基于 Telemetry 架构的数据中心网络纳秒级时间同步

张千里，张超凡，王继龙，唐翔宇，沈钲晨，王会
（清华大学网络科学与网络空间研究院，北京 100084）

摘 要：为了解决传统的局部时间同步架构与集中式管理脱节、需要额外交互开销、时间数据较少易受离群值影响等问题，针对数据中心网络的高精准时间同步需求，提出了基于 Telemetry 架构的纳秒级时间同步系统。该系统借助数据中心的背景流量收集时间信息，结合离群因子检测算法对汇报上来的大量时间信息进行分析处理，达成一种集中式的纳秒级自动同步，便于后续全局网络调度。所提系统在可编程交换机上实现并进行了评估，结果显示在多跳之间及不同链路速率下，节点间均可以达到纳秒级的高精准时间同步。

关键词：高精准时间同步；纳秒级；数据中心网络；Telemetry 架构
中图分类号：TP393
文献标识码：A
DOI: 10.11959/j.issn.1000−436x.2021147

Nanosecond level time synchronization in datacenter network based on Telemetry architecture

ZHANG Qianli, ZHANG Chaofan, WANG Jilong, TANG Xiangyu, SHEN Zhengchen, WANG Hui
Institute for Network Sciences and Cyberspace, Tsinghua University, Beijing 100084, China

Abstract: To solve the problems of the traditional local time synchronization architecture being disconnected from centralized management, requiring extra interaction overhead, meanwhile with less time data susceptible to outliers, the nanosecond level time synchronization system was proposed based on Telemetry for the high-precision time synchronization requirements of data center networks. Combined with the outlier factor detection algorithm, timestamp information within the background traffic of the data center was collected, the large amount of time information reported was processed and analyzed, and a centralized automatic synchronization with nanosecond level precision was eventually achieved, which was convenient for subsequent global network scheduling. The proposed system was implemented and evaluated on the programmable switch. Experimental results show that between multiple hops and at different link rates, the nodes can achieve high-precision time synchronization at the nanosecond level.

Keywords: high-precision time synchronization, nanosecond level, datacenter network, Telemetry architecture

1 引言

精准的时间同步对于分布式网络应用具有十分重要的意义，尤其是在数据中心网络中。以 Spanner 分布式数据库[1]为例，当网络时间同步精度可以达到 T 个时间单位以内时，为了维护数据的外部一致性，一次写操作完成后必须要等待 T 个时间单位的时钟不确定期，才能释放写锁，以保证该写操作的结果可以被其他节点正确地读取。在实时性要求较高的场景中，普通的时间同步精度（一般为毫秒级）将从根本上限制数据库的吞吐量和性能，所以纳秒级的高精准时间同步对于提升分布式数据库的性能是十分关键的。而且，高精准的时间同步可以让人们得到更精准的单向时延（OWD, one way delay）数据，以用于网络监测及研究。数据中心中细粒度的包级别调度可以减少拥塞、提升网络性能[2]，这也需要各节点间的精

收稿日期：2021−05−08；**修回日期**：2021−07−29
基金项目：国家重点研发计划基金资助项目（No.2017YFB0503703）
Foundation Item: The National Key Research and Development Program of China (No.2017YFB0503703)

准时间同步。另外，对某一时刻的网络状态进行快照记录也需要高精准的同步时钟[3]。总体来说，随着数据中心网络链路速度的不断提升，各类分布式应用以及网络管理调度对高精准的时间同步有着越来越精密的需求。

时间同步过程本质上是节点间单向时延的精确测算。若可以高精准地测算出节点间的单向时延，则可以再结合同一数据包在各个节点的时间戳，计算节点间的时间偏差（详见第 2 节）。

经典的时间同步方法如网络时间协议（NTP, network time protocol）[4]无法达到纳秒级的高同步精度；精准时间协议（PTP, precise time protocol）[5]实施成本较高，且高网络负载情况下效果较差。近年来，针对数据中心网络的高精准时间同步需求，有研究者设计了 DTP（datacenter time protocol）[6]、DPTP（data-plane time-synchronization protocol）[7]、HUYGENS[8]等时间同步方法，可以在数据中心网络中达到纳秒级的时间同步（详见第 2 节）。

但是，已有的时间同步方法还存在以下三点问题：1) 目前已有的时间同步方法都是局部的同步方式，数据中心网络的中心控制节点难以进行集中式的管理，也很难清晰地了解每对节点间的时间偏差情况，这与后续基于高精也是最关键的问题准时间同步的网络管理与调度是脱节的，该问题也是最关键的问题；2) 前述方法都要求同步时发送请求和回复数据包，许多还引入了自定义的新协议，应用于数据中心时需要额外的交互和管控；3) 一些时间同步方法由于协议设计不够精细（如 NTP[4]）或者硬件存储限制（如 DPTP[7]），可以用于计算同步偏差的时间数据较少，因此容易受到某些离群值的影响。

针对上述问题，本文结合软件定义网络（SDN, software defined network）[9]和网内计算[10]的思想，采用基于 Telemetry 架构的时间同步机制，要求网络中的节点在转发数据包的同时，向中心节点汇报数据包进出节点的时间信息，而中心节点得到各个节点的时间信息后，便可进行统计分析，计算出任意 2 个节点的时间偏差。该方法有以下四方面优势：1) 这种集中式的管理架构与后续基于高精准时间同步进行网络管理与调度是一脉相承的，中心节点可以计算出任意 2 个节点的时间偏差，之后可直接用于全局的网络管理优化；2) 与传统的时间同步架构不同，该架构不需要节点间进行请求回复数据包的交互，只需借助现有的流量来收集数据包进出节点的时间信息，并对汇报上来的时间信息进行分析处理，达成一种集中式的自动同步；3) 该方法可以在数据中心的大规模流量中得到大量的时间信息，并且结合一定的统计分析，达到更精准的纳秒级时间同步，尤其在去除离群值、优化最差情况方面表现较好；4) 该架构是流量密度自适应的，流量规模越大，得到的时间数据越多，就越能优化同步精度，而在流量很小甚至接近零的情况下，得到的时间信息较少，但此场景一般对于时间同步的需求也较低，因此该架构是一个流量密度自适应的自洽体系。

本文针对数据中心网络的高精度时间同步需求，提出了基于 Telemetry 架构的时间同步机制，并在可编程交换机上具体实施，在保证纳秒级同步精度的同时，改变了原来基于同步请求和回复数据包的模式，在中心节点汇总分析数据包进出节点的时间信息，方便后续基于高精准时间同步进行网络管理与调度。相比之前的时间同步方法，本文所提方法具有集中式分析管理、全局视野、大量数据分析、流量密度自适应性等特点。本文主要贡献如下。

1) 分析数据中心网络对于高精准时间同步的需求，提出了基于 Telemetry 架构的时间同步机制，借助数据中心的背景流量来收集数据包进出节点的时间信息，并对汇报上来的大量时间信息进行分析处理，达成一种集中式的自动同步，便于后续进行全局性网络管理调度。

2) 对中心控制节点初步计算的时间偏差数据再应用局部离群因子检测算法，去除其中的离群值，进一步提升时间同步精度，尤其在去除离群值、优化较差情况方面效果优异。

3) 在可编程交换机上实现了该基于 Telemetry 架构的时间同步系统，借助数据平面的可编程性，使用可编程协议无关包处理（P4, programmable protocol-independent packet processor）语言[11]编写了可编程交换机的转发逻辑，要求各节点在转发数据包的同时，向中心节点汇报数据包进出节点的时间信息。实验表明，通过中心节点对汇报上来的大量时间信息进行分析处理，在多跳之间、不同端口额定速率及实际数据包发送速率下，节点间均可以达到纳秒级的高精准时间同步；并且，任意 2 个节点的时间偏差都可以在中心节点得到，该全局视野之后可直接用于全局的网络管理优化。

2　相关工作

2.1　传统时间同步架构

传统的基于请求–回复数据包的同步流程如图 1 所示[4]。客户端和服务器节点进行请求数据包和回复数据包交互，当客户端节点收到回复数据包后，根据客户端发送时间戳 t_1、服务器接收时间戳 t_2、服务器发送时间戳 t_3 和客户端接收时间戳 t_4，按照式(1)计算出往返时延 RTT，再根据式(2)，取 RTT 的一半作为近似的单向时延，计算出与服务器节点的时钟偏差值 Offset，并根据该 Offset 值调整自己的时钟，与服务器时钟保持一致。

$$\mathrm{RTT}=(t_4-t_1)-(t_3-t_2) \tag{1}$$

$$\mathrm{Offset}=t_3-(t_4-\mathrm{OWD})\approx t_3-\left(t_4-\frac{\mathrm{RTT}}{2}\right)=\frac{(t_2-t_1)+(t_3-t_4)}{2} \tag{2}$$

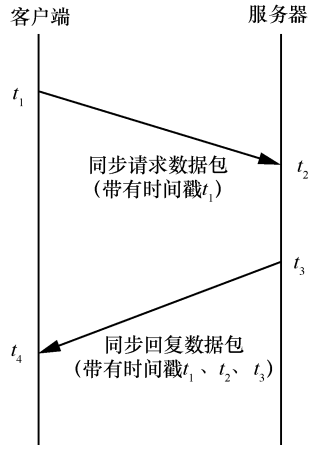


图 1　传统的基于请求–回复数据包的同步流程

传统的局部时间同步系统架构如图 2 和图 3 所示[4]。当系统局部的 2 个节点（主机 *A* 与主机 *B* 之间，或者交换机 S_1 与交换机 S_n 之间）要进行时间同步时，则这 2 个节点按照图 1 所示的流程进行同步请求和同步回复数据包交互。以主机 *A* 与主机 *B* 同步为例，*A* 发起同步请求，并收到 *B* 的同步回复数据包后，由时间戳信息 T_{AT_x}、T_{BR_x}、T_{BT_x}、T_{AR_x}（分别对应于 t_1、t_2、t_3、t_4）以及式(1)和式(2)计算与 *B* 的时钟偏差值，并调整自己的时钟与其一致。

但是在这种传统的局部时间同步架构下，数据中心网络的中心控制节点很难清晰地了解每对节点间的时间偏差情况，且难以进行全局性的集中式管理与调度；并且在这种架构下，同步的节点间需要发送请求和回复数据包，许多同步方法还引入了自定义的新协议，应用于数据中心时需要额外的交互和管控。另外，某些时间同步方法由于协议设计不够精细（如 NTP[4]）或者硬件存储限制（如 DPTP[7]），可用于计算同步偏差的时间数据较少，因此容易受到某些离群值的影响。

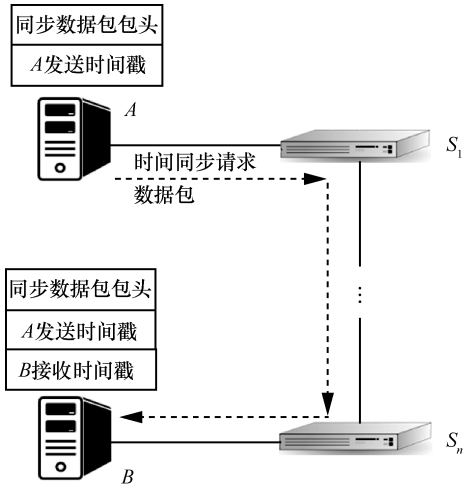


图 2　传统的局部时间同步系统架构（*A* 到 *B* 的同步请求）

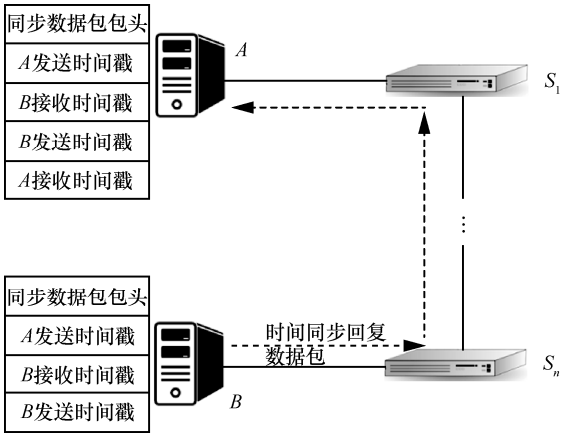


图 3　传统的局部时间同步系统架构（*B* 到 *A* 的同步回复）

本文仍会借鉴式(1)和式(2)进行 Offset 计算，但基于 Telemetry 架构重新设计了时间同步的流程，不再基于传统的请求–回复数据包的同步模式，而是借助数据中心的背景流量来收集数据包进出节点的时间信息，并对汇报上来的大量时间信息进行分析处理，达成一种集中式的自动高精准同步，便于后续进行全局性网络管理调度。

2.2　相关工作

网络时间协议[4]是互联网最古老的通信协议之一，目前仍广泛用于广域网和局域网的时间同步，在广域网中一般可以达到毫秒级的同步精度[12]，在

局域网中一般可达到微秒级的同步精度[13]。该方法同步精度较低，主要有以下三方面原因：1) 没有专门的硬件支持，时间戳本身精度较低，这从根本上限制了同步精度[14]；2) 协议设计在网络协议栈高层，请求回复数据包经过协议栈的时间不对称较大，导致 1/2 往返时延与真实的单向时延偏差较大[15]；3) 同步的 2 个节点可能跨越多跳，来回可能经过不同路径，面临不同的拥塞情况，也会导致来回的路径时延不对称[16]。

基于 IEEE 1588 的精准时间协议[5]在 NTP 的基础上进行了优化改进，引入了专门的硬件支持，可以获得更精准的时间戳，并且使用“透明时钟”以更精确的阶段性测量来减小来回路径的时间不对称性。在配置得当的情况下，PTP 一般可在局域网中达到几十或几百纳秒级的同步精度[17]，但该方法易受网络状态的影响，在链路拥塞时“透明时钟”往往不能很好地运行[18]，导致 PTP 在网络高负荷情况下表现较差[19]。

近年来针对数据中心网络的高精准时间同步需求，也有研究者进行了专门的研究工作。DTP[6]以专门设计的物理层架构 DTP-enable PHY 为基础，在原来 IEEE 802.3ae 规定的标准物理层中插入了一个 DTP 子层，专门用于时间同步；并且规定节点只与相邻的节点进行同步数据包交互（带有精确的时间信息），大幅减少了协议栈和来回路径的不对称性。该方法可以在数据中心场景中达到纳秒级的时间同步。

同样基于专门硬件的方法还有 DPTP[7]，它使用了基于可编程交换芯片的交换机，该交换机可以按照线路速率灵活处理数据包，并且可以给出各个阶段的精准时间戳。DPTP 使用可编程交换机中流水线各个阶段的精确时间戳进行计算，可以达到纳秒级的同步精度。

除了使用专门硬件的时间同步方法，还有一些方法如 HUYGENS[8]使用软件及算法相结合的方式达到了高精度的时间同步效果。该方法不需要专门硬件支持，只是利用网卡提供的硬件时间戳。通过更精致的计算，HUYGENS 可以在数据中心场景下达到纳秒级的同步效果。当然，由于算法较复杂，它也会带来较大的计算开销。

3 系统架构

3.1 基于 Telemetry 的时间同步系统架构

本文设计了基于 Telemetry 的时间同步系统架构，如图 4 和图 5 所示。

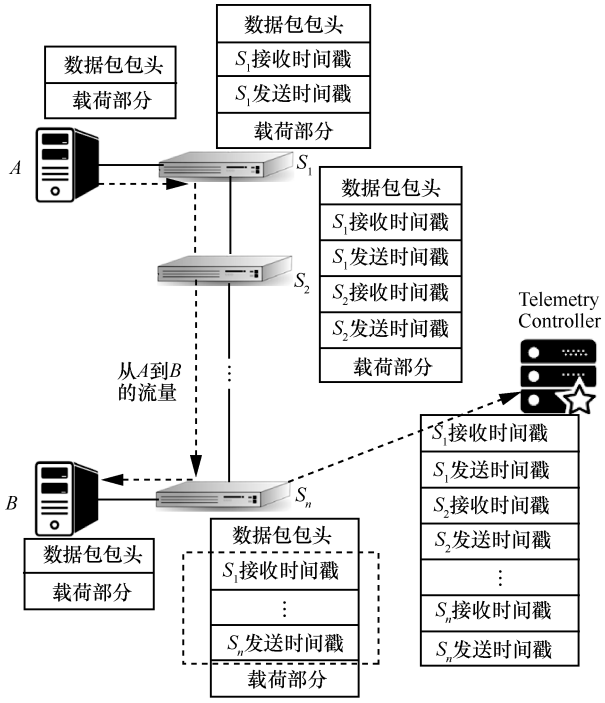


图 4 基于 Telemetry 的时间同步系统架构（*A* 到 *B* 流量）

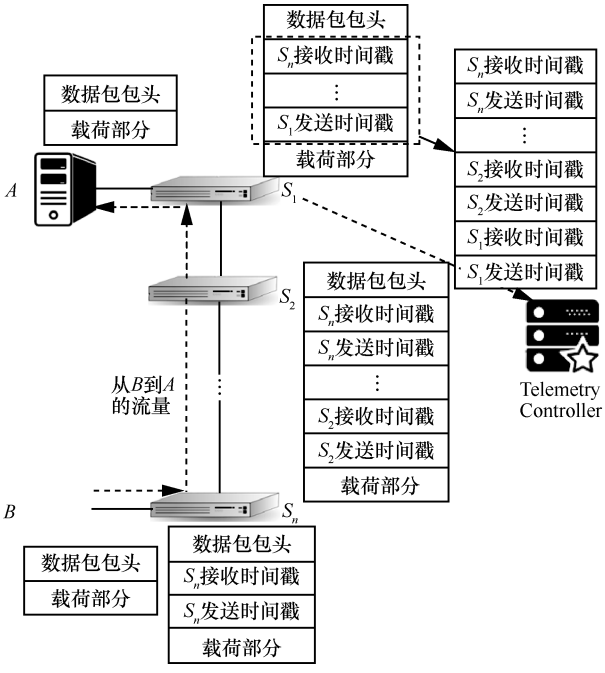


图 5 基于 Telemetry 的时间同步系统架构（*B* 到 *A* 流量）

图 4 中，当网络中存在从 A 到 B 的背景流量时，中间节点记录下接收数据包和发出数据包的时间戳，如 S_1 记录下数据包接收和发出的时间戳，分别记为 $T_{S_1R_x}$ 和 $T_{S_1T_x}$，S_2 在原来带内数据的基础上，再记录下 $T_{S_2R_x}$ 和 $T_{S_2T_x}$，依次类推，当数据包到达最后

一个中间节点 S_n 时，记录的所有时间戳数据 $\{T_{S_1R_x}, T_{S_1T_x}, T_{S_2R_x}, T_{S_2T_x}, \cdots, T_{S_nR_x}, T_{S_nT_x}\}$ 会被发送到中心控制节点 Telemetry Controller，然后 S_n 会将这些带内数据剥离，把原始的数据包发送给 B 节点。

类似过程如图 5 所示，当网络中存在从 B 到 A 的背景流量时，各个中间节点记录下接收数据包和发出数据包的时间戳，当数据包到达最后一个中间节点 S_1 时，记录的所有时间戳数据 $\{T'_{S_nR_x}, T'_{S_nT_x}, \cdots, T'_{S_2R_x}, T'_{S_2T_x}, T'_{S_1R_x}, T'_{S_1T_x}\}$ 会被发送到中心控制节点 Telemetry Controller，之后 S_1 会将这些带内数据剥离掉，把原始的数据包发送给 A 节点。

在这种时间同步系统架构下，节点间并没有显式地发送额外的交互数据包，不过数据中心的背景流量可以看作逻辑上的请求−回复数据包，Telemetry Controller 可以根据这些背景流量收集大量的时间戳数据，由此可以计算出任意 2 个中间节点的单向时延以及相对时间偏差值，依此可以进行时间同步，并且可以用于后续全局性的集中式管理与调度。以中间节点 S_1 和 S_n 为例，结合式(1)和式(2)可计算节点间单向时延及相对时间偏差值为

$$\mathrm{OWD}_{S_1,S_n} \approx \frac{\mathrm{RTT}_{S_1,S_n}}{2} = \frac{(T'_{S_1R_x} - T_{S_1T_x}) - (T'_{S_nT_x} - T_{S_nR_x})}{2} \tag{3}$$

$$\mathrm{Offset}_{S_1S_n} = \frac{(T_{S_nR_x} - T_{S_1T_x}) + (T'_{S_nT_x} - T'_{S_1R_x})}{2} \tag{4}$$

基于 $\{T_{S_1T_x}, T_{S_nR_x}, T'_{S_nT_x}, T'_{S_1R_x}\}$、式(3)和式(4)，可以计算出一组中间节点 S_1 和 S_n 的单向时延及相对时间偏差值。另外，中心控制节点也可以先计算出中间节点 S_1 和 S_2 的时钟偏差，再计算 S_2 和 S_3 的时钟偏差，依次类推，并将各个时间偏差值累加得到 S_1 和 S_n 的相对时间偏差值，如式(5)所示。

$$\mathrm{Offset}_{S_1S_n} = \sum_{i=1}^{n-1}\mathrm{Offset}_{S_iS_{i+1}} = \sum_{i=1}^{n-1}\frac{(T_{S_{i+1}R_x} - T_{S_iT_x}) + (T'_{S_{i+1}T_x} - T'_{S_iR_x})}{2} \tag{5}$$

后续会对式(4)和式(5)这 2 种计算时间偏差的效果进行对比评估（详见第 5 节）。

由于数据中心内存在大量的背景流量，Telemetry Controller 可以通过大量的时间戳数据计算出许多组时间偏差值，并通过一定的算法优化（本文使用的是局部离群因子（LOF, local outlier factor）检测法[20]，详见 3.2 节）来达到纳秒级的高精准时间同步。

基于 Telemetry 的时间同步架构有以下三方面优势：1) 在该架构下，中心控制节点的时间感知具有全局视野，可以计算出任意 2 个中间节点的单向时延以及时间偏差值，便于与后续全局网络管理调度相衔接；2) 该方法基于经典的 Telemetry 架构，时间信息捎带在 Telemetry 数据中，没有引入自定义的新协议，只需利用可编程数据平面对 Telemetry 的数据记录及汇总逻辑进行自定义；3) 由于数据中心有大量的背景流量，因此有大量的时间信息可供计算，可以再结合一些处理算法去离群值，提高时间偏差的测算精度。

3.2　基于 LOF 的 Telemetry Controller 分析处理

中心控制节点 Telemetry Controller 收到数据平面汇总上来的时间数据后，会对大量时间数据进行分析整理，通过式(4)或式(5)初步计算时间偏差值。在一些情况下，初步计算出的时间偏差值中可能会存在偏差较大的离群值，因此本文采用 LOF 检测法[20]去除其中的离群值，得到节点间更精确的时间偏差数据，以达成纳秒级的时间同步。

LOF 方法基于局部密度的概念而进行，局部密度显著低于其相邻节点的数据点被认为是离群值。

首先，对于某 2 个特定的中间节点，通过式(4)或式(5)分析整理大量时间数据，初步计算出了时间偏差值 $\{\mathrm{Offset}_1, \cdots, \mathrm{Offset}_x\}$。对于其中的每个数据点 m，计算该数据点与其 k 个最近邻节点的距离 $k\_\mathrm{dist}(m)$，并记所有与它距离小于或等于 $k\_\mathrm{dist}(m)$ 的其他数据点集合为 $N_k(m)$。

然后，对于每个数据点 m，计算 $N_k(m)$ 内所有数据点 o 与 m 的可达距离 $\mathrm{reach\_dist}(m,o)$

$$\mathrm{reach\_dist}(m,o) = \max(k\_\mathrm{dist}(o), \mathrm{dist}(m,o)) \tag{6}$$

对于每个数据点 m，计算其与 $N_k(m)$ 内所有数据点 o 可达距离均值的倒数，记为局部可达密度（LRD, local reachability density）

$$\mathrm{lrd}_k(m) = \frac{1}{\dfrac{\sum_{o\in N_k(m)}\mathrm{reach\_dist}(m,o)}{|N_k(m)|}} \tag{7}$$

最后，对于每个数据点 m，将计算出的局部可达密度与 $N_k(m)$ 内其他数据点的局部可达密度做比值并取平均，得到数据点 m 的 LOF 为

$$\mathrm{LOF}_k(m)=\frac{\sum_{o\in N_k(m)}\frac{\mathrm{lrd}(o)}{\mathrm{lrd}(m)}}{|N_k(m)|} \tag{8}$$

根据局部可达密度的定义，如果数据点 m 的 LOF 值远大于 1，则说明它的局部可达密度远小于邻近节点的密度，即该点是离群值。依照此原则筛除 $\{\mathrm{Offset}_1,\cdots,\mathrm{Offset}_x\}$ 中 LOF 值大于阈值的数据点，便可得到去离群值后的数据 $\{\mathrm{Offset}_1',\cdots,\mathrm{Offset}_y'\}$。

4　基于 Telemetry 的时间同步系统实现

基于 Telemetry 的时间同步系统实现如图 6 所示。在数据平面，本文利用可编程数据平面来自定义时间信息记录及汇总逻辑：在各个中间节点记录接收和发出数据包的时间戳，并在最后一跳中间节点将去除时间数据的原始数据包发送到端节点，将路径中记录下的时间数据发送到上层的 Telemetry Controller。

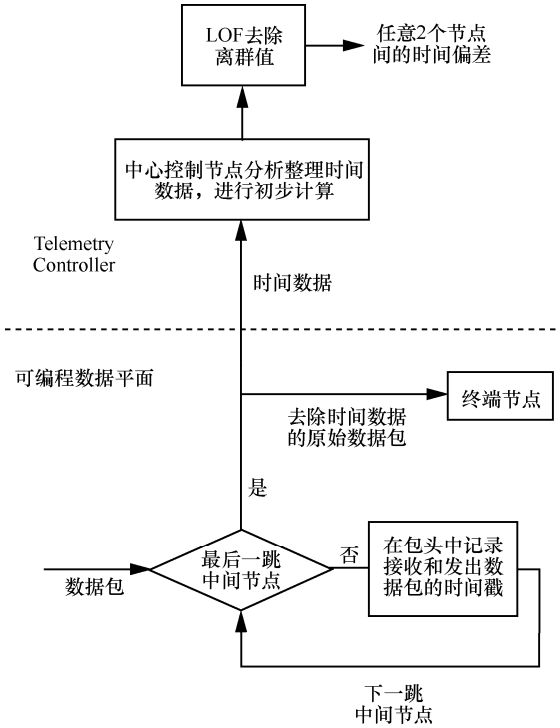


图 6　基于 Telemetry 的时间同步系统实现

在上层，Telemetry Controller 接收到汇报上来的时间戳后，会对大量的时间信息进行分析整理，利用式(4)或式(5)初步计算出任意 2 个节点的时间偏差，再结合 LOF 对大量数据进行处理，去除离群值，计算出任意 2 个中间节点间更准确的时间偏差，达到纳秒级的高精准同步精度。

4.1　基于 P4 的数据平面时间戳采集汇报

P4 语言[11]是软件定义网络[9]思想的最新展现形式。它在原来 OpenFlow 协议[22]控制平面可编程的基础上，带来了数据平面的灵活可编程性。

1) 在 RMT（reconfigurable match table）架构[23]的数据平面可编程芯片出现之后，基于这种专用的 ASIC（application specific integrated circuit）芯片，数据平面可以在不失高性能处理数据包的基础上，进行灵活的转发逻辑自定义，而用于定义数据平面逻辑的规范便是 P4 语言。

P4 抽象转发模型[11]如图 7 所示。数据包进入可编程交换机后会首先进入可重配置解析器（操作①），解析数据包头部；之后被解析出的头部会进入多级匹配–动作表，这些多级匹配–动作表以流水线的形式组织起来，分为入口流水线（操作②）和出口流水线（操作④）两部分，管理者可以根据 P4 语法规范去定义匹配到什么样的字段执行什么相应的操作，比如数据包修改、丢弃及出口选择；入口流水线与出口流水线之间数据包会进入一个缓冲区（操作③）；当出口流水线执行完毕后交换机会将修改后的数据包输出到指定的端口（操作⑤）。以上 5 个操作是 P4 抽象转发模型的基本操作，所有数据包进入可编程交换机都会经历这一流程，另外，P4 抽象转发模型也定义了数据包复制的操作，允许管理者自定义将数据包复制到其他输出端口（操作⑥）。

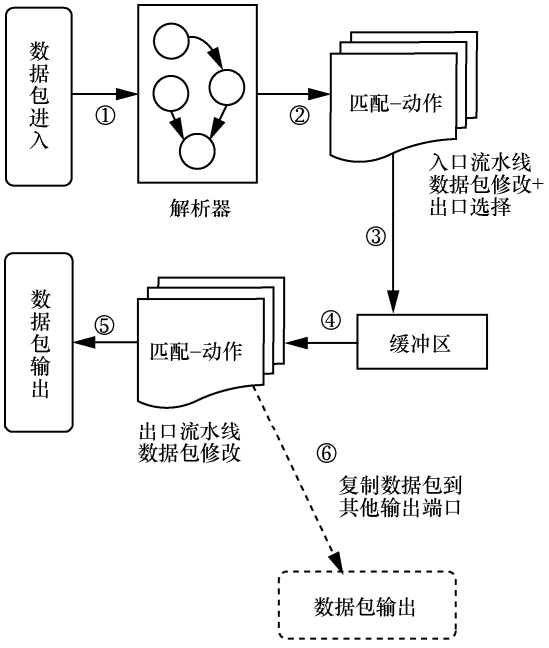


图 7　P4 抽象转发模型

具体到本文基于 P4 进行数据平面时间戳采集汇报，如图 6 所示，当数据包未到达最后一跳中间节点时，可编程交换机只对数据包执行图 7 中①~⑤操作，在数据包头中记录该节点接收和发出数据包的时间戳；当数据包到达最后一跳中间节点时，可编程交换机对数据包执行图 7 中①~⑥操作，一方面在出口流水线处将中间路径记录下的所有时间戳数据通过操作⑥复制，发送到上层的 Telemetry Controller，以待进一步分析整理，另一方面在出口流水线处将中间节点添加在数据包头中的所有时间戳去除，把去除时间数据的原始数据包发送到端节点。

4.2　基于 LOF 的 Telemetry Controller 分析处理

如图 6 所示，中心控制节点 Telemetry Controller 收到数据平面汇总上来的时间数据后，会对大量时间数据进行分析整理，按照式(4)或式(5)初步计算出 2 个节点间的时间偏差值，然后再用 LOF 方法[20]去除其中的离群值，得到节点间更精确的时间偏差，达成纳秒级的时间同步。LOF 方法的详细描述如算法 1 所示。

算法 1　LOF 方法

输入　中心控制节点根据 x 组中间节点 S_i 和 S_j 的时间数据（$\{T_{S_iR_x}, T_{S_iT_x}, \cdots, T_{S_jR_x}, T_{S_jT_x}\}$ 及 $\{T'_{S_jR_x}, T'_{S_jT_x}, \cdots, T'_{S_iR_x}, T'_{S_iT_x}\}$），通过式(4)或式(5)初步计算出的 x 个时间偏差 $\{\text{Offset}_1, \cdots, \text{Offset}_x\}$

输出　经过 LOF 方法去除离群值后的 y 个时间偏差 $\{\text{Offset}'_1, \cdots, \text{Offset}'_y\}$ $(y < x)$

1) 对于 $\{\text{Offset}_1, \cdots, \text{Offset}_x\}$ 中的每个数据点 m，计算其与它的第 k 个最近邻节点的距离 $k\_\text{dist}(m)$；

2) 对于每个数据点 m，记所有与它距离小于或等于 $k\_\text{dist}(m)$ 的其他数据点集合为它的 k 最近邻，记作 $N_k(m)$；

3) 对于每个数据点 m，根据式(6)计算 $N_k(m)$ 内所有数据点 o 与 m 的可达距离 $\text{reach}\_\text{dist}(m,o)$；

4) 对于每个数据点 m，根据式(7)计算局部可达密度，记作 $\text{lrd}_k(m)$；

5) 对于每个数据点 m，根据式(8)得到数据点 m 的局部离群因子 $\text{LOF}_k(m)$；

6) 筛除 $\{\text{Offset}_1, \cdots, \text{Offset}_x\}$ 中 LOF 值大于阈值的数据点，得到去离群值后的数据 $\{\text{Offset}'_1, \cdots, \text{Offset}'_y\}$。

5　系统评估

对时间同步效果的评估，具体而言就是比较测算得到的 2 个节点时间偏差值与真实节点间时间偏差值之间的误差，其关键在于怎样获取节点间真实的时间偏差。而在实际环境中由于设备差异、温度等实际因素的影响，节点间真实的时间偏差是很难精确获取的。本文应对这一挑战的办法为将一台物理可编程交换机抽象成逻辑上的多台可编程交换机，这样多台逻辑交换机之间的真实时间偏差值就是 0（因为它们物理上就是一台可编程交换机），这样将本文测算出的节点间时间偏差值与真实的时间偏差（即为 0）进行比较，就可以正常进行系统评估了。

具体操作时，如图 8 所示，本文将可编程交换机的端口间接上线路速率为 100 Gbit/s 的环回线，即可达到一台物理交换机抽象成多台逻辑交换机的效果。

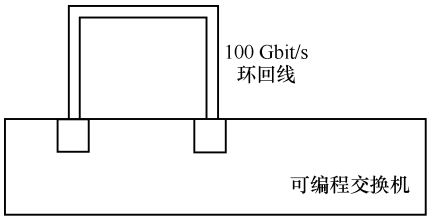


图 8　物理交换机抽象成多台逻辑交换机

5.1　实验环境

本文使用 Barefoot Tofino 可编程交换机、3 台戴尔服务器来搭建实验环境，其中，可编程交换机型号为 Wedge100BF-32X，配有 32 个 100 Gbit/s 端口，总体可以达到 3.2 Tbit/s 级别的线速处理能力；三台服务器中，其中一台配备 Intel X722 网卡，另外两台配备 Intel X710 网卡，均可达到 10 Gbit/s 级别传输速率。可编程交换机间的环回线使用 100 Gbit/s QSFP（quad small form-factor pluggable）线缆，最高可达到 100 Gbit/s 级别的传输速率。

可编程交换机与三台服务器搭建的物理拓扑如图 9 所示，交换机内部端口之间随机连接了 6 条 QSFP 回环线，分别为端口 1 与端口 3 之间，端口 9 与端口 11 之间，端口 14 与端口 16 之间，端口 15 与端口 19 之间，端口 18 与端口 20 之间，端口 21 与端口 24 之间。另外，交换机的 31 端口通过一分四模块分出 4 条 10 Gbit/s 的线缆，其中 3 条分别接

到 3 个服务器的网卡上，后续会将服务器 A、B 作为数据包收发的节点，模拟数据中心内的端节点，并将服务器 C 作为时间数据汇总分析的节点，模拟数据中心内的中心控制节点 Telemetry Controller。

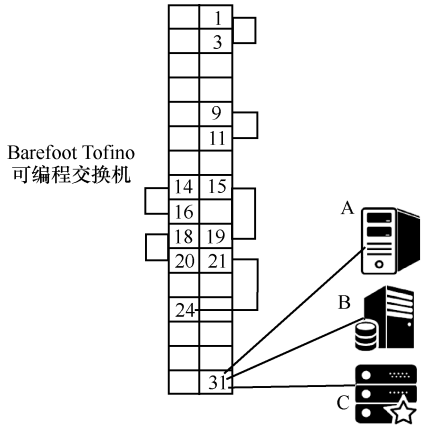


图 9 实验环境物理拓扑

由图 9 可知，可编程交换机内部端口间连接了 6 条回环线，所以该物理交换机可以抽象为 7 台逻辑上的交换机，并且服务器 A、B、C 可以访问任意一个逻辑交换机。为了便于理解，这里根据图 9 所示物理拓扑，画出了实验环境的逻辑拓扑，如图 10 所示。

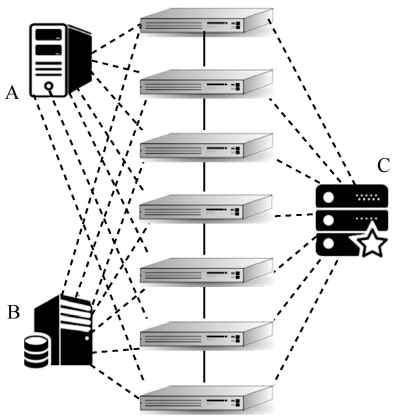


图 10 实验环境逻辑拓扑

如图 10 所示，后续本文进行评估时，会以服务器 A、B 为数据包收发的端节点，负责产生背景流量，并以服务器 C 作为中心控制节点，负责汇总分析最后一跳中间节点发来的时间数据，并根据前述方法计算出任意 2 个中间节点（逻辑上）的时间偏差，与真实时间偏差（0）比较，进行多方面系统评估。

5.2 多跳时间同步效果评估

本文使用图 10 所示的网络拓扑，对相隔多跳的中间节点时间同步效果进行了测试评估。由图 10 可知：1) 即使相隔许多跳（本文最高测试到了 10 跳），中心控制节点测算出的任意 2 个中间节点时间偏差的误差依然在 10 ns 以内，可以为任意 2 个中间节点达成纳秒级的时间同步；2) 使用式(5)这种依次计算各跳的时间偏差值并累加的测算方式（利用中间每一跳的时间戳），达成的时间同步效果要比式(4)这种只利用首尾节点时间戳的测算方式效果更好；3) 使用 LOF 方法进行离群值去除之后，可以进一步改善同步效果，尤其是在降低同步数据的标准差方面。

本文将各个逻辑交换机的端口速率均设为 100 Gbit/s，服务器 A 和 B 之间使用 DPDK（data plane development kit）的 Pktgen 工具互相发送数据包，产生背景流量，数据包发送速率均设为 1 Gbit/s。通过控制平面设置可编程交换机中的流表，可以使数据包经过多跳逻辑上的交换机。另外，当数据包到达最后一跳中间节点时，会向服务器 C 汇报中间路径记录下来的时间戳。

服务器 C 汇总分析收集到的时间数据，对任意 2 个中间节点，取逻辑上的 5 万对请求−回复数据包，分别用式(4)和式(5)进行时间偏差计算，取其平均值作为初步计算出的时间偏差，并与真实偏差（0）对比评估，即将平均值取绝对值，观测其偏离 0 的程度；另外，服务器 C 对 5 万对数据包初步计算出的 5 万个偏差值使用 LOF 方法进行离群值去除，再取平均值与真实偏差（即为 0）进行对比评估，时间同步误差（时间偏差均值的绝对值）和时间偏差标准差分别如图 11 和图 12 所示。

如前所述，使用 5 万对数据包进行计算得到的时间偏差平均值的绝对值即为与真实时间偏差（0）的误差。由图 11 可以看出，即便 2 个中间节点相隔许多跳（本文最高测试到了 10 跳），测算出的时间同步的误差也依然在 10 ns 以内。同时，由图 11 可以看出，中间节点并不是相隔得跳数越多，时间同步精度就会随之下降，比如无论采用哪种计算方式，图 11 中相隔 8 跳的 2 个中间节点达成的同步效果都比相隔 3 跳的中间节点同步效果还要好（计算出的 Offset 均值的绝对值更小，与 0 更接近），猜想这是因为数据包实际往返各条线缆时可能会产生正或负的硬件时间误差，造成请求和回复 2 个

方向的时间不对称性，不过由于时间误差可正可负，因此经过多跳时可能会产生正负误差相抵消的效果，比如图 11 中相隔 7 跳的中间节点，使用式(5)计算可以测算得误差在 1 ns 以内的高精度时间偏差值，该现象就是多条线缆产生的正负时间误差相抵消的一个例子。使用式(5)这种依次计算各跳时间偏差值并累加的测算方式（利用中间每一跳的时间戳），达成的时间同步效果要比式(4)这种只利用首尾节点时间戳的测算方式效果更好，多跳情况下每一跳都可以计算出比式(4)误差更小更精准的时间偏差值。另外，如果先使用 LOF 方法去除离群值再结合式(5)进行时间偏差计算，可以进一步小幅度降低测算时间偏差与真实时间偏差的误差，改善同步效果。

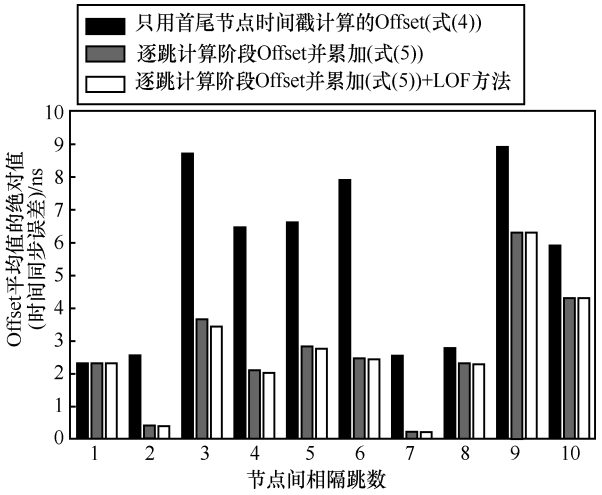


图 11　相隔多跳的节点间时间偏差均值的绝对值（时间同步误差）

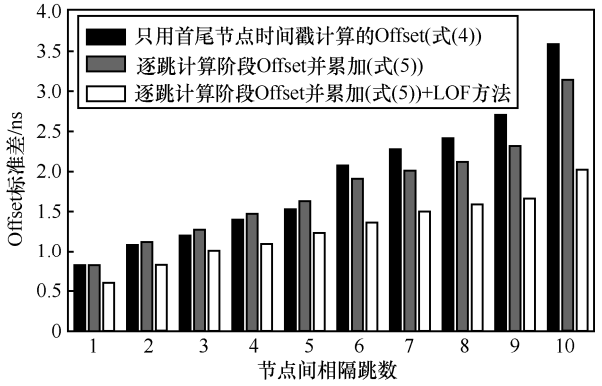


图 12　相隔多跳的节点间时间偏差标准差

由图 12 可以看出，相隔多跳的节点间时间偏差数据的标准差都在 4 ns 以内，时间偏差数据分布比较均匀，没有太大的波动，这意味着用较少的数据取平均即可得到一个准确的平均值。同时也注意到，中间节点相隔得跳数越多，时间偏差数据的标准差越大，相对而言数据分布得就越离散。另外，通过 LOF 方法去除离群值后，时间偏差数据的标准差得到显著降低，数据分布得更加均匀。

5.3　端口额定速率对时间同步效果的影响

5.2 节将各个逻辑交换机端口的额定速率均设为 100 Gbit/s，服务器 A、B 实际数据包发送速率均设为 1 Gbit/s，用以评估多跳时间同步的效果，结果显示均可以达到误差在 10 ns 以内的同步精度，且同步偏差测算值的标准差均在 4 ns 以内，数据分布比较均匀。

下面，以单跳为着眼点（此时式(4)和式(5)是等价的），评估逻辑交换机端口速率对时间同步效果的影响。可以发现，服务器 A、B 实际数据包发送速率仍保持 1 Gbit/s，当逻辑交换机端口速率设为 25 Gbit/s、40 Gbit/s、100 Gbit/s 时，2 个相邻的中间节点时间偏差测算值仍与 5.2 节中的结果保持一致，时间同步误差（Offset 均值的绝对值）均在 3 ns 以内，Offset 标准差均在 2 ns 以内，有着较好的同步效果，具体结果如表 1 所示。

表 1　端口额定速率非 10 Gbit/s 情况下时间同步效果

端口速率/（Gbit·s^{-1}）	Offset 均值/ns	Offset 标准差/ns
25	2.315	0.813
40	2.288	1.412
100	2.343	0.818

但是，当逻辑交换机的端口速率设为 10 Gbit/s 时，初始测算到的时间偏差数据中出现了大量的离群值，累积分布函数（CDF, cumulative distribution function）如图 13 中虚线所示，5 万个 Offset 数据平均值为 38.678 ns（即为与真实时间偏差的误差），标准差为 128.622 ns，时间同步效果较差。

应用 LOF 方法去除离群值后，可以大幅度提升时间同步效果，去除离群值后得到 42 507 个 Offset 数据，其 CDF 如图 13 中实线所示，数据平均值为 2.923 ns（即为与真实时间偏差的误差），标准差为 6.031 ns。可以看出，端口额定速率为 10 Gbit/s 导致的同步效果下降可以通过 LOF 方法得到修复。

5.4　实际数据包发送速率对时间同步效果的影响

5.2 节和 5.3 节将服务器 A、B 发送数据包的速率（背景流量速率）均设为了 1 Gbit/s，本节将测试更高的数据包发送速率会对时间同步效果产生怎样的影响。仍以单跳为着眼点，评估交换机端口

额定速率固定时，不同的数据包发送速率（1 Gbit/s、5 Gbit/s、10 Gbit/s）对时间同步效果的影响如表 2 所示。

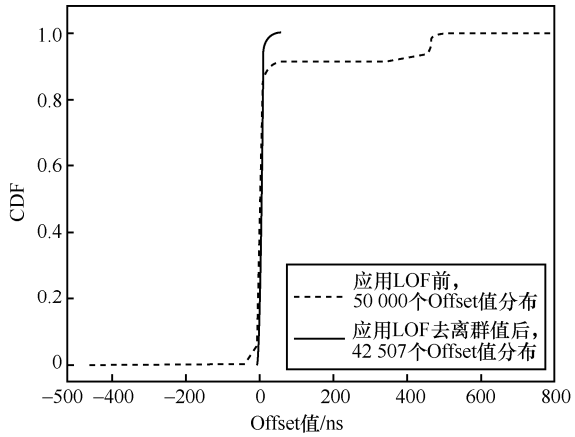


图 13 端口额定速率为 10Gbit/s 时 Offset CDF
（应用 LOF 前和应用 LOF 去离群值后）

表 2 实际数据包发送速率对时间同步效果的影响

端口额定速率−实际发送速率/(Gbit·s^{-1})	用于计算 Offset 的数据包数量	Offset 均值（同步误差）	Offset 标准差
10−1	50 000	**38.678**	**128.622**
10−5	40 800	6.327	**167.035**
10−10	17 248	**16.037**	**137.436**
25−1	50 000	2.315	0.813
25−5	41 072	2.323	0.808
25−10	18 400	2.33	0.793
40−1	50 000	2.288	1.412
40−5	41 202	2.267	4.452
40−10	18 630	2.251	3.951
100−1	50 000	2.343	0.818
100−5	41 498	1.599	1.3
100−10	17 312	2.288	0.798

由表 2 可以发现，当逻辑交换机端口速率设为 25 Gbit/s、40 Gbit/s、100 Gbit/s 时，无论实际发送速率如何增大，即使增大到服务器 A、B 网卡的最大发送速率 10 Gbit/s，已经出现了较明显的丢包现象（尤其在数据包发送速率为 10 Gbit/s 时，丢包导致可用于计算的数据包对不足 2 万对），时间同步的效果也不会有显著改变，同步误差都能控制在 3 ns 以内，Offset 标准差均在 5 ns 以内，仍能保持高精度的时间同步效果。

当逻辑交换机端口速率设为 10 Gbit/s 时，如表 2 中加粗的数据所示，同步误差较大，Offset 标准差远高于正常范围，说明出现了 5.3 节中的异常情况，初始测算到的时间偏差数据中产生了大量的离群值，导致时间同步效果较差。

对表 2 前三行，即端口速率为 10 Gbit/s 的 3 项计算结果（数据包发送速率分别为 1 Gbit/s、5 Gbit/s、10 Gbit/s）应用 LOF 方法去除离群值，结果如图 14 所示。

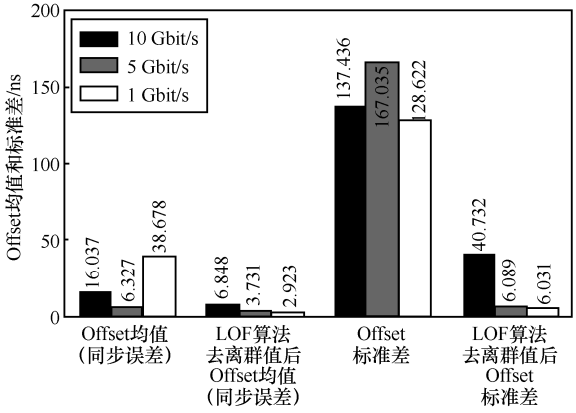


图 14 端口速率为 10 Gbit/s 时，应用 LOF 方法改进效果

由图 14 可知，当端口速率为 10 Gbit/s 时，无论数据包发送速率是 1 Gbit/s、5 Gbit/s 还是 10 Gbit/s，初始计算出的时间同步效果均较差，而使用 LOF 方法去除离群值后，3 种数据包发送速率下，时间同步误差都控制在了 10 ns 以内，标准差也有大幅度的下降，使同步效果得到修复。

通过 5.3 节和本节评估可知，实际数据包发送速率对于时间同步效果不会产生影响，即使发送速率增大到出现了较明显的丢包现象，时间同步效果也依然没有改变。另外，端口额定速率会对时间同步效果有较大的影响，当端口速率设为 25 Gbit/s、40 Gbit/s、100 Gbit/s 时，无论实际数据包发送速率如何，均有高精准的时间同步效果；当端口速率设为 10 Gbit/s 时，无论实际数据包发送速率如何，初始计算的时间偏差数据中总会出现大量的离群值，同步误差较大，Offset 标准差远高于正常范围，时间同步效果较差，不过由于端口速率设为 10 Gbit/s 导致的同步效果下降可以通过 LOF 方法得到修复，去除离群值后仍可以达到高精准的同步效果。

5.5 开销评估

5.5.1 可编程数据平面性能开销

本文设计的基于 Telemetry 架构的数据中心网络高精准时间同步系统主要基于可编程数据平面

和中心控制节点 Telemetry Controller，由于中心控制节点处理能力强大，可以执行较复杂的操作，而可编程交换机因为要保证线速，对于各方面资源的限制较多，容易成为系统瓶颈，因此关于性能开销，主要针对在可编程数据平面上带来的开销进行评估。

首先生成一个对照 P4 程序——将原来 P4 程序时间戳采集汇报部分删除，只保留转发数据包的逻辑，然后使用 P4 Insight 工具对时间采集汇报 P4 程序和对照 P4 程序进行了对比评估，结果显示采集汇报 P4 程序相比对照 P4 程序只多使用了一个可编程交换机的 stage（可编程交换机共有 12 个 stage），具体结果如表 3 所示。

表 3　　原始 P4 程序与对照 P4 程序交换机资源开销对比评估

交换机资源	原始 P4 程序 stage0	原始 P4 程序 stage1	对照 P4 程序 stage0
8 bit Action Slots	0	0	0
16 bit Action Slots	6.25%	6.25%	6.25%
32 bit Action Slots	3.13%	3.13%	3.13%
Gateway	18.75%	0	6.25%
Hash Bit	4.57%	4.33%	4.57%
SRAM	1.25%	1.25%	1.25%
Stats ALU	0	0	0
TCAM	0	0	0
VLIW Instruction	6.25%	6.25%	3.13%
Exact Match Search Bus	12.50%	6.25%	6.25%

由表 3 可以看出，相比对照 P4 程序，原始 P4 程序只在某些资源上有 2~3 倍的开销消耗，并且在每个 stage 消耗的资源最高不超过 20%。相对于 12 个 stage 的可编程交换机资源，本文的 P4 程序只多占用了一个 stage 并且每项资源消耗不超过该 stage 的 20%，这种程度的开销增长是可以接受的。

5.5.2　流量开销

除了在可编程数据平面上带来的计算开销，本文还分析了该基于 Telemetry 架构的高精准时间同步系统带来的额外流量开销，以及流量开销与同步精度间的关系。

5.2 节~5.4 节中的实验均为全采样模式，即可编程交换机会对所有经过的数据包记录时间信息，并在最后一跳中间节点上报到中心控制节点 Telemetry Controller。本节将分析更低的采样率（同时会有更少的额外流量开销）会对同步精度造成什么影响。

以单跳为着眼点，逻辑交换机端口速率设为 10 Gbit/s，服务器 A、B 实际数据包发送速率均设为 1 Gbit/s，服务器 A、B 间的背景流量依然为 5 万对数据包，由此分析了采样率分别为 1:1、100:1、1 000:1、10 000:1 情况下同步精度的变化，如图 15 所示。

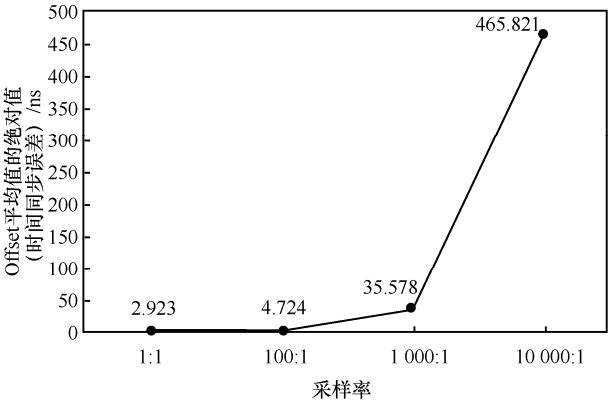


图 15　不同采样率（不同流量开销）时的同步精度

由图 15 可以看出，对背景流量的每个数据包进行时间戳记录及上报的全采样方法可以达到 2.923 ns 的时间同步精度。当采样率改为 100:1，即每 100 个背景流量数据包进行一次时间戳信息记录及上报后，由于时间信息变少，偏差数据带来的影响更大，同步精度降至 4.724 ns，依然在 10 ns 以内；当采样率改为 1 000:1 和 10 000:1 后，由于更易受到随机离群值的影响，时间同步精度有较大幅度的下降，尤其当采样率为 10 000:1 时，同步精度只有 465.821 ns。

下面，综合流量开销和同步精度，从系统实用性的角度继续进行综合分析。在高频通信场景即全采样模式下，可以达到最高的同步精度，但带来的开销也是最大的：若同步的 2 个节点间只相隔一跳，那么每一个正常通信数据包（背景流量）都会用于记录时间数据，并在最后一跳中间节点汇报给中心控制节点，这会带来一倍的额外信令开销；若同步的 2 个节点间隔许多跳（n 跳），由于本来系统就会有 n 跳的背景流量，则最后的汇报操作带来的相对信令开销会较小，这种情况下会增加 $1/n$ 的信令开销。综合来看，虽然全采样模式可以达到最高的同步精度，但是带来的信

令开销也较大，实际应用本文系统时可以综合考虑同步精度与信令开销，如果采用 100:1 的采样率，可以将流量开销降到 1/100，且仍能达到 10 ns 以内的高同步精度。

6 结束语

本文针对数据中心网络高精准时间同步的需求，分析了传统的局部时间同步架构存在的与集中式管理脱节、需要额外交互开销、时间数据较少易受离群值影响等问题，提出了基于 Telemetry 架构的纳秒级时间同步系统，借助数据中心的背景流量来收集数据包进出节点的时间信息，并在中心控制节点对汇报上来的大量时间信息进行分析处理，达成一种集中式的自动同步，便于后续进行全局性网络管理调度。另外，中心控制节点分析整理时间数据后，会将初步计算的时间偏差数据再应用 LOF 方法，去除其中的离群值，进一步提升时间同步精度，尤其在去除离群值、优化较差情况方面效果优异。

本文在 Barefoot Tofino 可编程交换机上实现了基于 Telemetry 架构的时间同步系统，借助数据平面的可编程性，使用 P4 语言编写了可编程交换机的转发逻辑，要求各节点在转发数据包的同时，向中心节点汇报数据包进出节点的时间信息。通过多方面系统评估发现，通过中心节点对汇报上来的大量时间信息进行分析处理，在多跳之间、不同端口额定速率及实际数据包发送速率下，均可以达到纳秒级的高精准时间同步，对可编程交换机带来的开销也可以接受；并且，任意 2 个节点的时间偏差都可以在中心节点得到，具有全局视野的纳秒级的时间同步效果之后可直接用于全局网络管理优化、细粒度的包级别调度、网络监测、网络快照的同步获取等任务。

未来的工作主要包括：1) 在时间同步系统中加入非可编程交换机，拓展系统的普适性；2) 进行更复杂拓扑和更长光缆长度等条件下的评估与改进。

参考文献：

[1] CORBETT J C, HOCHSCHILD P, HSIEH W, et al. Spanner: Google's globally distributed database[J]. ACM Transactions on Computer Systems, 2013, 31(3): 1-22.

[2] PERRY J, OUSTERHOUT A, BALAKRISHNAN H, et al. Fastpass: a centralized "zero-queue" datacenter network[C]//Proceedings of the 2014 ACM Conference on SIGCOMM. New York: ACM Press, 2014: 307-318.

[3] ZENG H Y, ZHANG S D, YE F, et al. Libra: divide and conquer to verify forwarding tables in huge networks[C]//11th USENIX Symposium on Networked Systems Design and Implementation. Berkeley: USENIX Association, 2014: 87-99.

[4] MILLS D L. Internet time synchronization: the network time protocol[J]. IEEE Transactions on Communications, 1991, 39(10): 1482-1493.

[5] EIDSON J C, FISCHER M, WHITE J. IEEE-1588™ standard for a precision clock synchronization protocol for networked measurement and control systems[C]//Proceedings of the 34th Annual Precise Time and Time Interval Systems and Applications Meeting. Piscataway: IEEE Press, 2002: 243-254.

[6] LEE K S, WANG H, SHRIVASTAV V, et al. Globally synchronized time via datacenter networks[C]//Proceedings of the 2016 ACM SIGCOMM Conference. New York: ACM Press, 2016: 454-467.

[7] KANNAN P G, JOSHI R, CHAN M C. Precise time-synchronization in the data-plane using programmable switching ASICs[C]//Proceedings of the 2019 ACM Symposium on SDN Research. New York: ACM Press, 2019: 8-20.

[8] GENG Y L, LIU S Y, YIN Z, et al. Exploiting a natural network effect for scalable, fine-grained clock synchronization[C]//Proceedings of the 15th USENIX Conference on Networked Systems Design and Implementation. Berkeley: USENIX Association, 2018: 81-94.

[9] KIRKPATRICK K. Software-defined networking[J]. Communications of the ACM, 2013, 56(9): 16-19.

[10] TOKUSASHI Y, DANG H T, PEDONE F, et al. The case for in-network computing on demand[C]//Proceedings of the Fourteenth EuroSys Conference. New York: ACM Press, 2019: 1-16.

[11] BOSSHART P, DALY D, GIBB G, et al. P4: programming protocol-independent packet processors[J]. ACM SIGCOMM Computer Communication Review, 2014, 44(3): 87-95.

[12] MURTA C D, JR P R T, MOHAPATRA P. QRPp1-4: characterizing quality of time and topology in a time synchronization network[C]//IEEE Globecom. Piscataway: IEEE Press, 2006: 1-5.

[13] PÁSZTOR A, VEITCH D. PC based precision timing without GPS[C]//Proceedings of the 2002 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems. New York: ACM Press, 2002: 1-10.

[14] LEE K S, WANG H, WEATHERSPOON H. SoNIC: precise realtime software access and control of wired networks[C]//10th USENIX Symposium on Networked Systems Design and Implementation. Berkeley: USENIX Association, 2013: 213-225.

[15] DAVIS M, VILLAIN B, RIDOUX J, et al. An IEEE-1588 compatible RADclock[C]//2012 IEEE International Symposium on Precision Clock Synchronization for Measurement, Control and Communication Proceedings. Piscataway: IEEE Press, 2012: 1-6.

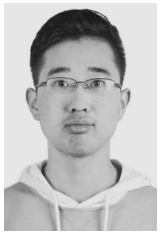
[16] FREEDMAN D A, MARIAN T, LEE J H, et al. Exact temporal characterization of 10 Gbps optical wide-area network[C]//Proceedings of the 10th Annual Conference on Internet Measurement. New York: ACM Press, 2010: 342-355.

[17] LI H. IEEE 1588 time synchronization deployment for mobile backhaul in China Mobile[R]. 2014.

[18] ZARICK R, HAGEN M, BARTOŠ R. Transparent Clocks vs. Enterprise Ethernet switches[C]//2011 IEEE International Symposium on Precision Clock Synchronization for Measurement, Control and Communication. Piscataway: IEEE Press, 2011: 62-68.
[19] WATT S T, ACHANTA S, ABUBAKARI H, et al. Understanding and applying precision time protocol[C]//2015 Saudi Arabia Smart Grid (SASG). Piscataway: IEEE Press, 2015: 1-7.
[20] BREUNIG M M, KRIEGEL H P, NG R T, et al. LOF: identifying density-based local outliers[C]//Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data. New York: ACM Press, 2000: 93-104.
[21] MCKEOWN N, ANDERSON T, BALAKRISHNAN H, et al. OpenFlow[J]. ACM SIGCOMM Computer Communication Review, 2008, 38(2): 69-74.
[22] BOSSHART P, GIBB G, KIM H S, et al. Forwarding metamorphosis[J]. ACM SIGCOMM Computer Communication Review, 2013, 43(4): 99-110.

[作者简介]

张千里（1975− ），男，内蒙古包头人，博士，清华大学副研究员，主要研究方向为下一代互联网体系结构、网络安全等。



张超凡（1998− ），男，山东聊城人，清华大学硕士生，主要研究方向为可编程数据平面、时间同步等。

王继龙（1973− ），男，黑龙江大兴安岭人，博士，清华大学教授、博士生导师，主要研究方向为下一代互联网体系结构、网络测绘等。

唐翔宇（1998− ），男，重庆人，清华大学硕士生，主要研究方向为位置网、室内定位等。

沈钲晨（1994− ），男，浙江余姚人，清华大学硕士生，主要研究方向为位置隐私等。

王会（1977− ），女，河南南阳人，博士，清华大学副教授、博士生导师，主要研究方向为互联网路由、流量工程等。