3.3.1 UDP 报文段结构

UDP 报文段结构如图 3-7 所示，它由 RFC 768 定义。应用层数据占用 UDP 报文段的数据字段。例如，对于 DNS 应用，数据字段要么包含一个查询报文，要么包含一个响应报文。对于流式音频应用，音频抽样数据填充到数据字段。UDP 首部只有 4 个字段，每个字段由两个字节组成。如前一节所讨论的，通过端口号可以使目的主机将应用数据交给运行在目的端系统中的相应进程（即执行分解功能）。长度字段指示了在 UDP 报文段中的字节数（首部加数据）。因为数据字段的长度在一个 UDP 段中不同于在另一个段中，故需要一个明确的长度。接收方使用检验和来检查在该报文段中是否出现了差错。实际上，计算检验和时，除了 UDP 报文段以外还包括了 IP 首部的一些字段。但是我们忽略这些细节，以便能从整体上看问题。下面我们将讨论检验和的计算。在 5.2 节中将描述差错检测的基本原理。长度字段指明了包括首部在内的 UDP 报文段长度（以字节为单位）。

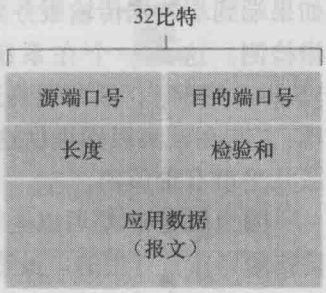

图 3-7 UDP 报文段结构

3.3.2 UDP 检验和

UDP 检验和提供了差错检测功能。这就是说，检验和用于确定当 UDP 报文段从源到达目的地移动时，其中的比特是否发生了改变（例如，由于链路中的噪声干扰或者存储在路由器中时引入问题）。发送方的 UDP 对报文段中的所有 16 比特字的和进行反码运算，求和时遇到的任何溢出都被回卷。得到的结果被放在 UDP 报文段中的检验和字段。下面给出一个计算检验和的简单例子。在 RFC 1071 中可以找到有效实现的细节，还可在［Stone 1998；Stone 2000］中找到它处理真实数据的性能。举例来说，假定我们有下面 3 个 16 比特的字：

0110011001100000
0101010101010101
1000111100001100

这些 16 比特字的前两个之和是：

0110011001100000
0101010101010101
1011101110110101

再将上面的和与第三个字相加，得出：

1011101110110101
1000111100001100
0100101011000010

注意到最后一次加法有溢出，它要被回卷。反码运算就是将所有的 0 换成 1，所有的 1 转换成 0。因此，该和 0100101011000010 的反码运算结果是 1011010100111101，这变为了检验和。在接收方，全部的 4 个 16 比特字（包括检验和）加在一起。如果该分组中没有引入差错，则显然在接收方处该和将是 1111111111111111。如果这些比特之一是 0，那