

二进制数据的编码、解码方法和装置

技术领域

本申请涉及网络通信技术领域，尤其涉及一种二进制数据的编码方法和装置、一种二进制数据的解码方法和装置。

5 背景技术

在计算机中很多资源以二进制数据的形式存储，例如图片、音视频流、可执行程序等，包含这些资源的文件通常都是二进制文件。但是在一些网络协议或网络应用中，能够正确传输的数据只能是通用字符，二进制数据在基于这些网络协议或网络应用进行传输时，需要先在发送端将二进制数据编码
10 为字符数据，在传输到接收端后，再将字符数据解码为二进制数据。

现有技术中，Base64 是最为常用的二进制数据编解码技术。Base64 采用 64 个字符对二进制数据进行编码，对 N 字节的二进制数据，将对应的 $8*N$ 位中连续的每 6 位分割为 1 个部分，每个部分的值都在 0 到 63 之间，将该值对应于 1 个 ASCII (American Standard Code for Information Interchange, 美国信息交换标准代码) 字符，将所有部分对应的 ASCII 字符拼接起来，即可
15 得到二进制数据的 Base64 编码。对应的解码过程与上述相逆，不再赘述。

可见，3 字节二进制数据的 Base64 编码数据的长度为 4 字节，一个二进制文件在转换为 Base64 编码文件后，文件长度增加了大约 33%。也就是说，Base64 编码后的二进制文件在传输时要占用的流量增加了三分之一左右。在
20 移动互联应用中，降低服务端与用户手持设备之间的交互流量对所有业务而言都是至关重要的问题。Base64 编码后的字符文件较大，难以满足一些苛刻的应用场景的要求。

发明内容

有鉴于此，本申请提供一种二进制数据的编码方法，包括：

获取二进制数据；所述二进制数据的长度为 k 字节， k 为自然数；

采用预定编码算法将 k 字节的二进制数据转换为 $(k+t)$ 字节的 7 位编码数据， t 为大于等于 k 除以 7 的最小整数。

本申请提供的一种二进制数据的解码方法，包括：

- 5 获取待解码的 7 位编码数据；所述 7 位编码数据的长度为 p 字节， p 为大于 1 的自然数；

采用预定解码算法将 p 字节的 7 位编码数据转换为 $(p-t)$ 字节的二进制数据， t 为大于等于 p 除以 8 的最小整数。

本申请还提供了一种二进制数据的编码装置，包括：

- 10 二进制数据获取单元，用于获取二进制数据；所述二进制数据的长度为 k 字节， k 为自然数；

7 位编码单元，用于采用预定编码算法将 k 字节的二进制数据转换为 $(k+t)$ 字节的 7 位编码数据， t 为大于等于 k 除以 7 的最小整数。

本申请提供的一种二进制数据的解码装置，包括：

- 15 待解码数据获取单元，用于获取待解码的 7 位编码数据；所述 7 位编码数据的长度为 p 字节， p 为大于 1 的自然数；

7 位解码单元，用于采用预定解码算法将 p 字节的 7 位编码数据转换为 $(p-t)$ 字节的二进制数据， t 为大于等于 p 除以 8 的最小整数。

- 20 由以上技术方案可见，本申请的实施例，在编码时，采用预定编码算法对二进制数据进行 7 位编码；在解码时，采用预定解码算法将 7 位编码恢复为二进制数据；由于 7 字节的二进制数据可以转换为 8 字节的 7 位编码数据，二进制数据在转换为 7 位编码数据后长度大约增加了 14.3%，因此采用本申请技术方案得到的编码文件比 Base64 编码文件的长度更小，在网络上传输时消耗的流量更少，能够适应更为苛刻的应用场景要求。

25 附图说明

图 1 是本申请实施例中一种二进制数据的编码方法的流程图；

图 2 是本申请实施例中一种二进制数据的解码方法的流程图;

图 3 是本申请实施例中第一种预定编码算法示例的示意图;

图 4 是本申请实施例中第二种预定编码算法示例的示意图;

图 5 是运行本申请实施例的设备的一种硬件结构图;

5 图 6 是本申请实施例中一种二进制数据的编码装置的逻辑结构图;

图 7 是本申请实施例中一种二进制数据的解码装置的逻辑结构图。

具体实施方式

在绝大多数编码方式中,十进制值 0 到 127 所对应的字符在网络传输中能够被各种网络协议或网络应用支持。例如,ASCII 编码、UTF-7 (Unicode Transformation Format 7, 7 位元 Unicode 转换格式)、7 位等宽编码等。也
10 就是说,在将二进制文件转换为上述这些编码方式的 7 位编码文件后,即可在各种应用中、基于各种协议进行文件传输。

7 位编码是采用一个字节的低 7 位来承载数据的编码方式。7 位编码可以将 7 字节的二进制数据转换为 8 字节的编码数据,相比于 Base64 编码将 3
15 字节的二进制数据转换为 4 字节的编码,其转换效率更高,转换后的编码文件更小,更为节省传输流量。

因此,本申请的实施例提出一种新的二进制数据的编码方法和对应的二进制数据的解码方法,分别用来将二进制数据转换为 7 位编码数据、以及将 7 位编码数据还原为二进制数据,以减小编码数据的长度,节约编码数据在
20 网络传输时消耗的流量,从而解决现有技术中存在的问题。

本申请的实施例可以应用在任何具有计算和存储能力的设备上,例如可以是手机、平板电脑、PC (Personal Computer, 个人电脑)、笔记本、服务器、虚拟机等物理设备或逻辑设备;也可以由两个或两个以上分担不同职责的物理或逻辑设备、相互协同来实现本申请实施例中的各项功能。其中,编
25 码方法和解码方法可以分别运行在不同的设备上,也可以运行在同一个设备上。

本申请的实施例中，二进制数据的编码方法的流程如图 1 所示。

步骤 110，获取二进制数据。

待转换的二进制数据可以是一个二进制文件，也可以是作为一个文件组成部分的一段二进制数据，不做限定。可以从某个网络中的存储位置读取二进制数据，也可以从指定的缓存中获得待转换的二进制数据，本申请的实施例同样不做限定。

设所获取的二进制数据的长度为 k (k 为自然数) 字节。

步骤 120，采用预定编码算法将 k 字节的二进制数据转换为 $(k+t)$ 字节的 7 位编码数据， t 为大于等于 k 除以 7 的最小整数。

10 一个字节二进制数据有效位是 8 位，而一个字节的 7 位编码数据有效位是 7 位。因此，在将二进制数据转换为 7 位编码数据时，1 到 7 个字节的二进制数据分别可以由 2 到 8 个字节的 7 位编码数据来表达，编码后增加的长度为 1 个字节；8 到 14 字节的二进制数据分别可以由 10 到 16 个字节的 7 位编码数据来表达，编码后增加的长度为 2 个字节；依次类推， k 字节的二进制数据可以由 $(k+t)$ 字节的 7 位编码数据来表达， t 为大于等于 k 除以 7 的最小整数。换言之，在将 k 字节的二进制数据转换为 7 位编码数据时，增加的长度为 t 个字节。

任何一种可以将 k 字节二进制数据一对一映射为 $(k+t)$ 字节的 7 位编码数据的算法都可以用作预定编码算法，本申请的实施例不做限定。

20 本申请的实施例中，二进制数据的解码方法的流程如图 2 所示。

步骤 210，获取待解码的 7 位编码数据。

类似的，待解码的 7 位编码数据可以是一个编码文件，也可以是作为一个文件组成部分的一段 7 位编码数据，不做限定。可以从某个网络中的存储位置读取 7 位编码数据，也可以从指定的缓存中获得待解码的 7 位编码数据，
25 本申请的实施例同样不做限定。

设待解码的 7 位编码数据的长度为 p (p 为大于 1 的自然数) 字节。

步骤 220，采用预定解码算法将 p 字节的 7 位编码数据转换为 $(p-t)$ 字

节的二进制数据， t 为大于等于 p 除以 8 的最小整数。

如前所述，2 到 8 个字节的 7 位编码数据分别可以表达 1 到 7 个字节的二进制数据，解码后减少的长度为 1 个字节；10 到 16 个字节的 7 位编码数据分别可以表达 8 到 14 字节的二进制数据，解码后减少的长度为 2 个字节；
5 依次类推， p 字节的 7 位编码数据可以表达 $(p-t)$ 字节的二进制数据， t 为大于等于 p 除以 8 的最小整数。换言之，在将 p 字节的 7 位编码数据转换为二进制数据时，减少的长度为 t 个字节。

与将二进制数据转换为 7 位编码数据时采用的预定编码算法相逆、能够将 p 字节的 7 位编码数据还原为编码前的 $(p-t)$ 字节的二进制数据的算法，
10 即可用来作为本申请实施例中的预定解码算法。

在一种实现方式中，可以基于以下思路来进行编码：一字节的二进制数据由 8 个位 (bit) 组成，由于 7 位编码只能使用一字节中的低 7 位，一字节的二进制数据需要一字节的 7 位编码和一个额外字节中的 1 位来表示。可以将一字节的 8 位二进制数据分拆为 2 个部分，将某个预定位（可以是第 0 位
15 到第 7 位中的任何一位）作为由额外字节表示的 1 位，除该预定位外剩余的 7 位由一字节的 7 位编码来表示。该额外字节中的其他 6 位可以用于携带其他 6 个字节的二进制数据的预定位。这样，每 7 个字节的二进制数据在映射为 7 位编码时增加 1 个字节，不足 7 个字节的二进制数据在映射 7 位编码时也会增加 1 个字节，则 k 字节的二进制数据在映射为 7 位编码时，将增加 t
20 个字节。

具体而言，这种实现方式中的预定编码算法可以按如下方式运行：在 k 字节二进制数据中提取每个字节的预定位，按照设定位顺序组合为 t 字节 7 位编码的组合位数据，将 k 字节二进制数据中每个字节剩余的 7 位作为 k 字节 7 位编码的剩余位数据；将 k 字节 7 位编码的剩余位数据和 t 字节 7 位编码的组合位数据按照设定字节顺序进行排列，从而得到 $(k+t)$ 字节的 7 位编码数据。
25

对应的，在解码时，这种实现方式的预定解码算法可以按如下方式运行：

由于 p 等于 $(k+t)$ ， p 字节 7 位编码数据中包括按照设定字节顺序排列的 t 字节 7 位编码的组合位数据和 k 字节 7 位编码的剩余位数据；根据设定字节顺序区分出哪些是 t 字节的组合位数据、哪些是 k 字节的剩余位数据后，按照设定位顺序从 t 字节的组合位数据中提取 k 位，将提取的每 1 位插入与该
5 位的设定位顺序对应的 k 字节剩余位数据的预定位，从而得到 k 字节二进制数据。

其中，设定位顺序和设定字节顺序可以任意设置，只要预定编码算法和预定解码算法采用相同的设定位顺序和设定字节顺序即可。例如，设定位顺序可以从一个字节组合位数据的第 0 位到第 $(s-1)$ 位、或从第 $(s-1)$ 位到第
10 0 位，依次对应于连续的 s (s 为不大于 7 的自然数) 字节剩余位数据；再如，设定字节顺序可以是 k 字节剩余位数据在 t 字节组合位数据之前，或 k 字节剩余位数据在 t 字节组合位数据之后，并且 k 字节剩余位数据按照所对应的二进制数据排列。

以下给出上述实现方式的两个具体例子：

15 第一个例子：请参见图 3，在一种预定编码算法中，可以将最高位作为预定位，依次将 k 字节二进制数据的最高位（第 7 位）按照从第 1 字节组合位数据的第 6 位到第 0 位、第 2 字节组合位数据的第 6 位到第 0 位这样的设定位顺序，生成 t 字节的组合位数据；将 k 字节二进制数据中的每个字节第
20 0 位到第 6 位作为 k 字节的剩余位数据；将 t 字节的组合位数据排列在 k 字节的剩余位数据之前，得到 k 字节二进制数据的 $(k+t)$ 字节 7 位编码数据。

在第一个例子的预定解码算法中，将 p ($p=k+t$) 字节 7 位编码数据的前 t 个字节作为组合位数据，后 k 个字节作为剩余位数据；从 t 字节的组合位数据中按照从第 1 字节组合位数据的第 6 位到第 0 位、第 2 字节组合位数据的第 6 位到第 0 位这样的设定位顺序，提取出 k 个预定位；将顺序提取出的 k
25 个预定位依次插入到 k 个剩余位数据的最高位，插入预定位后即可得到 k 字节的二进制数据。

第二个例子：请参见图 4，在另一种预定编码算法中，以第 4 位作为预

定位，将连续 7 字节的二进制数据分为一组，如果最后剩余不足 7 字节也作为一组，则 k 字节二进制数据将形成 t 个组。以组为单位，提取一组二进制数据中每个字节的预定位，按照第 0 位到第 6 位的设定位顺序生成一字节的组合位数据，将该组中每个字节二进制数据的第 5 位到第 7 位右移一位，得到 7 位编码的剩余位数据。将每组的组合位数据排列在该组的剩余位数据之前，以组为单位依次排列 t 个组，得到 (k+t) 字节的 7 位编码数据。

在第二个例子的预定解码算法中，将连续 8 字节的 7 位编码数据分为一组，如果最后剩余不足 8 字节也作为一组，则 p (p=k+t) 字节二进制数据将形成 t 个组。以组为单位，设某组中共 (s+1) (s 为不大于 7 的自然数) 个字节，则该组中第 1 个字节为组合位数据，第 2 到第 (s+1) 个字节为 s 字节的剩余位数据；将该组中组合位数据的第 0 位到第 s 位分别插入到该组中第 2 到第 (s+1) 字节剩余位数据的第 4 位，得到 s 字节的二进制数据。将从各个组得到的二进制数据依序排列，则可得到解码后 k 字节的二进制数据。

可见，本申请的实施例，在编码时，采用预定编码算法将二进制数据转换为 7 位编码数据；在解码时，采用对应的预定解码算法将 7 位编码数据还原为二进制数据。由于二进制数据在转换为 7 位编码数据后长度大约增加了 14.3%，远小于 Base64 编码增加的长度，应用本申请的实施例得到的编码文件在网络上传输时消耗的流量更少，能够适应更为苛刻的应用场景要求。

在本申请的一个应用示例中，采用一种编码算法将图片 origin.jpg 的数据转换为 7 位 ASCII 编码数据，对应的一种解码算法将 7 位 ASCII 编码数据还原为二进制数据。

实现编码方法的 JavaScript 程序如下：

```
var buf = new Buffer(1024); // 开辟编码时使用的缓冲区  
var data = fs.readFileSync('origin.jpg'); // data 为图片 origin.jpg 的二进制数据  
var length = data.length + Math.ceil(data.length / 7); // length 为转换后 7 位编码数据的长度
```

```
var body = new Buffer(length); //body 为转换后的 7 位编码数据
var offset = 0;
var swap = 0;
var count = 0;
5   for (var i = 0; i < data.length; i++) {
    swap |= ((data[i] & 0x80) >> (i % 7 + 1));
    count++;
    if (i === data.length - 1 || count % 7 === 0) {
      body.writeUInt8(swap, ++offset);
10   swap = 0;
    }
  }
  //从 data 的第一个字节开始,依次将每 7 个字节的最高位组成一个 ASCII
  码的组合位数据,其中每 7 个字节中第 7 个字节二进制数据的最高位在第 0
  位,第 1 个字节二进制数据的最高位在第 6 位;直到 data 的最后一个字节;
15 将组合位数据逐个写入 body
    for (var i = 0; i < data.length; i++) {
      body.writeUInt8(data[i] & 0x7f, ++offset);
    }
    //从 data 的第一个字节开始,依次将每个字节的低 7 位生成一个 ASCII
    码的剩余位数据,并且逐个写入 body
20   设 temp 为待解码的 7 位编码数据,实现解码方法的 JavaScript 程序如下:
    var len = temp.length - Math.ceil(temp.length / 8); //len 为解码后二进制数
    据的长度
    var buf = new Buffer(len); //buf 为解码后的二进制数据
    for (var i = 0; i < buf.length; i++) {
25   var swap = 0;
      swap |= ((temp.readUInt8(parseInt(i / 7)) << (i % 7 + 1)) & 0x80);
      swap |= temp.readUInt8(i + temp.length - len);
      buf.writeUInt8(swap, i);
    } //temp 的前 (temp.length - len) 个字节为组合位数据,之后为剩余位数
```

据;从组合位数据的第一个字节开始,依次提取每个字节的第 6 位到第 0 位,所提取的每位依次插入到剩余位数据的最高位,并写入 buf

与上述流程实现对应,本申请的实施例还提供了一种二进制数据的编码装置,和一种二进制数据的解码装置。这两种装置均可以通过软件实现,也可以通过硬件或者软硬件结合的方式实现。以软件实现为例,作为逻辑意义上的装置,是通过所在设备的 CPU (Central Process Unit, 中央处理器) 将对应的计算机程序指令读取到内存中运行形成的。从硬件层面而言,除了图 5 所示的 CPU、内存以及非易失性存储器之外,二进制数据的编码装置或二进制数据的解码装置所在的设备通常还包括用于进行无线信号收发的芯片等其他硬件,和/或用于实现网络通信功能的板卡等其他硬件。

图 6 所示为本申请实施例提供的一种二进制数据的编码装置,包括二进制数据获取单元和 7 位编码单元,其中:二进制数据获取单元用于获取二进制数据;所述二进制数据的长度为 k 字节, k 为自然数; 7 位编码单元用于采用预定编码算法将 k 字节的二进制数据转换为 (k+t) 字节的 7 位编码数据, t 为大于等于 k 除以 7 的最小整数。

一种实现方式中,所述预定编码算法包括:在 k 字节二进制数据中提取每个字节的预定位,按照设定位顺序组合为 t 字节 7 位编码的组合位数据;将 k 字节二进制数据中每个字节剩余的 7 位作为 k 字节 7 位编码的剩余位数据;将 k 字节 7 位编码的剩余位数据和 t 字节 7 位编码的组合位数据按照设定字节顺序进行排列,生成所述二进制数据的 7 位编码数据。

上述实现方式中,所述设定字节顺序包括: k 字节剩余位数据在 t 字节组合位数据之前或之后,按照所对应的二进制数据排列。

上述实现方式中,所述设定位顺序包括:将连续的 s 字节二进制数据中提取的预定位依次作为一个字节组合位数据的第 0 位到第 (s-1) 位、或第 (s-1) 位到第 0 位; s 为不大于 7 的自然数。

上述实现方式中,所述预定位为每个字节的最高位。

图 7 所示为本申请实施例提供的一种二进制数据的解码装置,包括待解

码数据获取单元和 7 位解码单元，其中：待解码数据获取单元用于获取待解码的 7 位编码数据；所述 7 位编码数据的长度为 p 字节， p 为大于 1 的自然数；7 位解码单元用于采用预定解码算法将 p 字节的 7 位编码数据转换为 $(p-t)$ 字节的二进制数据， t 为大于等于 p 除以 8 的最小整数。

5 一种实现方式中，所述预定解码算法包括：按照设定字节顺序，从 p 字节的 7 位编码数据中获取 k 字节的剩余位数据和 t 字节的组合位数据， k 为 p 减去 t 的差值；从 t 字节 7 位编码的组合位数据中提取 k 位，将提取的每 1 位插入与所述位的设定位顺序对应的 k 字节剩余位数据的预定位，得到 k 字节二进制数据。

10 上述实现方式中，所述设定字节顺序包括： k 字节剩余位数据在 t 字节组合位数据之前或之后，按照所对应的二进制数据排列。

 上述实现方式中，所述设定位顺序包括：一个字节组合位数据的第 0 位到第 $(s-1)$ 位、或第 $(s-1)$ 位到第 0 位，依次对应于连续的 s 字节剩余位数据； s 为不大于 7 的自然数。

15 上述实现方式中，所述预定位为每个字节的最高位。

 以上所述仅为本申请的较佳实施例而已，并不用以限制本申请，凡在本申请的精神和原则之内，所做的任何修改、等同替换、改进等，均应包含在本申请保护的范围之内。

20 在一个典型的配置中，计算设备包括一个或多个处理器（CPU）、输入/输出接口、网络接口和内存。

 内存可能包括计算机可读介质中的非永久性存储器，随机存取存储器（RAM）和/或非易失性内存等形式，如只读存储器（ROM）或闪存（flash RAM）。内存是计算机可读介质的示例。

25 计算机可读介质包括永久性和非永久性、可移动和非可移动媒体可以由任何方法或技术来实现信息存储。信息可以是计算机可读指令、数据结构、程序的模块或其他数据。计算机的存储介质的例子包括，但不限于相变内存（PRAM）、静态随机存取存储器（SRAM）、动态随机存取存储器（DRAM）、其他类型的随

机存取存储器 (RAM)、只读存储器 (ROM)、电可擦除可编程只读存储器 (EEPROM)、快闪记忆体或其他内存技术、只读光盘只读存储器 (CD-ROM)、数字多功能光盘 (DVD) 或其他光学存储、磁盒式磁带, 磁带磁磁盘存储或其他磁性存储设备或任何其他非传输介质, 可用于存储可以被计算设备访问的信息。按照本文中的界定, 计算机可读介质不包括暂存电脑可读媒体 (transitory media), 如调制的数据信号和载波。

还需要说明的是, 术语“包括”、“包含”或者其任何其他变体意在涵盖非排他性的包含, 从而使得包括一系列要素的过程、方法、商品或者设备不仅包括那些要素, 而且还包括没有明确列出的其他要素, 或者是还包括为这种过程、方法、商品或者设备所固有的要素。在没有更多限制的情况下, 由语句“包括一个……”限定的要素, 并不排除在包括所述要素的过程、方法、商品或者设备中还存在另外的相同要素。

本领域技术人员应明白, 本申请的实施例可提供为方法、系统或计算机程序产品。因此, 本申请可采用完全硬件实施例、完全软件实施例或结合软件和硬件方面的实施例的形式。而且, 本申请可采用在一个或多个其中包含有计算机可用程序代码的计算机可用存储介质 (包括但不限于磁盘存储器、CD-ROM、光学存储器等) 上实施的计算机程序产品的形式。