撤稿声明

撤稿文章名:	TCP 滑动窗口技术在 RS485 数据通信中的应用
作者:	王博文
* 通讯作者.	邮箱: 29681717@qq.com
期刊名:	计算机科学与应用 (CSA)
年份:	2018
卷数:	8
期数:	7
页码 (从X页到X页):	1072-1081
DOI (to PDF):	https://doi.org/10.12677/csa.2018.87119
文章ID:	1541044
文章页面:	https://www.hanspub.org/journal/PaperInformation.aspx?paperID=25967
撤稿日期:	2019-3-14

撤稿原因 (可多选):

☑ 所有作者

☐ 部分作者:

☐ 编辑收到通知来自于

- O 出版商
- O 科研机构:
- O 读者:
- O 其他:

撤稿生效日期: 2019-3-14

撤稿类型 (可多选):

☐ 结果不实

O 实验错误 O 数据不一致 O 分析错误 O 内容有失偏颇

O 其他:

☐ 结果不可再得

☐ 未揭示可能会影响理解与结论的主要利益冲突

☐ 不符合道德

☐ 欺诈

O 编造数据 O 虚假出版 O 其他:

☐ 抄袭 ☐ 自我抄袭 ☐ 重复抄袭 ☐ 重复发表 *

☐ 侵权 ☐ 其他法律相关:

☐ 编辑错误

O操作错误 O 无效评审 O 决策错误 O 其他:

☑ 其他原因:

出版结果 (只可单选)

☐ 仍然有效.

☑ 完全无效.

作者行为 失误(只可单选):

☐ 诚信问题

☐ 学术不端

☑ 无 (不适用此条，如编辑错误)

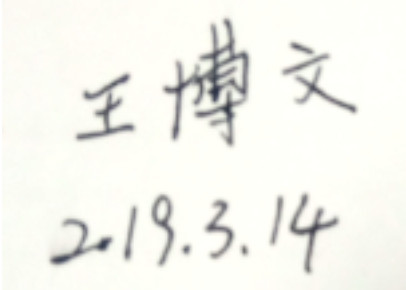
* 重复发表: "出版或试图出版同一篇文章于不同期刊."

历史
作者回应:
☐ 是，日期: yyyy-mm-dd
☑ 否

信息改正:
☐ 是, 日期: yyyy-mm-dd
☑ 否

说明:
“ TCP滑动窗口技术在RS485数据通信中的应用”一文刊登在2018年7月出版的《计算机科学与应用》2018年第8卷第7期第1072-1081页上。作者认为文中涉及的技术比较陈旧，无现实参考意义，文章尚有许多不足，需进一步完善，以免误导读者。根据国际出版流程，编委会现决定撤除此稿件：王博文. TCP 滑动窗口技术在 RS485 数据通信中的应用[J]. 计算机科学与应用, 2018, 8(7): 1072-1081. DOI: 10.12677/csa.2018.87119

所有作者签名：



Computer Science and Application 计算机科学与应用, 2018, 8(7), 1072-1081
Published Online July 2018 in Hans. http://www.hanspub.org/journal/csa
https://doi.org/10.12677/csa.2018.87119

Hans汉斯

Application of TCP Sliding Window Technology in RS 485 Data Communication

Bowen Wang

Chongqing Research Institute Co., Ltd. of China Coal Industry Group, Chongqing
Email: 29681717@qq.com

Received: Jun. 30th, 2018; accepted: Jul. 12th, 2018; published: Jul. 19th, 2018

Abstract

RS485 communication protocol is widely used in the field of industrial data communication. It has many advantages, such as simple network structure, strong anti-interference ability and long transmission distance. However, there is no unified standard for the implementation of RS485 communication protocol in the transmission layer, which requires the definition of each industry manufacturer, and most manufacturers use the query response mode of single data packet, which leads to the low efficiency of channel utilization. When the data transmission is small, the waste of time cost is not outstanding. However, with the continuous development of the digital intelligent equipment and the increase of the amount of communication data, the RS485 protocol also shows some drawbacks in the practical application, such as low transmission efficiency and low channel utilization. TCP protocol is a connection oriented, reliable, byte stream based transport layer communication protocol. TCP sliding window technology is a mature and reliable transmission layer communication protocol. It has the advantages of error control, flow control, high efficiency of data transmission and so on. The application of TCP sliding window technology to the RS485 communication protocol will greatly improve the current data communication in the industrial.

Keywords

TCP Sliding Window, RS485 Communication, Data Communication, TCP Protocol

TCP滑动窗口技术在RS485数据通信中的应用

王博文

中煤科工集团重庆研究院有限公司，重庆
Email: 29681717@qq.com

收稿日期：2018年6月30日；录用日期：2018年7月12日；发布日期：2018年7月19日

文章引用：王博文. TCP 滑动窗口技术在 RS485 数据通信中的应用[J]. 计算机科学与应用, 2018, 8(7): 1072-1081.
DOI: 10.12677/csa.2018.87119

摘　要

RS485通信协议在工业数据通信领域中被广泛采用，它具有组网简单、抗干扰能力强、传输距离远等诸多优势。但RS485通信协议在传输层的实现没有统一标准，需要各个工业厂商自己定义，而大部分厂家采用的是单数据包的询问应答模式，导致信道利用效率低。当数据传输量小的时候，时间成本的浪费并不突出，然而但随着数字智能设备的不断发展，通信数据量的不断增大，RS485协议在实际的应用中也凸现一些弊端，如传输效率低，信道利用率不高等问题。TCP协议是一种面向连接的、可靠的、基于字节流的传输层通信协议。TCP滑动窗口技术是一种成熟可靠的传输层通信协议，它具有差错控制、流量控制、较高的数据传输效率等优势，本文将TCP/IP滑动窗口的数据传输机制应用于RS584通信中，有效解决了RS485上层应用协议传输数据效率不高，时间成本浪费的问题，极大提高了传输效率。

关键词

TCP滑动窗口，RS485通信，数据通信，TCP协议

1. 引言

RS485 通信协议在工业领域应用比较广泛，例如智能仪表、加气机、加油机等智能终端设备在数据通信中广泛采用 485 通信协议。

RS485 接口组成的半双工网络，一般是两线制或四线制(只能采用点对点的通信方式，很少采用)，多采用屏蔽双绞线传输。这种接线方式为总线式拓扑结构，在同一总线上最多可以挂接 32 个结点[1]。在 RS485 通信网络中一般采用的是主从通信方式，即一个主机带多个从机。

RS485 通信较 RS232 通信有很多优点如接口是采用平衡驱动器和差分接收器的组合，抗共模干扰能力增强，即抗噪声干扰性。还有就是相比 RS232 通信，RS485 通信可以组网[2]。

RS485 通信在组网的情况下，主机与从机采用的是轮询的通信方式。一般是主机发起包含从机地址的点名信息，从机收到点名信息后解析报文，如果解析出的报文地址相匹配就会确认该报文是发送给自己的，就会在一定的时间段内回传给主机相应的信息。如果解析出的报文地址不匹配就会丢弃该报文。

RS485 通信也可以采用 点对点的方式，一台主机通过串口多功能卡扩展后可以支持多路串口，再用多路串口转多路 485 转换器可以实现多路点对点的通信方式[3] (图 1)。

较之组网的通信方式，多路点对点的通信方式因为效率比较高，程序实现简单而更广泛的应用。

RS485 通信协议在运用过程中通常采用询问应答的模式，这种模式造成了时间成本的浪费，而 TCP/IP 协议的滑动窗口技术能够有效解决这一问题。

2. RS485 协议的实际应用

工业领域中，设备仪器仪表利用 RS485 通信协议进行通信时，报文大致可以分为三类，通信握手类、指令类和数据类。通信握手报文一般用于诊断设备是否在线，获取设备当前的状态等。而指令类报文则

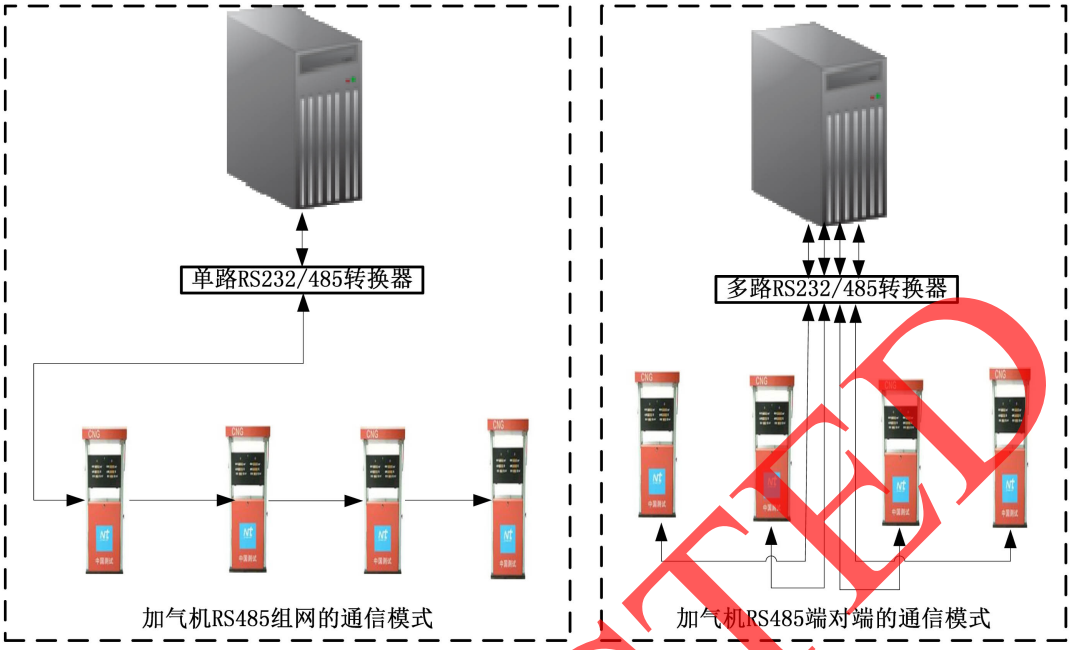


Figure 1. Two communication modes of RS485
图 1. RS485 的两种通信模式

包含读、写两种指令，一般通过上位机发出，让设备完成相应的动作[4]。数据类报文则包括关键的数据信息，如加气机的加气交易等等信息。数据类报文在以上三种报文中长度最长，优先级最高，传输中所占用的资源也最多。

加气机离线交易的传输效率

加气机的交易数据包含了加气量、单价、金额、POS 枪号等诸多关键信息，当加气机在线时会把交易信息以请求，应答的形式源源不断的发送至后台 PC 终端。而当加气机离线的情况下，交易数据只能累计存储在本机存储器里面，待通讯恢复正常后，加气机才开始传输离线交易。以一台加气机存储的最多离线交易记录条数(2 万条)为例，RS485 通信中常规的波特率是 9600 bps，那么传输万 2 万条离线交易的时间大约为：

$$T(\mathrm{s}) = \frac{20000(\text{条}) * (200\ \mathrm{Byte}/\text{条}) * (8\ \mathrm{bit}/\mathrm{Byte})}{9600\ \mathrm{bit}/\text{秒}} = 3333\ \text{秒}$$

RS485 考虑到通信抗干扰效果采用 9600 的波特率，3333 秒约为 1 个小时。

以上是信道满负荷最优情况下理论计算出的数据传输时间，最优是指单个设备与单个主机进行通讯，设备持续发送，主机持续接收，无错误报文出现。

而实际设备与主机进行通信时，为了保障数据的有效传输，往往引入了应答机制。也就是主机每次正确接收设备传输来的数据后，都要回复 ACK 应答包，设备收到了主机传输来的 ACK 应答包后，才开始下一个报文的上送如图 2 所示。

按照上文所述加气机 2 万条交易记录为例，传输时保守的估计为 2，大约为 2 个小时，信道的利用(不含 ACK 应答)约为 50%。所以，当批量传输数据的时候，RS485 的这种简单询问应答的传输机制会产生大量的时间成本，严重影响系统的实时性和可靠性。然而，TCP 的滑动窗口技术能有效的解决数据传输中的时间冗余，提高信道的利用率。

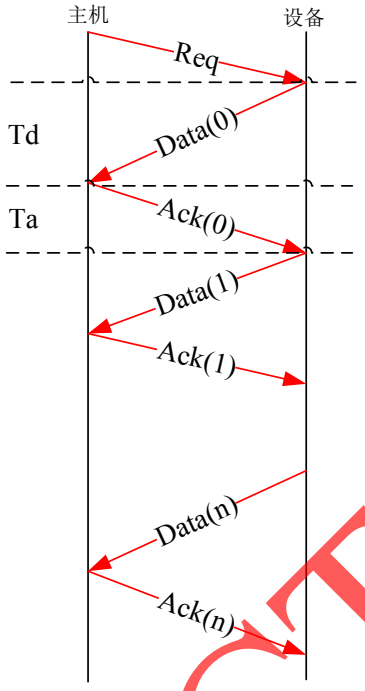


Figure 2. RS485 data transmission response example
图 2. RS485 数据传输应答示例

3. TCP 滑动窗口技术

滑动窗口协议是传输层进行流控的一种措施，接收方通过通告发送方自己的窗口大小，从而控制发送方的发送速度，从而达到防止发送方发送速度过快而导致自己被淹没的目的[5]。

滑动窗口原理分析

滑动窗口协议的基本原理就是在任意时刻，发送方都维持了一个连续的允许发送的帧的序号，称为发送窗口；同时，接收方也维持了一个连续的允许接收的帧的序号，称为接收窗口[6]。发送窗口和接收窗口的序号的上下界不一定要一样，甚至大小也可以不同。不同的滑动窗口协议窗口大小一般不同。发送方窗口内的序列号代表了那些已经被发送，但是还没有被确认的帧，或者是那些可以被发送的帧[7]。

TCP 是全双工协议，发送方和接收方是对等的。对于 TCP 会话的发送方，任何时候在其发送缓存内的数据都可以分为 4 类。

1) 已经发送并得到对方 ACK 的数据包
2) 已经发送但还未收到对方 ACK 的数据包
3) 未发送但对方允许发送的数据包
4) 未发送且对方不允许发送的数据包

已经发送但还未收到对方 ACK 的数据包和未发送但对方允许发送的数据包这两部分数据称之为发送窗口。

TCP 建立连接的初始，接收方会告诉发送方自己接收窗口大小，比如 10。

如图 3 所示，滑动窗口包含已发送但未收到确认的字节和允许发送但尚未发送的字节[8]。当图中红色的区域数据都被接收到之后，那么滑动窗口将会向右移动。窗口两个边沿的相对运动增加或减少窗口的大小从而让网络数据传输避免了拥堵。

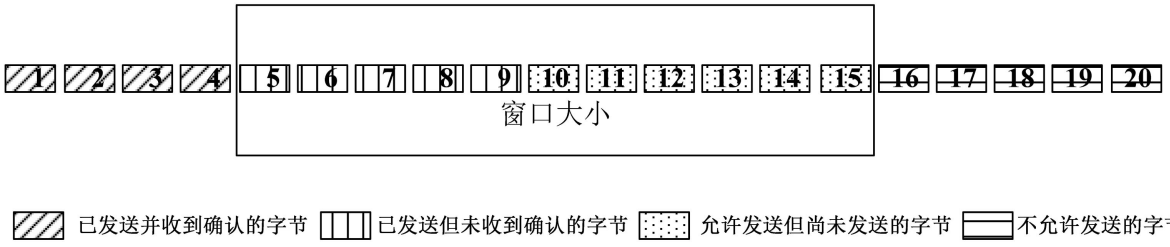


Figure 3. TCP sliding window data receive positive sequence of data
图 3. TCP 滑动窗口数据收到正序的数据

数据的接收可以分为正序或者乱序，正序是指接收方按发送方正常的发送序列接收到数据，而乱序是指接收方没有按照发送方正常的发送序列接收数据，也就是存在丢包。对于丢包的乱序数据，接收方暂时存储起来，避免网络重复传输。

如图 4 所示，字节(序号 6、7)比字节(序号 5)先收到接收方的确认，那么待字节 5 收到接收方的确认后，滑动窗口才会右移。

4. 滑动窗口技术在 RS485 数据通信中的应用

在 RS485 应用层通信协议中，发送方和接收方同时维护了一个发送缓存和接收缓存，窗口大小可以看作为 1，代表 1 个数据包。由于发送方接收到第 N 个数据包的响应后，才开始第 N + 1 个数据包的发送，那么接收方接收到的数据包必定是正序的[9]。

4.1. RS485 数据包的高效率传输模式

假设 RS485 应用层的数据传输协议采用 TCP 滑动窗口的方式，窗口的大小调整为 N，与 TCP 不同的是这里的 N 指的是 N 个数据包，而非字节数。

还以加气机离线交易数据上传为例，假设在通信效率很高的情况下，也就是主机发出的数据包，设备都能够有效响应，并且响应一次主机就能收到 ACK。

图 5(1)中是设备连续收到 N 包数据后，回复 N 个 ACK 的示例。由于 RS485 常用的是半双工的通信方式，所以图 5(1)中，主机在收到设备的离线交易数据之后，设备线程必须停止发送数据等待主机下发的 ACK 响应，这种模式与图 2 提到的发送应答模式在效率上并无多大区别。

实际的 ACK 应答包很短，有效数据很少，所以完全可以用一个 ACK 应答包来涵盖 Data(0)至 Data(N − 1)N 包数据的应答响应，如图 5 中(2)所示。这种模式下 ACK 响应的数量从 N 个减为 1 个(如表 1 所示)，效率大大提高了。

设备在发送 N 个数据包至主机后，立即启动定时器等待主机返回的 ACK，在定时器超时之前接收到 ACK 组合后便对 ACK 包进行解析。假设 N 个数据包的 ACK 全部收到，则滑动窗口右移，进行下一组数据包的发送。如果 N 个数据包中，有个别数据包没有收到，则发送相应的数据包，并启动定时器接收主机发送的下一个 ACK 组合。

4.2. RS485 数据传输的纠错与流控机制

同时为了保证 RS485 数据传输的可靠性，引入同步、顺序、流控、纠错的机制。

图 6(1)为本文所阐述的 RS485 数据传输的纠错原理图，主机和设备同时维护一个数据包为 N 的窗口。

1) 主机请求设备上送数据，发出 Req(Req 包含当前窗口序列号 Q，窗口大小 N，需要设备发送的

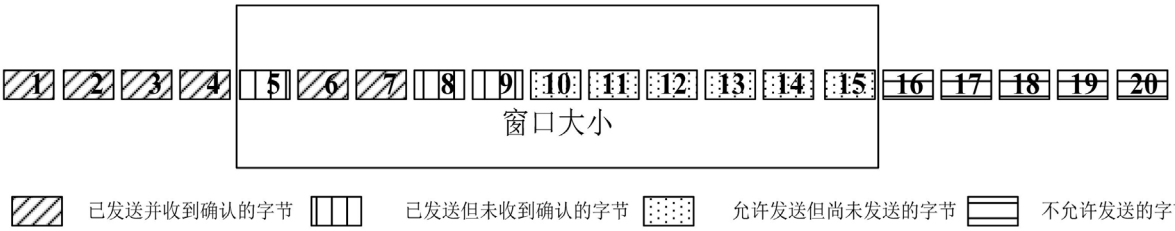


Figure 4. TCP sliding window receives disordered data
图 4. TCP 滑动窗口收到乱序的数据

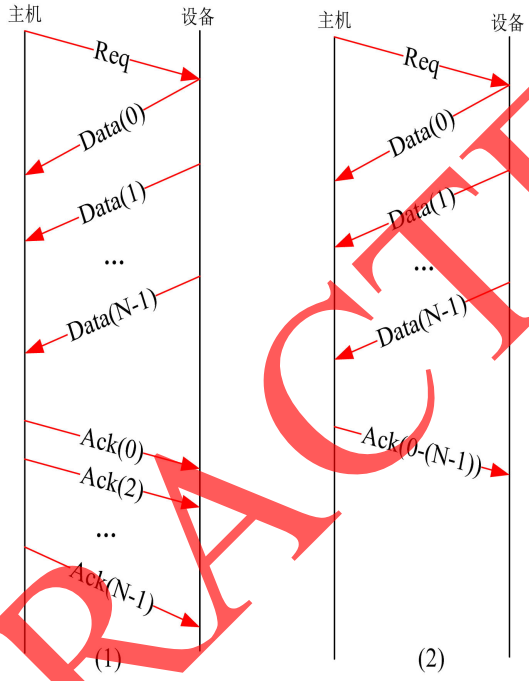


Figure 5. RS485 data transmission using TCP sliding window mode
图 5. RS485 数据传输采用 TCP 滑动窗口模式

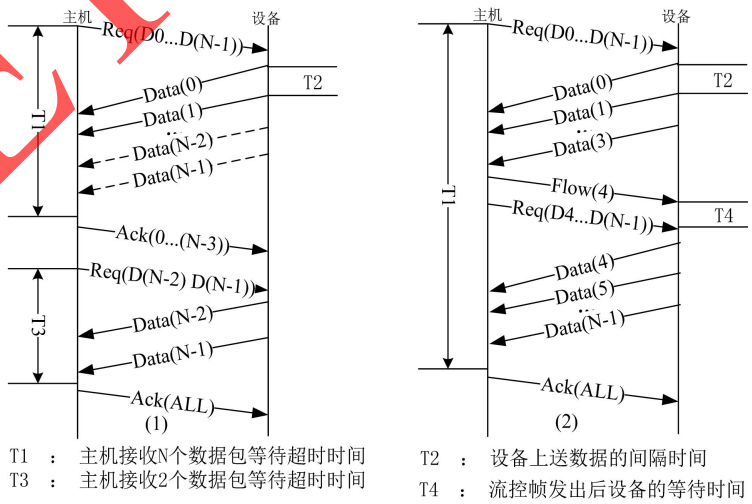


Figure 6. Error correction and flow control mechanism used in RS485 data transmission
图 6. RS485 数据传输采用的纠错和流控机制

Table 1. RS485 adopts TCP sliding window mode ACK message format
表 1. RS485 采用 TCP 滑动窗口模式 ACK 报文格式

包头	主机 ID	设备 ID	ACK(0)	ACK(1) …… ACK(8)	ACK(N-1)	包尾

数据包序列号 0 …… N − 1 等)，当 Req 发出后立即启动数据接收超时定时器，当定时器超时时触发数据包接收机制，接收机制用来统计接收到的有效数据包的个数[10]。

2) 主机接收机制判断当前的窗口 Q 未接收到完整的数据包，丢失了其中 D(N − 2)、D(N − 1)两个数据包，立即启动重传机制再次发出 Req (此时的 Req 仅包含需要重传的数据包 D(N − 2)、D(N − 1))。

3) 当主机收到当前窗口 Q 所有的数据包后，发出 Ack(ALL)指令，表明当前的数据包全部被有效接收，此时当前的窗口传输完成。

```
void PcRequest(void)
{
    u8 i = 0;
    static u8 NumNoReceiv = 0;
    u32 temp_map = 0;
    static u32 Receive_map = 0;
    static u8 state = 0;
    switch(state)
    {
    case 0:
        Receive_map = 0xFFFFFFFF;
        if(SendFL == TRUE)
        {SendFL == FALSE;state = 1;}
    break;
    case 1:
        while(i < WINSIZE)
        {
            if((temp_map&0x8000) > 0)
            {HasDataFlg = TRUE;NumNoReceiv++;}
            i++;
            temp_map <<= 1;
        }
        if(HasDataFlg == TRUE)
        {HasDataFlg = FALSE;state = 1;}
        else{Resault = TRUE;state = 0;}
    break;
    case 2:
        Uart_ReqDeviceData(Receive_map, PackArraySeq);
        state = 2;
        Timer_Start(NumNoReceiv*ONE_WAIT_TIME);
```

```
        break;
        case 3:
            if(Timer_Expire() == FALSE)
            {
                if(Receive(Packet(n)) == TRUE)
                {Receive_map &= (~(0x8000 >> n));}
            }
            else{state = 0;}
        default:
        break;
        }
}
void DeviceSendPackets(void)
{
        u8 i = 0;
        static u8 NumReadySend = 0;
        u32 temp_map = 0;
        static u8 state = 0;
        switch(state)
        {
            case 0:
            if(Uart_Receive(Send_map, PackArraySeq) == TRUE)
            {
                state = 1;
                temp_map = Send_map;
            }
            break;
            case 1:
                while(i < WINSIZE)
                {
                    if((temp_map&0x8000) > 0)
                    {HasDataFlg = TRUE;state = 2;break;}
                    NumReadySend = i;
                    temp_map <<= 1;
                    i++;
                }
                state = 0;
            break;
            case 2:
                HasDataFlg = FALSE;
```

```
                Uart_SendPcData(PackArraySeq, NumReadySend);
                state = 3;
                Timer_Start(ONE_SEND_WAIT_TIME);
            break;
            case 3:
                if(Timer_Expire() == FALSE){wait();}
                else{state = 1;}
            default:
            break;
        }
    }
```

以上算法中 PC 端和 DEVICE 同时维护一个值相同的 Receive_map 和 Send_map，PC 端不断查询 Receive_map 并把值发给 DEVICE 端，DEVICE 端查询接收到的 Receive_map (这里等于 Send_map)，并发送给 DEVICE 端。当所有 bitmap 里面代表的当前窗口的数据包发送完成后，当次数据传输完成。为了方便起见，本例的 bitmap 设置为 0xFFFFFFFF，每位代表一个数据包，共标志 32 个数据包的发送接收状态。此时的窗口大小也为 32。

为了更好的阐述 RS485 应用滑动窗口的思想，用状态机表示以上的算法如图 7 所示。

这样传输 N 个数据包，只有 1 个 ACK 总应答包，在通信良好的情况下，信道的利用率大大提高了。主机接收从机发送数据等待的时间大大减少了。并且该机制引入了纠错以及流控机制，有效避免了差错引起的拥堵以及冗余时间成本的浪费[11]。

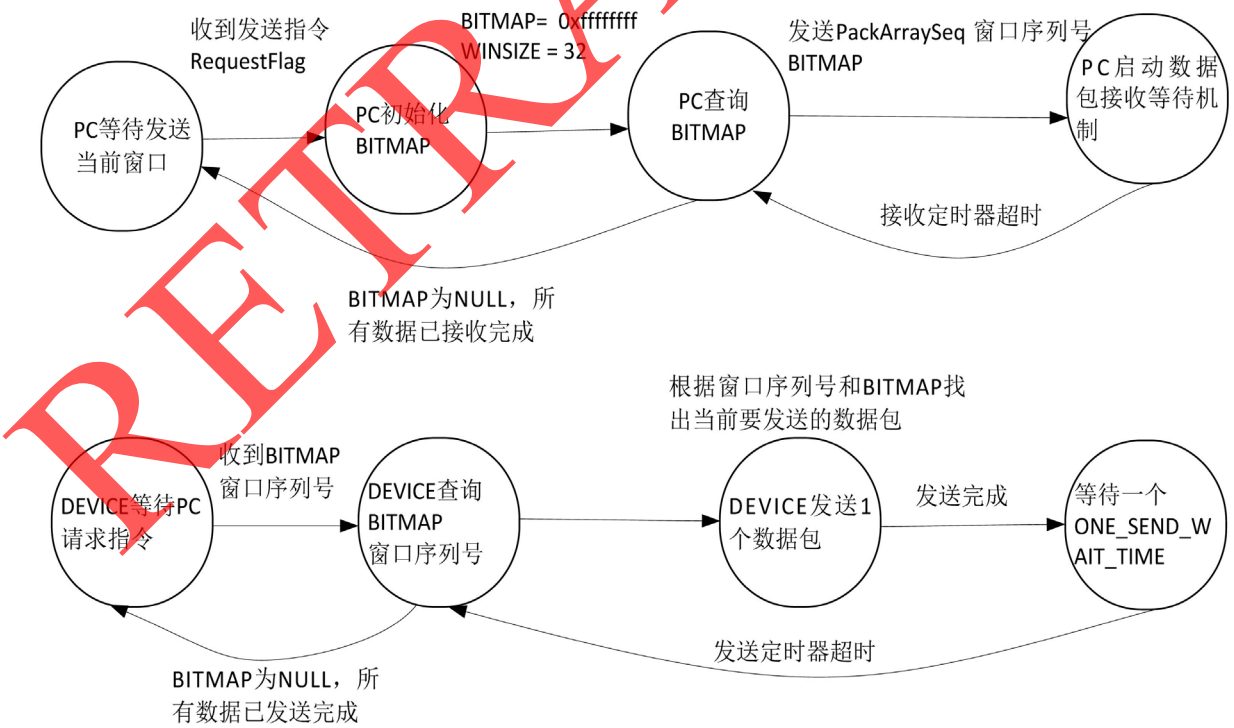


Figure 7. State machine representation of RS485 sliding window technology
图 7. 状态机表示 RS485 滑动窗口技术

5. 结束语

综上所述，可见工业设备数据通信领域，特别是 RS485 协议上层应用借鉴 TCP 滑动窗口协议可以极大的提高数据传输效率，有效的解决因通信速率低引起的数据堵塞，数据冗余，以及传输时间成本过高的问题。本文在 TCP 滑动窗口协议的基础上将纠错流控机制加入到 RS485 的传输协议当中，可以有效的提高数据传输的可靠性以及实时性。

参考文献

[1] 束长宝, 于照, 张继勇. 基于 TCP/IP 的网络通信及其应用[J]. 微计算机信息, 2011, 30(12): 12-15.

[2] 赵国锋, 邱作雨, 张毅. 基于单片机的嵌入式 TCP/IP 协议栈的设计与实现[J]. 计算机技术与发展, 2009, 31(3): 20-22.

[3] 赵甫哲. 高速 TCP/IP 网络拥塞控制算法研究[J]. 华中科技大学学报, 2009, 32(5): 37-39.

[4] 潘文婵, 章韵. Wireshark 在 TCP/IP 网络协议教学中的应用[J]. 计算机教育, 2010, 31(3): 7-8.

[5] 刘爽, 史国友, 张远强. 基于 TCP/IP 协议和多线程的通信软件的设计与实现[J]. 计算机工程与设计, 2010, 30(4): 50-52.

[6] 孟蕾, 陈文艺, 宋焕生. 嵌入式 TCP/IP 实现的研究和分析[J]. 微计算机信息, 2012, 30(6): 31-33.

[7] 徐海军, 刘金刚, 王益华. 基于 ARM 核的嵌入式 TCP/IP 协议栈简化实现[J]. 计算机应用研究, 2009, 30(9): 21-23.

[8] 郭传雄, 郑少仁. 嵌入式 TCP/IP 实现的研究和分析[J]. 计算机工程与设计, 2015, 31(6): 17-19.

[9] 韦安, 刘国平. 对 Linux 操作系统中 TCP/IP 网络协议的 IP 层排队分析[J]. 西安邮电学院学报, 2013, 31(3): 6-8.

[10] 王晓鹏. TCP/IP 下的 Socket 及 Winsock 通信机制[J]. 航空计算技术, 2011, 30(7): 16-19.

[11] 王海燕, 魏洪波. 嵌入式 WEB 中 TCP/IP 协议栈的设计与实现[J]. 微计算机信息, 2009, 31(9): 41-42.