

# Bug修改记录 10.28

## 1.18946: LTcpSocket的服务端和客户端之间传输中文数据，出现数据丢失

- 原因: LString 的 length() 接口返回中文多字节字符的长度是時候出现了问题(不是网络模块的问题)

```
void test2() {  
    LString s("您好,我是服务端");  
    std::cout << s.length() << std::endl;           // 8  
    std::cout << s.toStdString().size() << std::endl; // 22  
}
```

- 解决: 在 LTcpSocket 类中修改 sends() 函数的代码, 改用 std::string().size(), udp 部分也同步修改

```
9 /**  
10 * @brief 发送数据  
11 * @param data const LString &类型, 用于指向需发送的数据  
12 * @return true, 发送成功  
13 * @return false, 发送失败  
14 */  
15 bool LTcpSocket::sends(const LString &data) {  
16     // return sends(data.toStdString().c_str(), data.length());  
17     return sends(data.toStdString().c_str(), data.toStdString().size());  
18 };  
19  
20 /**
```

```
*/  
bool LUDPsocket::sends(const LString& data, LHostAddress add, uint16_t port) {  
    setPeerAddress(&add);  
    setPeerPort(port);  
  
    // return sends(data.toStdString().c_str(), data.length());  
    return sends(data.toStdString().c_str(), data.toStdString().size());  
};
```

- 测试: 在 LTcpDemo 的 test2() 函数中

```
lzx0626@DavidingPlus:~/DavidingPlus/Lark5/larksdk/build$ snippet/LTcpsServerDemo/LTcpsServerDemo  
25  
您好,我是客户端.  
lzx0626@DavidingPlus:~/DavidingPlus/Lark5/larksdk/build$
```

```
lzx0626@DavidingPlus:~/DavidingPlus/Lark5/larksdk/build$ snippet/LTcpsClientDemo/LTcpsClientDemo  
server_ip: 127.0.0.1  
server_port: 9999  
您好,我是服务端.  
lzx0626@DavidingPlus:~/DavidingPlus/Lark5/larksdk/build$ |
```

## 2.18836: LAbstractSocket的setbufferSize和bufferSize的注释 与实际功能不太相符

- 原因: "接收"敲错了

- 解决：头文件和源文件的对应部分已修改

```
/**
 * @brief 返回接收缓冲区大小
 * @return int类型，返回接受缓冲区的大小
 */
int bufferSize();

/**
 * @brief 设置接收缓冲区大小
 * @param size 想要设置的接受缓冲区大小
 * @return true 设置成功
 * @return false 设置失败
 */
bool setBufferSize(int size);
```

```
/**
 * @brief 返回接收缓冲区大小
 * @return int类型，返回接受缓冲区的大小
 */
int LAbstractSocket::bufferSize() {
    return pData->buffersize;
}

/**
 * @brief 设置接收缓冲区大小
 * @param size 想要设置的接受缓冲区大小
 * @return true 设置成功
 * @return false 设置失败
 */
bool LAbstractSocket::setBufferSize(int size) {
    pData->buffersize = size;
    return true;
}
```

### 3.18947: LTcpSocket的客户端buffer=0 调用receives接收服务端的消息后 程序抛出异常并崩溃

- 疑问：程序走到这个地方，我提前做了判断并且抛出异常的操作，如果符合我的判断，程序理应抛出异常并且结束啊，个人认为这一条 bug 不是很合理，当然我自己做了捕获异常的话那当然是没问题的
- 测试：在LTcpDemo的test2()中我设置了设置缓冲区大小为0，做了异常处理，异常被正常捕获，程序后续也正常进行

```
lzx0626@DavidingPlus:~/DavidingPlus/Lark5/larksdk/build$ snippet/LTcpsServerDemo/LTcpsServerDemo
缓冲区大小 0 不合理,请检查并且重试!
0
lzx0626@DavidingPlus:~/DavidingPlus/Lark5/larksdk/build$
```

```
lzx0626@DavidingPlus:~/DavidingPlus/Lark5/larksdk/build$ snippet/LTCPCClientDemo/LTCPCClientDemo
server_ip: 127.0.0.1
server_port: 9999
您好，我是服务端。
lzx0626@DavidingPlus:~/DavidingPlus/Lark5/larksdk/build$
```

## 4.18948: 建议给LAbstractSocket的buffer初始化一个合适的大小

- 解决: 设置为 C 标准 IO 库给出的缓冲区大小 BUFSIZ ( 8192 )

```
/**
 * @brief 结构类型，用于存储套接字使用到的信息数据
 */
struct AbstractSocketData {
    SocketType type = UnknownSocketType;
    NetworkProtocol protocol = UnknownNetworkLayerProtocol;
    int buffersize = BUFSIZ;
    // int buffersize = 0;
    // char *bytebuffer = nullptr;
    // int end = 0;
    LString bytebuffer;
};
```

- 测试: 在 LTcpDemo 的 test2() 中我不给 buffersize 设置任何值(假设我是用户, 我忘了), 看程序是否正常运行

```
lzx0626@DavidingPlus:~/DavidingPlus/Lark5/larksdk/build$ snippet/LTCPServerDemo/LTCPServerDemo
25
您好，我是客户端。
lzx0626@DavidingPlus:~/DavidingPlus/Lark5/larksdk/build$

lzx0626@DavidingPlus:~/DavidingPlus/Lark5/larksdk/build$ snippet/LTCPCClientDemo/LTCPCClientDemo
server_ip: 127.0.0.1
server_port: 9999
您好，我是服务端。
lzx0626@DavidingPlus:~/DavidingPlus/Lark5/larksdk/build$ |
```

## 5.18832: LAbstractSocket的peerAddress在整个通信过程中获取对方地址时 返回错误

- 原因: 学长的代码里面用了太多的指针, 我不知道为什么在调用函数的过程中指针指向的值丢失了, 也就是 peer->add 的部分丢失了, 导致后续获取不到对方的地址
- 解决: 因此我把整个项目用到指针的地方全部换掉了, 并且对真的带有指针的类做了深拷贝的处理, 防止了 double free ( LUDPdemo 里面之前的报错, 现在已经处理了)
- 测试: LTcpDemo 的 test3() 中我分别在相对 sends() 和 receives() 的各个位置都获取了 peerAddress() 的信息

```
lzx0626@DavidingPlus:~/DavidingPlus/Lark5/larksdk/build$ snippet/LTCPServerDemo/LTCPServerDemo
recv: hello, i am client.
server_ip: 127.0.0.1
server_port: 9999
client_ip: 127.0.0.1
client_port: 38868
sends: hello, i am server.
lzx0626@DavidingPlus:~/DavidingPlus/Lark5/larksdk/build$
```

```
lzx0626@DavidingPlus:~/DavidingPlus/Lark5/larksdk/build$ snippet/LTCPClientDemo/LTCPClientDemo
server_ip: 127.0.0.1
server_port: 9999
sends: hello,i am client.
server_ip: 127.0.0.1
server_port: 9999
recv: hello, i am server.
server_ip: 127.0.0.1
server_port: 9999
lzx0626@DavidingPlus:~/DavidingPlus/Lark5/larksdk/build$
```

## 6.18923: LHostAddress的getAddress 建议调整其内部处理

- 解决: 已做内部处理, 地址类型为 `Unknown` 的时候返回了空指针

```

3
4  /**
5   * @brief 获取用字符串表示的地址（网络字节序）
6   */
7  char* LHostAddress::getAddress() {
8      // 由于初始化的时候所有的构造函数都会现委托默认构造函数生
9      // if (nullptr == pAddr)
10         // throw LException("getAddress error");
11
12         if (Unknown == pAddr.type)
13             return nullptr;
14     }

```

## 7.18862: LHttpRequest和LHttpRequest的url()返回的数据不完整

- 原因: 在返回url的时候未考虑协议和端口号的显示
- 解决: 在 `LHttpRequest` 和 `LHttpRequest` 类的 `url()` 接口中需要补充对协议和端口

```

LString LHttpRequest::url() const {
    if (!m_pData->m_host.isEmpty()) {
        // url需要加上协议类型, 目前只支持http类型
        LString url("http://");
        url.append(m_pData->m_host);

        // 不是默认端口80需要指明端口
        if (80 != m_pData->m_port)
            url.append(":" + std::to_string(m_pData->m_port));

        url.append(m_pData->m_path);
        return url;
    } else {
        std::cout << "请检查是否设置host" << std::endl;
        return "";
    }
}

```

```

LString LHttpReply::url() const {
    // url需要加上协议类型，目前只支持http类型
    LString url("http://");
    url.append(m_pData→m_host);

    // 不是默认端口需要加上端口
    if (80 ≠ m_pData→m_port)
        url.append(":" + std::to_string(m_pData→m_port));

    url.append(m_pData→m_path);

    return url;
}

```

当然 LHttpReply 当中虽然有 m\_pData，但是完全没有管端口，我增添了一个接口 setPort()，和 setUrl() 对应

```

/**
 * @brief 设置URL。
 * @param reply 响应对象
 * @param request 请求对象
 */
void setUrl(LHttpReply *reply, LHttpRequest request);

/**
 * @brief 设置端口。
 * @param reply 响应对象
 * @param request 请求对象
 */
void setPort(LHttpReply *reply, LHttpRequest request);

```

在 LHttpControl 的各种类型的请求收到回复的时候都调用了 setPort() 这个方法，下面是 get 的例子

```

reply→setController(reply, this);
reply→setRequest(reply, request);
reply→setUrl(reply, request);
reply→setPort(reply, request);
reply→setOperation(1);
if (reply→readBufferSize() = 0) {
    reply→setReadBufferSize(65536);
}

```

- 测试：在 HttpRequestDemo 的 test2() 中，先用公司的，他的那个测试代码是80端口，但是他的数据有丢失，这个 bug 我还没修

```

lzx0626@DavidingPlus:~/DavidingPlus/Lark5/larksdk/build$ snippet/HttpRequestTest/HttpRequestTest
url1: http://192.168.1.211/get
url2: http://192.168.1.211/get
bufferSize: 65536
operation: GET

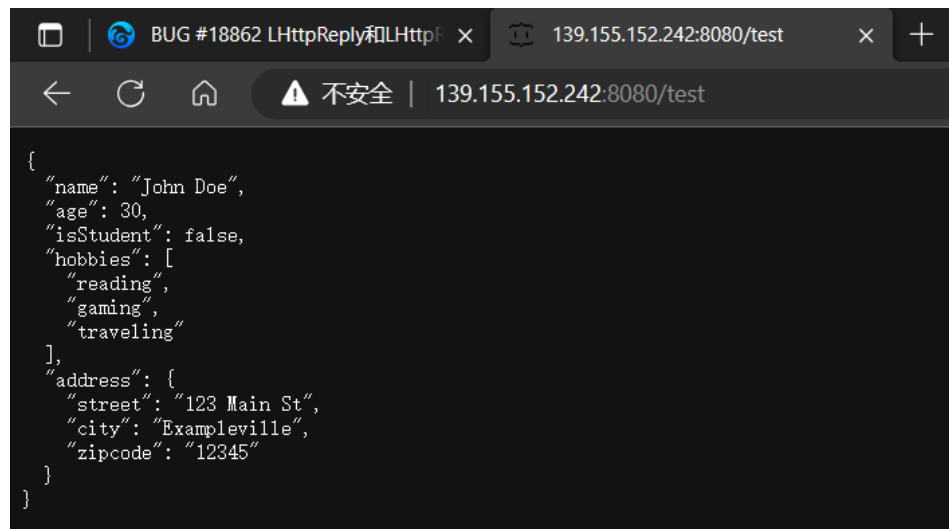
{
  "args": {},
  "headers": {
    "Connection": "keep-alive",
    "Host": "192.168.1.211"
  },
  "origin": "192.168.7.64",
  "url": "http://192.168.1.211/get"
}

lzx0626@DavidingPlus:~/DavidingPlus/Lark5/larksdk/build$ |

```

为了测试端口不为80，我下面用了我自己的云服务器测试

url 为: <http://139.155.152.242:8080/test>



```

{
  "name": "John Doe",
  "age": 30,
  "isStudent": false,
  "hobbies": [
    "reading",
    "gaming",
    "traveling"
  ],
  "address": {
    "street": "123 Main St",
    "city": "Exampleville",
    "zipcode": "12345"
  }
}

```

请求结果如下：

协议和端口都显示出来了，这个的数据又全部请求到了，有点奇怪...

```

lzx0626@DavidingPlus:~/DavidingPlus/Lark5/larksdk/build$ snippet/HttpRequestTest/HttpRequestTest
url1: http://139.155.152.242:8080/test
url2: http://139.155.152.242:8080/test
bufferSize: 65536
operation: GET

{
  "name": "John Doe",
  "age": 30,
  "isStudent": false,
  "hobbies": [
    "reading",
    "gaming",
    "traveling"
  ],
  "address": {
    "street": "123 Main St",
    "city": "Exampleville",
    "zipcode": "12345"
  }
}

lzx0626@DavidingPlus:~/DavidingPlus/Lark5/larksdk/build$

```

## 8.18738: LAbstractSocket的Lsocket函数名有误

- 解决：将函数名更改为 `createSocket()`

```

/**
 * @brief 创建socket套接字，将套接字的文件描述符存储下来，并判断是否创建成功
 * @return true 套接字创建成功
 * @return false 套接字创建失败
 */
bool createSocket();

```

## 9.18811: LUdpSocket的sends(unsigned char\* data,int length) 数据长度设置为0 在接收时程序崩溃

- 疑惑：在我这边的测试程序中跑出来正常，这条 bug 可能有问题
- 测试：在 LUdpDemo 的 test2() 中，按照 bug 的指示进行了重现

```

lzx0626@DavidingPlus:~/DavidingPlus/Lark5/larksdk/build$ snippet/LUDPServerDemo/LUDPServerDemo
receives:
lzx0626@DavidingPlus:~/DavidingPlus/Lark5/larksdk/build$ |

lzx0626@DavidingPlus:~/DavidingPlus/Lark5/larksdk/build$ snippet/LUDPCClientDemo/LUDPCClientDemo
sends: 您好，这里是客户端！
lzx0626@DavidingPlus:~/DavidingPlus/Lark5/larksdk/build$

```

## 10.18808: LUdpSocket的sends(unsigned char\* data,int length) 发送前未设置对方的地址 程序崩溃

- 疑惑：和第3条一样，我认为抛出异常程序就应该结束，我们自身作捕获异常的操作就可以了
- 测试：在 LUdpDemo 的 test1() 中，设置对方的IP和端口的代码就在下面

```

43
44 // bug1: 我设置不给定对方的地址或者端口
45 client.setPeerAddress(&ser_add);
46 client.setPeerPort(ser_port);
47

```