

实验五：网络应用程序设计实验

一、实验目的

- 1.学习基于mbed操作系统的网络应用程序的设计方法；
- 2.掌握TCP、UDP协议的通讯技术；
- 3.掌握调试程序的方法

二、实验仪器与设备

Mbed 开发板、连接线、计算机。

三、预备知识

（1）mbed 计算机网络应用初步

Mbed 提供了 EthernetInterface 类来完成和网络层及网络接口层相关的服务，包括物理接口的初始化、IP 地址及子网掩码的设置、网关的设置及 DNS 服务器的设置，另外 EthernetInterface 类中还包含了 Socket 类及其继承类 UDPSocket、TCPSocketServer、TCPSocketConnecion 类提供传输层服务，以及 Endpoint 辅助类用于设置传输层地址，其是 mbed 网络库实现的层次结构图：



使用 EthernetInterface 类提供的主要方法列表如下：

类名	方法	用途
EthernetInterface	static int init();	静态方法使用 dhcp 方式设置 mbed 节点 IP 地址信息返回 0 表示成功，返回负值表示失败
	static int init(const char* ip, const char* mask, const char* gateway);	静态方法，使用手动方式设置 mbed 节点 IP 地址信息，传入参数依次是 IP 地址、子网掩码、网关，返回 0 表示成功，返回负值表示失败

	static int connect(unsigned int timeout_ms=15000);	静态方法,启动网络物理接口,返回 0 表示成功, 返回负值表示失败
	static int disconnect();	静态方法,关闭网络物理接口,返回 0 表示成功, 返回负值表示失败
	static char* getMACAddress();	静态方法, 获取节点 MAC 地址字符串
	static char* getIPAddress();	静态方法, 获取节点 IP 地址字符串
	static char* getGateway();	静态方法, 获取节点网关字符串
	static char* getNetworkMask();	静态方法, 获取节点子网掩码字符串

(2) mbed TCP 应用程序基础

mbed使用TCPSocketServer实现TCP服务端功能使用 TCPSocketConnection实现 TCP 客户端功能,同时还提供了一个 EndPoint 辅助类,用户保存传输层地址信息,它们提供的主要方法如下:

类名	方法	用途
TCPSocketServer	TCPSocketServer();	构造函数实例化 TCPSocketServer 对象
	int bind(int port);	把 TCP 服务器绑定到 port 端口, port 的范围是 0-65535,返回 0 表示成功, -1 表示失败
	int listen(int backlog=1);	开始监听先前设定的网络端口, backlog 表示同时支持的连接数返回 0 表示成功, -1 表示失败

	int accept(TCPSocketConnection& connection);	接受一个TCP 客户端连接connection 表示用于处理该连接的 TCPSocketConnection 实例,返回 0 表示成功, -1 表示失败
	int close(bool shutdown=true);	关闭 TCP 连接, shutdown 表示是否
TCPSocketConnection	TCPSocketConnection();	构造函数, 实例化 TCPSocketConnection 对象
	int connect(const char* host, const int port);	连接到 TCP 服务器,传入的参数分别表示主机地址和端口号,返回 0 表示成功, -1 表示失败
	bool is_connected(void);	判断是否连接成功, true 表示成功, false 表示失败
	int send (char* data, int length);	发送数据到远程主机, data 为要发送的数据缓冲区, length 为要发送的数据字节数, 返回值为成功写入的字节数, -1 表示失败

	int send_all(char* data, int length);	发送所有数据到远程主机, data 为要发送的数据缓冲区, length 为要发送的数据字节数, 返回值为成功写入的字节数, -1 表示失败
	int receive(char* data, int length);	从远程主机接收数据, data 表示接收数据的存储缓冲区, length 表示缓冲区的大小, 返回值为实际接收的字节数, -1 表示失败
	int receive_all(char* data, int length);	从远程主机接收所有数据, data 表示接收数据的存储缓冲区, length 表示缓冲区的大小, 返回值为实际接收的字节数, -1 表示失败
	void set_blocking(bool blocking, unsigned int timeout=1500);	设置当前连接的工作方式, 可以是阻塞模式, 也可以是超时模式
	int close(bool shutdown=true);	关闭 TCP 连接, shutdown 表示是否
Endpoint	Endpoint(void);	构造函数, 实例化 Endpoint 类
	void reset_address(void);	重置并清空地址信息
	int set_address(const char* host, const int port);	设置地址信息, 传输参数为地址及端口号
	char* get_address(void);	返回地址字符串
	int get_port(void);	返回端口数值

有了以上各个类的帮助, 就可以比较方便地开发 TCP 应用程序

(3) mbed UDP 程序设计基础

mbed 提供了 UDPSocket 类用于 UDP 数据传输, 它提供的主要方法有:

类名	方法	用途
UDPSocket	UDPSocket();	构造函数, 实例化 UDPSocket 对象
	int init(void);	启动 UDP 客户端, 它和具体的端口无关, 返回 0 表示成功, -1 表示失败
	int bind(int port);	启动 UDP 服务端, 并绑定到 port 端口, 返回 0 表示成功, -1 表示失败
	int join_multicast_group(const char* address);	加入特定地址的组播分组, 返回 0 表示成功, -1 表示失败
	int set_broadcasting(bool broadcast=true);	让 UDP 工作在广播模式
	int sendTo(Endpoint &remote, char *packet, int length);	发送数据给特定的地址, remote 为一个具体的传输层地址, packet 为要发送的数据缓冲区, length 为要发送的数据大小, 返回 -1 表示失败, 其它则表示成功

	int receiveFrom(Endpoint &remote, char *buffer, int length);	从特定的地址接收数据，remote 为一个具体的传输层地址,buffer 为数据接收缓冲区,length 为数据接收缓冲区的大小，返回-1 表示失败，其
		它值表示接收到数据的大小
	void set_blocking(bool blocking, unsigned int	设置当前连接的工作方式，可以是阻塞模式，也可以是超时模式
	int close(bool shutdown=true);	关闭当前连接

四、实验内容

1、mbed TCP 客户端程序设计

TCP 客户端程序的特点是随时可以根据应用的需要发起或结束连接，服务端实现无需知道客户端的存在，而一旦与服务端建立连接后，双方可以任意收发数据，所以非常适合那些需要远程监控的场合，下面是一个简单的应用示例，它将自动上报用户的按键时间：

```
#include "EthernetInterface.h"
#include "rtos.h"
#include "OneWire.h" DigitalOut led1(LED1); DigitalOut led2(LED2);
OneWire ds2411(PI_29); InterruptIn btn(P2_8); EthernetInterface eth;
TCPSocketConnection socket;
const char* REMOTEIP = "192.168.1.11";
const int REMOTEPORT = 8088;
uint32_t pressedindex=0;
Queue<uint32_t, 255> queue;
void mbed_mac_address(char *mac)
{
    char romcode[8];
    ds2411.busInit();
    ds2411.getRomCode(romcode);
    memcpy(mac,romcode,6);
}
void net_task(void const *)
{
    //eth.init(); Use DHCP
    eth.init("192.168.1.100", "255.255.255.0", "192.168.1.1");
    eth.connect();
    printf("IP Address is %s\n", eth.getIPAddress());
    printf("MAC Address is %s\n", eth.getMACAddress());
    printf("GateWay Address is %s\n", eth.getGateway());
    while (socket.connect(REMOTEIP, REMOTEPORT)<0)
    {
```

```

printf("Unable to connect to (%s) on port (%d)\n", REMOTEIP,
REMOTEIP);
Thread::wait(1000); } }
void send_task(void const *)
{
while (true)
{
    osEvent evt = queue.get();
    if (evt.status == osEventMessage) {
        char hello[] = "Button was pressed\n";
        led2=!led2;
        socket.send_all(hello, sizeof(hello) - 1);
    }
}
}
void btnhandler()
{
    pressedindex++;
    queue.put(&pressedindex);
}
int main()
{
    Thread netTask(net_task, NULL, osPriorityNormal, 1024 * 4);
    Thread sendtask(send_task, NULL, osPriorityNormal, 1024 * 4);
    btn.fall(&btnhandler);
    while(1)
    {
        led1=!led1;
        Thread::wait(1000);
    }
    socket.close();
}

```

(2) mbed UDP 程序初步设计

由于 UDP 在使用时无需连接就可以收发数据，所以在代码的编写上比较简单，但也带来了问题，由于 UDP 在数据通讯的过程中并没有一个切实存在的连接，所以让服务端返回数据给客户端就变得复杂了，我们先看下面的代码，这里用的是 init 方法：

```

#include "EthernetInterface.h"
#include
"rtos.h"
EthernetInterf
ace eth;

```

```

UDPSocket
socket;
const char* REMOTEIP = "192.168.1.11";
const int REMOTEPORT = 8088;
    Endpoint echo_server;

void net_task(void const *)
{
    eth.init("192.168.1.100", "255.255.255.0", "192.168.1.1");
    eth.connect();
    printf("IP Address is %s\n", eth.getIPAddress());
    printf("MAC Address is %s\n", eth.getMACAddress());
    printf("GateWay      Address      is      %s\n",      eth.getGateway());
    socket.set_blocking(false, 100);
    socket.init();
}

void send_task(void const *)
{
    char out_buffer[] = "Hello World\n";
    while (1)
    {
        socket.sendTo(echo_server, out_buffer, sizeof(out_buffer));
        Thread::wait(1000);
    }
}

int main()
{
    echo_server.set_address(REMOTEIP, REMOTEPORT);
    Thread netTask(net_task, NULL, osPriorityNormal, 1024 * 4);
    Thread sendTask(send_task, NULL, osPriorityNormal, 1024 * 4);
    while (1)
    {
        char in_buffer[256];
        int n = socket.receiveFrom(echo_server, in_buffer,
        sizeof(in_buffer)); if (n>0)
        {
            in_buffer[n] = '\0';
            printf("%s\n",
                in_buffer);
        }
        Thread::yield();
    }
    socket.close();
    eth.disconnect();
}

```

}

五、实验要求

- 1、认真阅读预备知识和实验内容；
- 2、编写代码，编译、下载到 mbed 开发板，验证实验效果。
- 3、撰写实验报告。