嵌入式Internet中Nagle算法及其应用研究

王宝宝，余世明，王振宇
（浙江工业大学信息工程学院，杭州 310023）

[摘要] 嵌入式Internet中使用短帧数据包，使得网络带宽的利用率极低，拥塞发生的可能性加大。标准transmission control protocol（TCP）协议中应用Nagle算法减少短帧数量。通过ARM7 32位micro control unit（MCU）和上位机personal computer（PC）构建嵌入式Internet网络，分析Nagle算法的原理和工作机制。针对嵌入式系统中Nagle算法和上位机延迟确认策略交互产生的暂时性“死锁”问题，提出了在不修改Nagle算法的基础上，通过提高采样频率或者填充缓冲区的方法来避免暂时性“死锁”。测试表明，该方案是有效可行的。

[关键词] Nagle算法；死锁；延迟确认策略；ARM7；嵌入式Internet
[中图分类号] TP393 **[文献标识码]** A **[文章编号]** 1009-1742（2014）02-0101-05

1 前言

随着计算机网络技术的迅速发展和嵌入式系统的广泛应用，越来越多的嵌入式设备需要接入Internet，以实现远程数据采集、远程监控、自动报警等功能[1]，因此嵌入式Internet技术应运而生。由于其结合了嵌入式技术和Internet技术的优点，具有稳定性好、实时性强、数据传输速率高、成本低廉等特点，因此，近年来已取得了飞速发展和广泛的应用[2]。

嵌入式Internet技术的关键是在嵌入式系统中移植transmission control protocol（TCP）/IP协议栈[3]。嵌入式系统一般是基于8位、16位或者32位micro control unit（MCU）来实现，硬件资源无法与personal computer（PC）相比，所以嵌入式系统中移植的TCP/IP协议栈一般是经过简化处理的。TCP协议作为TCP/IP协议栈中最重要也是最复杂的协议，在移植的过程中，一般只是实现其基本的通信机制，如连接的建立与关闭机制、超时重传机制、数据包确认机制等[4]，而忽略了其丰富的拥塞控制算法。

本文基于ARM7 32位MCU构建嵌入式Internet。在通信测试中笔者等发现网络中存在大量的短帧数据包，导致带宽资源的严重浪费，甚至引起网络拥塞。针对短帧泛滥问题，在嵌入式TCP协议中实现Nagle算法。针对Nagle算法和延迟确认策略交互时产生的暂时性“死锁”问题，通过提高采样频率或者填充缓冲区的方式避免暂时性“死锁”的产生，在提高带宽利用率的同时确保了数据的实时传输。

2 嵌入式Internet技术的实现

本文所设计的嵌入式系统硬件结构如图1所示。按功能主要分为4部分：微控制器MCU、相关外围电路、网络接口和其他接口。微控制器采用NXP（恩智浦半导体）公司生产的ARM7 32位MCU LPC2368。相关外围电路包括电源电路、复位电路、JTAG电路、时钟电路、存储电路等。网络接口由

[收稿日期] 2012-11-06
[基金项目] 浙江省重大科技专项（2011C11089）
[作者简介] 余世明，1962年出生，男，甘肃天水市人，教授，博士，主要研究领域为模型预测控制与系统辨识、嵌入式系统在自动化装置的应用；E-mail：ysm@zjut.edu.cn

PHY（物理接口收发器）DM9161AEP、耦合隔离变压器J00-0065NL组成，负责物理层数据的收发。其他接口部分包括CAN（控制器局域网络）接口和232接口，主要完成相关传感器数据的采集和控制信号的转发功能。在软件上，根据系统要求，从传输层到网路层依次裁剪移植了TCP、ICMP（英特网控制报文协议）、IP、ARP（地址解析协议）协议，并在数据链路层实现了802.3标准。笔者等将PC定义为设备A，IP地址设置为10.1.60.2，将嵌入式设备定义为设备B，IP地址设置为10.1.60.168，构建嵌入式Internet系统。本文所涉及的数据统计和分析均基于该系统所获得。

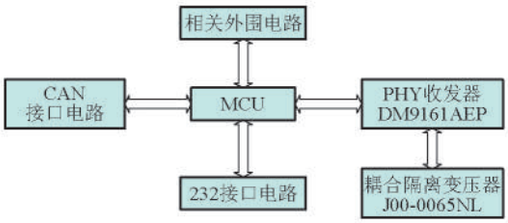

图1　系统硬件结构

Fig.1　Architecture of system's hardware

3　短帧泛滥及其解决方法

3.1　短帧问题

当TCP需要传输一个字节的数据时，在网络层就会产生41 Byte的分组，此时额外开销达到4 000 %[5]，导致网络的利用率极低。考虑到数据链路层和物理层的额外开销之后，这种情况下带宽的利用率更低[6]。嵌入式Internet中存在着大量的长度为1~4字节的数据帧，一方面这些短帧的存在使得网络带宽的利用率极低；另一方面，由于大量的短帧数据包的存在，使网络发生拥塞的可能性增加，容易引起数据传输时延不确定甚至网络崩溃。

3.2　组块技术（clumping）

20世纪60年代后期，短帧泛滥问题在Tymnet网络中第一次被提出，当时采取的方法是在TCP发送端对短数据报延迟一段时间（200~500 ms）再传输，以期在延迟过程中有若干数据到来并附加在同一数据报中发送出去，最终达到减少网络中短帧的目的[5]。这就是组块技术。针对组块技术，笔者等进行了如下测试：下位机（设备B）连续向上位机（设备A）发送10 000 Byte数据，TCP发送端分别延迟200 ms和500 ms。其中下位机应用层产生数据的速率为1 Byte/ms。表1为通过Wireshark抓包分析工具获取的数据包统计。

表1　不同延时下数据包统计

Table 1　Packets statistics under different delay time

延迟时间	数据流向	数据包/个	长度/Byte
500 ms	A→B	23	1 254
	B→A	22	11 220
200 ms	A→B	52	2 820
	B→A	52	12 870

当下位机（设备B）应用层以恒定的速率产生数据时，如表1所示，延迟500 ms产生的数据包数目比延迟200 ms少很多，字节总量也有相应的降低。其根本原因在于组块技术可以减少报头开销，减少TCP发送数据报数量和相应的应答数据报数量。但是通常情况下，应用层产生数据的速率是不固定的，或者说应用层相邻两个数据之间的时延是不确定的。所以组块技术最大的缺陷在于不能给出一个通用的延迟时间[5,7]，一个固定的时延不可能适用于所有情况。

3.3　Nagle算法

1984年，John Nagle提出Nagle算法。该算法应用在TCP发送端，其作用是决定TCP发送端何时通过TCP连接发送一个数据报[7]。算法描述为：如果发送端有很小的数据报需要发送，那么只要先前传送的数据报都已经被TCP接收端确认或者说发送端不存在未被确认的数据报，这时发送端就可以发送数据。前提是这个很小的数据报的长度小于TCP发送端的最大报文段长度（MSS）。其具体的工作流程如图2所示。

相比于组块技术，Nagle算法是一种自适应算法[6]，其本质是一种自适应的组块技术，它有两个触发条件：本端应用层所组块的大小和确认到达的时间。这两个触发条件决定了这种自适应组块技术的延迟时间。通过这两个触发条件，将组块技术的固定延迟转变为动态的延迟，使得带有Nagle算法的TCP拥有了一个动态的发送缓冲区，因此其具有很好的通用性。表2为嵌入Nagle算法的TCP抓包分析，这里应用层产生数据的速率为1 Byte/ms。

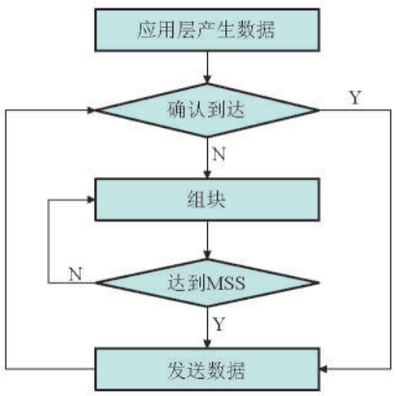

图2　Nagle算法流程

Fig.2　Procedure of the Nagle algorithm

表2　Nagle处理抓包分析

Table 2　Analysis of capture of nagle

包序号	抓包时间/s	时间间隔/s	长度/Byte
1	3.173 518	—	—
2	3.376 500	0.202 982	203
3	3.595 546	0.219 046	219
4	3.813 558	0.217 812	218
5	4.016 528	0.202 970	203
6	4.219 564	0.203 036	203

相比于组块技术的固定数据长度，采用Nagle算法的TCP发送数据长度是一个略大于200 Byte的波动值。波动的产生是由于Nagle算法将数据在网络中的传输时延这个因素考虑进来，至于为何是一个略大于200 Byte的值，将在下面进行详细介绍。

4　Nagle算法和延迟确认（delayed ACK）策略的交互

4.1　延迟确认策略

在标准TCP中还包括另外一种用来减少网络中短帧数量的算法：延迟确认策略。该算法应用在TCP接收端，它使得TCP接收端在接收到数据后并不立即产生确认包，而是延迟一段时间（典型值为200 ms）再进行确认。如果在这段延迟时间内有新的报文段到来，TCP接收端立即发送确认包[8]。延迟确认策略可以有效减少过多的短帧确认包的产生。

4.2　两种交互方式

不同于组块技术，Nagle算法是基于确认的到达来决定是否发送报文[6]，所以TCP发送端的Nagle算法很容易触发TCP接收端的延迟确认策略。图3为Nagle算法和延迟确认策略的两种交互方式。

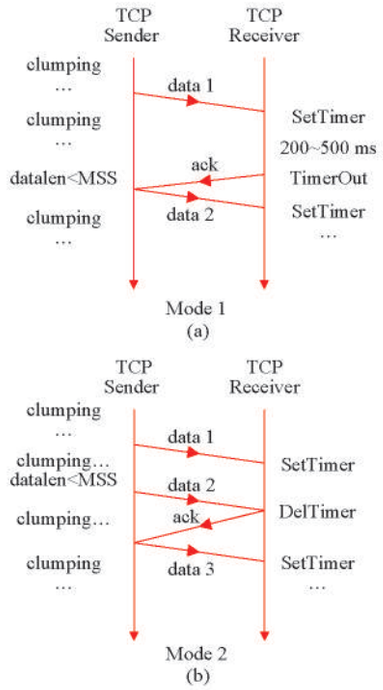

图3　Nagle-delayed ACK交互方式

Fig.3　Interactive mode between Nagle-delayed ACK

在Mode 1中，发送端发送data1给接收端，同时对后续产生的数据进行组块；接收到数据后，接收端设定一个200~500 ms的定时器。定时器超时后，接收端发送确认包，发送端在收到data1的确认之后发送data 2。这样，Mode 1在每一次数据交互过程中都会产生至少200 ms的时延，也被称为暂时性“死锁”（“deadlock”）[9]。暂时性“死锁”的存在，使得表2中的数据长度为一个略大于200 Byte的值。它给系统带来两大危害：a.产生频繁的定时器超时事件，而处理定时器超时事件在操作系统中是一个很耗资源的行为[8]；b.带来明显的延迟，影响实时性要求比较高的系统的正常运行[10,11]。

Mode 2与Mode 1的区别在于发送端发送data1之后，在接收端定时器超时之前，发送端生成了一个达到MSS的报文data 2并立即发送，接收端在连续收到两条未确认的报文之后立即发送确认包，同时删除定时器。Mode 2避免了暂时性“死锁”。

4.3　暂时性“死锁”的避免

针对暂时性“死锁”问题，一些网络实施者采取禁用Nagle算法的做法，但是也只是某一类应用中，更多的是对Nagle算法进行改进。文献[8~11]分别介绍了几种Nagle算法的改进算法，但都是基于协

议的修改，在资源有限的嵌入式系统中实现起来并不容易。

在这里，可以根据嵌入式系统中的数据信号类型，采取不同的处理方式，避免暂时性“死锁”。

针对随机信号，可以采用填充缓冲区的方式。具体实施如下：针对应用层发送缓冲区中的数据，如果是随机信号，则在该数据信号尾部添加无用的数据段，直至总的报文长度达到最大报文段长度，然后将填充后的数据交Nagle处理。前提是该随机信号不能容忍200 ms的时延。

笔者等模拟测试了该方案，设定应用层产生数据的速率为1 Byte/ms，MSS值为1 460 Byte。测试结果如表3所示。序号为3的包为随机信号，数据长度为2 Byte。在该数据尾部填充1 458 Byte的0xFF，然后将填充后的数据交由Nagle处理。由于此时数据长度达到MSS值，Nagle算法立即启动数据发送。在表3中可以看到，3号包从产生到发送只用了4.096 ms的时间，远远低于其他包的200 ms以上的延迟时间。

表3　填充缓冲区抓包分析

Table 3　Analysis of capture of buffer filling

包序号	抓包时间/s	时间间隔/s	长度/Byte
1	353.961 285	—	—
2	354.168 396	0.207 111	207
3	354.172 492	0.004 096	1 460
4	354.381 734	0.209 242	210
5	354.584 737	0.203 003	203
6	354.802 743	0.218 006	218

针对周期性信号，可以采用提高采样频率的方式。具体实施如下：针对需要周期性进行采样的数据，提高其采样频率，以期在200 ms时间内采样的总的数据长度达到或者超过MSS值。前提是这个周期信号不能容忍200 ms的时延。

表4为模拟测试结果，此时将应用层产生数据的速率提高到10 Byte/ms，MSS值设定为1 460 Byte。由于应用层每隔146 ms产生一个长度为1 460 Byte的报文段，这样在200 ms时间内就会两次触发Nagle算法的发送条件。一次是由于报文段数据长度达到MSS值，另一次是由于接收到发送端发来的确认，立即发送数据。从表4中可以看出，数据包数据长度和时间间隔基本吻合，没有产生额外的延迟。

通过上述两种处理方式，可以很好地避免Mode 1的交互方式，即避免暂时性“死锁”的产生，使得数据按照Mode 2的方式进行交互。

表4　提高采样频率抓包分析

Table 4　Analysis of capture of sampling frequency improving

包序号	抓包时间/s	时间间隔/s	长度/Byte
1	0.603 612	—	—
2	0.752 416	0.148 804	1 460
3	0.753 417	0.001 001	9
4	0.902 259	0.148 842	1 460
5	0.903 222	0.000 963	9
6	1.052 086	0.148 864	1 460
7	1.053 040	0.000 954	9

5　结语

本文针对嵌入式Internet中产生的大量小数据包所导致的网络带宽利用率低，容易产生拥塞等问题，在深入研究组块技术的基础上，详细分析了Nagle算法的工作机制以及其与延迟确认策略的交互方式。针对“死锁”问题，在嵌入式系统中可以采用提高采样频率或者填充缓冲区的方式，在不修改Nagle算法的基础上，避免暂时性“死锁”的产生，提高带宽利用率的同时保证数据传输的实时性。目前，基于Nagle算法和填充缓冲区、提高采样频率数据处理方式的嵌入式Internet系统已在计量泵远程监控系统中取得了很好的应用。

参考文献

[1] 陈蓉芳，王海滨，胡振华，等. 基于嵌入式Internet技术的电网远程监测系统[J]. 微计算机信息，2003，24(7-2)：78-80.

[2] 周志洪. 基于嵌入式网络技术的网络化自动抄表系统的研究[D]. 杭州：浙江大学，2005.

[3] 钟建国. 嵌入式Internet系统中TCP/IP协议的实现[J]. 陕西师范大学学报：自然科学版，2008，36(4)：16-19.

[4] 吴显伟. 嵌入式协议栈结构优化的研究与实现[D]. 哈尔滨：哈尔滨理工大学，2009.

[5] IETF RFC896-Jan. Congestion control in IP/TCP internetworks [S]. 1984.

[6] Minshall G，Saito Y，Mogu J C，et al. Application performance pitfalls and TCP’s Nagle algorithm[J]. ACM Performance Evalvation Review，2000，27(4)：36-44.

[7] 周志洪，王　勇，陈抗生. 基于Nagle算法的嵌入式TCP协议[J]. 浙江大学学报：工学版，2006，40(1)：41-44.

[8] IETF RFC813-July. Window and acknowledgement strategy in TCP[S]. 1982.

[9] Mogul J C, Minshall G. Rethinking the TCP Nagle algorithm[J]. Computer Communication Review (ACM SIGCOMM), 2001, 31(1):6-20.
[10] 梅小华. 提高51单片机TCP通信效率的软件方法[J]. 华侨大学学报:自然科学版,2011,32(2):235-237.
[11] 陈 立. 对Nagle算法的进一步研究[D]. 上海:复旦大学, 2002.

Nagle algorithm and its application research in embedded Internet

Wang Baobao, Yu Shiming, Wang Zhenyu
(College of Information Engineering, Zhejiang University of Technology, Hangzhou 310023, China)

[Abstract] The existence of small packets in embedded Internet lead to low bandwidth efficiency and even congestion. The Nagle algorithm was applied by standard transmission control protocol(TCP) protocol to reduce the number of small packets. The paper builds embedded Internet network based on ARM7 32 bits micro control unit(MCU) and personal computer(PC), analyses the principle and working mechanism of Nagle, and suggests an approach to resolve the temporary "deadlock" created by the interaction between the Nagle algorithm and the delayed ACK policy without modifying the Nagle algorithm through improving sampling frequency or filling the buffer in embedded system. The experimental results indicate that this approach is effective and reliable.

[Key words] Nagle algorithm; deadlock; delayed ACK policy; ARM7; embedded Internet