浙江理工大学学报(自然科学版),第 33 卷,第 4 期,2015 年 7 月
Journal of Zhejiang Sci-Tech University (Natural Sciences)
Vol. 33, No. 4, Jul. 2015

文章编号: 1673-3851 (2015) 04-0508-07

基于四层交换的工业实时同步交换技术的研究

杨　明, 赵新龙

(浙江理工大学机械与自动控制学院, 杭州 310018)

摘　要: 网络控制系统中通常存在着大量短帧结构的实时数据,当网络负载增加时,传统的端对端交换技术会产生明显的排队时延,使得传输效率低、带宽占用高。在四层交换技术基础上提出了适于工业控制的实时同步交换技术,在非周期数据传输阶段,对交换机各端口通信任务的数量和执行时间分别进行优化与调度,在周期数据传输阶段,执行数据包的接收、拆分、重组、转发四个步骤,实现了数据包的同步接收和同步转发。提出的方法减少了数据包在交换机中的排队时延,提高了数据传输的同步性,数据传输效率高,网络带宽占用小。仿真实验验证了该方法的可行性和优越性。

关键词: 交换式网络; 工业以太网; 实时; 网络控制系统; 调度

中图分类号: TP23　　**文献标志码**: A

0　引　言

以太网具有传输速率高、协议标准统一、成本低等优势,在工业网络控制系统中得到了越来越高的关注,以太网的实时性研究已成为热点[1-2]。传统端对端(end-to-end)交换式以太网相比于共享总线式以太网,虽然数据传输的实时性得到了改善,但由于各端口存在缓存队列,因而依旧会造成数据传输的不确定性,严重时则会造成缓存溢出并导致数据丢包[3-4]。

交换机缓存分为输出队列 OQ(output queue)模式和输入队列 IQ(input queue)模式,为保证数据传输的实时性通常需要采取一定的拥塞管理调度算法。OQ 模式主要有经典的 PQ、WFQ、RR、WRR、SRR 等算法,但该类调度算法并不能完全保证各数据传输的实时性,甚至存在缓存溢出的风险[5]。IQ 模式经典的调度算法为 iSLIP[6],与 OQ 模式相比,此类算法提高了数据传输的实时性,减少了数据丢包的可能[7]。但无论是 OQ 模式还是 IQ 模式,实时数据的接收与任务调度都需要交换机同步参与,因而会产生一定的处理时延,其时延量与通信数据量和交换机处理性能直接相关。在工业控制中,各节点的通信数据通常为短帧结构,数量多、分布广,而且网络数据流向具有汇总性,因而会导致产生很多零碎的小数据包。又由于传统端对端交换技术工作于异步模式下,不能同步处理缓存队列中的所有数据包,必须根据调度算法来处理各数据包的转发。综上这几点,传统端对端交换技术数据排队时延长、数据传输效率低、网络带宽占用高,所以不能很好地适应工业环境。

本文在四层交换技术的基础上提出了适用于工业控制的实时同步交换技术。该技术的优化与调度过程完全在非周期阶段进行,从而在一定程度上减少了周期阶段实时数据的传输时延。通过 TCP/UDP 端口来映射不同的交换机 IO 端口,从而使得一个数据包中可以包含多个目的端口的数据内容,这些数据在交换机中按照端口号进行拆分和重组之后再进行转发,将原本零碎的数据包进行了合并,可以提高数据传输效率,减小数据包头信息传输所占用的网络带宽以及排队时延,数据传输的同步性也得到了提升。

收稿日期: 2014－10－16

基金项目: 国家自然科学基金项目(61273184)

作者简介: 杨　明(1987－),男,山东淄博人,硕士研究生,研究方向为工业以太网的实时应用。

通信作者: 赵新龙,E-mail:zhaoxinlong@zstu.edu.cn

1　同步交换技术原理

1.1　同步交换原理与端口映射

为了实现数据包可以同时包含多个网络端口的数据信息,要求交换机除了可以对数据包进行存储和转发功能外,还要根据数据内容进行拆分与重组,对数据包中的内容进行区分与识别就需要利用四层交换技术。将 TCP/UDP 端口号映射到交换机不同的 IO 端口,而各 IO 端口又连接至不同的节点设备,这样便形成了 TCP/UDP 端口号与节点设备 IP 地址之间的映射关系,于是可以根据 TCP/UDP 端口号来区分发送至不同节点设备的数据信息,如图 1 所示。

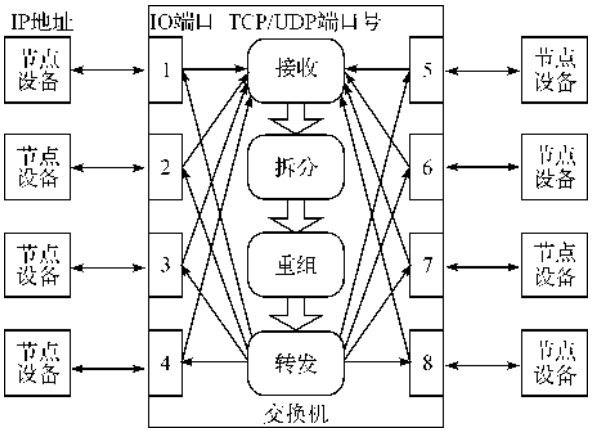

图 1　同步交换原理与端口映射

各节点设备通过 IO 端口将各自的数据包同时发送至交换机内部,交换机将接收到的数据包根据 TCP/UDP 端口号进行拆分,然后将相同端口号的数据重组为新的数据包,最后根据 TCP/UDP 端口号的映射关系向各节点设备同时转发数据。将数据转发过程分为了“接收-拆分-重组-转发”四个步骤,将端对端交换中的排队过程转化为了同步交换过程。

1.2　交换矩阵

传统交换结构通常分为交叉矩阵式、高速总线式、共享内存式等,这些都是针对端对端的交换模式,其作用是建立起一条连接源端口与目的端口之间的连线。而本文的同步交换技术是一种多输入多输出技术,它建立的是数据包内容的拆分与重组之间的转换关系,因此可以在输入端口与输出端口之间建立一个交换矩阵,用来存放不同端口的数据,交换矩阵原理如图 2 所示。

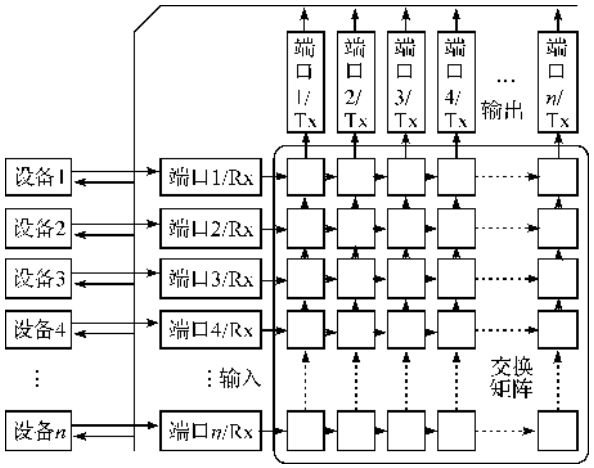

图 2　交换矩阵原理

图 2 中交换矩阵的每一行为来自外部节点设备(源端口)的输入数据包,每个进入交换机内部的数据包通过区分 TCP/UDP 端口号来找到对应的输出端口(目的端口)的位置,并将数据缓存于二者的交点处,于是每个输入的数据包会根据自己的目的端口而被拆分成若干的数据块,此过程即为“拆分”。当所有源端口的数据包进入交换机存储完毕后,交换矩阵的每一列即为各转发端口的数据,加上头信息便形成了转发数据包,即实现了数据包的“重组”。通过“拆分-重组”的两个步骤,便完成了一次数据包的转发过程。通过上述过程可知,数据包何时转发取决于各源端口中最迟完成“拆分”处理的那个数据包。可见,为了尽快达到转发条件,这就要求各数据包应能同时到达交换机,这样才能保证数据包可以实现同步处理和同步转发,节点间不会因排队等待而产生时延,因此各节点的转发过程是同步进行的。

1.3　封装格式

考虑到工业控制对数据传输实时性的要求,由于 UDP 的封装格式比 TCP 占用更小的网络开销,因此 UDP 报文适于控制信息和状态信息等实时数据的传输,而 TCP 报文因其可靠性的传输,因而适合一些如配置信息等非实时报文的传输。但不论采用何种报文传输,根据 1.2 节所述的同步转换原理,各节点设备的发送数据包应包含至各端口的报文分组,而转发数据包只需针对本端口的报文即可,因此数据包的封装格式如图 3 所示。

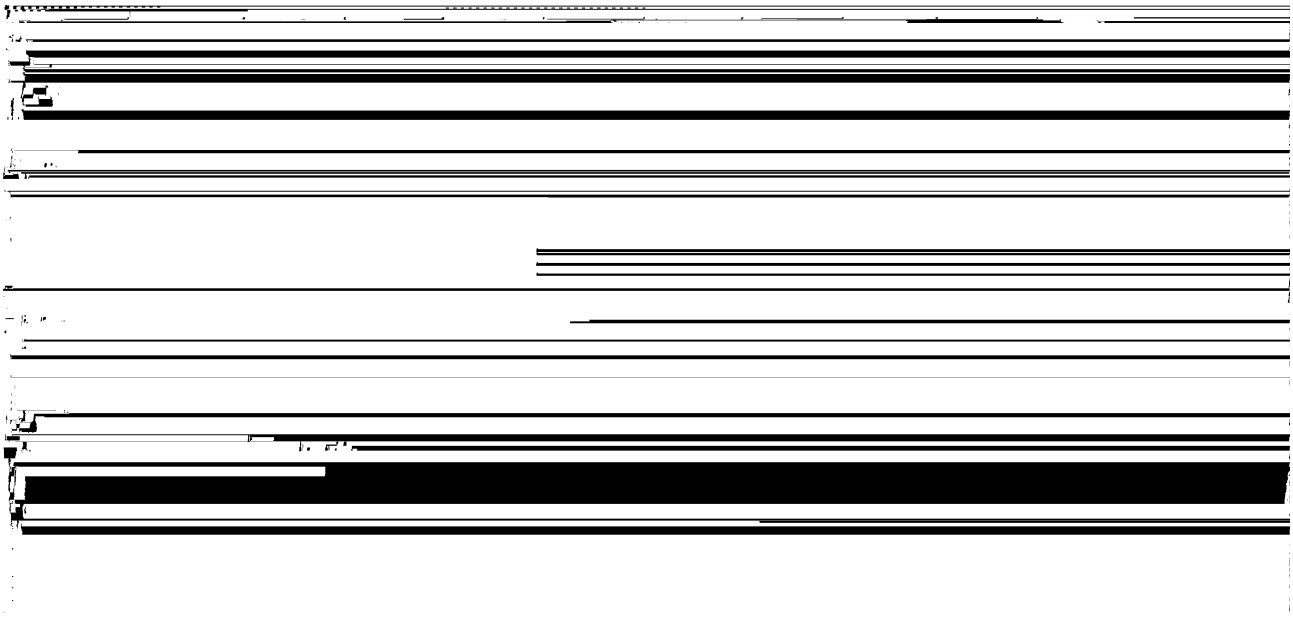

图 3 发送帧与转发帧的格式及其对应关系

2 数据包的优化与调度算法

2.1 数据包传输时延分析

通过第 1 节所述的同步交换原理,实现同步交换有两个关键,一是各节点设备所发送的数据包应可以同时到达交换机,二是交换机所生成的新数据包可以同时转发。每个数据包都要经历“生成、发送、交换和转发”四个阶段,假设每个数据包的交换处理时延相同,则据此建立各节点数据包发送与转发时序图[8-9],如图 4 所示。

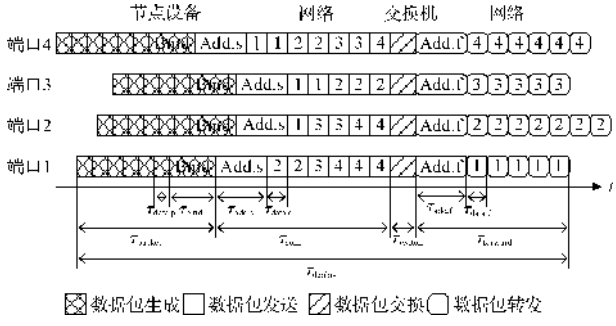

图 4 数据包发送与转发时序图

设某端口待发送的数据量为 $h_{\text{data. s}}$,每个数据的生成时延为 $\tau_{\text{data. p}}$,数据包的装帧时延为 τ_{bind},发送时延为 $\tau_{\text{data. s}}$,数据包帧头、帧尾、前导码等附加数据传输中产生的时延、线路传输时延和时钟裕量[10]等总合为 $\tau_{\text{add. s}}$,数据包在交换机内部的处理时延均为 τ_{switch},交换处理完毕后该端口的转发数据量为 $h_{\text{data. f}}$,单个数据的转发时延为 $\tau_{\text{data. f}}$,附加部分时延为 $\tau_{\text{add. f}}$,于是可求得数据包的生成时间 τ_{packet},发送时延 τ_{send},转发时延 τ_{forward},总时延 τ_{delay}分别为:

$$\tau_{\text{packet}}=h_{\text{data. s}}\tau_{\text{data. p}}+\tau_{\text{bind}} \tag{1}$$

$$\tau_{\text{send}}=h_{\text{data. s}}\tau_{\text{data. s}}+\tau_{\text{add. s}} \tag{2}$$

$$\tau_{\text{forward}}=h_{\text{data. f}}\tau_{\text{data. f}}+\tau_{\text{add. f}} \tag{3}$$

$$\tau_{\text{delay}}=\tau_{\text{packet}}+\tau_{\text{send}}+\tau_{\text{switch}}+\tau_{\text{forward}} \tag{4}$$

2.2 数据包信号数量优化算法

由于工业控制中对数据的传输时延是有一定限制要求的,设实时指标为 J,数据传输的总时延应小于 J。在理想情况下,各端口数据的发送量和转发量应相同,这样可以既保证实时性,又具有最大的网络传输效率,求出理想情况下的预估发送数据量 $h_{\text{data. est}}$为:

$$h_{\text{data. est}}=\frac{J-\tau_{\text{bind}}-\tau_{\text{add. s}}-\tau_{\text{switch}}-\tau_{\text{add. f}}}{\tau_{\text{data. p}}+\tau_{\text{data. s}}+\tau_{\text{data. f}}} \tag{5}$$

设在发送数据包中每个数据帧大小为 $\sigma_{\text{data. s}}$,帧头帧尾等附加部分大小为 $\sigma_{\text{add. s}}$,于是可以求出每个数据包满足最小 MAC 帧和最大 MAC 帧大小要求的数据量 $h_{\text{min. s}}$和 $h_{\text{max. s}}$为:

$$h_{\text{min. s}}=\text{CEIL}[(64-\sigma_{\text{add. s}})/\sigma_{\text{data. s}}] \tag{6}$$

$$h_{\text{max. s}}=\text{FLOOR}[(1518-\sigma_{\text{add. s}})/\sigma_{\text{data. s}}] \tag{7}$$

式(6)-(7)中:运算符 CEIL[-]表示小数进位求整,运算符 FLOOR[-]表示省略小数求整,下同。

则每次发送的数据包中数据个数 $h_{\text{data. s}}$至少符合条件 $h_{\text{min. s}}\leqslant h_{\text{data. s}}\leqslant h_{\text{max. s}}$。若 $h_{\text{data. est}}>h_{\text{max. s}}$,则取 $h_{\text{data. est}}=h_{\text{max. s}}$。同理可得到转发数据包的最小数据量 $h_{\text{min. f}}$和最大数据量 $h_{\text{max. f}}$:

$$h_{\text{min. f}}=\text{CEIL}[(64-\sigma_{\text{add. f}})/\sigma_{\text{data. f}}] \tag{8}$$

$$h_{\text{max. f}}=\text{FLOOR}[(1518-\sigma_{\text{add. f}})/\sigma_{\text{data. f}}] \tag{9}$$

在工业控制中包含大量的周期性实时数据信息,是工业数据通信的主要组成部分,这些数据的发

送时刻和数量是可知的,对于突发的非周期性实时数据,其数据量往往不会特别大,因而不会对周期性实时数据的传输产生较大的影响。

在数据传输周期时刻 t_{period} 到来之前的非周期数据传输阶段,各节点设备统计即将进行传输的数据量并发送 TCP 报文给交换机,交换机在接收到所有节点设备发来的报文后便可生成本传输时刻的初始转发次序表,此表用于调度安排各节点设备每次执行哪个转发端口的任务。转发次序表可以通过类似于循环调度 RR(round robin)的方式建立。从节点设备所对应端口的下一个端口号开始,依次循环查找转发端口是否有数据要传输,有则记录一个,无则查找下一个,直到本发送端口所有转发数量都记录完毕为止。

但初始转发表并不是最优方案,因为某个转发端口的数据量有可能会超出实时指标 J 或以太网最大帧 h_{max_f} 的限制,因此还需要对转发次序表进行整体的优化,并调整某些不合理的转发次序,使得每次传输的数据包均符合 J 和 h_{max_f} 的要求,优化算法如图 5 所示。

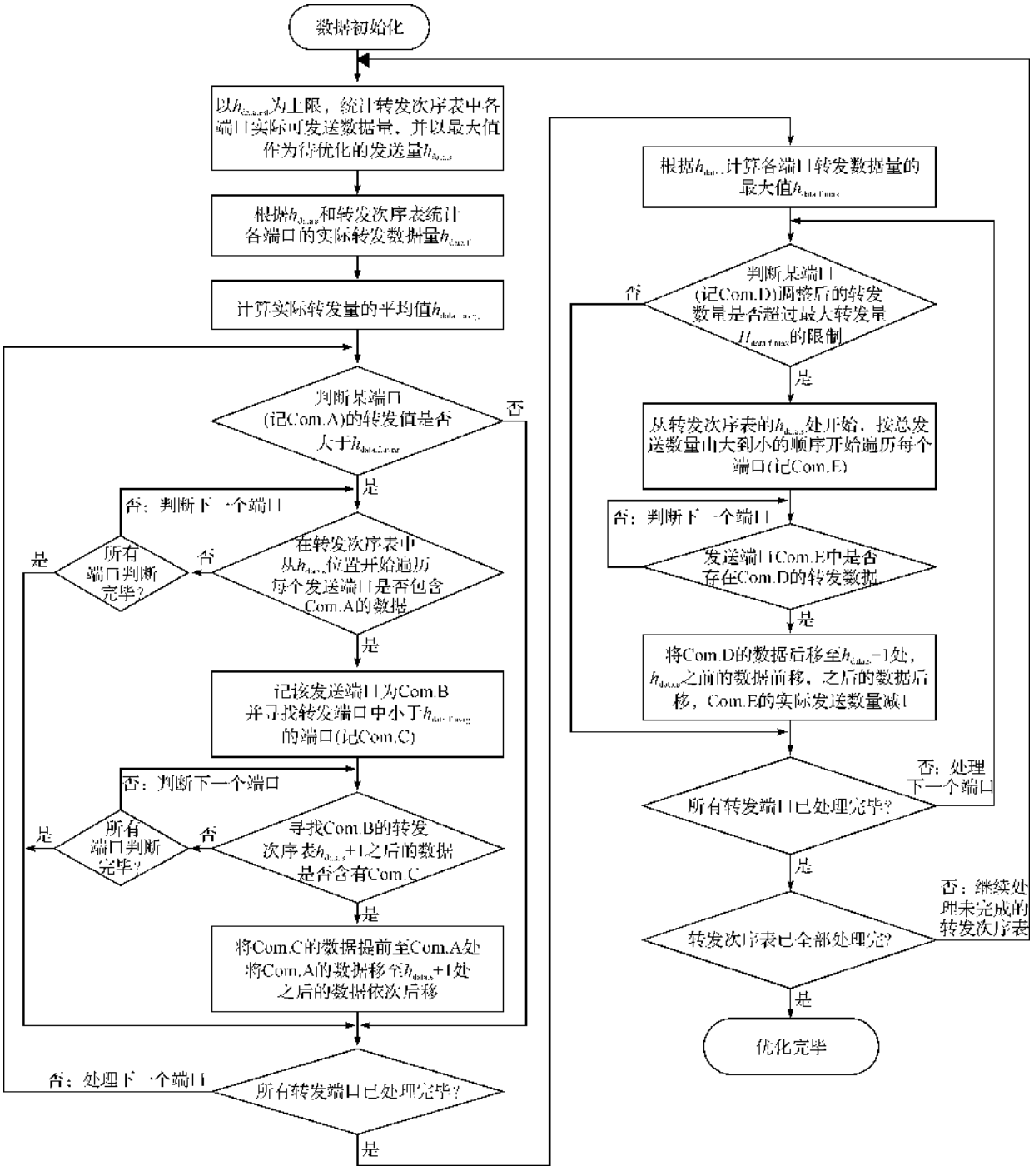

图 5　转发次序表优化算法流程

2.3 数据包起始时刻调度算法

优化完后便可以根据该转发次序表得到一系列的数据信息，包括各源端设备的数据包发送次数，每次发送的数据量 $h_{\text{data\_s}}$，每次转发的数据量 $h_{\text{data\_f}}$。通过这些数据结合式(1)－(4)便可得到数据包在各传输阶段的时延数据。由于要求各源端设备每次发送的数据包必须同时到达交换机，依此来推算每次数据包的起始时刻，其中首次数据包起始时刻为(上标 1 表示第 1 次传输)：

$$t^1_{\text{packet}}=t_{\text{period}}+\tau^1_{\max}-\tau^1_{\text{packet}}-\tau^1_{\text{send}} \tag{10}$$

其中：t_{period} 为本次数据包传输的时刻，$\tau^1_{\max}$ 为各源端设备第 1 次数据传输中的数据包生成时延和传输时延之和的最大值(上角标 k 为端口号)：

$$\begin{aligned}&\tau^1_{\max}=\text{MAX}(\tau^{1,k}_{\text{packet}}+\tau^{1,k}_{\text{send}})\\&k=1,2,\cdots,n\end{aligned} \tag{11}$$

当需要发送两次或两次以上的数据包时，那么起始时刻则需要根据各阶段时延值的不同而进行推算：

暂记 $\tau^i_{\max}=\text{MAX}(\tau_{\text{switch}}+\tau^{i-1}_{\text{forward}})$，$i$ 表示第 i 次传输，然后从第 1 个交换机的 IO 端口开始，如果 $\tau^i_{\text{packet}}\leqslant\tau^{i-1}_{\text{send}}$ 且 $\tau^i_{\text{send}}>\tau^i_{\max}$，则 $\tau^i_{\max}=\tau^i_{\text{send}}$；如果 $\tau^i_{\text{packet}}>\tau^{i-1}_{\text{send}}$ 且 $\tau^i_{\text{packet}}+\tau^i_{\text{send}}-\tau^{i-1}_{\text{send}}>\tau^i_{\max}$，则 $\tau^i_{\max}=\tau^i_{\text{packet}}+\tau^i_{\text{send}}-\tau^{i-1}_{\text{send}}$；否则 $\tau^i_{\max}$ 保持不变，然后查询下一个端口。当遍历完所有端口后，将 $\tau^i_{\max}$ 代入式(12)即可得出数据包的起始时刻：

$$t^i_{\text{packet}}=t^{i-1}_{\text{packet}}+\tau^{i-1}_{\text{packet}}+\tau^{i-1}_{\text{send}}+\tau^i_{\max}-\tau^i_{\text{packet}}-\tau^i_{\text{send}} \tag{12}$$

按照上述方法对剩余的数据进行不断的迭代运算，便可以得到每轮发送的数据包起始时刻数据，并生成一张调度时刻表，内容包含数据包的起始时刻、每个数据包执行的任务数量以及执行哪些转发端口的任务。交换机将该调度时刻表通过 TCP 报文发送给各端口的节点设备，然后各节点设备按照此调度时刻表执行各自的内部任务，生成数据并打包发送。按此调度时刻表即可保证每个数据包可以同时到达交换机进行同步数据交换，传输时延也可符合实时指标 J 的要求。

3 结果与讨论

3.1 仿真环境

本节通过 Matlab 环境下的 Truetime2.0 工具箱[11-12]建立两个 32 端口交换机的仿真环境，分别采用本文提出的实时同步交换技术和传统的端对端交换技术。从数据平均传输时延、电机转速同步误差、传输效率、各端口占用网络带宽的平均值四个方面展开对比与分析。

两种交换技术均使用以下相同的设定参数，任务执行周期为 6 种，分别是 10、50、100、150、200 ms 和 300 ms。各源端口的任务转发表由 Matlab 随机生成，经统计，32 个端口在各周期下传输的总数据量分别为 517、479、489、501、526、505 个，每个数据长度为 16Byte。在每个仿真环境中建立两个电机转速控制系统，其传递函数为 $G=\frac{1\,000}{s^2+s}$。控制器位于节点 2 中，采用 PID 控制，每 0.5s 改变一次阶跃信号。转速采样分别来自节点 1 和节点 8，每 10 ms 进行一次采样并通过交换机发送给节点 2。其余任务数据只进行传输而不参与控制，即模仿网络拥塞环境。实时指标 J 均设为 1 ms，在上述条件下分别在 10、100 Mbps 和 1 000 Mbps 的网络带宽下进行对比测试，其中 10 Mbps 和 100 Mbps 下的 MAC 最小帧为 64Byte，而在 1 000 Mbps 下为 640Byte。最后在 100 Mbps 网络带宽下进行 10 倍数据量的对比测试，即各周期下传输的总数据量分别为 5 170、4 790、4 890、5 010、5 260、5 050 个。

3.2 结果分析

图 6－图 9 为传统端对端交换技术(虚线)和本文实时同步交换技术(实线)的各项仿真数据的对比结果。图 6 为所有数据平均传输时延比较。从图 6 中可以看到，在 10 Mbps(图 6a)和 100 Mbp(图 6b)传输速率下，在网络轻载时两种交换技术的时延值相差不大。但当数据量增加时(图 6c)以及在 1 000 Mbps 的带宽下(图 6d)，由于数据包可以合并更多的数据量，数据包总数明显的减少，数据包的排队时延也就随之减小，从而本文的同步交换技术在时延方面取得了明显的优势。

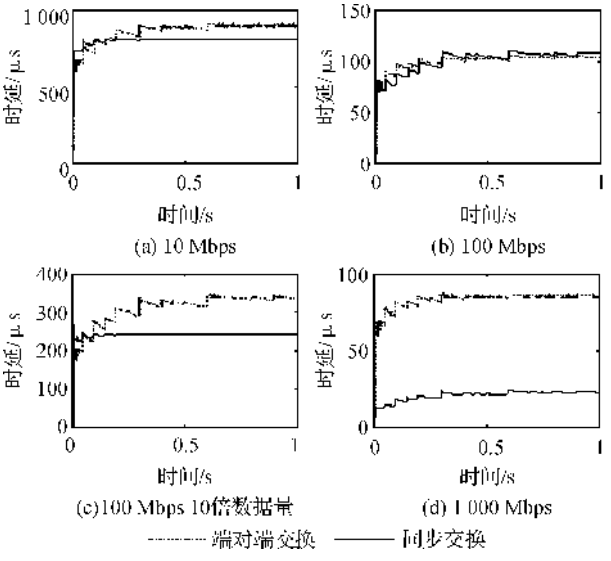

图 6 各数据平均传输时延

图 7 为电机转速同步误差比较。由于数据传输时延的不同,当节点 2 的控制器中同时改变两个电机的给定值时,节点 1 和节点 8 所采样的转速信号可能不会同时反馈至节点 2 中,因而在进行阶跃响应时会造成一定的转速同步误差。从图 7 中可以看出,在各种网络测试环境下,本文的同步交换技术(实线)的转速同步误差明显优于端对端交换技术(虚线),原因在于数据包可以同步的接收与转发,各信号具有相近的传输时延。

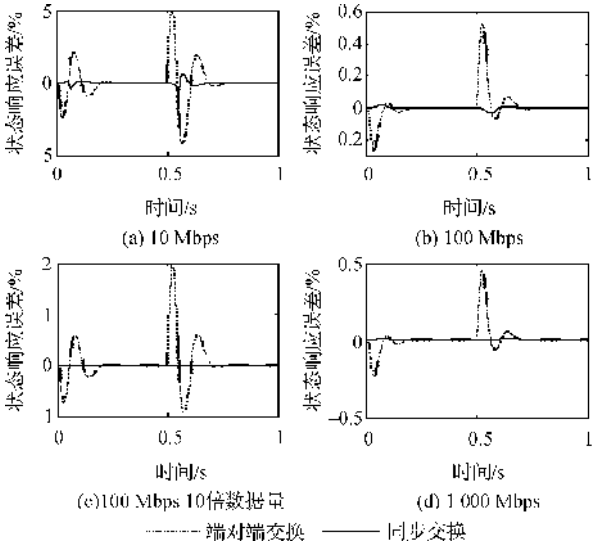

图 7　转速同步误差

图 8 为传输效率的比较,即有效数据占总传输数据的比率。在网络重载时(图 8c),由于两种交换技术都受 MAC 帧大小的限制,两者的传输效率相差不大。而在网络轻载时(图 8a、b、d)本文的同步交换技术在传输效率方面的优势则十分明显,说明本文方法的确起到了合并数据包,提高传输效率的作用。

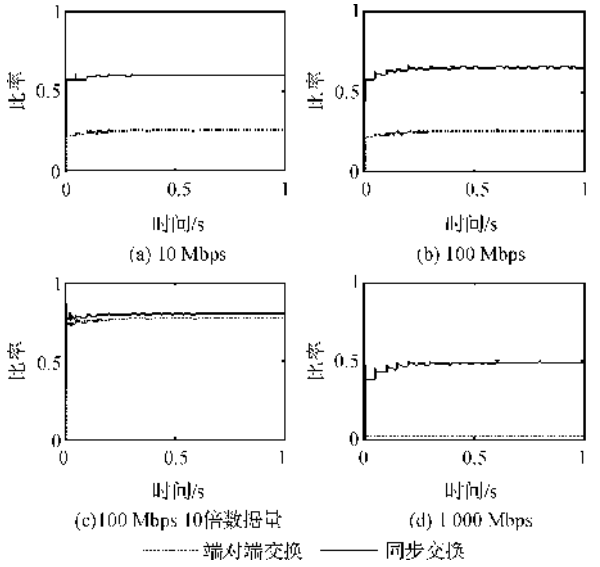

图 8　传输效率

图 9 为交换机各端口占用网络带宽的平均值比较。图中可以看出,在网络重载时(图 9c),两种交换技术由于受到 MAC 帧大小的限制,所以生成的数据包数量几乎相同,从而带宽占用量也就相差不大。但是在网络轻载时(图 8a、b、d),由于本文的同步交换技术进行了数据合,减少了数据包的数量,故网络带宽的占用情况也就得到了明显的降低。

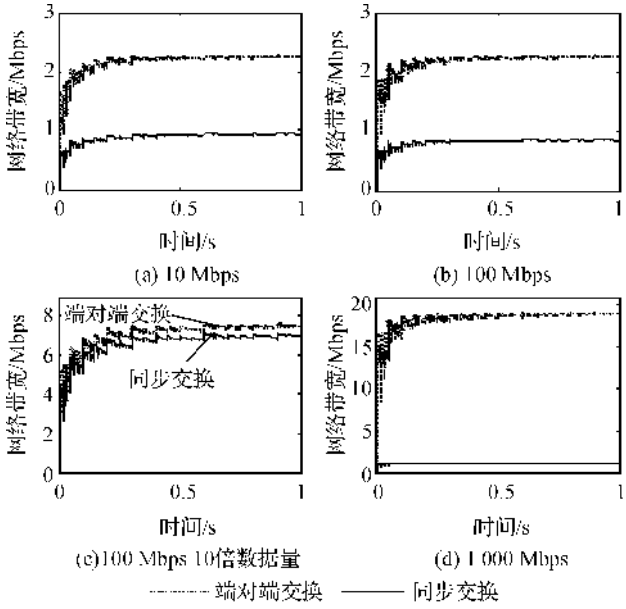

图 9　各端口占用网络带宽的平均值

综上所述,在 10 Mbps 和 100 Mbps 的带宽下,虽然两种交换技术的时延相差不大,但是对电机的同步控制、数据传输效率以及实际所占带宽,本文的同步交换技术则有明显的改善。而在 10 倍数据量下,二者的数据传输效率和实际所占带宽均被 MAC 帧的大小所限制,故区分度不大,但是在时延和同步控制效果上本文的同步交换技术则可以显现其优势所在。在 1 000 Mbps 的带宽下,由于最小 MAC 帧的要求从 64Byte 上升到了 640Byte,在数据量较少时,端对端交换技术会造成对 MAC 帧的利用率严重不足,而采用同步交换技术的设计原则是通过合并数据来解决这个问题,因而在各项指标上都有十分明显的改善效果。

4　结　论

本文所提出的实时同步交换技术基于四层交换技术,利用 TCP/UDP 端口号将传统端对端交换模式下所产生的众多碎小的数据包合并为一个数据包进行传输,在交换机内部再根据 TCP/UDP 端口号进行数据的拆分、合并与转发。其优化与调度过程不会占用周期数据发送阶段,而且可以对输入交换机的数据

流量进行有效的控制，从而使数据包可以得到更快的处理与转发。与传统端对端交换技术相比，本文方法具有传输时延低、数据同步性好、传输效率高、带宽占用小等优点，适合工业环境下的实时通信。

参考文献：

[1] Felser M. Real-time ethernet—industry prospective[J]. Proceedings of the IEEE, 2005, 93(6): 1118-1129.

[2] Gupta R A, Chow M Y. Networked control system: overview and research trends[J]. IEEE Transactions on Industrial Electronics, 2010, 57(7): 2527-2535.

[3] Fan C, Tao X, Ke T W. Real-time performance evaluation of line topology switched ethernet[J]. International Journal of Automation and Computing, 2008, 5(4): 376-380.

[4] 陈本源. 基于交换式以太网的实时工业通信研究[J]. 科技创新导报, 2013, 26: 1-4.

[5] 佟为明，赵 晶. 交换式工业以太网优先级调度机制的研究[J]. 仪器仪表学报, 2007, 28(12): 2197-2201.

[6] McKeown N W. Scheduling algorithms for input-queued cell switches[D]. Berkeley: University of California, 1992.

[7] 陈本源，周祖德. 面向控制网络的交换机调度机制研究[J]. 计算机工程与应用, 2012, 48(34): 33-39.

[8] Li M. Delay analysis of networked control systems based on 100 M switched ethernet[J]. The Scientific World Journal, 2014, 2014: 751491.

[9] 魏 玲，薛定宇，鄂大志，等. 网络控制系统中网络时延与建模分析[J]. 仪器仪表学报, 2008, 29(6): 1323-1327.

[10] 鲁 立，冯冬芹，褚 健，等. 传输时延和时钟同步对以太网控制系统的影响[J]. 控制理论与应用, 2010, 27(6): 793-798.

[11] Cervin A, Henriksson D, Lincoln B, et al. How does control timing affect performance? Analysis and simulation of timing using jitterbug and truetime[J]. IEEE Control Systems Magazine, 2003, 23(3): 16-30.

[12] Peng D G, Zhang H, Lin J J, et al. Simulation research for networked cascade control system based on TrueTime[C]//Proceedings of the 8th World Congress on Intelligent Control and Automation (WCICA), 2011: 485-488.

Research of Industrial Real-time Synchronous Switching Technology Based on 4-layer Switch

YANG Ming, *ZHAO Xin-long*

(School of Mechanical Engineering and Automation, Zhejiang Sci-Tech University, Hangzhou 310018, China)

Abstract: In the network control system, there are a lot of real-time data with short frame structure. The traditional end-to-end switching technology will generate obvious queuing delay when network load increases. Thus, it also has low transmission efficiency and high bandwidth utilization. This paper presents real-time synchronous switching technology for the network control system based on 4-layer switch network technology. In the aperiodic data transfer phase, the number of communication tasks at each port of switch machine and the execution time are optimized and scheduled, respectively. In the periodic data transfer phase, packets are received, split, reorganized and forwarded synchronously. Synchronous reception and forwarding of packets are realized. The method reduces queuing delay of packets in switch machine and improves data transmission synchronism, with high data transmission efficiency and lower bandwidth utilization. Simulation experiment verifies feasibility and superiority of this method.

Key words: switch network; industrial ethernet; real-time; network control system; scheduling

(责任编辑：康 锋)