一、 UDP 概述

UDP 只是在IP 的数据报服务上增加了一点复用和分用的功能以及差错检验的功能。

1. UDP 是无连接的。发送数据之前不用建立连接（套接字socket）-->减少了开销和发送数据之前的延时。

2. UDP 尽最大努力交付。即不保证可靠交付。

3. UDP 是面向报文的。UDP 一次交付一个报文，既不合并，也不拆分。所以应用层应该选择合适大小的报文，过大（IP 层可能进行分片）或过小（数据报的首部的相对长度太大）都会降低IP 层的效率。

4. UDP 没有拥塞控制。

5. UDP 支持一对一、一对多、多对多的交互通信。

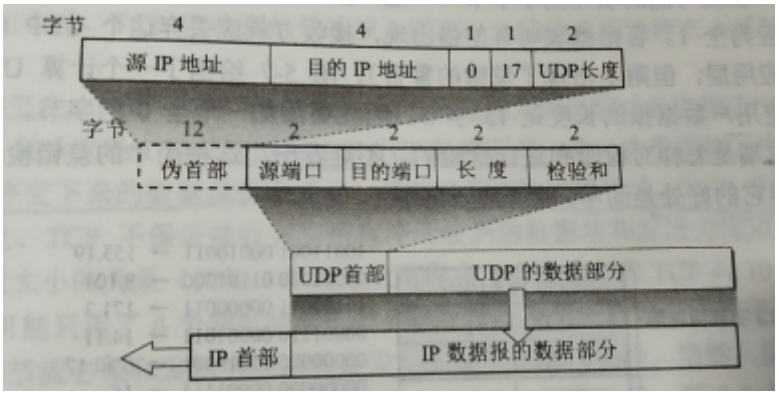
6. UDP 的首部开销小。8个字节，TCP 20个字节。

二、UDP的首部格式

首部字段由4 部分组成，每部分2 个字节，共8 个字节。

源端口号 (16位)	目的端口号 (16位)
UDP长度 (16位)	UDP校验和 (16位)
数据	

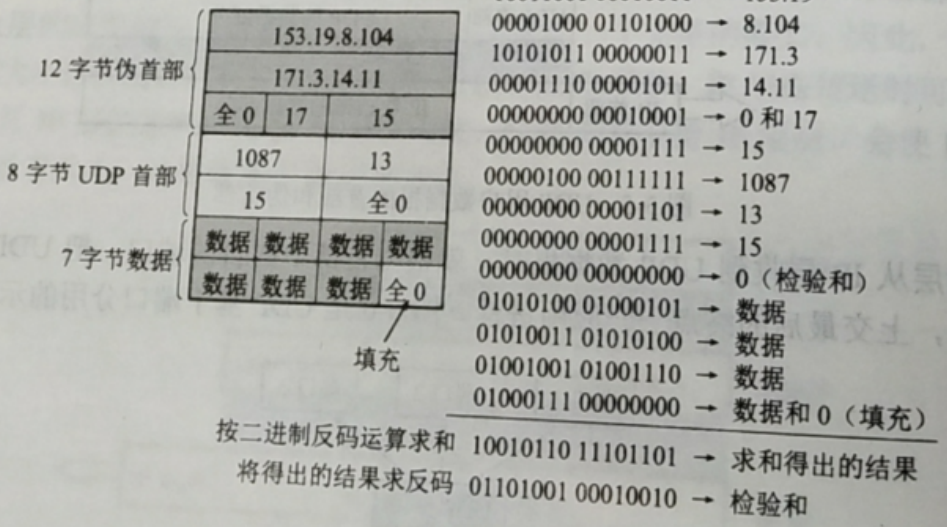
1. 源端口
2. 目的端口
3. 长度：用户数据报的长度，其最小值为8（仅有首部）
4. 检验和：检验用户数据报在传输中是否有错，有错则丢弃。



在计算检验和时，会在UDP 用户数据报之前增加一个伪首部，伪首部不向上/下传递，仅仅用于检验。

送也不向上递交，而仅仅是为了计算检验和。

UDP 计算检验和的方法和计算 IP 数据报首部检验和的方法相似。但不同的是：IP 数据报的检验和只检验 IP 数据报的首部，但 UDP 的检验和是**把首部和数据部分一起都检验**。在发送方，首先是先把全零放入检验和字段。再把伪首部以及 UDP 用户数据报看成是由许多 16 位的字串接起来的。若 UDP 用户数据报的数据部分不是偶数个字节，则要填入一个全零字节（但此字节不发送）。然后按二进制反码计算出这些 16 位字的和。将此和的二进制反码写入检验和字段后，就发送这样的 UDP 用户数据报。在接收方，把收到的 UDP 用户数据报连同伪首部（以及可能的填充全零字节）一起，按二进制反码求这些 16 位字的和。当无差错时其结果应为全 1。否则就表明有差错出现，接收方就应丢弃这个 UDP 用户数据报（也可以上交给应用层，但附上出现了差错的警告）。图 5-7 给出了一个计算 UDP 检验和的例子。这里假定用户数据报的长度是 15 字节，因此要添加一个全 0 的字节。读者可以自己检验一下在接收端是怎样对检验和进行检验的。不难看出，这种简单的差错检验方法的检错能力并不强，但它的好处是简单，处理起来较快。



三、UDP 应用

基于 UDP 的“城会玩”的五个例子

我列举几种“城会玩”的例子。

“城会玩”一：网页或者 APP 的访问

原来访问网页和手机 APP 都是基于 HTTP 协议的。HTTP 协议是基于 TCP 的，建立连接都需要多次交互，对于时延比较大的目前主流的移动互联网来讲，建立一次连接需要的时间会比较长，然而既然是移动中，TCP 可能还会断了重连，也是很耗时的。而且目前的 HTTP 协议，往往采取多个数据通道共享一个连接的情况，这样本来为了加快传输速度，但是 TCP 的严格顺序策略使得哪怕共享通道，前一个不来，后一个和前一个即便没关系，也要等着，时延也会加大。

而**QUIC**（全称**Quick UDP Internet Connections**，**快速 UDP 互联网连接**）是 Google 提出的一种基于 UDP 改进的通信协议，其目的是降低网络通信的延迟，提供更好的用户互动体验。

QUIC 在应用层上，会自己实现快速连接建立、减少重传时延，自适应拥塞控制，是应用层“城会玩”的代表。这一节主要是讲 UDP，QUIC 我们放到应用层去讲。

“城会玩”三：实时游戏

游戏有一个特点，就是实时性比较高。快一秒你干掉别人，慢一秒你被别人爆头，所以很多职业玩家会买非常专业的鼠标和键盘，争分夺秒。

因而，实时游戏中客户端和服务端要建立长连接，来保证实时传输。但是游戏玩家很多，服务器却不多。由于维护 TCP 连接需要在内核维护一些数据结构，因而一台机器能够支撑的 TCP 连接数目是有限的，然后 UDP 由于是没有连接的，在异步 IO 机制引入之前，常常是应对海量客户端连接的策略。

另外还是 TCP 的强顺序问题，对战的游戏，对网络的要求很简单，玩家通过客户端发送给服务器鼠标和键盘行走的位置，服务器会处理每个用户发送过来的所有场景，处理完再返回给客户端，客户端解析响应，渲染最新的场景展示给玩家。

如果出现一个数据包丢失，所有事情都需要停下来等待这个数据包重发。客户端会出现等待接收数据，然而玩家并不关心过期的数据，激战中卡 1 秒，等能动了都已经死了。

游戏对实时要求较为严格的情况下，采用自定义的可靠 UDP 协议，自定义重传策略，能够把丢包产生的延迟降到最低，尽量减少网络问题对游戏性造成的影响。

“城会玩”四：IoT 物联网

一方面，物联网领域终端资源少，很可能只是个内存非常小的嵌入式系统，而维护 TCP 协议代价太大；另一方面，物联网对实时性要求也很高，而 TCP 还是因为上面的那些原因导致时延大。
Google 旗下的 Nest 建立 Thread Group，推出了物联网通信协议 Thread，就是基于 UDP 协议的。

“城会玩”五：移动通信领域

在 4G 网络里，移动流量上网的数据面对的协议 GTP-U 是基于 UDP 的。因为移动网络协议比较复杂，而 GTP 协议本身就包含复杂的手机上线下线的通信协议。如果基于 TCP，TCP 的机制就显得非常多余，这部分协议我会在后面的章节单独讲解。