佛山大学《物联网通信技术课程》实验报告

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 实验名称 | IPV6通信 | | | 成绩 |  |
| 专业班级 | 22物联网工程2班 | | |
| 姓名 |  | 学号 |  | 学期 | 2024-2025（1） |
| 指导教师 | 陆海波 | 实验性质 | 验证性 | 实验日期 | 2024.12.07 |
| 一、实验目的 | | | | | |
| （1）学习低功耗无线传感网（IPV6）网络协议栈  （2）学习基于contiki系统Rime协议栈 | | | | | |
| 二、实验内容 | | | | | |
| 1.启动IAR开发环境并加载指定的工程文件。工程文件路径为：配套光盘下的“源代码\物联网通信技术\实验8 IPV6通信实验-V20161122\project\iar\simple\contiki\_cc2530.eww”。  2.实验包含两个独立的工作区，分别为“udp-server”和“udp-client”。首先，切换至“udp-server”工作区，在项目树中找到并打开位于“platform->cc2530dk->contiki-main.c”的源文件。在此文件中，取消#define UDP\_CLIENT宏定义的注释状态，并确保#define UDP\_SERVER宏定义处于启用状态，以便进行编译，具体操作请参考图4.1的指导。  IMG_256  图4.1 选择udp-server工作空间  3.完成项目的编译过程。若编译无误并成功执行，将在“..\contiki\_cc2530\_iar\project\iar\simple\udp-server\Exe\”路径下生成名为contiki\_cc2530\_server.hex的十六进制固件文件。   1. 进行设备编程： a. 利用CC Debugger工具将开发板连接至计算机，并启动SmartRF Flash Programmer程序。 b. 在SmartRF Flash Programmer中，选择上一步编译得到的contiki\_cc2530\_server.hex固件文件，并将该固件烧录至编号为7的节点上的CC2530微控制器。     图4.2 烧录服务器程序   1. **2）**在固件烧录完成后，通过点击SmartRF Flash Programmer软件界面上的“Read IEEE”按钮来获取服务器的IEEE地址。这一步是必要的，以确保我们能够识别和记录服务器的网络标识。例如，本次实验中，服务器的IEEE地址被成功读取，显示为00 12 4B 00 05 43 42 B2（十六进制格式），具体详情可参见图4.3。：     图4.3 读服务器的IEEE地址  **5、在“Workspace”工作区中，选择“udp-client”项目。在项目结构视图中，找到并双击platform->cc2530dk->contiki-main.c文件。在此文件中，将#define UDP\_SERVER 1的宏定义注释掉，同时确保#define UDP\_CLIENT 1的宏定义保持启用状态，以便为客户端配置正确的编译环境，具体操作可参考图4.4的指示。**    图4.4 选择udp-client工作空间  **6、**编译程序，选择生成的contiki\_cc2530\_client.hex文件，并下载到8号节点的CC2530单片机上。    图4.6 烧录服务器程序 | | | | | |
| 三、实验结果 | | | | | |
| 问题分析：  IPV6，作为互联网协议的最新版本，其设计初衷是为了应对IPv4地址资源耗尽的问题。在本次实验中，我们的目标是利用IPV6技术来构建一个服务器与客户端之间的通信桥梁，实现它们之间的网络连接。  源代码：  **server.c**  /\*-----------------------------------------------------------------\*/  *PROCESS(udp\_server\_process, "UDP server process"); //创建服务器进程*  *AUTOSTART\_PROCESSES(&udp\_server\_process);*  */\*-------------------------------------------------------------\*/*  *static void*  *tcpip\_handler(void)*  *{*  *uint8\_t n,Crc\_buf[2];*  *memset(buf, 0, MAX\_PAYLOAD\_LEN);*  *if(uip\_newdata()) { //无线接收到数据*  *leds\_on(LEDS\_RED); //点亮LED*  *len = uip\_datalen(); //获取接收的数据长度*  *puthexstr(uip\_appdata,len);//通过串口输出*  *#if SERVER\_REPLY*  *//获取地址和端口号*  *uip\_ipaddr\_copy(&server\_conn->ripaddr, &UIP\_IP\_BUF->srcipaddr);*  *server\_conn->rport = UIP\_UDP\_BUF->srcport;*  *#endif*  *}*  *leds\_off(LEDS\_RED);//熄灭LED*  *return;*  *}*  */\*----------------------------------------------------------------\*/*  *PROCESS\_THREAD(udp\_server\_process, ev, data)*  *{*  *static struct etimer et;*  *PROCESS\_BEGIN();*  *putstring("Starting UDP server\r\n");*  *#if BUTTON\_SENSOR\_ON*  *putstring("Button 1: Print RIME stats\r\n");*  *#endif*  *#if SERVER\_RPL\_ROOT*  *create\_dag();*  *#endif*  *server\_conn = udp\_new(NULL, UIP\_HTONS(0), NULL);//创建一个网络*  *udp\_bind(server\_conn, UIP\_HTONS(3000)); //设置端口号*  *PRINTF("Listen port: 3000, TTL=%u\n", server\_conn->ttl);*  *while(1) {*  *PROCESS\_YIELD();*  *if(ev == tcpip\_event) { //监听端口3000*  *tcpip\_handler();*  *#if (BUTTON\_SENSOR\_ON && (DEBUG==DEBUG\_PRINT))*  *} else if(ev == sensors\_event && data == &button\_sensor) {*  *print\_stats();*  *#endif /\* BUTTON\_SENSOR\_ON \*/*  *}*  *}*  *PROCESS\_END();*  *}*  注意：将以下红色部分代码的地址改为服务器的IPv6地址，需要将上步获取服务器的IEEE地址转换成IPv6地址。IEEE 802.15.4地址转换为IPv6地址的方法如下:  若IEEE地址为：00-12-4B-00-05-43-42-B2；  那么它的链路本地地址为：FE-80-00-00-00-00-00-00-**02-12-4B-00-05-43-42-B2**，请注意IEEE地址的首字节00被翻转为02，可简写为FE80::212:4B00:543:42B2。  **Client.c**  /\*---------------------------------------------------------------------------\*/  PROCESS(udp\_client\_process, "UDP client process"); //UDP客户端进程  #if BUTTON\_SENSOR\_ON  PROCESS\_NAME(ping6\_process);  AUTOSTART\_PROCESSES(&udp\_client\_process, &ping6\_process);  #else  AUTOSTART\_PROCESSES(&udp\_client\_process);  #endif  /\*---------------------------------------------------------------------------\*/  static void  tcpip\_handler(void)  { //监听端口信息  leds\_on(LEDS\_GREEN); //点亮LED  if(uip\_newdata()) { //若有信息  puthexstr(uip\_appdata , uip\_datalen()); //将信息通过串口输出  }  leds\_off(LEDS\_GREEN); //熄灭LED  return;  }  /\*------udp\_client\_process start------------------------------\*/  PROCESS\_THREAD(udp\_client\_process, ev, data)  {  static struct etimer et;  uip\_ipaddr\_t ipaddr;  char ipaddrstr[32];  PROCESS\_BEGIN();  PRINTF("UDP client process started\r\n");  //设置链路单播地址 即服务器的地址  //刚才的地址为：FE-80-00-00-00-00-00-00-**02-12-4B-00-05-43-42-B2**  uip\_ip6addr(&ipaddr, 0xfe80,0,0,0,0x0212,0x4b00,0x0543,0x42B2);  //这里输入刚才得到的链路本地地址  /\* new connection with remote host \*/  l\_conn = udp\_new(&ipaddr, UIP\_HTONS(3000), NULL); //创建连接  if(!l\_conn) {  PRINTF("udp\_new l\_conn error.\r\n");  }  udp\_bind(l\_conn, UIP\_HTONS(LOCAL\_CONN\_PORT)); //设置端口号  PRINTF("Link-Local connection with ");  PRINT6ADDR(&l\_conn->ripaddr);  PRINTF(" local/remote port %u/%u\r\n",  UIP\_HTONS(l\_conn->lport), UIP\_HTONS(l\_conn->rport));    //设置全局单播地址  /\* uip\_ip6addr(&ipaddr, 0xaaaa,0,0,0,0x0212,0x4b00,0x0543,0x42B2);  g\_conn = udp\_new(&ipaddr, UIP\_HTONS(3000), NULL);  if(!g\_conn) {  PRINTF("udp\_new g\_conn error.\r\n");  }  udp\_bind(g\_conn, UIP\_HTONS(GLOBAL\_CONN\_PORT));  PRINTF("Global connection with ");  PRINT6ADDR(&g\_conn->ripaddr);  PRINTF(" local/remote port %u/%u\r\n",  UIP\_HTONS(g\_conn->lport), UIP\_HTONS(g\_conn->rport));  \*/  etimer\_set(&et, SEND\_INTERVAL); //设置时间  while(1) {  PROCESS\_YIELD();  if(etimer\_expired(&et)) {  timeout\_handler();  etimer\_restart(&et);  } else if(ev == tcpip\_event) { //监听端口  tcpip\_handler();  }  }  PROCESS\_END();  }  C**ontiki-main.c**  由于server和client两个项目都是共用一个main.c函数，所以要以宏UDP\_SERVER和UDP\_CLIENT 分别对串口接收中断进行相应的处理。  while(1) {  #if UDP\_SERVER //选择服务器时  if(U0\_Rflag==1)  { //发送给客户端  uip\_udp\_packet\_send(server\_conn,U0\_Rx\_buf , U0\_Rx\_count);  memset(U0\_Rx\_buf, 0, 60); //缓存区清零  // puthexstr(U0\_Rx\_buf,U0\_Rx\_count);  U0\_Rflag=0;  U0\_Rx\_count=0; //接收长度清零  }  #endif    #if UDP\_CLIENT //选择客户端时  if(U0\_Rflag==1)  {  // puthexstr(U0\_Rx\_buf,U0\_Rx\_count);  //发送给服务器  uip\_udp\_packet\_send(this\_conn, U0\_Rx\_buf, U0\_Rx\_count);  memset(U0\_Rx\_buf, 0, 60);  U0\_Rx\_count=0; //接收长度清零  U0\_Rflag=0; //接收标志清零  }  #endif  运行结果：  重新启动 IPv6 模块时，先启动 IPv6 服务器创建网络，过 2s左右再启动 IPv6 客户端。  1. 首先，使用串口线将IPv6服务器（标识为7号节点）与计算机相连。确保将白色开关拨至最右侧位置，以便DB9接口与CC2530单片机成功对接。接着，启动串口调试助手软件，并根据下图所示的配置进行设置，完成设置后激活串口通信。  2、用串口线连接IPv6客户端(8号节点)，白色开关拨打最右边(DB9与CC2530  连接);打开串口调试助手软件，按下图设置完成后打开串口。  口。     1. 实验结果及分析   从运行结果来看，成功将IPV6服务端和IPV6客户端接通。 | | | | | |
| 四、实验总结 | | | | | |
| 在本次关于IPV6通信的实验中，我获得了对物联网通信技术的深刻洞察。通过亲手搭建服务器与客户端之间的通信链路，我对这一领域的知识有了更加透彻的掌握。目睹服务器与客户端顺利连接，并在串口调试工具上接收到成功提示的那一刻，我感到无比的兴奋和满足。此次实验不仅让我对IPV6通信的基础理论有了初步的了解，还让我掌握了实际操作的相关技巧。通信的每一个环节都是紧密相连的，需要我细心操作，以确保通信的成功实施。作为一项前沿的通信技术，IPV6的学习让我站在了通信技术的最前沿，这次实践经历无疑是一次宝贵的技术探索之旅。 | | | | | |