Journal of System Simulation

Volume 28 | Issue 11 Article 7

8-13-2020

CAN Bus Simulation System Based on OPNET

Yantao Liu
College of Engineering, Bohai University, Jinzhou 121013, China;

Xu Jing
College of Engineering, Bohai University, Jinzhou 121013, China;

Guiyang Xia
College of Engineering, Bohai University, Jinzhou 121013, China;

Qin Na
College of Engineering, Bohai University, Jinzhou 121013, China;

CAN Bus Simulation System Based on OPNET

Abstract

Abstract: CAN (Controller Area Network) is a typical kind of serial communication protocol. To solve the problem of bus contention, CAN adopts the mechanism of non-destructive bitwise arbitration. In order to realize simulation study on this mechanism, *simulation models of Medium Access Control (MAC),* process models, node models and network model of CAN *were built*. End-to-End delay of the simulation network was obtained by simulating these models under the condition of whether there existed a bus contention. The correctness of these models was proved by extensive simulation experiments and rigorous analyses. The models can be used in simulation study and industry application.

Keywords

CAN bus, bus contention, bitwise arbitration, network simulation

Recommended Citation

Liu Yantao, Xu Jing, Xia Guiyang, Qin Na. CAN Bus Simulation System Based on OPNET[J]. Journal of System Simulation, 2016, 28(11): 2692-2700.

第 28 卷第 11 期 系统仿真学报© Vol. 28 No. 11
2016 年 11 月 Journal of System Simulation Nov., 2016

一种基于 OPNET 的 CAN 总线仿真系统

刘宴涛，徐静，夏桂阳，秦娜
（渤海大学工学院，辽宁 锦州 121013）

摘要：CAN(Controller Area Network)总线是一种典型的串行通信协议，它采用非破坏性的比特仲裁机制解决总线冲突。为实现 CAN 总线比特仲裁机制的仿真研究，基于 OPNET 仿真环境建立了 CAN 总线协议介质访问控制子层(MAC, Multiple Access Control)的仿真模型，以及 CAN 总线的进程模型、节点模型和网络模型。通过运行所建立的仿真模型，统计存在总线争用和不存在总线争用两种情况下仿真网络的端到端延时情况。运行了大量的仿真实验并对仿真结果进行了严格的理论分析，证明了 MAC 进程模型的正确性。所建立的仿真模型可用于对 CAN 总线进行仿真研究和工业应用。

关键词：CAN 总线；总线冲突；比特仲裁；网络仿真
中图分类号：TP393 文献标识码：A 文章编号：1004-731X (2016) 11-2692-09
DOI: 10.16182/j.issn1004731x.joss.201611007

CAN Bus Simulation System Based on OPNET

Liu Yantao, Xu Jing, Xia Guiyang, Qin Na
(College of Engineering, Bohai University, Jinzhou 121013, China)

Abstract: CAN (Controller Area Network) is a typical kind of serial communication protocol. To solve the problem of bus contention, CAN adopts the mechanism of non-destructive bitwise arbitration. In order to realize simulation study on this mechanism, *simulation models of Medium Access Control (MAC),* process models, node models and network model of CAN *were built*. End-to-End delay of the simulation network was obtained by simulating these models under the condition of whether there existed a bus contention. The correctness of these models was proved by extensive simulation experiments and rigorous analyses. The models can be used in simulation study and industry application.

Keywords: CAN bus; bus contention; bitwise arbitration; network simulation

引言

在总线型网络中，信息是在一个共享传输介质上传输的，只要有一个节点通过总线向其他节点发送数据，总线的信道资源就会被占用。如果在同一时刻有两个或多个节点发送数据，必然会引起总线信道的争用，亦即产生冲突。控制器局域网(Controller Area Network，CAN)和以太网的介质访问控制协议 MAC(Multiple Access Control)都是基于总线竞争的机制，不可避免地会出现数据包的冲突。如何解决冲突对有效利用网络资源和提高通信有效性是十分关键的[1]。以太网采用载波侦听多路访问/冲突检测(Carrier Sense Multiple Access with Collision Detection，CSMA/CD)的机制解决总线冲突；CAN 总线是国际上应用十分广泛的串行通信协议，和以太网不同的是，CAN 总线采用的是基于比特仲裁的竞争机制[2]，即载波

收稿日期：2015-03-23 修回日期：2015-06-05；
基金项目：国家自然科学基金(61471045,61227001)；山东航天创新基金(2014JJ005)；
作者简介：刘宴涛(1975-)，男，吉林蛟河，博士，副教授，研究方向为网络仿真、网络编码；徐静(1989-)，女，山东临沂，硕士生，研究方向为网络控制、基于网络编码的拓扑识别。

侦听多路访问/比特仲裁(Carrier Sense Multiple Access with Bitwise Arbitration，CSMA/BA)。基于不同的仿真环境和建模方法，研究人员对 CAN 协议及其相关性能进行了大量的仿真研究工作，例如采用 UML(Unified Modeling Language)的建模方法建立了基于 CAN 总线的分布式控制系统模型[3]；利用 Petri 网实现 CAN 总线的建模[4-5]。此外，研究人员还分别在嵌入式仿真环境[6]、Stateflow 仿真环境[7]下实现了 CAN 总线的仿真；文献[8]还提出了 CAN 总线的实时计算模型。基于 Petri 网络的仿真工具本身仿真能力有限、计算能力不足，而且上述文献所提到的建模方法并都没有实现 CAN 总线的层次化建模，研究人员开始将目光转向利用 OPNET 实现 CAN 协议的仿真，如建立了 CAN 节点的应用层模型和 MAC 层进程模型[9]，但所建模型主要针对混合动力电动车 CAN 网络；利用 OPNET 仿真环境对 CAN 总线网络的总线负载、错误处理等性能进行仿真分析[10]，不过其节点应用层行为需要事先通过外部输入文件进行设定；利用 OPNET 实现了 CAN 总线网络的分层建模[11]，但该模型中 MAC 进程有限状态机的设计复杂，而且还需要单独设计 MAC 子层的帧间空隙子进程模型和错误帧处理子进程模型，这在一定程度上降低了仿真模型的运行效率；文献[12]对文献[11]中的节点模型和 MAC 进程模型进行了一定程度的改进，并仿真了节点优先级、传输速率、填充位等因素对网络实时性的影响[13]；也在 OPNET 仿真环境下建立了 CAN 总线的仿真模型，但该模型把以太网中 CSMA/CD 的 MAC 接入机制用于 CAN 总线，这与 CAN 总线协议标准要求的 CSMA/BA 接入机制不一致。

基于对 CAN 总线协议及其比特仲裁机制的研究，本文利用 OPNET 建立了 CAN 总线协议各子层的仿真模型，重点放在建立 CAN 总线 MAC 层的有限状态机模型。本文 MAC 子层有限状态机的设计方案复杂度低，不需要单独建立 MAC 帧间空隙模型，而且仲裁时间窗口、随机退避时间的设置更符合 CAN 总线网络的实际状况。最后，设置了 3 个仿真实验分别考察在无冲突、人为设定冲突和随机产生冲突的情形下网络的端到端延时情况，并对网络端到端延时进行了严格的理论分析和计算。由于 OPNET 采用模块化协议栈结构，我们建立的 LLC 和 MAC 模块可以供其他研究者即拿即用，像搭积木一样构建自己的协议栈。其他开发人员可以集中精力开发自己的应用层模型(应用层不是 CAN 总线标准的构成部分)，大大减少了下层协议的开发工作量。此外，本文采用的 CAN 总线 MAC 建模的思路和方法也可以供研究人员参考。

1 CAN 总线协议

1.1 CAN 协议分层

根据 OSI 参考模型的分层方法，CAN 协议[2]可以划分为物理层(Physical Layer，PL)和数据链路层(Data Link Layer，DLL)两层。其中，物理层又可分为物理信号子层、物理介质连接、介质相关接口 3 个子层；数据链路层由逻辑链路控制子层(Logical Link Control，LLC)和介质访问控制子层(Medium Access Control，MAC)组成。LLC 子层提供数据传输、远程数据请求等服务。MAC 子层是 CAN 协议的重要组成部分，主要完成发送数据的封装和接收数据的解封装、错误检测以及媒体接入管理等功能。MAC 子层的数据帧有 7 个字段，包括帧起始(SOF)、仲裁场(ARBITRATION Field)、控制场(CONTROL Field)、数据场(DATA Field)、循环冗余检测(CRC Field)、应答场(ACK Field)以及帧结束(EOF)。图 1 为 MAC 子层的数据帧格式。

SOF (1 bit)	ARBITRATION (12 bits)	CONTROL (6 bits)	DATA (32 bits)	CRC (15 bits)	ACK (2 bits)	EOF (7 bits)

图 1 MAC 子层数据帧格式

https://dc-china-simulation.researchcommons.org/journal/vol28/iss11/7
DOI: 10.16182/j.issn1004731x.joss.201611007

第 28 卷第 11 期 系统仿真学报 Vol. 28 No. 11
2016 年 11 月 Journal of System Simulation Nov., 2016

1.2 比特仲裁原理

CAN 采用总线型网络拓扑结构。当总线处于空闲状态时，任何节点都可以向总线发送数据，但在同一时刻只能有一个节点占用总线信道资源，否则必然会引起数据包的冲突问题。为了更好地解决冲突，CAN 总线采用 CSMA/BA 的总线接入机制。在这种机制中，显性位"0"的优先级高于隐性位"1"，也就是说当一个显性位和一个隐性位被同时发送到总线上时，仲裁胜出的是显性位。MAC 数据帧的仲裁场字段在信道争用过程中起到至关重要的作用，该字段由两部分组成，一部分是来自 LLC 数据包的 Identifier 字段，另一部分是用于区分远程帧和数据帧的 RTR 字段。Identifier 字段标识了数据帧的优先级，Identifier 字段值越小，数据帧的优先级越高。当两个或多个节点检测到信道空闲时，这些节点可能会同时向总线发送数据帧，这必然会导致数据包的冲突。为了避免冲突，各节点在发送数据帧之前先试探性地向总线发送帧起始字段和仲裁场字段。当多个节点发送的比特流同时进入信道时，各个比特在信道中参与逻辑"与"运算。各个节点会比较自己发送的比特和收到的比特，如果某个节点发送的比特为 1，而收到的比特为 0，则说明存在优先级更高的数据帧在准备发送，这时该节点仲裁失败。每个比特发送完后，会判决出节点是仲裁胜出还是失败，所有胜出的节点将继续进行后续比特的逐位仲裁，直到所有参与仲裁的比特全部发送完毕。这样，最终只有一个节点仲裁胜出并取得总线访问权，而其他仲裁失败的节点等待总线空闲时再次争取总线的访问权。

2 CAN 协议建模

2.1 网络模型

为了模拟 CAN 总线的工作过程，本文建立了如图 2 所示的网络模型，该网络有 6 个总线节点，它们之间通过 CAN 总线相连。

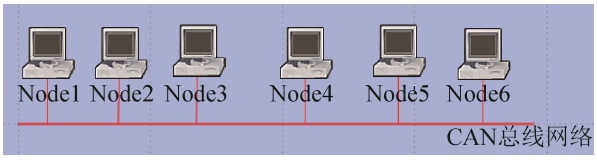

图 2 CAN 总线仿真网络模型

2.2 节点模型

根据 CAN 总线协议规范，本文创建了如图 3 所示的总线节点模型。其中，APP 模块代表应用层，模拟完成总线设备应用层的各种功能；LLC 模块完成总线设备 LLC 子层的功能；MAC 模块为 MAC 子层，完成总线设备 MAC 子层的各种功能。图 3 中在 RECEIVER 和 MAC 之间的虚线为状态统计线，用于收集总线状态。

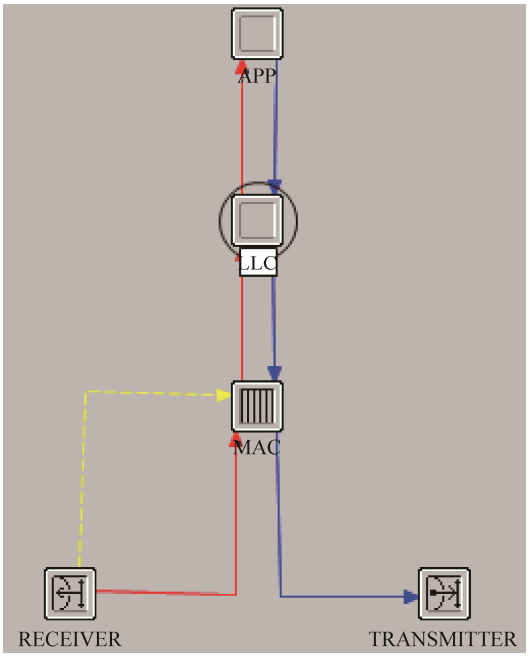

图 3 CAN 总线节点模型

2.3 进程模型

2.3.1 APP 进程模型

在 OPNET 中，进程模型主要用于模拟模块的功能，图 4 为本文建立的 APP 进程模型。该 APP 进程模型由 5 个状态构成，其中状态 INIT0 和 INIT1 主要用于初始化操作，读取网络或节点的参数配置；IDLE 为空闲状态；REC_PK 状态主要用于接收来自 LLC 子层的数据包；GEN_PK 状

态负责产生数据包，设定网络模型中节点 Node1 到 Node6 发送的数据帧的优先级依次降低。

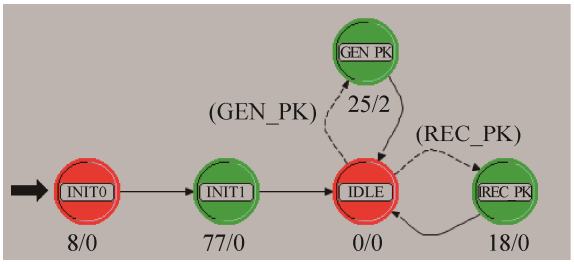

图 4 APP 进程模型

2.3.2 LLC 进程模型

LLC 模块主要实现数据包在 APP 模块与 MAC 模块之间的传输功能，是这两个模块相互传送数据的桥梁。本文建立的 LLC 进程模型如图 5 所示。在 LLC 进程中，状态 REC_PK 负责接收数据包，并根据所收数据包的来源做出相应的动作。如果数据包来自 APP 模块，则 REC_PK 进行解封装操作，并按照 LLC 数据包的格式重新封装后传送至 MAC 模块；如果数据包来自 MAC 模块，则按照 APP 模块数据包的包格式重新封装后传送给 APP 模块。

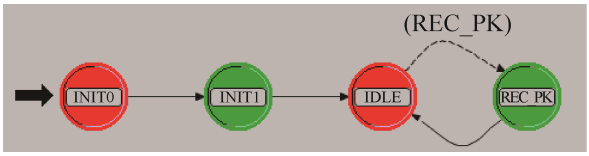

图 5 LLC 进程模型

2.3.3 MAC 进程模型

在 CAN 协议中，MAC 子层主要负责总线接入功能，是 CAN 协议数据链路层的核心。因此在 CAN 协议各子层的建模中，MAC 子层的建模是重点和难点。本文建立的 MAC 进程模型(如图 6 所示)除了初始化状态和空闲状态外，还包括非强制状态 BEGIN_TO_TRANS、BACKOFF、JUDGEMENT 以及强制状态 DELIVERY，状态跳转线上的数字表示不同的跳转条件(见表 1)。本文 MAC 进程模型有限状态机使用的状态个数少，状态间的跳转逻辑清晰，并且采用数组方式处理仲裁数据包，因此在设计复杂度上要低于已有文献的 MAC 建模方案。

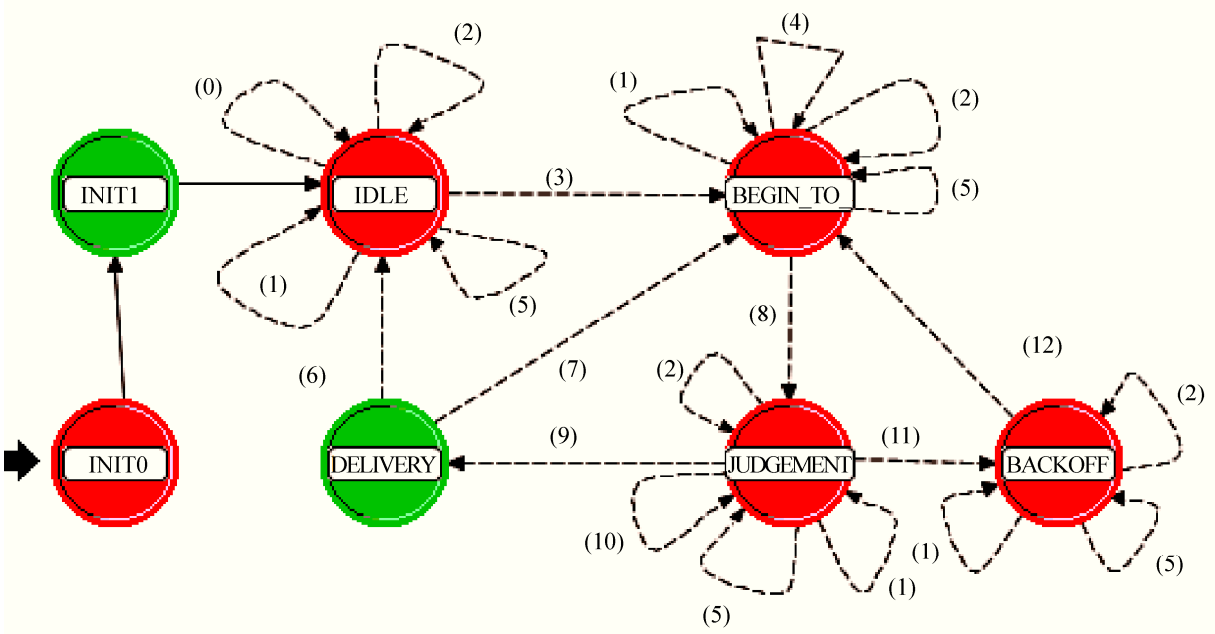

图 6 MAC 进程模型

DOI: 10.16182/j.issn1004731x.joss.201611007

第 28 卷第 11 期 系统仿真学报 Vol. 28 No. 11
2016 年 11 月 Journal of System Simulation Nov., 2016

表 1 MAC 进程状态间的跳转条件

数字	跳转条件	注释
0	(REC_PK_UP)/PROCESS_PK_UP()	处理 LLC 模块数据包
1	(REC_PK_DOWN)/PROCESS_PK_DOWN()	处理 MAC 模块数据包
2	(BUS_STATE_CHANGE)/UPDATE_BUS_STATE()	更新总线状态
3	SHOULD_SEND	准备发送
4	WAIT_INTERFRAME_SPACE_END	等待帧间空隙结束
5	(COLLECT_STATISTICS)/COLLECT_BITS_NUMBER()	收集统计量
6	(SUBQUE_EMPTY)/IN_TRANSMISSION=0	队列为空、无数据等待传输
7	!SUBQUE_EMPTY	子队列非空
8	(GOTO_ARBITRATION)/SET_WINDOW	送去仲裁、设置仲裁时间窗口
9	WIN	仲裁胜出
10	(OUTOF_ARBITRATION)/GET_ARBITRATION_RESULT()	仲裁结束、获取仲裁结果
11	(LOSE)/SET_BACKOFF_TIMER()	仲裁失败、准备退避
12	TIMER_OUT	随机退避时间结束

在 CSMA/BA 的建模工作中，仲裁时间窗口是一个非常关键的参数，它的设置直接关系到建模的准确性和有效性。每个比特发送到总线上所需要的时间称为比特时间，当发信机的发射速率为 500 kbps 时，比特时间为 2 us。每个总线节点把由 SOF 字段及 ARBITRATION 字段(共 13 个比特)构成的仲裁数据包发送到总线上共需要 26 us。为了使总线上所有节点都能够接收到这个仲裁数据包，还需要 2 us 的数据传播时间。因此 MAC 进程中的仲裁时间窗口被设定为 28 us。另外，在 CAN 总线协议标准中，为了保证数据帧能够被节点正确接收并做好接收下一帧的准备，规定总线上两个 MAC 帧之间必须插入 6 us 的帧间空隙。

CAN 节点在发送数据之前，先试探性地向总线上发送 SOF 和 ARBITRATION 字段，同时节点也会接收总线上其他节点发送的 ARBITRATION 字段。各节点将收到的所有 ARBITRATION 字段值存入到一个数组中。经过一个仲裁时间窗口后，各节点读取该数组中的所有 ARBITRATION 字段值，并比较这些值的大小。每个节点的优先级互不相同，因此 ARBITRATION 字段值也各不相同，MAC 进程通过比较 ARBITRATION 字段值的大小就可以判断出哪个节点仲裁胜出，哪些节点失败。下面结合图 6 所示的有限状态机解释所建 MAC 进程是如何实现仲裁的。仿真初始化完成后，进程将停留在 IDLE 状态并等待数据包的到来。当有数据包到来时，进程跳转到 BEGIN_TO_TRANS 状态，如果总线空闲且前一帧数据已发送完毕，MAC 进程将读取并发送数据包的 SOF 字段及 ARBITRATION 字段，同时，节点也会接收总线上其他节点发送的 SOF 字段及 ARBITRATION 字段，并将所有 ARBITRATION 字段值(包括节点本身发送的)存入到定义好的数组中，以便在仲裁时比较值的大小。当跳转到 JUDGEMENT 状态时，各节点在仲裁时间窗口结束后读取存在数组中的 ARBITRATION 字段值，并与节点自己的 ARBITRATION 值进行比较。如果发现存在比自己的 ARBITRATION 字段值更小的节点，则说明存在比自己优先级更高的数据包等待发送，此时 ARBITRATION 字段值更小的节点仲裁胜出。如果节点仲裁胜出，其 MAC 进程的有限状态机从 JUDGEMENT 状态跳转到 DELIVERY 状态，在这个状态下，MAC 进程读取 ARBITRATION 字段之后的字段值并发送出去；而其他仲裁失败的节点将从 JUDGEMENT 跳转 BACKOFF 状态，在经过一段随机的退避时间后，节点会在总线空闲时再次争取总线的访问权。

在本进程中设置的随机退避时间由 100*tbit 以及在区间[0,10*tbit]之间服从均匀分布的随机时间两部分组成，即

$t_{backoff}$=100*tbit+op_dist_uniform(0,10*tbit) (1)

其中，tbit 即比特时间，取值 2 us；函数 op_dist_uniform()为 OPNET 的核心函数，该函数用来在区间[0,10*tbit]上随机选取一个值。

2.4 链路模型

在对所建模型进行仿真前还需要设置 CAN 总线的链路模型。本仿真实验将链路的 coll model 属性设为 NONE，将属性 ecc model，error model，propdel model 及 txdel model 分别设置为 dbu_ecc，dbu_error，dbu_propdel 及 dbu_txdel，并设置总线数据速率为 500 kbps。

3 仿真实验及结果分析

本节基于图 2 所示的网络模型，设置了三个仿真实验，设置仿真时间为 50 s。运行仿真并收集仿真结果，同时仿真网络的端到端延时情况给出了理论分析。仿真实验基于 WINDOWS XP 操作系统，仿真软件采用 OPNET 14.5，并采用 Microsoft Visual Studio 2010 作为底层编译器。

3.1 端到端延时的定义及计算方法

网络参数端到端延时的定义如下：

端到端延时是指数据帧从发送节点开始发送到接收节点正确接收所经历的时间，该延时包括数据传输延时 t_a、数据在总线上的传播延时 t_b 和其他延时 t_c 三个部分。

首先，数据传输延时是指发送节点从发信机将数据帧所有比特发送到总线上所需要的时间，由于本文中设定的发射机数据速率是 500 kbps，所以

t_a=数据帧所有比特数/500 kbps (2)

其次，数据传播延时是指数据帧在总线上进行传播所需要的时间，该值与电磁波在总线上的传播速度有关，本仿真实验将传播速度设为 0.65 倍光速；

为了使各节点能够正确地接收数据帧，CAN 协议规定两个 MAC 帧之间必须经过 6 us 的帧间空隙；为使各个节点能够正确接收总线上所有节点发送的仲裁数据包，本文设置了 2 us 的仲裁数据包接收时间；在仲裁中失败的节点需要经过一段随机的退避时间后再次争取总线的访问权。因此，其他延时主要包括帧间空隙、仲裁数据接收时间和随机退避时间等延时。

3.2 仿真实验 1 及结果分析

仿真设置：仿真网络中的 Node3 作为目的接收节点，其他总线节点分别在 5 s，10 s，15 s，20 s 和 25 s 向 Node3 发送数据，此时不存在总线访问冲突问题。运行仿真，得到了仿真实验 1 中仿真网络端到端延时的仿真结果，如图 7 所示。

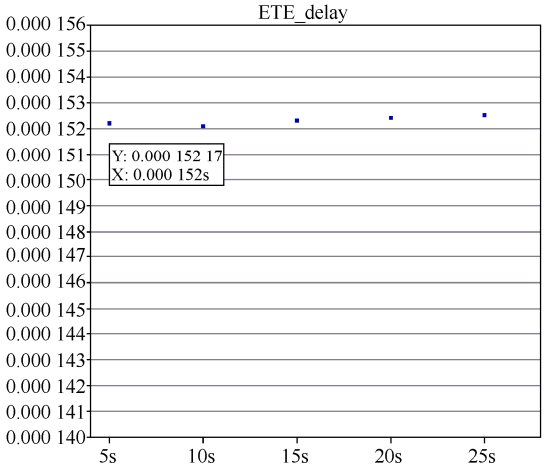

图 7 端到端延时仿真结果

仿真实验 1 中端到端延时的理论分析：由 MAC 子层的数据帧可知，节点向总线上发送的 MAC 数据帧共计 75 bit。由公式(2)可得在数据速率为 500 kbps 的总线上数据传输延时为

$$t_a = 75/500\,000 = 0.000\,15\ \text{s}$$

在本仿真网络中，相距最远的两个节点间的距离为 132.86 m，因此总线上的数据传播延时最大不超过

$$t_{max} = 132.86/(0.65\times 3\times 10^8) \approx 68.13\times 10^{-8}\ \text{s},$$

第 28 卷第 11 期 系统仿真学报 Vol. 28 No. 11
2016 年 11 月 Journal of System Simulation Nov., 2016

约为 0.68 us。

在仿真实验 1 中，各节点在不同时刻向 Node3 发送数据，因此端到端延时的计算不需要考虑由随机退避时间及帧间空隙引起的延时。以 Node1 为例计算 Node1 与 Node3 之间的端到端延时 t_{13}，Node1 到 Node3 的数据传播延时 $t_b = 17.34\times10^{-8}$ s。仿真实验 1 中其他延时 $t_c = 0.000\,002$ s，所以 Node1 与 Node3 之间的端到端延时为

$$t_{13} = t_a + t_b + t_c = 0.000\,15 + 17.34\times10^{-8} + 0.000\,002 \approx 0.000\,155\,2\ \text{s}$$

该结果与图 7 所示的端到端延时结果(图 7 中的第一个点)一致。其他节点到目的接收节点 Node3 之间端到端延时的计算方法和上述方法相同，由于各节点到 Node3 的距离有所不同，使得它们之间数据传播延时的计算值不同，从而最终得到的端到端延时也会有差异，所以在图 7 中采集到的 5 个延时结果的取值略有不同。

3.3 仿真实验 2 及结果分析

仿真设置：总线节点 Node1 和 Node2 在第 5 秒、Node4 和 Node5 在第 10 s 时向 Node3 发送数据，此时总线上有两个节点同时访问总线，总线上存在访问冲突。运行仿真，可以得到仿真实验 2 的网络端到端延时仿真结果，如图 8 所示。

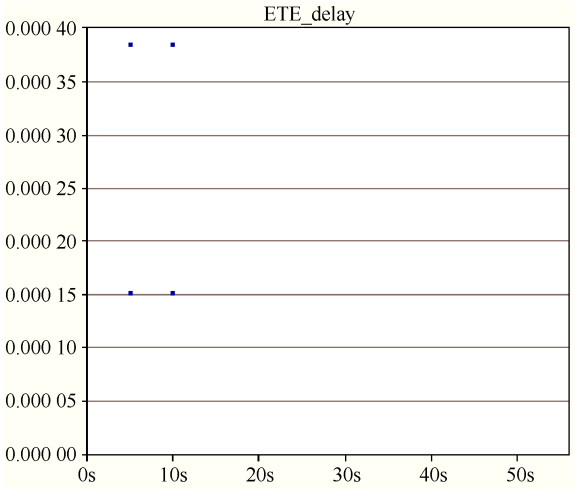

图 8 端到端延时仿真结果

以第 5 s 为例分析仿真实验 2 中仿真网络的端到端延时。在第 5 s 时，节点 Node1、Node2 同时向总线节点 Node3 发送数据。Node1 的优先级高于 Node2，因此在总线仲裁过程中 Node1 胜出，Node2 仲裁失败。Node2 会在 Node1 发送完数据后再向 Node3 发送数据。在仿真实验 2 中，Node1 与 Node3 之间端到端延时的计算方法与仿真实验 1 相同，即

$$t_{13} = 0.000\,15 + 17.34\times10^{-8} + 0.000\,002 \approx 0.000\,155\,2\ \text{s}$$

Node2 与 Node3 的数据传输延时为

$$t_a = 75/500\,000 = 0.000\,15\ \text{s}$$

数据传播延时为

$$t_b = 16.51/(0.65\times3\times10^8) \approx 8.467\times10^{-8}\ \text{s}$$

其他延时为

$$t_c = 0.000\,2 + (0 + 0.000\,02)/2 + 0.000\,006 + 0.000\,002 = 0.000\,218\ \text{s}$$

其中，前两项为随机退避时间的均值，第 3 项为 6 us 的帧间空隙，最后一项为 2 us 的仲裁数据包接收时间。综上，Node2 到 Node3 的总延时为

$$t_{23} = t_a + t_b + t_c \approx 0.000\,368\ \text{s}$$

该计算结果与图 8 所示的仿真实验结果一致。

第 10 s 时的端到端延时分析与以上分析相同。综上所述，仿真实验 2 的理论计算结果与图 8 所示的端到端延时仿真结果一致。

3.4 仿真实验 3 及结果分析

仿真设置：总线节点 Node1，Node2，Node4，Node5 和 Node6 按照泊松分布产生数据包，并向总线节点 Node3 发送。运行仿真，可以得到如图 9 所示的端到端延时仿真结果。为了清晰起见，放大第 30 s 附近的细节，示于图 10。如前所述，本文所有实验中节点的优先级设置为从 Node1 到 Node6 依次降低。在仿真实验 3 中，数据包的产生时间间隔服从均值为 1 s 的泊松分布。采用 ODB 单步调试和 Visual C++联合调试等方法，我们发现在第 30 s 时有 7 个数据包同时访问总线。将图 10 中的 7 个点自下而上依次标记为序号 1~7。表 2 给出了第 30 s 时各个点所对应的数据包的产生源节点以及所经历的延时等信息。

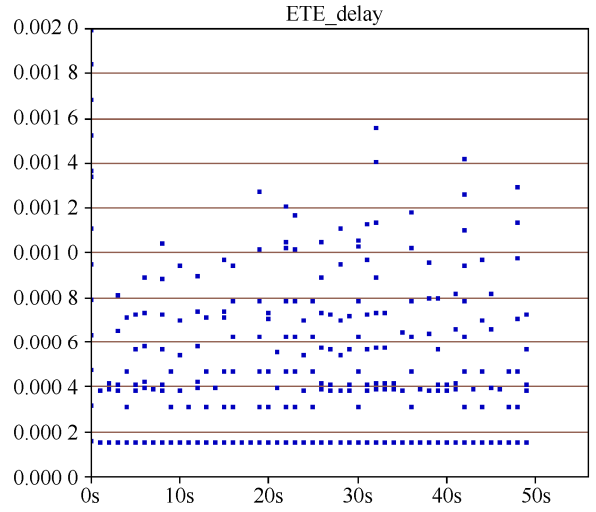

图 9　端到端延时仿真结果

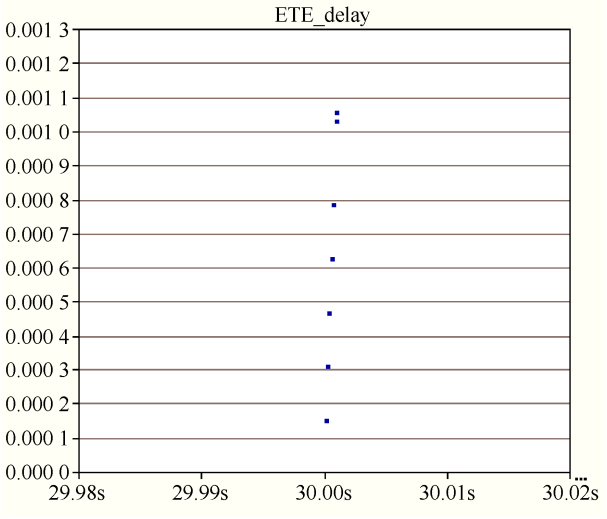

图 10　第 30 s 时的端到端延时仿真结果

表 2　第 30 s 时各数据包详细信息

点序号	源节点	优先级	产生时间/s	Node3 接收时间/s	ETE 延时/s
1	Node1	1	30	30.000 152 17	0.000 152 17
2	Node1	1	30	30.000 310 174	0.000 310 174
3	Node2	2	30	30.000 468 174	0.000 468 174
4	Node2	2	30	30.000 626 17	0.000 626 17
5	Node4	4	30	30.000 784 82	0.000 784 82
6	Node5	5	30	30.001 030 098	0.001 030 098
7	Node6	6	30	30.001 056 098	0.001 056 098

表 3 给出了图 10 中端到端延时 3 个组成部分(即传输延时 t_a，传播延时 t_b，其他延时 t_c)的理论计算结果。在仿真实验中，每个节点发送的数据包的长度均为 75 bit，因此每个数据包的传输延时 t_a 都是 0.000 15 s；各节点到 Node3 距离的不同使得数据传播延时存在差异，如表 2 第 4 列 t_b 所示；总延时的差别主要由其他延时 t_c 造成。具体地说，由于优先级不同，Node1 将优先接入网络，然后顺次接入 Node2，Node4，Node5，Node6。低优先级的节点竞争失败后，需要随机退避一段时间后再次参与总线竞争。需要说明的是，在表 3 中应用公式(1)计算退避时间的理论值时，我们采用了在取值范围内随机生成退避时间的方法。由于随机数的生成机制不同，我们的计算结果和 OPNET 仿真结果可能存在微小差异。

表 3　图 10 中端到端延时的具体计算结果

序号	ETE 延时/s	t_a/s	t_b/s	t_c/s
1	0.000 152 173	0.000 15	17.34×10^{-8}	0.000 002
2	0.000 310 346	0.000 15	17.34×10^{-8}	0.000 160 173
3	0.000 468 431	0.000 15	8.467×10^{-8}	0.000 318 346
4	0.000 626 174	0.000 15	8.467×10^{-8}	0.000 476 089
5	0.000 784 499	0.000 15	32.49×10^{-8}	0.000 634 174
6	0.001 022 418	0.000 15	41.82×10^{-8}	0.000 872
7	0.001 190 508	0.000 15	50.79×10^{-8}	0.001 040 0

通过以上仿真实验及仿真分析结果可以得知，本文建立的仿真模型实现了数据包按照优先级接入总线，同时也很好地模拟了实际物理网络中 MAC 子层处理随机数据包冲突的情况。

4　结论

在 OPNET 仿真环境下建立了 CAN 总线网络的进程模型、节点模型和网络模型，特别是完成了 CAN 总线 MAC 子层比特仲裁机制的建模。仿真实验与理论分析的一致性表明本文建立的模型很好地模拟了 CAN 总线的 CSMA/BA 仲裁机制。我们所建立的 MAC 层有限状态机具有复杂度低，状态数少，跳转逻辑清晰等特点，这种建模思路可供相关仿真工作者参考。同时本文建立的 LLC 模块和 MAC 模块可供研究人员即拿即用，减少了开发下层协议的工作量，使他们可以专注于应用层协议的设计，这对实际工程应用具有非常大的帮助。

DOI: 10.16182/j.issn1004731x.joss.201611007

第 28 卷第 11 期 系统仿真学报 Vol. 28 No. 11
2016 年 11 月 Journal of System Simulation Nov., 2016

参考文献:

[1] Andrew S Tanenbaum. 计算机网络 [M]. 熊桂喜, 王小虎译. 北京: 清华大学出版社, 1996: 184-192.

[2] Robert Bosch GmbH. CAN Specification Version2.0 [S], 1991.

[3] Kotzian J, Srovnal V. CAN based distributed control system modeling using UML [C]// 2003 IEEE International Conference on Industrial Technology. USA: IEEE, 2003: 1012-1017.

[4] 宋小庆, 任维彬, 陈克伟, 等. 基于有色 Petri 网的 CAN 总线仿真与性能分析 [J]. 装甲兵工程学院学报, 2011, 25(1): 75-78.

[5] 周杨. 基于 Petri 网的 CAN 总线性能分析与评估 [J]. 哈尔滨理工大学学报, 2009, 14(3): 47-50.

[6] 刘辉, 谭延亮, 焦铬. CAN 总线仿真环境的构建及在教学中的应用 [J]. 计算机时代, 2014 (6): 44-46.

[7] 曹剑馨, 梁庚. CAN 总线通信控制协议的仿真与性能分析 [J]. 中兴通信技术, 2014, 20(3): 48-51.

[8] Pinho L M, Vasques F. Reliable real-time communication in CAN networks [J]. IEEE Transactions on Computers (S0018-9340), 2003, 52(12): 1594-1607.

[9] 罗禹贡, 董珂, 冯能莲, 等. 基于 OPNET 的混合动力电动车 CAN 建模与仿真 [J]. 清华大学学报(自然科学版), 2005, 45(5): 689-692.

[10] 刘珩, 杨杰, 安建平, 等. 网络仿真技术在 CAN 总线协议分析中的应用 [J]. 计算机仿真, 2003, 20(增 1): 397-400.

[11] 李晓汀, 丁凡, 熊华钢. 基于 OPNET 的 CAN 网络建模与仿真 [J]. 北京航空航天大学学报, 2009, 35(3): 284-346.

[12] 刘明芹, 付东翔, 王亚刚. 基于 OPNET 的 CAN 总线实时性的仿真与分析 [J]. 通信技术, 2014, 47(3): 281-285.

[13] 徐超, 李正平, 汪长勤. 基于 CSMA/CD 的 CAN 总线访问的建模与仿真的研究 [J]. 仪器仪表学报, 2008, 29(4): 866-869.

(上接第 2691 页)

[6] Akay B, Karaboga D. A modified artificial bee colony algorithm for real-parameter optimization [J]. Information Sciences (S0020-0255), 2012, 192(6): 120-142.

[7] Brajevic I, Tuba M. An upgraded artificial bee colony (ABC) algorithm for constrained optimization problems [J]. Journal of Intelligent Manufacturing (S0956-5515), 2013, 24(4): 729-740

[8] Vitorino L N, Ribeiro S F, Bastos-Filho C J A. A mechanism based on Artificial Bee Colony to generate diversity in Particle Swarm Optimization [J]. Neurocomputing (S0925-2312), 2015, 148: 39-45.

[9] 暴励, 曾建潮. 自适应搜索空间的混沌蜂群算法 [J]. 计算机应用研究, 2010, 27(4): 1331-1335. (Bao Li, Zeng Jianchao. Self-adapting search space chaos-artificial bee colony algorithm [J]. Application Research of Computers, 2010, 27(4): 1331-1335.)

[10] 高卫峰, 刘三阳, 黄玲玲. 受启发的人工蜂群算法在全局优化问题中的应用 [J]. 电子学报, 2012, 40(12): 2396-2403. (Gao Weifeng, Liu Sanyang, Huang Lingling. Inspired Artificial Bee Colony Algorithm for Global Optimization Problems [J]. Chinese Journal of Electronics, 2012, 40(12): 2396-2403.)

[11] Karaboga D, Beyaza G. A quick artificial bee colony (qABC) algorithm and its performance on optimization problems [J]. Applied Soft Computing (S1568-4946), 2014, 23(5): 227-238.

[12] 毕晓君, 王艳娇. 加速收敛的人工蜂群算法 [J]. 系统工程与电子技术, 2011, 33(12): 2755-2761. (Bi Xiaojun, Wang Yanjiao. Artificial bee colony algorithm with fast convergence [J]. Journal of systems Engineering and electronics, 2011, 33(12): 2755-2761.)

[13] Rahnamayan S, Tizhoosh H R, SALAMA M M A. Opposition-Based differential evolution [J]. IEEE Transactions on Evolutionary Computation (S1089-778X), 2008, 12(1): 64-79.

[14] Rahnamayan S, Tizhoosh H R, Salama M M A. Opposition versus randomness in soft computing techniques [J]. Applied Soft Computing (S1568-4946), 2008, 8(2): 906-918.

[15] 韩江洪, 李正荣, 魏振春. 一种自适应粒子群优化算法及其仿真研究 [J]. 系统仿真学报, 2006, 18(10): 2969-2971. (Han Jianghong, Li Zhengrong, Wei Zhenchun. Adaptive Particle Swarm Optimization Algorithm and Simulation [J]. Journal of System Simulation (S1004-731X), 2006, 18(10): 2969-2971.)

[16] Gandomi A, Yang Xin-She, Alavi A. Cuckoo search algorithm: a metaheuristic approach to solve structural optimization problems [J]. Engineering with Computers (S1435-5663), 2013, 29(1): 17-35.