doi:10.3969/j.issn.1672-4348.2018.03.010

停等式 ARQ 协议的 SPIN 模型检测

黄丽丽

(福建工程学院 生态环境与城市建设学院, 福建 福州 350118)

摘要: 介绍停等式 ARQ 协议的工作原理,并使用 Promela 对其进行建模,利用 SPIN 对所建模型进行检测,证明了所建模型具有停等式 ARQ 协议的性质。讨论对停等式 ARQ 协议进行攻击的方法,使用 Promela 语言对攻击者进行建模,并利用 SPIN 的图形界面工具 XSPIN 模拟了攻击过程,验证了攻击的有效性。

关键词: 停等式 ARQ; SPIN; Promela; 模型检测
中图分类号: TP393.08　　**文献标志码**: A　　**文章编号**: 1672-4348(2018)03-0253-06

Model checking of stop-and-wait ARQ via SPIN

HUANG Lili
(School of Ecological Environment and Urban Construction, Fujian University of Technology, Fuzhou 350118, China)

Abstract: The working principle of the stop-and-wait ARQ was introduced, whose model was established by using the Promela language. The established model was proved, through model testing conducted with SPIN, to have the nature of the stop-and-wait ARQ. Then the method of attacking the ARQ protocol was discussed, and the attacker was modeled by using the Promela language. The attack process was simulated by using SPIN's graphical interface tool XSPIN, which proved the effectiveness of the attack.

Keywords: stop-and-wait ARQ; SPIN; Promela; model checking

随着科学技术的快速发展,计算机网络已渗透到社会的各个领域,并在日常生活中发挥着越来越重要的作用。作为计算机与网络连接的标准,通信协议已是网络发挥作用的基础,如何保证其可靠性和安全性成为一个热门的研究问题。早期的通信协议大多采用自然语言进行描述,但由于自然语言自身存在的模糊性和二义性等局限,导致通信协议容易出现错误。因此,越来越多的研究人员开始关注通信协议的形式化描述与验证,并促使通信协议成为一种更为严谨的规范。但是,如何对通信协议进行建模并完成安全检测仍然是一件非常困难的工作,且不同通讯协议在建模、验证及性能分析等各方面或阶段的表现也大相径庭[1]。在众多通信协议中,停等式自动请求重传(auto repeat request, ARQ)协议是数据通信网络中最常见的差错控制协议之一。因此,识别停等式 ARQ 存在的弱点有助于改进当前网络协议的不足,从而提高数据通信的安全性。

1　停等式 ARQ 协议

在数据通信网络中,自动请求重传协议常被用于处理信道中的差错控制。ARQ 通过接收端请求发送端重新传输错误数据报文的策略,实现报文在数据链路层的可靠传输。ARQ 协议分成为 3 类[2]:(1)停等式 ARQ,即每一帧确认接收后再发送下一帧;(2)连续式 ARQ,即一次发送多个

收稿日期: 2018-04-16
作者简介: 黄丽丽(1988-),女,江西上饶人,研究实习员,研究方向:高校教学管理、模型检测。

帧后再确认接收;(3)选择性重传 ARQ,即有选择地重传出错帧。可以看出,以上 3 种 ARQ 协议的主要区别是对出错的数据报文的处理方式不同。从停等式 ARQ 至选择性重传 ARQ,其复杂性递增,效率也递增。其中,停等式 ARQ 协议虽然最为简单,但应用的范围却不小,尤其是在硬件无线通讯领域,由于其简单且易于实现而备受青睐[3]。

停等式 ARQ 协议的工作原理可按发送端和接收端分别进行描述。对于发送端,它每发送完一帧便停止发送,并等待接收端的确认。只有当接收到了来自接收端的确认帧之后,发送端才继续发送下一帧。对于接收端,它每收到一个无差错的帧,便将其转发给上层软件,并向发送端发送确认帧。若接收端收到有差错的帧,便直接丢弃该帧,而不做任何其他操作。因此,为了保证报文在数据链路层的可靠传输,发送端必须对所发送的帧进行编号。由于发送端每次只发送一帧,因此停等式 ARQ 协议仅使用一个比特对帧进行编号,即只有 0 号帧和 1 号帧两种类型。发送端依次循环发送 0 号帧和 1 号帧,即先发送 0 号帧,再发送 1 号帧,再下次又发送 0 号帧,如此循环反复。接收端发送确认帧时,则采用“ACKn”的格式来表示已经正确接收到了第 $n-1$ 号帧,当前正在等待接收第 n 号帧。

对于报文在数据传输中可能出现的不同状况,发送端和接收端将有不同的处理机制。如图 1 所示,当数据帧或确认帧出现错误,发送端和接收端会依据停等式 ARQ 协议执行相应的处理操作。

(1)正常情况下,发送端和接收端有序地发送数据帧和确认帧。由于发送端只有在收到确认帧的情况下才会发送下一次帧,所以接收端能够及时接收所有的数据帧,数据在传输过程中不出会现延时或丢失。

(2)发送端发出的数据帧出错。在这种情况下,接收端将丢弃该帧,且不发送确认帧。因此,如果系统要求发送端一定要收到接收端的确认帧后才能发送下一帧,那么接收端在发送数据帧之后就会进入等待,从而导致死锁。为了解决这个问题,系统可要求发送端每发送完一个帧,就启动一个计时器来计算等待时间。当等待时间超过了预先设定的重传时间阈值,那么发送端便重新发送前一个数据帧。显然,重传时间阈值的设定会影响数据传输的效率,因此数据传输往往会有所延时。

(3)发送端发出的数据帧正确,而确认帧发生异常。在这种情况下,发送端无法正确接收到确认帧。当等待时间超过重传时间,发送端将重新发传前一帧。然而,当接收端收到这个重复帧时,它将丢弃该帧,并再次发送确认帧。由此可发现,一旦没有对数据帧进行编号,就可能导致接收端无法确认收到的帧是否重复。

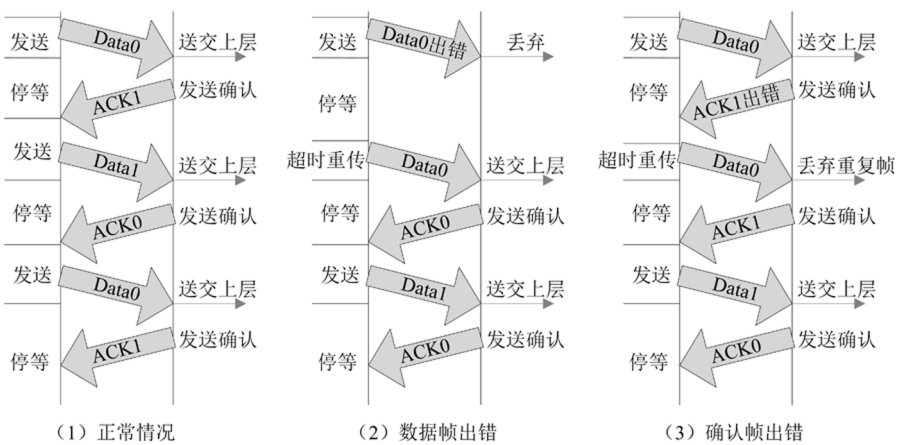

图 1 停等式 ARQ 协议工作原理

Fig.1 Working principle of the stop-and-wait ARQ

2　Promela 语言建模

SPIN 是一种著名的分析验证并发系统逻辑一致性的模型检测工具[4]，它所提供的系统建模语言 Promela 能够用于直观且明确地对有限状态系统进行建模。除了允许动态创建并行的进程之外，Promela 还能够在进程之间进行同步和异步通信。

2.1　消息传递的建模

停等式 ARQ 协议是用于保证数据有序、可靠传递的协议，因此对消息传递的模拟是必不可少的。Promela 语言提供了一种先进先出的消息通道的机制，来模拟消息在进程之间的传递。通道的容量可以任意确定，同时每个消息可以包含多个变量。本文采用消息在通道内的传递来模拟帧在信道中的传输。

在停等式 ARQ 协议中，对于发送端而言，每个帧要包含消息的编号和内容。同时，为了模拟帧的正确传输和错误传输的两种不同情况，还应包含一个帧的状态标记，区别正确传送的帧和错误传送的帧。对于接收端而言，它所传递的消息需要包含确认编号 ACKn，同样为了模拟确认帧的正确传输和错误传输的两种不同情况，确认帧中也应该包含一个帧的状态标记。参考文献[5]的描述方法，本文将帧的状态和通道标记定义如下：

mtype = {mesg, ack, err}; chan s_r = [10] of {mtype, dbyte, mbyte}; chan r_s = [10] of {mtype, dbyte, mbyte}。

其中，mesg、ack 和 err 分别代表该帧是消息帧、确认帧或者错误帧。通道 s_r 用于模拟发送端向接收端发送数据帧所使用的信道，通道 r_s 则模拟接收端向发送端发送确认帧所使用的信道。通道中所传输的每个消息都包含 3 个域，mtype 代表帧的状态标记，dbyte 和 mbyte 分别代表数据和帧编号。

此外，本文在发送端使用 out! msg(o,s) 和 out! err(0,0)来模拟数据帧的正确传输和错误传输两种情况。在接收端，则使用 out! ack(0, es) 和 out! err(0, 0)来模拟确认帧的正确传输和错误传输两种情况。

2.2　超时重传机制的建模

在停等式 ARQ 协议中，出现超时的情况只有数据帧丢失、数据帧出错、确认帧丢失和确认帧出错 4 种：

(1)数据帧丢失：接收端收不到数据帧就不会发送确认帧，导致超时。

(2)数据帧出错：接收端接收到错误的数据帧，将出错的数据帧丢弃，不发送确认帧，导致超时。

(3)确认帧丢失：发送端收不到确认帧，导致超时。

(4)确认帧出错：发送端收到错误的确认帧，将出错的确认帧丢弃，导致超时。

为了标识以上不同情况，本文设置了一个 byte 类型的变量 to 作为超时标记。当它的值为 1 时，表示会出现传输超时的情况，进而启动重传选项。在数据丢失、出错和确认帧丢失时，就将 to 的值赋为 1，并启动重传。当发送端接收到出错的确认帧时，直接启动重传。

2.3　发送端建模

为了证明数据帧是被有序传递的，可在数据帧所传送的数据 o 设置为一个从 0 递增到MAX-1 的量。如果所有的数据帧都被正确且有序地送到了接收端，则数据帧中 o 的值就应该是有序递增。因此，可对发送端建模如下：

模型 1　发送端模型

```
proctype sender(chan in,out)
{
    int o;
    byte s, r;
    o = MAX-1;
    s = 0;
    do
    /* 让待传送的数据递增 */
        ::o=(o+1) % MAX;
again:if
    /* 模拟数据帧的正确传送 */
        ::out! mesg(o,s);
    /* 模拟数据帧的错误传送 */
        ::out! err(0,0); cnt++;}
    /* 模拟数据帧丢失,启动重传选项 */
        ::to = 1;
      fi
      if
        ::(to == 1) ->
```

```
            atomic{to = 0; goto again}
    /* 接收到出错的确认帧,重传 */
        ::in? err(0,0); -> goto again;
    /* 接收到正确的确认帧,修改帧编号 */
        ::in? ack(0, r) ->
            s = 1 - s;assert (s == r);
      fi
    od
}
```

2.4 接收端建模

本文设置了变量 ei 和 es 来分别代表预期数据和预期帧编号。当接收端接收的数据帧编号是正确的,则修改预期帧编号 es,并发送确认帧。由于发送的确认帧可能在传输过程中出现正确传输、错误传输和丢失 3 种情况,因此采用以下方式分别对错误传输和丢失进行模拟:(1)当确认帧丢失的情况,将超时标记 to 赋值为 1;(2)如果该帧的编号不正确,则不修改预期帧编号 es,而是再发送一次确认帧。因此,可对接收端的建模如下:

模型 2 接收端模型

```
proctype receiver(chan in,out)
{
    int i, ei;
    byte s, es;
    ei = 0;
    es = 0;
    do
       ::in? mesg(i,s);->
          if
    /* 数据帧编号正确 */
          ::( s == es)->
              es= (es + 1) % MAX;
              ei=(ei+1) % MAX;
              if
    /* 模拟确认帧的正确传送 */
                 ::out! ack(es,0);
    /* 模拟数据帧的错误传送 */
                 ::out! err(0,0);
                 ::to=1;
              fi
          ::(s ! = es)->
              if
    /* 再次传送确认帧 */
                 ::out! ack(0, es);
                 ::out! err(0, 0);
                 ::to = 1;
              fi
          fi
/* 接收到错误的数据帧,启动重传选项 */
       ::in? err(0,0)->to = 1;
    od
}
```

2.5 主进程建模

最后,在得到发送端模型 sender 和接收端 receiver 的基础上,可构建主进程模型 init,创建上述两个进程的实例,并给它们传递通道:

模型 3 主进程模型

```
proctype init( )
{
    chan s_r = [10] of {mtype, int, byte};
    chan r_s = [10] of {mtype, int, byte};
    atomic
    {
        run sender (r_s, s_r);
        run receiver (s_r, r_s)
    }
}
```

3 SPIN 模型检测

停等式 ARQ 协议主要是用于实现数据的有序、可靠的传递。基于文献[5]所构建的模型以及文献[6]所描述的带参性质形式化描述方法的基础上,本文从发送端行为、接收端行为和活性 3 个方面进一步验证所建模型是否正确。

3.1 发送端行为

如果协议是正确的,发送方总是处于这样的状态:已发送完 0 号数据帧正在等待 1 号确认帧,或已发送完 1 号数据帧正在等待 0 号确认帧。可以通过检验 sender 进程中数据帧的编号和确认帧的编号来检测这个性质。输入以下 LTL 语句进行检测:

```
#define cdn1 (cnt == 0)
#define cdn2 ((r == 0 && s == 1) || (r == 1 && s == 0))
```

cdn1-><>cdn2

检测结果为“erros :0”,这说明在任何时刻,发送方已发送的数据帧的编号与等待接收的确认帧的编号都是不相等的。

3.2 接收端行为

如果协议是正确的,接收方总是处于这样的状态:已发送完 0 号确认帧正在等待 1 号数据帧,或已发送完 1 号确认帧正在等待 0 号数据帧。可以通过检验 reciever 进程中数据帧的编号和确认帧的编号来检测这个性质。输入以下 LTL 语句进行检测:

#define cdn1 (cnt == 0)

#define cdn3 ((s == 0 && es == 1) || (s == 1 && es == 0))

cdn1-><>cdn3

检测结果为“erros :0”,这说明在任何时刻,接收方已接收的数据帧的编号与要发送的确认帧的编号都是不相等的。

3.3 活性

协议的活性是指协议运行时一些好的事情会发生,比如预定的事情会产生、指定的状态会到达、应该进行的协议活动会进行等。在这个协议中,活性是指发送方在给接收方发送数据后,接收方一定会给予应答。接收方在接收数据后,通道内的帧数量为 0。而接收方在应答后,会改变预期接收的数据 ei,从而使其与已接收的数据 i 不同。输入以下 LTL 语句进行检测:

#define cdn1 (cnt == 0)

#define cdn4 (ei != i)

cdn1->[]<>cdn4

检测结果为“erros :0”,这说明在任何时刻,接收方在接收到发送方传送来的数据后,都会给予应答。

4 攻击者建模及检测

停等式 ARQ 协议与其它的底层协议一样,也是一种很脆弱的协议[7]。常见的对停等式 ARQ 协议攻击方法有两种,分别是对传输顺序和对传输内容的攻击。针对这一问题,本文使用 Promela 语言设计一个攻击者的模型,随机执行以下两种攻击:

(1)对数据传输过程进行一次攻击:捕获发送端发送的一个数据帧,并伪造一个确认帧返回给发送端。

(2)拦截 sender 进程的发送的一个数据帧,并将其伪造成一个重复帧发送给 reciever 进程。然后,再冒充 reciever 进程向 sender 进程发送一个确认帧。

由于假设数据有序递增,因此可以很容易地计算出预期接收的数据。对这个性质进行检测,要在接收到正确的数据帧、并确定该帧的编号正确后,加上一个断言语句“assert(i == ei)”。因此,与上述攻击者模型对应的 Promela 语言模型可表述为:

模型 4　攻击者模型

```
proctype attack(chan in,out)
{
    int i, ei;
    byte s, es;
    ei = 0;
    es = 0;
    do
    /*拦截数据帧*/
        ::atomic{in? mesg(i,s);cnt--}->
    /*伪造确认帧*/
            es=1-s;
            atomic{out! ack(0, es); cnt++};
            break;
        ::atomic{in? mesg(i,s);cnt--}->
            es=1-s;
    /*发送确认帧和重复数据帧*/
            atomic{in! mesg(i,1-s);
                out! ack(0, es);}
    od
}
```

使用该攻击者模型对停等式 ARQ 协议在模型进行攻击,发现以下攻击过程。attack 进程拦截了 0 号数据帧,伪造 1 号确认帧发送给 sender 进程,再将 0 号数据帧的编号改为 1,发送给 reciever 进程。receiver 进程被帧编号所欺骗,认为这是一个重复的数据帧,将其丢弃,并重复发送 0 号确认帧。sender 进程在收到伪造的 1 号确认帧后,发送出 1 号数据帧,并期待 0 号数据帧。接着,重复发送的 0 号确认帧到达,而 sender 进程却误认为 1 号数据帧已经被正确接收,发出新的 0 号数据帧。这个新的 0 号数据帧被 receiver 进程

接收,结果造成接收的数据与期待的数据不一致,违背了断言“assert(i == ei);”。这样,attack 进程就成功地完成了一次攻击,使传输过程出现了漏帧。一旦去掉断言“assert(i == ei);”,传输过程虽然仍会正常进行,但在传输过程中,attack 进程在不断进行攻击,从而不断地造成漏帧。更为严重的是,attack 进程在每一次攻击都会拦截一个数据帧,并通过欺骗令发送和接收双方都感觉不到,使传输持续进行。但在传输结束后,接收方所接收到的数据却是不完整的。

由上述的攻击者模型可发现,对停等式 ARQ 协议的攻击是很容易的,攻击方只需要进行简单的拦截和伪造就可以破坏整个传输过程。而且攻击方所付出的代价是很微小的,不需要监控整个传输过程,在任意时刻进行攻击都可以达到破坏传输的目的。尤其是对传输顺序的攻击,只需要进行一次,就可以使整个传输过程停滞,浪费大量的通信资源。

5 结语

本文介绍了停等式 ARQ 协议的模型检测,并使用 Promela 语言对消息传递、超时重传机制、发送端、接收端和主进程都分别进行建模。在此基础上,从发送端行为、接收端行为和活性 3 个方面进一步验证所建模型是否正确。最后,使用 Promela 语言设计一个攻击者模型来模拟对停等式 ARQ 协议的攻击行为。

参考文献:

[1] 李新宇.用于通信网络协议开发的形式化方法[J].中国新通信,2014(15):100-101.

[2] LU D L, CHANG J F. Performance of ARQ protocols in nonindependent channel errors[J]. IEEE Trans Commun, 1993, 41(5): 721-730.

[3] 徐佑军,谭敦茂,朱建武,等.蓝牙无线链路质量的分析、测试与改善[J].计算机工程与应用,2004,40(12):129-131,211.

[4] HOLZMANN G J. The model checker SPIN[J]. IEEE Transactions on Software Engineering,1997,23(5):279-295.

[5] 陈义,唐郑熠.通信协议的 Promela 语言建模与检测[J].福建电脑,2016,32(3):39-40.

[6] 徐永生.带参性质的形式化描述与证明[D].成都:电子科技大学,2015.

[7] 吴勇,李祥.基于 TLA 的 ARQ 协议描述与验证[J].计算机安全,2012(8): 40-43.

(特约编辑:黄家瑜)