doi:10.11887/j.cn.201801012 http://journal.nudt.edu.cn

SpaceWire网络混合路由机制设计*

姚 睿[1],羊宇中[1],吴 军[2]
(1. 南京航空航天大学 自动化学院,江苏 南京 211106;2. 北京控制工程研究所,北京 100190)

摘 要:针对星上系统总线多元性导致的星载网络接口和协议不能标准化的发展瓶颈,基于SpaceWire总线协议,通过将静态路由(时间触发)与动态路由(事件触发)机制结合,实现了控制数据和载荷数据共用网络。静态路由完全遵循SpaceWire-D协议,在保证确定性传输的同时,通过启发式调度算法首次实现了多时间窗并行调度,并提出利用最大公约数法设计时间窗,以提高网络吞吐量;动态路由通过对随机事件和载荷数据分配优先级,实现传输路径冲突时对紧急任务的优先处理。在OPENT中搭建网络系统仿真模型,对所提出的路由机制进行了仿真。实验结果表明,静态路由时段网络吞吐量较现有调度算法有明显提高,动态路由实现了紧急事件优先传输。

关键词:SpaceWire-D;静态路由;动态路由;启发式调度算法;时间窗
中图分类号:510.3030 **文献标志码**:A **文章编号**:1001-2486(2018)01-078-08

Design of hybrid routing mechanism for SpaceWire networks

YAO Rui[1], *YANG Yuzhong*[1], *WU Jun*[2]
(1. College of Automation and Engineering, Nanjing University of Aeronautics and Astronautics, Nanjing 211106, China;
2. Beijing Institute of Control Engineering, Beijing 100190, China)

Abstract: Aimed at the bottleneck of the non-uniform interface and protocol of spaceborne network caused by multiple bus standards, the time-triggered static routing and event-triggered dynamical routing based on the SpaceWire bus protocol were combined to make control data and payload data share the same network. The static routing mechanism was fully abided by the SpaceWire-D protocol to ensure deterministic data delivery, in which the heuristic scheduling algorithm was adopted to realize the multi-slot schedule for the first time, and the time-slot was designed by using the greatest common divisor to improve the throughput. And the dynamical routing mechanism can insure that the critical random event be processed by allocating priority levels to random event and payload data preferentially when the transmission routes conflict with each other. In addition, a simulation model for the network system was set up in OPNET to evaluate the proposed routing mechanism. Results show that the throughput of the network is improved significantly during static routing time as compared with the existing scheduling algorithm, and different transactions can be processed according to their priority during the dynamical routing time as expected.

Key words: SpaceWire-D; static routing; dynamical routing; heuristic scheduling algorithm; time-slot

随着星载电子设备数据量的迅猛增长和数据类型的不断扩展,星载数据总线也随着不断发展和完善。SpaceWire总线是由欧空局对IEEE 1355-1995和IEEE1596.3协议进一步改进而来的一种用于星载数据高速传输的总线标准,其点对点、高速、全双工串行总线的通信方式,可满足航天器高速、可升级、低功耗、低成本的通信链路接口要求[1-2]。

然而,由于SpaceWire交换机采用的虫洞路由机制存在阻塞问题[3],导致传输延时不确定,无法保证控制数据的实时传输。因此,星载网络往往分为控制网和数据网两部分[4]。控制网采用1553B或控制器局域网络(Controller Area Network, CAN)等低速总线,实现星上测控信息的交换与共享;数据网采用SpaceWire总线,实现星上大容量载荷数据的交换、处理和传输。这种分层架构可充分发挥多种总线的传输优势,但同时导致星载网络的复杂化,使星载设备的可移植性、继承性大打折扣。因而在SpaceWire网络中同时实现控制数据和载荷数据的传输目前是业界的研究热点。

为了解决网络中可能遇到的阻塞问题,确保在SpaceWire网络中传输控制流时延迟时间的确

* **收稿日期**:2016-10-18
基金项目:国家自然科学基金资助项目(61402226);高容积比智能化微小卫星柔性控制平台技术研究资助项目(D020103);江苏省产学研前瞻性联合研究资助项目(BY2015003-06)
作者简介:姚睿(1974—),女,河南邓州人,副教授,博士,硕士生导师,E-mail: yaorui@nuaa.edu.cn

定性,英国邓迪大学提出了 SpaceWire - D 协议草案[5]。该草案通过传递时间码规定时间窗,并为每个业务发起者提供授权时间窗,在授权时间窗内允许授权业务发起者向任意目标发送业务,以确保控制流和数据流传输的延迟时间的确定性。该草案很好地避免了网络资源冲突,从而解决了网络拥堵问题,保证了延迟时间的确定性。但该草案仅给出了 3 种类型的调度原则,即简单调度、并发调度、多时间窗调度,而未给出具体的调度算法。针对 SpaceWire 调度表设计问题,文献[6]研究了简单调度表的设计方法,通过模拟退火调度算法在冗余网络路径中寻找适当的路径,使得传输延时限制在时间窗内,同时减少了每个周期时间窗的数目。文献[7]研究了并发调度表的设计方法,采用启发式算法依据传输延时相近原则对事务分组,并通过仿真评估了算法性能。然而,文中仅给出了时间窗大小、时间窗数量与传输任务数量的关系,并未给出多时间窗调度的设计方案。

SpaceWire - D 协议保证了网络中所有事务传输延时的确定性,可满足实时性要求较高的周期数据(如航天器的实时姿态、速度、高度等,需要通过调度算法实现确定性传输)的传输需求。然而,除了此类事务外,网络系统中往往还存在实时性要求不高的载荷数据(如视频)以及不可预测的随机传输任务(如任务中某些紧急命令或故障信息的传输)。前者对传输确定性和实时性要求较低,不需要满足严格时限要求;后者为不具有周期性的随机事件,但其传输至关紧要,因为如果某些紧急任务得不到及时处理,可能导致整个任务的失败。所以在网络系统设计中,除了确定性传输的静态路由外,动态路由不可或缺。为保证网络传输延时的确定性,同时兼顾随机事件,文献[8]设计了一种基于时间触发[9]的 SpaceWire 通信协议,并设计调度算法来确保时间触发消息传输延时的确定性,同时提出了为不具备周期性特征的随机数据保留自由竞争时间窗口的思想。然而作为一种上层通信协议,其网络节点的工作方式、数据包格式等均需在原标准基础上进行较大改动,且未给出自由竞争时间窗的具体设计方法和随机事件的传输方法。文献[10]搭建了能够兼容事件触发和时间触发的 SpaceWire 仿真网络模型,并对事件触发和时间触发网络场景下的网络消息传输端到端延时进行了分析和研究。但该方法仅给出了时间触发消息和事件触发消息的延时时间仿真,未给出具体调度算法,也未深入研究网络吞吐量的问题。

为了实现 SpaceWire 网络在统一总线星载网络中的应用,确保实时性要求较高的周期性事务传输延时的确定性,并兼容原有网络协议和网络资源,本文提出了基于 SpaceWire - D,兼顾时间触发和事件触发的静、动结合的高速 SpaceWire 路由机制。将每个调度周期划分为静态路由和动态路由两部分。在静态路由时段,通过启发式算法实现控制数据确定性传输的同时,对传输任务进行多时间窗并发调度,采用最大公约数法设计时间窗的大小和数量,以提高传输效率。在动态路由时段,设计具有优先级的轮询调度方法,实现载荷数据和随机事件的传输。

1　通信协议设计

1.1　网络模型

为了确保动态路由和静态路由共用网络,使星上模块统一使用 SpaceWire 标准接口,所有信息均按照 SpaceWire 通信协议进行传输,本文采用图 1 所示模型描述网络系统。整个系统由静态节点、动态节点、时间码节点、接收节点及路由系统组成。时间码节点用于产生时间码,控制网络同步。路由系统用于实现数据包的寻址转发。接收单元用于描述大容量存储器等,仅作为数据包传输目的,不需要主动发起事务传输。静态节点主要传输实时性要求较高的轨道控制和实时姿态等数据;因数据传输实时性高,且具有周期性,要求每个周期均实时传输,所以称为时间触发消息。时间触发消息通过静态路由调度算法来保证其传输延时的确定性。动态节点主要传输实时性要求较低的星上图像、视频传输业务和随机发生的传输信号,该类数据称作事件触发消息。此类业务实时性要求低,因此放在动态周期传输。另外由于随机信号不仅周期不确定,而且可能要求紧急传输,所以本文在动态路由时段引入优先级设计,使随机信号能实现优先传输。

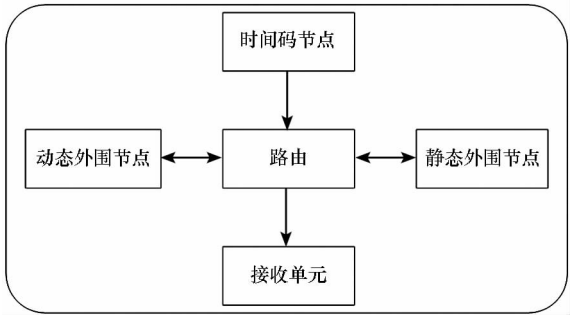

图 1　网络系统模型

Fig. 1　Model of network system

1.2 网络时间基准

时间码可以触发任务发起者查询调度表是否有任务传输,同时可以实现网络时间基准的同步。时间码由时间主节点发送,按照设定的时间大小向全网广播;一个通信周期结束后,时间码复位到零。两个相邻时间码的时间间隔为一个时间窗,节点接收一个时间码伴随着一个新的时间窗的开始,如图 2 所示。时间码的触发机制如图 3 所示,每个发起者(具有发送功能的节点)接收到时间码时,将查询自身是否有事务需要在该时间窗内传输,如有则发送,否则等待下一时间码。

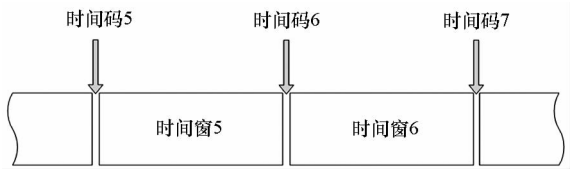

图 2 时间窗与时间码

Fig. 2 Time-slot and time-code

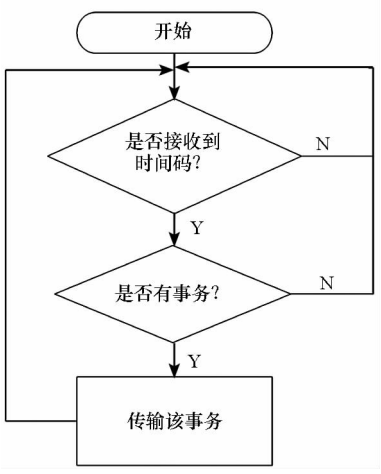

图 3 时间码触发机制

Fig. 3 Trigger mechanism of time-code

1.3 链路延时

SpaceWire 网络链路延时由数据包大小、网络带宽、路由器处理数据的性能、路由跳数决定。SpaceWire 网络为虫洞路由网络,根据虫洞路由的特点,网络中一包数据在无资源冲突情况下的端到端传输时延为:

$$t_D = n(t_p + t_h) + t_p + t_T \tag{1}$$

其中:n 为数据包从源到目的传输经过的路由器个数;t_p 为电信号在物理链路上的传播延时;t_h 为 SpaceWire 路由器处理数据包首部花费的时间,大小由路由器的硬件设计决定;t_T 为数据发送时延,大小等于数据通信量(bit)与数据传输速率 D(bit/s)之比。

同时,对于随机事件传输延时需要计算最大延时,即在链路拥堵情况下的延时。最大阻塞时延表达式为[11]:

$$T_{\text{block}} = \sum [\max d(f_{\text{in}}, next(f_{\text{in}}, l)) + t_h] \tag{2}$$

其中,$d(f_{\text{in}}, next(f_{\text{in}}, l))$是任意相同优先级数据流 f_{in}在下一跳链路的最大传输延时。同时,f 在路由器输出链路 l 上无阻塞传输延时为:

$$T_{\text{unblock}} = d(f, next(f, l)) + t_h \tag{3}$$

最大传输延时即为式(2)与式(3)之和:

$$d(f, l) = \sum [\max d(f_{\text{in}}, next(f_{\text{in}}, l)) + t_h] + d(f, next(f, l)) + t_h \tag{4}$$

2 网络路由机制设计

2.1 总体调度机制

本文静态路由和动态路由对网络的共用通过分时复用实现,每个通信周期包括静态路由时段 T_s 和动态路由时段 T_d,分别由若干个时间窗组成。静态路由时段 T_s 用于传输对实时性要求较高的控制数据流,采用时间触发方式并根据 SpaceWire - D 多时间窗并发调度原则保证其高效确定性传输。动态路由时段 T_d 用于传输对实时性要求不高的数据流和随机事件,采用事件触发方式基于虫洞路由原则进行传输,以充分利用 SpaceWire 网络高速传输的优势。同时,为了确保紧急随机事件得到优先处理,本文对动态路由时段的传输任务设定传输优先级。另外,根据动态时段传输数据的不确定性,为避免未传输完毕的数据包对下一通信周期造成影响,所以在通信周期结束前增加一个数据包清空时段 T_c,被清空的数据包将在下一动态周期重新传输。设计过程中,需保证整个网络通信周期满足约束 $T_a = T_s + T_d + T_c$,如图 4 所示。

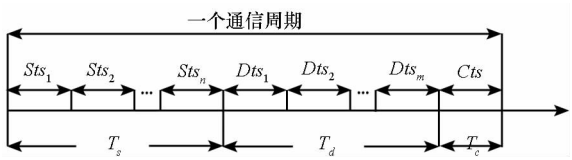

图 4 时间周期分布

Fig. 4 Distribution of cycle time

2.2 静态路由机制

在静态路由时段,为避免因虫洞路由拥堵而导致的延时无法预测,使用 SpaceWire - D 协议来保证传输延时的确定性。静态路由设计时,需要为外围节点提供路由表,供节点接收到时间码时

查询,以判断是否有任务需要传输。SpaceWire - D 的调度原则包括简单调度、并发调度和多时间窗调度。简单调度在每个时间窗只允许一个发起者向任何目的节点传输事务,因此网络效率较低。并发调度在每个时间窗允许多个链路资源不冲突的发起者同时传输事务,因此其链路效率较简单调度高。但是,简单调度和并发调度均需保证事务的传输在单个时间窗完成,因此需要对大数据包进行拆分。多时间窗调度允许一个事务的传输跨越多个时间窗,取消了对数据包大小的限制,更加高效灵活。因此,为提高链路利用效率和传输灵活性,本文采用多时间窗并行调度原则设计静态路由表。在确定性网络中每个任务传输路径确定后,其单独传输延时是确定的,因此本文基于以下思想设计调度表。

1)在保证传输路径不冲突的前提下,将传输延时相近的任务分在同一组并实现任务最大化,使得分组最少,以缩短静态路由传输时间。

2)根据实时调度(Rate-Monotonic, RM)算法[12]思想为每组任务分配授权时间窗,即在静态路由时段,传输延时相对小的分组优先传输。RM 调度算法简单易行,在每个通信周期可实现传输任务等待数量最少。

设计静态调度表时,任务分组方法如算法 1 所示。首先输入所有任务的相关数据,将传输任务按时间延时大小从小到大排序;然后逐个查询,将网络路径不冲突的任务分为一组,循环操作,直至分组完成。

算法 1　启发式调度算法

Alg. 1　Heuristic scheduling algorithm

1. 输入 n 个待调度传输任务、传输通信占用资源以及传输延时;
2. 按照传输延时大小升序排序;
3. 初始化数值 m = 0, j = 0, flag[n] = {0},其中 m 为组任务个数、j 为分组数、flag[n]数组是对已分组任务标记是否已分组,为 1 则已分组,否则未分组;
4. 搜寻没有被分组的任务 n_i, for i = j to i = n - 1 建立新的分组 ARR_j,并标记 flag[i] = 1, m + + ;
5. for i = i + 1 to i = n - 1;寻找与 ARR_j 分组中不冲突且未分组的任务,并标记 flag[i] = 1, m + + ;
6. j + + , if (m < n) 回到 4,
   else 结束分组。

分组后,为每组任务分配时间窗。时间窗的大小由相邻两个时间码间的时间间隔决定。采用简单调度时,时间窗的大小由网络规模大小(路由跳数)、数据包大小及带宽确定。网络规模越大、数据包越大则时间窗越大。根据时间窗的大小和数量可以确定系统的静态周期。采用多时间窗调度时,时间窗的大小可以不受网络规模和数据包大小的限制。但是如果时间窗太小,时间码发送频率太高,任务发起节点需要频繁检测调度表,过多的时间用于检测任务。时间窗设计过大,由于某些任务传输时间远小于时间窗的大小,导致时间窗的资源浪费,网络效率较低。因此时间窗大小的设计是确定性传输效率提高的关键。

为提高网络效率,本文采用最大公约数法设计时间窗大小,取调度算法分组后每组最大任务传输延时的最大公约数作为时间窗大小,以实现时间窗资源的充分利用。

为了说明时间窗设计方法,假设存在 9 个传输任务,通过调度算法得到如图 5 所示的调度表,矩形长度代表每个任务延时大小,9 个任务分成 4 组,其中任务{3,5,8,2}是分组后每组延时最大的传输任务,取它们传输延时的近似最大公约数(适度偏大)作为时间窗大小。

时间窗大小确定后,根据式(5)计算每组任务所需时间窗的数量 N_i。然后根据 RM 算法思想为每组任务分配授权时间窗。

$$N_i = T_i / T_{ts} \tag{5}$$

式中,T_i 为第 i 组任务的最大传输延时,T_{ts} 为时间窗大小。最终可得静态路由时段包含时间窗的数量 N 为:

$$N = \sum N_i \tag{6}$$

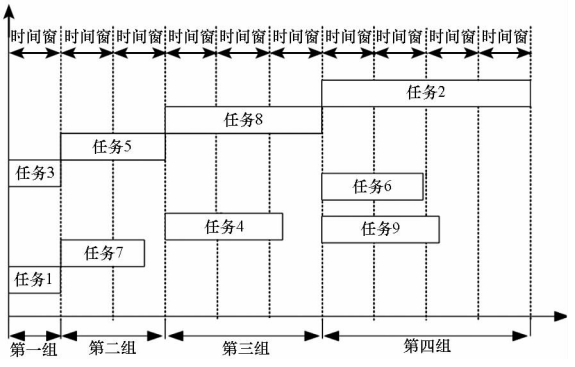

图 5　启发式调度算法与时间窗设计

Fig. 5　Heuristic scheduling algorithm and time-slot design

2.3　动态路由机制

静态路由时段已为大部分传输任务建立了调度表,动态路由时段只需传输一些随机、动态、无法预测的任务以及载荷数据。为了提高传输速度和网络效率,动态路由总体采用基于事件触发的虫洞路由机制[13]。但是动态信号的传输也有轻

重缓急之分。如,一些报警信号或某些时刻的视频数据均属于动态信号,但相对而言报警信号更急需处理。若不识别优先级轮询调度,可能导致报警信号因为延时得不到及时处理,因此本文提出对不同事件设定不同优先级,以保证需要紧急处理的事件优先传输。

SpaceWire 协议下的数据包格式如图 6 所示,由包头、数据和包尾组成,其中数据字符格式如图 7所示。其中,包尾有正常结束符(normal End Of Pocket, EOP)和错误包结束符(Error End of Packet, EEP)两种。为了尽可能减少对协议的修改和降低设计工作量,本文为不同事件设置优先级时,将优先级信息包含在包头地址中,如图 8 所示。使用 8 位地址的高 2 位描述传送数据的优先级,仅使用低 6 位描述目的地址。6 位地址最多仍可识别 $2^6 = 64$ 个目的地址,在大部分网络中是可行的。2 位优先级的权值可设置 00、01、10、11 四个优先级。当传输任务发生冲突,即在同一时刻两个传输任务的目的端口相同时,优先级高的传输任务可实现优先传输。该方法充分运用了包头地址识别方法识别优先级,未引进更多数据包,且符合 SpaceWire - D 协议,也未改变路由器结构,简化了设计过程,减少了设计工作量。

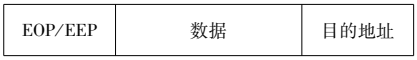

图 6 数据包格式

Fig. 6 Data packet format

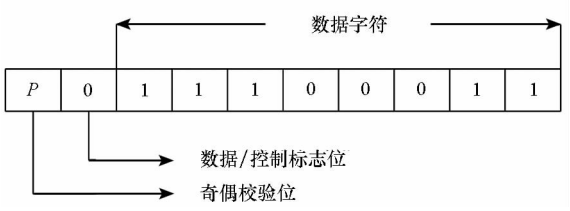

图 7 数据字符格式

Fig. 7 Data character format

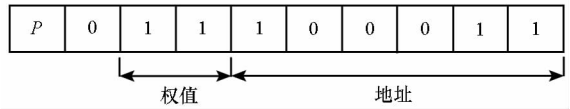

图 8 地址与优先级权值设计

Fig. 8 Design of address and priority value

本文采用的具有优先级的轮询动态路由调度算法的流程图如图 9 所示,主要步骤如下:

1)查询所有端口是否有任务需要传输;

2)分别识别地址 D_i 与权值 V_i。下标代表第几个端口,如 D_3 代表第三个端口缓存的传输任务的地址,V_3 代表第三个端口缓存的传输任务的优先级权值;

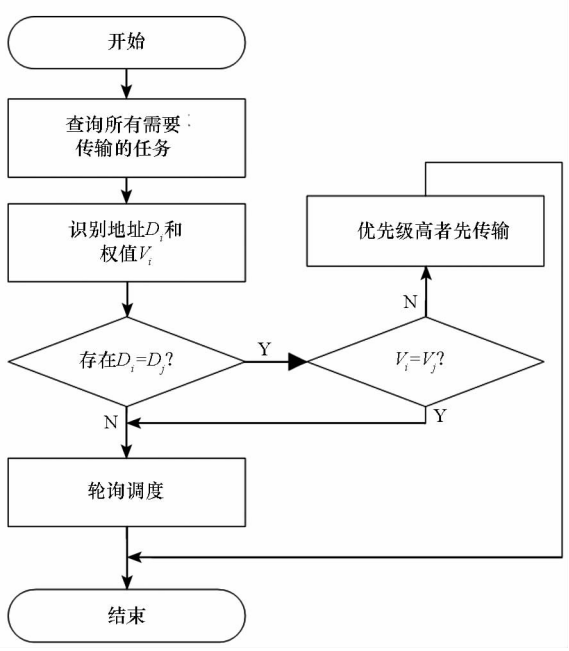

图 9 动态调度算法

Fig. 9 Dynamical scheduling algorithm

3)判定识别各任务的目的端口是否有冲突,即 $D_i = D_j$? 若不存在,则运用轮询调度传输,否则判别优先级;若权值相等,则轮询调度,否则优先级高者先传输。

动态路由时段包含时间窗的数量取决于通信周期和静态路由时段的大小,总体上需要保证动态路由时段、静态路由时段和清空时间窗的总和不能超过通信周期的大小。而通信周期的大小受实时性要求较高的周期性信号的周期约束。另外,由于动态路由过程存在阻塞延时的不确定性,因此要求动态节点具有备份功能,即对发出的数据包进行备份,收到接收确认包后销毁,否则在下一通信周期的动态路由时段再重新传输。为避免动态路由时段因未传输完毕的任务而影响下一通信周期的任务传输时,本文在每个通信周期的最后附加一个清空时间窗,用来清除动态路由时段未传输完毕的任务。

3 实验论证与分析

3.1 系统模型的搭建

为了评估本文所提出的网络路由机制的性能,在 OPNET[14-15] 中搭建了如图 10 所示包含一个路由和 5 个节点的系统仿真模型。图 10 中,router 为路由器,节点 node_5 为时间码节点,节点 node_1、node_2、node_3、node_4 作为外围节点,外围节点间

可相互传输数据包。为了简化分析和设计,这里假设系统仅包含单个路由器,且数据包的大小只有 2 种,传输任务如表 1 所示,其中 1 -2 代表源节点为 node_1,目的节点为 node_2 的传输路径。

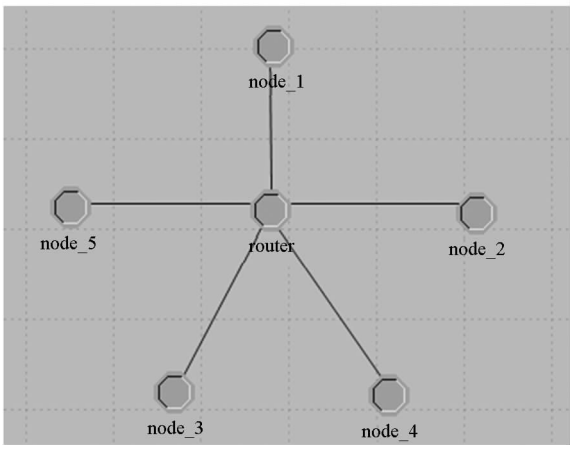

图 10　SpaceWire 系统网络

Fig. 10　SpaceWire network system

表 1　传输任务

Tab. 1　Transmission task

源节点	目的节点			
	1	2	3	4
1	1 -1	1 -2	1 -3	1 -4
2	2 -1	2 -2	2 -3	2 -4
3	3 -1	3 -2	3 -3	3 -4
4	4 -1	4 -2	4 -3	4 -4

图 11 给出了分别使用文献[7]、文献[8]和本文方法通过对表 1 给出的传输任务生成的调度表,其中矩形长度代表传输延时大小。方框内的数字代表传输任务的源节点和目的节点,如 2 -1,表示源节点为 2,目的节点为 1。调度表 1 为文献[8]方法生成的调度表,分组结果为:{1 -1,2 -2,3 -3,4 -4}、{1 -2,2 -3,3 -4,4 -1}、{1 -3,2 -4,3 -1,4 -2}和{1 -4,2 -1,3 -2,4 -3},静态周期为 T1。该方法可保证传输路径不冲突,但未考虑任务的传输延时,仅考虑在不冲突情况下每组任务最大化。调度表 2 和调度表 3 分别为使用文献[7]方法与本文方法的分组结果。由于二者均根据传输延时相近原则对传输任务分组,故分组结果相同,均为:{1 -1,2 -4,3 -3,4 -2}、{1 -2,2 -1,3 -4,4 -3}、{1 -3,2 -2,3 -1,4 -4}和{1 -4,2 -3,3 -2,4 -1}。但是,由于文献[7]为并发调度算法,其调度表的静态周期仍为 $T1$;而本文采用多时间窗并行调度算法,并通过最大公约数法设计时间窗,虽然时间窗数量增加了,但其静态周期为 $T2$。可见 $T2 < T1$,说明采用本文方法所需静态周期减少了。

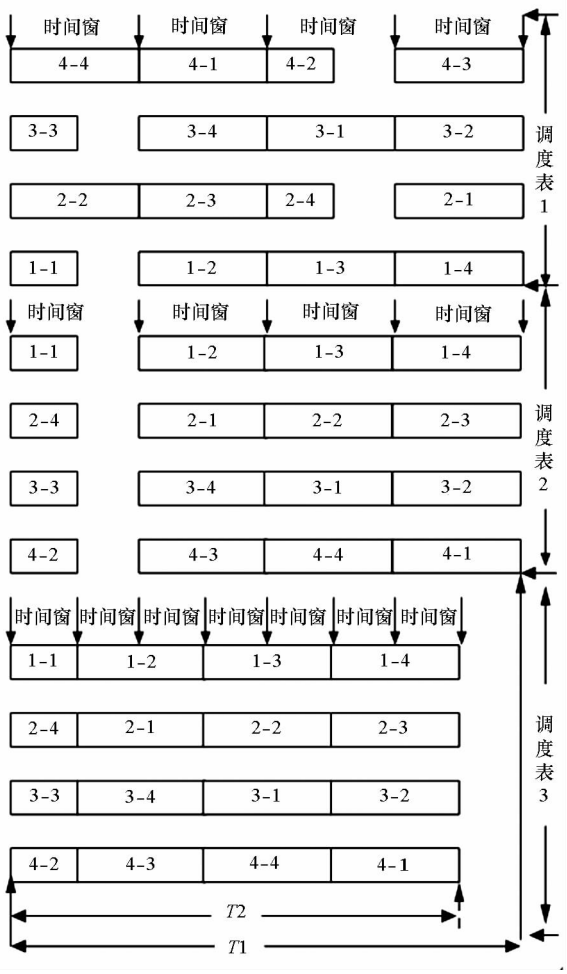

图 11　静态路由调度表

Fig. 11　Static routing scheduling table

图 12 和图 13 所示分别为调度表 1(文献[7]方法)和调度表 3(本文方法)实现的确定性传输仿真实验数据,横轴表示仿真时间,竖轴表示网络传输延时。由仿真结果可见,二者均实现了确定性传输。

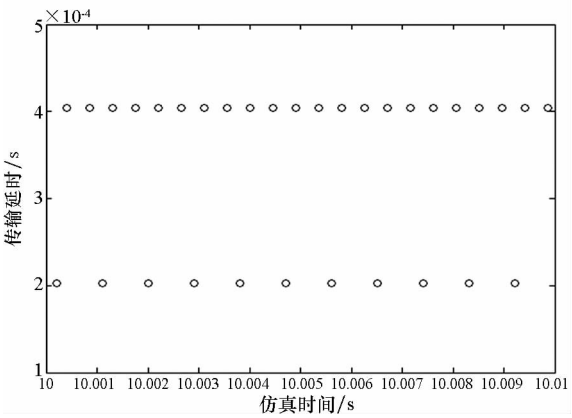

图 12　静态调度表 1 传输延时仿真结果

Fig. 12　Simulation result of static scheduling table 1 transmission delay

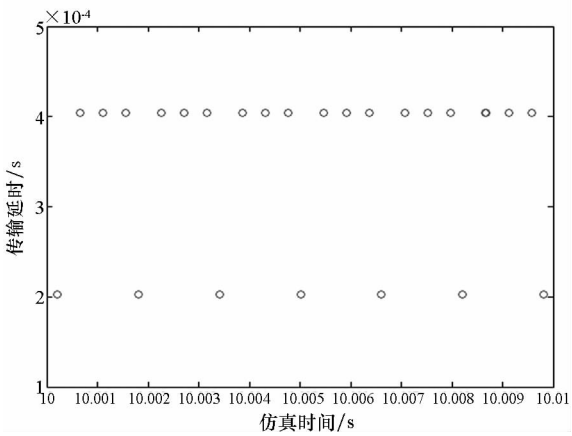

图 13 静态调度表 3 传输延时仿真结果

Fig. 13 Simulation result of static scheduling table 3 transmission delay

图 14 和图 15 分别为两种方法的吞吐率仿真图,横轴表示仿真时间,竖轴表示吞吐率大小。由仿真结果可以看出,本文的静态调度算法(调度表 3)实现的确定性传输的吞吐率明显高于文献[7](调度表 1)的吞吐率。

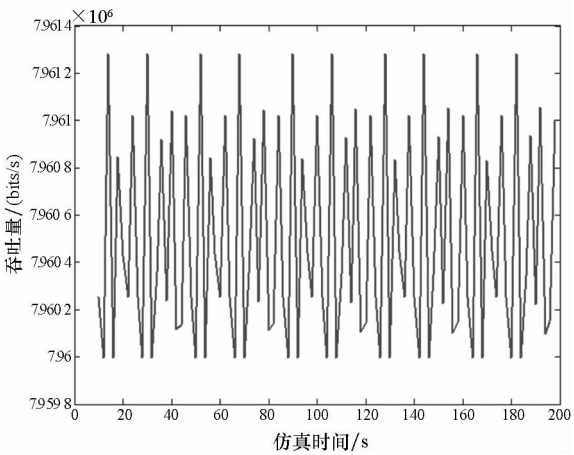

图 14 静态调度表 1 吞吐率仿真结果

Fig. 14 Simulation result of static scheduling table 1 throughput

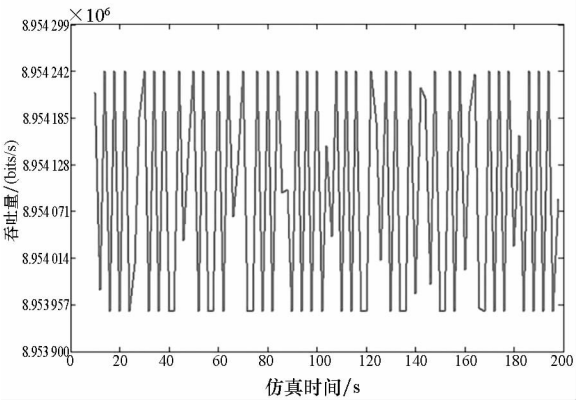

图 15 静态调度表 3 吞吐率仿真结果

Fig. 15 Simulation result of static scheduling table 3 throughput

3.2 动态路由仿真实验分析

图 16 所示为采用本文提出的带优先级的轮询动态路由调度算法,在 3 个传输任务发生路径冲突时传输延时的仿真结果,其中圆圈、方框和五角星标识的任务的优先级由高到低递减。3 个传输任务在无阻塞独立传输延时均为 2.022 96E - 4 s,由于本文在虫洞路由的基础上增加了优先级标记与识别功能,因而圆圈表示的最高优先级任务传输延时最短,仍为 2.022 96E - 4 s;方框标识的优先级次高的任务传输延时次之,为 3.022 96E - 4 s;而五角星代表的优先级最低的任务传输延时最大,为 4.022 96E - 4 s。由仿真结果可见,本文提出的带优先级的轮询动态路由调度算法可识别事件优先级,确保紧急事件优先传输。

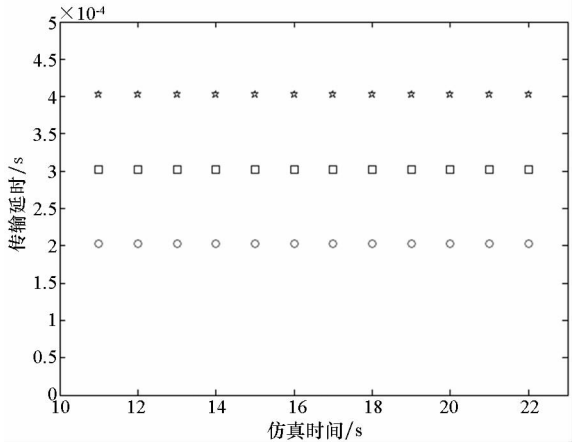

图 16 动态路由传输延时仿真

Fig. 16 Simulation result of dynamical routing transmission delay

4 结论

为使控制数据和载荷数据共享 SpaceWire 网络,本文提出了基于 SpaceWire - D、兼顾时间触发和事件触发的静、动结合的高速 SpaceWire 路由机制。在 OPENT 中搭建了网络系统仿真模型,实现了单一网络的数据传输。实验结果表明,静态路由时段网络吞吐量较现有调度算法有明显提高,动态路由能确保紧急事件优先传输。

在静态路由时段,将启发式调度算法与最大公约数法时间窗设计相结合,首次实现了延时相近且不冲突的传输任务的多时间窗并发调度;采用近似最大公约数法设计时间窗大小,虽然可能导致时间窗个数增加,但对整个网络吞吐率有明显提升。在动态路由时段,利用 SpaceWire 高速传输的特性,设计了具有优先级的轮询调度方法,

在保证载荷数据高速传输的同时,能使突发性紧急事件得到及时处理,因而具有更好的实用性。

参考文献(References)

[1] ECSS. Space engineering: SpaceWire-links, nodes, routers and networks: ECSS-E-50-12C[S]. European Space Agency for the Member of ECSS, 2003.

[2] ECSS. Space engineering: spacewire-links, nodes, routers and networks: ECSS-E-ST-50-12C[S]. European Space Agency for the Member of ECSS, 2008.

[3] Yi D L, Yu L X, Fei H D. SpaceWire standard and improved wormhole router design[C]//Proceedings of IEEE Aerospace Conference, 2012.

[4] Tunesi L, Armbruster P. On-board hierarchical network[J] Proceedings of the SPIE, 2004, 5234: 539 - 549.

[5] Parkes S, Gibson D, Ferrer A. SpaceWire-D: deterministic data delivery with SpaceWire[C]//Proceedings of the 3rd International SpaceWire Conference, St. Petersburg, 2010.

[6] Chen Y, Takada M, Kurachi R, et al. A scheduling method of RMAP packets for SpaceWire-D[C]//Proceedings of International SpaceWire Conference, 2013.

[7] Raszhivin D, Sheynin Y, Abramov A. Deterministic scheduling of SpaceWire data streams networks and protocols, short paper[C]//Proceedings of International SpaceWire Conference, 2013: 205 - 208.

[8] 杨志, 李国军, 杨芳, 等. SpaceWire 星载网络通信协议设计[J]. 宇航学报, 2012, 33(2): 200 - 209.
YANG Zhi, LI Guojun, YANG Fang, et al. Design of communication protocol for SpaceWire on-board networks[J]. Journal of Astronautics, 2012, 33(2): 200 - 209. (in Chinese)

[9] Steiner D, Maiter R, Jameux D. Time-triggered services for SpaceWire[C]//Proceedings of International SpaceWire Conference, 2008.

[10] 代真, 何锋, 熊华钢. 混合机制下的 SpaceWire 传输延时仿真分析[J]. 计算机工程与设计, 2015, 36(1): 1 - 5, 102.
DAI Zhen, HE Feng, XIONG Huagang. Transmission delay simulation and analysis in SpaceWire under combined mechanism[J]. Computer Engineering and Design, 2015, 36(1): 1 - 5, 102. (in Chinese)

[11] 陈熙之, 刘晓锋, 杨明川. SpaceWire 总线网络实时传输性能研究[J]. 通信技术, 2012, 45(9): 93 - 95, 99.
CHEN Xizhi, LIU Xiaofeng, YANG Mingchuan. Research on real time transmission performance of SpaceWire network[J]. Communication Technology, 2012, 45(9): 93 - 95, 99. (in Chinese)

[12] Liu C L, Layland J W. Scheduling algorithms for multiprogramming in a hard-real-time environment[J]. Journal of the ACM, 1973, 20(1): 46 - 61.

[13] 钟雪艳. 基于 FPGA 的 SpaceWire 路由器的设计和验证[D]. 南京: 南京航空航天大学, 2013.
ZHONG Xueyan. Design and verification of SpaceWire router based on the FPGA[D]. Nanjing: Nanjing University of Aeronautics and Astronautics, 2013. (in Chinese)

[14] 侯剑儒, 陈晓敏. SpaceWire 时延抖动的仿真[J]. 国防科技大学学报, 2013, 35(5): 114 - 119.
HOU Jianru, CHEN Xiaomin. The simulation on the delay jitter of SpaceWire[J]. Journal of National University of Defense Technology, 2013, 35(5): 114 - 119. (in Chinese)

[15] 陈敏. OPNET 网络仿真[M]. 北京: 清华大学出版社, 2004.
CHEN Min. OPNET network simulation[M]. Beijing: Tsinghua University Press, 2004. (in Chinese)