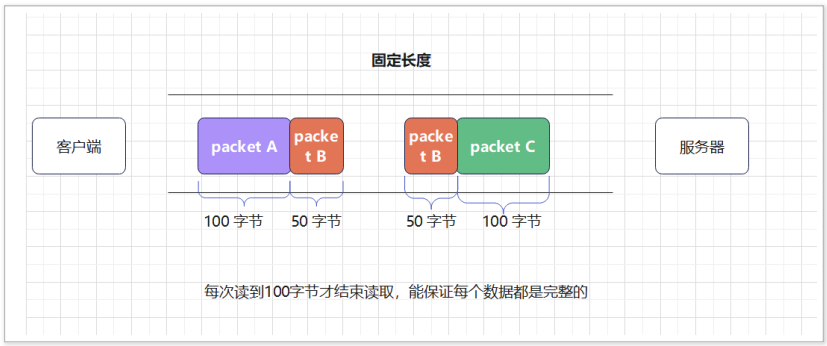
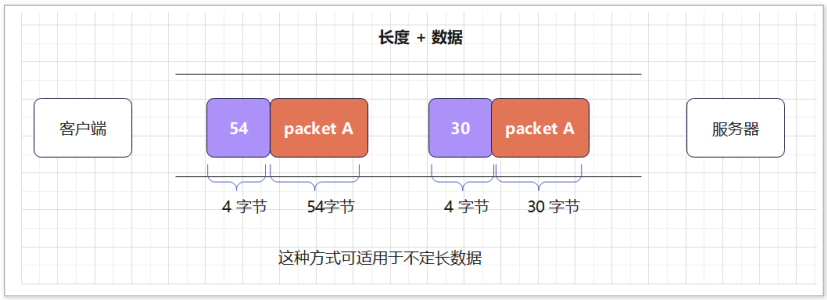
3.2 tcp 粘包解决方案_物联网 / 嵌入式工程师 - 慕课网

> 慕课网慕课教程 3.2 tcp 粘包解决方案涵盖海量编程基础技术教程，以图文图表的形式，把晦涩难懂的编程专业用语，以通俗易懂的方式呈现给用户。

- tcp 粘包的解决方案如下:
  - 方式一：使用定长数据包, 每次必须要读取固定长度的数据, 适用于数据长度是固定的场景
  - 

  - 方式二：使用数据长度 + 数据的方式，先接收数据长度，再根据长度接收数据, 这里就结合第一种方式，进行固定长度接收, 这种方式适用于不定长数据场景
  - 

  -
  - 方式三：使用特殊间隔符，如换行等来区分数据包的边界, 使用较少
-

这里主要实现第 2 种方式，适用于不定长数据

- step 1：发送时，分两次发送，第一次发送数据长度，第二次发送数据
-

```
for(;;){

        length = strlen(buffer);

        pbuffer = (char *)malloc(length + 4);
        memcpy(pbuffer,&length,4);
        memcpy(pbuffer + 4,buffer,length);

        sbytes = send(sfd,pbuffer,length + 4,0);
        if (ret == -1){
            perror("[ERROR] Failed to send.");
            exit(EXIT_FAILURE);
        }
        usleep(100);

    }
```

- step 2：接收时，分两次接收，第一次接收数据长度，第二次接收数据
-

```
int count  = 0;

    for(;;){
        length = 0;
```

```
    total_received = 0;

    rbytes = recv(cfd,&length,4,0);
    if (rbytes == -1){
        perror("[ERROR] Failed to recv.");
        exit(EXIT_FAILURE);
    }
    for(;;){

        rbytes = recv(cfd,buffer + total_received,length - total_received,0);
        if (rbytes == -1){
            perror("[ERROR] Failed to recv.");
            exit(EXIT_FAILURE);
        }else if (rbytes == 0){
            printf("The client has been shutdown.\n");
        }else if (rbytes > 0){
            total_received += rbytes;
            if (total_received == length)
                break;
        }
    }
        printf("buffer : %s\n",buffer);
    sleep(1);
}
```

- 完整代码
  - client.c
  - ```
    #include <stdio.h>
    #include <stdlib.h>
    #include <string.h>
    #include <strings.h>
    #include <unistd.h>
    #include <netinet/in.h>
    #include <arpa/inet.h>
    #include <sys/types.h>
    #include <sys/socket.h>


    int main(int argc,char *argv[])
    {
        int sfd,ret;
        ssize_t sbytes = 0,rbytes = 0;
        char buffer[] = "Hello,server";
        int length;
        char *pbuffer = NULL;
        struct sockaddr_in svr_addr;
        if (argc != 3){
            fprintf(stderr,"Usage : %s < ip > < port >.\n",argv[0]) ;
            exit(EXIT_FAILURE);
        }


        sfd = socket(AF_INET,SOCK_STREAM,0);
        if (sfd == -1){
            perror("[ERROR] Failed to socket.");
            exit(EXIT_FAILURE);
        }

        bzero(&svr_addr,sizeof(struct sockaddr_in));
        svr_addr.sin_family = AF_INET;
        svr_addr.sin_port = htons(atoi(argv[2]));
        svr_addr.sin_addr.s_addr = inet_addr(argv[1]);

        ret = connect(sfd,(const struct sockaddr *)&svr_addr,sizeof(struct sockaddr_ir
        if (ret == -1){
            perror("[ERROR] Failed to connect.");
            exit(EXIT_FAILURE);
        }
        for(;;){

            length = strlen(buffer);

            pbuffer = (char *)malloc(length + 4);
            memcpy(pbuffer,&length,4);
            memcpy(pbuffer + 4,buffer,length);

            sbytes = send(sfd,pbuffer,length + 4,0);
            if (ret == -1){
                perror("[ERROR] Failed to send.");
                exit(EXIT_FAILURE);
            }
            usleep(100);
    ```
    ```

```
    }


    close(sfd);

    return 0;
}
```

- server.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <strings.h>
#include <unistd.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <sys/types.h>
#include <sys/socket.h>

#define BACKLOG 10


int main(int argc,char *argv[])
{
    int sfd,cfd,ret;
    struct sockaddr_in svr_addr,cli_addr;
    char buffer[1024] = {0};
    ssize_t sbytes = 0,rbytes = 0;
    int length;
    int total_received;
    socklen_t len = sizeof(struct sockaddr_in);
    if (argc != 3){
        fprintf(stderr,"Usage : %s < ip > < port >.\n",argv[0]) ;
        exit(EXIT_FAILURE);
    }


    sfd = socket(AF_INET,SOCK_STREAM,0);
    if (sfd == -1){
        perror("[ERROR] Failed to socket.");
        exit(EXIT_FAILURE);
    }

    bzero(&svr_addr,sizeof(struct sockaddr_in));
    svr_addr.sin_family = AF_INET;
    svr_addr.sin_port = htons(atoi(argv[2]));
    svr_addr.sin_addr.s_addr = inet_addr(argv[1]);


    ret = bind(sfd,(const struct sockaddr *)&svr_addr,sizeof(struct sockaddr));
    if (ret == -1){
        perror("[ERROR] Failed to bind.");
        exit(EXIT_FAILURE);
    }

    ret = listen(sfd,BACKLOG);
    if (ret == -1){
        perror("[ERROR] Failed to listen.");
        exit(EXIT_FAILURE);
    }


    cfd = accept(sfd,(struct sockaddr *)&cli_addr,&len);
    if (ret == -1){
        perror("[ERROR] Failed to accpet.");
        exit(EXIT_FAILURE);
    }

    printf("ip : %s port :%d\n",inet_ntoa(cli_addr.sin_addr),ntohs(cli_addr.sin_port));
    for(;;){
        length = 0;
        total_received = 0;

        rbytes = recv(cfd,&length,4,0);
        if (rbytes == -1){
            perror("[ERROR] Failed to recv.");
            exit(EXIT_FAILURE);
        }
        for(;;){

            rbytes = recv(cfd,buffer + total_received,length - total_received,0);
            if (rbytes == -1){
                perror("[ERROR] Failed to recv.");
```

```
            exit(EXIT_FAILURE);
        }else if (rbytes == 0){
            printf("The client has been shutdown.\n");
        }else if (rbytes > 0){
            total_received += rbytes;
            if (total_received == length)
                break;
        }
    }
        printf("buffer : %s\n",buffer);
    sleep(1);
}

close(sfd);

return 0;
}
```

```
ben@ubuntu:~/class/week15/codes/part3/A02server$ ./server 10.226.42.58 8888
ip : 10.226.42.58,port : 55122
 hello,abcde
 hello,abcde
 hello,abcde
 hello,abcde
 hello,abcde
 hello,abcde
```

- 从上面结果可以分析, 数据包没有出现粘包或者拆包的情况
-
-
    - 编写客户端与服务器的代码解决 tcp 粘包问题

全文完