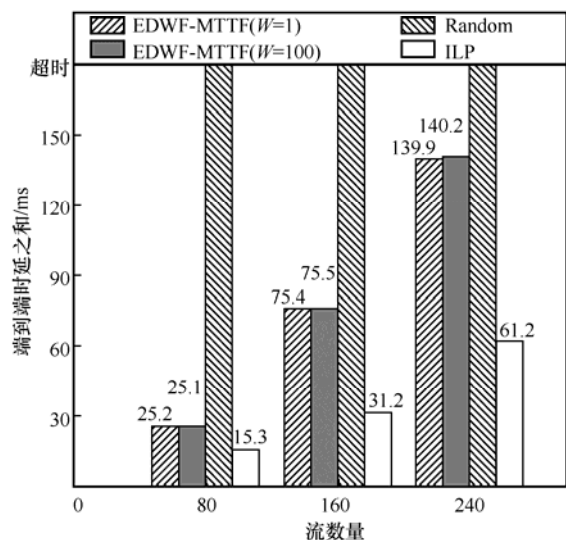
(a) 平均求解时间



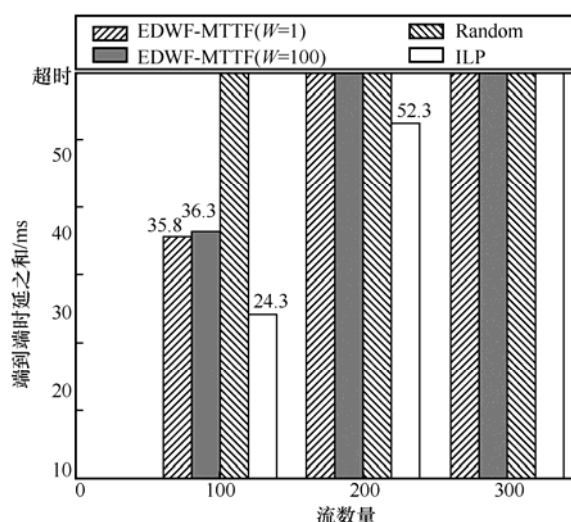
(b) 规划的流数量



(b) 规划的流数量



(c) 端到端时延之和



(c) 端到端时延之和

图 7 拓扑编号 1 实验结果

图 8 拓扑编号 2 实验结果

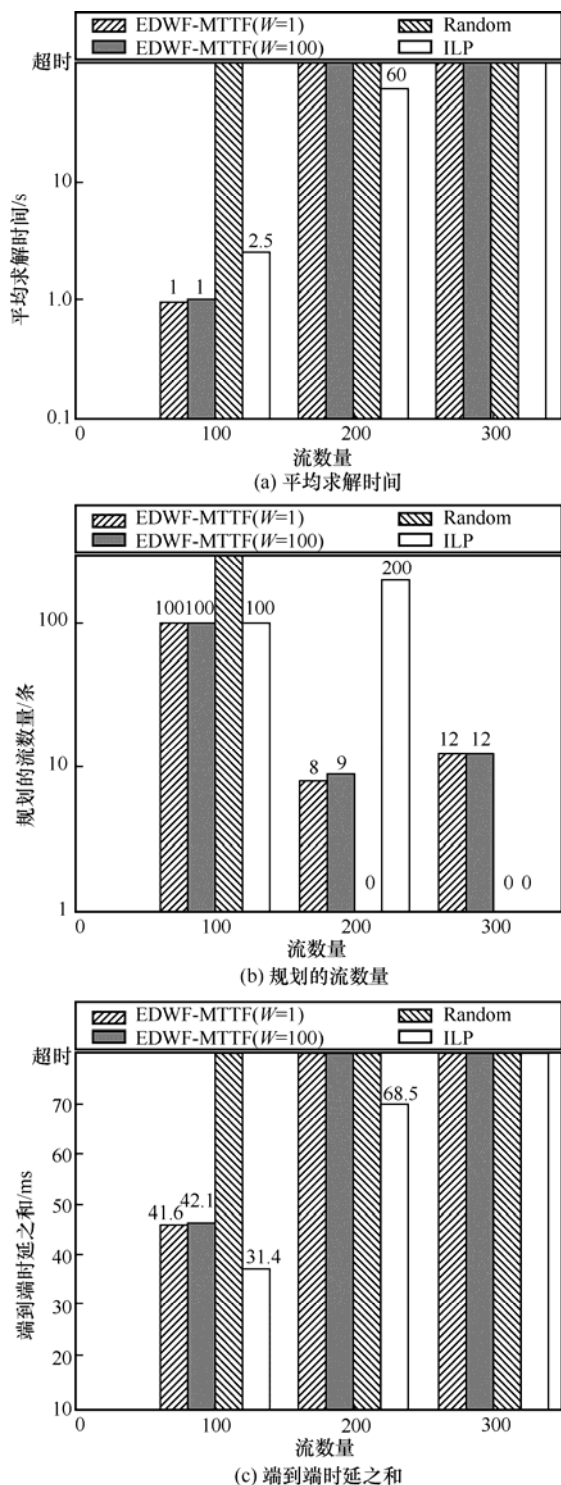


图9 拓扑编号3实验结果

流条数在 100~200 时, ILP 算法能在 60 s 内找到最优解, 但是求解时间较长, 而基于 EDWF-MTTF 排序的启发式方法能在较短时间内求解, 虽然得到的目标值与最优解之间存在一定

的差距, 但这个差距相较于求解时间之间的差距是微不足道的。

当流条数在 200~300 时, ILP 算法能在 60 s 内得到当前的最优解, 而非全局最优解, 而基于 EDWF-MTTF 排序的启发式方法可能无法求解。当流条数大于 300 时, ILP 算法在 60 s 内无法求解。在星形拓扑结构中, 流量相近的情况下, 不同的终端节点数量并未对本算法的性能产生明显的影响。因此, 基于 EDWF-MTTF 排序的启发式方法在终端节点增加, 网络规模扩大时依旧适用。

综上, 本文提出的基于 EDWF-MTTF 排序的启发式方法适用于流条数适中的情况, 在流条数较多的情况下, 为某条流分配的传输开始时间与之前已经分配的时间产冲突而导致算法无法继续运行。

4.3 基于启发式的回溯增强方法性能分析

第 4.2 节中实验结果证明在流条数大于 200 的情况下, 基于 EDWF-MTTF 排序的启发式方法可能无法求解, 因此, 在此方法的基础上提出了基于启发式的回溯增强方法。首先需要选择一种良好排序机制的启发式方法, 它可以尽可能规划更多的流, 并且求解的目标值较优, 这样能减少回溯方法运行的时间, 根据第 4.2 节中的实验结果, 发现基于 EDWF-MTTF 排序的启发式方法在权重设置为 1 的性能更好, 所以在此权重的基础上进行回溯。

本文将回溯方法中的 step 设置为 $\lceil t_i^{e,a,b} / 100 \rceil$, 作为对比, 将另一种回溯方法的 step 值设为 1。对于回溯方法, 仅仅在第 4.2 节中基于 EDWF-MTTF 排序的启发式方法无法求解的拓扑下实验, 实验结果如图 10、图 11 所示。

经过多次实验, 从实验结果中可以得出, 在基于 EDWF-MTTF 排序的启发式方法的基础上做回溯能在不同的网络拓扑结构下成功调度 200 条以上的流, 这是基于 EDWF-MTTF 排序的启发式

方法无法实现的,甚至能在不同拓扑下求解 300 条以上流,而这种情况 ILP 算法也无法在 60 s 内求解,(图 10、图 11 中的超时都代表在规定时间内无法求解)。同时,基于启发式的回溯增强方法的平均求解时间小于 10 s。相对而言,step 设置为 $\lceil t_i^{e,a,b} / 100 \rceil$ 的方法比 step 设置为 1 的方法平均求解时间更低,因为它跳过了某些可行解,同时这种方法损失了一定的目标值最优性,但是这个损失远远小于缩短的求解时间。综上,step 设置为 $\lceil t_i^{e,a,b} / 100 \rceil$ 的基于启发式的回溯增强方法能对不

同的网络拓扑,在平均 10 s 内成功调度 200~300 条流,较大程度优化了基于 EDWF-MTTF 排序的启发式方法。

5 结束语

本文针对工业场景中时间敏感型数据调度问题,设计了基于 EDWF-MTTF 的启发式调度算法。该算法包括了第一阶段的 EDWF-MTTF 流量排序机制,并基于此机制设计了第二阶段的基于 EDWF-MTTF 排序的启发式方法,在流量适中的

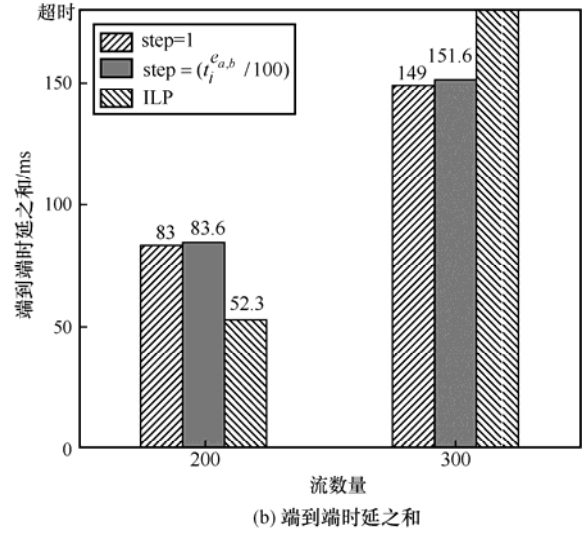
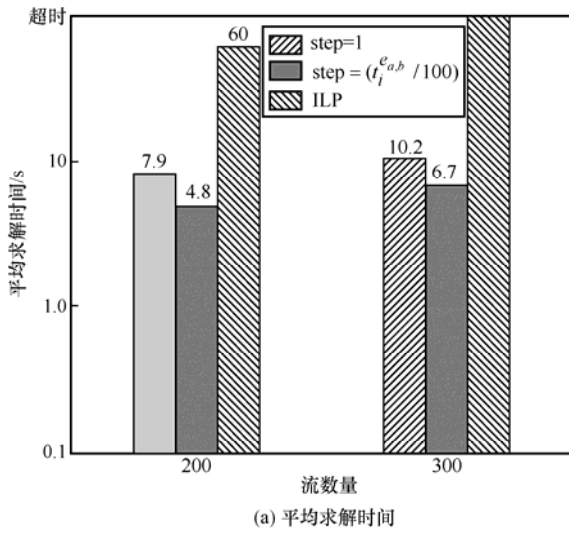


图 10 星形拓扑结构下基于启发式的回溯增强方法实验结果

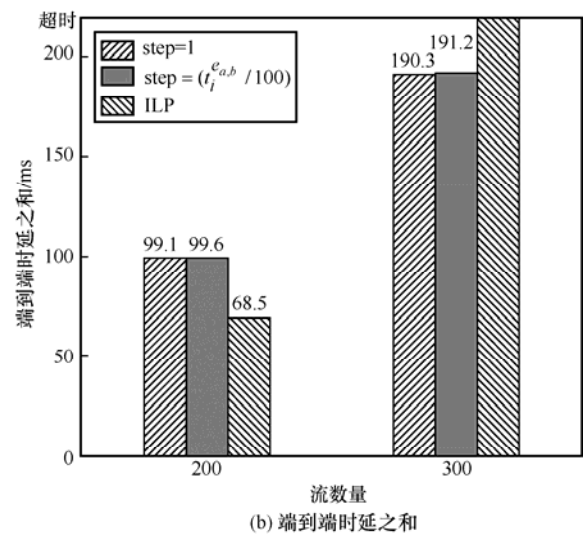
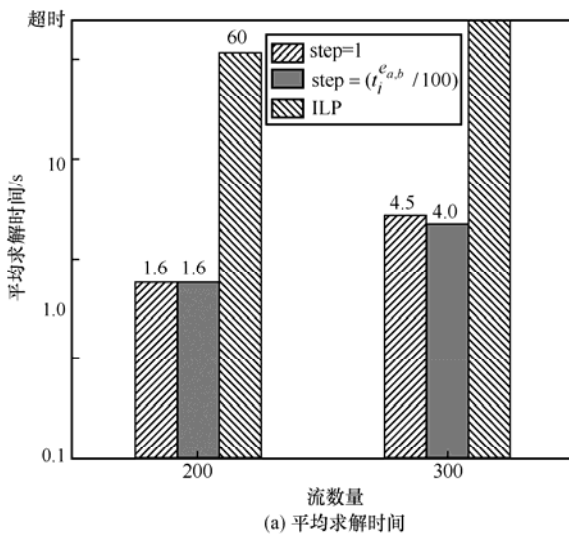


图 11 线型拓扑结构下基于启发式的回溯增强方法实验结果



情况下能够在较短时间内获得较优目标值的调度表,在第三阶段使用基于启发式的回溯增强方法对算法进行优化,解决了第二阶段基于 EDWF-MTTF 排序的启发式方法在流规模扩大情况下成功率下降的问题。实验结果表明,本文提出的基于 EDWF-MTTF 的启发式调度算法能在 10 s 内规划高达 300 条周期性 TSN 流量,突破了经典的 ILP 算法在同等规模下短时间内无法求出最优解的问题。在下一步的研究中,将继续研究更加丰富、更加复杂的应用场景,提高算法的灵活性和适用性。

参考文献:

- [1] LO BELLO L, STEINER W. A perspective on IEEE time-sensitive networking for industrial communication and automation systems[J]. *Proceedings of the IEEE*, 2019, 107(6): 1094-1120.
- [2] 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.
- [3] CRACIUNAS S S, OLIVER R S, CHMELÍK M, et al. Scheduling real-time communication in IEEE 802.1Qbv time sensitive networks[C]//*Proceedings of the 24th International Conference on Real-Time Networks and Systems*. New York: ACM Press, 2016: 183-192.
- [4] POP P, RAAGAARD M L, CRACIUNAS S S, et al. Design optimisation of cyber-physical distributed systems using IEEE time-sensitive networks[J]. *IET Cyber-Physical Systems: Theory & Applications*, 2016, 1(1): 86-94.
- [5] CRACIUNAS S S, OLIVER R S, STEINER W. Formal scheduling constraints for time-sensitive networks[J]. *arXiv preprint*, 2017, arXiv: 1712.02246.
- [6] DOS SANTOS A C T, SCHNEIDER B, NIGAM V. TSN-SCHED: automated schedule generation for time sensitive networking[C]//*Proceedings of the 2019 Formal Methods in Computer Aided Design (FMCAD)*. Piscataway: IEEE Press, 2019: 69-77.
- [7] LI Q, LI D, JIN X, et al. A simple and efficient time-sensitive networking traffic scheduling method for industrial scenarios[J]. *Electronics*, 2020, 9(12): 2131.
- [8] HOUTAN B, ASHJAEI M, DANESHTALAB M, et al. Synthesizing schedules to improve QoS of best-effort traffic in TSN networks[C]//*Proceedings of the 29th International Conference on Real-Time Networks and Systems*. New York: ACM Press, 2021: 68-77.
- [9] ATALLAH A A, HAMAD G B, MOHAMED O A. Routing and scheduling of time-triggered traffic in time-sensitive networks[J]. *IEEE Transactions on Industrial Informatics*, 2020, 16(7): 4525-4534.
- [10] HELLMANN D, HAUG L, HILDEBRAND M, et al. How to optimize joint routing and scheduling models for TSN using integer linear programming[C]//*Proceedings of the 29th International Conference on Real-Time Networks and Systems*. New York: ACM Press, 2021: 100-111.
- [11] VLK M, HANZÁLEK Z, BREJCHOVÁ K, et al. Enhancing schedulability and throughput of time-triggered traffic in IEEE 802.1Qbv time-sensitive networks[J]. *IEEE Transactions on Communications*, 2020, 68(11): 7023-7038.
- [12] DÜRR F, NAYAK N G. No-wait packet scheduling for IEEE time-sensitive networks (TSN)[C]//*Proceedings of the 24th International Conference on Real-Time Networks and Systems*. New York: ACM Press, 2016: 203-212.
- [13] RAAGAARD M L, POP P, GUTIÉRREZ M, et al. Runtime reconfiguration of time-sensitive networking (TSN) schedules for fog computing[C]//*Proceedings of the 2017 IEEE Fog World Congress (FWC)*. Piscataway: IEEE Press, 2017: 1-6.
- [14] PAHLEVAN M, TABASSAM N, OBERMAISSER R. Heuristic list scheduler for time triggered traffic in time sensitive networks[J]. *ACM SIGBED Review*, 2019, 16(1): 15-20.
- [15] PAHLEVAN M, OBERMAISSER R. Genetic algorithm for scheduling time-triggered traffic in time-sensitive networks[C]//*Proceedings of the 2018 IEEE 23rd International Conference on Emerging Technologies and Factory Automation (ETFA)*. Piscataway: IEEE Press, 2018: 337-344.
- [16] ARESTOVA A, HIELSCHER K S J, GERMAN R. Design of a hybrid genetic algorithm for time-sensitive networking[C]//*Proceedings of International Conference on Measurement, Modelling and Evaluation of Computing Systems*. Cham: Springer, 2020: 99-117.
- [17] 曹志鹏, 刘勤让, 刘冬培, 等. 面向时间敏感网络的流量调度方法[J]. *计算机工程*, 2021, 47(7): 168-175, 182.
- CAO Z P, LIU Q R, LIU D P, et al. Traffic scheduling method for time-sensitive network[J]. *Computer Engineering*, 2021, 47(7): 168-175, 182.
- [18] JIA H Z, NEE A Y C, FUH J Y H, et al. A modified genetic algorithm for distributed scheduling problems[J]. *Journal of Intelligent Manufacturing*, 2003, 14(3): 351-362.
- [19] WANG Y, CHEN J D, NING W, et al. A time-sensitive network scheduling algorithm based on improved ant colony optimization[J]. *Alexandria Engineering Journal*, 2021, 60(1): 107-114.
- [20] GAVRILUT V, ZHAO L X, RAAGAARD M L, et al. AVB-aware routing and scheduling of time-triggered traffic for TSN[J]. *IEEE Access*, 2018, 6: 75229-75243.
- [21] GAVRILUT V, POP P. Scheduling in time sensitive networks (TSN) for mixed-criticality industrial applications[C]//*Proceedings of the 2018 14th IEEE International Workshop on Factory Communication Systems (WFCS)*. Piscataway: IEEE Press, 2018: 1-4.

- [22] YEN J Y. Finding the K shortest loopless paths in a network[J]. Management Science, 1971, 17(11): 712-716.
- [23] FEO T A, RESENDE M G C. Greedy randomized adaptive search procedures[J]. Journal of Global Optimization, 1995, 6(2): 109-133.
- [24] 曹旭, 周学哲, 宋亚平, 等. 基于时延抖动与信道使用量约束的供热网络无线 TSN 流量调度算法[J]. 移动通信, 2023, 47(8): 28-32, 45.
- CAO X, ZHOU X Z, SONG Y P, et al. A traffic scheduling algorithm for wireless TSN in heating networks based on delay jitter and channel usage constraints[J]. Mobile Communications, 2023, 47(8): 28-32, 45.
- [25] BERISA A, ZHAO L X, CRACIUNAS S S, et al. AVB-aware routing and scheduling for critical traffic in time-sensitive networks with preemption[C]//Proceedings of the 30th International Conference on Real-Time Networks and Systems. New York: ACM Press, 2022: 207-218.
- [26] SERNA OLIVER R, CRACIUNAS S S, STEINER W. IEEE 802.1Qbv gate control list synthesis using array theory encoding[C]//Proceedings of the 2018 IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS). Piscataway: IEEE Press, 2018: 13-24.
- [27] CRACIUNAS S S, OLIVER R S. Combined task- and network-level scheduling for distributed time-triggered systems[J]. Real-Time Systems, 2016, 52(2): 161-200.

[作者简介]



李传煌 (1980-), 男, 博士, 浙江工商大学教授、硕士生导师, 之江实验室双聘专家, 主要研究方向为新一代网络技术、边缘计算、人工智能应用。



廖君虎 (1998-), 男, 浙江工商大学硕士生, 主要研究方向为软件定义网络、移动网络优先等。



宣家栋 (1998-), 男, 硕士, 浙江工商大学毕业生, 主要研究方向为软件定义网络、时间敏感网络等。

徐琪 (1992-), 男, 博士, 之江实验室副研究员, 主要研究方向为多模态网络、软件定义网络、时间敏感网络等。

张慧峰 (1992-), 女, 博士, 之江实验室助理研究员, 主要研究方向为多模态网络、新型网络拓扑结构、时间敏感网络等。

朱俊 (1981-), 男, 博士, 之江实验室高级工程师, 主要研究方向为新型网络架构、软件定义网络、网络管理与控制技术、网络协议设计与优化、卫星网络等。

邹涛 (1974-), 男, 博士, 之江实验室研究员, 主要研究方向为多模态网络技术和新型网络体系架构。

张汝云 (1973-), 男, 博士, 之江实验室研究员, 主要研究方向为多模态网络技术、工业互联网和网络通信安全。