[19] 中华人民共和国国家知识产权局

[51] Int. Cl[7]
H04L 29/06

[12] 发明专利申请公开说明书

[21] 申请号 01806173.7

[43] 公开日 2003 年 5 月 21 日

[11] 公开号 CN 1419771A

[22] 申请日 2001.3.9 [21] 申请号 01806173.7

[30] 优先权

[32] 2000. 3. 9 [33] US [31] 09/522,363

[86] 国际申请 PCT/US01/07573 2001.3.9

[87] 国际公布 WO01/67709 英 2001.9.13

[85] 进入国家阶段日期 2002.9.6

[71] 申请人 诺基亚有限公司

地址 芬兰埃斯波

共同申请人 希姆·列

[72] 发明人 希姆·列

[74] 专利代理机构 中国专利代理(香港)有限公司

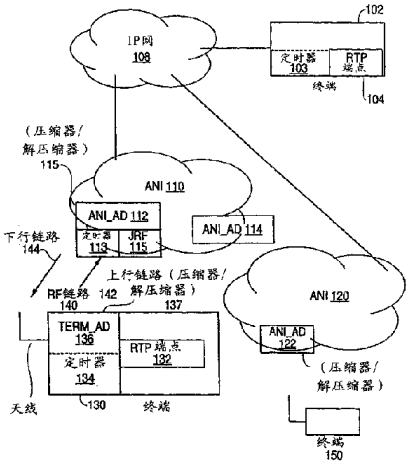
代理人 栾本生 罗 朋

权利要求书 7 页 说明书 42 页 附图 12 页

[54] 发明名称 压缩数据分组中头域的技术

[57] 摘要

提供了一种基于定时器的头压缩/解压缩技术以及一种基于定时器和参考的技术。 源生成一个头域，如 RTP 时间戳。 包括该头域的分组发送到压缩器，压缩器基于来自源的分组的头域以及抖动质量来计算压缩的头域。 通过计算压缩器之前的网络在分组发送上的抖动影响以及计算压缩器和解压缩器之间的网络在分组发送上的抖动影响来计算压缩的头域。 包括压缩头域的分组发送到包括本地定时器的解压缩器。 解压缩器通过基于自从前一个分组到达以来流逝的时间以及前一个分组中的域值计算头域的近似值，来解压缩压缩的头域。 然后基于分组中提供的压缩的头域来纠正头域的近似值。



ISSN1008-4274

1. 一种在网络中源和接收器之间利用基于定时器的压缩技术来发送当前分组的当前头域的方法，包括:

从压缩器向解压缩器提供头域的初始值;

基于当前分组的当前头域和抖动来在压缩器计算当前分组的压缩的头域，

其中所述计算步骤包括步骤:

计算源和所述解压缩器之间的网络在分组发送上的抖动影响，以及

计算作为域值一部分的压缩头域，所述部分是抖动的函数;

在解压缩器接收当前分组的压缩头域;

在解压缩器处基于当前分组的压缩头域的接收和解压缩的前一个分组的头域的接收之间流逝的时间以及前一个分组的解压缩的域值来估算当前分组的头域; 以及

基于在解压缩器处接收的压缩的头域来改正估算的当前头域。

2. 根据权利要求 1 的方法，其中所述计算抖动影响包括步骤:

计算在压缩器之前的网络的抖动影响; 以及

计算压缩器和解压缩器之间的网络的抖动影响。

3. 根据权利要求 2 的方法，其中压缩器和解压缩器之间的网络的所述抖动影响设置为抖动的上界值。

4. 根据权利要求 2 的方法，其中所述计算在压缩器之前的网络的抖动影响包括:

利用关于参考分组的信息计算当前分组的抖动影响。

5. 根据权利要求 2 的方法，其中所述计算在压缩器之前的网络的抖动影响包括:

利用关于所述当前分组以及预定数量的前面分组的每个的信息计算当前分组的抖动影响。

6. 根据权利要求 2 的方法，其中所述计算在压缩器之前的网络的抖动影响包括:

利用关于所述当前分组以及直到参考分组的每个前面分组的信息计算当前分组的抖动影响。

7. 根据权利要求 1 的方法，其中所述在压缩器处计算当前分组

的压缩头域包括:

计算当前分组的压缩头域为域值的 k 个最低有效位，其中 k 是一个整数。

8. 根据权利要求 1 的方法，其中所述头域包括时间戳。

9. 根据权利要求 1 的方法，其中所述头域包括 RTP 时间戳。

10. 根据权利要求 1 的方法，其中所述在压缩器处计算当前分组的压缩头域包括:

将域值转换为另一个值，称为压缩值，其占用更少的比特来表示;以及

计算当前分组的压缩头域为所述压缩值的 k 个最低有效位，其中 k 是一个整数。

11. 一种解压缩网络中从压缩器发送到解压缩器的当前分组的当前头域的方法包括步骤:

在解压缩器处接收当前分组的压缩头域，所述压缩头域在压缩器中已经计算为域值的一部分，所述域值计算为源和所述解压缩器之间的网络在分组发送上的抖动影响;

在解压缩器处基于自从前一个压缩头域到达解压缩器以来流逝的时间以及前一个分组的解压缩的域值来计算当前分组的头域的近似值;

在解压缩器处基于当前分组的压缩头域来计算当前分组的头域纠正量;以及

通过将当前分组的头域的近似值调整一个基于头域纠正量的一个量来在解压缩器处解压缩当前分组的压缩头域。

12. 根据权利要求 11 的方法，其中通过计算压缩器之间的网络的抖动影响以及计算压缩器和解压缩器之间的网络的抖动影响来计算源和所述解压缩器之间的网络在分组发送上的所述抖动影响。

13. 根据权利要求 12 的方法，其中压缩器和解压缩器之间的网络的所述抖动影响设置为抖动的上界值。

14. 根据权利要求 12 的方法，其中所述计算在压缩器之前的网络的抖动影响包括:

利用关于参考分组的信息计算当前分组的抖动影响。

15. 根据权利要求 12 的方法，其中所述计算在压缩器之前的网络

的抖动影响包括:

利用关于所述当前分组以及预定数量的前面分组的每个的信息计算当前分组的抖动影响。

16. 根据权利要求 12 的方法，其中所述计算在压缩器之前的网络的抖动影响包括:

利用关于所述当前分组以及直到参考分组的每个前面分组的信息计算当前分组的抖动影响。

17. 根据权利要求 11 的方法，其中压缩头域计算为域值的 k 个最低有效位，其中 k 是一个整数。

18. 根据权利要求 11 的方法，其中压缩头域计算为压缩值的 k 个最低有效位，其中 k 是一个整数；并且

其中解压缩器基于自从前一个分组到达以来流逝的时间以及前一个分组的解压缩器压缩值来计算压缩值的近似值。

19. 一种在系统的第一个和第二个网络实体之间执行移交的方法，当移动解压缩器位于第一和第二个区域中时，第一和第二个网络实体分别将移动解压缩器接口到源终端，所述方法包括:

在第一个网络实体中以及在移动解压缩器中接收来自源终端的头域的初始值，第一个网络实体将位于第一个区域中的移动解压缩器接口到源终端;

在第一个网络实体中接收来自源终端的寻址到移动解压缩器的第一个分组的头域;

在第一个网络实体中压缩第一个分组的头域并且将第一个分组的第一个压缩头域发送到移动解压缩器，所述第一个压缩的头域计算为域值的一部分，所述域值计算为源终端和移动解压缩器之间的网络在分组发送上的第一个抖动影响;

在移动解压缩器中基于自从前一个分组到达以来流逝的时间以及前一个分组的解压缩域值来接收和解压缩第一个分组的第一个压缩的头域;

移动解压缩器从第一个区域移动到第二个区域;

将初始化信息发送到第二个网络实体以便为压缩而初始化第二个网络实体;

在第二个网络实体中接收并且压缩来自源终端寻址到移动解压缩

器的第二个分组的头域，并且将第二个分组的第二个压缩头域发送到移动解压缩器，所述第二个压缩头域计算为域值的一部分，所述域值计算为源终端和移动解压缩器之间的网络在分组发送上的第二个抖动影响以及向第二个网络实体发送初始化信息的时间；以及

在移动解压缩器基于自从前一个分组到达以来流逝的时间以及前一个分组的解压缩域值来接收和解压缩第二个分组的第二个压缩的头域。

20. 根据权利要求 15 的方法，其中通过计算压缩器之前的网络的抖动影响以及计算压缩器和解压缩器之间的网络的抖动影响来计算源终端和所述解压缩器之间的网络在分组发送上的所述第一个和第二个抖动影响的每一个。

21. 根据权利要求 20 的方法，其中压缩器和解压缩器之间的网络的所述抖动影响设置为抖动的上界值。

22. 根据权利要求 19 的方法，其中所述第一个分组是紧接移交之前的分组并且所述第二个分组是紧接移交之后的分组。

23. 根据权利要求 20 的方法，其中所述计算在压缩器之前的网络的抖动影响包括:

利用关于参考分组的信息计算当前分组的抖动影响。

24. 根据权利要求 20 的方法，其中所述计算在压缩器之前的网络的抖动影响包括:

利用关于所述当前分组以及预定数量的前面分组的每个的信息计算当前分组的抖动影响。

25. 根据权利要求 20 的方法，其中所述计算在压缩器之前的网络的抖动影响包括:

利用关于所述当前分组以及直到参考分组的每个前述分组的信息计算当前分组的抖动影响。

26. 根据权利要求 19 的方法，其中所述头域包括时间戳。

27. 根据权利要求 19 的方法，其中压缩头域计算为域值的 k 个最低有效位，其中 k 是一个整数。

28. 根据权利要求 19 的方法，其中压缩头域计算为压缩域值的 k 个最低有效位，其中 k 是一个整数；并且

其中解压缩器基于自从前一个分组到达以来流逝的时间以及前一

个分组的解压缩器压缩值来计算压缩值的近似值。

29. 一种在第一个和第二个网络实体之间执行移交的方法，当移动压缩器位于第一和第二个区域时，第一个和第二个网络实体分别将移动压缩器接口到接收器终端，所述方法包括:

在第一个网络实体中接收来自移动压缩器的头域的初始值;

在第一个网络实体中接收从移动压缩器接收的第一个分组的第一个压缩的头域，所述第一个压缩的头域在移动压缩器中计算为域值的一部分，所述域值计算为源和解压缩器之间的网络在分组发送上的第一个抖动影响;

在第一个网络实体中基于自从前一个分组到达以来流逝的时间以及前一个分组的解压缩域值来解压缩第一个分组的第一个压缩的头域;

移动压缩器从第一个区域移动到第二个区域;

将初始化信息发送到第二个网络实体以便为压缩初始化第二个网络实体;

在第二个网络实体中接收从移动压缩器接收的第二个分组的第二个压缩头域，所述第二个压缩头域在移动压缩器中计算为域值的一部分，所述域值计算为源和解压缩器之间的网络在分组发送上的第二个抖动影响以及向第二个网络实体发送初始化信息的时间；以及

在第二个网络实体中基于自从前一个分组到达以来流逝的时间以及前一个分组的解压缩域值来解压缩第二个分组的第二个压缩的头域。

30. 根据权利要求 29 的方法，其中通过计算移动压缩器之前的网络的抖动影响以及计算压缩器和解压缩器之间的网络的抖动影响来计算源和所述解压缩器之间的网络在分组发送上的所述第一个和第二个抖动影响。

31. 根据权利要求 30 的方法，其中压缩器和解压缩器之间的网络的所述抖动影响设置为抖动的上界值。

32. 根据权利要求 29 的方法，其中所述第一个分组是紧接移交之前的分组并且所述第二个分组是紧接移交之后的分组。

33. 根据权利要求 30 的方法，其中所述计算在移动压缩器之前的网络的抖动影响包括:

利用关于参考分组的信息计算当前分组的抖动影响。

34. 根据权利要求 30 的方法，其中所述计算在移动压缩器之前的网络的抖动影响包括:

利用关于所述当前分组以及预定数量的前面分组的每个的信息计算当前分组的抖动影响。

35. 根据权利要求 30 的方法，其中所述计算在移动压缩器之前的网络的抖动影响包括:

利用关于所述当前分组以及直到参考分组的每个前面分组的信息计算当前分组的抖动影响。

36. 根据权利要求 29 的方法，其中所述头域包括时间戳。

37. 根据权利要求 29 的方法，其中压缩头域计算为压缩值的 k 个最低有效位，其中 k 是一个整数；并且

其中解压缩器基于自从前一个分组到达以来流逝的时间以及前一个分组的解压缩的压缩值来计算压缩值的近似值。

38. 根据权利要求 29 的方法，其中压缩头域计算为压缩值的 k 个最低有效位，其中 k 是一个整数。

39. 一种通信系统包括:

提供多个分组的源，每个分组包括一个头域，所述源耦合到网络中;

包括解压缩器的接收器;

通过网络实体和接收器之间的网络耦合到网络中以及接收器中的网络实体，所述网络实体包括用于为从所述源发送的并且指向所述接收器的至少一些分组执行头域压缩的压缩器，所述网络实体包括用于计算指向所述接收器终端的分组中的抖动并且丢弃具有大于预定值的抖动的分组的抖动减少功能，

其中抖动计算为由源和接收器中包括的解压缩器之间的网络引起的全部抖动量。

40. 根据权利要求 39 的通信系统，其中通过计算压缩器之前的网络的抖动影响以及计算压缩器和解压缩器之间的网络的抖动影响来计算所述抖动。

41. 根据权利要求 39 的通信系统，其中压缩器和解压缩器之间的网络的所述抖动影响设置为抖动的上界值。

42. 根据权利要求 40 的通信系统，其中通过利用关于参考分组的信息计算当前分组的抖动影响来计算压缩器之前的网络的所述抖动。

43. 根据权利要求 40 的通信系统，其中所述计算在压缩器之前的网络的抖动影响包括:

利用关于所述当前分组以及预定数量的前面分组的每个的信息计算当前分组的抖动影响。

44. 根据权利要求 40 的通信系统，其中所述计算在压缩器之前的网络的抖动影响包括:

利用关于所述当前分组以及直到参考分组的每个前面分组的信息计算当前分组的抖动影响。

45. 根据权利要求 39 的通信系统，其中压缩头域计算为压缩值的 k 个最低有效位，其中 k 是一个整数；并且

其中解压缩器基于自从前一个分组到达以来流逝的时间以及前一个分组的解压缩的压缩值来计算压缩值的近似值。

压缩数据分组中头域的技术

技术领域

本发明涉及一种用于压缩数据分组中头域的方法和设备。更特别地，本发明涉及一种利用基于定时器和参考的方案来压缩数据分组中头域的方法和设备。

对于基于互联网协议（IP）的多媒体，实时传输协议（RTP）主要用于用户数据报协议（UDP/IP）之上。在 RFC 1889 中详细描述了 RTP。组合的 IP/UDP/RTP 头的大小对于 IPv4 至少是 40 字节，对于 IPv6 至少是 60 字节。在频谱效率是主要关注点的系统（如蜂窝网）中认为每个分组 40-60 字节的系统开销是很重的。因此，需要合适的 IP/UDP/RTP 头压缩机制。在 RFC2508 中描述了当前头压缩方案，其能够将 40/60 字节的 IP/UDP/RTP 头在点到点链路上压缩到 2 个或 4 个字节。已有的头压缩算法基于这样的观察事实，即在会话长度期间分组流中 IP 分组头的大多数域保持恒定。因此，可能通过在解压缩器中建立压缩状态（完整头信息）并且通过从压缩器到解压缩器传送最少量的头信息来压缩头信息。

RFC2508 基于大部分时间，从一个分组到另一个分组改变的 RTP 域，如 RTP 时间戳，可通过线性外推法预测的想法。基本上要发送的唯一信息是序号，用于错误和分组丢失检测（以及上下文 ID）。当发送器确定线性外推法不能用于当前分组时，就发送关于紧接着前面分组的第一个顺序差异信息。为初始化会话，发送完整的头。除此之外，当接收器确定有分组丢失时（通过序号增加超过 1 检测到）接收器明确地请求发送器发送完整的头以便使得能够重新同步。

但是，在 RFC2508 中定义的头压缩不很适合于某些环境（如蜂窝或无线环境），其中带宽非常珍贵并且错误很常见。在 RFC2508 头压缩方案中，假设 RTP 时间戳大部分时间都是线性增长模式。当头符合该模式时，基本上在压缩的头中仅需要一个短序号。当头不符合该模式时，在压缩头中发送当前头和前一个头的时间戳之间的差异。通过使用编码表可能实现进一步优化。这个方法有三个缺点。第一个是其对错误不健壮，因为前一个头的丢失会使当前头的解压缩无效。

第二个是 RTP 时间戳差异或跳跃会非常大，因此会溢出编码查找表。例如，如果介质是语音，则无声间隔可引起这样大的差异。第三个是结果编码的差异大小是变化的，使得更难预测和管理要分配的带宽。

因此，需要一种能够容纳域的值中（如 RTP 时间戳的值中）任意跳跃，产生更一致或固定的大小，并且对错误更健壮的头压缩方案。

发明概述

根据本发明的实施方案，提供了基于定时器的头解压缩技术。RTP 源生成头域，如 RTP 时间戳。时间戳通过网络发送到压缩器。在压缩器中，使用减少抖动功能（JRF）来确定接收的时间戳（头）的抖动是否过多。如果抖动过多，则丢弃分组。否则，压缩器基于 RTP 时间戳以及时间戳的初始值来计算压缩的头域（压缩的时间戳）。压缩的时间戳表示计算为源和解压缩器之间的网络在分组传输上的影响的抖动。计算的抖动是表示源和压缩器之间的网络在分组传输上的影响的网络抖动以及表示压缩器和解压缩器之间的网络在分组传输上的影响的无线电抖动的累积。应该指出，这里使用的术语“网络”是一个广义的术语，因此不应该排除，例如，无线电信网中的无线电链路。这样，包括压缩时间戳的 RTP 分组通过链路或网络发送到解压缩器。

解压缩器通过首先基于位于终端的定时器的当前值（也就是，基于流逝的时间）来计算时间戳的估计值或近似值来解压缩压缩的时间戳。然后基于分组头中提供的压缩时间戳来改进或改正时间戳的近似值。照这样，基于本地定时器和当前头中提供的压缩时间戳来重新生成当前分组（头）的时间戳。然后将分组和重新生成的时间戳提供给 RTP 端点处理。

本发明基于定时器的方案包括几个优点。这里使用的术语“基于定时器的方案”包括使用压缩的时间戳的基于定时器方案以及在这里公开的基于定时器和参考的方案。压缩的时间戳的大小（或其它头域）是固定的并且很小。而且，其大小不作为无声间隔的长度的函数而变化。在 RTP 源的定时器过程（生成时间戳）和解压缩器的定时器过程之间不需要同步。而且，这个技术对错误很健壮，因为在压缩头中的部分时间戳信息是自包含的并且只需要与解压缩器定时器值相结合来生成完整的 RTP 时间戳值。头的丢失或错误不会使后续的压缩头失效。

本发明的第二个实施例提供了一种头剥离方案，其中头（如包括 RTP 时间戳）在发送前从 RTP 分组中剥离或去掉。头剥离器以及头生成器通过类似电路连接（如电路或虚拟电路）或基本恒定比特速率的信道连接。在初始化之后，头剥离器从每个分组中剥离或去掉头（包括去掉时间戳和序号），并且然后将无头的分组发送到头重新生成器。为消除在头剥离器处的分组抖动，分组可按照头中的 RTP 时间戳（TS）按时间间隔发送。因此，在这个实施方案中，在 RTP 分组中没有明确地提供时间戳（也没有压缩的时间戳）。而是，基于头剥离器和重新生成器之间的基本固定比特率信道来隐含地为头重新生成器提供定时信息。可以以几种不同方式提供基本固定比特率信道。

在这第二个实施方案中，初始化发生之后（例如，为头重新生成器提供初始序号和时间戳），头重新生成器可以通过每 T 毫秒使本地时间戳计数器增加 TS_stride 来为后续分组重新生成时间戳，并且通过每个分组持续时间使本地 SN 计数器增加 1 来重新生成分组序号。因为在头剥离器和重新生成器之间提供基本固定的比特率信道，其中没有引入分组抖动，所以这些域可仅基于本地时钟或计数器重新生成。因此，在初始化之后，这些头域可仅参考本地时钟在头重新生成器中重新生成。

但是，会出现一个或多个基本的不连续事件（如分组大小改变或 TS_stride，时间戳中的非线性偏移等），如果不解决，很可能使仅依赖于本地定时器或时钟来进行域重新生成的头剥离方法失效。头串是一系列具有已知或线性可预测域的分组头。通过几个不连续事件的任何一个可导致从一个串到另一个的转变。当这发生时，头剥离器识别不连续事件并且将与事件相关的更新的头信息发送到头重新生成器，以便使时间戳和序号重新生成继续。当有切换时，也可使用一种提供更新的头信息的类似技术。

附图简述

结合附图的下列详细描述将使本发明更显而易见。

图 1 是说明根据本发明的示例实施方案的一个系统的框图；

图 2 是说明根据本发明的实施方案 RTP 分组的未压缩格式的图；

图 3 是说明根据本发明的示例实施方案未压缩的 RTP 头格式的图；

图 4 是说明根据本发明的示例实施方案压缩的 RTP 头格式的图；

图 5 是说明根据本发明的实施方案头压缩和解压缩的示例操作的图；

图 6 是说明根据本发明的另一个实施方案头压缩和解压缩的示例操作的图；

图 7 是说明根据本发明的实施方案切换的示例操作的图；

图 8 是说明根据本发明的示例实施方案的示例堆栈的框图；

图 9 是说明根据本发明的示例实施方案在消息中提供的信息的表；

图 10 是说明根据本发明的示例实施方案的切换过程的图；

图 11 是说明根据本发明的示例实施方案带内初始化过程的图；

图 12 是说明根据本发明的示例实施方案带外初始化过程的图；

图 13 是说明根据本发明的第一种方法计算网络抖动的步骤的图；

图 14 是说明根据提出作为本发明的选项 1 的第二种方法计算网络抖动的步骤的图；

图 15 是说明根据提出作为本发明的选项 2 的第三种方法计算网络抖动的步骤的图；

实现本发明的最佳模式

I. 利用压缩时间戳的基于定时器的方案

A. 结构

图 1 是说明根据本发明的示例实施方案的一个系统的框图。终端 102 连接到 IP 网络 108 上。终端 102 可以是运行 RTP/UDP/IP，并且提供 RTP 分组中分组语音采样用于在网络 110 上发送的个人计算机等。终端 102 包括识别这个终端（如，包括 IP 地址、端口号等）是 RTP 分组的源还是目的地的 RTP 端点 104。提供 IP 网络作为例子，但是，也可以替代使用其他类型的分组交换网等。应该指出这里使用的术语“网络”意指一个广义的术语，因此不排除，如无线电信网中的无线电链路。终端 102 也包括用于生成时间戳的本地定时器 103。

接入网基础结构（ANI）110 连接到 IP 网络 108。无线终端 130 通过射频（RF）链路 140 耦合到 ANI 110。无线终端 130 如这里所描述的，例如，依赖于其环境可以是无线压缩器或无线解压缩器。当分组的源或分组的目的地与无线终端 130 分离时尤其出现这种情况。RF

链路 140 包括上行链路 142（从终端 130 到 ANI 110）和下行链路 144（从 ANI 110 到终端 130）。ANI 110 将区域中的一个或多个无线（或射频）终端（包括终端 130）接口到 IP 网络 108，包括有线信号（从 IP 网络提供）和无线或 RF 信号（提供给终端 130 或从中提供）之间的转换。因此，ANI 110 使得从 IP 网 108 接收的 RTP 分组能够在 RF 链路 140 上发送到无线终端 130，并且使得来自终端 130 的 RTP 分组能够在 IP 网络 108 上发送到另一个终端，如终端 102。

根据本发明的实施方案，ANI 110 包括一个或多个 ANI 适配器（ANI_AD），如 ANI_AD 112 和 ANI_AD 114，其中每个优选地包括一个定时器。每个 ANI_AD 执行头压缩（在下行链路传输之前）和解压缩（在上行链路传输之后）。从 IP 网络 108 接收的 RTP 分组的头（或者一个或多个头域，如时间戳）在下行链路 142 上发送到终端 130 之前由 ANI_AD 112 压缩，并且从终端 130 接收的分组头在发送到 IP 网络 108 之前由 ANI_AD 112 解压缩。因此，每个 ANI_AD 被认为是压缩器/解压缩器 115。每个 ANI_AD 将位于区域内一个特定或不同范围的终端接口到 IP 网络 108。ANI_AD 112 包括用于实现基于定时器的解压缩技术的定时器 113。ANI_AD 112 还包括抖动减少功能（JRF）115，其运行来测量在网络 108 上接收的分组（或头）上的抖动并且丢弃具有过多抖动的任何分组/头。

额外的 ANI，如 ANI 120，提供用来将位于额外的区域内的其他终端接口到 IP 网络 108。ANI 120 类似地包括一个或多个 ANI_AD，如 ANI_AD 122（图 1）。每个 ANI_AD 包括一个定时器和一个 JRF。

终端 130 包括 RTP 端点 132，其是 RTP 分组的源和/或目的地（接收器）。终端 130 包括执行头压缩（对于要在上行链路 142 发送的分组）和解压缩（在下行链路 144 上接收的分组上）的终端适配器（term_AD）136。因此，终端适配器（term_AD）136 被认为是头压缩器/解压缩器 137，类似于 ANI_AD。

终端适配器（term_AD）136 还包括用于计算当前头的 RTP 时间戳的近似值（或估计值）的定时器 134（接收器定时器）。然后终端适配器（term_AD）136 使用 RTP 头中的附加信息来改进或改正时间戳的近似值。根据本发明的实施方案，基于 RTP 头中提供的压缩时间戳来改进或调整时间戳的近似值。照这样，可使用本地定时器和压缩

的时间戳来为每个 RTP 头重新生成正确的时间戳。可提供其他终端（如终端 150），每个包括其自己的 RTP 端点、终端适配器和定时器。

图 1 中所示的配置仅提供作为例子，并且本发明不限于此。更确切的，图 1 仅提供了一个例子，其中 RTP 数据在带宽非常宝贵并且错误并非不常见的数据链路或系统（如无线链路 140）上发送。本发明不限于无线链路，而是适合于广泛的各种链路（包括有线链路等）。

其中基于定时器的头压缩和解压缩方案的一个示例应用或系统是在蜂窝系统上传输 VoIP（或 IP 电话）分组。当 VoIP 应用于蜂窝系统时，因为无线或空中（RF）接口的有限带宽，最小化 IP/UDP/RTP 头的开销很重要。在这样的系统中，例如，ANI_AD 将 IP 网接口到运行 IP/UDP/RTP 并且具有用于接收无线或 RF 链路上的 RTP 分组的蜂窝或 RF 接口的计算机终端（如终端 130）上。这仅仅是本发明的压缩/解压缩技术的一个示例应用。

图 2 是说明根据本发明的实施方案的 RTP 分组的未压缩格式的图。如图 2 所示，未压缩的 RTP 分组包括 IP 头、UDP 头 212、RTP 头 214 以及可以是语音采样的有效载荷 216。

图 3 是说明根据本发明的一个示例实施方案的未压缩 RTP 头格式的图。如图 3 所示，未压缩的 RTP 头包括时间戳（TS）310、序号（S.N.）312 以及一些其他域 314。因 IP 网 108 的分组交换特性，RTP 分组会失序到达。在 RTP 接收器或 RTP 目的地（例如图 1，终端 130）使用序号 312 来按正确的顺序组装 RTP 语音采样。但是，RTP 分组中的序号不反映域中任何非线性的改变（例如，语音信号的无声间隔）。因此，提供时间戳（TS）310 来指示每个分组的相对定时。

如上所指，有每个 RTP 分组中 IP/UDP/RTP 头提供的 40-60 字节的头开销太大了的担心。特别地，对于在低速率或有限带宽链路（如链路 140）上运行的 RTP 分组来说 4 字节 RTP 时间戳特别难以负担。结果，需要一种机制来有效地压缩 RTP 头并且特别是压缩 RTP 头中的时间戳域。

在 RFC 2508 中描述的头压缩技术初始地发送一个完整（未压缩）的 RTP 分组，包括到 RTP 目的地/接收器的所有的域。在连接期间头的许多域是静态的，并且因此，不需要在初始分组发送和接收之后发送。对于大多数分组，从分组到分组只有序号和时间戳改变。根据 RFC

2508，在接收器通过在存储在接收器的那些域的以前值中加入（固定的）一阶差分来更新非静态域（例如，时间戳和序号）。例如，对于每个分组每个接收的 RTP 分组的序号自动加 1。非静态域中的额外跳跃或改变（也就是与一阶差分不同的）必须单独发送到接收器。不幸的是，在 RFC 2508 中，前面头的丢失将使接收器中的解压缩无效。而且，差异的大小是变化的，使得利用 RFC 2508 的压缩技术很难管理和预测带宽。

根据本发明的实施方案，提供了一种头压缩技术可用于更有效地压缩分组头的 RTP 时间戳（或其它域）。根据本发明的实施方案，压缩方案可适合 RTP 时间戳值中的任意跳跃，而生成一个固定大小的压缩的 RTP 头（或固定大小的 RTP 时间戳）。

图 4 是说明根据本发明的示例实施方案的压缩 RTP 头格式。如图 4 所示，压缩的 RTP 头由指示消息类型的消息类型 410、识别改变的域的位屏蔽 412、以及压缩时间戳域 414 组成。如果在分组头中提供压缩的时间戳，则消息类型 410 指示压缩的时间戳。根据本发明的实施方案，压缩的时间戳域 414 包括指示分组之间时间流逝的值的 k 个最低有效位（lsbs）。根据本发明的实施方案，压缩的时间戳域 414 提供源计数器值（或计数器差值）的一部分（也就是 k 个最低有效位）。源计数器用于为每个 RTP 分组头生成时间戳。可选域 416 可用于为在位屏蔽 412 中识别的那些域提供更新的或改变的域。

B.时间戳压缩和解压缩的整体操作

根据本发明的实施方案，将简要描述 RTP 时间戳的压缩和解压缩。根据实施方案，在 RTP 端点（如终端 102 的 RTP 端点 104）生成 RTP 分组并且寻址到另一个 RTP 端点。在这个例子中，RTP 端点 104 是要发送到终端 130 的 RTP 端点 132(目的地)的一个或多个 RTP 分组的源。RTP 分组头包括基于时钟在 RTP 源（例如终端 102）生成的时间戳。

RTP 分组在 IP 网络 108 上路由到 ANI 110 的 ANI_AD 112。ANI_AD 112 压缩 RTP 分组的头（多个头）中的一个或多个域。特别地，ANI_AD 将 RTP 时间戳 310（图 3）压缩到压缩的时间戳 414（图 4）中。头中的其它域通过去掉它们或使用一些其它技术压缩。然后包括压缩时间戳 414 的 RTP 分组在 RF 链路 140 的下行链路 144 上发

送到终端 130。

一接收到带压缩头（也就是压缩的时间戳 414）的 RTP 分组，终端 130 的终端适配器（term_AD）136 就解压缩时间戳值。终端适配器 136 通过首先基于定时器 134 的当前值计算时间戳的估计值或近似值来解压缩压缩的时间戳 414。然后基于分组头中提供的压缩的时间戳 414 来改进或改正时间戳的近似值。照这样，基于本地定时器（定时器 134）以及当前头中提供的压缩时间戳来重新生成当前分组（头）的时间戳。分组头的其它域（如序号）也重新生成。然后分组和重新生成的时间戳提供给 RTP 端点 132 来处理。然后 RTP 端点 132 按适当的顺序（如由序号确定的）以及由重新生成的时间戳规定的适当的定时（例如，说明任何无声间隔）重放语音采样。

ANI_AD 112 还在 RF 链路 140 上接收压缩头（包括压缩时间戳），并且利用上述基于定时器的解压缩技术解压缩时间戳。因此，ANI_AD 112 典型地包括定时器以便使 ANI_AD 能够解压缩上述压缩的时间戳。类似的，终端 130 的 term_AD 136 还在 RTP 分组在 RF 链路 140 上发送到 ANI 110 之前压缩 RTP 分组的时间戳。为简化本发明的示例实施方案的解释，主要描述将针对下行链路 144。根据本发明的实施方案，RTP 分组在两个方向上发送（上行链路 142 和下行链路 144）。因此，ANI 110 的 ANI_AD 112 和终端 130 的 term_AD 作为时间戳压缩器（对于在 RF 链路上头/分组的发送）和解压缩器（在 RF 链路 140 上接收的压缩头的接收之后）操作。

C. 时间戳压缩和解压缩的示例实施方案

将简要描述时间戳压缩和解压缩的示例实施方案。假设 RTP 分组中的数据是语音数据。定义下列变量和公式仅用于辅助解释本发明的一些特性，但是本发明不限于此。而且，本发明不限于使用相同或类似类型变量的系统，并且不限于执行下述特定计算的系统。变量和计算仅提供作为本发明的示例实施方案。

T-RTP 语音采样之间的时间间隔。（如果每个 RTP 分组中提供一个语音采样，则 T 也是 RTP 分组头之间的间隔。）

TS-时间戳

TS_stride-每 T 毫秒 RTP 时间戳增加 TS_stride。换句话说，对于每个新的 RTP 分组，RTP 时间戳增加 TS_stride。TS_stride 是一

个依赖于语音编解码器的常量（例如，100）。TS_stride 提供给接收器（终端 130）和 ANI_AD 112。

TS0-在 RTP 接收器接收的会话的第一个头的 RTP 时间戳。会话的第一个头被认为是同步头，因为其用于同步。TS0 是在会话开始提供给压缩器（例如，ANI_AD 112）和解压缩器（如 term_AD 136）的 RTP 时间戳的初始值（用于同步）。根据实施方案，通过接收带有未压缩头的 RTP 分组（包括提供 TS0 的未压缩时间戳）初始化或同步 ANI_AD 和 term_AD。根据本发明的实施方案，基于定时器的解压缩技术要求在压缩头可以被正确解压缩（也就是因此解压缩器可以正确地重新生成时间戳）之前为时间戳压缩器（也就是 ANI_AD 112）和解压缩器（也就是 term_AD 136）提供初始时间戳 TS0（例如通过未压缩的初始或同步头）。

分组头 m 的 RTP 时间戳（在时间 m*T 毫秒时生成）=TS0+TS_stride*m。这假设对于每个语音采样有一个头。如下述示例所示，对于每个分组头多个语音采样（例如 3 个语音采样）的情况可以扩展这个公式。

m-指示已经发送的语音采样的数量的整数。在会话开始时 m 被重新设置或清除为 0。M 正比于（或指示）会话开始以来流逝的时间量。每 T 毫秒 m 增加 1。

TS_current=TS0+m_current*TS_stride; 当前分组头的当前时间戳。

接收器定时器-在 RTP 接收器（或 RTP 目的地）处的定时器，如终端 130 的定时器 134。本地接收器定时器典型地自由运行并且在会话开始时不重置。而是，当先前一个分组头接收时通过从接收器定时器值中减去当前头的定时器值可以获得在 RTP 接收器处接收两个分组头之间流逝的时间。通过允许接收器定时器自由运行，一个接收器定时器可由许多流或会话共享。替代地，接收器定时器可在每个会话开始时重置。在会话开始时（也就是，一接收到初始化头）重置或清除接收器定时器需要每个会话或流有一个专用接收器定时器（定时器过程）。在初始化头中会话的第一个未压缩的时间戳（TS0）可提供给 ANI_AD 和 term_AD。提供第一个头来初始化压缩器（ANI_AD 112）和解压缩器（term_AD 136）。然后每 T 毫秒接收器定时器增加 1。

ANI_AD 112（压缩器）使用 TS0 值来压缩后续 RTP 分组头的时间戳。term_AD 136（解压缩器）使用 TS0 值来解压缩压缩的时间戳值（例如，来重新生成后续接收的 RTP 头中的时间戳）。

current_timer-当接收当前头时在 RTP 接收器（例如终端 130）中的定时器值

last_timer-当接收最后一个头时在接收器处的时间值。（current_timer 存储为 last_timer 用于时间戳的下一个头的计算）。

m_last-最后接收的头的 m 值；m 指示从初始化头以来流逝的语音帧的数量。

为压缩当前分组的时间戳，ANI_AD 112 计算 m 的当前值如下：m_current=(TS_current-TS0)/TS_stride。因此，ANI_AD 从当前时间戳中减去时间戳的初始值（在会话开始时）。这个差值除以时间戳跨度值（TS_stride）。但是，在有些实施方案中，不必要实际执行除法操作。可使用其它技术来恰当地生成 m_current，而不需要执行需要高处理器系统开销的除法操作。

然后提供 m_current 的 K 个最低有效位作为压缩的时间戳 414。然后包括压缩的时间戳 414 的 RTP 分组在 RF 链路 140 上发送到 RTP 目的地或接收器（例如终端 130）。

在 RTP 接收器（例如，终端 130），终端适配器（Term_AD）136 解压缩压缩的时间戳 414。前一个头的 current_timer 值首先存储为 last_timer。然后，当当前头到达时，term_AD 136 读取接收器定时器 134 的值并且将其作为 current_timer 存储在存储器中。接着，计算 timer_diff 为：timer_diff=current_timer-last_timer。ANI_AD 通过找到整数 d 计算 m_current 的确切值，其中：

（$-L/2<d\leq L/2$，其中 $L=2^k$）因此：　（等式 1）

（d+m_last+timer_diff）的 k 个最低有效位=压缩时间戳 414（对于当前头）。

（等式 2）

如指出的，接收的压缩时间戳也是 k 位。一旦利用等式 1 和 2 计算出 d，然后 TS_current 就可计算为：

TS_current=TS0+（d+m_last+timer_diff）*TS_stride。

（等式 3）

在等式 3 中，m_current 的实际或正确值显示在圆括号中为（d+m_last+timer_diff）。m_last+timer 是 m_current 的近似值，而 d 是 m_current 的近似值和 m_current 的正确值之间的差值。而且，TS0+（m_last+timer_diff）*TS_stride 是当前时间戳值的近似值，并且 d*TS_stride 是近似的当前时间戳值和当前时间戳的实际（或正确）值之间的差值。

因此，可以看出 RTP 接收器首先基于当前头和（正确地解压缩的）前一个头的接收之间流逝的时间来计算当前时间戳的近似值（或估计值）为近似的当前时间戳= TS0+（m_last+timer_diff）*TS_stride。然后由 d*TS_stride 的数量来调整或改正近似的当前时间戳以便计算正确的当前时间戳值（TS_current）。

TS_current 计算之后，当前 RTP 分组（包括其重新生成的或解压缩的时间戳，TS_current）提供给 RTP 端点 132。这个压缩和解压缩的过程对 RTP 端点是透明的。

图 5 是说明根据本发明的实施方案头压缩和解压缩的示例操作的图。这个例子应用上述的一些特定公式来说明本发明的一些特性。在这个示例实施方案中，假设在 RTP 源 502 和在 RTP 接收器 504 处的定时器具有相同的频率但是典型地不同步。在 RTP 源的定时器（例如，每 T 毫秒增加 1）用于生成时间戳，而在 RTP 接收器的定时器（例如定时器 134）用于重新生成或解压缩 RTP 时间戳。

参见图 5，在会话开始时，在 RTP 源生成初始化头 508，包括初始时间戳值（TS0）。初始化头 508 发送到 ANI 并且然后转发到 RTP 接收器 504（例如终端 130）。初始化头中的时间戳没有压缩。一接收到初始化值，初始时间戳值（TS0）就和 TS_stride 一起存储在 ANI_AD 的存储器中。根据一个实施方案，两个初始化头发送到 ANI_AD。然后 ANI_AD 可计算 TS_stride 为第二个时间戳-第一个时间戳。Term_AD 可类似地计算 TS_stride 或接收分组中的值。

类似地，当在 RTP 接收器（终端 130）接收到初始化头 508 时，初始时间戳（TS0）和 TS_stride 一起存储在存储器中。而且，一接收 510 到初始化头 508（图 5），m_current 就清除或重置为零（0），并且然后读取接收器定时器并存储为 initial_receiver_timer 516。代替在会话开始时读取定时器，接收器定时器可重置或清除。在这个例子

中，为简单起见在会话开始时读取的接收器定时器值恰好为零（0）。因此，因为在会话开始时自由运行的定时器读取为零，所以图 5 所示的例子可应用于两个实施方案（只读取接收器定时器，或将其重置为零）。同样，清除 m_current 没有必要，但是可以替代地记录 m_current 的值。此后每 T 毫秒接收器定时器增加（例如，1）（与在 RTP 源 502 处用于生成时间戳的定时器相同的频率）。初始化头 508 在固定延迟（整块（bulk）延迟 512）和变化的延迟（累积抖动 514）之后到达 RTP 接收器 502。

接着，RTP 源 502 生成下一个 RTP 分组（初始化头之后的会话的第一个 RTP 分组）。这个 RTP 分组在初始化头生成之后 3*T 毫秒生成，并且因此典型地包括例如三个（3）语音采样。其它数字也可能。因此，这个分组的头的时间戳是：TS（1）=TS0+3*TS_stride，如图 5 所示。TS（1）指在初始化后 3T 毫秒之后生成的时间戳。在这个例子中，假设 TS_stride 是例如 100。TS0 假设为例如 0。因此，TS（1）=300。

这个分组的时间戳值 TS（1）在 ANI_AD 接收并且基于 TS（1）（时间戳值）、TS0（初始化时间戳值）和 TS_stride（每 T 毫秒时间戳增加的量）来压缩。根据一个示例实施方案，压缩的时间戳可以计算为 m_current 的 k 个最低有效位。ANI_AD 112 计算 m 的当前值为：m_current=(TS_current-TS0)/TS_stride。在这个例子中，假设 TS_stride 为例如 100。在这个例子中，m_current 计算为：m_current=（300-0）/100=3。在这个例子中 k 为二（2）。因此，m_current（二进制为 11）的两个最低有效位提供作为图 5 的这个分组的压缩时间戳 414（CTS1）。

压缩时间戳（CTS1）到达 RTP 接收器 502 并且在 RTP 接收器的 term_AD 136 为当前分组重新生成或解压缩时间戳 TS（1）。current_timer 的值（0）存储为 last_timer 并且 m_current 存储为 m_last。m_current 以前在会话开始时（也就是接收到同步头时）设置为零。接收器定时器值（在这种情况下为 3）读取并且存储为 current_timer。然后 Timer_diff 计算为 current_timer-last_timer，为 3-0=3。Timer_diff+m_last 是 m_current 的近似值。

接着 term_AD 136 利用等式（1）和（2）计算 m_current 的确切

或正确值。利用等式（2），（d+m_last+timer_diff）的两个最低有效位=CTS1（当前头的压缩时间戳）。在这种情况下，m_last 是零（0），timer_diff 是三（3）并且 CTS1 是三（3）。因此，（d+0+3）的两个最低有效位=3。因此，d 等于零。

利用等式(3)，然后这个分组的解压缩时间戳计算为 TS(1)=TS0+（d+m_last+timer_diff）*TS_stride。因此，作为结果，TS（1）=0+（0+0+3）*100=300。然后这个分组的解压缩时间戳，TS（1）=300，与 RTP 数据以及其它解压缩的头域一起提供给 RTP 源的 RTP 端点 132。m_current 的正确或实际值是（d+m_last+timer_diff）。因此，对于这个分组，可以看出 m_current 的近似值与 m_current 的正确值相同（但是在一般情况下不是这样）。然后 m_current 更新为 3。

下一个分组和时间戳在 RTP 源生成，包括时间戳 TS（2）=0+6*100=600。在 ANI_AD，TS（2）=600 压缩到压缩的时间戳中作为（600-0）/100=6 的 2 个最低有效位。在这种情况下，6 以二进制表示为 110。因此，110 的两个最低有效位为 10。因此，以二进制表示 CTS2=10。

然后因整块延迟和累积抖动，在接收器定时器到达值 7 之后，在 term_AD 136 接收到这个分组的压缩时间戳（CTS2）。current_timer（3）的值存储为 last_timer 并且 m_current(3)存储为 m_last。当前接收器定时器值（在这种情况下为 7）读取并且存储为 current_timer。然后 Timer_diff 计算为 current_timer-last_timer，即 7-3=4。Timer_diff+m_last 是 m_current 的近似值，为 7。

接着，term_AD 136 利用等式（1）和（2）计算 m_current 的确切值或正确值。利用等式（2），（d+m_last+timer_diff）的两个最低有效位=CTS2（当前头的压缩时间戳）。在这种情况下，m_last 为 3，timer_diff 为 4 并且 CTS2 为 10（二进制，十进制为 3）。等式 2 为 d 解答如下：2 lsbs（d+3+4）=2。7 以二进制表示为 111。因此，d=-1。d 是 m_current 的近似值和 m_current 的实际值之间的差值。将 d 插入等式(3)，这个分组的时间戳计算为 TS(2)=0+(-1+3+4)*100=600。因此 RTP 接收器的 term_AD 136 基于本地定时器和压缩的时间戳正确地重新生成了（例如解压缩）RTP 时间戳。

应该指出，不象前面的技术，在一个或多个分组没有到达 RTP

接收器的情况下没有必要重新发送初始化头。换句话说，RTP 源和接收器之间的同步仅在会话或连接开始时需要一次。这是因为，在 RTP 接收器基于 m_last 和 timer_diff 来计算当前时间戳。Timer_diff 计算为 current_timer-last_timer。因此，m_last 和 last_timer 的值对应于最后分组，而不管哪个分组最后接收（例如，不管在“最后”分组发送之后的分组被错误地丢弃或丢失）。结果，根据本发明的基于定时器的压缩方案对错误健壮并且因为在检测到错误（例如一个或多个分组丢弃或丢失）的情况下不需要发送新的同步分组（例如包括所有头的完整的未压缩值），减少了带宽需要。

在正常操作中，m_current 的近似值和确切值之间的差异由下列引起:

a) RTP 时间戳和接收器的源之间的累积抖动；实际延迟=整块延迟+累积抖动，其中整块延迟是固定的并且累积抖动从一个头到下一个而变化，并且 0≤累积抖动≤最大累积抖动；以及

b) 依赖于定时器实现，定时器过程和解压缩过程之间的可能的异步。因为异步，定时器值（current_timer）中可能出现加或减 1（+或-1）的错误。

图 6 是说明根据本发明的另一个实施方案的头压缩和解压缩的示例操作的图。与图 5 类似，图 6 是说明抖动和定时器异步的影响的图。在图 5 中，接收器定时器仅在会话开始时重置或清除。（因为接收器定时器允许继续运行，因此这一点是不必要的。）但是，在图 6 所示的示例实施方案中，对于每个分组接收器定时器被重置或清除为零（0）。因此，当接收到压缩的分组头时，读取定时器值，其指示上述的 timer_diff 值（因为定时器指示自从上一个分组头之后流逝的时间）。有许多不同的方法来实现本发明。重要的是应该测量指示最后成功解压缩时间戳和当前时间戳之间流逝的时间的定时器差值（图 5 中描述的 timer_diff）（如由本地接收器定时器测量）。

参见图 6，头 n 用时间戳=TS0+3*TS_stride 生成。头 n 的这个时间戳被压缩并且发送到 RTP 接收器，并且解压缩。然后在接收器的定时器重置。接下来的头，（n+1）、（n+2）以及 （n+3）生成并且发送，但是仅接收到头（n+3）（也就是头 n+1 和 n+2 丢失）。为简单起见，头(n+2)和(n+3)在图 6 中未示出。头（n+1）在图 6 中显示为头 m+n。

头(m+n)与时间戳 TS=TS0+6*TS_stride 一起生成并且发送。头(m+n)的这个时间戳被压缩然后发送到 RTP 接收器。定时器值是 4（指示 timer_diff）。这个值用于为头(m+n)的时间戳解压缩。因此，图 6 的例子非常类似于图 5 所示的例子，除了在图 6 中定时器在接收每个头后重置。

不管使用的哪种技术（图 5 或图 6），可使用有效的基于定时器的压缩方案。但是，累积抖动过多，将不可能基于压缩的时间戳来重新生成正确的时间戳。在许多情况下，k 必须满足下列条件以便使得图 5 和/或图 6 所示的基于定时器的压缩方案正确工作：

[条件 1]（最大整数抖动+2）$<2^k$，

其中最大整数抖动（MIJ）是以 T 毫秒为单位表示的最大累积抖动，四舍五入为下一个更高的整数。例如，如果 T=20 毫秒，15 毫秒的最大累积抖动导致 MIJ=1。2 加到 MIJ 中以便考虑定时器异步引起的可能的错误。

因为会话的实时需求，在正常操作中累积抖动仅是几次 T 毫秒。因此，在这种情况下，k 值等于 4 足够了，因为在 RTP 接收器可以改正多达 16 个语音采样（也就是 16*T 毫秒）差异。异常或错误的情况会导致抖动超过正常值。压缩器的上行比特流中可以加入抖动减少实体以便确保抖动，如压缩器所看到的，保持在可接受的限度之内。

图 5 和/或 6 中所示的时间戳压缩方案的优点包括：

a) 时间戳的大小是固定的并且很小。压缩头典型的由指示消息类型的消息类型（k1 比特）、指示哪个域在改变的位屏蔽，以及包含 m_current 的 k 个最低有效位的域（k 比特）组成。假设如在 RFC 2508 中使用相同的 4 位 MSTI 位屏蔽，并且 k1=4，当仅 RTP TS 改变时（这种情况是最常见的）压缩头的大小是 1.5 字节。而且，这个大小不作为无声间隔长度的函数而改变。

b) 如图 6 中所示，接收器定时器与 RTP 源定时器在相同的频率运行（用于生成原始时间戳）；不需要源定时器和接收器定时器之间的相位同步（因为是测量的接收器定时器的时间流逝用于重新生成时间戳）。

c) 在接收器，在定时器过程和解压缩过程之间不需要同步。例如定时器过程每 T 毫秒使定时器增加 1，而当接收到新的头时唤醒解

压缩过程执行解压缩。但是，不要求定时器增加的点与头接收的点相一致或同步（参见图 6）。

d) 因为压缩头中的部分 RTP TS 信息是自包含的并且仅需要与接收器的定时器相结合来生成完整的 RTP TS 值，所以对错误有健壮性。头的丢失或破坏不会使后续的压缩头无效。

e) 压缩器不需要为 RTP TS 压缩/解压缩的目的维护或存储存储器或值。

D. 切换

根据实施方案，每个 ANI_AD 被分配到一个特定区域（例如，位于特定区域中的接口终端）。终端（如终端 130）可以从一个区域移动到另一个区域。当终端从一个区域移动到另一个区域时，终端必须从一个 ANI_AD 切换或转换到另一个 ANI_AD。

需要考虑的切换的一种情况是 ANI_AD 间切换，其中从旧的 ANI_AD 转换到新的 ANI_AD 会导致中断。问题是在切换期间怎样维持信息的连续性以便在切换后，在 term_AD 136 和新的 ANI_AD 的压缩/解压缩没有中断的继续。

1、 下行链路

在作为终端（例如，图 1 的终端 130）的接收器端没有不连续性。压缩器的角色是从一个 ANI_AD 传送到另一个。在切换之后，头在新的路径上通过新的 ANI_AD 代替旧的 ANI_AD 路由。除此之外，依赖于系统设计，在切换期间可能需要或不需要重新路由转变中的分组。转变中的分组是那些由源生成的但是在切换时还没有到达接收端的分组。重新路由试图将转变中的分组发送到终端。

为执行切换，旧的 ANI_AD 必须将会话的时间戳的初始值（TS0）以及 TS_stride 传送到新的 ANI_AD。这两个值使得新的 ANI_AD 能够继续压缩从 RTP 源（例如终端 102）接收到的新的时间戳（在新的分组头中）。使 Current_header 成为 term_AD 在切换之后解压缩的第一个头，并且其 TS_current 是其 RTP 时间戳。只要满足下列条件，term_AD 可解压缩 TS_current:

[条件 2]（下行链路瞬时整数抖动+2）$<2^k$,

其中下行链路瞬时整数抖动（DTIJ）是以 T 毫秒为单位表示的 current_header 的下行链路瞬时抖动，四舍五入到下一个更高的整数。

下行链路瞬时抖动定义为=Current_header 的全部延迟-旧路径上的整块延迟。如果 Current_header 不是重路由转换中分组的头，则 Current_header 的全部延迟也是新路径上的整块延迟+新路径上 Current_header 的累积抖动。因此，下行链路瞬时抖动=新路径上的整块延迟-旧路径上的整块延迟+Current_header 的累积抖动。

如果 Current_header 是重路由转换中分组的头，则 Current_header 的全部延迟=由路由和重新路由引起的全部延迟。实际上，系统应该优选地设计为将下行链路瞬时抖动保持为与稳定状态（也就是没有切换）的累积抖动相同的值范围。因此，基于这些假设（并不总是适用），如果条件 1（上面指出的）满足，则对于下行链路没有特定的切换相关的问题。

2、 上行链路

在这个上行链路描述中，终端（例如终端 130）的 term_AD 136 压缩时间戳并且将其在 RF 链路 140 上发送到本地或对应的 ANI_AD。在这种情况下 RTP 源是终端 130。即使 RTP 源（终端 130）改变物理位置（要求在 ANI_AD 切换），接收器的（解压缩器的）角色从一个 ANI_AD 转换到另一个。RTP 源保持固定在终端（如图 1，终端 130）上。

图 7 是说明根据本发明的实施方案切换的示例操作的图。为最小化空中接口的系统开销，有些信息需要从旧的 ANI_AD 710 传送到新的 ANI_AD 712 用于切换。该信息是在旧 ANI_AD 处的定时器值。旧的 ANI_AD 710 读取（或获得快照）旧 ANI_AD 处定时器的当前值（T_u）并且将其与 TS0、TS_stride 以及 m_current、块 714（图 7）一起发送到新的 ANI_AD。新的 ANI_AD 恢复从（T_u）开始增加其定时器。使 T_transfer 715（图 7）成为传送定时器的时间。而且，旧的和新的 ANI_AD 处的定时器过程有最多 T 毫秒的相位差异。使 Current_header 成为切换之后由新的 ANI_AD 解压缩的第一个头，并且 TS_current 成为其 RTP 时间戳。只要满足下列条件，新的 ANI_AD 就可以解压缩 TS_current:

[条件 3]（上行链路瞬时整数抖动+2+1）$<2^k$,

其中上行链路瞬时整数抖动（UTIJ）是以 T 毫秒为单位表示的上行链路抖动，四舍五入到下一个更高的整数。上行链路瞬时抖动定

义为=Current_header 的全部延迟-旧路径上的整块延迟+T_transfer。因为 Current_header 的全部延迟=新路径上的整块延迟+Current_header 的累积抖动，所以上行链路瞬时抖动=新路径上的整块延迟-旧路径上的整块延迟+ Current_header 的累积抖动+T_transfer。与下行链路相比，考虑到旧 ANI_AD 定时器和新 ANI_AD 定时器之间的相位差值而要增加 1。

明确的，图 7 还说明了上行链路瞬时抖动，其包括整块延迟差值以及 T_transfer。在这个例子中，旧的 ANI_AD 决定在定时器增加定时器之前准备切换。因此，其向 ANI_AD 712 发送 T_u=0。T_transfer 大约是 T 毫秒。在新的 ANI_AD 712，因定时器过程的异步，在定时器获得增加前几乎有 T 毫秒流逝。对头（n+m）在新路径上还有累积抖动。结果，当接收到头（n+m）时读取的定时器值是 2，而实际值应该是 4。因此，有-2 的偏差。只要满足条件 3，偏差可以消除并且 RTP 时间戳可以正确地被解压缩。

根据一个实施方案，T_u 在连接旧的和新的 ANI_AD 的高速信令网上发送。因此，时间 T_transfer 应该至多只是很小的 T 毫秒。但是，必须考虑 T_u 的传送不成功，或不够及时的情况。在那些情况下，新的 ANI_AD 将通知 term_AD 发送完整（未压缩）的 RTP 时间戳直到接收到确认。

E、抖动减少

根据本发明的实施方案，在满足下列条件时可以预测使用压缩的时间戳和本地接收器定时器的基于定时器的压缩方案。

[条件 1]（最大整数抖动+2）$<2^k$，

[条件 2]（下行链路瞬时整数抖动+2）$<2^k$，

[条件 3]（上行链路瞬时整数抖动+2+1）$<2^k$。

因会话实时的要求，可以合理的期望上述各种抖动与正常操作中的几个 T 毫秒的相似。因此，k 的一个很小的值，例如 4 通常足够使任何偏差或错误得到纠正。但是，从 RTP 源到接收器的路径上存在异常情况（失败等）或其它情况，其中抖动变得过多（其中不能基于压缩的时间戳和本地接收器定时器来生成正确的时间戳）。为解决这些情况，可提供抖动减少功能（JRF）115（图 1）作为压缩器的前端，来过滤出（或丢弃）具有过多抖动的分组（例如，上述条件 1，2 或 3

中的任何一个不满足时）。

为筛选出或识别具有过多 MIJ 的分组，抖动减少功能（JRF）计算在网络 108 上接收的每个分组的抖动。如果测量的分组抖动大于 2^k –2，则认为是过度抖动并且丢弃分组。否则，压缩（如上所述）头（或头域）并且然后发送到接收器终端（例如终端 130）。

JRF 计算当前分组的抖动如下：抖动=（TS2-TS1-JRF timer_diff）的绝对值，其中 TS2 是当前分组的时间戳，TS1 是前一个分组的时间戳，并且 JRF timer_diff 是当前分组和前一个分组之间 JRF_timer 的差值（流逝的时间）。这个抖动值可与 2^k –2 相比。如果抖动大于 2^k – 2，则丢弃分组。否则，在 ANI_AD 压缩分组头并且带有压缩头的分组发送到 RTP 接收器。

这个抖动减少功能（JRF）115 是限制由接收器终端接收的分组上的抖动的一种有效技术（因为在 RF 链路上引入的抖动被认为是可以忽略的）。而且，JRF 运行以便更有效地利用 RF 链路 140 上的可用带宽。在没有 JRF 115 的情况下，具有大于 2^k –2 的抖动的一个或多个分组会通过 RF 链路 140 发送到 RTP 接收器。但是，在接收器，如果抖动过多（也就是不满足条件 1），则不能生成正确的时间戳值，导致接收器丢弃分组。因此，JRF 仅运行来过滤出具有过多抖动在接收器处无论如何将被丢弃的那些分组（避免链路 140 上有价值带宽的浪费）。

H、头剥离方案

本发明的第二个实施方案提供基于定时器的头剥离方案，其中头或者一个或多个头域（例如，包括 RTP 时间戳）在经过低带宽链路（例如经过图 1 的 RF 链路 140）发送之前从 RTP 分组中剥离。在这种情况下，在 RTP 分组中没有明确地提供时间戳。而是，定时信息隐含地提供给头重新生成器用于基于头剥离器（例如，存在在 ANI_AD 中）和头重新生成器（例如，在终端 130 中存在）之间的基本固定比特率信道或类似电路的连接来增加本地定时器。

A、 头剥离概述

头剥离基于这样的思想，即对于某些应用或业务，不需要传输包含在 IP/UDP/RTP 头中的所有信息，或者因为其不变化，或者因为其对应用/业务不必要。基本语音是一个典型的例子。为了提供一种等价

于已有的蜂窝语音业务的业务（例如，通过图 1 的 RF 链路 140），必须的唯一的变化头信息是 RTP 时间戳（TS）。还要求保持 RTP 序号（SN）的透明。这里透明（对于 SN）指剥离/重新生成之后的 SN 等于原始的 SN。头剥离依赖于由类似电路的连接或基本固定比特率的信道（其中没有引入分组抖动）提供的暗示的定时信息以便仅基于本地定时器或计数器来重新生成 RTP 时间戳。这消除了明确地发送时间戳（或者甚至是发送压缩的时间戳）的需要。为获得 SN 的透明，压缩的 SN 可与来自类似电路的信道或连接的定时信息结合使用。类似电路的连接优选地提供具有基本固定比特率的信道。当没有语音采样（例如无声间隔）时，信道可能会或不会分配给其它业务和/或用户。

这种头剥离方案的优点包括:

a) 最低的头系统开销，任何其它方案无法相比（甚至小于上述图 1-6 中的压缩头技术）。

b) 对错误的健壮性，因为来自类似电路的传输或基本固定比特率信道的定时信息固有地不受错误影响

c) 如果需要时在通话期间转换到头压缩的可能性（例如，图 1-6 的技术）。如果通话变成多媒体时，因为非语音介质添加到语音中，所以这会非常有用。而且，注意头剥离不托管或排除统计复用，其如果实施，可在较低层出现。

图 8 是说明根据本发明的示例实施方案的示例堆栈的框图。显示了头剥离器堆栈 802 和头重新生成器堆栈 830。作为例子，头剥离器堆栈 802 说明用于从分组上剥离一个或多个头域的一些组件，而头重新生成器堆栈 830 说明用于重新生成分组头的一些组件。头剥离器堆栈 802 可以在例如一种类型的 ANI 适配器（例如，图 1，ANI_AD 112）中提供，而头重新生成器堆栈 830 可存在于例如，一种类型的终端适配器（例如，图 1，term_AD 136）中。

参见图 8，头剥离器堆栈 802 包括 RTP 和 UDP 层 804、IP 层 806。RTP/UDP/IP 层生成 RTP 分组 808（其包括 RTP 头中的时间戳）。接着，在头剥离器堆栈 802，RTP 分组 808 提供给头剥离器（HS）810 用于剥离或去除一个或多个头或头域。提供层 L1 和 L2 812，其中例如，L2 是数据链路层并且 L1 是物理层。按需要可提供其它层。类似地，头重新生成器堆栈 830 包括相应的 L1 层和 L2 层 820，重新生成

头(包括RTP时间戳)以便提供完整的RTP分组824(包括RTP/UDP/IP头)的头重新生成器(HR)822。分组824提供给IP层826并且然后提供给UDP和RTP层828。头剥离器堆栈802和头重新生成器堆栈830的层L1和L2通过链路815或空中接口(如RF链路140)或通过网络进行通信。例如,VoIP分组在链路815(例如,无线链路或网络)上发送之前通过头剥离器810传送。在接收端(在头重新生成器堆栈830),头重新生成器822在递交到接收者之前重新生成头。层L1/L2提供类似电路的连接,也就是提供头剥离器810和头重新生成器822之间的基本固定比特率信道。除此之外,为了最大效率,除了优化的信道编码和交织之外,层L1还执行类似不均匀比特保护的话音有效负荷优化。注意头剥离的概念不管有效负荷优化是否进行都是适用的。

在操作中,头剥离器(HS)810消除了进入RTP分组中的抖动,并且根据头中的RTP时间戳(TS)将其回放。这里,消除抖动意味着根据时间戳在类似电路的连接或基本固定比特率信道上调度语音采样的发送。换句话说,分组在去除或剥离头之后,在类似电路的信道或基本固定的比特率信道上在基于分组中其时间戳的时间发送。利用例如抖动减少功能(JRF 115,图1)丢弃具有过多抖动的分组。头重新生成器(HR)822重新构建IP/UDP/RTP域,其可分为以下几类:

a) 静态:在会话期间值不改变,例如IP地址。

b) 非静态:从一个分组到另一个分组值原则上改变,但是实际上,对于语音,在头剥离期间需要保留的唯一的非静态域是RTP时间戳(TS)。RTP序号(SN)也保留。静态域可以转换一次并且对于所有的在会话开始时作为初始化阶段完整头的一部分。可使用可靠的发送机制(例如,利用来自RTP接收器的确认或肯定应答来确认初始化信息的接收)。将简单讨论时间戳和序号。

1、 RTP时间戳(TS)

在语音的情况下,RTP时间戳(TS)作为在RTP源的壁钟(也就是源定时器)的函数线性增加。如果连续语音采样之间的时间间隔是T毫秒,则头n的RTP时间戳(在时间n*T毫秒生成)=头0的RTP时间戳(在时间0生成)+TS_stride*n,其中TS_stride和T是依赖于语音编解码器的常数。如果每个话音(语音)采样一个分组则

也是这样。更一般地，RTP 时间戳（TS）是 TS0+m* TS_stride 的形式，其中 TS0< TS_stride 并且 m 是整数。在抖动消除之后在头剥离器（HS）可以看到相同的行为。

在会话或连接开始时，执行初始化过程来初始化 RTP 接收器（也就是，初始化头重新生成器）。在初始化阶段，头剥离器一直发送初始化信息（Init_info）直到从接收器接收到肯定应答。Init_info（n）基本上由完整的 IP/UDP/RTP 头 n（包括初始时间戳和序号）组成。RTP 序号用于识别这个特定的初始化头，因为后续的初始化头包括更大的序号（假设第一个初始化头没有被确认）。

在头重新生成器（HR）822，当正确地接收到 Init_info（n）时，HR 822 发送 ack(n)。一旦头重新生成器（HR）822 确认完整头，HS 810 就停止发送完整头。HR 822 还启动一个初始化为在 Init_info（n）中接收的 RTP 时间戳的本地时间戳计数器。TS 计数器类似于图 1 中的接收器定时器，但是 TS 计数器每 T 毫秒增加 TS_stride（而不是 1，但是其与接收器定时器原理相同）。对于后续剥离的语音帧（也就是已经剥离或去掉头的 RTP 分组），RTP TS 从时间戳（TS）计数器重新生成。接收器定时器（TS 定时器）与在 RTP 源处使用的时钟或定时器（也就是源定时器）有相同的频率以便生成时间戳。而且，类似电路的连接提供基本固定比特率，并且因此，分组延迟从分组到分组不变化或改变。结果，因基本固定的比特率信道而没有分组抖动。因此，在 RTP 接收器接收包括初始化时间戳值（TS0）的初始化信息之后，RTP 接收器可仅基于时间戳计数器（或接收器定时器）来为每个后续分组（在初始化之后）重新生成正确的时间戳。

在头剥离器 810 和头重新生成器 822 之间提供的基本固定比特率信道仅需要在预定的时间周期内在头剥离器 810 和头重新生成器 822 之间提供预定比特数，但是这个功能可执行各种不同的方式。例如，信道可以是专用于剥离器 810 和重新生成器 822 的固定比特率信道或者在几个用户之间共享。信道可提供，例如每毫秒一比特，或提供每 100 毫秒 100 比特，但是在 100 毫秒周期内数据速率可能不固定（也就是是变化的）。作为额外的例子，信道可通过在头剥离器和头重新生成器之间的一个或多个数据字符组提供预定数量的比特。例如，信道可每 10 毫秒提供 1000 比特的组块或字符组。因此，基本固定比特

率信道只需要在预定时间周期提供预定数量的比特，但是可利用不同的技术实现这一点。

2、 RTP 序号（SN）

RTP SN（如由 HS 810 看到的）从一个分组到另一个一般增加 1。唯一的例外是当分组丢失或失序时。在上行链路中，因为头剥离器（HS）810 和 RTP 源彼此非常接近，所以一般不会发生分组丢失或失序。因此，下列应用于下行链路。HS 810 在剥离分组头之前进行有限的缓存来试图对分组重新排序。带有 RTP SN n 的分组如果在具有 RTP SN（n+1）的分组的头被剥离时仍没有接收到，则认为丢失。带有 RTP SN m 的分组如果在其接收时，带有 RTP SN k 的分组的头已经剥离，并且 k>m，则认为失序。重排序缓存的长度是设计参数。缓存太长会导致过大的延迟，而缓存太短会导致太多丢弃的分组。该参数还依赖于 IP 网 108 提供的 HS 810 的上行比特流的质量。HR 822 维护一个作为其 SN 最佳估计值的 SN 计数器。通过观察 Init_info，HR 822 可获得初始 SN 以及已知为分组大小（p_size）的分组中包含的比特数。HR 822 用 Init_info 中的 SN 初始化 SN 计数器。然后 HR 822 “计数”基本固定比特率信道上接收的语音比特并且对于语音的每 p_size 个比特使 SN 计数器增加 1（当没有接收分组时例如无声间隔时不增加）。根据实施方案，HR 822 实际上不计数接收的比特。而是，每个分组持续时间在 HR 822 处 SN 计数器加 1，其中分组持续时间是接收比特（p_bits）分组需要的时间。因此，分组持续时间是分组大小（p_size）和比特率（在类似电路的连接上固定）的函数。

因此，可以看出在初始化发生之后（向 HR 822 提供初始化 SN 和 TS），HR 822 通过每 T 毫秒将 TS 计数器增加 TS_stride，以及通过每个分组持续时间将 SN 计数器加 1 可为后续分组生成时间戳。因此，在初始化之后，这些域可以仅参考本地时钟（假设 HR 822 已知 TS_stride 和分组持续时间）在 HR 822 重新生成。SN 计数器基于时间（分组持续时间）而不是基于实际到达比特的计数来增加对错误更健壮。如果在到达 HR 822 之前丢失一个或多个比特，则 SN 计数器将反映真实值并且将不受丢失比特的影响。

B、 不连续性和串

上述描述指示 TS 和 SN 在通过链路（例如 RF 链路 140）传输之

前可由 HS 810 完全剥离，并且然后由维护本地时钟或定时器（例如，每 T 毫秒增加 TS 计数器 TS_stride 并且每个分组持续时间使 SN 计数器加 1）的 HR 822 重新生成。但是，会出现一个或多个基本不连续事件，如果不解决，很可能使上述基于定时器的重新生成方法无效。一些不连续事件包括:

a) "新突发"事件: 分组 n 和 n+1 之间 TS 差值的瞬时改变（新谈话突发开始）; 这也可以描述为时间戳（TS）的非线性改变或偏移。

b) "大小改变"事件: 由分组中封装的语音帧的数量和/或语音帧大小的改变引起的 RTP 分组大小（p_size）的改变。

c)"跨度改变"事件: TS_stride 的改变（例如，由有效负荷类型 PT 引起的）。

我们定义头串为一系列分组头因此所有的分组大小相同（p_size），序号连续，也就是 n、(n+1)、(n+2)等，并且连续分组的时间戳（TS）以相同的增量 TS_stride 分隔。换句话说，头串被认为是具有共有的一些分组域（例如分组大小）的一串头，以及经过连续分组线性增加的其它域，如 SN 和 TS。一串通常是一个谈话突发（例如，无声间隔之间提供的一系列语音采样）。

由任何不连续事件，单一地或甚至是组合地导致从一个串到另一个串的转变。在这个方案中，当一个串开始时（并且前一个串已经结束），HS 810 确定发生了哪个不连续事件，并且相应地向 HR 822 发送需要的串初始化（string_init）信息。

图 9 是说明根据本发明的示例实施方案在消息中提供的信息的表。Init_info 典型地包括一个完整的头（包括完整的 SN 和 TS），并且在会话开始时从 HS 810 发送到 HR 822（以便初始化 HR 822）。在继续发送无头的数据分组之前，HS 810 继续重新发送 init_info 直到接收到来自 HR 822 的确认。之后，可能出现一个或多个串，要求从一个串到另一个串变化的域或者值的额外更新。这些改变的值利用 string_init 提供给 HR 822。

String_init 包括 p_size 值（如果它从前一个串以来已经变化），以及 TS_stride 值（如果它从前一个串以来已经变化）。如果从一个串到另一个串 TS 没有非线性偏移发生，则 HR 822 可以基于在旧串中使用的 TS 计数器来继续重新生成 TS。但是，如果在串之间时间戳

（TS）中有非线性偏移发生（也就是，丢失定时），则更新的时间戳必须在 string_init 中明确地从 HS 810 发送到 HR 822。只要满足上述条件 1，更新的 TS 就可作为上述压缩的时间戳 414（见图 4）发送。否则，如果不满足条件 1，则必须向 HR 822 发送完整的更新时间戳。

在确认模式，在 HS 810 向 HR 822 发送 string_init 之后，在 HS 810 可以向 HR 822 进一步发送数据分组（无头分组）之前，HS 810 要求 HR 822 确认（或肯定应答）更新的串信息（string_init）的接收。在确认或肯定应答模式，HS 810 重复地向 HR 822 发送 string_init 直到 HS 810 从 HR 822 接收 string_init 消息的肯定应答为止。在从 HR 822 接收到肯定应答之后，然后 HS 810 将串的剩余分组作为剥离头的分组发送（因为新串分组的 TS 和 SN 现在可以仅利用本地时钟或定时器重新生成）。对 string_init 的肯定应答要求（在确认模式）禁止 HS 810 在不通知 HR 822 的情况下发送新串。例如，如果 HS 810 发送新的 string_init 消息（例如，提供与不连续事件相关的更新域或信息）而 HS 810 和 HR 822 之间的链路临时中断，则 HS 810 不能继续发送头剥离的分组直到先接收来自 HR 822 的肯定应答为止。

一旦 HS 810 确信 HR 822 已经接收 string_init 信息，然后在剩余串没有头的情况下发送语音帧（例如，数据分组）。对于这些无头的帧，利用在 HR 822 处的本地时钟重新生成 TS 和 SN。

HS 810 可以确定下列事件：

a) “新突发”事件：SN=n 的分组和 SN=（n+1）的分组之间的 TS 差值与 TS_stride 不同。这意味着新串或谈话突发的开始。在这种情况下，为确保 SN 同步，string_init 由 SN 或压缩的 SN（C_SN）组成。如果没有 SN 信息要发送，则 HR 822 不能确定每个分组持续时间 SN 计数器加 1 能给出准确的 SN。这是因为已经有一个链路断开，期间 HS 810 和 HR 之间的语音比特丢失。

b) “大小改变”事件：SN=n 的 RTP 分组的大小与以前接收的分组不同；这将影响分组持续时间的值（SN 计数器增加的速率）。String_init 包括新的 p_size 值。

c) “跨度改变”事件：由分析 RTP 分组中的有效负荷（PT）域确定；String_init 包括新的 TS_ stride 值。

这些不连续事件仅提供作为例子。可能出现其它类型的不连续事

件。

事件可以组合发生（复合事件）。在那种情况下，string_init 包括来自相应基本事件的所有信息。例如，如果“新突发”与“大小改变”一起发生，string_init={C_SN,新 p_size 值}

C、 发送 init_info、string_init 的过程

init_info 通常在肯定应答模式发送，由此 HS 810 发送 init_info 直到 HR 822 肯定应答为止。String_init 可在肯定应答或不肯定应答模式发送。在肯定应答模式，HS 810 在每个分组发送 string_init 直到 HR 822 肯定应答为止。一旦接收到肯定应答，HS 810 对于剩余串仅发送语音比特，没有任何头。在不肯定应答模式，HS 810 在仅为剩余串发送语音比特之前发送 string_init 一定（预定）的次数。优选地，string_init 在串期间可以以一定的间隔重复以确保 HR 822 同步（例如，有正确的值）。

包括“大小改变”或“跨度改变”基本事件的混合事件典型地要求在肯定应答模式发送 string_init。在那种情况下，string_init 携带一个代号。代号是一个无论何时 p_size 或 TS_stride 改变都增加的计数器。其用于 p_size 或 TS_stride 快速接连改变的情况，以便跟踪哪个改变已经由 HR 822 肯定应答。例如，如果 p_size 从值 p_size_0 变为 p_size_1，然后再变为 p_size_2，则 HS 810 将发送带有代号例如为 3 的包含 p_size_1 的 string_init，然后随后发送带有代号 4 的包含 p_size_2 的 string_init。如果不携带肯定应答的 string_init 的代号，则跟着第二个 string_init 的肯定应答的接收是不明确的。如果混合事件仅是“新突发”，则 String_init（C_SN）可在肯定应答或不肯定应答模式发送。不肯定应答基于 C_SN 至少在每个谈话突发开始时会重复的思想。因此 HR 822 永不与其 SN 重新同步的可能性渐进地小。除此之外，如果 SN 失去同步，则仅是由 HS 810 和 HR 822 之间的分组丢失引起。因此，SN 失去同步的影响是重新生成的 SN<正确的 SN。这只是一个瞬时的不一致，通过下一个 C_SN 一接收到就使 SN 增加差值来纠正。SN 增加超过 1 将由接收 RTP 端点解释为分组丢失（多个丢失），并且一般不影响接收分组本身的回放。不肯定应答模式也允许分配一个信道以稳定状态携带肯定应答，也就是在呼叫建立之后和切换之间。

D、 切换

当在站终端可以从一个网络适配器（ANI_AD）移动到另一个的蜂窝系统或其它系统中应用头剥离/重新生成时，应该考虑切换或移交。

切换可以模型化为经过三个阶段：切换准备、切换执行和切换完成。有称为切换（HO）管理器的功能（在 ANI 110 中提供）来决定开始切换准备。传统地，切换准备由与目标系统交换信令消息以便保留目标系统中的资源以及获得目标小区必要的信息组成。通过源 HO 管理器向接收器终端（或移动站）发送 HO 命令以及关于目标小区的信息来启动切换执行。响应 HO 命令，终端（或移动站）执行切换。切换完成包括在终端或移动站和目标系统之间的信令交换，通知源，以及释放不再需要的资源（例如在源处）。

1、 上行链路

ANI_AD 对于上行链路数据传输（见图 1，上行链路 142）担当 HR 822。必须为目的 ANI_AD 提供必要的信息以便重新生成完整的头。主要限制包括在切换（HO）期间 RTP TS 和 RTP SN 的连续性。

图 10 是说明根据本发明的示例实施方案切换过程的图。终端 130（或移动站 MS）作为例子，在步骤 902 利用 string_init 消息通知源 ANI_AD 112 分组大小已经改变。源 ANI_AD 112 在步骤 904 将这个改变确认到 p_size。随后，终端 130 移动到目标 ANI_AD 114 覆盖的新的区域，并且 HO 管理器 901 在步骤 906 通知源 ANI_AD 准备切换（移交）。然后源 ANI_AD 在步骤 908 向目的 ANI_AD 114 发送一个 HO_initialization（HO_init_u）信息。HO_init_u 是完整 IP/UDP/RTP 头的估计视图。估计视图由最后重新生成的头组成，但是 RTP TS 替换为 TS0_u、m_last_u、TS_stride_u、以及 TS 值 Timer_u。这些值与 TS_last 有关，最后重新生成头的 RTP TS 如下：TS_last=TS0_u+m_last_u*TS_stride_u。TS Timer_u 是在源 ANI_AD 处的计数器，每 T 毫秒增加 1。除此之外，HO_init_u 包括 p_size_u（上行链路方向的分组的当前大小）。目标 ANI_AD 从 HO_init_u 中得出静态域，以及改变域（RTP TS 和 RTP SN）的近似初始值。在步骤 910 切换命令从 HO 管理器 901 发送到终端 130（移动站），导致终端 130 转换并且现在使用目标 ANI_AD 进行通信。但是，因为可使

用其它技术启动切换，所以 HO 管理器不是必须的。

切换被认为中断了任何进行中的串。因此，在切换完成之后，要发送的第一个语音采样总是类似新串来处理，在步骤 912 其要求发送初始化信息（HO_sync_u）。有三个重要的时间点：HO 准备开始的 ST1，MS 接收 HO 命令的 ST2，以及源 ANI_AD 取得 HO_init_u 中要发送的其内部信息的快照的时间 ST3。使 HOT 成为从 ST1 到 ST2 流逝的时间。根据系统设计，HOT 有一个上限：HOT<HOT_max。第四个重要的时间点是 ST4：当终端 130 在 HO 之后想要恢复在目标系统中发送语音时的第一时间。在 ST4，终端 130（MS）确定到时间 ST2-HOT_max，p_size_u 最近的改变是否已经确认。如果是，则终端确定 HO_init_u 包含 p_size_u 的最新值。因此，不需要在 HO_sync_u 中包括它。这是因为时间点的顺序是 ST1<ST3<ST2。否则，终端 130（MS）将在 HO_sync_u 中包括新的 p_size_u 值。相同的算法适用于 TS_stride_u。

在所有情况下，HO_sync_u 包括 C_SN。需要 C_SN 是因为有 HO 引起的中断。如果在源和目标系统中的比特率、分组持续时间等不同步，则需要 C_TS。很可能是这种情况。HO_sync_u 优选地在肯定应答模式下发送。

目标 ANI_AD 114 使用 HO_init_u 和 HO_sync_u 来重新生成下列完整头。除了 TS 和 SN 之外的所有域都从 HO_init_u 拷贝。通过解压缩 HO_sync_u 中的 C_SN 获得 SN。通过解压缩 HO_sync_u 中的 C_TS 确定 TS。

2、 下行链路

HS 角色从一个 ANI_AD 转变为另一个。在移交之后，头在新的路径上通过新的 ANI_AD 代替旧的 ANI_AD 路由。结果，在终端 130（MS）处在 RTP TS 重新生成的定时中有不连续。

为处理上行链路的切换，当 HO 管理器决定开始切换准备时，其通知源 ANI_AD。然后源 ANI_AD 向目标 ANI_AD 发送 HO_initialization（HO_init_d）信息。HO_init_d 由 p_size_d 和 TS_stride_d 组成，这些和其代号一起是 MS 最后确认的值。当目标 ANI_AD 在 HO 之后想要发送语音的第一时间，目标 ANI_AD 必须发送 HO_sync_d。HO_sync_d 由 C_TS 和 C_SN 组成。如果新的 p_size

与 p_size_d 不同，则 HO_sync_d 还包括新的 p_size 值。如果不是，则 HO_sync_d 只包括 p_size_d 的代号。MS 使用代号来取回正确的 p_size。这假设 MS 将 p_size 的最后几个值与其代号一起保持在存储器中。相同的算法适用于 TS_stride。直到 MS_AD 肯定应答之后才发送 HO_init_d。HO_sync_d 在肯定应答模式发送。切换过程在图 2 中描述。显示的情况是：p_size_u 最近的改变在不迟于时间 ST2-HOT_max 已经确认。

E、 发送消息

上述每个信息可以在带内或带外发送。在带内方法中，信息通过窃取最低有效语音位在语音信道上发送。在带外方法中，建立专用瞬时信道并且在接收到肯定应答时拆除。带内和带外的组合也有可能，由此试图用带外方法，但是如果没有资源用于瞬时信道则带内方法是后退的解决方案。确认可在带内发送，或在其自身专用肯定应答信道上带外发送，或在其它专用瞬时信道（TIC）等上捎带确认地带外发送。

1、 带内

不管类似电路的语音信道如何实现，其可以模型化为每 T 毫秒传输 B 比特。如果 S 是比特计的语音帧的大小，则 S≤B。通过想象的语音编解码器，Init_info 预计大于 S。因此 Init_info 不能在单一语音帧的空间里发送。但是，有一个因数 R>=1，以致（R-1）*S<H≤R*S。通过将其分割为 B 比特的块并且每 T 毫秒发送一个块可以在类似电路的信道上发送 Init_info（n）。一个完整的头将消耗 R 个连续语音采样的空间。图 11 是说明根据本发明的示例实施方案带内初始化的图。如果有连续的语音活动，则发送的 Init_info 是 Init_info（0）、Init_info（R）、Init_info（2R）等，直到接收到肯定应答为止。在图 11，这些 Init_info 消息显示为 Init_info 500 和 Init_info 502。头剥离器肯定应答 Init_info 500，但不是在 HS 810 发送第二个 Init_info 分组 502 之前。下一个分组 504 作为分组有效负荷 504（没有头）从 HS 810 发送到 HR 822。然后 HR 822 重新生成 SN 和 TS 以及其它头域。

Init_info（0）代替语音采样 0、1、...、（R-1），Init_info（R）代替语音采样 R、（R+1）、...、（2R-1）等。如果有不连续的语音活动，假定头 0 跟随有 L*T 毫秒的无声间隔，则重复 Init_info（0）：其它信

息（string_init、HO_sync_u、Ack）,大小都小于 S，因此其适合于语音帧的空间。其窃取最低有效语音比特。为简单起见，分析不考虑信道编码引起的扩展，但是有或没有信道扩展该概念都是合法的。图 3 中显示带内情况的初始化过程。

2、 带外

图 12 是说明根据本发明的示例实施方案带外初始化的图。在带外方法中，建立具有合适带宽的单独信道以便正好与在语音信道上传送的语音同时传送 Init_info。单独的信道称为瞬时初始化信道（TIC）。系统试图为 TIC 分配足够的带宽以便能够每 T 毫秒发送一次完整的头。TIC 设计为与语音信道有固定的定时关系。

通过分配瞬时确认信道（TAC）确认可在带外发送，或者带外发送，但是在前向瞬时信道上捎带确认。HO_sync_u 可在瞬时上行链路切换同步信道（TUHOSC）上带外发送。当 HO_sync_u 肯定应答时，拆除 TUHOSC。相同的方法应用于使用瞬时下行链路切换同步信道（TDHOSC）的 HO_sync_d 上。

3、 失败的情况

有到切换执行完成时目标 ANI_AD 还没有 HO_init 的情况。原因包括在两个 ANI_AD 之间的信令网上过多的延迟，需要快速执行切换等。在那些情况下，网络向 MS 发送通知，其然后象在通话开始时一样重新开始初始化过程。

4、 P_size 和 TS_stride 是固定的常见情况

P_size 和 TS_stride 是固定的情况到目前为止对于语音是最常见的。在这种情况下，P_size 和 TS_stride 可能改变引起的任何考虑都不适用。普遍的方案是简化的。不需要 HO_init_d。HO_sync_d 和 HO_sync_u 仅携带 C_SN 和 C_TS。String_init 携带 C_SN。仅当从一个串到另一个串有定时改变时才携带 C_TS。终端（MS）不需要在存储器中保持 p_size 和 TS_stride 的最近几个值。在 HO 的情况下，终端（MS）不需要确定是否在 HO_sync_u 中包括 p_size。

如上所述，IP/UDP/RTP 头中对于基本语音必须的唯一信息是静态域，并且 RTP 时间戳（TS）以及 RTP 序号（SN）也非常值得要。这里描述的方案对这些信息域实现了透明，并且提供有利的头系统开销压缩效率。在切换中保持了所有静态和非静态域的连续。因为带内

以及带外方法是可能的，所以带宽管理也变得更简单。因为对 RTP TS 和 RTP SN 保持了透明，因此甚至可以在这里描述的头剥离方案和头压缩方案之间来回转换以保持对所有域的透明。例如当在语音中加入另一个介质时需要转换到头压缩。

III、基于定时器和参考的方案

A、 基于定时器和参考的方案概述

基于定时器和参考的方案是基于这样的观察（1）RTP 时间戳当在 RTP 源处生成时与分组之间流逝时间的线性函数相关，以及（2）RTP TS 是 TS0+index*TS_stride 的形式，其中 TS0 和 TS_stride 是固定的，并且 index 是整数（在下文中 index 称为压缩的 RTP TS）.

因此，在正常操作中，在解压缩器接收的 RTP 时间戳也是与连续增加的定时器，与仅由源和解压缩器之间的累积抖动生成的失真相关的。因为累积抖动包括“网络”抖动（源和解压缩器之间的抖动）和“无线电”抖动（压缩器和解压缩器之间的抖动），所以压缩器通过将无线电抖动的上界加到观察的网络抖动中可计算出累积抖动的上界。然后压缩器只发送压缩的 RTP TS 的“k”个最低有效位作为压缩的 RTP TS。解压缩器通过首先计算近似值，并且然后用压缩的 RTP TS 中的信息改进近似值以便确定准确值来解压缩 RTP TS。通过将与自从接收到前一个解压缩头以来流逝的时间成正比的值加到前面解压缩的头的 RTP TS 中来获得近似值。RTP TS 的确切值确定为与近似值最接近的一个值，其相应的压缩 RTP TS 的 k 个最低有效位与压缩的 RTP TS 相匹配。压缩器选择 k 值作为使得解压缩器能够基于累积抖动的上限正确解压缩的最小值。

B、 语音的情况

首先，将关于语音描述基于定时器和参考的方案。作为例子，如果连续语音采样之间的时间间隔是 20 毫秒，则头 n 的 RTP 时间戳（在时间 n*20 毫秒生成）=头 0 的 RTP 时间戳（在时间 0 生成）+TS_stride*n，其中 TS_stride 是依赖于语音编解码器的常数。因此，进入解压缩器的头中的 RTP TS 也遵循作为时间的函数的线性模式，但是因源和解压缩器之间的延迟抖动，不是很接近。在正常运行时（没有崩溃或失败），延迟抖动限制于满足会话实时业务的要求。

在这个方案中，接收器利用定时器获得当前头（要压缩的那个）

的 RTP TS 的近似值，然后用从压缩的头中获得的额外信息来改进近似值。

例如，假设如下：

Last_header 是最后成功解压缩的头，其中 TS_last 是最后的 RTP TS，并且 p_TS_last 是最后压缩的 RTP TS（在接收器）；

T 是两个连续语音采样之间的正常时间间隔；

TS_stride 是每 T 毫秒的 RTP TS 增加值；

Current_header 是要解压缩的当前分组的头，其中 TS_current 是当前 RTP TS，并且 p_TS_current 是当前压缩的 RTP TS；

RFH 是压缩器接收到其肯定应答的头的序号，其中 TS_RFH 是 RTP TS，并且 p_TS_RFH 是压缩的 RTP TS；

定时器是每 T 毫秒增加的定时器，其中压缩器和解压缩器每个都维护一个自己的定时器，分别称为 S_timer 和 R_timer；

T_RFH 是 RFH 被接收到时定时器的值，并且 T_current 是当 Current_header 被接收到时相同定时器的值；以及

N_jitter（n,m）是观察的与头 m 相关的头 n（头 n 在头 m 之后接收）的网络抖动，

其中 N_jitter（n,m）由压缩器计算如下：

N_jitter(n,m)=Timer(n,m)-(头 n 的压缩的 RTP TS-头 m 的压缩的 RTP TS)，

其中 Timer(n,m)是从头 m 到头 n 流逝的时间，以 T 毫秒为单位表示。N_jitter（n,m）可以是正的或负的。在压缩器的 N_jitter 是网络抖动，量化为 T 毫秒为单位。

R_jitter（n,m）是由压缩器预测的与头 m 相关的头 n 的无线电抖动。R_jitter 仅依赖于压缩器-解压缩器信道（CD-CC）的特性。R_jitter 不需要精确计算，对于 R_jitter 有一个好的上界就足够了。例如，上界可以是 Max-radio_jitter，如果已知，是在 CD-CC 上的最大抖动。

因此，根据上述，分组的累积抖动计算为网络抖动和无线抖动的和：

而且，RTP TS 计算如下：

RTP TS=TS0+index*TS_stride，

其中 TS0<TS_stride 并且 index 是整数。

因此 TS_last=TS0+index_last*TS_stride，并且

TS_current=TS0+index_current*TS_stride。

1、 压缩器

压缩器在压缩的头中发送 k 个最低有效位 p_TS_current。

压缩器运行下列算法来确定 k:

计算 Max_network_jitter;

计算 J1= Max_network_jitter+ Max_radio_jitter+J,

其中 J=2 是考虑由压缩器和解压缩器引起的量化错误的因数，其可以是+1 或-1；以及

找到满足以下条件的最小的整数 k:

（2*J1+1）$<2^k$。

可以根据三种不同的方法计算压缩器处的网络抖动，即图 13 所示的第一种方法，图 14 所示的第二种方法以及图 15 所示的第三种方法。第二种和第三种方法下面分别描述为选项 1 和选项 2。第一种方法对于计算网络抖动足够了。但是，在压缩器处计算网络抖动的优选方法是下面分别描述为选项 1 和选项 2 的第二种和第三种方法。

如图 13 所示，根据第一种方法，利用关于紧接着前面分组的信息计算压缩器处特定分组的网络抖动。因此，例如，利用关于分组 1 的信息计算分组 2（j2）的网络抖动，利用关于分组 2 的信息计算分组 3（j3）的网络抖动，利用关于分组 3 的信息计算分组 4（j4）的网络抖动，利用关于分组 4 的信息计算分组 5（j5）的网络抖动。

因此，根据图 13，分组 2 的网络抖动等于计算的抖动 j2，分组 3 的网络抖动等于计算的抖动 j3，分组 4 的网络抖动等于计算的抖动 j4，并且分组 5 的网络抖动等于计算的抖动 j5。

选项 1:

图 14 说明了用于为选项 1 的第二种方法计算网络抖动的步骤。在选项 1 中利用关于参考分组的信息计算特定分组的网络抖动。因此，假设如图 14 所示分组 2 是参考分组，利用关于参考分组 2 的信息计算分组 3 的抖动 j3，利用关于参考分组 2 的信息计算分组 4 的抖动 j4，并且利用关于参考分组 2 的信息计算分组 5 的抖动 j5。

根据如图 14 所示的选项 1 的第二种方法，如果假设抖动 j3=2，

抖动 j4=3，并且抖动 j5=-1，则在分组 5 之前 N_jitter_min=2 并且 N_jitter_max=3，而在分组 5 N_jitter_min=-1 并且 N_jitter_max=3。因此，在分组 5 的最大网络抖动=N_jitter_max- N_jitter_min=4。因此，分组 5 的 Max_network_jitter 是 4。下面提出根据选项 1 的方法计算网络抖动的等式及其描述。

根据选项 1 的方法当前分组的网络抖动计算如下：

N_jitter（current_header，RFH）=（T_current-T_rfh）-(p_TS_current-p_TS_RFH);

更新 N_jitter_max 和 N_jitter_min，其中 N_jitter_max 定义为 Max{N_jitter(j,RFH)}，对于从 RFH 并且包括 RFH 发送的所有的头 j。N_jitter_min 定义为 Min{N_jitter(j,RFH)}，对于从 RFH 并且包括 RFH 发送的所有的头 j；并且

计算 Max_network_jitter=（N_jitter_max-N_jitter_min）。

应该指出，N_jitter_max 和 N_jitter_min 可以是正数或负数，但是（N_jitter_max-N_jitter_min）是正数。

选项 2：

图 15 说明了用于为选项 2 的第三种方法计算网络抖动的步骤。在选项 2 中利用感兴趣的分组与每个预定数量的前面分组之间的抖动计算来计算特定分组的网络抖动。预定数量的前面的分组定义为一个窗口并且这样的窗口可以是任意值。在图 15 所示的例子中，窗口有四个前面分组的值。窗口可以设置为任何其它值，例如，7 个分组。而且，窗口可以设置为，例如，等于从最后参考的分组以来的分组数的值。

如图 15 所示，利用关于分组 1j（5，1）、分组 2，j(5，2)、分组 3j（5,3）以及分组 4，j(5,4)的信息计算分组 5 的网络抖动。如图 15 所示，为分组 5 关于每个分组计算的网络抖动分别是关于分组 1 是 j(5,1)=-2，关于分组 2 是 j(5,2)=3，关于分组 3 是 j(5,3)=4，并且关于分组 4 是 j(5,4)=7，然后 max_network_jitter=7。下面提出根据选项 2 的第三种方法计算网络抖动的等式和描述。

根据选项 2 的方法计算当前分组的网络抖动如下：

对于在当前头之前发送的所有头计算的 N_jitter（current_header，j）=（T_current-T_j）-(p_TS_current-p_TS_j)，

并且属于窗口 W,其中 T_j 是当接收头 j 时的定时器值，并且 p_TS_j 是头 j 的压缩 RTP TS; 并且

在窗口 W 中所有 j 上计算 Max_network_jitter=| 最大 N_jitter(Current_header，j)|。

在来自解压缩器的反馈可用的情况下，窗口 W 包括从知道正确接收（例如，确认的）最后一个头以来发送的头。在没有反馈的情况下，窗口 W 包括发送的最后 L 个头，其中 L 是参数。

2、 解压缩器

为解压缩 Current_header 的 RTP TS，接收器以 T 毫秒为单位计算从接收 last_header 以来流逝的时间。该时间，Timer(Current_header， Last_header) 加到 p_TS_last，以便给出 p_TS_current 的近似值。然后接收器通过选择其 k 个最低有效位与压缩的 RTP TS 相匹配的最接近近似值的值确定 p_TS_current 的精确值。然后 TS_current 计算为 TS0+（p_TS_current）*TS_stride。

Timer(Current_header,Last_header) 可计算为（T_current-T_last),其中当 Current_header 和 Last_header 分别接收时 T_current 和 T_last 是 R_timer 的值 。

3、 正确性检验

为了检验基于定时器和参考的方案的正确性，假设如下:

Approx_TS 是 p_TS_current 的近似值，由解压缩器计算为 p_TS_last+ Timer(Current_header,Last_header); 并且

Exact_TS 是 p_TS_current 的精确值。

然后基于上述:

|Approx_TS-Exact_TS|<=| Jitter(Current_header,Last_header)|;

因在压缩器的 Max_network_jitter 的定义:

| Jitter(Current_header,Last_header)| < J1,

其中 J1=Max_network_jitter+Max_radio_jitter+J。

J 是考虑由压缩器和解压缩器处的定时器引入的量化错误加入的因数，其可以是+1 或-1。因此 J=2 足够。

因此，其遵循:

|Approx_TS-Exact_TS| < J1

为了没有不明确地计算 Exact_TS，选择 k 以致于满足（2*J1+1）

$<2^k$就足够了。

4、 在压缩器之前分组失序的情况

可由减少的 RTP 序号（RTP SN）检测到分组失序。当其发生时，压缩器可以利用不同的方案，例如，VLE 解码压缩的 RTP TS。利用压缩头中合适的指示位通知解压缩器不同的解码。

另一种选择是采用正常的基于定时器和参考的方案算法-失序很可能导致更大的 k 值。

5、 上行链路

在无线系统中，对于上行链路方向，网络抖动是零（因为 RTP 源和压缩器都位于无线终端中），并且无线电抖动通常被限制并且控制以便保持非常小。因此，想要的 k 非常小并且固定，其最小化了头大小的波动。这是对于带宽管理非常有意义的优点，因为对于上行链路，终端通常必须向网络要求越来越多的带宽。而且，没有分组失序。因此基于定时器的方案特别适合于上行链路。

6、 下行链路

对于下行链路方向，网络抖动不是零，但是整体抖动通常很小以满足实时要求。想要的 k 值仍然很小并且通常固定。在 k 中会有更大的波动，但是带宽管理不是问题，因为网络控制带宽分配。

7、 移交

在蜂窝系统中，有 MS 到网络无线链路和网络到 MS 无线链路，分别称为上行链路和下行链路。当压缩/解压缩应用于蜂窝链路时，有一个基于 MS 的功能，MS_AD(MS 适配器)，其分别完成上行链路和下行链路的压缩和解压缩。有一个基于网络的实体，称为 ANI_AD(接入网基础结构适配器)，分别完成上行链路和下行链路的解压缩和压缩。

要考虑的移交特殊情况是ANI_AD间的移交，其中从旧的ANI_AD转换到新的 ANI_AD 可导致中断。问题是如何在移交过程中保持信息的连续性以便在移交之后，在 MS_AD 和新的 ANI_AD 处的压缩/解压缩没有中断的继续。

有两种替代的方法用于移交，描述如下：

a、 第一种方法

第一种方法使用以与 1999 年 3 月 9 日 K.Le 提交的序号 09/522,497

的相关申请“用于头压缩的一种有效移交过程（AN EFFICIENT HANDOFF PROCEDURE FOR HEADER COMPRESSION）”中所公开的握手方法获取在 ANI_AD 和 MS_AD 之间交换的上下文信息的快照的方案。对于 RTP TS，上下文信息包括参考头的完整 RTP TS。在移交之后，压缩器（对于上行链路是 MS_AD，对于下行链路是 ANI_AD）利用基于定时器的方案临时停止并且发送关于参考值的压缩的 RTP TS。例如，可使用 1999 年 3 月 9 日提交的序号为 09/522,497 的 K.Le 的“用于头压缩的一种有效移交过程（AN EFFICIENT HANDOFF PROCEDURE FOR HEADER COMPRESSION）”的相关申请中所公开的 VLE 解码。一旦接收到确认，压缩器就使用确认的值作为 RFH，并且转换回基于定时器的方案。

b、 第二种方法

第二种方法在移交过程中保持使用基于定时器的方案。

i、 下行链路

在是 MS 的接收器端没有中断。压缩器的任务从一个 ANI_AD 转移到另一个。移交之后，头通过新的 ANI_AD 代替旧的 ANI_AD 在新路径上路由。

-压缩器

旧的 ANI_AD 利用握手的方法将下列信息的快照转移到新的 ANI_AD：T_RFH、p_TS_RFH、S_Timer 的当前值、TS0 以及 TS_stride。（快照值用*表示，例如，T_RFH*）。新的 ANI_AD 用从旧的 ANI_AD 接收的 S_Timer 的当前值初始化其 S_Timer 并且开始每 T 毫秒增加该定时器。用旧的 ANI_AD 的当前 S_Timer 值初始化其 S_Timer 是一个概念的描述。如果有单一的 S_Timer 由多个流共享，则实际 S_Timer 不重新初始化。而是，记录该 S_Timer 与来自旧的 ANI_AD 的值之间的偏移。在将来的计算中考虑该偏移。为压缩移交后的第一个头，新的 ANI_AD 发送 p_TS_current 的 k 个最低有效位。新的 ANI_AD 将 k，使用的位数确定如下：

J2=N_jitter(Current_header , RFH*) 的 上 界 +Max_radio_jitter+J，

其中选择 k 满足条件（2*J2+1）$<2^k$。

在上面，Max_radio_jitter 是在新的 ANI_AD 和 MS_AD 之间的

段上的最大抖动。

N_jitter(Current_header，RFH*)的上界计算如下：

|Timer(Current_header，RFH*)-(p_TS_current-p_TS_RFH*)|+T_transfer，其中 Timer(Current_header，RFH*)是(T_current-T_RFH*)；

T_current 是当接收到 Current_header 时新的 ANI_AD 中 S_Timer 的值；

T_RFH*是从旧的 ANI_AD 接收的值；

T_transfer 是以 T 毫秒为单位表示的从旧的 ANI_AD 向新的 ANI_AD 传送上下文信息的时间的上界；并且

J=2。

-解压缩器

为了解压缩 Current_header 的 RTP TS，接收器计算以 T 毫秒为单位的自从接收 RFH 以来流逝的时间。该时间，Timer(Current_header，RFH)加到 p_TS_RFH，以便给出 p_TS_current 的近似值。然后接收器通过选择其 k 个最低有效位与压缩的 RTP TS 相匹配的与近似值最接近的值来确定 p_TS_current 的精确值。然后 TS_current 计算为 TS0+(p_TS_current)*TS_stride。

从接收 RFH 以来流逝的时间可计算为(T_current-T_RFH)。

-失败的情况

当上下文信息不能以及时的方式转移到新的 ANI_AD 时，新的 ANI_AD 将发送完整的 RTP TS 直到接收到确认为止。

ii、 上行链路

解压缩器的任务从一个 ANI_AD 转移到另一个。压缩器保持固定在 MS。

-解压缩器

旧的 ANI_AD 利用握手的方法将下列信息的快照传送到新的 ANI_AD：T_RFH*、p_TS_RFH*、R_Timer 的当前值、TS0、以及 TS_stride。新的 ANI_AD 利用从旧的 ANI_AD[2]接收的 R_Timer 的当前值初始化其 R_Timer 并且开始每 T 毫秒增加该定时器。用旧的 ANI_AD 的当前 R_Timer 值初始化 R_Timer 是一个概念的描述。如果有单一的 R_Timer 由多个流共享，则实际 R_Timer 不重新初始化。

而是，记录该 R_Timer 与来自旧的 ANI_AD 的值之间的偏移。在将来的计算中考虑该偏移。为解压缩移交后的恰好第一个头，新的 ANI_AD 计算 Timer(Current_header，RFH),将其加到 p_TS_RFH*，以便给出 p_TS_current 的近似值。然后接收器通过选择其 k 个最低有效位与压缩的 RTP TS 相匹配的与近似值最接近的值来确定 p_TS_current 的精确值。然后 TS_current 计算为 TS0+（p_TS_current）*TS_stride。

Timer(Current_header，RFH)可估计为（T_Current-T_RFH*）。T_current 是当接收 Current_header 时 R_Timer 的值。

-压缩器

MS_AD 发送 p_TS_current 的 k 个最低有效位。其将要使用的位数 k 确定如下：

计算 J2=N_jitter(Current_header，RFH*) 的上界 +Max_radio_jitter+J，

其中选择 k 满足条件（2*J2+1）$<2^k$。

这里，Max_radio_jitter 是在新的 ANI_AD 和 MS_AD 之间的段上的最大抖动。

N_jitter(Current_header，RFH*) 的上界计算为 |Timer(Current_header，RFH*)-(p_TS_current-p_TS_RFH*)|+T_transfer，

其中 Timer(Current_header，RFH*)是（T_current-T_RFH*）；

T_current 是当接收到 Current_header 时新的 ANI_AD 中 S_Timer 的值；

T_RFH*是从旧的 ANI_AD 接收的值；

T_transfer 是以 T 毫秒为单位表示的从旧的 ANI_AD 向新的 ANI_AD 传送上下文信息的时间的上界；并且

J=2

-失败的情况

当上下文信息不能以及时的方式转移到新的 ANI_AD 时，新的 ANI_AD 将通知 MS_AD，其发送完整的 RTP TS 直到接收到确认为止。

8、 方案的性能

因谈话的实时要求，希望正常操作中的累积抖动最多仅有 T 毫秒的几倍。因此，大约 4 或 5 的 k 值足够了，因为多达 16 到 32 的语音采样的抖动可以被纠正。

这个方案的优点如下:

压缩头的大小是固定的并且很小。压缩头典型的由指示消息类型的消息类型（k1 比特）、指示哪个域在改变的位屏蔽，以及包含 index_current 的 k 个最低有效位（k 比特）的域组成。假设使用 4 位 MSTI 位屏蔽，并且 k1=4，当仅 RTP TS 改变时（这种情况是最常见的）压缩头的大小是 1.5 字节。而且，这个大小不作为无声间隔长度的函数而改变。

不需要定时器过程和解压缩器过程之间的同步。

因为压缩头中的部分 RTP TS 信息是自包含的并且仅需要与接收器的定时器相结合来生成完整的 RTP TS 值，所以对错误有健壮性。头的丢失或破坏不会使后续的压缩头无效。

压缩器几乎不需要维护存储器信息:

选项 1 中的 T_RFH、p_TS_RFH、N_jitter_max、N_jitter_min、TS0、以及 TS_stride 以及选项 2 中的{T-j、p_TS_j}，对于窗口 W 中的所有的 j，TSO，以及 TS_stride。

C、 抖动减少

因谈话的实时要求，可以合理的期望上述各种抖动与正常操作中的几个 T 毫秒的相似。但是，不能排除抖动更大并且因此需要一个更大的 k 的情况。例如在从 RTP 源到接收器的路径上可以有异常情况（失败等），其中抖动变得过多。而且，有想要或者值得要固定值的 k 的情况。为解决这些情况，可实现抖动减少功能作为压缩器的前端，来过滤出具有过多抖动的分组（也就是，抖动超过某个阈值）。

在固定的情况（没有移交），抖动计算为 J1 并且与固定阈值比较如下:

J1=（n_jitter_max-N_jitter_min）+Max_radio_jitter+J。

在移交情况下，抖动计算为 J2 并且与移交阈值比较如下:

J2=|Timer(current_header,RFH*)-(p_TS_current-p_TS_RFH*)|+T_transfer+Max_radio_jitter+J。

关于固定非移交情况的主要差异是 T_transfer 的增加。实际上，

为了能够在100毫秒执行移交，T_transfer必须限制为大约100毫秒，因此T_transfer=大约5或6以T毫秒为单位（T=20毫秒）。值k=5足够了。

固定和移交的阈值可以相同或不同。

D、 视频的情况

在RTP视频源的情况下，不需要真的分组之间有固定时间间隔，并且此外，RTP TS不需要从一个分组到另一个增加固定的跨度。但是，RTP TS和分组间的时间间隔是不连续的。因此，如下：

分组m的RTP时间戳=分组0的RTP时间戳（在时间0生成）+TS_stride*[index+adjust(m)]，

其中TS_stride是依赖于编解码器的常数，并且adjust(m)是依赖于m并且反映关于语音中类似的线性行为的差值的整数；以及

两个连续分组之间的时间间隔是T毫秒数的整数倍。

下面，在RTP源的行为称为调节的线性行为。与语音使用相同的符号，TS_last=TS0+TS+stride*[index_last+adjust(index_last)]，并且

TS_current=TS0+TS_stride*[index_current)+adjust(index_current]。调节参数可以是正数或负数。因此，与语音相比的主要差别是额外的术语调节。

RTP TS是进入解压缩器的头，也遵守作为时间函数的调节的线性模式，因源和解压缩器之间的延迟抖动，不是很接近。在正常运行中（没有崩溃或失败），延迟抖动限制为满足谈话实时业务的要求。

如上所述假设当前头的压缩的RTP TS=index_current+adjust(index_current)。关于p_TS_current将使用例如，相同的符号。

压缩器

压缩器在压缩的头中发送p_TS_current的k个最低有效位。确定k的算法与语音相同。

解压缩器

使用的算法与语音相同。

1、 移交

为语音描述的用于移交的两个替代方法也适用于视频。

2、 k 的值

对于语音，显示 k=4 或 5 足够了（2^k=16 或 32）。在视频的情况下，因为调节所以需要更大的 k 值。因为视频的结构为每秒 30 帧，|Adjust|<30。因此，k=7 或 8 在正常运行中应该足够了。

这里特别说明和/或描述了本发明的几个实施方案。但是，应该理解在不背离本发明的精神和预定范围的情况下本发明的修改和变化由上述教导覆盖并且在所附的权利要求的范围之内。

虽然本发明已经被结合附图详细地进行了描述，因为在不背离本发明的精神和其范围的情况下可进行本领域的普通技术人员可认知的许多改变和修改，因此其不限于这些细节。

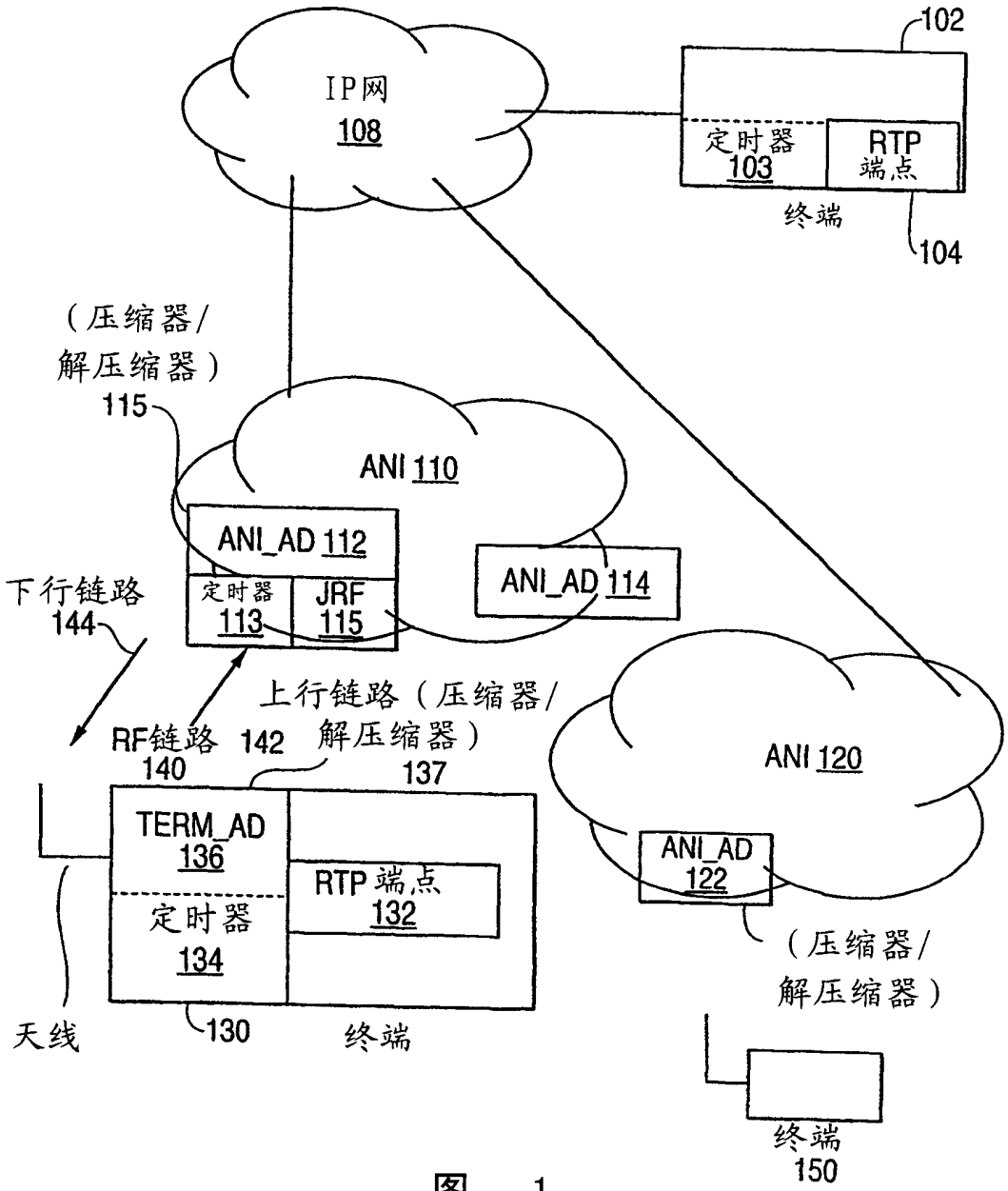

IP网
108
102
定时器
103
RTP
端点
终端
104
（压缩器/
解压缩器）
115
ANI 110
ANI_AD 112
定时器
113
JRF
115
ANI_AD 114
下行链路
144
RF链路 142
140
上行链路（压缩器/
解压缩器）
137
ANI 120
TERM_AD
136
RTP 端点
132
定时器
134
ANI_AD
122
（压缩器/
解压缩器）
天线
130
终端
终端
150

图 1

RTP分组格式

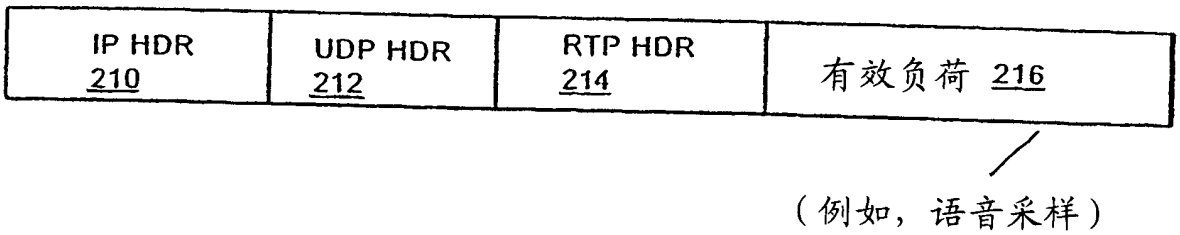


图 2

未压缩RTP头格式

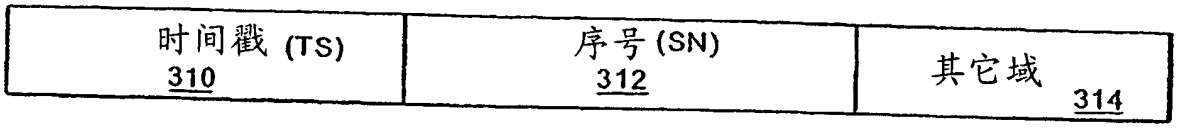


图 3

压缩RTP头格式

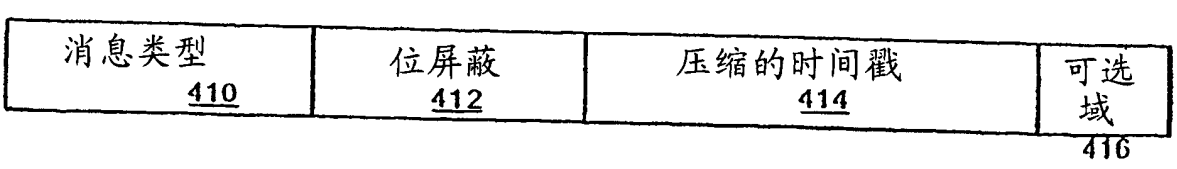


图 4

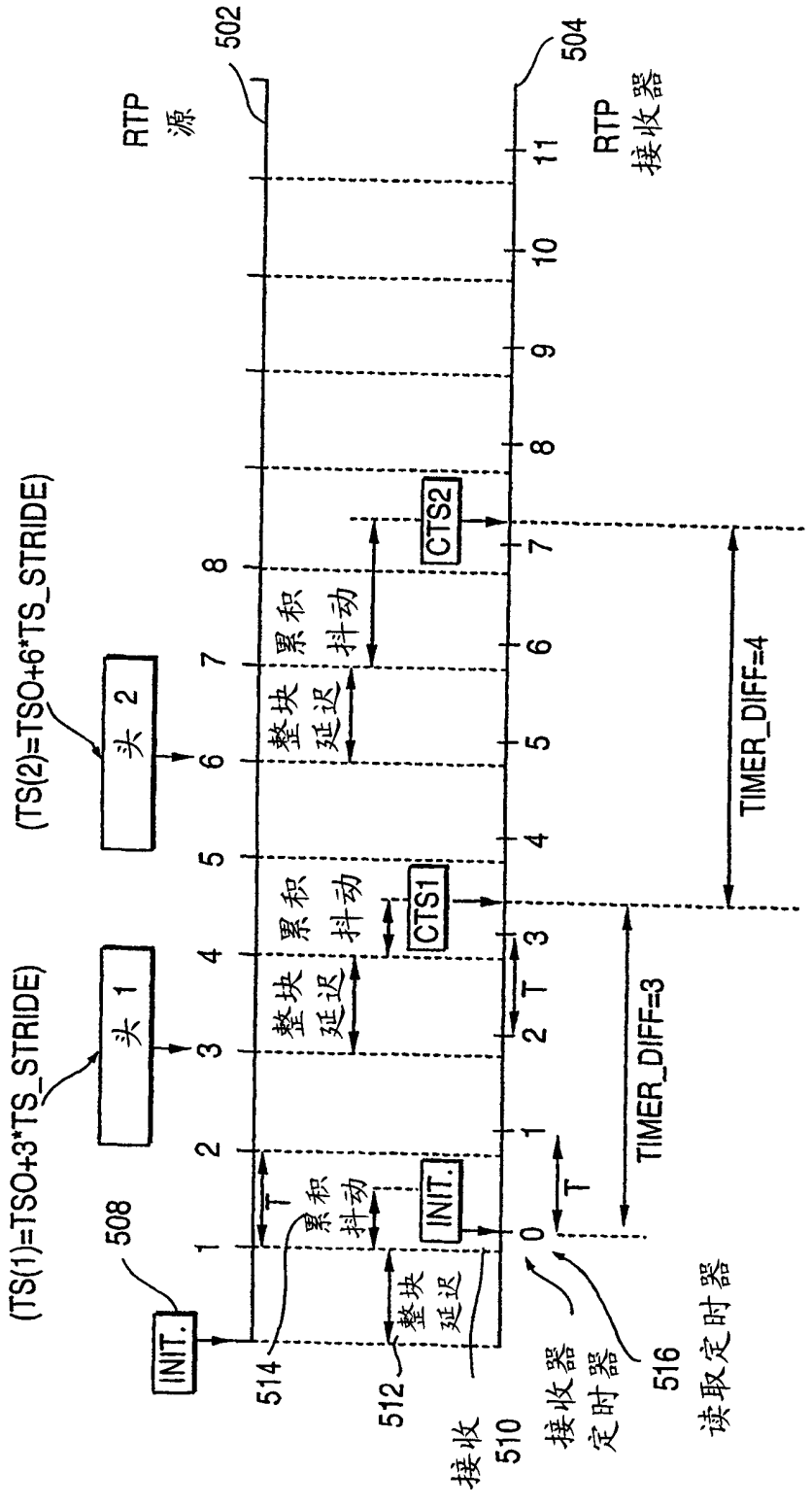

RTP
源
502
RTP
接收器
504
(TS(1)=TSO+3*TS_STRIDE)
(TS(2)=TSO+6*TS_STRIDE)
头 1
头 2
INIT.
508
累积
抖动
整块
延迟
CTS1
CTS2
T
TIMER_DIFF=3
TIMER_DIFF=4
514
512
510
516
接收
接收器
定时器
读取定时器
0
1
2
3
4
5
6
7
8
9
10
11

图 5

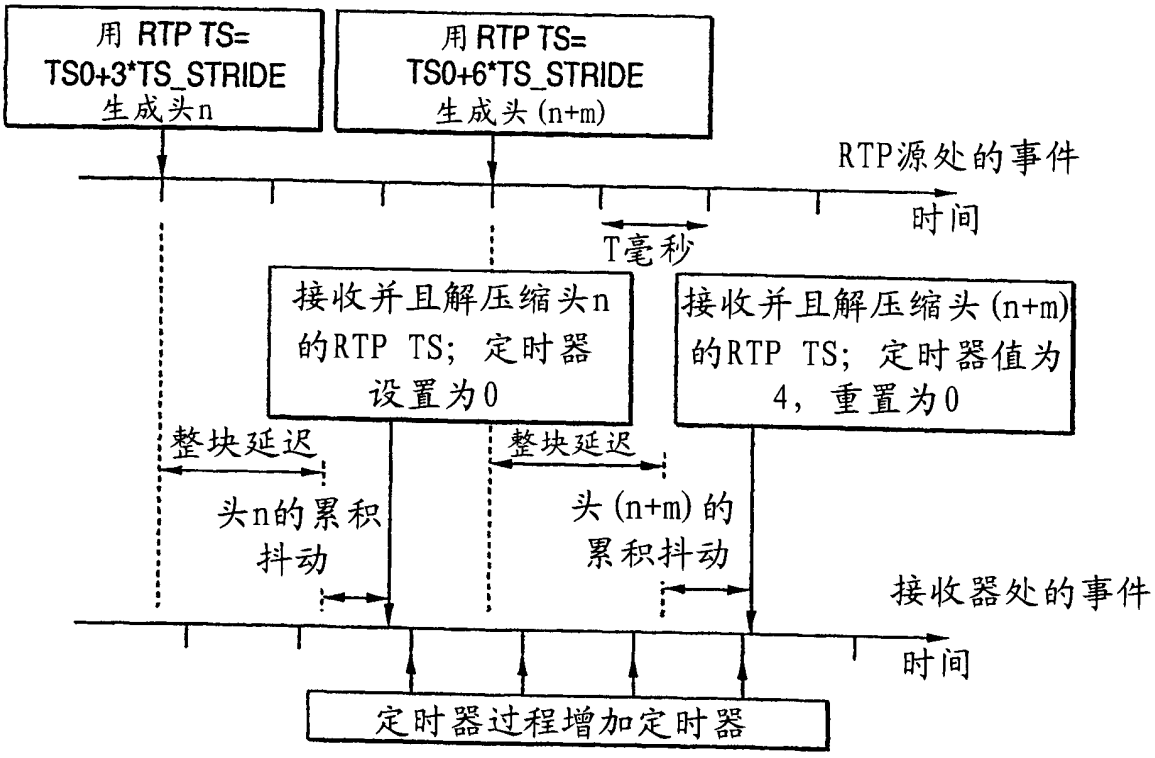

用 RTP TS=
TS0+3*TS_STRIDE
生成头n
用 RTP TS=
TS0+6*TS_STRIDE
生成头(n+m)
RTP源处的事件
时间
T毫秒
接收并且解压缩头n的RTP TS；定时器设置为0
接收并且解压缩头(n+m)的RTP TS；定时器值为4，重置为0
整块延迟
整块延迟
头n的累积抖动
头(n+m)的累积抖动
接收器处的事件
时间
定时器过程增加定时器

图 6

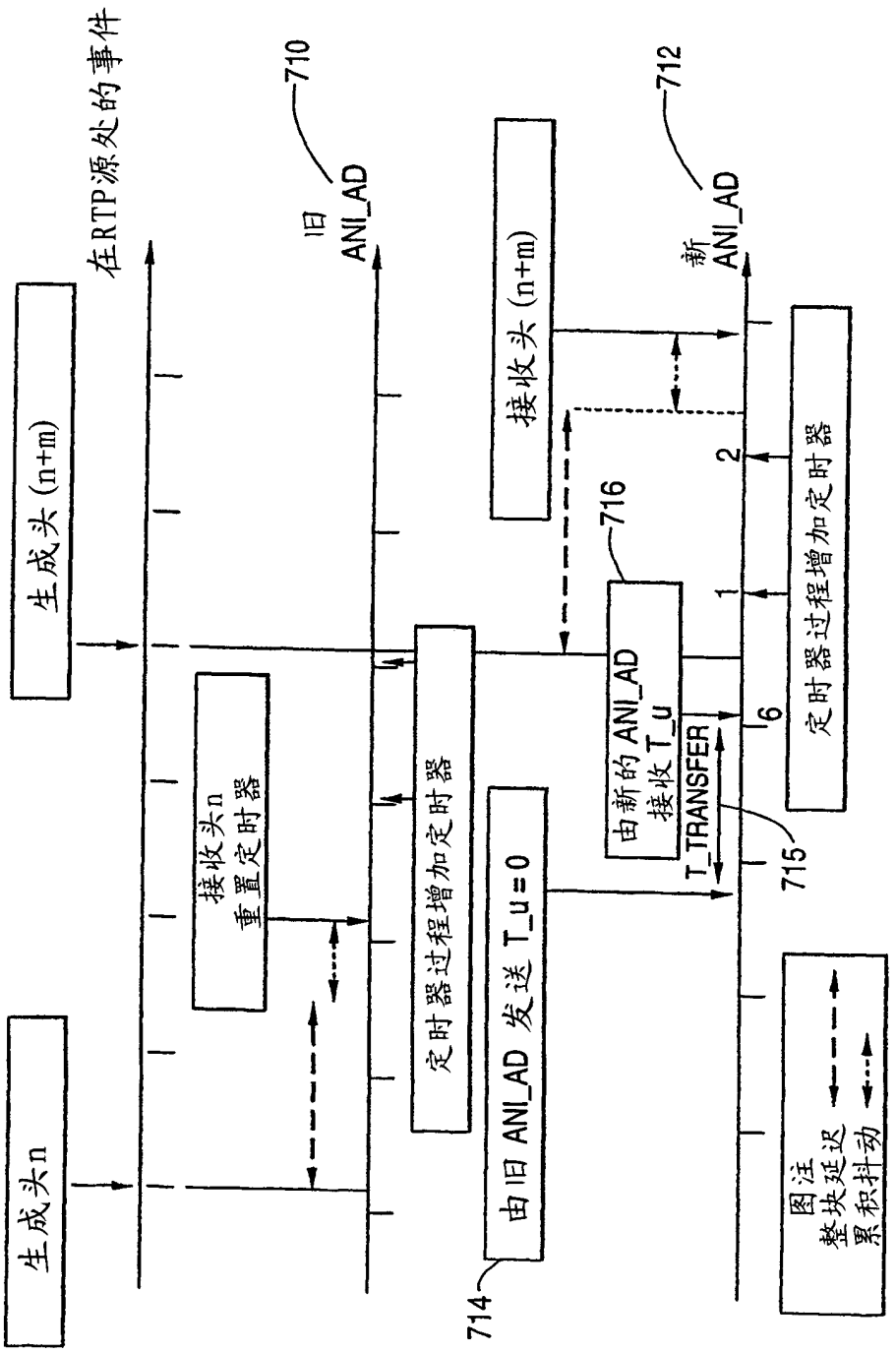

在RTP源处的事件
生成头n
生成头(n+m)
接收头n
重置定时器
定时器过程增加定时器
旧
ANI_AD
710
由旧 ANI_AD 发送 T_u=0
714
接收头(n+m)
716
由新的 ANI_AD
接收 T_u
T_TRANSFER
715
新
ANI_AD
712
定时器过程增加定时器
图注
整块延迟
累积抖动

图 7

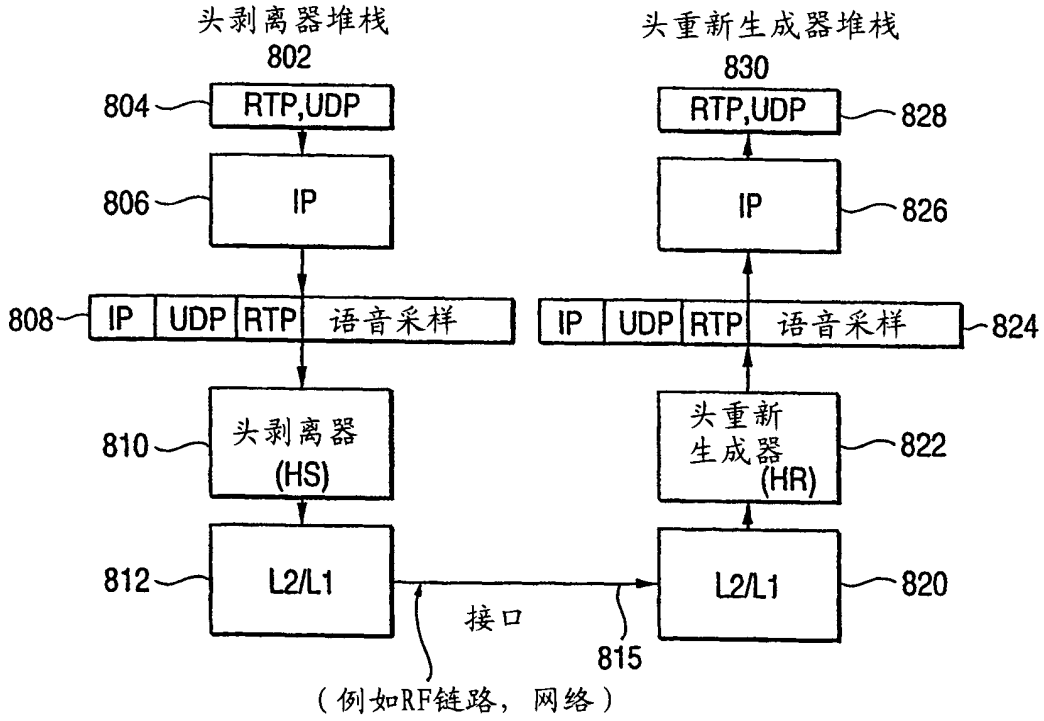

头剥离器堆栈
802
804
RTP,UDP
806
IP
808
IP
UDP
RTP
语音采样
810
头剥离器
(HS)
812
L2/L1
头重新生成器堆栈
830
RTP,UDP
828
IP
826
IP
UDP
RTP
语音采样
824
头重新
生成器
(HR)
822
L2/L1
820
接口
815
(例如RF链路，网络)

图 8

信息	内容	大小
Init_info(n)	完整IP/UDP/RTP头；n在RTP SN中隐含指定	至少大约40字节=320比特；在空中接口发送
String_init(n)	C_SN，C_TS（如果从一个串到下一个不保持定时），P_size（不太可能），TS_stride（不太可能）；n在C_SN中隐含指定	如果只有C_SN大约8比特，如果有C_SN和C_TS，为12比特
HO_init_u(n)	完整IP/UDP/RTP头，但是RTP TS由TS0_u, m_last_u, TS_stride_u, TS Timer_u, p_size_u代替；n在RTP SN隐含指定	稍大于完整头；在ANI_AD之间的无线网络上发送
HO_init_d(n)	与其代号一起的P_size_d 和 TS_stride_d	大小依赖于P_SIZE和TS_stride的编码；在ANI_AD之间的无线网络上发送
HO_sync_u(n)	C_SN, C_TS（可能）p_size_u, TS_stride_u; n在C_SN中隐含指定	如果只有C_SN，大约8比特，如果有C_SN和C_TS，为12比特
HO_sync_d(n)	C_SN, C_TS（可能）p_size_d, TS_stride_d; n在C_SN中隐含指定	如果只有C_SN，大约8比特，如果有C_SN和C_TS，为12比特
Ack(n)		几个比特

图 9

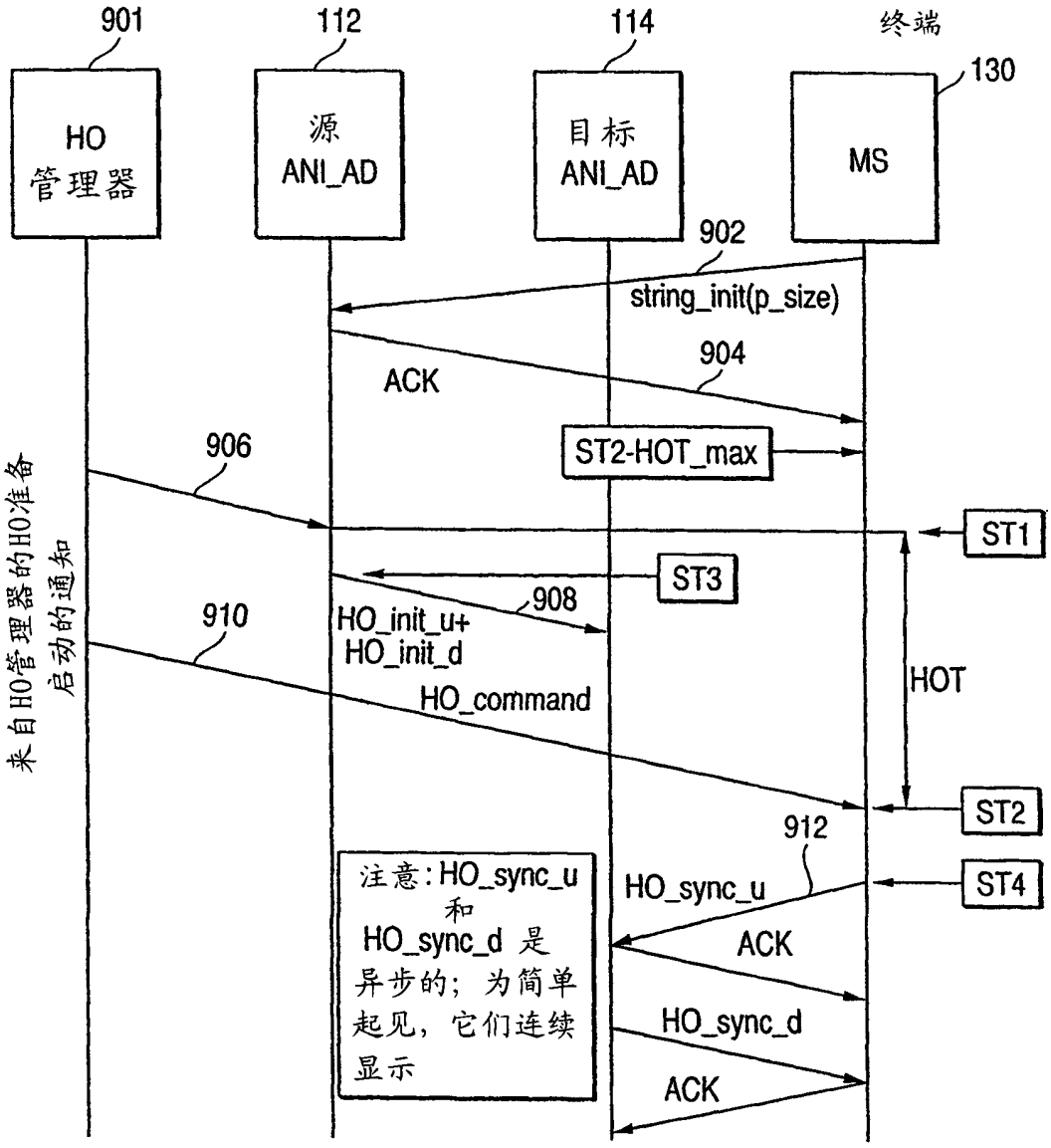

901
112
114
终端
130
HO
管理器
源
ANI_AD
目标
ANI_AD
MS
902
string_init(p_size)
904
ACK
906
ST2-HOT_max
来自HO管理器的HO准备
启动的通知
ST1
ST3
908
910
HO_init_u+
HO_init_d
HO_command
HOT
ST2
912
注意:HO_sync_u
和
HO_sync_d 是
异步的；为简单
起见，它们连续
显示
HO_sync_u
ST4
ACK
HO_sync_d
ACK

图 10

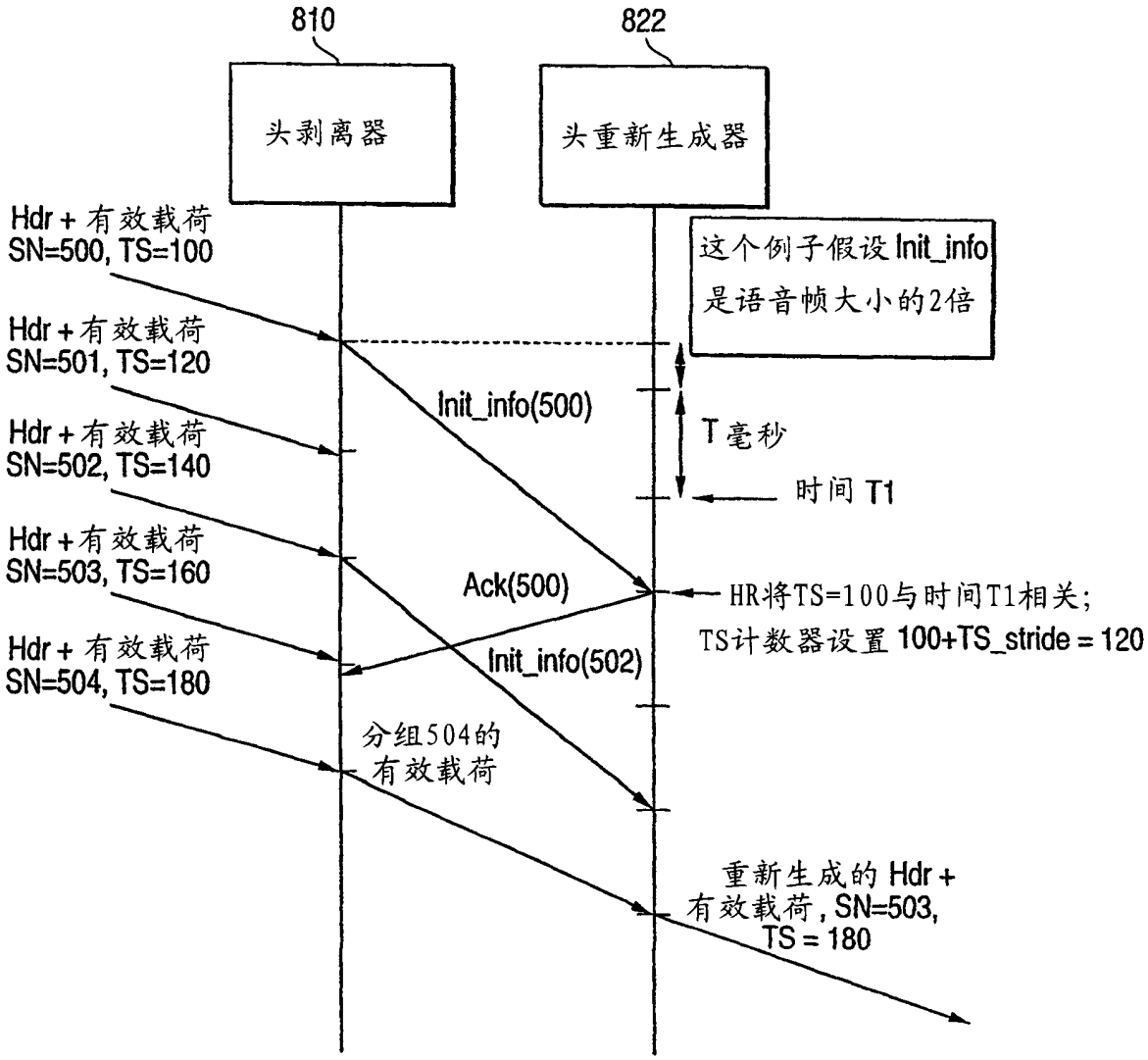

810
822
头剥离器
头重新生成器
Hdr + 有效载荷
SN=500, TS=100
这个例子假设 Init_info
是语音帧大小的2倍
Hdr + 有效载荷
SN=501, TS=120
Init_info(500)
T 毫秒
Hdr + 有效载荷
SN=502, TS=140
时间 T1
Hdr + 有效载荷
SN=503, TS=160
Ack(500)
HR将TS=100与时间T1相关;
TS计数器设置 100+TS_stride = 120
Hdr + 有效载荷
SN=504, TS=180
Init_info(502)
分组504的
有效载荷
重新生成的 Hdr +
有效载荷, SN=503,
TS = 180

图 11

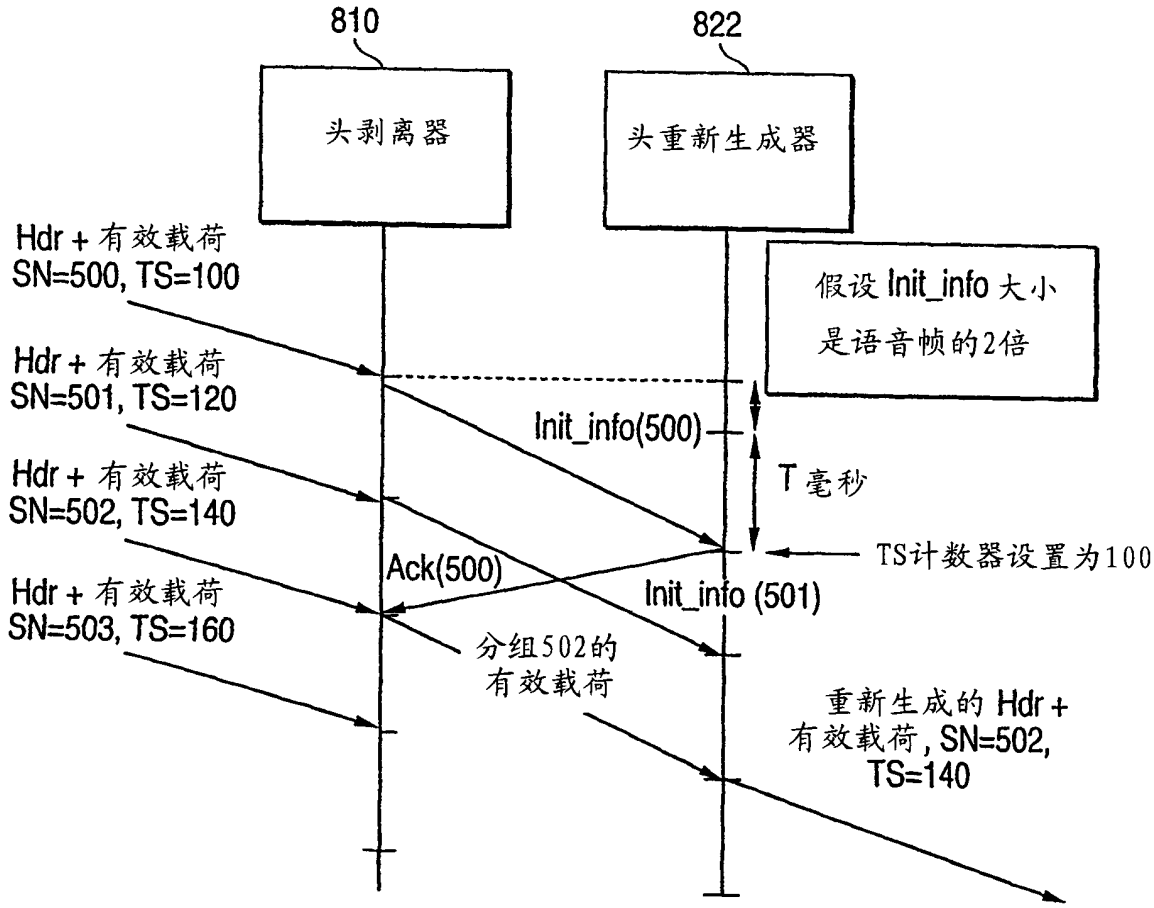
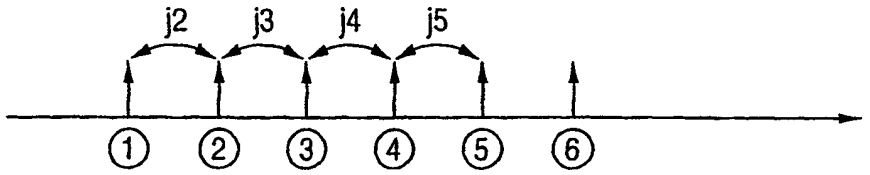

810
822
头剥离器
头重新生成器
Hdr + 有效载荷
SN=500, TS=100
假设 Init_info 大小
是语音帧的2倍
Hdr + 有效载荷
SN=501, TS=120
Init_info(500)
T 毫秒
Hdr + 有效载荷
SN=502, TS=140
TS计数器设置为100
Ack(500)
Hdr + 有效载荷
SN=503, TS=160
Init_info (501)
分组502的
有效载荷
重新生成的 Hdr +
有效载荷, SN=502,
TS=140

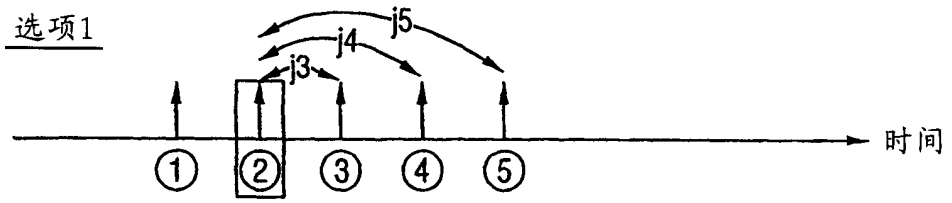
图 12



j2 = 关于1的分组2的抖动
j3 = 关于2的分组3的抖动
j4 = 关于3的分组4的抖动
j5 = 关于4的分组5的抖动 ← 在分组5完成

分组2的网络抖动 = j2
分组3的网络抖动 = j3
分组4的网络抖动 = j4, 等

图 13



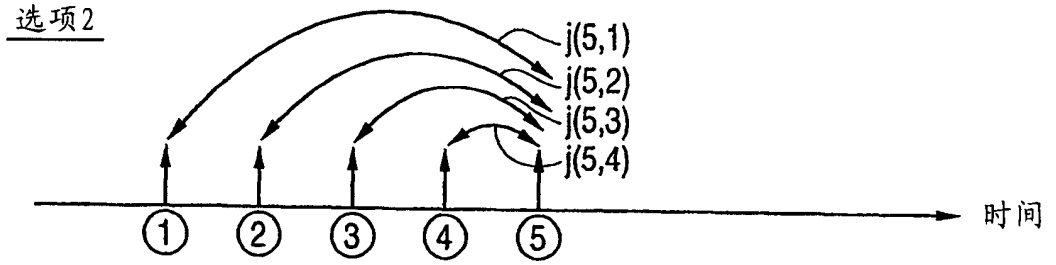
优选分组2

N_JITTER(3, 2) = j3 = 关于参考的分组3的抖动
假设 j3 = 2
N_JITTER(4, 2) = j4 = 关于参考的分组4的抖动
假设 j4 = 3
N_JITTER(5, 2) = j5 = 关于参考的分组5的抖动
假设 j5 = -1
在分组5之前: N_JITTER_MIN = 2 N_JITTER_MAX = 3
在分组5: N_JITTER_MIN = -1 N_JITTER_MAX = 3

N_JITTER_MAX_N_JITTER_MIN = 3 - (-1) = 4

对于分组5，最大网络抖动=4

图 14



计算关于1的分组5的抖动：j(5,1) = 2 = N_JITTER (5,1)
计算关于2的分组5的抖动：j(5,2) = 3 = N_JITTER (5,2)
计算关于3的分组5的抖动：j(5,3) = 4 = N_JITTER (5,3)
计算关于4的分组5的抖动：j(5,4) = 7 = N_JITTER (5,4)
对于分组5，最大网络抖动=7

图 15