浙江理工大学学报，2020，43(4)：535-541
Journal of Zhejiang Sci-Tech University
DOI:10.3969/j.issn.1673－3851(n).2020.04.015

基于压缩和差分算法的嵌入式平台远程更新设计与分析

包晓安，李　刚，文艺霏，李嘉钰，陈迪荣，杜　鹏
（浙江理工大学信息学院，杭州 310018）

摘　要：传统的嵌入式远程更新方案普遍采用整包更新方式，这种方式更新数据量大，占用网络宽带时间长，同时也增加了设备的功耗。针对这些问题，提出了以减少更新数据量为核心的两种远程更新方案。这两种方案分别使用 LZ77 压缩和 BSDiff 差分算法处理更新包，减少需要传输的数据量；同时，在数据传输方面增加断点续传功能，实现终端从断线的地方继续获取数据，以此达到节省设备流量和功耗的目的。另外，设计了一种 FLASH 分区方式，优化了本地自更新操作的流程，去除了多余的拷贝操作。将这两种方案在 STM32F407ZGT6 和 NB-IoT 搭建的嵌入式平台中进行验证测试。实验结果显示，这两种更新方案的平均更新效率与整包更新的方式相比分别提升了 26.44%和 72.17%。

关键词：嵌入式平台；远程更新；LZ77；BSDiff；断点续传
中图分类号：TP319　　**文献标志码**：A　　**文章编号**：1673-3851 (2020) 07-0535-07

Design and analysis of remote update of embedded platform based on compression and difference algorithm

BAO Xiaoan, *LI Gang*, *WEN Yifei*, *LI Jiayu*, *CHEN Dirong*, *DU Peng*
(School of Information Science and Technology, Zhejiang Sci-Tech University, Hangzhou 310018, China)

Abstract: The traditional embedded remote update scheme generally adopts the whole package update method, which has massive update data, takes up a long time of network broadband, and increases the power consumption of the equipment too. Aiming at these problems, two remote update schemes with the core of reducing the amount of updated data were proposed. The two schemes use LZ77 compression and BSDiff differential algorithm respectively to process the updated package so as to reduce the amount of data to be transmitted. At the same time, the function of breakpoint resume was added in the aspect of data transmission so that the terminal could continue to obtain data from the breakpoint, thus achieving the purpose of saving device flow and power consumption. In addition, a FLASH partitioning method was designed to optimize the process of local self-update operation and eliminate redundant copy operation. The two schemes were tested in the embedded platform built by STM32F407ZGT6 and NB-IoT. The experimental results show that the average update efficiencies of these two update schemes are improved by 26.44% and 72.17% respectively compared with the whole package update method.

Key words: embedded platform; remote update; LZ77; BSDiff; breakpoint resume

收稿日期：2019－11－04　　**网络出版日期**：2020－01－02
基金项目：浙江省重点研发计划项目(2020C03094)；浙江省自然科学基金青年基金项目(LQ20F050010)；浙江理工大学本科生科研创新计划重点项目(2019ZD-28)；浙江理工大学本科生科研创新计划一般项目(2019YB-24)
作者简介：包晓安(1973－　)，男，浙江东阳人，教授，硕士，主要从事软件工程、智能信息处理方面的研究。

(C)1994-2020 China Academic Journal Electronic Publishing House. All rights reserved.　http://www.cnki.net

0 引 言

随着 NB-IoT、GPRS 等无线通信技术的发展[1],越来越多的嵌入式无线终端设备应用到了日常生活中,这些设备功能的完善、bug 的修复等都离不开系统固件的更新,为了更好地维护这些终端设备,远程在线更新固件的功能就显得格外重要。但在传统的远程固件更新方案中,通常采用整包更新的方式,尽管新旧更新包之间的代码功能差异不大,但仍然需要通过无线的方式将整个更新包下载到硬件内存中,然后再启动更新,这种更新的方式占用网络宽带时间长,并且在网络拥堵的情况下容易出现丢包、断连的情况,进而影响设备的功耗和升级的效率。因此如何提高远程升级的效率、节约设备功耗成为嵌入式终端设备需要解决的关键问题。

已有很多学者对相关问题进行了研究。马维华等[2]将编译、链接后的模块化文件转化为能用串口传输的二进制文件,实现了通过串口对产品系统进行升级。Jurkovic 等[3]提出了一种自动软件或固件升级的算法,提供更高级别的灵活性,有效防止远程更新中可能出现的故障。魏民等[4]将 LWM2M 协议引入到物联网终端远程升级方案中,降低了终端规模升级对网络冲击,提高远程更新的可靠性和稳定性。唐洪富等[5]提出了通过 GPRS 通信的方式实现对产品的在线远程更新,这种方法取消了通过有线的方式升级系统,提高了产品的可维护性。以上均是采用了整包更新的方式,在优化程序流程的方面实现在线更新或提高设备更新效率。整包更新的方式虽然易于实现,可靠性高,但浪费了较多的网络流量,同时增加了更新过程中所耗费的时间。针对以上问题,本文以减少更新数据量的思想为核心,分别使用 LZ77 压缩和 BSDiff 差分算法处理更新包,提出了一种基于压缩和差分算法的嵌入式平台远程更新系统,同时对无线网络传输和本地自更新流程进行了优化,进一步减少了嵌入式系统更新时需要下载的数据量和安装升级的时间。

1 算法分析

1.1 LZ77 压缩算法

LZ77 是一种基于字典的无损压缩算法[6-9],首先分别定义一个不同长度的前向缓冲区和滑动窗口,待压缩的数据会按顺序进入到前向缓冲区中,假设前向缓冲区中有 n 个数据,它们分别为 $X_1,X_2,\cdots,X_n$,此时这些数据会组成字典 $S_f=\{(X_1),(X_1,X_2),\cdots,(X_1,X_2,\cdots,X_n)\}$。之后,这些数据又会按顺序进入滑动窗口,假设滑动窗口中有 m 个数据,它们分别为 $X_1,X_2,\cdots,X_m$,此时这些数据组成字典 $S_w=\{S_1,S_2,\cdots,S_m\}$,其中 $S_i=\{(X_i),(X_i,X_{i+1}),\cdots,(X_i,X_{i+1},\cdots,X_m)\},1\leqslant i\leqslant m$。

图1为 LZ77 算法中前向缓冲区和滑动窗口的分布示例图,图中字母为例举的待压缩数据。随着前向缓冲区和滑动窗口的移动,字典 S_f 与 S_w 中的内容也会不断改变。每次滑动一位便在字典 S_f 中寻找能与字典 S_w 匹配的最长字符串,若能匹配则标记成由偏移量、匹配数据长度、新字符三个参数构成的编码,否则输出未匹配的字符。LZ77 算法用这些标记代替冗余的数据,进而达到压缩数据的目的。

图1 LZ77 前向缓冲区和滑动窗口示例图

1.2 BSDiff 差分算法

在嵌入式系统中,尽管添加或删除少量的源代码,编译出来的二进制可执行文件仍然会产生很大的差异,这是由于修改过的源代码,经过编译之后,会使原来的代码块和数据块所指向的地址发生大量的变化,但这种变化是有规律可循的,可以由两条规律来总结:

a)其中没有被新代码所影响的区域,变化是很稀疏并且发生变化的字节之间差异较小[10],一般相隔几个字节;

b)其中被影响的区域虽然变化很大,但是大都属于代码块和数据块指针的整体偏移[11],并且这个偏移值为一个固定值。

为了展示这两个规律,图2列举了插入一行代码之后新旧可执行文件内容字节的差异。由图2可知,将新旧两个可执行文件的内容进行对比之后,有许多字节的差异为0,因此可以得知差异文件是具有高度可压缩性[12]。

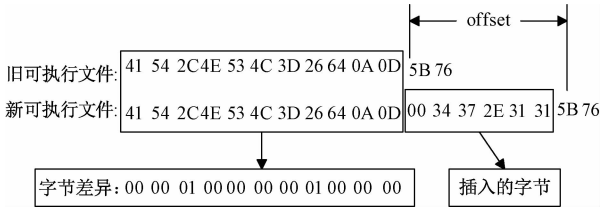

图2 新旧可执行文件的内容差异示意图

因此,BSDiff 差分算法的主要思想可以分为以下几个步骤:

a)将旧文件的数据进行后缀排序[13];

(C)1994-2020 China Academic Journal Electronic Publishing House. All rights reserved. http://www.cnki.net

b)使用二分查找找到能匹配的最长字符串的长度[14]，并计算出差异部分和新增部分；

c)将差异部分和新增部分进行压缩，生成差异 Patch 包。

2　嵌入式平台的远程更新方案及实现

2.1　远程更新系统方案优化

为解决传统整包更新方案中数据量大、网络下载的时间长、高能耗的问题，本文以减少更新包大小的为核心，对传统整包更新的方案进行了优化，设计了以下两种远程更新方案。

方案一以 LZ77 压缩算法为核心，其整体框图如图 3 所示。首先将新版本的可执行文件放至服务器，运行 LZ77 压缩算法，将可执行文件中的数据进行数据压缩生成压缩包；其次通过 NB-IoT 无线网络将压缩包下发至终端设备，嵌入式终端设备收到数据之后，会运行 LZ77 解压程序，将数据还原成完整更新包，接下来由终端设备校验更新的数据；最后执行重启更新。

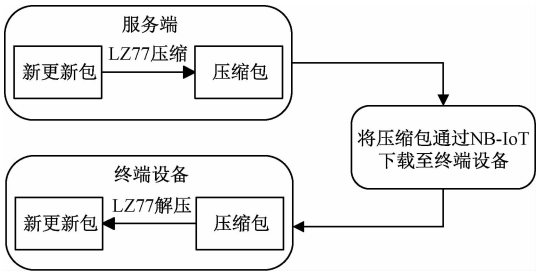

图 3　LZ77 远程更新整体框图

方案二以 BSDiff 差分算法为核心，其整体框图如图 4 所示。首先服务器需要保存上一版本的升级包，每当有终端设备需要升级时，先将新版本升级包放至服务器，由服务器运行 BSDiff 程序，将新旧更新包里的数据进行比对运算，得出 2 个版本之间的 Patch 补丁包[15]，再通过 NB-IoT 无线网络将补丁包的数据发送至终端设备；设备终端收到补丁包之后，会运行 BSPatch 程序将旧二进制更新包的数据与补丁包的数据进行合并运算，生成完整的新更新包，由终端设备校验更新的数据；最后执行重启更新程序完成更新。

2.2　通信协议设计

本嵌入式平台远程更新系统主要由服务器端和设备终端两大部分组成，其中服务器端主要负责三个工作：a)保存更新包；b)生成更新文件；c)与设备终端进行数据交互。设备终端选用意法半导体公司的 STM32F407ZGT6 芯片(以下简称 STM32)作为

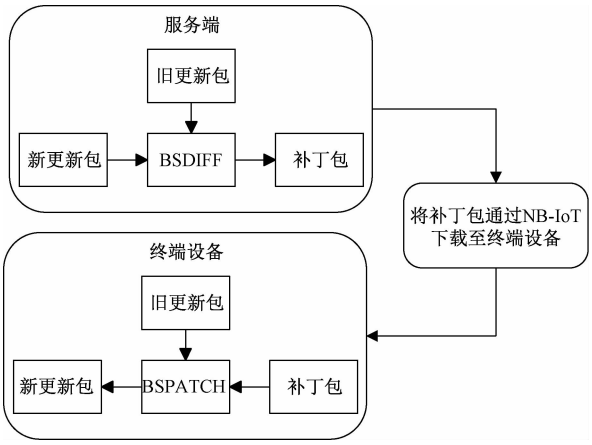

图 4　BSDiff 远程更新整体框图

MCU，该芯片拥有 1 MB 的片内 FLASH、192 KB 的 SRAM 和最高 168 MHz 的时钟频率，系统另选采用移远 BC26 模组作为 NB-IoT 无线通信模块。

BC26 模组具有丰富的外部接口(UART、SPI、ADC 等)和多种网络协议栈(TCP、CoAP、MQTT 等)[16]，主要使用 AT 指令的方式执行任务。为此，本文将 STM32 和 BC26 的 UART 连接起来，采用面向连接的 TCP 协议进行数据通信，其中主要用到的相关 AT 指令如表 1 所示。

表 1　BC26 模组 AT 通信指令及说明

指令	指令说明
AT+QSOC=1,1,1	创建 TCP socket
AT+QSOCON=0,Port,"IP"	连接指定端口和 IP 的服务器
AT+QSOSEND=0,Len,Data	向服务器发送长度为 Len 的数据
AT+QSORF=0,Len	读取长度为 Len 数据
AT+QSODIS=0	断开与服务器的连接
AT+QSOCL=0	关闭 TCP socket

当 STM32 通过串口向 BC26 模组发送 AT 指令时，BC26 模组会通过串口返回"OK"、"NULL"等信息告知 STM32 刚刚执行的 AT 指令是否已经成功执行。

STM32 上电之后通过串口发送相应的 AT 指令让 BC26 与服务器建立 TCP 连接，并主动查询是否有新版本。发现有新版本后，继续向服务器请求更新包的数据。本文设计了终端与服务器之间的通信协议，表 2 为终端向服务器查询升级协议包的组成结构，包含协议头、协议长度、操作码等。其中协议长度为一帧数据包中不包含校验码的总长度，校验码为一帧数据包中不包含校验码本身的总数据的和(字节数取最低 1 byte)，协议长度和校验码的数据用来校验接收到的数据包是否有效。

(C)1994-2020 China Academic Journal Electronic Publishing House. All rights reserved.　http://www.cnki.net

表2 查询升级协议组成

设备	字段名	字节数/byte
终端	协议头	1
	协议长度	2
	操作码	1
	当前版本号	2
	校验码	1
服务器端	协议头	1
	协议长度	2
	操作码	1
	升级包总大小	2
	校验码	1

在服务器返回的数据协议中,当升级包总大小这一位的值为0x0000时,表示服务器端没有新版本不需要升级;当升级包总大小这一位的值不为0x0000时,表示需要升级并且它实际的值为升级包的总字节数,这时终端会根据表3中请求升级数据的协议包格式继续向服务器请求升级包的数据。

表3 请求升级数据协议组成

设备	字段名	字节数/byte
终端	协议头	1
	协议长度	2
	操作码	1
	单包数据长度	2
	包序号	2
	校验码	1
服务器端	协议头	1
	协议长度	2
	操作码	1
	包序号	2
	升级包数据	N
	校验码	1

由于BC26模组在每一帧的发送和接收数据量上有限制,因此在代码中设置终端每包向服务器端请求的单包数据长度为0x0100(256 byte),即每次向服务器端请求256 byte的升级包数据,而最后一个升级包按实际长度传输;然后,在每传完一帧数据后,终端将这一帧数据进行字符串处理,取出升级包数据,直到所有的数据包传输完毕;最后,根据本文提出的两种方案,分别运行LZ77解压或者BSPatch程序还原出新更新包,再由终端进行数据校验、拷贝等操作。

2.3 IAP流程优化

在应用编程(In application programming, IAP)支持用户使用引导程序对MCU的片内FLASH进行擦除烧写,从而更新本地应用程序。传统IAP的原理是将MCU的片内FLASH划分为Bootloader引导区和User主程序区两个区域,还需借助片外FLASH来保存所有更新包的数据,其更新操作步骤如下:

a)将所有更新包的数据保存至片外FLASH;

b)执行重启,进入Bootloader引导程序;

c)将保存在片外FLASH的更新包数据拷贝至片内User主程序区;

d)跳转至新程序入口;

e)下次更新重复以上操作。

本系统对IAP功能进行了优化,将STM32的片内FLASH划分为三个区域,如图5所示,分为一个Bootloader引导区和两个User主程序区,并分别定义User1与User2 FLASH的起始地址。优化过后的IAP采用User1和User2主程序区交替更新的方式,优化过后的更新操作步骤如下:

a)将所有更新包的数据直接保存在User2中,并设置标志位;

b)执行重启,进入Bootloader引导程序;

c)跳转至User2主程序入口;

d)下次更新将更新包的数据保存在User1中,设置标志位,执行b)步骤,之后跳转至User1主程序入口。

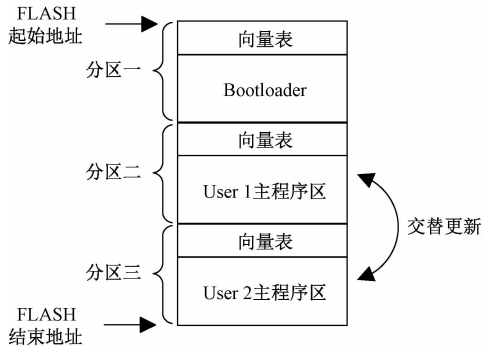

图5 STM32片内FLASH划分

在FLASH之间的数据交换中,大部分时间会花在数据的擦除和拷贝上,优化后的IAP不仅节约了片外FLASH空间的使用,而且去掉了更新包数据拷贝的这一个环节,进一步减少了本地自更新所需要的时间。

2.4 嵌入式平台远程更新整体实现步骤

基于压缩和差分算法的嵌入式平台远程更新整体流程如图6所示。首先通过LZ77解压缩算法或者BSDiff差分算法来减少更新包的大小;其次在无线网络传输数据的环节增加了断点续传功能,在出现断电、网络断开等异常情况时,将前一个的升级包

(C)1994-2020 China Academic Journal Electronic Publishing House. All rights reserved. http://www.cnki.net

序号写入片外 FLASH 中，等再上电或者网络状态好的情况下，继续从记录的升级包序号开始请求升级包数据，无需从头开始请求升级包数据；最后通过优化本地执行 IAP 功能，节省了 FLASH 之间数据拷贝时间。通过这三个步骤，极大地节省网络带宽和终端设备的功耗，并提升了系统更新的效率。

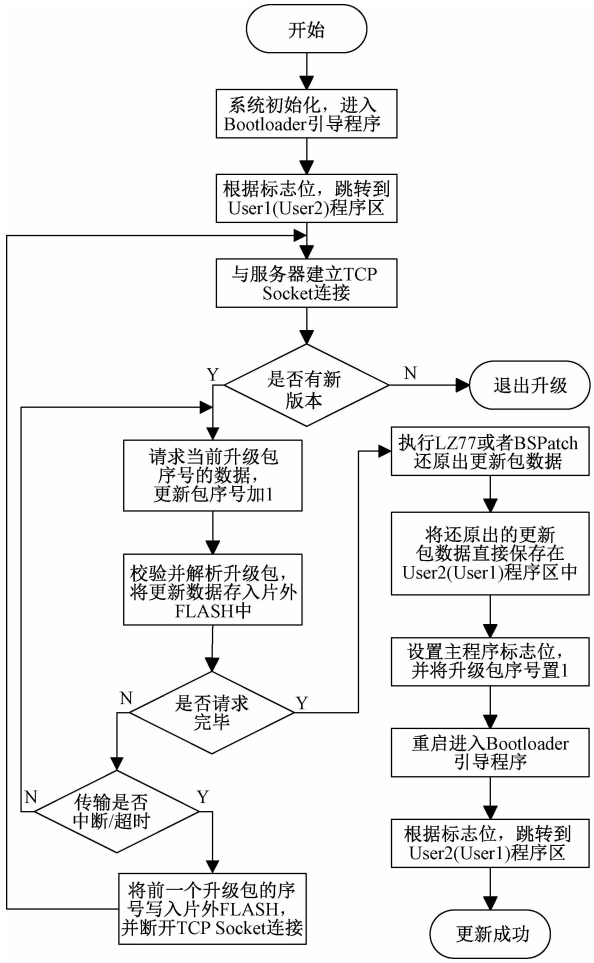

图 6　基于嵌入式平台远程更新流程

基于压缩和差分算法的嵌入式平台远程更新系统的具体步骤为：

步骤 1：系统在上电之后，对 GPIO、时钟、串口等功能进行初始化，保证系统可以正常运行，进入 Bootloader 引导程序。

步骤 2：根据主程序的标志位，跳转至 User1(User2)程序区启动。

步骤 3：STM32 通过串口向 BC26 模组发送相应的 AT 指令，让其创建 TCP Socket 并连接服务器。

步骤 4：连接完成之后，按照表 2 的通信协议将终端当前的软件版本号的信息发送给服务器端，在服务器端返回的应答包当中，若升级包总大小的值为 0x0000 时，表明不需要升级，随后将退出升级；若值不为 0x0000 时，则表明需要升级并执行步骤 5。

步骤 5：按照表 3 的通信协议和当前片外 FLASH 中记录的升级包序号，向服务器端请求升级包数据并将升级包序号加 1，随后通过解析服务器端返回数据帧的升级包数据这位的数据，并将解析出来的数据通过 Flash__Write(wBuf, WriteAddr, NumByteToWrite)函数写入片外 FLASH 中，其中 wBuf 为要储存的升级包数据，WriteAddr 为准备存入的起始地址，NumByteToWrite 为要存储的总字节数。

步骤 6：用升级包数据总量除以 256 byte 计算一共需要请求多少个数据包，若所有升级包都已经请求完，将会执行步骤 8，否则执行步骤 7。

步骤 7：没有请求完时会进一步判断，若是因为断电、信号不好等异常造成通信中断或超时，则会记录前一个升级包的序号，将其写入片外 FLASH 并断开 TCP Socket 连接，跳转步骤 3；若传输没有中断或超时，则会跳转步骤 5 进行下一个升级包的请求。

步骤 8：当所有的升级包请求完毕之后，会根据对应的实验方案执行 LZ77__Decode 解压程序或者 BSPatch 程序还原出更新包。

步骤 9：将还原出的所有的更新包数据使用 Iap__WriteBin(BinAddr, BinBuf, BinSize)函数保存到 User2(User1)主程序中，其中 BinAddr 为 User2(User1)主程序区的起始地址，BinBuf 为完整的更新包数据，BinSize 为更新包数据的总长度。

步骤 10：保存完毕之后，设置主程序的标志位，并将升级包序号重置为 1。

步骤 11：将系统重启，随后又进入 Bootloader 引导程序，根据刚刚设置的主程序标志位，执行 Iap__LoadBin(BinAddr)函数初始化堆栈指针并跳转到值为 BinAddr 的用户程序区中[17-18]，即完成本次升级并记录该软件版本号。

3　实验及结果分析

本系统使用 STM32 与 BC26 模拟嵌入式终端设备，在云服务器上运行 TCP 服务器和 MySQL 数据库，实验开始运行之前将新版本的升级包数据存入数据库中，并将 STM32 的串口 1 与电脑相连，在电脑上运行串口调试助手实时显示日志，根据日志判断 STM32 的运行状态。本实验平台实物图如图 7 所示。

(C)1994-2020 China Academic Journal Electronic Publishing House. All rights reserved.　http://www.cnki.net

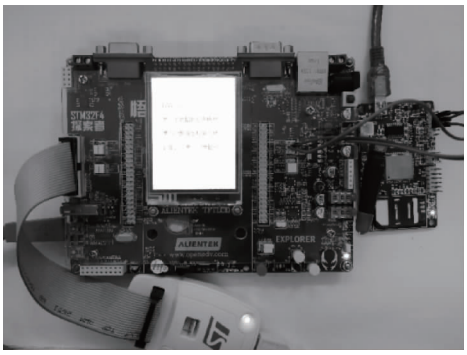

图7 实验平台实物照片

本系统测试了三组不同大小的源代码,对每组进行了以下三种不同的更新实验,并使用优化后的IAP流程:

a)使用传统的整包更新方式,将更新包的所有数据传输至终端。

b)使用LZ77压缩算法处理更新包,仅将压缩包数据传输至终端。

c)使用BSDiff差分算法处理更新包,生成补丁包,仅将补丁包数据传输至终端。

上述所有测试用例中,数据压缩率、更新执行时间等都是十分重要的指标,现定义压缩率 R_{compr} 为式(1):

$$R_{\mathrm{compr}}/\% = \frac{L_{\mathrm{total}} - L_{\mathrm{uncompr}}}{L_{\mathrm{total}}} \times 100 \qquad (1)$$

其中:L_{total} 为升级包的总字节数,L_{uncompr} 为需要传输的总字节数。

所有测试用例均在相同的网络环境下执行,将三组不同的源代码进行实验,每组别之间的新旧版本的代码差异呈递增关系,每组测试用例测试20次,实验结果如表4所示。

表4 实验结果对比

实验组别	更新方案	需传输字节数/byte	压缩率/%	平均更新时间/s
组1	整包更新	25463	0	23.87
	LZ77 + IAP优化	19462	23.57	17.98
	BSDiff + IAP优化	2450	90.38	3.19
组2	整包更新	28003	0	27.35
	LZ77 + IAP优化	21019	24.94	20.21
	BSDiff + IAP优化	6914	75.31	7.49
组3	整包更新	37156	0	37.74
	LZ77 + IAP优化	27100	37.06	26.97
	BSDiff + IAP优化	15492	58.89	16.13

使用LZ77解压缩算法处理升级包的方案,压缩率在23.57%到27.06%之间,与整包更新相比更新效率平均提高了26.44%。

使用BSDiff差分算法处理升级包的方案,从数据结果可知,当新旧源代码差异较小时,该方案压缩率是使用LZ77压缩算法的3.8倍,之后随着新旧源代码之间的差异变大,得到的差分补丁包的大小就越大,压缩率就越来越低,但仍然拥有较高的压缩率,数值在58.89%到90.38%之间,与整包更新相比更新效率平均提高了72.17%。

从实验结果与分析可知,本文中提出的两种方案均可以提高终端在远程升级中的效率,即节省了终端在远程更新中的网络流量开销和更新时间,达到了预期结果。

4 结束语

本文研究了嵌入式平台下的远程更新,综合考虑了嵌入式终端性能弱、内存小的特点,对传统整包更新的方式和IAP自更新流程进行改进,提出了以LZ77压缩和BSDiff差分算法两种方案来减少更新包的大小,借此达到节约网络资源、终端功耗和内存的目的,并对这两种方案进行了验证,相应的实验结果表明已取得了预期成果。使用BSDiff差分算法处理更新包的远程更新系统已投入智慧校园项目中,且运行效果较好,但考虑到拥有更庞大的更新包时,怎么进一步优化算法以提升数据压缩率和减少空间、时间复杂度也是另一个重要的研究课题。

参考文献:

[1] Moon Y, Ha S, Park M, et al. A methodology of NB-IoT mobility optimization[C]// 2018 Global Internet of Things Summit (GIoTS). Bilbao: IEEE, 2018: 1-5.

[2] 马维华,赵怀林,朱纪洪,等.电推进系统计算机的在线优化升级研究与设计[J].现代电子技术,2019,42(6):38-42.

[3] Jurkovic G, Sruk V. Remote firmware update for constrained embedded systems[C]//2014 37th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO). Opatija: IEEE, 2014: 1019-

(C)1994-2020 China Academic Journal Electronic Publishing House. All rights reserved. http://www.cnki.net

1023.

[4] 魏民，王艺.物联网云平台终端远程更新技术研究与应用[J]. 电信科学，2018，34(10)：137-142.

[5] 唐洪富，王肖楠.基于 GPRS 通信的 MCU 固件远程更新技术的实现[J].微型机与应用，2016，35(14)：77-78.

[6] Ziv J，Lempel A. A universal algorithm for sequential data compression[J]. IEEE Transactions on Information Theory，1977，23(3)：337-343.

[7] 许晓飞，陈亮. 应用整数小波变换的 LZ77 电力数据压缩算法[J]. 西安工程大学学报，2018，32(3)：337-342.

[8] 刘爱东，李知宇，王丰，等. 一种支持嵌入式标校系统的数据压缩算法[J]. 计算机与数字工程，2018，46(12)：2607-2610.

[9] 李冰，王超凡，顾巍，等. Gzip 压缩的硬件加速电路设计[J]. 电子学报，2017，45(3)：540-545.

[10] 方兵兵. 基于 STM32 的嵌入式软件远程升级研究[D]. 宁波：宁波大学，2017：14-20.

[11] 陈志龙，倪桂强，姜劲松，等. 基于动态字典的增量更新算法[J]. 解放军理工大学学报(自然科学版)，2015，16(5)：426-432.

[12] 王栋梁，汤利顺，陈博，等. 智能网联汽车整车 OTA 功能设计研究[J]. 汽车技术，2018(10)：29-33.

[13] Onuma Y，Nozawa M，Terashima Y，et al. Improved software updating for automotive ECUs：Code Compression[C]// 2016 IEEE 40th Annual Computer Software and Applications Conference (COMPSAC). Atlanta：IEEE，2016：319-324.

[14] 张敏，韩俊刚，李涛. 基于 Android 平台差异化增量更新的实现[J]. 西安邮电大学学报，2014，19(1)：82-85.

[15] Teraoka H，Nakahara F，Kurosawa K. Incremental update method for resource-constrained in-vehicle ECUs[C]// 2016 IEEE 5th Global Conference on Consumer Electronics. Kyoto：IEEE，2016：1-2.

[16] 宋洪儒，王宜怀，杨凡. 基于窄带物联网智能燃气表系统设计与实现[J]. 传感器与微系统，2019，38(3)：113-116.

[17] 徐立国，李德建，于宝东，等. 一种支持在线升级的 NOR Flash 控制器设计[J]. 电子技术应用，2019，45(10)：50-57.

[18] 蒋存波，焦阳.基于 STM32 处理器的 WSN 节点固件在线系统升级方法[J]. 计算机应用与软件，2016，33(5)：222-225.

(责任编辑：康　锋)

(C)1994-2020 China Academic Journal Electronic Publishing House. All rights reserved.　http://www.cnki.net