doi:10.11887/j.cn.201903008 http://journal.nudt.edu.cn

基于扩展标记变迁模型的时钟同步协议正确性验证*

曲国远[1],徐晓飞[1],刘威廷[2,3,4],王沁煜[2,3,4],贺 飞[2,3,4]
(1. 中国航空无线电电子研究所,上海 200233;2. 清华大学 软件学院,北京 100084;
3. 北京信息科学与技术国家研究中心,北京 100084;4. 信息系统安全教育部重点实验室,北京 100084)

摘 要:时钟同步协议是时间触发网络的一个重要组成部分,是时间触发网络实时性和确定性的关键。本文基于扩展标记变迁模型对时钟同步协议进行建模,基于模型检测方法对协议是否满足正确性属性进行验证。验证结果证明了在不同启动场景下时钟同步网络协议的正确性,也表明了扩展标记变迁模型对于协议验证的有效性。

关键词:形式化方法;协议验证;模型检测
中图分类号:TP331.2 **文献标志码**:A **文章编号**:1001-2486(2019)03-042-08

Correctness verification of clock synchronization protocol via extended labeled transition system models

QU Guoyuan[1], *XU Xiaofei*[1], *LIU Weiting*[2,3,4], *WANG Qinyu*[2,3,4], *HE Fei*[2,3,4]
(1. China Aeronautical Radio Electronics Research Institute, Shanghai 200233, China;
2. School of Software, Tsinghua University, Beijing 100084, China;
3. Beijing National Research Center for Information Science and Technology, Beijing 100084, China;
4. Key Laboratory for Information System Security, Ministry of Education, Beijing 100084, China)

Abstract: Clock synchronization protocol, as an important component of the time-triggered networks, is the key to the instantaneity and certainty of the time-triggered networks. The modeling and verification of clock synchronization protocol were studied. The protocol's behaviors were modeled by the extended labeled transition systems, and its correctness was verified by model checking technique. The verification results certified the correctness of this protocol even under different startup scenarios. Experimental results also show the effectiveness of extended labeled transition systems in protocol verification.

Keywords: formal methods; protocol verification; model checking

时间触发网络[1-3]被广泛应用于航空航天、轨道交通、军事装备等实时性要求高的系统中。在时间触发网络中,所有网络消息都由预先定义的时刻进行触发。时间触发网络能够保证网络中每一个终端在某一个时刻最多只能发送一条消息。相比于传统的基于事件触发的网络,时间触发网络具有更好的通信实时性和确定性。

时间触发网络必须实现网络中所有设备的时钟同步。考虑每一个网络设备都有自己的本地时钟,不同设备的本地时钟之间存在偏移,并且这个偏移在经过一定时间的累积后,可达到一个不可忽视的程度。时间触发网络利用时钟同步协议建立一个全局统一的系统时钟,并且基于该系统时钟调度网络中的所有通信。

时间触发网络在我国航空航天、轨道交通等领域有很好的应用前景。在将该网络引入这些应用领域前,需要进行适应性的补充与修改。如何证明修改后的时钟同步协议仍然具有正确性成为制约时间触发网络在这些系统中进行应用的关键。

本文以扩展标记变迁模型描述协议行为,以时序逻辑刻画正确性属性,基于模型检测工具Beagle[4],对协议是否满足属性进行形式化验证。

1 扩展标记变迁模型

模型检测[5-6]是一种用于有限状态并发系统的形式化验证技术,在电路验证、通信协议验证、软件验证中已经取得了巨大的成功。给定系统 M

* 收稿日期:2018-03-27
基金项目:航空科学基金资助项目(2015ZC15001);国家部委基金资助项目(3030603);国家自然科学基金资助项目(61672310,61272001,91218302)
作者简介:曲国远(1982—),男,黑龙江七台河人,高级工程师,硕士,E-mail:qu_guoyuan@careri.com

和正确性属性 P,模型检测通过搜索 M 的所有状态(或者所有执行路径)来判定 M 是否满足 P。

在给定网络节点的情况下,时间触发网络[1]就是一个典型的有限状态并发系统。标记变迁(Labeled Transition System, LTS)模型是一种经典的、针对有限状态并发系统的形式化模型。

定义 1(LTS 模型)　令 AP 为原子命题集合,一个标记变迁模型是一个四元组 $M=(\Sigma,S,S_0,R)$,其中 Σ 是标记的有限集,S 是状态的有限集,$S_0\subseteq S$ 为初始状态的集合,$R\in S\times\Sigma\times S$ 是变迁关系。

设 $s,t\in S$ 为系统的两个状态,$\alpha\in\Sigma$ 为系统的一个标记,如果$\langle s,\alpha,t\rangle\in R$,则称之为系统的一条变迁。设 $r\in R$ 为系统 M 的一条变迁,以 $l(r)$ 表示 r 上对应的标记。

标记变迁模型中 S 为系统中所有状态的集合。当系统状态空间较大时,难以直接使用 LTS 进行建模。扩展标记变迁(Extended Labeled Transition System, ELTS)模型在 LTS 的基础上扩展了变量,能够更方便地描述规模较大的系统。

定义 2(ELTS 模型)　令 AP 为原子命题集合,一个扩展标记变迁模型是一个七元组 $M=(\Sigma,X,S,S_0,R,\tau,\eta)$,其中$(\Sigma,S,S_0,R)$是一个 LTS 模型;$X$ 是变量的有限集;τ 为每条变迁 $r\in R$ 定义了一个被称为守护条件的表达式 $g(r)$;η 为每条变迁 $r\in R$ 定义了一个对 X 中变量进行赋值的动作序列 $a(r)$。

ELTS 模型的状态是对 $X\cup L$ 的一组赋值,称 ELTS 中的 L 为控制状态集合。设 $r\in R$ 为系统的一条变迁,r 常被表示为:

$$l(r)[g(r)]:a(r)$$

图 1 给出了一个信号灯控制系统的 ELTS 模型。该模型中含有 2 个控制状态,即 Red 和 Green,分别代表红色和绿色信号灯。模型中含有 1 个变量 x,代表时间。为简单起见,图中守护条件为 true 时可以省略,动作序列为空时也可以省略。该信号灯控制系统的初始控制状态为 Red,从 Red 到 Green 的边

$$t_3[x=15]/x++$$

表示该变迁在 x 的值等于 15 时可以被触发,并且执行该变迁会导致 x 的值加 1。其他边可以类似地进行分析。

定义 3(简单安全属性验证)　给定一个模型 M 和一条属性 P,如果在 M 的所有状态上 P 都成立,则称 M 满足 P(记作 $M\models P$)。

以图 1 为例,假设属性为 $x<60$,显然该属性在 M 的所有状态上都成立,所以 $M\models P$ 成立。

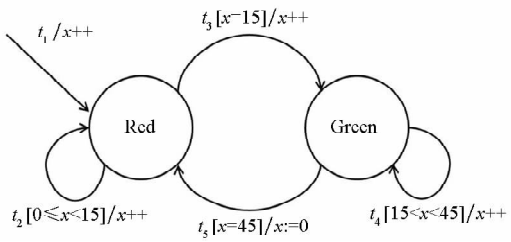


图 1　ELTS 示例图

Fig. 1　An ELTS example

2　时钟同步协议

时间触发网络[1]按照事先设定的时刻来协调网络中的数据交换。在时间触发网络中,不同网络终端的本地时钟之间存在偏移,并且这个偏移会随着时间累积。为实现时间触发网络,必须借助时钟同步协议,将网络中所有设备的时钟误差控制在一定范围内。

时间触发网络[1]中的设备主要包括交换机和终端。在时间触发网络中,一般称终端设备为同步主(Synchronize Master, SM),交换机为压缩主(Compression Master, CM)。图 2 给出了一个包括 4 个同步主和 1 个压缩主的时间触发网络,其中实线代表设备之间的物理连接。

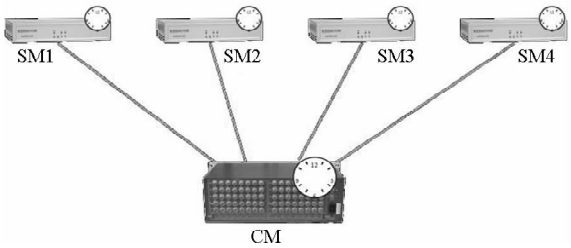


图 2　时间触发网络示意图

Fig. 2　The time-triggered network

时钟同步协议的主要过程是 CM 收集 SM 的本地时钟,计算全局时钟,然后再分发给 SM 的过程。CM 和 SM 之间的数据交换主要依赖协议控制帧(Protocol Control Frame, PCF)进行。处于空闲状态的 CM 接收到 SM 的 PCF 后,进入收集状态。处于收集状态的 CM 开启一个时间窗口,并在该时间窗口内等待后续 PCF。如果 CM 在时间窗口内接收到一个新的 PCF,将向后延长一个窗口长度的等待时间。当 CM 收集到的 PCF 数量达到阈值或在一个窗口时间内没有收集到任何新的 PCF,CM 进入到计算状态。在计算状态下,CM 对收集到的所有时钟进行计算,并将计算结果分发给所有 SM,供 SM 进行时钟同步。

时间触发网络支持双 CM 备份。一个包含 2 个 CM 的时间触发网如图 3 所示。在双 CM 备份网络中,为防止主 CM 故障导致网络结构瘫痪,设置备用 CM 用于故障恢复。所有 SM 会同时向两个 CM 发送 PCF,两个 CM 独立地运行同步协议,将分别计算全局时钟并发送给所有 SM。SM 在收到来自两个 CM 的全局时钟时,有多种策略,例如只采用主 CM 的时钟、采用两个 CM 时钟的平均值等。本文假定 SM 采用前一个策略,即只使用主 CM 时钟。

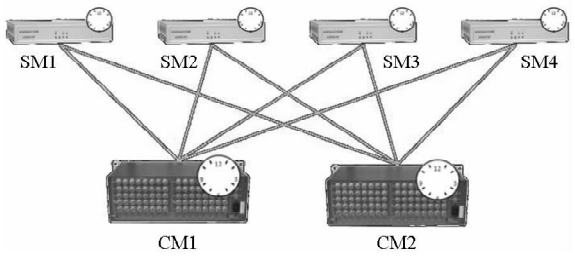


图 3 双 CM 冗余时间触发网络

Fig. 3 A time-triggered network with double CMs

将时钟同步协议应用到航空航天等实际系统中时,系统启动是相对容易出错的时刻。本文讨论了多种启动模式下时钟同步协议的正确性。

1)单 CM 启动模式:在普通启动模式下,系统中仅含一台交换机(即 CM)。这种启动方式比较简单。在初始条件下,CM 上电后处于非同步状态。主管理器正常启动后启动 PCF 发送,CM 收到 PCF 后进入同步状态,并转发 PCF。各个 SM 上网后,同步到 CM 转发的 PCF 时间,周期性地发送 PCF,参与时间同步过程。

2)双 CM 同步启动模式:在该模式下,系统中包含两台 CM,并且两台 CM 同步启动。首先双 CM 上电后处于非同步状态,主管理器争权成功,通过双通道启动 PCF 发送,两个 CM 收到 PCF 后,都进入同步状态,并转发 PCF。各个 SM 上网后,从工作网络接收 PCF,并同步时钟,周期性地通过双通道发送 PCF,参与时间同步过程。

3)双 CM 异步启动模式:双 CM 异步启动的工作模式允许先在单 CM 的情况下工作一段时间后再将另一个 CM 加入时间同步网络。与同步启动模式相比,异步启动更灵活且适用性更广。首先,单 CM 上电,系统进入时间同步状态。另一个 CM 上电,进入非同步状态。各个 SM 节点通过双通道发送 PCF 给双 CM,CM 进入同步状态,并同步到接收的时钟,用第一个接收到的时钟设置本地时钟,可以和接收到几个时钟无关。各个 SM 周期性地发送 PCF 参与同步,各个 SM 依旧同步到工作网络。

3 协议建模

基于扩展标记变迁模型[4]对时钟同步协议进行建模,所建模型主要包含 4 个模块,分别是压缩主 CM 模块、同步主 SM 模块、触发器 TRIGGER 模块和监视器 MONITOR 模块。

3.1 CM 模块

CM 模块包含五个控制状态,分别为:初始状态 START、空闲状态 IDLE、收集状态 COLLECTION、延迟状态 DELAY 和计算状态 CALC。CM 模块的 ELTS 模型如图 4 所示。为方便阅读和理解,在图 4 中只标出了每条迁移边对应的同步标记,守护条件和动作序列都被隐去。

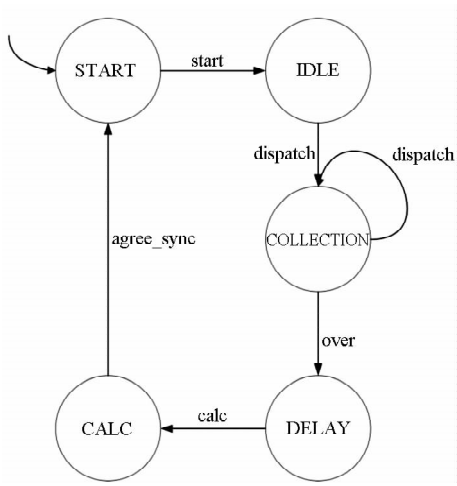


图 4 CM 模型示意图

Fig. 4 The CM model

处于 START 状态的 CM 接收到 start 同步标签后进入 IDLE 状态,此时 CM 会等待 SM 发送 PCF。当收到第一个 PCF 后,CM 进入 COLLECTION 状态。处于 COLLECTION 的 CM 循环等待不同的 SM 发来的 PCF,直到收到的 PCF 数量等于所有 SM 的数量。之后 CM 进入 DELAY 状态,延迟一段时间并通过时间同步算法计算出统一时钟,然后进入 CALC 状态,通过 calc 同步标签对各个 SM 进行时间同步。最后在转移回 START 状态之前,CM 通过 agree_sync 同步标签通知 MONITOR 模块已经完成时间同步,从而模拟分发统一时间的 PCF,等待下一个循环周期的开始。

下面定义图 4 中每一条迁移边的守护条件和动作序列。首先引入下列变量:

```
int clock;
int current_reading_index;
int SM_amount;
int delay_timeout;
```

其中,clock 是 CM 的本地时钟,current_reading_index 是 CM 当前收到的 PCF 数量,SM_amount 是系统中 SM 数量,delay_timeout 代表了 CM 计算统一时间时需要延迟的时间长度。

假设 r 是模型中从控制状态 s_0 到 s_1 的一条迁移边,$l(r)$,$g(r)$和 $a(r)$分别表示 r 的标记、守护条件和动作序列。本文使用下面的文法表示 r:

```
from s0 to s1 on l(r) provided g(r) do a(r);
```

根据上面的文法,CM 模块从 START 状态到 IDLE 状态的变迁可以表示为:

```
from START to IDLE on start do {
    clock = 0;
    SM_amount = 5;
    current_reading_index = 0;
    delay_timeout = 5;
};
```

这条变迁的守护条件为 true,执行该变迁将对 CM 中的各个变量进行初始化操作。

CM 模块从 IDLE 状态到 COLLECTION 状态的变迁表示为:

```
from IDLE to COLLECTION on dispatch_i
provided (current_reading_index = = 0)
do {
    current_reading_index = 1;
};
```

该变迁必须在 current_reading_index = =0 的条件下由第 i 个 SM 发出的 dispatch$_i$ 触发,执行该变迁将赋值 current_reading_index = 1。

CM 模块从 COLLECTION 状态到 COLLECTION 状态的变迁表示为:

```
from COLLECTION to COLLECTION
on dispatch_i
provided
    current_reading_index < SM_amount
do {
    current_reading_index =
        current_reading_index + 1;
};
```

这是 CM 收集不同 SM 发出的 dispatch$_i$ 时执行的变迁,该变迁反复执行,直到 CM 收到的 PCF 数量等于系统中 SM 的总数,每次执行导致 current_reading_index 的计数加 1。

DELAY 状态模拟了 CM 收集完所有 SM 发来的本地时钟后进行全局时钟计算所产生的延迟。这里的 delay_timeout 为预先设定的一个延迟时间。CM 模块从 COLLECTION 状态到 DELAY 状态,从 DELAY 状态到 CALC 状态的变迁的守护条件均为 true,动作序列都为空,其文法表示省略。模型中定义这两条变迁的目的主要是为了与其他模块同步。

CM 模块从 CALC 状态到 START 状态的变迁表示为:

```
from CALC to START on agree_sync
provided delay_timeout = = 0 do {
    current_reading_index = 0;
    delay_timeout = 0;
    clock = clock + 1;
};
```

该变迁主要用于各个变量值的重置,以及利用 agree_sync 标签与 MONOITOR 模块同步,表示 CM 的一个计算周期结束。

3.2　SM 模块

SM 模块的行为与网络中包含的 CM 个数相关。下面分情况讨论单 CM 网络和双 CM 网络中 SM 模块的形式化模型。

3.2.1　单 CM 网络的 SM 模块

单 CM 的 SM 模型包含四个控制状态,分别是:初始状态 START、空闲状态 IDLE、分发状态 DISPATCH 和同步状态 SYNC。其 ELTS 模型如图 5 所示。同样,在图 5 中只标出了每条迁移边对应的同步标记,隐去了守护条件和动作序列。

处于 START 状态的 SM 模块收到 start 标签后进入 IDLE 状态。当 SM 的本地时钟到达预定的发送时刻,SM 会发送一个 PCF 给 CM,然后进入 DISPATCH 状态。SM 在进入 DIAPATCH 状态后会等待,直到 CM 计算出全局时钟。最后,SM 收到 CM 计算的全局时钟,进入 SYNC 状态完成时钟同步。

为详细定义 SM 模块的每条迁移,需要引入下列变量:

```
int clock;
int dispatch_timeout;
int deviation;
```

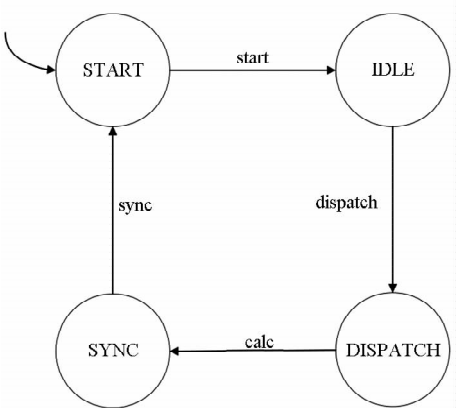


图5 单CM的SM模型

Fig. 5 SM model in a single-CM network

其中,clock是SM的本地时钟,dispatch_timeout是预先设定的发送时间点,deviation则是为了模拟现实的晶振误差和传输延迟所设定的一个时间偏移。模型中会设定deviation为一个0到5范围内的随机数。

SM模块从START状态到IDLE状态的变迁表示为:

```
from START to IDLE on start do {
    clock = dispatch_timeout + deviation;
    dispatch_timeout = 0;
    deviation = 0;
};
```

这个迁移由start标签触发,会将SM的本地时钟置成dispatch_timeout与deviation的和,表示CM收到的SM的本地时钟存在时间偏移。

SM模块从IDLE状态转移到DISPATCH状态的变迁表示为:

```
from IDLE to DISPATCH on dispatch_i provided
deviation = =0 && dispatch_timeout = = 0;
```

该迁移需要满足deviation和dispatch_timeout的值都为零的守护条件才能触发,执行此迁移将发送第i个SM的$dispatch_i$标签给CM模块。

SM模块从DISPATCH状态到SYNC状态的变迁表示为:

```
from DISPATCH to SYNC on calc do {
    clock = CM.clock;
};
```

当处于DISPATCH状态的SM收到CM模块发出的calc标签时,表示CM已经完成了时间同步,此时需要将SM的本地时钟更新成CM的全局时钟,并转移至SYNC状态。

SM模块从SYNC状态到START状态的变迁表示为:

```
from SYNC to START on sync;
```

当SM处于SYNC状态时代表了SM的本地时钟已经处于了同步状态,需要等待所有其他SM都到达同步状态后一起返回START状态完成一个同步周期。

3.2.2 双CM网络的SM模块

双CM网络中SM模块的ELTS模型如图6所示。相比于单CM网络的模型,该SM模型主要多了一个DISPATCH状态,用来记录当前SM已经给某个CM发送了本地时钟的情况。注意一个SM对于不同的CM的传输延迟可能不同,所以在SM模型中,对于不同CM其对应的deviation的值也不同,触发dispatch标签的时间也不一致。双CM的模型会根据触发的dispatch标签编号进入对应的DISPATCH状态,然后等待本地时钟值到达触发另一个dispatch标签时进入END状态,代表SM模块已经发送。

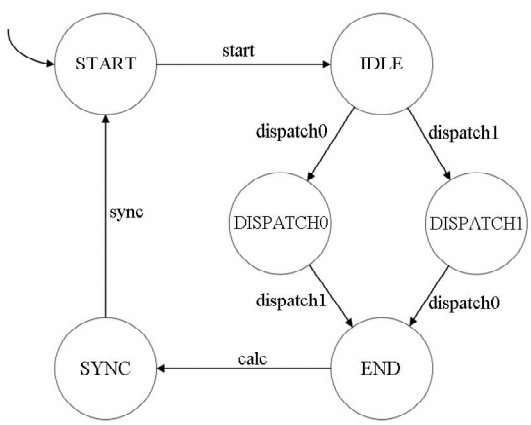


图6 双CM的SM模型

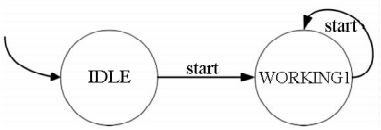
Fig. 6 SM model in a double-CMs network

3.3 触发器模块

为了实现不同的启动模式,本文设计了一个触发器模块TRIGGER。基本思想是通过TRIGGER给当前应启动的SM和CM发送启动标签来模拟不同的启动场景。

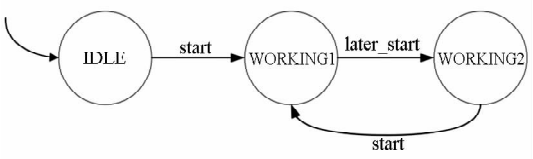
同步和异步启动模式下TRIGGER模块的ELTS模型分别如图7(a)和图7(b)所示。触发器一开始处在IDLE状态并等待启动。对于同步启动模式(无论单CM或双CM),触发器只需向所有的SM和CM以及MONITOR发送一次start标签即可。对于异步启动模式,触发器先转移至WORKING1状态,并向需要先启动的模块发送

start 标签,然后再向后启动的模块发送 later_start 标签去触发它们启动。在完成一个完整的同步周期后,通过一个 start 标签回到 WORKING1 状态,重新进入启动状态等待新一个同步周期的开始。



(a) 同步启动模式

(a) Synchronous model



(b) 异步启动模式

(b) Asynchronous mode

图 7　TRIGGER 模型

Fig. 7　The TRIGGER model

3.4　监视器模块

为了更方便地对协议模型进行验证,设计一个监视器模块 MONITOR。MONITOR 监测系统中所有 SM 的本地时钟,并判断其同步精度是否在预先设定的范围之内。

监视器模块的 ELTS 模型如图 8 所示。监视器模块包含四个控制状态,分别为: START, SMALL, BIG 和 BAD。START 为初始状态。SMALL 表示当前网络处于同步状态,各个 SM 本地时钟之间的差值应该较小(或者近似相等),否则监视器进入 BAD 状态。BIG 表示当前网络正在运行时钟同步协议,各个 SM 本地时钟之间的差值可能较大,但也应该在一个预先设定的范围之内。如果超出该预先设定的范围,监视器也进入 BAD 状态。

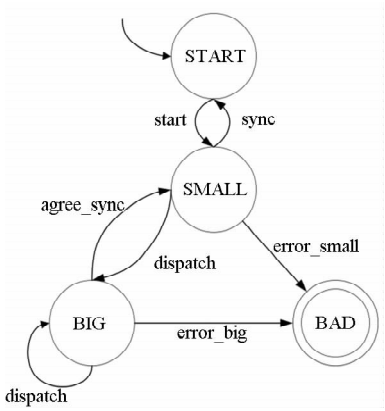


图 8　MONITOR 模型

Fig. 8　The MONITOR model

处于 START 状态的监视器收到 start 标签后迁移到 SMALL 状态。从 SMALL 状态转移到 BIG 状态需要等待任意一个 SM 发出 dispatch_i 标签来触发,代表时间同步算法已经启动。TMP 状态是为处理双 CM 模型所设计的临时状态,单 CM 模型中没有这个状态。处于 BIG 状态的 MONITOR 收到 CM0 发出的 agree_sync_0 或 CM1 发出的 agree_sync_1 标签后转移到 TMP 状态,继续等待另一个 CM 发来的 agree_sync 标签,然后转移至 SMALL 状态。

MONITOR 模块中变量声明为:

```
int dispatch_num;
int max_drift;
```

其中,dispatch_num 是当前已经发出 PCF 的 SM 数量,max_drift 是预先设定的用来判断各个 SM 的本地时钟时间差是否过大的阈值。

MONITOR 模块从 START 状态到 SMALL 状态的变迁说明为:

```
from START to SMALL on start do {
    dispatch_num = 0;
    max_drift = 10;
};
```

主要执行对 MONITOR 变量的初始化。

MONITOR 模块从 SMALL 状态到 BIG 状态的变迁说明为:

```
from SMALL to BIG on dispatch_i provided
dispatch_num = = 0 do {
    dispatch_num = dispatch_num + 1;
};
```

处于 SMALL 状态的 MONITOR 收到任一个 SM 发出的 dispatch_i 标签后触发该变迁,并将 dispatch_num 的值加 1。

MONITOR 模块从 BIG 状态到 BIG 状态的变迁说明为:

```
from BIG to BIG on dispatch_i provided dispatch_num > 0
and dispatch_num < CM_amount * SM_amount do {
    dispatch_num = dispatch_num + 1;
};
```

当 dispatch_num 大于 0 且小于 CM 数量乘以 SM 数量时,每个 SM 发出的 dispatch_i 都会触发这个变迁并使 dispatch_num 加 1。

MONITOR 模块从 SMALL 状态到 BAD 状态的变迁说明为:

```
from SMALL to BAD provided! (SM_i.clock == SM_j.clock);
```

其中,“!”表示否定。处于 SMALL 状态的 MONITOR 会检测每两个 SM 之间的本地时钟是否相同。如果存在不同的时钟则会进入 BAD 状态。

MONITOR 模块从 BIG 状态到 BAD 状态的变迁说明为:

```
from BIG to BAD provided (SM_i.clock > SM_j.clock + max_drift) && (dispatch_num == SM_amount);
```

处于 BIG 状态的 MONITOR 会检测每两个 SM 之间的本地时钟之差是否超过设定的阈值 max_drift。如果存在两个 SM 之间的时钟差超过 max_drift 且收到 PCF 的 SM 数量 dispatch_num 等于 SM 总数 SM_amount,则会进入 BAD 状态。

当验证包含两个 CM 的系统时需要在 BIG 状态到 SMALL 状态的中间加入 TMP 状态用来接收两个不同 CM 的时钟同步信号。

MONITOR 模块从 BIG 状态到 TMP 状态,以及从 TMP 状态到 SMALL 状态的变迁说明为:

```
from BIG to TMP on agree_sync_i do {
    dispatch_num = 0;
};
from TMP to SMALL on agree_sync_i;
```

在双 CM 的情况下,MONITOR 需要收到两个 CM 完成同步分别发来的标签 agree_sync_0 和 agree_sync_1,才代表整个时钟同步过程结束,因此需要一个中间状态 TMP 来缓存收到一个 agree_sync_*i* 标签的情况。

4 协议验证

4.1 验证属性

为保证时钟同步协议模型的正确性,验证下列两条属性。

4.1.1 时钟误差属性

时钟同步协议的基本目标是保证任何时刻所有 SM 本地时钟的误差不会超出一个阈值。根据前面对监视器模块的建模,这一属性可以归结为 MONITOR 模块永远不会到达 BAD 状态,即

INVARSPEC ! (MONITOR.location == BAD)

4.1.2 同步完成属性

除了时钟误差属性外,还需要保证 SM 和 CM 运行的时钟同步算法最后都能正常终止,到达终止状态。采用反证法,分别验证

INVARSPEC ! (CM.location == CALC)

INVARSPEC ! (SM.location == SYNC)

即 CM 永远不会到达 CALC 终止状态,SM 永远不会到达最终的 SYNC 状态。如果针对上面两条属性的验证都返回 false,说明 CM 和 SM 可以到达终止状态,同步完成属性成立。反之,对于任何一条属性的验证返回 true,说明对应模块永远不能到达终止状态,同步算法不终止。

4.2 验证环境

本节对 4.1 小节提到的不同属性,在不同的 SM 和 CM 数量配置下进行了验证,试验环境为:Intel(R) Xeon(R) CPU E5 - 2620 v3 @ 2.40 GHz,32 056 M内存,操作系统为 Ubuntu 16.04.2 LTS。

实验采用 Beagle[4] 执行模型检测。Beagle 是清华大学软件学院开发的一个基于 ELTS 的模型检测工具。Beagle 工具中分别基于二叉决策图(Binary Decision Diagram, BDD)和可满足性(SATisfiability, SAT)实现了基本的模型检测方法,并在此基础上,提出了一系列针对构件化系统的高效模型检测算法[7-10]。

4.3 结果分析

对时钟同步协议是否满足时钟误差属性(以下简称 P1)和同步完成属性(以下简称 P2)进行验证。实验中,根据单 CM 和双 CM 的不同启动方式,以及 SM 的数量分别构建了时钟同步协议的模型,并执行验证。

验证结果如表 1 所示,其中第 1 列表示验证的属性,第 2 列和第 3 列表示网络中包含的 CM 和 SM 数量,第 4 列为启动模式(第 2 节中介绍的 3 种启动模式分别以 1,2,3 表示)。采用限界模型检测执行验证,第 5 列表示限界模型检测展开的步数,第 6 列表示验证所需的时间,第 7 列表示属性验证的结果。注意属性 P1 为 true、属性 P2 为 false 为期望的模型验证结果。

从表 1 中可以发现,在给定的网络结构下,时钟同步属性和同步完成属性的验证结果都与期望一致。因此时钟同步协议在这些网络结构下的正确性得到验证。另外,可以观察到,验证所需时间与模型展开步数正相关。这是因为当展开步数增加时,整个模型的规模变大,验证所需的时间因此也将增大。

表 1　实验结果

Tab. 1　Experimental results

属性	CM 数	SM 数	启动方式	步数	用时/s	结果
P1	1	5	1	7	0.15	true
P1	1	5	1	61	177.24	true
P1	2	5	2	15	48.45	true
P1	2	5	3	14	31.32	true
P2	1	5	1	7	0.12	false
P2	1	5	1	52	87.53	false
P2	2	5	2	15	58.07	false
P2	2	5	2	33	1420.48	false
P2	2	5	3	16	48.3	false
P2	2	5	3	35	1561.16	false

5　相关工作

模型检测在协议验证方面得到了广泛的应用,例如路由协议验证、密码协议验证、网络协议验证等。在文献[11]中,Roy 等采用概率时间自动机来验证在无线局域网 IEEE0802.11 协议中的介质访问控制层协议,采用确定路径压缩算法优化模型,减少协议中的冗余,以应对验证中的状态爆炸空间问题。在文献[12]中,Esparza 等验证了 Population protocol 的正确性,该协议是主要应用在传感器网络中的形式化模型,用于在不同设备之间获得一致的值,设备与设备之间通过交互以改变自身状态。在文献[13]中,He 等验证了发布订阅分布式协议的正确性。

Steiner 和 Dutertre 等对 TTEthernet 中的时钟同步协议标准进行了验证[14-16]。他们基于 SAL model checker 实现了对同步算法的正确性分析,并验证了在协议规定下,系统满足的同步精度范围。与他们的工作相比,本文采用了一种新的建模语言(即扩展标记变迁模型),建模过程说明了该语言对于时钟同步协议验证的有效性。特别地,本文所建立的模型中考虑了时间触发网络的不同启动模式。该特性是时间触发网络国际标准中没有规定的内容,但却是将时间触发网络引入到我国航空领域某系统的设计中所必须考虑的一个适配性问题。本文结果为该系统的设计提供了可信依据。

6　结论

本文给出了一种基于扩展标记变迁模型对时钟同步协议进行建模的方法,并基于模型检测工具对其正确性进行了验证。除了标准中规定的时钟同步协议外,还对多个启动场景下协议的正确性进行了分析与验证。验证结果证明了在复杂启动场景下时钟同步网络协议的正确性,也表明了扩展标记变迁模型对于协议验证的有效性。

参考文献(References)

[1] Time-triggered Ethernet: SAE AS6802[S]. SAE International, 2011.

[2] Steiner W. TTEthernet specification[R]. TTA Group, 2008.

[3] Kopetz H, Ademaj A, Grillinger P, et al. The time-triggered Ethernet (TTE) design[C]//Proceedings of 8th IEEE International Symposium on Object-oriented Real-time Distributed Computing, 2005.

[4] He F, Yin L Z, Wang D X, et al. Beagle user manual[R]. Beijing: Tsinghua University, 2013.

[5] Clarke E M, Grumberg O, Peled D A. Model checking[M]. UK: MIT Press, 1999.

[6] McMillan K L. Symbolic model checking[M]. USA: Springer US, 1993: 25-60.

[7] He F, Yin L Z, Wang B Y, et al. VCS: a verifier for component-based systems [C]//Proceedings of 11th International Symposium on Automated Technology for Verification and Analysis, 2013: 478-481.

[8] Yin L Z, He F, Gu M, et al. Clause replication and reuse in incremental temporal induction [C]//Proceedings of International Conference on Engineering of Complex Computer Systems, 2014.

[9] Yin L Z, He F, Gu M. Reusing search tree for incremental SAT solving of bounded model checking[C]//Proceedings of International Conference on Engineering of Complex Computer Systems, 2013: 85-92.

[10] Yin L Z, He F, Gu M. Optimizing the SAT decision ordering of bounded model checking by structural information[C]// Proceedings of 7th International Symposium on Theoretical Aspects of Software Engineering, 2013: 23-26.

[11] Roy A, Gopinath K. Improved probabilistic models for 802.11 protocol verification[C]//Proceedings of International Conference on Computer Aided Verification, 2005: 239-252.

[12] Esparza J, Ganty P, Leroux J, et al. Verification of population protocols [J]. Acta Informatica, 2017, 54: 191-215.

[13] He F, Baresi L, Ghezzi C, et al. Formal analysis of publish-subscribe systems by probabilistic timed automata [C]// Proceedings of 27th IFIP WG 6.1 International Conference on Formal Methods for Networked and Distributed Systems, 2007: 247-262.

[14] Steiner W, Dutertre B. SMT-based formal verification of a TTEthernet synchronization function[C]//Proceedings of 15th Workshop on Formal Methods for Industrial Critical Systems, 2010: 148-163.

[15] Steiner W, Dutertre B. Automated formal verification of the TTEthernet synchronization quality [C]//Proceedings of NASA Formal Methods Symposium, 2011: 375-390.

[16] Steiner W, Dutertre B. The TTEthernet synchronization protocols and their formal verification [J]. International Journal of Critical Computer-Based Systems, 2013, 4(3): 280-300.