doi:10.11887/j.cn.202405020 http://journal.nudt.edu.cn

时间敏感网络延时组成的精确分析与测量方法

付文文,全 巍,姜旭艳,孙寅涵,孙志刚*
(国防科技大学 计算机学院,湖南 长沙 410073)

摘 要:在依据时间敏感网络(time-sensitive networking, TSN)标准估算相邻节点之间的最大传输延迟 Δt 时,将不可避免地引入额外的无效等待时间,这种等待时间被称为"泡沫延时"。分析了泡沫延时对增加端到端延时和降低规划成功率的负面影响,并通过细粒度地分析 Δt 的延时组成,首次提出 Δt 的精确测量方法。基于精确的 Δt,消除了 TSN 规划时产生的泡沫延时。基于两款定制的 TSN 交换设备搭建了真实的测试环境,测试结果显示,泡沫延时至少占端到端延时的 26.4%,并且消除泡沫延时后规划成功率提升了 8.9% ~ 39.1%。

关键词:时间敏感网络;端到端延时;规划成功率;延时参数测量;泡沫延时
中图分类号:TN915.03 **文献标志码**:A **文章编号**:1001-2486(2024)05-189-11

论文拓展

Precise analysis and measurement method for delay composition of time-sensitive networking

*FU Wenwen, QUAN Wei, JIANG Xuyan, SUN Yinhan, SUN Zhigang**
(College of Computer Science and Technology, National University of Defense Technology, Changsha 410073, China)

Abstract: When estimating the maximum transmission delay Δt between adjacent nodes according to TSN (time-sensitive networking) standards, it was inevitable to introduce additional ineffective waiting time, which was referred to as "bubble delay". The negative effect of bubble delay on increasing the end-to-end delay and decreasing the planning success rate was analyzed, and the precise measurement method of Δt was proposed for the first time based on analyzing the delay composition of Δt in fine granularity. Based on the exact Δt, the bubble delay caused by TSN planning was eliminated. A real test environment was built based on two customized TSN switching devices. The test results show that the bubble delay accounted for 26.4% of the end-to-end delay at least, and the planning success rate increased by 8.9% to 39.1% after eliminating the bubble delay.

Keywords: time-sensitive networking; end-to-end delay; planning success rate; delay parameter measurement; bubble delay

时间敏感网络(time-sensitive networking, TSN)通过在传统以太网技术基础上增加时间相关功能(时间同步和时间感知整形等),使其能够继承传统以太网良好兼容性和高带宽等特点,同时保证关键流量传输的确定性和实时性[1-2],关键流量在 TSN 中是被调度的对象,因此也被称为被调度流量(scheduled traffic, ST)。这使 TSN 技术的应用场景从音视频传输和工业控制扩展至 5G 前传、汽车自动化和航空航天等多个领域[3]。

与软件定义网络[4](software-defined network, SDN)类似,TSN 系统集成了数据平面与控制平面的组件。系统工作前,TSN 规划工具(控制平面组件)采用约束求解的方式规划所有 ST 报文的传输路径以及在传输路径上各节点的发送时间,各 TSN 设备(数据平面组件)根据规划结果执行按时转发操作,进而保证 ST 流的确定性传输。延时约束是规划求解时必不可少的一类约束,包括流约束和流隔离约束等[5-6]。延时约束的核心输入参数是 ST 报文在相邻节点间传输可能经历的最大传输延时。Δt 表示在相邻节点

收稿日期:2022-05-12
基金项目:国防科技大学并行与分布处理国家级实验室开放基金资助项目(WDZC20205500110)
第一作者:付文文(1994—),男,江西南昌人,博士研究生,E-mail:fuwenwen94@nudt.edu.cn
* **通信作者**:孙志刚(1973—),男,江苏东海人,研究员,博士,博士生导师,E-mail:sunzhigang@nudt.edu.cn
引用格式:付文文,全巍,姜旭艳,等. 时间敏感网络延时组成的精确分析与测量方法[J]. 国防科技大学学报, 2024, 46(5): 189-199.
Citation: FU W W, QUAN W, JIANG X Y, et al. Precise analysis and measurement method for delay composition of time-sensitive networking[J]. Journal of National University of Defense Technology, 2024, 46(5): 189-199.

中,ST报文从上一节点开始调度输出至开始到达下一节点的输出队列中可能经历的最大延时。

TSN标准并未直接提供延时参数 Δt,其提供报文长度相关延时 *dependentDelay*(*len*)和报文长度无关延时 *independentDelay*[7]。报文在设备中经历的这两种延时之和表示报文在设备内头进头出的交换延时。不同TSN设备的实现架构和端口速率等特征可能不同,使得不同设备间的延时参数相差也很大。例如,在千兆TSN设备中,*dependentDelay*(*len*)一般为十微秒级,*independentDelay* 一般为微秒级,然而在百兆TSN设备中,*dependentDelay*(*len*)一般为百微秒级,*independentDelay* 为十微秒级。

当相邻节点采用不同的TSN设备时,目前的TSN延时参数不能用以准确地计算 Δt。TSN规划器在设置延时约束时需要根据TSN标准提供的延时参数对 Δt 进行预估,预估值设为 $\Delta t\_Estimate$。为了保证延时约束的正确性,$\Delta t\_Estimate$ 必须不小于实际的 Δt,因此 $\Delta t\_Estimate$ 的取值为相邻节点的最大交换延时之和,这导致 $\Delta t\_Estimate$ 可能远大于实际的 Δt。

为了对上述现状的后果进行分析,本文提出泡沫延时概念,以表示 $\Delta t\_Estimate$ 中超过实际 Δt 的无效延时。泡沫延时必然造成ST报文在交换节点中无效的等待延时,进而导致其传输延时急剧增加,增加的端到端延时是传输路径中各相邻节点间的泡沫延时之和。此外,泡沫延时使ST报文可分配的发送时间解空间缩小,进而降低规划成功率,导致规划算法的计算延时增加(具体分析见2.2节)。由于部分TSN应用对ST报文端到端延时和规划算法的计算延时具有苛刻要求(例如汽车中的动力总成和底盘控制要求ST报文的传输延时保持在10 μs以内[8],而且TSN系统中的虚拟机迁移要求规划工具进行高频的规划更新,为了提升系统的工作效率,规划算法的计算延时需要尽可能低[9]),因此消除泡沫延时以降低端到端延时和降低规划算法的计算延时成为目前亟待解决的问题。

为了消除泡沫延时,本文细粒度地分析 Δt 的延时组成。通过分析得出,Δt 包括相邻节点中上游节点的最大输出延时 *egressDelayMax*,下游节点的最大输入延时 *ingressDelayMax*,相邻节点间的链路传播延时和时钟偏差最大值 δ。与TSN标准提供的延时参数类似,*ingressDelayMax* 和 *egressDelayMax* 都由报文长度相关延时的最大值和报文长度无关延时的最大值组成。

为了获取精确的 Δt 值,本文提出了泡沫释放方法(bubble release method,BRM)。该方法首先通过精确地测量 Δt 的各组成延时,进而计算出准确的 Δt。基于准确的 Δt,该方法进一步将有泡沫延时约束优化成无泡沫延时约束,以消除泡沫延时带来的负面影响。

为了验证BRM的有效性,本文基于OpenTSN开源项目[10]定制了两款TSN交换设备,并搭配TSN测试仪(T200)搭建了真实的测试环境。在该环境中,本文首先通过BRM获取相邻节点间的精确 Δt 值,而后通过控制变量法对延时约束进行验证。

1 背景知识

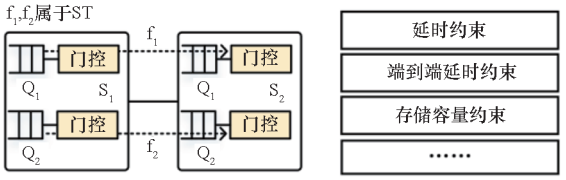
1.1 TSN基本原理

TSN技术的基本功能包括时间同步和时间感知整形。其中时间同步功能用以提供全局统一的时间观念。其功能原理为主从式时钟同步,即主时钟将本地时间分发给所有从时钟,从时钟根据主时钟分发的时间调整本地时钟,进而实现主从时钟的一致性[11]。

在TSN中各节点时间同步的前提下,时间感知整形功能[12]保证ST报文传输延时的确定性和实时性。该机制需要数据平面中TSN设备和控制平面中规划器协同实现。其中TSN设备在每个输出队列后附上一个门控(类似于开关)。当门控状态为开,其对应队列中的数据允许被调度输出。反之,则不允许。每个输出端口中所有输出队列的门控状态都记录在对应的门控列表(gate control lists,GCL)中,GCL内容由TSN规划器统一计算。

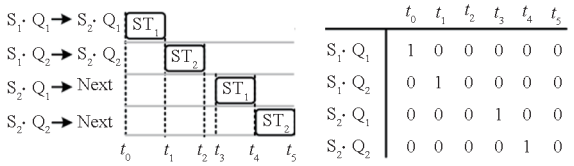
如图1所示,时间感知整形机制的工作流程主要分为以下四步。第一步,收集场景特征参数,如图1(a)所示。TSN规划器收集ST流特征(包括报文发送周期、报文长度、传输延时要求等)、设备特征(包括端口速率、设备延时参数等)和网络特征(网络拓扑等)。第二步,设置规划约束,如图1(b)所示。TSN规划器根据收集的特征对TSN系统进行建模,并设置与场景特征相适配的约束(例如延时约束),以保证规划结果满足目标应用的所有要求。第三步,计算发送时间窗口,如图1(c)所示。TSN规划器通过约束求解的方式计算满足所有约束时,每个ST报文在传输路径中的各节点输出队列的发送时间。发送时间一般以窗口的形式表示,包括发送开始时间和发送结束

时间。第四步,生成 GCL,如图 1(d)所示。TSN 规划器将 ST 报文的规划发送时间转换成每个输出端口的 GCL。转换规则为当有 ST 报文规划传输时,对应队列的门控状态为开,其他时间该队列的门控状态为关。例如,根据图 1(c)可知,ST_1 报文在 S_1 节点的 Q_1 队列发送时间窗口为[t_0, t_1],因此 S_1 节点的 Q_1 队列在时间点 t_0 时开门控,在时间点 t_1 时关门控,如图 1(d)所示。



(a) 场景特征　(b) 规划算法约束

(a) Scenario features　(b) Planning algorithm constraints



	t_0	t_1	t_2	t_3	t_4	t_5
$S_1 \cdot Q_1$	1	0	0	0	0	0
$S_1 \cdot Q_2$	0	1	0	0	0	0
$S_2 \cdot Q_1$	0	0	0	1	0	0
$S_2 \cdot Q_2$	0	0	0	0	1	0

(c) 规划结果　(d) 门控列表

(c) Planning solution　(d) Gate control lists

图 1　时间感知整形机制工作流程

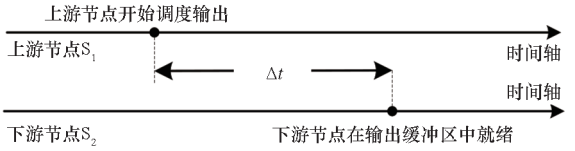
Fig. 1　Workflow of time-aware shaper

1.2　延时约束

为了保证规划的发送时间与实际的发送时间一致,进而保证 ST 流的传输确定性和实时性,TSN 规划器计算发送时间时需要设置一系列与场景特征相适配的规划约束,其中延时约束是必不可少的一类约束。延时约束指约束函数中将 Δt 作为变量的约束,例如参考文献[5-6]规划算法中提出的流约束和流隔离约束。

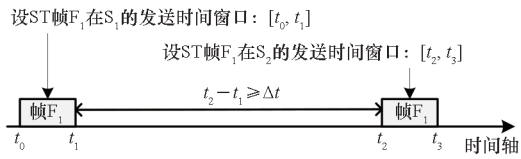
Δt 定义:在相邻节点间传输时,报文从上游节点的输出队列中开始调度输出至在下游节点输出队列中就绪所可能经历的最大延时,如图 2(a)所示。就绪状态指 ST 报文可立即从输出队列中被调度输出。

以延时约束中典型的流约束为例。流约束保证 ST 报文在任意节点的输出队列就绪后才允许被调度输出,如图 2(b)所示。以 ST_1 在相邻节点 S_1 和 S_2 间传输为例,流约束的表达式为 $t_2 - t_1 \geqslant \Delta t$,其中 t_1 为 ST_1 在上游节点规划的发送结束时刻, t_2 为 ST_1 在下游节点规划的开始发送时刻。流约束可保证 ST 报文在上游节点的发送窗口为[t_0,t_1]时,ST 报文在下游节点的输出队列中就绪的时刻必然早于 t_2,使 ST 报文在下游节点的实际开始发送时间也为 t_2,进而实现规划与实际发送时间的一致性。



(a) 时段 Δt

(a) Δt period



(b) 流约束(一种延时约束)

(b) Flow constraint(a delay constraint)

图 2　Δt 和流约束示意

Fig. 2　Δt and flow constrain

1.3　TSN 标准中的延时参数

TSN 标准并未直接提供 Δt,其提供报文长度相关延时 *dependentDelay* 和报文长度无关延时 *independentDelay*。这两种延时之和为设备的交换延时 *switchDelay*(文中所有参数含义如表 1 所示)。

表 1　参数列表

Tab. 1　Parameter list

参数	含义
Δt	报文在相邻节点间传输可能经历的最大传输延时
$\Delta t\_Estimate$	Δt 的估计值
bubbleDelay	泡沫延时, $\Delta t\_Estimate - \Delta t$
dependentDelay	报文在 TSN 设备中经历的报文长度相关延时的总和
independentDelay	报文在 TSN 设备中经历的报文长度无关延时的总和
deadline	应用允许的最大端到端延时
e2eDelay	端到端延时
spaceSize	有效解集合的个数
ingressDelayMax	设备的输入延时最大值
egressDelayMax	设备的输出延时最大值
linkDelay	链路传播延时
δ	时钟偏差最大值

报文长度相关延时(*dependentDelay*)定义:将交换延时划分成每个模块的处理延时。若计算报文在某个模块的处理延时需要将报文长度

作为输入参数，则该模块的处理延时属于 *dependentDelay*，例如存储和交换报文所需的延时。目前 TSN 设备只提供所有模块中报文长度相关延时之和 *dependentDelay*。该延时参数具有最大值 *dependentDelayMax* 和最小值 *dependentDelayMin*。

报文长度无关延时（*independentDelay*）定义：将交换延时划分成每个模块的处理延时。若报文在某个模块的处理延时与报文长度无关，则该模块的处理延时属于 *independentDelay*，例如处理报文首部所需的延时。目前 TSN 设备只提供所有模块中报文长度无关延时之和 *independentDelay*。该参数也包括最大值 *independentDelayMax* 和最小值 *independentDelayMin*。

2 动机

2.1 泡沫延时的产生

TSN 标准提供的报文长度相关延时和报文长度无关延时只能累加成设备的交换延时。当上下游节点使用不同 TSN 设备时，Δt 不等于任一节点的报文长度相关延时、报文长度无关延时、交换延时或任意两种延时之和。TSN 标准中的延时参数如图 3 所示，Δt 包含时间段 5 的部分延时、时间段 6、时间段 7 和时间段 8 的部分延时，而基于标准研发的 TSN 设备仅提供图 3 方框内的 4 个延时参数，无论方框内的延时参数怎么组合都不能构成精确的 Δt。因此基于 TSN 标准中提供的延时参数，无法精确计算 Δt，需要对 Δt 进行估算（预估值设为 $\Delta t\_Estimate$），并将图 2(b) 所示流约束设置成 $t_2 - t_1 \geqslant \Delta t\_Estimate$。

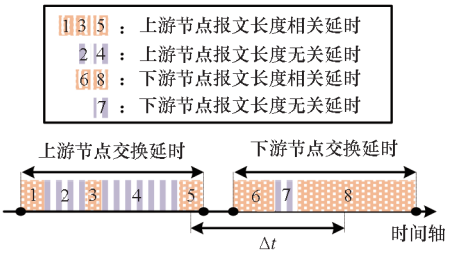


图 3 TSN 标准中的延时参数

Fig. 3 Delay parameters in TSN standard

若 $\Delta t\_Estimate$ 小于实际的 Δt，则可能发生规划发送时间与实际发送时间不一致的现象，造成 ST 帧传输延时不确定现象。具体以图 4 为例（传输 128 B 的 ST 报文时，实际 Δt 的取值为 6 μs）。在该示例中，设 $\Delta t\_Estimate$ 为 2 μs，满足 $\Delta t\_Estimate < \Delta t$，设 ST_1 报文在 S_1 节点的发送时间设为 [0,1] μs，在 S_2 节点的发送时间设为 [4,5] μs（$4 - 1 > \Delta t\_Estimate$，满足流约束）。此时当 ST 报文在 S_1 节点按照规划的时间发送时，ST 可能在 6 μs（$=0 + \Delta t$）才开始到达 S_2 节点的输出队列，使得 ST 报文在 S_2 节点的实际发送时间大于 6 μs，进而导致规划的发送时间与实际的发送时间不一致。由于 ST 报文未按照规划的发送时间发送，则可能占用其他 ST 报文的时间资源，产生崩溃式连锁反应，造成 ST 报文传输延时不确定现象。这对于任一 TSN 应用都是难以接受的，因此为了保证流约束的正确性，$\Delta t\_Estimate$ 取值必须恒大于实际的 Δt。

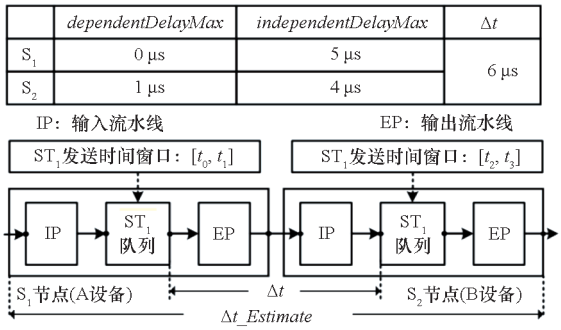


图 4 相邻节点的延时参数

Fig. 4 Delay parameters of adjacent nodes

为了分析上述现象，本文提出泡沫延时概念。泡沫延时（*bubbleDelay*）：因规划不当导致 ST 报文在下游节点中的无效等待延时。其表达式为 $bubbleDelay = \Delta t\_Estimate - \Delta t$。

根据 TSN 标准提供的延时参数，$\Delta t\_Estimate$ 的取值只能为上游节点的最大交换延时，或下游节点最大交换延时，或上下游节点的最大交换延时之和。为了保证 $\Delta t\_Estimate$ 大于 Δt 恒成立，$\Delta t\_Estimate$ 的取值必为上下游节点的最大交换延时之和。以图 4 为例，$\Delta t\_Estimate$ 的取值为 10 μs（$=5 + 1 + 4$），此时泡沫延时为 4 μs（$=10 - 6$）。

2.2 泡沫延时的负面影响

泡沫延时的负面影响主要包括增加端到端延时和降低规划解空间。

增加端到端延时。泡沫延时越大，ST 报文在下游节点的无效等待延时越长，进而导致端到端延时也相应地增加，额外增加的部分是传输路径中各相邻节点间的泡沫延时之和。当传输路径中各节点都为千兆 TSN 设备时，泡沫延时之和容易达到数十微秒或百微秒级，这对 ST 传输延时具有苛刻要求的 TSN 应用是难以接受的[3]。具体以汽车中底盘控制场景为例，其拓扑可简化成图 4

所示的线性拓扑。底盘控制要求 ST 报文的传输延时(报文开始进入 A 设备至从 B 设备输出的所需时间)保持在 10 μs 以内[3]。当规划时采用有泡沫延时约束时,ST 报文从 A 设备缓冲区开始发出至该报文从 B 设备缓冲区开始发出的延时至少为 10 μs(即 $\Delta t\_Estimate$ 值),加上报文在 A 设备的输入延时和 B 设备的输出延时,必然导致无法找到可行解。因此针对该类场景需要消除延时约束中的泡沫延时,以实现无等待传输,进而获取可行解。

降低规划解空间。解空间为满足所有规划约束的发送时间集合的个数,其中每个集合都记录了所有 ST 报文在传输路径中各节点的规划发送时间窗口。另一个相对的概念为搜索空间。搜索空间为发送时间集合的个数,包括满足或不满足所有规划约束的解。解空间/搜索空间为搜索一次获取可行解的概率。搜索空间不变时,解空间越小,每次搜索获取可行解的概率越小,相应地所需的搜索次数就越多,这无疑会增加计算可行解所需的时间,导致难以满足部分应用对规划延时的要求。

解空间的大小与规划约束密切相关。设 ST 流的报文在传输路径中第 i 个节点的发送时间为$[t_{2i},t_{2i+1}]$,其中 $i\geqslant 0$(当 $i=0$ 时,表示源节点)。流传输约束如式(1)所示,Δt_i 表示第 i、$i-1$节点间的 $\Delta t\_Estimate$。当流传输约束满足式(2)所示的关系时,可获取 ST 流的最小端到端延时。

$$t_{2i}-t_{2i-1}\geqslant \Delta t_i, i\geqslant 1 \tag{1}$$

$$t_{2i}-t_{2i-1}=\Delta t_i, i\geqslant 1 \tag{2}$$

而且为了保证 ST 流的实时性要求,应用要求 ST 流的端到端延时具有上限,上限值设为 *deadline*,因此需要设置 *deadline* 约束,其表达式如式(3)所示,其中第 j 个节点为该 ST 流的目的节点。

$$t_{2j+1}-t_0\leqslant deadline \tag{3}$$

根据式(2)可获取最小端到端延时 $e2eDelay(\min)$,其表达式如式(4)所示。其中 *len* 为报文长度,*Rb* 为链路传输速率,*len*/*Rb* 等于 $t_{2j+1}-t_{2i}$,即报文发送窗口尺寸。

$$e2eDelay(\min)=\sum_{i=1}^{j}\Delta t_j+j\frac{len}{Rb} \tag{4}$$

结合式(3)和式(4),当解空间大于 0 时,可知式(5)所示表达式恒成立。

$$\sum_{i=1}^{j}\Delta t_j+j\frac{len}{Rb}\leqslant deadline \tag{5}$$

因此各节点累计的等待延时小于 *deadline* 与 $e2eDelay(\min)$的差值,即满足式(6)所示表达式时,规划的发送时间集合为流约束可行解。

$$\sum_{i=1}^{j}\Delta t_j+j\frac{len}{Rb}\leqslant deadline \tag{6}$$

进而可将求解空间大小问题转换成典型的排列组合问题,即将小于 $\sum_{i=1}^{j}\Delta t_j+j\frac{len}{Rb}$ 个相同的小球放入编号不同的 j 个盒子里。采用隔板法可获取解空间的大小 *spaceSize*。

$$spaceSize=\sum_{k=0}^{(deadline-\sum_{i=1}^{j}\Delta t_j-j\times\frac{len}{Rb})}c_{j+k-1}^{j-1} \tag{7}$$

根据解空间表达式可知,当 Δt_i 值越大时,解空间越小,进而导致求可行解所需的计算延时增加。

综上所述,为了降低端到端延时且增加解空间,缩小或消除泡沫延时成为目前亟待解决的关键问题。

3　Δt 延时的精确组成

缩小或消除泡沫延时的关键是精确获取 Δt。为了精确获取 Δt,首先对 Δt 延时组成进行细粒度分析,如图 5 所示。Δt 延时组成包括上游节点的输出延时最大值,链路传播延时,时钟偏差最大值,和下游节点的输入延时最大值。

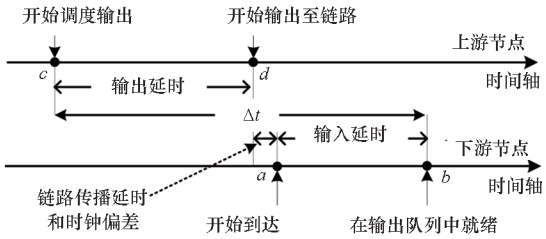


图 5　Δt 延时组成

Fig. 5　Δt delay composition

Δt 中的输入、输出延时基于典型的 TSN 设备实现架构进行描述,如图 6 所示。ST 报文在典型的 TSN 设备内首先经输入流水线处理,然后存储在共享缓冲中,同时报文摘要(携带报文关键信息)缓存在相应的输出队列中。当 ST 报文规划的发送时间到达时,输出流水线中的调度器将对应的 ST 报文摘要取出,而后根据摘要携带的地址信息从报文共享缓存中读取相应的报文并输出。

输入延时最大值 *ingressDelayMax* 定义:报文的开始进入设备至对应报文摘要到达输出队列所需的最大延时,即报文由图 6 中 a 点传输至 b 点

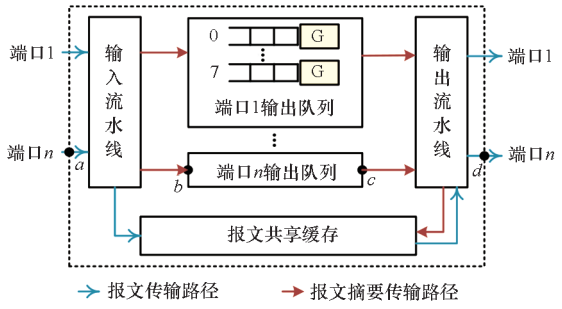


图 6 典型的 TSN 实现架构

Fig. 6 Typical TSN implementation architecture

可能经历的最大延时。*ingressDelayMax* 的组成部分包括报文长度相关延时和报文长度无关延时。其中报文长度相关延时使用函数式 $F_1(len)$ 表示,该函数式以报文长度 *len* 为变量,表示不同长度的报文在输入过程中经历的最大报文长度相关延时不同。报文长度无关延时使用常量 C_1 表示。

输出延时最大值 *egressDelayMax* 定义:报文摘要开始被调度至对应报文从链路输出时所需的最大延时,即报文由图 6 中 *c* 点传输至 *d* 点可能经历的最大延时。*egressDelayMax* 的组成部分也包括报文长度相关延时和报文长度无关延时。与输入延时最大值相似,输出延时最大值中报文长度相关延时使用函数式 $F_2(len)$ 表示,报文长度无关延时使用常量 C_2 表示。

链路传播延时 *linkDelay* 定义:报文开始从上游节点输出到链路至该报文开始输入至下游节点所需的最大延时,可通过链路长度除以链路传播速率计算得出,链路传播速率为常量。

时钟偏差最大值 δ 定义:上下游节点的时钟差值,其值与上下游节点的同步频率和晶振品质等因素相关。为了 Δt 的取值恒大于实际的 Δt 值,在设置 Δt 时需要考虑最坏情况下的时钟偏差。并且为了简化计算,时钟偏差一般设置为 TSN 域中任意两节点最大的时钟差值。例如,最大的主从时钟偏差为 100 ns,最小的主从时钟偏差为 −100 ns,则时钟偏差设置为 200 ns。

4 泡沫释放方法

基于 Δt 的精确组成,提出 BRM。该方法对设备中组成 Δt 的各延时参数组成进行精确测量,具体步骤如下。

步骤 1:测量时钟偏差最大值 δ。首先在 TSN 中设定质量最好的时钟为主时钟(可通过最佳主时钟算法选举产生或静态配置[13]),其他时钟设定为从时钟。而后收集一段长时间(一般以 d 为时间单位)内所有时钟的主从时钟差值(从时钟值大于主时钟值时,差值为正,否则为负)。最后将收集到的时钟差值的最大值减去最小值,进而计算出 δ。

步骤 2:测量链路传播延时 *linkDelay*。该步可直接测量链路长度,将链路长度除以链路传播速率(3.0×10^8 m/s),进而计算出链路传播延时。当链路长度稍短时,*linkDelay* 可忽略。

步骤 3:测量输出延时最大值 *egressDelayMax*。该步要求测试仪与被测设备进行时钟同步,并且测试仪支持记录 ST 报文的发送与接收时间戳,具体操作如算法 1(伪代码)所示。首先将被测设备与测试仪直连,并且构造一条 ST 流,剩余带宽构造满负荷的背景流量。而后配置 ST 报文在被测设备的发送时间窗口 $[t_m, t_{m+1}]$,并且收集 ST 报文经被测设备到达测试仪的接收时间戳 t_r,则 $t_r - t_m$ 为粗略的输出延时。为了获取被测设备更精确的输出延时,还需减去被测设备至测试仪的单程链路传播延时,和报文开始进入测试仪至记录接收时间戳所经历的延时(本文称为接收记录延时 *rxRecordDelay*,由测试仪厂家提供)。基于每个 ST 报文的接收时间戳都能计算出对应的输出延时,然后对长时间收集的所有输出延时进行比较,以获取输出延时最大值 *egressDelayMax*。最后,不

算法 1 *egressDelayMax* 测量

Alg. 1 Measuring *egressDelayMax*

将测试仪与被测设备直连,测试仪注入一条特定报文长度的 ST 流和满负荷的背景流;
设 ST_{num} 为该条 ST 流的报文数量,
设定 ST 流在被测设备的发送时间窗口为 $[t_m, t_{m+1}]$。

1. 测试仪记录 ST 报文的接收时间戳 t_r;
2. 计算 $egressD_{temp} = t_r - t_m$;
3. 根据步骤 2 测量 *linkDelay*;
4. 记录接收记录延时 *rxRecordDelay*;
5. $egressDelay = egressD_{temp} - linkDelay - rxRecordDelay$;
6. 根据报文序号给测量的输出延时进行标号;
7. **For**($i=0; i<ST_{num}; i++$):
8. **If** $egressDelay_{i+1} > egressDelay_i$:
9. $egressDelayMax = egressDelay_{i+1}$;
10. **Else** $egressDelayMax = egressDelay_i$;
11. 调整 ST 长度,重复测量 *egressDelayMax*;
12. 分析得出 *egressDelayMax* 的表达式

断改变 ST 流的报文长度并通过上述步骤收集 *egressDelayMax*,通过分析可获取输出延时与报文长度的函数式 $egressDelayMax = F_1(len) + C_1$。

步骤 4:测量输入延时最大值 *ingressDelayMax*。输入延时等于交换延时(未在输出队列中排队)减去对应输出延时的最大值。具体步骤如算法 2 所示。交换延时的测量操作与输出延时类似:首先将被测设备与测试仪直连,并且构造一条 ST 流,剩余带宽构造满负荷的背景流量。而后通过设置门控使测试仪发出的 ST 报文到达被测设备的输出队列后立即转发输出给测试仪。测试仪收集 ST 报文的发送和接收时间戳。接收时间戳与发送时间戳的差值为粗略的交换延时。为了更精确地获取被测设备的交换延时,还需减去测试仪的回环延时 *loopD*(当 ST 报文从测试仪输出后直接从测试仪的另一个接口输入时,ST 报文的接收时间戳减去发送时间戳的差值为回环延时)。然后将获取的交换延时减去当前的输出延时,可计算得出输入延时。每个 ST 报文的时间戳都能测出对应的输入延时,再对长时间收集的所有输入延时进行比较,以获取被测设备的输入延时最大值 *ingressDelayMax*。最后不断改变报文长度并通过上述方法收集输入延时,进而可获取输入延时与报文长度的关系式 $ingressDealyMax = F_2(len) + C_2$。

算法 2　*ingressDelayMax* 测量

Alg. 2　Measuring *ingressDelayMax*

将测试仪与被测设备直连,测试仪注入一条特定报文长度的 ST 流和满负荷的背景流;
设 ST_{num} 为该条 ST 流的报文数量,
设 ST 流在被测设备中对应的队列门控状态持续为开。

1. 测试仪记录 ST 报文发送和接收时间戳,t_s 和 t_r;
2. 计算 $forwardD_{temp} = t_s - t_r$;
3. 记录厂家提供的测试仪回环延时 *loopD*;
4. $forwardDelay = forwardD_{temp} - loopD$;
5. 同时根据步骤 3 测量出 *egressDelay*;
6. $ingressDelay = forwardDelay - egressDelay$;
7. 根据 ST 报文序号给测量的输入延时进行标号;
8. **For**($i = 0; i < ST_{num}; i++$):
9. **If** $ingressDelay_{i+1} > ingressDelay_i$:
10. $ingressDelayMax = ingressDelay_{i+1}$;
11. **Else** $ingressDelayMax = ingressDelay_i$;
12. 调整 *ST* 报文长度,重复测 *ingressDelayMax*;
13. 分析得出 *ingressDelayMax* 的表达式

步骤 5:设置无泡沫延时约束。基于上述延时可计算出精确的 Δt,进而可将现有延时约束中的 $\Delta t\_Estimate$ 替换成精确的 Δt。具体以流约束为例,可将有泡沫流约束优化成式(8)所示的无泡沫流约束。使用无泡沫延时约束替换有泡沫延时约束是优化现有 TSN 规划算法的思路。该思路对于所有 TSN 规划算法都是通用的,与规划算法所采用的具体策略无关,因此使用已有的 TSN 规划算法[5,14-15]验证优化思路的正确性。

$$\begin{cases} t_{2i} - t_{2i-1} \geqslant \Delta t_i, i \geqslant 1 \\ \Delta t_i = \delta + linkDelay + egressDelayMax + ingressDelayMax \end{cases} \tag{8}$$

5　评估

为了对 BRM 的有效性进行测试,基于 OpenTSN 开源项目在 FPGA 平台上实现了两款 TSN 交换设备,并搭建了真实的 TSN 环境。其中一款交换设备为交换机 TSw1,该交换机基于 FPGA 实现,面向低功耗与低延时的场景,例如工业互联网。另一款交换设备为交换机 TSw2 该交换机面向 TSN 应用的高可靠性需求,在 TSw1 的输出逻辑中增加了完整性校验功能以监测 TSN 交换逻辑是否产生不可预测性的错误。

5.1　实验一:Δt 的组成延时测量

为了设置无泡沫延时约束,首先测量组成 Δt 的延时。由于组成 Δt 的延时参数是反映设备能力的参数,其值与网络拓扑的规模和形状无关,可对被测设备进行离线测量(即测量拓扑和设备部署拓扑无须一致)。设置如图 7 所示的测量拓扑。在该拓扑中,T200 测试仪支持时钟同步,并且支持给发送和接收的报文记录时间戳,PC 机用于产生满负荷的尽力而为(best-effort, BE)背景流。由于实验采用的网络长度为 20 cm,按照步骤 2 得出的 *linkDelay* 为 0.66 ns。由于该值太小,本实验忽略 *linkDelay*。

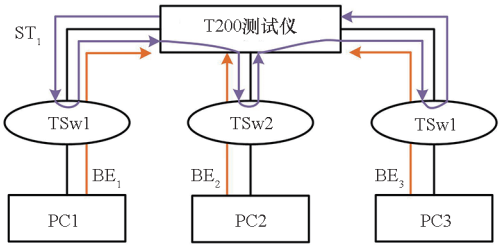


图 7　实验一拓扑

Fig. 7　Topology of experiment 1

时钟偏差最大值 δ 测量。由于时间同步是

TSN 技术的基础，本实验首先对 TSw1 和 TSw2 的 δ 进行测试。在该时钟同步测试实验中，*TSw*1（左）为主时钟（授时），TSw2 和 TSw1（右）为从时钟（对时），同步周期为 100 ms（即每隔 100 ms 执行一次同步操作）。时钟同步结果如图 8 所示，TSw2 与主时钟的时钟偏差始终保持在0 ~ 42 ns，TSw1（右）与主时钟的同步偏差保持在 - 48 ~ 0 ns，可知本实验中 δ 为 90 ns。

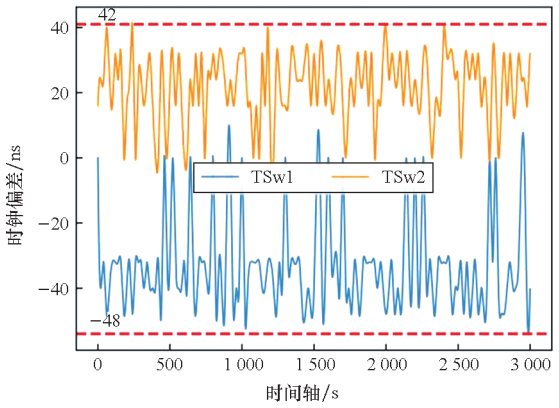


图 8　时钟同步结果

Fig. 8　Time synchronization result

输出延时 *egressDelayMax* 测量。在测试被测设备的输出延时时，T200 测试仪发送一条报文长度为 64 B 的 ST 流，3 个 PC 端发送动态的 BE 流。根据 BRM 的步骤 3 测量被测设备的输出延时，TSw1 的输出延时如图 9 蓝色三角形所示，*egressDelayMax* 为 1 522 ns。TSw2 的输出延时如图 10 蓝色三角形所示，*egressDelayMax* 为2 054 ns。

输入延时 *ingressDelayMax* 测量。根据 BRM 的步骤 4 测量输入延时，TSw1 和 TSw2 的头进头出交换延时如图 9 和图 10 红色五角星所示，交换延时的最大值分别为 3 372 ns 和 3 904 ns。输入延时分别如图 9 和图 10 绿色圆圈所示，其 *ingressDelayMax* 相同，都为 1 897 ns。

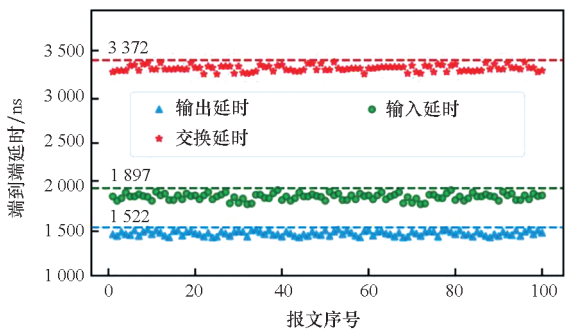


图 9　TSw1 节点的输入、输出、交换延时

Fig. 9　Ingress, egress and forwarding delay of TSw1

在输入与输出延时测量过程中，3 台 PC 端始终往 ST 流的传输路径中增加动态的 BE 流（流量速率和报文长度保持变化），然而测出的输入与输出延时基本保持稳定，可知 BE 流并未对 ST 流的输入与输出延时产生干扰。

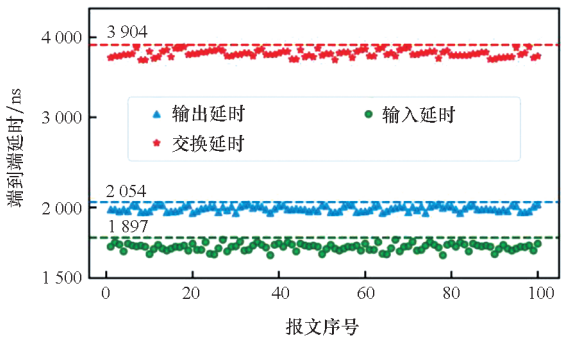


图 10　TSw2 节点的输入、输出、交换延时

Fig. 10　Ingress, egress and forwarding delay of TSw2

为了进一步获取 TSN 设备输入、输出延时最大值中的报文长度相关延时和报文长度无关延时，本实验改变 ST 报文的长度。ST 报文在 TSw1 和 TSw2 的 *egressDelayMax*、*ingressDelayMax* 分别如图 11 和 12 所示。根据图 11、图 12 结果可分析出，TSw1 的 *egressDelayMax*、*ingressDelayMax* 与报文长度无关。TSw2 的 *ingressDelayMax* 与报文长度无关，*egressDelayMax* 中报文长度无关延时最大值为 1 542 ns（ = 2 054 ns - 64 B/1 Gbit · s^{-1}），报文长度相关延时的表达式为 *len*/*Rb*，其中 *Rb* 为 1 Gbit · s^{-1}，即经历了一次存储交换过程，与实际相符。

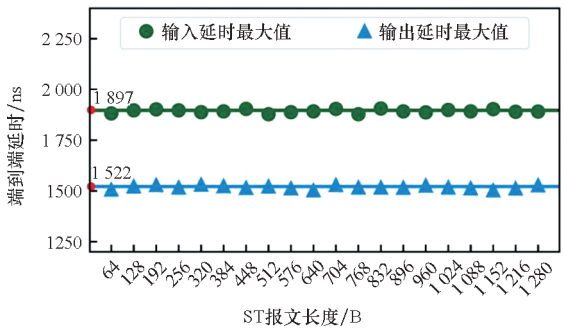


图 11　TSw1 的输入、输出延时最大值

Fig. 11　Maximum ingress and egress delay of TSw1

5.2　实验二：无泡沫延时约束的效果验证

为了证实前文分析结果与 BRM 的正确性，基于自研的两款 TSN 设备对图 4 所示的示例进行复现。该实验拓扑如图 13 所示，含一组异构的相邻节点，可作为绝大多数真实 TSN 场景拓扑的子集。因此该实验结果可反映真实场景（5G、汽车和航空航天等网络）中异构相邻节点中的境况。

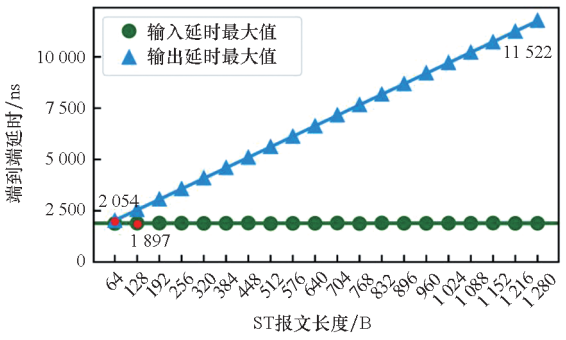


图 12　TSw2 的输入、输出延时最大值

Fig. 12　Maximum ingress and egress delay of TSw2

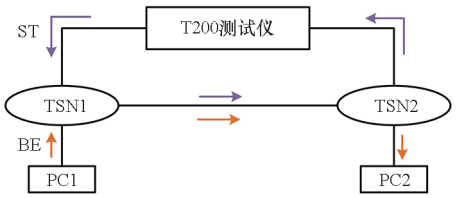


图 13　实验二拓扑

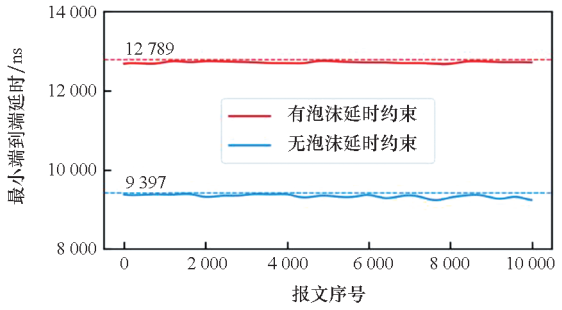
Fig. 13　Topology of experiment 2

由于本实验的链路长度很短，链路传播延时可忽略（与实验一相同）。TSw1 节点与 TSw2 节点间的 Δt 包括 TSw1 节点的最大输出延时（1.522 μs）、TSw2 节点的最大输入延时（1.897 μs）和时钟偏差 δ（0.090 μs）。为了降低计算复杂性，本实验将 Δt 设置为 $\lceil 1.522 + 1.897 + 0.090 \rceil = \lceil 3.509 \rceil = 4$ μs。由于分别测量 Δt 的组成延时属于悲观的测量方法，且纯硬件的测量方式可使精度保持在硬件时钟级（8 ns），结余的 0.491 μs（=4 μs − 3.509 μs）足够容忍时钟级测量误差。

当求解发送时间窗口时，沿用具有代表性的规划算法[5]。该算法是 TSN 技术龙头企业（TTTech 公司）在 2018 年于顶级会议论文中提出，并且得到了实际运用，因此该算法思想（如网络模型和约束形式化表达）在学术界和工业界都具有代表性。对该算法进行复现，并开源了复现代码，算法源码见 https://gitee.com/opentsn/open-planner。

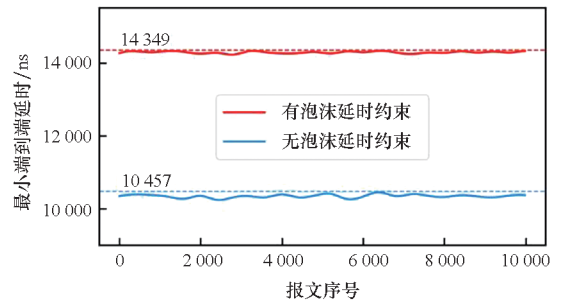
设 ST 报文在 TSw1 节点和 TSw2 节点的规划发送时间窗口分别为 $[t_1, t_2]$ 和 $[t_3, t_4]$。无泡沫延时约束（流约束）设置为 $t_{2i} - t_{2i-1} = 4$ μs。然而有泡沫延时约束（流约束）将 Δt 设置为相邻节点的交换延时最大值，即 $t_{2i} - t_{2i-1} = 1.897 + 1.522 + 1.897 + 1.542 + len/Rb = \lceil 6.858 + len/Rb \rceil$ μs。

实验结果指出泡沫延时大小与 ST 流数量和带宽，BE 流数量、带宽和报文长度都无关，仅与 ST_1 的报文长度相关。该发现与实验一的结论是一致的。改变 ST 报文长度，ST 报文在不同延时约束下，端到端延时如图 14 所示，其收益如表 2 所示。相比于有泡沫延时约束，当 ST 报文长度不同时，无泡沫延时约束可降低 26.4% ~30.8% 的端到端延时。



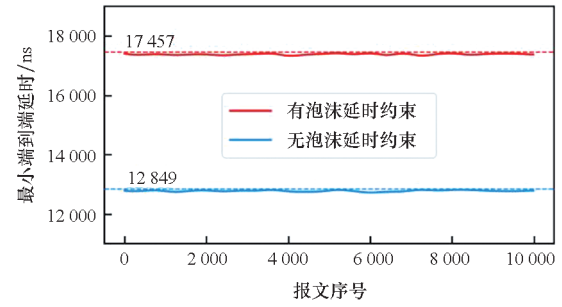
（a）报文长度为 64 B

（a）Frame size is 64 B



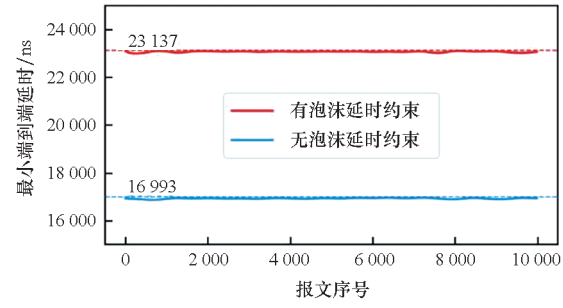
（b）报文长度为 128 B

（b）Frame size is 128 B



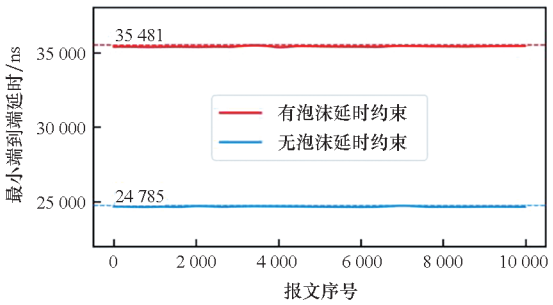
（c）报文长度为 256 B

（c）Frame size is 256 B



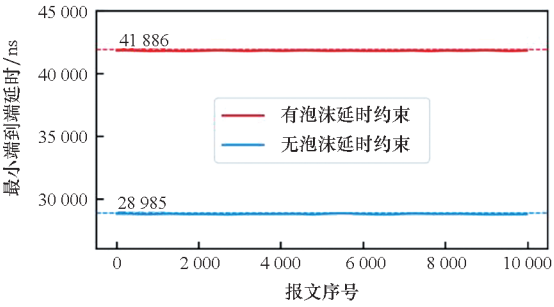
（d）报文长度为 512 B

（d）Frame size is 512 B



(e) 报文长度为 1 024 B

(e) Frame size is 1 024 B



(f) 报文长度为 1 280 B

(f) Frame size is 1 280 B

图 14 当 ST 报文长度不同时,有和无泡沫延时约束下的最小端到端延时

Fig. 14 Minimum end-to-end delay under foam delay constraints with and without different ST frame size

表 2 无泡沫延时约束的端到端延时收益

Tab. 2 End-to-end delay benefit without bubble delay constraint

帧长度/B	有泡沫延时下的端到端延时/ns	无泡沫延时下的端到端延时/ns	传输延时收益/%
64	12 789	9 397	26.5
128	14 349	10 457	27.1
256	17 457	12 849	26.4
512	23 137	16 993	26.6
1 024	35 481	24 785	30.1
1 280	41 886	28 985	30.8

当 ST 应用要求的 *deadline* 为典型的100 μs时(智能驾驶辅助系统中,ST 报文理想的端到端延时),根据式(7)所示的解空间表达式可计算出不同延时约束下的解空间。无泡沫延时约束的解空间收益如表3 所示,当 ST 报文长度不同时,相比于有泡沫延时约束,无泡沫延时约束增加 8.9% ~39.1% 的解空间。

表 3 无泡沫延时约束的解空间收益

Tab. 3 Solution space benefit without bubble delay constraint

帧长度/B	有泡沫延时下的解空间	无泡沫延时下的解空间	解空间收益/%
64	4 186	4 560	8.9
128	4 095	4 465	9.0
256	3 916	4 371	11.6
512	3 570	4 186	17.3
1 024	2 926	3 828	30.8
1 280	2 628	3 655	39.1

6 结论

本文针对现有延时约束中的缺陷,首次提出泡沫延时概念,并基于泡沫延时对该缺陷进行分析,明确指出基于 TSN 标准提供的延时参数设置延时约束中的关键参数 Δt 会额外引入泡沫延时,导致 ST 报文的端到端延时增加且规划解空间缩小。为了弥补该缺陷,对 Δt 的延时组成进行了细粒度分析,并且提出了 BRM 以指导用户精确地测量 Δt 与设置无泡沫延时约束。在真实的测试环境中证实了 BRM 的有效性。具体结果为无泡沫延时约束可降低 26.4% ~30.8% 的端到端延时,并且增加 8.9% ~39.1% 的规划解空间。

致谢

感谢华芯通科技有限公司提供的 TSN 测试仪,以及华芯通唐路、陈波提供的技术支持。

参考文献(References)

[1] LAN/MAN Standards Committee of the IEEE Computer Society. IEEE standard for local and metropolitan area networks-bridges and bridged networks: IEEE Std 802.1Q-2014 [S]. New York: the Institute of Electrical and Electronics Engineers, Inc., 2014.

[2] YAN J L, QUAN W, YANG X R, et al. TSN-builder: enabling rapid customization of resource-efficient switches for time-sensitive networking [C]//Proceedings of the 57th ACM/IEEE Design Automation Conference (DAC), 2020.

[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] MCKEOWN N, ANDERSON T, BALAKRISHNAN H, et al. OpenFlow: enabling innovation in campus networks [J]. ACM SIGCOMM Computer Communication Review, 2008, 38(2): 69-74.

[5] 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, 2016.

[6] SERNA OLIVER R, CRACIUNAS S S, STEINER W. IEEE 802.1Qbv gate control list synthesis using array theory encoding[C]//Proceedings of the IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS), 2018.

[7] ISO, IEC. Telecommunications and exchange between information technology systems: requirements for local and metropolitan area networks: Part 1Q: bridges and bridged networks amendment 31: Stream Reservation Protocol (SRP) enhancements and performance improvements: International Standard ISO/IEC/IEEE 8802-1Q-2021[S]. Switzerland: the Institute of Electrical and Electronics Engineers, Inc., 2021

[8] YANG J Y, AI B, YOU I, et al. Ultra-reliable communications for industrial internet of things: design considerations and channel modeling[J]. IEEE Network, 2019, 33(4): 104 – 111.

[9] YU Q H, WAN H, ZHAO X B, et al. Online scheduling for dynamic VM migration in multicast time-sensitive networks[J]. IEEE Transactions on Industrial Informatics, 2019, 16(6): 3778 – 3788.

[10] QUAN W, FU W W, YAN J L, et al. OpenTSN: an open-source project for time-sensitive networking system development[J]. CCF Transactions on Networking, 2020, 3: 51 – 65.

[11] LAN/MAN Standards Committee of the IEEE Computer Society. IEEE standard for local and metropolitan area networks—timing and synchronization for time-sensitive applications: IEEE Std 802.1AS-2020[S]. New York: the Institute of Electrical and Electronics Engineers, Inc., 2020.

[12] LAN/MAN Standards Committee of the IEEE Computer Society. IEEE standard for local and metropolitan area networks—bridges and bridged networks-amendment 25: enhancements for scheduled traffic: IEEE Std 802.1Qbv-2015[S]. New York: the Institute of Electrical and Electronics Engineers, Inc., 2015.

[13] Technical Committee on Sensor Technology (TC-9) of the IEEE Instrumentation and Measurement Society. IEEE standard for a precision clock synchronization protocol for networked measurement and control systems: IEEE Std 1588 – 2019[S]. New York: the Institute of Electrical and Electronics Engineers, Inc., 2020.

[14] STEINER W. An evaluation of SMT-based schedule synthesis for time-triggered multi-hop networks[C]//Proceedings of the 31st IEEE Real-Time Systems Symposium, 2010.

[15] VLK M, HANZALEK Z, BREJCHOVA 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.