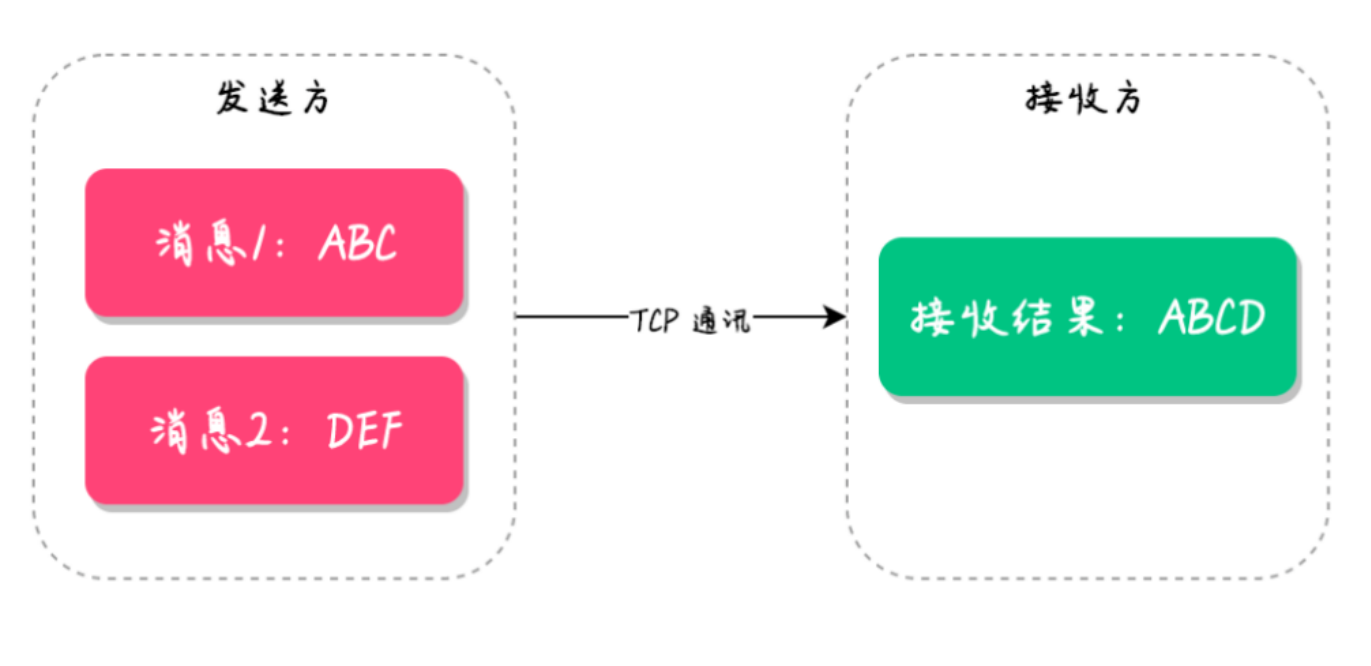
解决粘包和半包问题

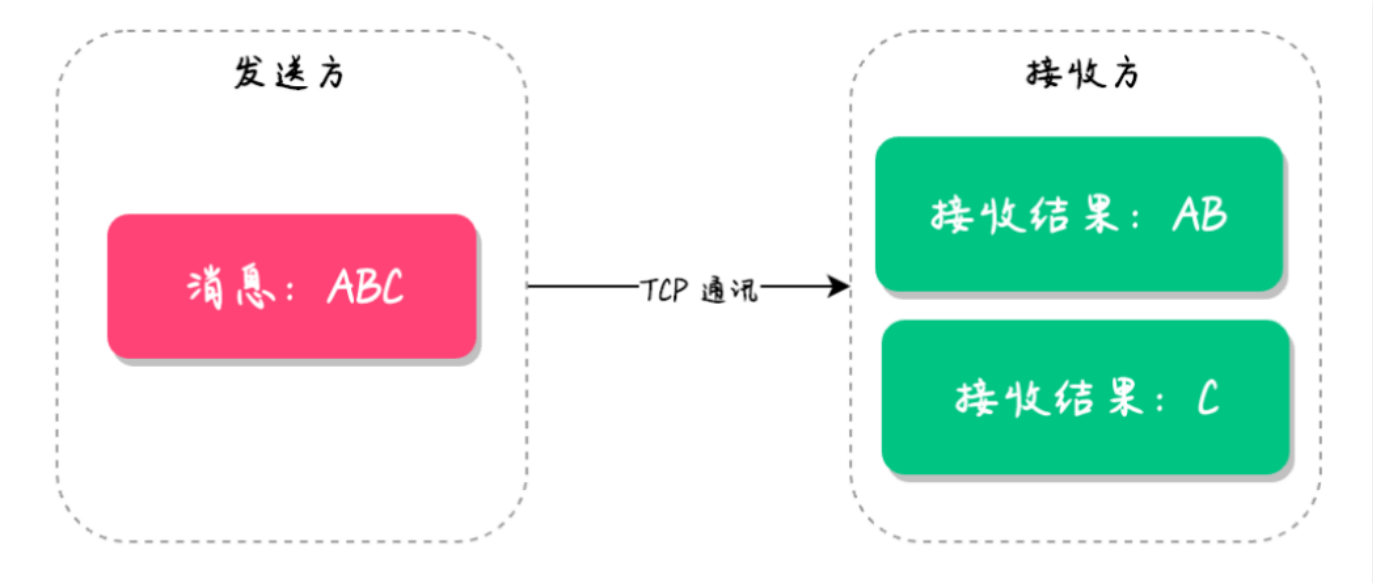
粘包问题是指发送方的多个数据在接收方接收时被“粘”在一起，而不是按照发送方发送的原始数据包的长度

比如发送了 ABC 和 DEF，但另一端接收到的却是 ABCD



半包问题是指只接收到部分数据，而不是完整的数据，可能时因为网络传输异常或者接受缓冲区不足

当发送的消息是 ABC 时，另一端却接收到的是 AB 和 C 两条信息



造成这两个问题的原因：

粘包的主要原因：

- 发送方每次写入数据 < 套接字（Socket）缓冲区大小；
- 接收方读取套接字（Socket）缓冲区数据不够及时。

半包的主要原因：

- 发送方每次写入数据 > 套接字（Socket）缓冲区大小；
- 发送的数据大于协议的 MTU (Maximum Transmission Unit，最大传输单元)，因此必须拆包。

如果我们上网去查的话，一般会给出三种解决方案：

1. 固定长度（如果指定长度的空间用不完，增加了不必要的传输，通常会选择对没用的空间进行 0 填充）
2. 设计 数据长度 + 数据 的格式（客户端和服务端要采用相同的格式进行通信）
3. 以特殊字符结尾（如果数据中包含特殊字符将出现半包问题，所以你需要对规定的特殊字符如果在数据中出现要进行特殊处理，或者不允许它出现）

数据长度+数据 C++

```cpp
#include <iostream>
#include <memory>
#include <string>
#include <cstring>

// 客户端把需要发送的数据 转换成 指定的格式 （数据长度+数据）
void encapMessage(const char *Server_data, std::string &Result){
  char tmpbuf[1024];
  memset(tmpbuf,0,sizeof(tmpbuf));
  size_t len = strlen(Server_data); // 获取数据长度
  memcpy(tmpbuf,&len,4);
  memcpy(tmpbuf+4,Server_data,len);
  // 调用send方法发送 tmpbuf，这里我们暂时保存在Result中，相当于传递给客户端了
  Result.assign(tmpbuf,len+4);
}

//更加简洁的方法

/*
void encapMessage(const std::string &buf,std::string &Result){
  size_t len = buf.size();
  Result.append((char*)&len,4);
  Result.append(buf.c_str(),len);
}
*/


// 服务器把客户端发来的数据 进行处理，取出传递的实际数据
bool pickMessage(std::string &Client_data,std::string &inputBuffer){
  int len = 0;
  memcpy(&len,Client_data.data(),4);    //拿到 数据长度
  if (Client_data.size() < len + 4) return false;        //报文内容不完整
  inputBuffer = Client_data.substr(4,len);  // 切割掉 数据长度，留下 数据
  return true;
}

int main() {

  const char *Sdata = "xiaoyangst11";
  std::string Ssend;
  std::string Crecv;
  std::cout<<"传递待封装的数据："<<Sdata<<std::endl;
  encapMessage(Sdata,Ssend);
  std::cout<<"传递封装好的数据："<<Ssend<<std::endl;
```

```
    pickMessage(Ssend,Crecv);
    std::cout<<"解析封装的数据的结果：  "<<Crecv<<std::endl;
    return 0;
}
```

测试：

```
C:\Users\xy\Desktop\Project\Demo\cmake-build-debug\Demo.exe
传递待封装的数据：xiaoyangst11
传递封装好的数据：□□□□xiaoyangst11
解析封装的数据的结果：xiaoyangst11


进程已结束，退出代码为 0
```

https://mp.weixin.qq.com/s/ODxGlLrohCveH-2m-BSDWQ