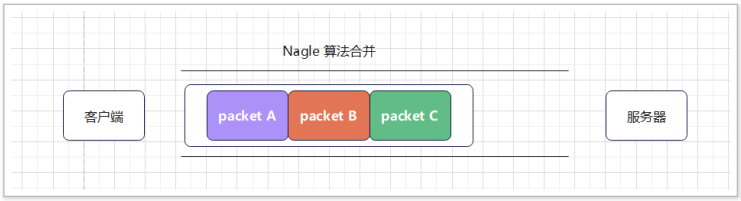
3.1 tcp 粘包原因分析_物联网 / 嵌入式工程师 - 慕课网

> 慕课网慕课教程 3.1 tcp 粘包原因分析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。

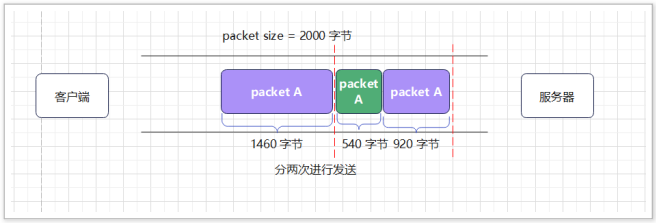
- **tcp 粘包：使用 tcp 协议进行数据传输时,** 发送方发送的若干个数据包到接收方接收时粘成一包，从接收缓冲区看，后一包数据的头紧接着前一包数据的尾。

- 从上面可以看出粘包主要分为两种情况:
- 多个完整的数据包粘在一起
- 一个数据包中包含另一个数据包的一部分
- tcp 粘包问题会影响到有数据结构的数据包，会导致数据包解析出现问题
    - 在前面的发布与订阅框架中，就使用自定义结构的数据包
    - ```
      #define TOPIC_SZ 64
      #define CONTENT_SZ 64

      typedef struct packet{
          char topic[TOPIC_SZ];
          pid_t pid;
          enum work_mode mode;
          char content[CONTENT_SZ];
      }packet_t;
      ```
- TCP 协议是面向连接的、可靠的、基于字节流的传输层 通信协议
- 产生 tcp 粘包原因并非 tcp 协议本身引起的, 主要原因是 TCP 协议在底层是字节流，并不关注, 应用层的消息边界, 可以是由发送方引起，也可以由接收方引起
    - 发送方:
        - 发送了多个比较小的数据包, 一般是小于 tcp 内核缓冲区, 默认情况下,tcp 采用了 Nagle 算法, 会合并连续的小的数据包一次性发送
      ```



- 发送方一次性发送的数据大于 MTU，则会发生拆包，将字节流进行切片分成多个包进行发送
  - MTU 表示最大传输单元，默认设置的大小为 1500 字节，在去掉 tcp header(tcp 协议头) 与 ip header(ip 协议头) 后为 1460 字节



- 接收方:
  - 当发送方的速度大于接收方的速度时，在缓冲区中缓存了多个数据包，一次性读取，则会发生读取的多个数据包

- 在 tcp 粘包测试时，主要测试当发送方的速度大于接收方速度的时候

- 客户端代码如下:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <unistd.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 sbuffer[1024] = {0};
    char rbuffer[1024] = {0};

    struct sockaddr_in svr_addr;
    if (argc != 3){
        fprintf(stderr,"Usage : %s < ip > < port >.\n",argv[0]);
        return -1;
    }

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

    printf("sfd = %d\n",sfd);
    bzero(&svr_addr,sizeof(svr_addr));
    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));
    if (ret == -1){
        perror("[ERROR] connect():");
        exit(EXIT_FAILURE);
    }

    for(;;){

        strcpy(sbuffer,"hello,abcde");

        sbytes = send (sfd,sbuffer,strlen(sbuffer),0);

        if (sbytes == -1){
            perror("[ERROR] send(): ");
```

```
            exit(EXIT_FAILURE);
        }

        usleep(100);
    }



    close(sfd);
    return 0;
}



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

#define LISTEN_SZ 10

int main(int argc,char *argv[])
{
    if (argc != 3){
        fprintf(stderr,"usage : %s < ip > < port >.\n",argv[0]);
        exit(EXIT_FAILURE);
    }

    int sfd,ret,cfd;
    struct sockaddr_in  svr_addr,cli_addr;

    ssize_t sbytes,rbytes;
    char buffer[1024] = {0};

    sfd = socket(AF_INET,SOCK_STREAM,0);
    if (sfd == -1){
        perror("[ERROR] 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_in));
    if (ret == -1){
        perror("[ERROR] bind(): ");
        close(sfd);
        exit(EXIT_FAILURE);
    }


    ret = listen(sfd,LISTEN_SZ);
    if (ret == -1){
        perror("[ERROR] listen(): ");
        close(sfd);
        exit(EXIT_FAILURE);
    }

    socklen_t len = sizeof(struct sockaddr_in);
    bzero(&cli_addr,sizeof(struct sockaddr));
    cfd = accept(sfd,(struct sockaddr *)&cli_addr,&len);
    if (cfd == -1){
        perror("[ERROR] accept(): ");
        exit(EXIT_FAILURE);
    }

    printf("ip : %s,port : %d\n",inet_ntoa(cli_addr.sin_addr),ntohs(cli_addr.sin_port));

    for(;;) {

        memset(buffer,sizeof(buffer),0);
        rbytes = recv(cfd,buffer,sizeof(buffer),0);
        if (rbytes == -1){
            perror("recv(): ");

            exit(EXIT_FAILURE);
        }else if (rbytes == 0){
            printf("The client is offline.\n");

            exit(EXIT_FAILURE);
```

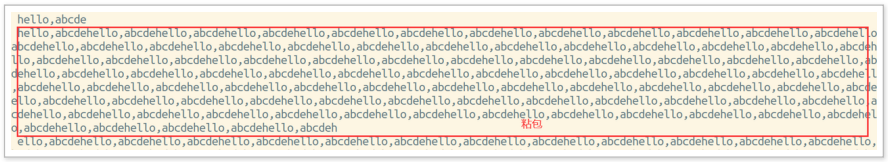
```
        }else if (rbytes > 0){

            printf(" %s\n",buffer);

        }

        sleep(1);
    }

    close(cfd);
    close(sfd);
    return 0;
}
```



1. 理解 tcp 粘包的原因

2. 编写代码验证 tcp 粘包

全文完