

引用格式:潘雄,邓威,苑政国. SpaceWire 网络层分析的时间自动机模型[J]. 微电子学与计算机, 2019, 36(1): 1-5. [PAN X, DENG W, YUAN Z G. Modeling and Analysis of SpaceWire Network Based on Timed Automata[J]. Microelectronics & Computer, 2019, 36(1): 1-5.]

SpaceWire 网络层分析的时间自动机模型

潘 雄,邓 威,苑政国

(北京航空航天大学 光电技术研究所,北京 100083)

摘 要:SpaceWire 总线作为航天器数据/控制的“神经中枢”,其网络层的结构 and 应用设计是系统可靠性的重要影响因素.为了在 SpaceWire 总线网络层设计部署过程中,对其进行形式化分析,发现设计缺陷,提高可靠性.提出了一个用于 SpaceWire 网络层验证的形式化分析模板框架,将网络层的核心要素:实时数据包、终端节点、路由器和路由机制都使用时间自动机建模.然后根据具体案例将之实例化,并在 UPPAAL 模型检验工具中根据规范提出性质进行检验.典型案例的分析验证了所提出的方法的有效性.

关键词: SpaceWire;时间自动机;模型检验;UPPAAL

中图分类号: TP301.1;TP316

文献标识码: A

文章编号: 1000-7180(2019)01-0001-05

Modeling and Analysis of SpaceWire Network Based on Timed Automata

PAN Xiong, DENG Wei, YUAN Zheng-guo

(Institute of Optics and Electronics, Beihang University, Beijing 100083, China)

Abstract: As the "nerve center" of spacecraft data/ control, SpaceWire structure and application design of its network layer are important influencing factors of system reliability. In the process of designing and deploying SpaceWire network, in order to analyze it formally, find defects and improve the reliability, a formal analysis template framework for SpaceWire network layer analysis is proposed. All the components of the network layer: real-time packet, terminal node, router and routing mechanism are modeled as timed automata. Then a simplified case is built in the UPPAAL model checking tool, and the extraction property is verified. The analysis of typical cases verifies the effectiveness of the proposed method.

Key words: SpaceWire; timed automata; model checking; UPPAAL

1 引言

SpaceWire 是欧空局(ESA)为应对复杂空间任务而提出的一种面向航天应用的高速(2~400 Mbits/s)、双向的、全双工的串行总线网络.近年来,SpaceWire 得到了快速发展应用,典型的如 NASA LRO、ESA ExoMars、JAXA ASHARO-1 等^[1].

对 SpaceWire 进行仿真和验证,是设计和部署过程中的必须步骤.从协议仿真的角度来看,目前存

在多种仿真平台,如 OPNET、DCNSimulator、NS-2 等,文献[2]即研究了 SpaceWire 网络的 OPNET 模型.从验证方法学的角度来说,上述仿真模拟只可能是执行了系统的部分行为,也就是说只能证明错误存在,不能证明错误不存在.

以完备性为目标的形式化方法是系统验证的发展方向.目前存在的 SpaceWire 相关的形式化研究主要是底层的实现或机制的形式验证,如文献[3-4]分别对 SpaceWire 的 DS 编码和译码的门级电路与规范

一致性进行了基于 HOL4 逻辑系统的定理证明、文献[5]在 ACL2 定理证明器中对 SpaceWire 信誉逻辑的 xMAS 模型进行形式化并验证、文献[6]对链路接口检错机制的 VHDL 代码的在 NuSMV 工具中进行了验证、文献[7]考虑了现实世界的随机概率因素,在 PRISM 中对 SpaceWire 交换层链路建立马尔科夫链决策过程模型(Markov Decision Process, MDPs)。而在无线传感网,汽车总线,航空总线上,采用形式化方法进行建模分析,已取得显著成效。

SpaceWire 网络层的结构和应用分布合理是系统安全性的重要方面,其不仅是逻辑要求严格的,也是时间要求严格的。因此,本文提出基于时间自动机理论^[8]的精确形式语义来对 SpaceWire 网络层进行抽象建模,在模型检验工具 UPPAAL 中对其进行验证,为可靠的 SpaceWire 系统设计部署提供帮助。

2 SpaceWire 拓扑结构及路由方式分析

SpaceWire 主要特点在于其网络化的结构,和去中心化的数据交换方式。如图 1, SpaceWire 网络由多个链路、终端节点和路由器组成。通常,终端节点就是我们的应用设备:传感器、数据处理单元、遥测子系统、大容量存储器等。

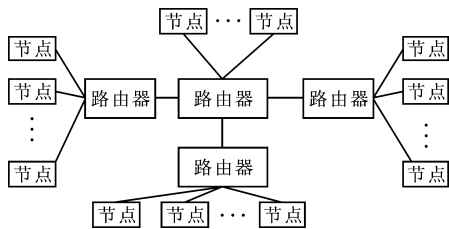


图 1 SpaceWire 网络

由于应用环境的要求,SpaceWire 尤其注重路由器上存储空间的节省,比如防止空间射线可能导致单粒子翻转而引发系统错误,因而采用虫洞路由(Wormhole routing)的方式进行寻址传输:当一个数据包需要传输时,发出一个包头字节,进行路由寻址。如果下一跳链路是空闲的,则继续向前寻址,直到达到目标节点,源节点和目标节点之间的链路被全部占有,才开始正式进行数据传输,数据直接从源节点发送到目标节点,路由器上不进行存储。如果下一跳链路被其他数据包占用,则其需要等待。

虫洞路由具有许多优势,如路由器上无需数据包缓存,支持任意包格式,快速路由切换等,但存在一个主要的问题即不确定性的阻塞。如果一个数据包的前向路径上存在尚未准备好的点或正在被占

用,则其被阻塞。如图 2 所示,由于 Node₁ 所发出的数据正在传输,使得由 Node₂ 待发送数据包的寻址在 Router₂ 的 4 端口被阻塞,尽管它还没有进行传输,但是已经占据了被阻塞端口之前的链路,又使得 Node₃ 待发送的数据包被阻塞在 Router₁ 的 4 端口。

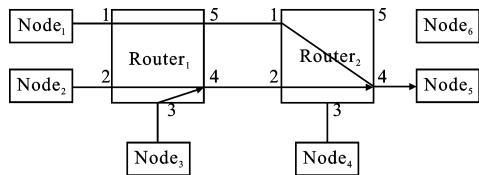


图 2 SpaceWire 阻塞示意

3 SpaceWire 网络层的时间自动机建模

3.1 时间自动机建模概述

时间自动机理论由 Alur 和 Dill 建立,是有限状态自动机在连续时间上的一种自然扩展,它提供一种简单且有效的方法描述带有时间因素的状态转换计算机模型,为实时系统的过程建模和模型验证提供了重要的理论依据。

定义:一个时间自动机表示为一个六元组 $\langle S, S_0, \Sigma, X, I, E \rangle$, 其中, S 为有限状态集合; $S_0 \subseteq S$ 是初始状态; Σ 是事件或指定信息的集合; X 是时钟变量的集合; I 是保卫公式的集合, E 即有向边(edge)的集合, 状态转移 $\langle s, s', \alpha, \delta, \lambda \rangle \in E$ 表示系统在事件 α 和时钟约束 δ 都被满足的两个条件同时作用下,从状态 s 转移到状态 s' , λ 为转换过程中的赋值序列。

人们研发了多种时间自动机的模型检验工具,如 UPPAAL、Kronos、TimedCOSPAN 等,其中由瑞典乌普萨拉大学(Uppsala University)和丹麦奥尔堡大学(Aalborg University)共同开发的 UPPAAL 应用最为广泛。在 UPPAAL 中每个过程被描述为有限控制结构,实数值时钟,和变量组成的时间自动机。UPPAAL 提供,过程之间通过同步通道(chan 类型)来进行通信,发送消息被表示为 $c!$, 而接受消息被表示为 $c?$ 。除通常的状态外,还有紧急状态(urgent locations):在此状态上,无时间流失;以及限制状态(committed locations):当同一时刻有多个转移发生时,为避免出现错误,强制约束位置转移的顺序,系统的下一次位置转移必须从此位置发出。图 3 说明了表达形式,其中 s_1 , s_2 为初始状态, s_3 为 urgent 状态, s_4 为 committed 状态, reset 为同步通道。

3.2 模型构造

在分析 SpaceWire 网络层的结构和行为后,将

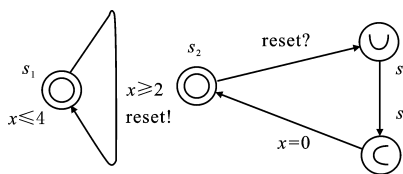


图 3 UPPAAL 模型意义辅助说明

其分为几个组成部分:数据包,终端节点,路由器端口,以及路由调度机制分别建立时间自动机.

3.2.1 数据包时间自动机

对网络中的行为需求进行分析,将数据包简单分为两种,一类是周期性自动生成,一类则是依赖触发性的,这种依赖关系(如控制,应答)可以通过依赖矩阵即二维数组表示.这种依赖关系实际上就是SpaceWire应用层任务流的抽象建模.如RMAP读任务为例,在网络中产生两个数据包,其应答数据包依赖于读命令数据包而产生.如:bool Depend[][] =

$\begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$ 即表示任务 1 依赖于任务 3, 任务 3 依赖

于任务 2. 通过这种方式可以对复杂应用层行为之间的关系进行抽象.

如图 4 是数据包时间自动机模板.实例中,每个数据包都是数据包模板参数化属性(如长度,路径等)的实例化,这些信息及所用处理函数编辑在模型声明中.数据包的时间自动机使用一个编号参数 id,作为数据包的索引.

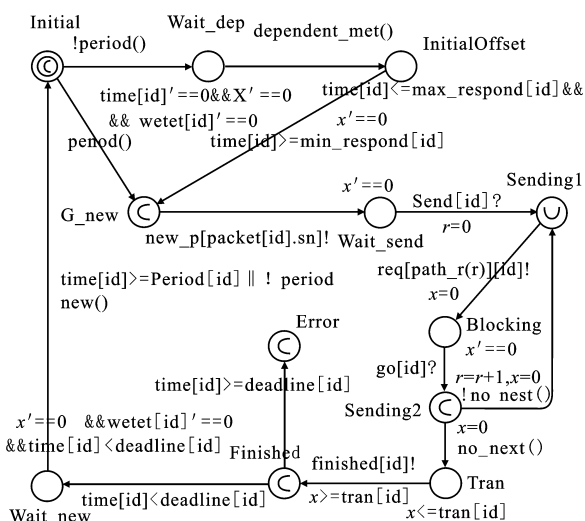


图 4 数据包时间自动机.

(1)Initial 为初始状态,周期性数据直接到 G_new 状态. 非周期则到 Wait_dep 状态,等待触发, dependent_met() 函数判断触发满足后,在响应时间

段 $\text{min_respond}[\text{id}]$ 到 $\text{max_respond}[\text{id}]$ 之间转移到 G_{new} 状态, 表示生成新数据包. 形如 $x' = 0$ 表示 x 时钟暂停, 即 $\text{wetet}[\text{id}]$ 和 $\text{timed}[\text{id}]$ 和从 InitialOffset 或 G_{new} 开始计时.

(2)在 G_new 后立即发出 new_p[packet[id][sn]]! 到相应节点转移到 Wait_send, packet[id][sn]是数据包源节点编号. 之后接收 send[id]? 进入 Sending1 状态,表示开始寻址发送.

(3) r 用于标志数据包目前已经经过的路由跳数, 在 Sending1 后发出 $\text{req}[\text{path_r}(r)][\text{id}]!$ 信号, 请求路径上该跳路由的相应端口进入 Blocking 状态. 收到 $\text{go}[\text{id}]?$ 进入 Sending2 状态, 此时通过 $\text{no_next}()$ 函数判断是否还有下一跳路由, 若有, 则转 Segding1 状态继续下一跳寻址, 若无则进入 Tran 状态, 表示寻址结束, 开始传输.

(4)在 Tran 上,时钟 x 开始计时,tran[id]为存储于声明按照 SpaceWire 恒定速率和数据包长度计算所得的传输时间.传输完成后发出 finished[id]! 信号,到达 Finished 状态.wetet[id]纪录从数据产生到完成传输的时间.

(5) 在 Finished 时, 若 $\text{timed}[\text{id}] \geq \text{deadline}[\text{id}]$ 即出现超时情形, 到达 Error 状态. 若正常, 则到达 Wait_new 状态. 之后周期信号等待周期时间到达, 非周期直接回到 Initial 状态, new() 函数表示将所有时钟, 变量重置.

3.2.2 节点时间自动机

节点的主要作用即存储和发送数据. 如图 5 是节点的时间自动机模板, 编号索引参数位 `nid`.

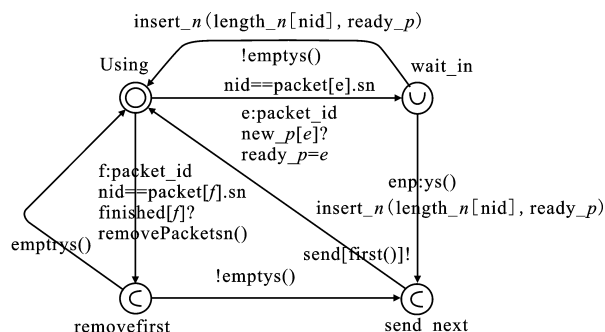


图 5 节点时间自动机

(1) Using 为初始状态,当收到 new_p[e]? 即生成新数据包到达 wait_in 状态,若节点队列不为空,函数 insert_n(~)将该新数据包插入节点队列最后,并回到 Using. 若节点队列为空,插入到队首,并转到 send_next 状态.

(2)在 Using 上,当收到 finished $[f]$? 表示当

前正在传输的数据传输完成, `removePacketsn()` 函数删除节点队首, 转移到 `removefirst` 状态. 若队列为空, 转回到 `Using`. 若队列不为空, 转到 `send_next`.

(3) 在 `send_next` 上, 立即发出 `send[first()]!` 信号, 表示发出此时的队首数据, `first()` 为获取队首数据包编号的函数.

3.2.3 路由器及路由机制时间自动机

路由器实质上可视为路由端口的集合, 所以如图 6 时间自动机其实是路由端口的行为模型, 路由器编号为 `rid`.

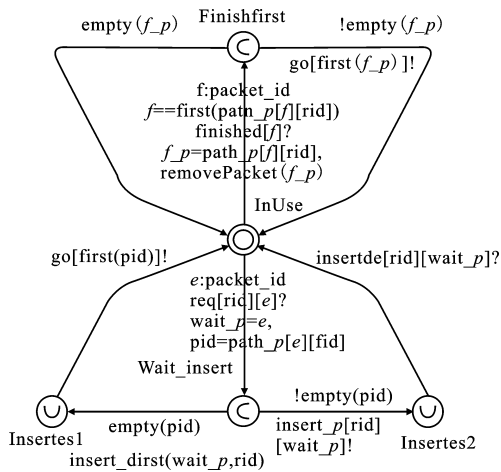


图 6 路由器时间自动机

(1) `InUse` 为初始状态, 当收到请求 `req[rid][e]!` 后转移到 `Wait_insert` 状态; 当收到 `finished[id]?` 信号后, `removePacket(f_p)` 删除端口队首并转移到 `Finishfirst` 状态. 其中 `path_p[e][rid]` 表示 e 数据包在 `rid` 路由器上所经过的端口号.

(2) 在 `Wait_insert` 状态, 若该端口队列为空, 使用 `insert_first(~)` 将申请的数据包编号插入到队首, 转移到 `Inserted1` 状态, 然后发出 `go[first(pid)]!` 信号转回 `InUse`; 若端口队列非空, 则发出 `insert_p[rid][wait_p]!` 信号给路由机制自动机, 转到 `Inserted2` 状态, 收到 `insert[rid][wait_p]?` 表示按照路由机制插入完成, 并回到 `InUse`.

(3) 在 `Finishfirst` 状态, 若端口队列为空, 直接回到 `InUse`; 若队列不为空, 发出 `go[first(f_p)]!` 表示继续通过下一个数据, 然后回到 `InUse`.

SpaceWire 网络层路由机制就是一个排序过程, 根据数据包的相应属性来处理队列中的顺序. 基础的有先入先出(FIFO)策略、轮询策略和优先级策略等. 路由竞争机制模型和路由端口模型相互独立, 所以复杂调度策略可以更好构建并加入到系统模板

中, 方便系统模板的扩展.

如图 7 是路由机制时间自动机, 初始状态是 `Idle`, 当收到 `insert_p[rid][wait_p]?`, 执行 `insert_FIFO(e)` 函数将 e 数据包插入到队列中, 转移到 `Inserting` 状态, 然后立即发出 `insert[rid][wait_p]!` 回到 `Idle`.

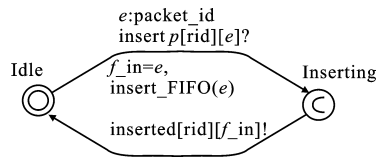


图 7 路由机制时间自动机

4 实例验证

本节使用一个简化案例来验证上述模型的有效性, 实验环境为 CPU Intel i7700HQ, 8GRAM, UP-PAAL4. 1. 19.

4.1 实例构建

在如图 8 这个网络中, 左边的应用终端 A_i 发送周期性的数据流 f_i 到 A_M , A_M 周期性发送 f_6 给 A_P , 同时, 每当 A_P 接收了一个 f_6 , 在 $[0.1, 0.2]$ 毫秒内响应发送一个 f_7 到 A_5 .

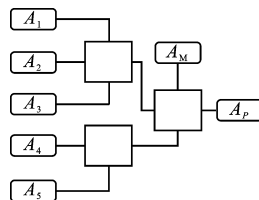


图 8 SpaceWire 案例

按照上述配置好数据流属性数组、路径参数和依赖矩阵建立 SpaceWire 网络层的分析模型. 若链路 50 Mbps, 考虑校验等因素, 简单设置 5 byte/ μ s.

4.2 验证

为了分析实例的需求是否安全性实时性满足, 按照设计规约的约束, 有两种方式可以执行: 完全遍历的模型检验和统计模型检验. 模型检验通过遍历状态空间来核验性质. 主要用于得到定性结论. 统计模型检验是对内存爆炸问题的一个半形式化折衷, 并不保证绝对的精确, 通过抽取样本, 根据一定的统计性算法, 可得到一些更具体的定量结论.

(1) 模型检验

不失一般性, 按照 BNF 的严格语法, 可以提出以下三个典型待验证性质:

Property1 所有数据包都能在时限内完成传输:

A [] forall (id:packet_id) notPACKET (id). Error

Property2 上层应用的依赖响应关系成功执行:
 $E\langle \rangle (\text{Depend}[6][5] == 1 \text{ and } \text{PACKET}(5). \text{G_new} \rightarrow \text{PACKET}(6). \text{Finished})$

Property3 数据包从产生到完成传输的时延:
 $\text{supwetet}[0], \text{wetet}[1], \text{wetet}[2] \cdots$

UPPAAL 在运行 176s, 并使 121 476 kB/238 526 kB 的最高驻存/虚拟内存后, 确认 Property1 被满足, 运行 13 s, 并使用 1 146 kB/2 761 kB 的最高驻存/虚拟内存后, 确认 Property2 被满足. 性质 3 用于最坏时延结果在表 1 中.

表 1 参数设置及最坏时延结果

Flow	f_1	f_2	f_3	f_4	f_5	f_6	f_7
Size/byte	4 000	500	2 000	500	4 500	2 500	300
sup/ms	2.3	2.3	2.3	2.3	2.3	0.5	0.26

(2)统计模型检验

为说明统计模型检验方法分析的作用, 按照 WMTL(weighted metric temporal logic)抽取如下典型性质:

Property4 数据 f_1 最坏时延的期望值:
 $E[\leq 10\,000; 1\,000](\text{max}; \text{wetet}[1])$

SMC 静态参数被设置为: 伪阴性(α)和伪阳性(β)为 $\alpha = \beta = 0.001$, 概率不确定参数为 $\epsilon = 0.01$, 正概率上界($+\delta$)和负概率下界($-\delta$)为 0.005. 得到结果 $2.294\,2 \pm 0.011\,471$ ms, 如图 9 为在 Plot Composer 中显示概率密度分布. 在 sup 性质得到的结果基础上, 通过概率密度分布图可以得到分布区间, 提供更多信息, 有助于分析.

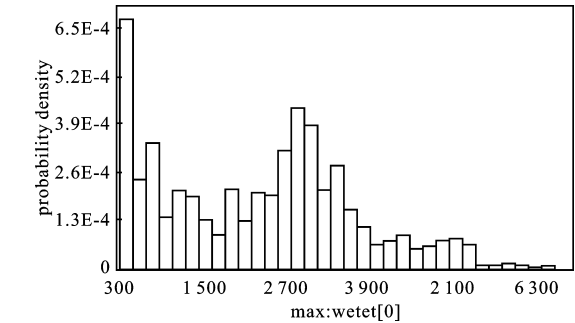


图 9 wetet[1]概率密度分布

5 结束语

通过对 SpaceWire 网络系统网络层结构和机制进行深入分析, 本文提出了用于 SpaceWire 网络系统实时性, 安全性分析的时间自动机形式化框架模板. 框架涵盖虫洞路由、路由机制和上层任务依赖等, 可以有效模拟实例化 SpaceWire 网络层. 在模型基础上, 借助形式化模型检验和半形式化统计模型检验技术可以对系统的相应定性和定量性质进行分析, 可以作为传统仿真测试的一个形式补充, 提高 SpaceWire 网络层设计的安全性实时性保证.

参考文献:

[1] European Space Agency. SpaceWire-home[EB/OL]. [2018-01-19]. <http://www.spacewire.esa.int/content/Home/HomeIntro.php>, 2017.

[2] 侯剑儒. SpaceWire 重要特性与仿真研究[D]. 北京: 中国科学院大学, 2013.

[3] 李黎明, 关永, 吴敏华, 等. 运用定理证明的形式化方法验证 SpaceWire 编码电路[J]. 小型微型计算机系统, 2012, 33(6):1372-1376.

[4] 张玉鹏, 施智平, 关永, 等. SpaceWire 译码电路在 HOL4 中的形式化验证[J]. 小型微型计算机系统, 2013, 34(8):1959-1963.

[5] 李艳春, 李晓娟, 关永, 等. 基于 xMAS 模型的 SpaceWire 信誉逻辑的形式化验证[J]. 计算机科学, 2016, 43(2):113-117.

[6] 董玲玲, 关永, 李晓娟, 等. 用 LTL 模型检验的方法验证 SpaceWire 检错机制[J]. 计算机工程与应用, 2012, 48(22):88-94.

[7] 李月星, 李晓娟, 关永, 等. SpaceWire 协议的形式化建模与概率分析[J]. 小型微型计算机系统, 2013, 34(9):2025-2029.

[8] 金仙力, 孙国梓, 杨庚. 基于层次时间自动机的动态行为取证建模方法[J]. 微电子学与计算机, 2012, 29(11):18-21.

作者简介:

潘 雄 男, (1978-), 博士, 副教授. 研究方向为光纤陀螺、单粒子防护、形式化验证.

邓 威(通讯作者) 男, (1993-), 硕士研究生. 研究方向为单粒子防护、形式化验证. E-mail: lyfyimeng@outlook.com.

苑政国 男, (1992-), 博士研究生. 研究方向为单粒子防护、形式化验证.