文章编号：2096-1472(2018)-08-33-03
DOI:10.19644/j.cnki.issn2096-1472.2018.08.010

一种简化mbuf的设计与实现

张雪锋
(国家知识产权局专利局专利审查协作四川中心，四川 成都 610011)

摘　要：mbuf全称为“memory buffer”，主要用于保存进程和网络接口间互相传递的用户数据，也用于保存源与目标地址，套接字选项等。本文基于TCP/IP协议栈中mbuf的设计思想，设计了一种简化的mbuf，提供给应用软件能方便的操作可变长缓存、在缓存的头部和尾部添加协议头数据、从缓存中移除数据，同时通过内存的零拷贝技术，有效地提高CPU的利用率、节省存储空间的占用，并且可移植性强，具有较高的实用价值。

关键词：TCP/IP；mbuf；零拷贝；嵌入式
中图分类号：TP3.0　**文献标识码**：A

The Design and Implementation of a Simplified Mbuf

ZHANG Xuefeng
(*Patent Examination Cooperation Sichuan Center of Patent Office,Chengdu* 610011,*China*)

Abstract:A Memory Buffer (mbuf) is mainly used for saving the user data transmitting between the process and the network interface,as well as for storing sources and target addresses,and socket options.This article,based on the concept of mbuf in the TCN/IP protocol stack,proposes a simplified mbuf,providing an easily operated variable length buffer,adding protocol data at front and back of the buffer,and removing data from the buffer.By implementing the technology of memory zero-copy,the MBUF efficiently promotes CPU utilization,saves memory capacity,and increases portability and the practical value.

Keywords:TCP/IP;mbuf;zero-copy;embedded

1 引言(Introduction)

在有数据通信需求的软件系统中，数据包的存储结构，以及传递方式直接影响到整个软件系统的处理性能。有效的数据包存储管理可以实现内存的“零拷贝”(zero-copy)。零拷贝技术可以减少数据拷贝和共享总线操作的次数，消除通信数据在存储器之间不必要的中间拷贝过程，从而有效地提高通信效率[1]。存储器管理最经典的当属TCP/IP协议栈中的mbuf。mbuf全称为“memory buffer”，主要用途是保存在进程和网络接口间互相传递的用户数据，但也用于保存其他各种数据，如源地址和目标地址、套接字选项等[2,3]。使用mbuf进行报文的传输，接收报文并把报文上送上层应用的过程中，报文传输是“零拷贝”，即不需要拷贝报文内容，只需要传送mbuf地址。

TCP/IP协议栈的复杂性决定了mbuf设计的复杂性，而在我们的更多数据通信软件应用中，其实仅需要精简的mbuf支持，比如能方便的操作可变长缓存，能在缓存的头部和尾部添加协议头数据(如下层封装来自上层的协议数据)、能从缓存中移除数据(如上层解封装来自下层的协议数据)，能实现内存的零拷贝[4]。基于此需求，本文秉承TCP/IP协议栈的mbuf设计思想，重新设计了一种简化的mbuf库，将其组织成链表的形式装载数据报文，采用零拷贝技术在协议栈各层间进行传递。

2 mbuf数据库设计(Design of mbuf database)

本文中设计的mbuf库，采用mbuf头部管理信息和数据存储区进行分开管理的思想，这样的设计优点是，在申请的数据空间不够时，可以动态的获取更大的数据存储空间。为mbuf库设计两种数据库资源，一种是mbuf节点资源池，mbuf节点用于存放mbuf管理信息；一种是mbuf数据区资源池，mbuf数据区资源用于存放真正的应用数据。

2.1 mbuf节点

mbuf节点即提供给用户申请使用的mbuf资源，主要存放一些管理信息，mbuf节点的数据区用于存放用户应用数据，在mbuf资源申请时进行动态分配，mbuf节点数据结构主要字段设计如图1所示。

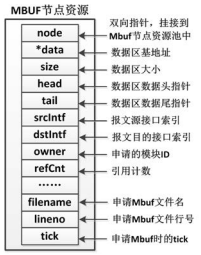

图1 mbuf节点

Fig.1 Mbuf node

2.2 mbuf节点资源池

mbuf节点资源池由“mbuf空闲链表”进行管理，“mbuf空闲链表”是一个双向链表，所有未使用的mbuf节点，均通过自身的node双向指针挂接到“mbuf空闲链表”中，mbuf节点资源池在初始化时一次申请多个mbuf，申请的mbuf个数和长度由用户指定，如图2所示。

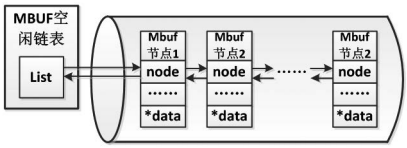

图2 mbuf节点资源池

Fig.2 Mbuf node pool

2.3 mbuf数据区

mbuf数据区用于存放真正的应用数据，数据结构主要字段设计如图3所示。其中首尾部魔术字用于意外踩内存时的协助定位，mbuf节点的结构与linux内核协议栈的skb_buf相似[5]，在保存报文的内存块前后分别保留headroom和tailroom，以方便应用解封报文，headroom默认128字节，可以通过宏进行调整。数据区即真正存放用户数据的区域，大小不小于申请的size的长度。

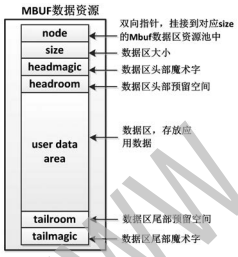

图3 mbuf数据区

Fig.3 Mbuf data

2.4 mbuf数据区资源池

mbuf数据资源池，由“mbuf数据区资源池数组”进行管理，每一类mbuf数据区资源池中，均通过双向链表list挂接所有具有相同字节长度的数据资源，如图4所示。

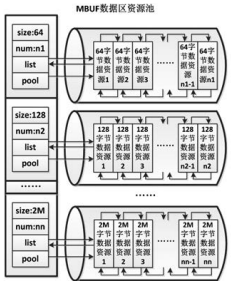

图4 mbuf数据区资源池

Fig.4 Mbuf data pool

3 mbuf功能(Function of mbuf)

3.1 mbuf初始化

使用mbuf库的应用软件应根据自身的使用需求进行mbuf资源配置，即要使用的每一类mbuf的字节长度，以及对应的mbuf个数，如表1所示。

表1 mbuf配置表

Tab.1 Mbuf configuration

序号	字节长度	个数
1	64字节	n1
2	128字节	n2
3	256字节	n3
……	……	……
n	2M字节	nn

3.2 mbuf申请

mbuf申请时，从“mbuf空闲链表”尾部获取出一个空闲的mbuf节点资源，然后依次遍历mbuf数据资源池，查找一个最小的大于申请字节长度的数据资源，并挂接到mbuf节点资源的数据区指针上，申请成功的mbuf节点资源如图5所示。

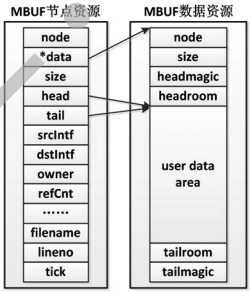

图5 mbuf申请

Fig.5 Mbuf malloc

申请成功后的mbuf节点资源与mbuf数据区资源建立了一定的链接关系，mbuf节点资源中的data指针将指向mbuf数据资源头。mbuf节点资源中head字段指向当前数据区头，tail字段指向当前数据区尾，初次申请成功，head和tail均指向数据头部。

3.3 mbuf释放

mbuf释放时，需要释放两类资源，首先将mbuf数据资源释放到对应字节长度的mbuf数据资源池中，然后清除mbuf节点资源中的管理信息[5]，并将mbuf节点资源释放到mbuf空闲链表中。mbuf释放时，只减引用计数，只有当引用计数为零时，才真正释放mbuf资源：

3.4 mbuf数据存储

为了增加应用程序的使用灵活性，应用程序向mbuf中存放应用数据时，提供两种方式存储。一种是使用mbuf提供的接口进行安全存储，还有一种是直接提供mbuf的数据区指针给用户，由用户自行操作mbuf数据区指针，这种方式就要求自行控制写入的数据长度，避免越界。

3.4.1 mbuf头部数据追加

使用TCP/IP协议栈[6]向网络上发送报文时，当本层协议接收到上层协议报文后，将在报文头部封装本层协议头，比如从IP层到数据链路层再到物理层，至上而下，逐层进行协议封装，基于此应用场景，此功能提供给用户，在当前mbuf数据区首部追加指定长度的数据。如图6所示。

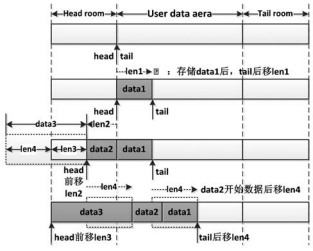

图6 mbuf头部数据追加

Fig.6 Mbuf append at head

头部数据追加使用的是头部预留空间，当头部预留空间不足时，总空间足够时，首先会将数据整体向后移动，在头部留出足够的空间扩展数据。当头部预留空间不足，并且用空间也不足时，此时将重新为mbuf节点申请一个更大的数据资源，然后在头部追加指定长度的数据，同时旧的数据资源将被释放掉。

3.4.2 mbuf尾部数据追加

此功能提供给用户，将当前数据拷贝到mbuf数据区尾部，实现原理同头部空间追加，尾部空间扩展时，如果尾部预留空间不足而用空间足够时，同样会将数据整体向前移动，然后再将数据追加到尾部。如果尾部空间不足且总空间也不足，此时会申请更大的数据区资源，然后再进行尾部数据追加，同时旧的数据区资源将被释放掉[7]。

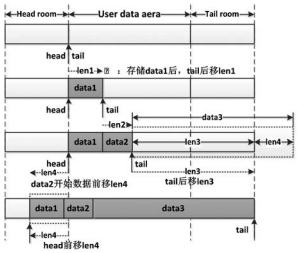

图7 mbuf尾部数据追加

Fig.7 Mbuf append at tail

3.4.3 mbuf头部数据移除

使用TCP/IP协议栈从网络上接收报文时，当本层协议接收到下层协议报文后，将从报文头部摘除本层协议头，比如从物理层到数据链路层再到IP层，至下而上，逐层进行协议解封装，基于此应用场景，此功能提供给用户，在当前mbuf数据区首部移除指定长度的数据。

3.4.4 mbuf尾部数据移除

此功能提供给用户，从mbuf数据区尾部移除指定长度的数据。

3.5 mbuf拷贝

3.5.1 mbuf深拷贝

重新申请一个跟当前mbuf一样size的mbuf，并将原有mbuf中的数据和管理信息一一拷贝到新申请的mbuf中，此时新申请的mbuf是一个新的资源，地址不一样。

3.5.2 mbuf浅拷贝

在实际使用中，有可能出现一个mbuf报文需要上送给多个应用使用，每一个应用处理完之后自行释放mbuf，此种应用场景提供mbuf浅拷贝机制mbuf资源使用同一个资源，地址不变，在一次浅拷贝时只增加mbuf的引用计数，当每一个应用处理完毕释放mbuf时，会将引用计数减1，直到最后一个使用mbuf的应用释放mbuf时，引用计数为零，才真正执行释放mbuf相关资源的动作。

3.6 mbuf安全设计

在使用mbuf的系统中，难以避免使用了mbuf之后没有释放或释放不及时而导致的资源泄漏[7,8]，严重的将会引起整个系统mbuf耗尽，所以安全性设计是mbuf设计中重点需要考虑的。有效的调试手段，可以准确地定位到代码中哪一行没有释放mbuf。

3.6.1 查看mbuf节点资源池

在使用mbuf库的应用软件中最常见的错误使用方法是申请mbuf后未释放而导致的资源泄露，如果资源泄露的地方较多，最终将导致mbuf资源耗尽。mbuf库提供的调试命令可随时查看，监控mbuf资源池的使用情况[8]。图8是以一个测试应用程序申请mbuf后未释放的情况为例，展示了在测试程序的运行过程中，通过在shell窗口下面敲入mbuf调试命令，查看到的mbuf节点资源池使用情况及未释放mbuf的文件名和列号。

```
showMbuf
(tShell0): system Mbuf information:
(tShell0): ------------------------------------------------------------------------------
(tShell0): Mbuf Node pool:
(tShell0):      node number: 1024
(tShell0):      node free: 1023
(tShell0):      node used: 1
(tShell0): ------------------------------------------------------------------------------
(tShell0): Mbuf Node Used[1]:
(tShell0): owner   size    address        position
(tShell0):     2    128    0x5cd2a90    D:/Test/src/test.c Line:220
(tShell0): ------------------------------------------------------------------------------
```

图8 查看mbuf节点资源池

Fig.8 Show Mbuf node pool

3.6.2 查看mbuf数据资源池

图9也是以测试程序为例，在shell窗口下查看mbuf数据资源池命令使用情况，通过显示的详细信息，可以通过地址查看数据内容，通过首尾魔术字判断mbuf资源是否被踩内存。

```
showMbufData 1
(tShell0): the 1 data pool information:
(tShell0): size: 256
(tShell0): total: 128
(tShell0): pool: 0x5cd2a80
(tShell0): free: 127
(tShell0): used: 1
(tShell0): ------------------------------------------------------------------------------
(tShell0): index   size   address     headmagic    tailmagic
(tShell0):     0:    256   0x5cd2a90   0xabbaabba   0xcddccddc
(tShell0):     1:    256   0x5cd2b64   0xabbaabba   0xcddccddc
(tShell0):     2:    256   0x5cd2c38   0xabbaabba   0xcddccddc
(tShell0):    ...:    ...    ...          ...          ...
(tShell0):   127:    256   0x5cd93bc   0xabbaabba   0xcddccddc
(tShell0): ------------------------------------------------------------------------------
```

图9 查看mbuf数据资源池

Fig.9 Show Mbuf data pool

4 结论(Conclusion)

本文设计的简化mbuf库，目前已较为广泛的应用于内部使用的嵌入式软件中，支持ARM、PPC、DSP各个处理器，同时PC机软件也支持使用。本文中对mbuf的简化设计策略，即满足了应用软件的方便的操作可变长缓存的使用需求，同时也基于零拷贝的思想，有效地提高CPU的利用率、节约存储空间的占用，并且可移植性强，具有较高的实用价值。

参考文献(References)

[1] Lu Hai,LiqingLi,Xudong.Mbuf Optimizing Implementation Applied in Embedded System[J].IEEE International Conference on Network Infrastructure and Digital Content,2014,4(3):503-506.

[2] W.Richard Stevens.TCP/IP详解,卷2:实现[M].北京:人民邮电出版社,2016.

[3] Chengcheng Yang,Peiquan Jin.Efficient Buffer Management for Tree Indexes on Solid State Drives[J].International Journal of

(下转第29页)