(19)中华人民共和国国家知识产权局

(12)发明专利申请

(10)申请公布号 CN 104067594 A
(43)申请公布日 2014.09.24

(21)申请号 201280062414.6
(22)申请日 2012.11.01
(30)优先权数据
61/554,434 2011.11.01 US
61/589,855 2012.01.23 US
61/614,408 2012.03.22 US
61/645,562 2012.05.10 US
61/647,414 2012.05.15 US
13/563,590 2012.07.31 US
(85)PCT国际申请进入国家阶段日
2014.06.17
(86)PCT国际申请的申请数据
PCT/US2012/063115 2012.11.01
(87)PCT国际申请的公布数据
WO2013/067219 EN 2013.05.10
(71)申请人 高通股份有限公司
地址 美国加利福尼亚
(72)发明人 M·G·卢比 N·K·利昂
R·A·戈尔米 T·施托克哈默
(74)专利代理机构 永新专利商标代理有限公司 72002
代理人 张扬 王英
(51)Int.Cl.
H04L 29/08(2006.01)
H03M 13/37(2006.01)

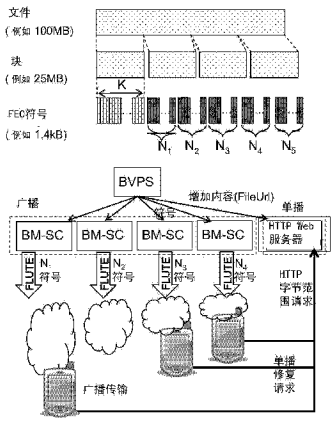
权利要求书4页 说明书53页 附图32页

(54)发明名称

在HTTP服务器之间分配源数据和修复数据的内容传送系统

(57)摘要

通过分组交换网络来传送数据对象，接收机接收具有足够信息的经编码的符号（诸如修复符号、广播或多播），以便基于需要什么源符号或子符号，或者丢失了什么源符号或子符号，形成针对所需要的其它符号的请求。这些请求可以是以单播或者请求方式来进行。请求和广播可以由不同的实体来执行。广播服务器可以生成和存储修复符号，而源服务器可以以源形式来存储内容。请求可以是以位置和长度开始的单播HTTP字节范围请求（例如，URL）。可以将请求与文件的起始位置进行对齐。接收机可以计算符号或者子符号在文件中的起始和结束字节位置，获得传统的HTTP服务器可用于文件修复的指示。当来自多个接收机的字节范围请求重叠时，修复服务器可以请求修复数据的广播。



CN 104067594 A

1. 一种使用耦合到分组交换网络的电子设备或系统来接收一个或多个数据对象的方法，其中，所述一个或多个数据对象的源数据是通过分组中的经编码的符号表示的，使得所述源数据可至少近似地根据所述经编码的符号恢复的，所述方法包括：

a) 通过广播信道来接收经编码的符号，其中，经编码的符号的值是至少部分地根据源符号的值来导出的；

b) 确定将内容恢复到期望的完整程度所需要的额外符号的数量；

c) 确定一个或多个文件的一个或多个字节范围的相应集合，其中，所述相应集合与恢复所述内容所需要的额外符号相对应；

d) 使用指向服务器的、指定所述一个或多个字节范围的相应集合的请求，生成针对至少所述数量的额外符号的请求；

e) 发送所述请求；

f) 接收所请求的额外符号中的至少一些；以及

g) 在恢复所述内容时，使用所接收的额外符号结合通过所述广播信道接收的所述经编码的符号。

2. 根据权利要求 1 所述的方法，其中，所述期望的完整程度是所述内容的完全恢复。

3. 根据权利要求 1 所述的方法，其中，所述请求是 HTTP 字节范围请求，并且多个请求中的至少一个请求是 HTTP 字节范围请求、是针对于非连续字节范围。

4. 根据权利要求 1 所述的方法，其中，所述请求是针对于额外符号的连续集合。

5. 根据权利要求 4 所述的方法，其中，所述额外符号的连续集合以源符号起始，所述源符号是针对远程地接收内容的多个设备中的每个设备的相同的起始符号。

6. 根据权利要求 1 所述的方法，其中，所述请求是单播 HTTP 字节范围请求。

7. 根据权利要求 1 所述的方法，其中，所述额外符号是源符号。

8. 根据权利要求 1 所述的方法，其中，所述额外符号是源符号和修复符号。

9. 根据权利要求 1 所述的方法，其中，所述经编码的符号全部是修复符号，并且所述额外符号全部是源符号。

10. 根据权利要求 1 所述的方法，其中，所述源数据包括源符号的源块的有组织集合，并且被进一步组织成源子块和源子符号的集合。

11. 根据权利要求 10 所述的方法，其中，所述请求是针对于子符号的字节范围。

12. 根据权利要求 10 所述的方法，其中，所述请求是针对于额外子符号的连续集合。

13. 根据权利要求 12 所述的方法，其中，所述连续集合以源子符号起始，所述源子符号是针对远程地接收内容的多个设备中的每个设备的相同的起始子符号。

14. 根据权利要求 10 所述的方法，其中，所述请求是针对于源子符号。

15. 根据权利要求 10 所述的方法，其中，所述请求是针对于源子符号和修复子符号。

16. 一种呈现远程接收的内容的设备，包括：

电子接口，其用于从广播信道接收数据；

用于通过所述广播信道来接收经编码的符号的逻辑单元，其中，经编码的符号的值是至少部分地根据与要远程接收的所述内容相对应的源符号的值来导出的；

用于确定将内容恢复到期望的完整程度所需要的额外符号的数量的逻辑单元；

用于确定一个或多个文件的一个或多个字节范围的相应集合的逻辑单元，其中，所述

相应集合与恢复所述内容所需要的额外符号相对应；

用于使用指向服务器的、指定所述一个或多个字节范围的相应集合的请求，生成针对至少所述数量的额外符号的请求的逻辑单元；

用于发送所述请求的接口；

用于接收所请求的额外符号中的至少一些的逻辑单元；以及

解码器，其在恢复所述内容时，使用所接收的额外符号结合通过所述广播信道接收的所述经编码的符号。

17. 根据权利要求 16 所述的设备，其中，所述期望的完整程度是所述内容的完全恢复。

18. 根据权利要求 16 所述的设备，其中，所述请求是 HTTP 字节范围请求，并且多个请求中的至少一个请求是 HTTP 字节范围请求、是针对于非连续字节范围。

19. 根据权利要求 16 所述的设备，其中，所述请求是针对于额外符号的连续集合。

20. 根据权利要求 19 所述的设备，其中，所述额外符号的连续集合以源符号起始，所述源符号是针对远程地接收内容的多个设备中的每个设备的相同的起始符号。

21. 根据权利要求 16 所述的设备，其中，所述额外符号是源符号。

22. 根据权利要求 16 所述的设备，其中，所述额外符号是源符号和修复符号。

23. 根据权利要求 16 所述的设备，其中，所述经编码的符号全部是修复符号，并且所述额外符号全部是源符号。

24. 根据权利要求 16 所述的设备，其中，所述源数据包括源符号的源块的有组织集合，并且被进一步组织成源子块和源子符号的集合。

25. 根据权利要求 16 所述的设备，其中，所述请求是针对于子符号的字节范围。

26. 根据权利要求 16 所述的设备，其中，所述请求是针对于额外子符号的连续集合。

27. 根据权利要求 26 所述的设备，其中，所述连续集合以源子符号起始，所述源子符号是针对远程地接收内容的多个设备中的每个设备的相同的起始子符号。

28. 根据权利要求 16 所述的设备，其中，所述请求是针对于源子符号。

29. 根据权利要求 16 所述的设备，其中，所述请求是针对于源子符号和修复子符号。

30. 根据权利要求 16 所述的设备，还包括：

用于从所述网络接收 HTTP 字节范围请求是否被文件服务器支持的指示的逻辑单元。

31. 根据权利要求 16 所述的设备，还包括：

用于从所述网络接收服务器支持哪些格式的指示的逻辑单元。

32. 根据权利要求 31 所述的设备，还包括：

用于从所述网络接收所述设备应当尝试从向其进行请求的服务器的优先次序的指示的逻辑单元。

33. 根据权利要求 16 所述的设备，还包括：

用于当所述设备支持一种以上的请求格式时，接收指示要使用哪种请求格式的偏好选择或要求的逻辑单元。

34. 根据权利要求 16 所述的设备，还包括：

用于接收仅向所述设备广播修复符号的指示的逻辑单元。

35. 根据权利要求 34 所述的设备，其中，当所述设备接收到所述指示时，所述设备被限制于始终使用针对额外数据的前缀请求。

36. 根据权利要求 16 所述的设备，还包括：

用于接收要求所述设备使用针对额外数据的前缀请求的指示的逻辑单元。

37. 根据权利要求 16 所述的设备，还包括：

用于计算符号或者子符号的文件中的起始字节位置，以及所述符号或者所述子符号的所述文件中的结束字节位置的逻辑单元。

38. 一种通过分组交换网络从电子设备或系统传送一个或多个数据对象的方法，其中，所述一个或多个数据对象的源数据是通过分组中的经编码的符号表示的，使得所述源数据可至少近似地根据所述经编码的符号恢复的，所述方法包括：

a) 如果所述源数据尚未被组织成源符号的源块，则将所述源数据组织成源符号的源块的有组织集合；

b) 生成经编码的符号，其中，经编码的符号的值是至少部分地根据源符号的值来导出的，并且其中，是修复符号的经编码的符号具有用于进行与源块相对应的生成的范围；

c) 以广播方式，将修复符号和 / 或源符号作为经编码的符号提供给目的设备，其中，所述广播为目的设备确定所广播的数据的符号和块结构提供足够的信息，

其中，所述足够的信息至少包括：将所述源符号的源块的有组织集合中的源符号映射到可用于向支持文件请求与字节范围请求的服务器进行字节范围请求的字节范围的信息。

39. 根据权利要求 38 所述的方法，还包括：

d) 在所述目的设备处，确定哪些源符号是不可根据所接收的广播符号来解码的；

e) 在所述目的设备处，确定可使用具有字节范围修饰符的文件请求来请求的文件的一个或多个字节范围的集合；以及

f) 使用所述字节范围修饰符进行所述文件请求。

40. 根据权利要求 38 所述的方法，其中，所述文件请求是包括 URL 和字节范围的 HTTP 客户端请求。

41. 根据权利要求 40 所述的方法，其中，所述字节范围在被请求的文件的起始处开始。

42. 根据权利要求 38 所述的方法，还包括：

d) 根据接收的请求，确定所接收的请求是否是响应于用于针对源自于广播过程的丢失的修复符号的恢复的修复过程；

e) 根据所接收的请求，确定所接收的请求是否是响应于来自与广播过程无关的客户端的直接请求；以及

f) 记录确定结果。

43. 根据权利要求 42 所述的方法，其中，通过读取在所接收的请求中使用的 URL 的全部或者一部分，来执行确定。

44. 根据权利要求 38 所述的方法，其中，所述源符号的源块的有组织集合被进一步组织成源子块和源子符号的集合。

45. 一种使用耦合到分组交换网络的电子设备或系统来接收一个或多个数据对象的非暂时性计算机程序产品，其中，所述一个或多个数据对象的源数据是通过分组中的经编码的符号表示的，使得所述源数据可至少近似地根据所述经编码的符号恢复的，所述产品包括：

a) 用于通过广播信道来接收经编码的符号的程序代码，其中，经编码的符号的值是至

少部分地根据源符号的值来导出的；

b) 用于确定将内容恢复到期望的完整程度所需要的额外符号的数量的程序代码；

c) 用于确定一个或多个文件的一个或多个字节范围的相应集合的程序代码，其中，所述相应集合与恢复所述内容所需要的额外符号相对应；

d) 用于使用指向服务器的、指定所述一个或多个字节范围的相应集合的请求，生成针对至少所述数量的额外符号的请求的程序代码；

e) 用于向所述服务器发送所述请求的程序代码；

f) 用于接收所请求的额外符号中的至少一些的程序代码；以及

g) 用于使用所接收的额外符号结合通过所述广播信道接收的所述经编码的符号，来恢复所述内容的程序代码。

46. 根据权利要求 45 所述的产品，其中，所述期望的完整程度是所述内容的完全恢复。

47. 根据权利要求 45 所述的产品，其中，所述请求是 HTTP 字节范围请求，并且多个请求中的至少一个请求是 HTTP 字节范围请求、是针对于非连续字节范围。

48. 根据权利要求 45 所述的产品，其中，所述请求是针对于额外符号的连续集合。

49. 根据权利要求 48 所述的产品，其中，所述额外符号的连续集合以源符号起始，所述源符号是针对远程地接收内容的多个设备中的每个设备的相同的起始符号。

50. 根据权利要求 45 所述的产品，其中，所述经编码的符号全部是修复符号，并且所述额外符号全部是源符号。

51. 根据权利要求 45 所述的产品，其中，所述源数据包括源符号的源块的有组织集合，并且被进一步组织成源子块和源子符号的集合。

52. 根据权利要求 51 所述的产品，其中，所述请求是针对于子符号的字节范围。

53. 根据权利要求 53 所述的产品，其中，所述连续集合以源子符号起始，所述源子符号是针对远程地接收内容的多个设备中的每个设备的相同的起始子符号。

在 HTTP 服务器之间分配源数据和修复数据的内容传送系统

[0001] 相关申请的交叉引用

[0002] 本申请要求以下专利申请的优先权，并是其非临时申请：2011 年 11 月 1 日提交的、题目为“Unicast Repair Service and Server Augmenting Broadcast File Delivery System”的美国临时专利申请 No.61/554,434（代理案号为 No.121519P1）；2012 年 1 月 23 日提交的、题目为“Unicast Repair Service and Server Augmenting Broadcast File Delivery System”的临时专利申请 No.61/589,855（代理案号为 No.121519P2）；2012 年 3 月 22 日提交的、题目为“Unicast Repair Service and Server Augmenting Broadcast File Delivery System”的临时专利申请 No.61/614,408（代理案号为 No.121519P3）；2012 年 5 月 10 日提交的、题目为“Content Delivery System with Allocation of Source Data and Repair Data Among HTTP Servers”的美国临时专利申请 No.61/645,562（代理案号为 No.121519P4）；以及 2012 年 5 月 15 日提交的、题目为“Content Delivery System with Allocation of Source Data and Repair Data Among HTTP Servers”的美国临时专利申请 No.61/647,414（代理案号为 No.121519P5），故出于所有的目的以引用方式将上述临时申请的全部内容并入本文。

[0003] 本公开内容可以涉及以下共同转让的专利或申请，故出于所有目的，以引用方式将这些专利或者申请中的每一个的全部内容并入本文：

[0004] 1）授予 Michael G.Luby 的、题目为“Information Additive Code Generator and Decoder for Communication Systems”的美国专利 No.6,307,487（下文称为“Luby I”）；

[0005] 2）授予 Michael G.Luby 的、题目为“Information Additive Group Code Generator and Decoder for Communication Systems”的美国专利 No.6,320,520（下文称为“Luby II”）；

[0006] 3）授予 M.Amin Shokrollahi 的、题目为“Systems and Processes for Decoding a Chain Reaction Code Through Inactivation”的美国专利 No.6,856,263（下文称为“Shokrollahi I”）；

[0007] 4）授予 M.Amin Shokrollahi 的、题目为“Systematic Encoding and Decoding of Chain Reaction Codes”(hereinafter “Shokrollahi II”的美国专利 No.6,909,383（下文称为“Shokrollahi II”）；

[0008] 5）授予 M.Amin Shokrollahi 的、题目为“Multi-Stage Code Generator and Decoder for Communication Systems”的美国专利 No.7,068,729（下文称为“Shokrollahi III”）；

[0009] 6）授予 Michael G.Luby、M.Amin Shokrollahi 和 Mark Watson 的、题目为“File Download and Streaming System”的美国专利 No.7,418,651（下文称为“Luby III”）；

[0010] 7）专利人为 Michael G.Luby 和 M.Amin Shokrollahi 的、题目为“In-Place Transformations with Applications to Encoding and Decoding Various Classes of Codes”的美国专利公开 No.2006/0280254（下文称为“Luby IV”）；

[0011] 8) 专利人为 M.Amin Shokrollahi 的、题目为“Multiple-Field Based Code Generator and Decoder for Communications Systems”的美国专利公开 No.2007/0195894(下文称为“Shokrollahi IV”);

[0012] 9) 以 M.Amin Shokrollahi 等人的名义在 2009 年 10 月 23 日提交的、题目为“Method and Apparatus Employing FEC Codes with Permanent Inactivation of Symbols for Encoding and Decoding Processes”的美国专利申请 No.12/604,773(下文称为“Shokrollahi V”);

[0013] 10) 以 Michael G.Luby 的名义在 2009 年 8 月 19 日提交的、题目为“Methods and Apparatus Employing FEC Codes with Permanent Inactivation of Symbols for Encoding and Decoding Processes”的美国临时申请 No.61/235,285(下文称为“Luby V”);

[0014] 11) 以 Michael G.Luby、M.Amin Shokrollahi 的名义在 2010 年 8 月 18 日提交的、题目为“Methods and Apparatus Employing FEC Codes with Permanent Inactivation of Symbols for Encoding and Decoding Processes”的美国专利申请 No.12/859,161(下文称为“Shokrollahi VI”);

[0015] 12) 以 Michael G.Luby、Thadi M.Nagaraj 的名义在 2011 年 8 月 9 日提交的、题目为“Broadcast Multimedia Storage and Access Using Page Maps when Asymmetric Memory is Used”的美国专利申请 No.13/206,418(下文称为“Luby VI”);

[0016] 13) 以 Michael G.Luby、Nikolai Leung、Ralph Gholmieh、Thomas Stockhammer 的名义在 2012 年 5 月 10 日提交的、题目为“Content Delivery System with Allocation of Source Data and Repair Data Among HTTP Servers”的美国专利申请 No.61/645,562(下文称为“Luby VII”);

[0017] 参考文献

[0018] 出于所有目的,以引用方式将下列参考文献中的每一个的全部内容并入本文:

[0019] [Albanese96] 或 [PET],“Priority Encoding Transmission”, Andres Albanese、Johannes Blomer、Jeff Edmonds、Michael Luby 和 Madhu Sudan, IEEE Transactions on Information Theory, vol.42, no.6(1996 年 11 月);

[0020] [Albanese97] 或 [PET-Patent],授予 A.Albanese, M.Luby, J.Blomer, J.Edmonds 的、题目为“System for Packetizing Data Encoded Corresponding to Priority Levels Where Reconstructed Data Corresponds to Fractionalized Priority Level and Received Fractionalized Packets”的美国专利 No.5,617,541(1997 年 4 月 1 日发布);

[0021] [ALC], Luby, M.Watson, M.Vicisano, L.,“Asynchronous Layered Coding(ALC) Protocol Instantiation”, IETF RFC5775(2010 年 4 月);

[0022] [FEC BB], Watson, M., Luby, M., and L.Vicisano, “Forward Error Correction(FEC)Building Block”, IETF RFC5052(2007 年 8 月);

[0023] [FLUTE], Paila, T., Luby, M., Lehtonen, R., Roca, V., Walsh, R., “FLUTE--File Delivery over Unidirectional Transport”, IETF RFC3926(2004 年 10 月);

[0024] [LCT], Luby, M., Watson, M., Vicisano, L., “Layered Coding Transport(LCT) Building Block”, IETF RFC5651(2009 年 10 月);

[0025] [Luby2007] 或 [Raptor-RFC-5053], M.Luby, A.Shokrollahi, M.Watson, T.

Stockhammer, “Raptor Forward Error Correction Scheme for Object Delivery”, IETF RFC5053(2007 年 9 月）；

[0026] [Luby2002], Luby, M., Vicisano, L., Gemmell, J., Rizzo, L., Handley, M., 和 J. Crowcroft, “The Use of Forward Error Correction (FEC) in Reliable Multicast”, IETF RFC3453(2002 年 12 月）；

[0027] [Matsuoka] 或 [LDPC-Extensions], “Low-Density Parity-Check Code Extensions Applied for Broadcast-Communication Integrated Content Delivery”, Hosei Matsuoka, Akira Yamada 和 Tomoyuki Ohya, Research Laboratories, NTT DOCOMO, Inc., 3-6, Hikari-No-Oka, Yokosuka, Kanagawa, 239-8536, 日本；以及

[0028] [RaptorQ-RFC-6330], M. Luby, A. Shokrollahi, M. Watson, T. Stockhammer, L. Minder, “RaptorQ Forward Error Correction Scheme for Object Delivery”, IETF RFC6330, Reliable Multicast Transport(2011 年 8 月）；

[0029] [Roca] 或 [LDPC-RFC-5170], V. Roca, C. Neumann, D. Furodet, “Low Density Parity Check (LDPC) Staircase and Triangle Forward Error Correction (FEC) Schemes”, IETF RFC5170(2008 年 6 月）。

技术领域

[0030] 概括地说，本发明涉及通信系统中对数据的编码和解码，更具体地说，涉及对数据进行编码和解码，以便以高效的方式考虑所通信的数据中的错误和差距以及处理不同的文件传送方法的通信系统。

背景技术

[0031] 用于通过通信信道在发送方和接收方之间传输文件的技术是许多文献的主题。优选地，接收方期望以某种确定的程度，接收由发送方通过信道所发送的数据的准确拷贝。在信道不具有完美的保真度的情况下（这涵盖几乎所有物理可实现的系统），一个考虑的问题是如何处理传输中丢失或乱码的数据。与损坏的数据（错误）相比，丢失的数据（擦除）通常更容易处理，这是由于接收方无法总是告知损坏的数据是错误接收的数据。已开发了许多纠错编码来纠正擦除和 / 或错误。

[0032] 典型地，基于与通过其来发送数据的信道的失真有关的某些信息以及所发送的数据的本质来选择使用的具体编码。例如，在已知信道具有较长时段的失真的情况下，突发错误编码可能最适合于这种应用。而在失真仅是短暂的情况下，预计有稀少的错误，则简单的奇偶校验码可能是最佳的。

[0033] 如本文中所使用的“源数据”指在一个或多个发送方处可获得以及使用接收机获得（通过从具有或者不具有错误和 / 或擦除等的发送的序列中恢复）的数据。如本文中所使用的“经编码的数据”指被传送的并且可以用于恢复或者获得源数据的数据。在简单的情况下，经编码的数据是源数据的拷贝，但如果所接收的经编码的数据与发送的经编码的数据不相同（由于差错和 / 或擦除），则在该简单情况中，由于缺少关于源数据的额外数据，因此该源数据可能无法被完整地恢复。传输可以通过空间或者时间。在更复杂情况下，经编码的数据是基于源数据变换生成的，并且是从一个或多个发送方发送给接收方的。如果

发现源数据是经编码的数据的一部分，则可以将编码称为“系统化的”。在系统化编码的简单示例中，将关于源数据的冗余信息附加到该源数据的尾部以形成经编码的数据。

[0034] 此外，如本文中所使用的“输入数据”指出现在FEC（前向纠错）编码器装置或者FEC编码器模块、组件、步骤等的输入端的数据，而“输出数据”指出现在FEC编码器的输出端的数据。相应地，期望输出数据出现在FEC解码器的输入端，并且期望FEC解码器基于其处理的输出数据来输出该输入数据（或其对应数据）。在一些情况下，输入数据是（或者包括）源数据，而在一些情况下，输出数据是（或者包括）经编码的数据。例如，如果在FEC编码器的输入端之前没有进行处理，则输入数据是源数据。然而，在一些情况下，将源数据处理成不同的形式（例如，静态编码器、逆编码器或者另一种处理），以生成提供给FEC编码器的中间数据而不是源数据。

[0035] 在一些情况下，发送方设备或者发送方程序代码可以包括一个以上的FEC编码器，即，在一系列的多个FEC编码器中将源数据转换成经编码的数据。类似地，在接收机处，可以存在一个以上的FEC解码器，以用于根据所接收的经编码的数据来生成源数据。

[0036] 数据可以被认为是划分成了符号。编码器是根据源符号或输入符号的序列来生成经编码的符号或输出符号的计算机系统、设备、电子电路等，而解码器是根据接收的或恢复的经编码的符号或输出符号来恢复源符号或输入符号的序列的对应方。编码器和解码器在时间和/或空间上被信道间隔开，并且任何接收的经编码的符号可能不是与相应发送的经编码的符号完全一样，其可能没有如其被发送地以完全相同的序列被接收。符号的“大小”可以以比特来测量，无论该符号是否实际被分解成一个比特流，其中，当一个符号是从2^M个符号的字母表中选出的时，该符号具有M个比特的大小。在本文的很多示例中，以八位字节来测量符号，并且代码可能跨越256种可能的字段（在每一个八位字节中，存在256种可能的8比特模式），但应当理解的是，可以使用不同单位的数据测量，并且公知的是以各种方式来测量数据。在一般文献中，术语“字节”有时与术语“八位字节”互换使用，用以指示8比特值，虽然在一些上下文中，“字节”指示X比特值，其中X不等于8（例如，X＝7）。在本文中，术语“八位字节”和“字节”可以互换使用。除非另外指出，否则本文中的示例并不限于每一符号具有特定整数的比特或者非整数的比特。

[0037] Luby I描述了用于以计算高效、存储高效和带宽高效的方式，来解决纠错的编码（例如，链式反应码）的使用。链式反应编码器所产生的经编码的符号的一种属性在于：只要接收到足够的经编码的符号，接收机就能够恢复原始文件。具体而言，为了以较高的概率来恢复原始的K个源符号，接收机需要近似K+A个经编码的符号。

[0038] 针对给定情形的“绝对接收开销”使用值A来表示，而“相对接收开销”可以计算成比率A/K。绝对接收开销是指除了信息理论的最小数据数量之外，还需要接收多少额外数据的测量值，其取决于解码器的可靠性，根据源符号的数量(K)进行变化。类似地，相对接收开销(A/K)是指：相对于要恢复的源数据的大小，除了信息理论的最小数据数量之外，还需要接收多少额外数据的测量值，其也取决于解码器的可靠性，并根据源符号的数量(K)进行变化。

[0039] 对于基于分组的网络上的通信来说，链式反应码是非常有用的。但是，其有时是计算量相当地大。如果在使用链式反应码或者另一种无速率码进行编码的动态编码器之前，使用静态编码器对源符号进行编码，则解码器可能能够更频繁或者更容易地进行解码。例

如，在 Shokrollahi I 中示出了这些解码器。在其所示出的示例中，源符号是静态编码器的输入符号，静态编码器产生用于动态编码器的输入符号的输出符号，动态编码器产生作为经编码的符号的输出符号，其中该动态编码器是无速率编码器，无速率编码器可以按照相对于输入符号的数量，不具有固定的速率的数量，来生成多个输出符号。静态编码器可以包括一个以上的固定速率编码器。例如，静态编码器可以包括 Hamming 编码器、低密度奇偶校验（“LDPC”）编码器、高密度奇偶校验（“HDPC”）编码器等。

[0040] 链式反应码具有下面的属性：由于一些符号是解码器根据所接收的符号来恢复的，因此这些符号能够用于恢复额外符号，这些额外符号随后可以用于恢复更多的符号。优选地，在解码器处求解的符号的链式反应可以继续进行，使得在使用完接收符号池之前，恢复出所有期望的符号。优选地，执行链式反应编码和解码处理的计算复杂度是较低的。所期望的符号可以是对所有原始的源符号进行恢复所需要的所有符号，或者某种期望的完整程度（其与所有原始源符号相比更低）。

[0041] 解码器处的恢复过程可以涉及：确定接收到哪些符号、生成用于将这些原始输入符号映射到接收的那些经编码的符号的矩阵、随后对该矩阵进行求逆、执行该逆矩阵和接收的经编码的符号的向量的矩阵相乘。在典型的系统中，这种过程的强力实施可能消耗过多的计算工作和存储器要求。当然，对于特定的接收的经编码的符号集来说，可能不能恢复出所有的原始输入符号，即使可以恢复，可能也需要非常大的计算量来计算出该结果。

[0042] 前向纠错（“FEC”）对象传输信息（“OTI”）或“FEC OTI”

[0043] 基于接收机接收的 FEC OTI（或者能够推断），该接收机可以确定文件传输的源块和子块结构。在 [Raptor-RFC-5053] 和 [RaptorQ-RFC-6330] 中，FEC 有效载荷 ID 是 (SBN,ESI)，其中在 [Raptor-RFC-5053] 中，源块编号 (SBN) 是 16 比特，经编码的符号 ID(ESI) 是 16 比特，而在 [RaptorQ-RFC-6330] 中，SBN 是 8 比特，ESI 是 24 比特，如本申请的图 1 中所示出的。这种 FEC 有效载荷 ID 格式的一种缺点在于：必须预先确定用于向 SBN 和向 ESI 分配的 FEC 有效载荷 ID 的比特的数量，有时很难确定足够用于所有文件传输参数的适当混合。

[0044] 例如，当使用 [Raptor-RFC-5053] 时，仅具有 $2^{16} = 65,536$ 个可用的 ESI 在一些情形下可能是受限的，这是由于在一些情况下，可能存在具有 8,192 个源符号的源块，因此，经编码的符号的数量只是 8 个更大的一个因子，对于可以使用的可能编码速率进行限制，以便在该情况下，被限于不下降到低于 1/8。在该示例中，可以是有 $2^{16} = 65,536$ 个可用的源块，其可以是大于所使用的（例如，每一个具有 1,024 字节的 8,192 个源符号），能够支持的文件的大小是 524GB，在很多应用中，与所需要的相比，其是幅度的两倍量级。

[0045] 再举一个例子，当使用 [RaptorQ-RFC-6330] 时，仅具有 $2^8 = 256$ 个可用的 SBN，在一些情形下可能是受限的，这是由于对于 4GB 文件，如果将每一个源块限制于 8MB（其可以是这种情况：如果最大子块大小是 256KB，最小子符号大小是 32 个字节，则符号大小是 1,024 个字节），则将源块的数量限制于 256，则将文件大小限制为 2GB。在该示例中，其可以是，与所使用的相比，可用的 $2^{24} = 16,777,216$ 个可能的经编码的符号更多，例如，具有 8,192 个源符号，可能的经编码的符号的数量更大 2,048 倍，在一些应用中，不需要使用这些经编码的符号。

[0046] 另一种期望的属性是提供用于在文件的不同部分之间，对编码的传输划分优先次

序的能力（其有时称为不等错误保护（“UEP”）。例如，可能期望的是，与剩余的 90％相比，对文件的前 10％进行更强的防止分组丢失的保护。例如，[LDPC-Extensions] 描述了如何对 [LDPC-RFC-5170] 进行扩展，以便提供 UEP 的支持。在该情况下，对实际的 FEC 编码自身进行修改，以便针对文件的不同部分，提供不同程度的奇偶校验。但是，该方法存在着一些缺陷。例如，可能不期望必须对 FEC 编码自身进行修改来提供 UEP，这是由于其使实现和测试 FEC 编码自身变得复杂。此外，作为 [LDPC-Extensions] 的图 6 中所示的结果，就针对文件的不同部分所提供的抵御丢包而言，该方法所获得的性能远远不是最佳的。

[0047] 用于提供 UEP 文件传输能力的一种方式（如 [PET] 和 [PET-Patent] 中所描述的），是根据文件的不同部分的优先级和大小，为该文件的不同部分分配每一个分组的不同部分。但是，关注的是如何以下面的方式对这些 UEP 方法进行综合：独立于文件的其它部分，将文件的每一个不同部分划分到一些源块和子块，例如以便支持该文件的每一个部分的小内存解码，同时在每一个分组中提供 FEC 有效载荷 ID，这使接收机能确定在每一个分组中包含该文件的每一个部分的什么符号。使用格式 (SBN，ESI) 的 FEC 有效载荷 ID 来支持这种能力是非常困难的，至于文件的每一个部分，用于文件的第一部分的分组中的符号的相应 SBN 和 ESI，可以与用于该文件的第二部分的分组中的符号的 SBN 和 ESI 不相同。

[0048] 在一些情况下，需要专用的服务器，使用更通用、传统的硬件系统来实现、支持和维持该专用服务器，以支持内容传送是更昂贵的。因此，期望具有实现起来不太复杂、用于传送内容和修复符号的方法。

发明内容

[0049] 在文件传输方法和装置的实施例中，将内容提供给文件传输系统，使得表示该内容的源块和符号能以单播方式可用，以广播或者多播方式来提供修复块和符号。可以提供其它途径和冗余。可以通过一个实体来实现源块和符号的单播供应，在具有或者不具有第一实体执行的特定处理的情况下，广播或多播部分由另一个实体来提供。

[0050] 在特定的实施例中，使一组内容（数据、图像、音频、视频等）可用于由很大数量的用户进行操作或者使用的很大数量的终端用户设备（“UE”），以便向这些用户呈现该内容，通常在给定的用户异步地请求该内容之后不久，就开始向该用户进行呈现，并优选地继续该呈现直到其结束为止（除非用户终止其）。在该实施例中，将内容以源形式存储在一个或多个单播服务器，用于该内容的修复符号在广播服务器处生成和存储，并从其向多个 UE 进行广播或多播。替代地，不进行存储，只要生成了修复符号，就对该内容进行广播。随后，UE 从广播服务器接收一些数量的修复符号，确定在该过程中是否有修复符号丢失，确定（至少近似地）为了根据额外符号和所接收的修复符号来完全地恢复该内容，而所需要的多个另外符号，随后从单播服务器请求该数量的符号。再举一个例子，从广播服务器广播或者多播至少一些源符号，确定（至少近似地）为了完全地恢复该内容的一部分（该 UE 将要播放的部分）所需要的多个另外符号，随后从单播服务器请求该数量的符号或者子符号（其可以是修复子符号或者修复符号）。

[0051] 在一些情况下，针对单播服务器的请求具有 HTTP 字节范围请求的形式。当 HTTP 字节范围请求指定文件的 URL、该文件中的起始位置和该请求的长度（例如，请求的符号的数量、或者请求的子符号的数量、或者与一组连续的子符号或符号相对应的字节范围请求）

时，可以对这些请求进行配置，使得这些请求中的全部或者大部分将该文件的初始位置用作起始位置。这将允许下游高速缓存更频繁地履行请求，这是因为一旦其关于给定的文件，对最大的请求进行了高速缓存，则其便具有要提供的所有必需数据（这是由于所有更小的请求是该高速缓存的字节范围的一个子集）。

[0052] 在特定的实施例中，单播服务器是能够担任字节范围请求的简单、传统 HTTP web 服务器。因此，可以在无需了解有任何特定的广播被执行的情况下，对这些单播服务器进行设计。

[0053] 下面结合附图的详细描述将提供对于本发明的本质和优点的更好理解。

附图说明

[0054] 图 1 是示出传统 FEC 有效载荷 ID 的图；图 1A 示出了针对 Raptor-RFC5053 的 FEC 有效载荷 ID，而图 1B 示出了针对 RaptorQ-RFC-6330 的 FEC 有效载荷 ID。

[0055] 图 2 是示出用于基本通用文件传输（“UFD”）方法的 FEC 有效载荷 ID 的图。

[0056] 图 3 是示出发送方基本 UFD 方法的流程图。

[0057] 图 4 是示出接收方基本 UFD 方法的流程图。

[0058] 图 5A- 图 5B 是示出文件的符号的 (SBN、ESI) 标识，以及文件的符号的映射到和映射自相对应的通用文件符号标识符（“UFSI”）的示例。

[0059] 图 6 是示出发送方通用文件传输、不等错误保护（“UFD-UEP”）方法。

[0060] 图 7 是示出接收方 UFD-UEP 方法的流程图。

[0061] 图 8A-8B 示出了包括两个部分的文件的 (SBN、ESI) 标识的示例，其中每一个部分具有不同的优先级。

[0062] 图 9 示出了与图 8A 和图 8B 相对应的、在来自文件的两个部分的经编码的符号的 (SBN、ESI) 标识符之间的映射的示例，这些分组包含针对这些部分的经编码的符号连同在每一个分组中包括的 UFSI。

[0063] 图 10 示出了使用 [RaptorQ-RFC-6330] 的简单 UEP 文件传输方法的性能。

[0064] 图 11 示出了简单 UEP 文件传输方法和 UFD-UEP 文件传输方法之间的示例性能比较，其中这两种方法均使用 [RaptorQ-RFC-6330]。

[0065] 图 12 示出了一个文件的文件传输、多个文件的文件传输和 UFD 绑定的文件传输方法之间的示例性能比较，其中所有这些方法均使用 [RaptorQ-RFC-6330]。

[0066] 图 13 是可以用于生成、发送和接收 Raptor、RaptorQ 或者作为文件传输的一部分的其它分组的通信系统的框图。

[0067] 图 14 是一种可以进行文件传输的通信系统的视图，其中一个接收机从多个发送方（其通常是独立的）接收输出符号。

[0068] 图 15 是一种可以进行文件传输的通信系统的视图，其中多个接收机（其可以是独立的）从多个发送方（其通常是独立的）接收输出符号，以便与只使用一个接收机和 / 或只使用一个发送方相比，以更少的时间来接收输入符号。

[0069] 图 16 描述了可以用于使用 HTTP 流媒体服务器来提供文件传输的块请求流媒体系统的示例。

[0070] 图 17 示出了图 16 的块请求流媒体系统的单元，其示出了客户端系统的单元中的

更多细节，所述客户端系统耦合到块服务基础设施（“BST”）以接收由内容摄取系统所处理的数据，如可以用于文件传输的。

[0071] 图 18 示出了可以用于准备进行文件传输的文件的摄取系统的硬件 / 软件实现。

[0072] 图 19 示出了客户端系统的硬件 / 软件实现，其中该客户端系统可以用于接收向该客户端系统传送的文件。

[0073] 图 20 示出了一种通用 FEC OTI 元素格式的示例。

[0074] 图 21 示出了一种特定于方案的 FEC OTI 元素格式的示例。

[0075] 图 22 示出了基本 FLUTE 文件传输。

[0076] 图 23 示出了基本 FLUTE 分组格式。

[0077] 图 24 示出了在使用子块的发送方处发生的子块编码。

[0078] 图 25 示出了在使用子块的接收机处发生的子块解码。

[0079] 图 26 示出了使用子块的文件传输。

[0080] 图 27 示出了多个源块的处理。

[0081] 图 28 示出了使用 FEC 和 FLUTE 的工作流。

[0082] 图 29 示出了广播 / 修复、单播 / 源配置。

[0083] 图 30 示出了系统如何在 MBMS 承载上只广播修复符号。

[0084] 图 31 示出了通过 HTTP 字节范围请求，使用单播修复。

[0085] 图 32 示出了用于第一源块的子块范围请求计算的示例。

[0086] 图 33 示出了用于第二源块的子块范围请求计算的示例。

[0087] 图 34 示出了原始顺序 HTTP 文件格式和分区结构的示例。

[0088] 图 35 示出了具有 UOSI 顺序的源符号的示例。

[0089] 图 36 示出了具有 UOSI 顺序的修复符号的示例。

[0090] 图 37 示出了扩展原始顺序 HTTP 文件格式的示例。

[0091] 图 38 示出了扩展原始顺序 HTTP 文件格式的另一个示例。

[0092] 图 39 示出了用于不具有子块的原始顺序 HTTP 文件格式的字节范围计算的示例。

[0093] 图 40 示出了用于不具有子块的扩展原始顺序 HTTP 文件格式的字节范围计算的示例。

[0094] 图 41 示出了用于具有子块的扩展原始顺序 HTTP 文件格式的字节范围计算的示例。

具体实施方式

[0095] 在本申请的实施例中，文件传输通过用于发送文件的编码器 / 发射机系统和用于接收文件的接收机 / 解码器系统来执行。对进行的传输的格式进行协调，使得解码器理解编码器怎样进行了编码。如下面的各个示例中所示，除非另外指出，否则文件传输是通用对象传输的一个示例，通过这些示例，显而易见的是，可以将对象处理成文件，反之亦然。

[0096] 在分组传送系统中，将数据组织成一些分组，并发送成分组。每一个分组具有允许接收机确定该分组中有什么内容，以及其如何布局的元素。使用本申请所描述的技术，提供了用于当使用前向纠错（“FEC”）时，发送分组的灵活性。

[0097] 使用这些技术，可以提供不等 FEC 保护，以及文件的绑定传输。公知的是，与将多

个文件连接在一起形成更大的文件，并在传输时保护该更大的文件相比，当将多个文件传送成单独的文件时，传输时发生分组丢失的弹性更小。但是，需要作为较小文件的组合的该大型文件的结构的信令，接收机通常需要恢复整个大文件，来恢复该大文件中的较小文件里的任何一个，即使接收机只是对恢复这些较小文件的一个子集感兴趣。

[0098] 因此，优选的文件传输系统或者方法应当允许多个源块和每一个源块的多个经编码的符号的任意灵活组合，该组合用作一个文件的文件传输结构。在典型的实现中，一个源块是 FEC 操作（例如，生成修复符号）的范围。例如，可以根据全部来自一个源块的一个或多个源符号，来生成修复符号。在这些情况下，每一个源块是可以独立解码的。这在解码器处是有用的（如果在接收到所有数据之前，需要对被传输的一些数据进行解码和处理的话）。通过本发明公开内容，应当显而易见的是，如果源块非常的小，则接收的修复符号将可用于恢复较小数量的源符号，但是，其源块非常大，这使接收机解码和 / 或处理和 / 或使用该源块中的任何源符号都要花费更多的时间，这是因为其要花更长的时间对更大的源块进行解码。

[0099] 文件传输方法应当提供防止分组丢失的高效保护，在按照不同的优先级来保护文件的不同部分的情况下，支持文件的传输，其中文件的每一个部分可以与该文件的其它部分具有不同的源块结构和子块结构。同样，在一些情况下，将文件视作为对象的特定示例，但在本申请，应当理解的是，本申请使用的用于描述文件的传输和处理的示例，还可以用于或许没有称为文件的数据对象，例如，来自于数据库的大量的数据、视频序列的一部分等。

[0100] 文件 / 对象传输系统或方法应当在下面能力的基础上，提供多个小型文件 / 对象的传输 ：大型文件 / 对象的保护高效、小型文件 / 对象结构的简单信令、以及接收机在无需恢复所有这些小型文件 / 对象的基础上，独立地恢复这些小型文件 / 对象的一个子集的能力。

[0101] 文件传输系统可以包括广播部分和单播部分。为了易读性起见，“广播”可以是意味着“广播、多播和 / 或用于向多个用户服务共同数据的其它机制”。“单播”指代数据从一个源移动到一个目的地，但一个逻辑源可以包括多个组成部分，一个逻辑目的地可以包括多个组成部分。在单播配置中，通常存在一个源服务器和一个目的地客户端，其中源服务器等待来自客户端的请求，通过专门向请求的客户端发送所请求的数据（如果准许的话），对接收的请求进行响应。如本领域所公知的，针对大量目的地的单播传输，与针对这些相同数量的目的地的广播传输相比，可能产生更多的可扩展性挑战。通常，对于 HTTP 传输来说，贯穿整个网络部署多个高速缓存的服务器，以增加服务器传输伸缩能力。但是，这种方法并不必然地增加网络容量，网络容量是用于向移动设备传送内容的通常瓶颈。

[0102] 现在描述具有这些期望的质量的系统的示例。

[0103] 基本通用文件传输（“UFD”）方法和系统

[0104] 现在描述基本通用文件传输（“UFD”）方法和相应的系统，其中 UFD 与现有的文件传输方法相比，具有显著的优点。用于基本 UFD 方法的前向纠错（“FEC”）有效载荷 ID 包括通用文件符号标识符（“UFSI”）字段，例如，其可以是 32 比特字段。现在描述用于基本 UFD 方法的发送方和接收方方法。同样，当文件称为对象时，“UFSI”可以替代地称为“UOSI”（通用对象符号标识符）。

[0105] 图 1 是示出传统 FEC 有效载荷 ID 的图 ；图 1A 示出了用于 Raptor-RFC5053 的 FEC

有效载荷 ID，而图 1B 示出了用于 RaptorQ-RFC-6330 的 FEC 有效载荷 ID。

[0106] 图 2 是示出用于基本通用文件传输（“UFD”）方法的 FEC 有效载荷 ID 的图。后一种方法可以是更灵活的。

[0107] 图 3 示出了发送方基本 UFD 方法。发送方可以使用现有的方法来生成 FEC 对象传输信息（“OTI”），例如，如 [Raptor-RFC-5053] 或者 [RaptorQ-RFC-6330] 中所描述的（例如，参见 [RaptorQ-RFC-6330] 的第 4.3 节），使用该 FEC OTI 来确定发送该文件时将使用的源块和子块结构，确定 (SBN、ESI) 对和该文件的经编码的符号之间的关系（例如，参见 [RaptorQ-RFC-6330] 的第 4.4 节）。

[0108] 例如，如 [RaptorQ-RFC-6330] 中所描述的，所生成的 FEC OTI 可以是 (F,Al,T,Z,N)，其中 F 是要发送的文件的大小，Al 是用于确保子符号在存储边界对齐的对齐因子，其中存储边界是 Al 的倍数，T 是生成的并在该传输中发送的符号的大小，Z 是为了对该文件进行传输而划分成的源块的数量，N 是为了传输而将每一个源块所划分成的子源块的数量。如图 3 的步骤 300 中所示出的。

[0109] 发送方可以用分组来形成要发送的经编码的符号，基于源块，使用现有的方法来生成用于这些经编码的符号的 SBN 和 ESI，如果使用子块划分的话，则还使用子块结构，例如，如 [RaptorQ-RFC-6330] 中所描述的。在发送经编码的符号的每一次，发送方都确定针对该经编码的符号要生成的 SBN A 和 ESI B，如图 3 的步骤 310 中所示，随后发送方可以使用现有的技术（例如，[RaptorQ-RFC-6330] 中所描述的那些技术），基于 (SBN,ESI) = (A,B) 来生成该经编码的符号的值，如图 3 的步骤 320 中所示。随后，将用于该经编码的符号的 UFSI C 计算成 C = B*Z+A，如图 3 的步骤 330 中所示。

[0110] 发送方可以在分组中发送该经编码的符号，其中该分组的 FEC 有效载荷 ID 被设置为该经编码的符号的 UFSI C，如图 3 的步骤 340 中所示。随后，发送方可以确定是否有更多的经编码的符号要发送，如图 3 的步骤 350 中所示，如果是的话，则发送方可以生成另外的经编码的符号来发送，如去往图 3 的步骤 310 的“是”分支所示出的，如果没有，则发送方可以完成操作，如去往图 3 的步骤 360 的“否”分支所示出的。

[0111] 存在发送方基本 UFD 方法的多种变型。例如，发送方可以在至少一些分组中，发送一个以上的经编码的符号，在该情况下，可以将 FEC 有效载荷 ID 设置为该分组中所包含的第一经编码的符号的 UFSI，可以对该分组中所包含的其它符号进行选择，使得其相应 UFSI 值是连续的。例如，如果在该分组中携带有三个经编码的符号，第一个这种符号具有 UFSI = 4,234，那么其它两个经编码的符号可以是分别具有 UFSI4,235 和 4,236。举一些其它替代的示例，发送方可以预先确定要生成多少经编码的符号，在生成这些经编码的符号中的任何一个之前，确定用于要生成的所有经编码的符号的 (SBN,ESI) 值。再举一个例子，可以在无需生成 (SBN,ESI) 值的中间步骤的情况下，直接生成 UFSI 值。

[0112] 再举一种变型的示例，可以生成其它形式的 FEC OTI。例如，可以将基 UFSI BU 包括在用于该文件的 FEC OTI 中，其可以如下所述地使用 :FEC 发送方和接收机针对于一个分组中所包含的经编码的符号而使用的 UFSI 是 U+BU，其中 U 是在携带该经编码的符号的分组中所携带的 UFSI。因此，例如，如果一个分组携带 U = 1,045，在 FEC OTI 中的基 UFSI 是 BU = 2,000,000，那么该经编码的符号是 2,001,045。基 UFSI 的使用具有一些优点。第一，在 [FLUTE]、[ALC]、[LCT]、[FEC BB] 中描述的协议簇将传输对象标识符（其还称为 TOI）与要

传输的文件或对象的 FEC OTI 进行关联。可以在不同的时间，或者在不同的会话中，发送同一文件的经编码的符号，这些经编码的符号可以与不同的 TOI 相关联。此外，有利的是，针对与每一个不同的 TOI 相关联的分组，能够发送以 UFSI = 0 起始的编码分组。通过具有将基 UFSI 指定成 FEC OTI 的一部分的能力，不同的基 UFSI 可以与针对该文件将发送的经编码的符号的每一个 TOI 相关联，而无需针对不同的 TOI，发送重复的经编码的符号。例如，可以在与 TOI = 1 相关联的分组和与 TOI = 2 相关联的分组中，发送同一文件的经编码的符号，其中与 TOI = 1 相关联的基 UFSI 被设置为 0，与 TOI = 2 相关联的基 UFSI 被设置为 1,000,000。随后，TOI = 1 和 TOI = 2 的编码分组，可以包含 UFSI0、1、2 等的序列，只要针对具有 TOI = 1 的文件，发送少于 1,000,000 个经编码的符号，那么在与这两个 TOI 相关联的发送的经编码的符号之中，不存在重复的经编码的符号。

[0113] 参照图 4，描述了接收机基本 UFD 方法。接收机可以使用现有的技术，来确定具有与上面针对发送方所描述的相同格式的 FEC OTI(F,A1,T,Z,N)，如图 4 的步骤 400 中所示。例如，可以将 FEC OTI 嵌入在 FLUTE 会话描述中，或者可以将 FEC OTI 编码到 URL 中，或者可以通过 SDP 消息来获得 FEC OTI。在步骤 410 中，接收机查看是否接收到更多的经编码的符号，其可以停留在该步骤，直到满足下面条件为止：接收机接收到另一个经编码的符号（在该情况下，接收机转到步骤 430），或者接收机确定将会接收到更多的经编码的符号，在该情况下，接收机转到步骤 420，并尝试使用其它方式来恢复该文件，例如使用针对文件修复服务器的 HTTP 请求，或者接收机可以等待另一个会话，以便在稍后时间接收更多的经编码的符号，或者接收机可以确定该文件不能被恢复。

[0114] 当另一个经编码的符号可用时，在步骤 430，接收机确定该经编码的符号的 UFSI C，接收该经编码的符号的值。在步骤 440，接收机基于源块的数量 Z 和 UFSI C，计算 A = C 模 Z，以及 B = floor(C/Z)，在步骤 450，接收机将用于该经编码的符号的 (SBN,ESI) 设置为 (A,B)，在步骤 460，接收机存储该经编码的符号的值和 (A,B)，以便用于文件恢复。在步骤 470，接收机确定是否接收到足够的经编码的符号来恢复该文件，如果是，则转到步骤 480 来恢复该文件，否则的话，转到步骤 410 来接收更多的经编码的符号。

[0115] 存在接收机基本 UFD 方法的多种变型。例如，接收机可以在至少一些分组中，接收一个以上的经编码的符号，在该情况下，可以将 FEC 有效载荷 ID 设置为该分组中所包含的第一经编码的符号的 UFSI，该分组中的其它符号可以具有连续的相应 UFSI 值。例如，如果在该分组中携带有三个经编码的符号，第一个这种符号具有 UFSI = 4,234，那么其它两个经编码的符号可以是分别具有 UFSI4,235 和 4,236，在该分组中携带的 UFSI 可以是第一经编码的符号的 UFSI = 4,234。举一些其它替代的示例，接收机可以在尝试恢复之前，预先确定要接收多少经编码的符号。再举一个例子，接收机可以做一些特定于 FEC 编码的处理，其用于确定是否接收到足够的经编码的符号来恢复该文件。再举一个例子，在恢复过程中，可以在无需生成 (SBN,ESI) 值的中间步骤的情况下，直接地使用 UFSI 值。再举一个例子，文件的恢复可以与经编码的符号的接收同时地发生。再举一个例子，可以使用其它形式的 FEC OTI 信息。

[0116] 将基本 UFD 方法与在 [RaptorQ-RFC-6330] 中描述的技术进行组合，以确定源块和子块结构提供很多种利益。例如，先前的方法所称的源符号（为了发送文件，其通过 SBN 和 ESI 的组合来进行标识），可以被视作为通过 UFSI 来标识的文件符号（当使用基本 UFD 方

法时）。使 F 表示要发送的文件的以字节计算的大小，使 T 表示在发送该文件时，用于 FEC 编码 / 解码目的的符号大小，因此 KT = ceil(F/T) 是该文件中的符号的总数，其中 ceil(x) 是大于或等于 x 的最小整数。

[0117] 当确定源块结构和子块结构时（例如，如在 [RaptorQ-RFC-6330] 中所描述的），使用上面所描述的基本 UFD 方法，将符号的标识从 (SBN, ESI) 格式转换成 UFSI 格式和从 UFSI 格式转换成 (SBN, ESI) 格式，用于文件符号的 UFSI 的范围是 0, 1, 2, …, KT-1，根据该文件生成的任何修复符号将具有范围 KT, KT+1, KT+2 等中的 UFSI。这种属性允许通过简单地将一个符号的 UFSI 与 KT 的值进行比较，来确定该符号是原始文件的一部分，还是根据该文件所生成的修复符号。例如，这可以用于使不支持 FEC 解码的接收机，能够基于在分组中携带的 UFSI 值，以及基于用于该文件的 KT 的值，来确定哪些符号是原始文件的一部分（以及其在该文件中的位置），哪些符号可以忽略成修复符号。

[0118] 图 5A 和图 5B 示出了一种示例，在该情况下，文件大小是 F = 28,669 个字节，符号大小是 T = 1,024 个字节，因此 KT = ceil(F/T) = 28。在这两个附图中，源块的数量是 Z = 5。在图 5A 和图 5B 中，在顶部和 / 或旁边，示出了文件的符号的 (SBN, ESI) 标签，其中每一行对应于一个源块，每一列对应于具有相同 ESI 值的符号。在底部示出了这些符号的相应 UFSI 标签。在该情况下，UFSI = 27 的符号（即，关于 UFSI 标签的第 28 个符号）具有为了 FEC 编码和解码的目的，而以零填充的其最后 (KT*T)-F = 3 个字节，但是不发送该符号的这最后三个填充的字节。在该示例中，具有 UFSI28 或者更大值的任何经编码的符号是修复符号，其中具有 UFSI28 的经编码的符号是根据 SBN = 3 的源块来生成的，具有 UFSI29 的经编码的符号是根据 SBN = 4 的源块来生成的，具有 UFSI30 的经编码的符号是根据 SBN = 0 的源块来生成的等。这种属性具有多种其它的优点，如本领域普通技术人员所认识到的。

[0119] 基于 UFD 方法的另一种优点在于：如果按照其 UFSI 的顺序（即，以 UFSI 顺序 0、1、2、3、4、…、）来发送经编码的符号，那么用于 Z 个源块的经编码的符号是以交织顺序来发送的，即，首先发送这 Z 个源块中的每一个里具有 ESI0 的 Z 个经编码的符号，接着发送这 Z 个源块中的每一个里具有 ESI1 的 Z 个经编码的符号等。对于大部分传输来说，这种简单的发送顺序是足够的和优选的。但是，如果分组丢失经历某种周期性（其可能与源块的数量 Z 相同步），则潜在的更佳发送顺序是在发送这些符号之前，对每一组的 Z 个 UFSI 连续经编码的符号进行随机地置换，即，以随机置换的顺序来发送具有 UFSI0, …, Z-1 的前 Z 个经编码的符号，随后，以随机置换的顺序来发送具有 UFSI Z, …, 2*Z-1 的下面 Z 个经编码的符号等。应当理解的是，贯穿本说明书，除非另外指出，否则“随机”可以包括伪随机。

[0120] 相比于先前已知的方法，使用基本 UFD 方法能提供多种其它的优势。例如，当使用包括预定大小的 SBN 和 ESI 字段的 FEC 有效载荷 ID 时，存在单独的预先确定的数量的可能源块，或者每一源块具有多个可能的经编码的符号。例如，导致 32 比特 FEC 有效载荷 ID 的 8 比特 SBN 和 24 比特 ESI，将可能的源块的数量限制于 256，将每一个源块的可能的经编码的符号的数量限制于 16,777,216。相反，包括 UFSI 字段的 FEC 有效载荷 ID 只限制用于一个文件的可能经编码的符号的总数，而独立于该文件的源块结构。例如，当使用导致 32 比特 FEC 有效载荷 ID 的 32 比特 UFSI 时，可以针对一个文件所生成的经编码的符号的总数是 4,294,967,296，独立于该文件被划分成了多少个源块，并且独立于该文件的子块结构（如

果使用子块划分的话）。因此，如果符号的大小是1,024个字节，那么在该示例中，文件大小可以是多达4GB，经编码的符号的数量可以是1,024乘以该文件大小，其独立于该文件是被划分成一个源块、16,384个源块、还是4,194,304个源块。再举一个例子，文件大小可以是2TB，经编码的符号的数量可以是该文件大小的两倍。在所有情况下，针对该文件所生成的经编码的符号的数量，都独立于该文件的子块结构（如果使用子块结构的话）。

[0121] 用于不等错误保护文件传输服务的通用文件传输方法

[0122] 大部分的先前文件传输方法都不支持不等错误保护（“UEP”）文件传输。存在着一些通过改变实际的FEC编码（其中该实际FEC编码用于对文件的不同部分进行编码），来支持UEP的现有方法，例如，当前在ISDB-Tmm（陆地移动多媒体）标准中所指定的那些方法、使用[LDPC-Extensions]中所示出的方法。除了在使用UEP时，必须改变实际的FEC编码的缺点之外，另外的缺点在于：这些方法所提供的保护与理想差距很大。

[0123] 举例而言，考虑在[LDPC-Extensions]的图6中示出其结果的示例。在该示例中，将以分组来发送的大小为1,000KB的文件具有两个部分，每一个分组包含1KB符号：该文件的第一部分具有30KB的大小，通过100个奇偶校验、或者修复分组来保护，该文件的第二部分具有970KB的大小，针对该文件发送了总共1,000个源分组和1,000个奇偶校验分组。所提供的保护由于两种原因而与理想差距很大。一种原因在于：基于[LDPC-RFC-5170]，使用的FEC编码通常需要显著的接收开销来恢复源块，即，与一个文件中存在的源分组相比，需要接收更多的分组来恢复该文件。第二种问题在于：该方法本质上使用与用于发送和保护该文件的第二部分不相同的分组集，来发送和保护该文件的第一部分。对于第二种问题来说，在接收针对该文件的第一部分所发送的130个分组之外的30个分组时的方差可能很大，这是由于该文件的第一部分是通过这种较少数量的分组来发送的。

[0124] 可以通过一种扩展来改进在[LDPC-Extensions]中所描述的UEP文件传输方法，本申请称为“简单UEP”文件传输方法。该简单UEP文件传输方法使用用于文件传输的现有技术，并且基于优先级，针对该文件的每一个部分，使用不同数量的保护，将该文件的各部分作为单独文件来传送，随后可以发送该文件的各部分之间的逻辑连接，使得接收机知道所传送的文件是同一文件的一部分。例如，该简单UEP文件传输方法可以使用上面的示例中的[RaptorQ-RFC-6330]，通过发送总共130个分组来传送该文件的前30KB部分，其每一个分组包含根据该第一部分所生成的大小为1,024个字节的经编码的符号，随后可以通过发送总共1870个分组，将该文件的第二部分970KB传送成单独的文件，其每一个分组包含根据该第二部分所生成的大小为1,024个字节的经编码的符号。因此，针对发送成单独文件的该文件的两个部分，发送了总共2,000个分组。该简单UEP文件传输方法是相对于[LDPC-Extensions]中描述的方法的改进，这是由于不对FEC编码自身进行修改，并且由于与[LDPC-Extensions]的图6中所示出的相比，在不同的分组丢失状况下来传输该文件的两个部分的性能更出众，如本申请的图10中所示出的。

[0125] 这种差别的一种可能来源是由于：在[RaptorQ-RFC-6330]中所详细说明的FEC编码，相比在[LDPC-RFC-5170]中所详细说明的FEC编码具有更优的恢复属性。但是，这种简单UEP文件传输方法仍然承受上面所描述的第二种问题。

[0126] [PET]和[PET-Patent]提供了用于提供UEP文件传输服务的潜在改进方法，其中每一个分组包含：基于该文件的每一个部分的优先级，来自于该部分的指定数量的经编码

的数据。[PET] 的直接合并将会在每一个分组中包括针对该文件的每一个部分的适当大小的经编码的符号，随后包括单独的 FEC 有效载荷 ID，其包括针对该文件的每一个部分的 (SBN, ESI) 对。但是，该方法由于几种原因而不具有优势。

[0127] 例如，当使用 FEC 有效载荷 ID（其包括针对每一个部分的预定大小的 SBN 和 ESI 字段）时，针对该文件的每一个部分的每一个源块，存在单独的预定数量的可能源块或者多个可能的经编码的符号。例如，针对 d 个部分的每一个部分具有 8 比特的 SBN 和 24 比特的 ESI，导致 (32*d) 比特的 FEC 有效载荷 ID，其将每一部分的可能的源块数量限制于 256，将每一个源块的可能的经编码的符号的数量限制于 16,777,216。此外，如果针对 d 个部分的每一个部分，FEC 有效载荷 ID 大小是 32 比特，那么这意味着在每一个分组中，针对所有部分具有总共 32*d 比特的 FEC 有效载荷 ID，例如，如果 d = 10，那么仅对于每一个分组中的 FEC 有效载荷 ID 报头来说，这将是 320 比特，或者等同地 40 个字节。

[0128] 可以将用于文件传输的基本 UFD 方法进行扩展，以便提供不等错误保护 (UEP) 文件传输服务，如下面所详细描述的，其相对于先前的 UEP 文件传输方法，提供显著的优点。这里，这些扩展的方法称为“UFD-UEP”文件传输方法。这些 UFD-UEP 文件传输方法可以使用在 [PET] 和 [PET-Patent] 中所描述的方法里的一些。

[0129] 现在更详细地描述一种示例性的 UFD-UEP 文件传输方法。在该方法中，发送方将大小为 F 的文件划分成大小为 $F_0, F_1, \cdots, F_{d-1}$ 的 d>1 个部分，因此 F 等于关于 i 的 F_i 之和。发送方将分组大小 T 划分成大小为 $T_0, T_1, \cdots, T_{d-1}$ 的 d 个部分，因此 T 等于关于 i 的 T_i 之和。这种 T 的划分是基于 $F_0, F_1, \cdots, F_{d-1}$ 和相应的文件部分的优先级。比率 F_i/T_i 确定为了恢复该文件的第 i 部分，需要接收多少个分组（其假定使用理想的 FEC 编码将该文件的第 i 部分保护成一个源块），因此比率 F_i/T_i 越小，则文件的第 i 部分的优先级越高。在实现时，可能需要比 F_i/T_i 稍微更多的分组，来恢复该文件的第 i 部分，例如，由于 FEC 编码是不完美的，并呈现某种接收开销，或者由于该文件的该部分被划分成多个源块，用于一些源块的经编码的符号按照与其它源块相比更高的比率丢失，或者由于 F_i/T_i 不是整数。举一个 UEP 的例子，假定 F = 1MB, T = 1,024octets, d = 2, F_0 = 32KB, F_1 = F-F_0 = 992KB, and T_0 = 64octets, T_1 = T-T_0 = 960 个字节。在该示例中，F_0/T_0 = 512，因此 512 个分组的理想接收允许该文件的第 0 部分的恢复，而 F_1/T_1 = 1,058.13，因此 1,059 个分组的理想接收允许该文件的第 1 部分的恢复。因此，在该示例中，可以根据恢复该文件的第 1 部分所需要的多个分组的大约一半，来恢复该文件的第 0 部分。

[0130] 应当注意，在该示例中，如果不使用 UEP（即，d = 1，因此 F_0 = F = 1MB, T_0 = T = 1,024 个字节），那么需要 F_0/T_0 = 1,024 个分组来恢复该文件。因此，在前面的段落中所描述的 UEP 示例中，与不使用 UEP 时相比，文件的第 1 部分的恢复需要稍微更多的分组，这是由于该文件的第 0 部分的优先级更高。可以在中找到这种基础平衡的分析研究。

[0131] 存在着发送方 UFD-UEP 方法基于 $F_0, F_1, \cdots, F_{d-1}$ 和不同的文件部分的优先级，来生成 T 的划分 $T_0, T_1, \cdots, T_{d-1}$ 的多种方式。应当注意，如果选择 T_i，使得 $F_i/T_i \approx F/T$，那么认为该文件的第 i 部分具有平均优先级，分别根据是 $F_i/T_i < F/T$，还是 $F_i/T_i > F/T$，第 i 部分的优先级可以是相对更高或者更低。

[0132] 现在参照图 6 来描述用于生成 FEC OTI 的发送方 UFD-UEP 方法。给定 d、F_0、F_1、…、F_{d-1}、T_0、T_1、…、T_{d-1}、A1、WS 的值，可以独立于使用现有的方法，例如使用在第 4.3 中的

[RaptorQ-RFC-6330] 里描述的方法，将 FEC OTI 计算成应用于该文件的 d 个部分中的每一个的，也就是，对于每一个 i = 0,…,d-1，发送方可以生成用于文件第 i 部分的 FEC OTI，其确定如何使用在第 4.3 中的 [RaptorQ-RFC-6330] 里描述的方法，将其划分成源块和子块，将 F_i 处理成文件大小，将 T_i 处理成用于在每一个分组中携带针对第 i 部分的信息的符号大小。因此，独立于文件的其它部分，发送方来生成针对该文件的第 i 部分的 FEC OTI。在本申请的图 6 的步骤 600 中，示出了该过程。

[0133] 此外，发送方还可以生成该文件的第 i 部分向源块和子块的划分，以及针对该文件的第 i 部分的经编码的符号的 (SBN,ESI) 之间的映射，如何使用现有的方法（例如，在第 4.4 节和第 5 节中的 [RaptorQ-RFC-6330] 里描述的方法）来根据该文件的第 i 部分来生成经编码的符号。独立于该文件的其它部分，将这些 UFD-UEP 方法应用于该文件的第 i 部分，因此该文件的不同部分可以具有不同的源块和子块结构，具体而言，每一个源块可以存在不同数量的源符号，在该文件的不同部分之间存在不同数量的源块，这是由于将这些方法独立地应用于该文件的每一个部分。

[0134] 优选地，对齐因子 Al 对于文件的所有部分来说是相同的，具体而言，优选的是，对于每一个部分 i，T_i 的值是 Al 的倍数。此外，例如，若使用在 [RaptorQ-RFC-6330] 的第 4.5 节中描述的方法，来导出 FEC OTI，则优选的是，当导出针对该文件的每一个部分的源块和子块结构时，使用相同的 Al 和 WS 的值。针对 Al 使用相同的值，确保在接收机处，在按照 Al 的倍数字节对齐的存储器上进行解码，针对 WS 使用相同的值，确保在接收机处需要在随机存取存储器 (RAM) 中进行解码的最大块大小，对于该文件的所有部分来说是相同的。但是，这里不需要本申请所描述的方法，即，如果将不同的 Al 值用于不同的文件，则这些方法在无需修改的情况下进行应用，如下面所进一步描述的。存在着一些应用，在这些应用中，针对文件的不同部分使用不同的 WS 值，对于导出 FEC OTI 来说是优选的，例如，如果期望使用更少的存储器来恢复该文件的更高优先级部分。

[0135] 存在一些应用，在这些应用中，针对不同的部分使用不同的对齐因子是有利的。例如，具有 4 字节对齐的存储器的低端接收机和具有 8 字节对齐的存储器的高端接收机，可以对高优先级部分进行解码，而低优先级部分只能由高端接收机进行解码。在该示例中，有利的是，针对高优先级部分使用 Al = 4，使得低端接收机可以对这些部分进行高效地解码，有利的是，针对低优先级部分使用 Al = 8，使得与具有 Al = 4 的部分相比，高端接收机可以对于对具有 Al = 8 的这些部分进行更高效地解码。

[0136] 特定于文件第 i 部分的发送方 UFD-UEP 方法所生成的相应 FEC OTI，包括 F_i、T_i、Z_i、N_i，其中 Z_i 是将该文件的第 i 部分所划分到的源块的数量，N_i 是将该文件的第 i 部分的每一个源块所划分到的子块的数量。因此，整体上，发送方 UFD-UEP 方法针对该文件所生成的 FEC OTI 可以包括：$(d, Al, F_0, T_0, Z_0, N_0, F_1, T_1, Z_1, N_1, \cdots, F_{d-1}, T_{d-1}, Z_{d-1}, N_{d-1})$。其它版本的 FEC OTI 也是可用的，例如，当 d 是固定的，并因此不需要在 FEC OTI 中显式地列出时，或者当使用其它方法来指示源块结构和子块结构时（如果使用的话）。

[0137] 使用 UFD-UEP 方法的发送方，将针对该文件的每一个部分的一个经编码的符号进行组合，以便在一个分组中进行发送，用于该分组的 FEC 有效载荷 ID 包括 UFSI 值 C。当要发送一个分组时，发送方确定将用作用于该分组的 FEC 有效载荷 ID 的 UFSI 值 C，如图 6 的步骤 610 中所示出的。例如，将使用的 UFSI 值可以是连续的，例如，UFSI 值 0、1、2、3、…、

等。对于给定的 UFSI 值 C，在针对文件的第 i 部分的分组中，将放置在分组的第 i 部分中的大小为 T_i 的经编码的符号，具有 SBN A_i 和 ESI B_i，其计算成 A_i = C 模 Z_i，B_i = floor(C/Z_i)，如图 6 的步骤 620 中所示出的。对于，针对分组的 d 个部分中的每一个，生成这些 d 个经编码的符号，随后将 UFSI C 与聚合大小为 T 的这些 d 个经编码的符号一起在分组中进行发送，如图 6 的步骤 630、640 和 650 中所示出的。发送方 UFD-UEP 方法继续生成和发送编码分组，如在图 6 的步骤 660 中进行的确定所示出的。

[0138] 参照图 7，来描述接收机 UFD-UEP 方法。接收机可以使用现有的技术，来确定具有与上面关于发送方所描述的相同格式的 FEC OTI(d, Al, F_0, T_0, Z_0, N_0, F_1, T_1, Z_1, N_1, …, F_{d-1}, T_{d-1}, Z_{d-1}, N_{d-1})，如图 7 的步骤 700 中所示出的。例如，可以将 FEC OTI 嵌入在 FLUTE 会话描述中，或者可以将 FEC OTI 嵌入到 URL 中，或者可以通过 SDP 消息来获得 FEC OTI。在步骤 710 中，接收机查看是否接收到更多的分组，其可以停留在该步骤，直到满足下面条件为止：接收机接收到另一个分组（在该情况下，接收机转到步骤 730），或者接收机确定将不会接收到更多的分组，在该情况下，接收机转到步骤 720，并确定是否恢复了该文件的足够部分并进行停止，或者尝试使用其它方式来恢复该文件的其它部分，例如使用针对文件修复服务器的 HTTP 请求，或者接收机可以等待另一个会话，以便在稍后时间接收更多的分组。

[0139] 当另一个分组可用时，在步骤 730，接收机确定所接收的分组的 UFSI C，对于每一个 i = 0、…、d-1，从针对每一个 i = 0、…、d-1 的分组中提取大小为 T_i 的经编码的符号。在步骤 740，对于每一个 i = 0、…、d-1，接收机基于源块的数量 Z_i 和 UFSI C，计算 A_i = C 模 Z_i，以及 B_i = floor(C/Z_i)，在步骤 750，接收机将用于第 i 部分的经编码的符号的 (SBN, ESI) 设置为 (A_i, B_i)，在步骤 760，接收机存储用于第 i 部分的经编码的符号的值和 (A_i, B_i)，以便用于该文件的第 i 部分的恢复。在步骤 770，针对每一个 i = 0、…、d-1，接收机确定是否接收到足够的经编码的符号来恢复该文件的第 i 部分，如果是，则转到步骤 780 来恢复该文件的第 i 部分，否则的话，转到步骤 710 来接收更多的分组。

[0140] 存在接收机 UFD-UEP 方法的多种变型。例如，发送方可以在至少一些分组中，发送针对文件的每一个部分的一个以上的经编码的符号，接收机可以在至少一些分组中，接收针对文件的每一个部分的一个以上的经编码的符号，在该情况下，可以将 FEC 有效载荷 ID 设置为与该分组中所包含的针对每一个部分的第一经编码的符号相对应的 UFSI，针对该分组中的每一个部分的其它符号可以具有连续的相应 UFSI 值。例如，如果在该分组中携带有针对该文件的每一个部分的三个经编码的符号，针对每一个部分的第一符号与 UFSI = 4,234 相对应，那么针对每一个部分的其它两个经编码的符号可以是分别与 UFSI4,235 和 4,236 相对应，在该分组中携带的 UFSI 可以是 UFSI = 4,234。

[0141] 举一些其它替代的示例，接收机可以在尝试恢复之前，预先确定要接收多少经编码的符号，或者可以计算在该会话期间的分组丢失统计，并基于此来决定将尝试恢复该文件的哪个部分。再举一个例子，接收机可以做一些特定于 FEC 编码的处理，其用于确定是否接收到足够的经编码的符号来恢复该文件的各个部分。再举一个例子，在针对各个文件部分的恢复过程中，可以在无需生成 (SBN, ESI) 值的中间步骤的情况下，直接地使用 UFSI 值。再举一个例子，文件的一部分的恢复可以与经编码的符号的接收同时地发生。

[0142] 再举一个例子，可以使用其它形式的 FEC OTI 信息。例如，可以独立于其它部分，将基 UFSI BU_I 指定在用于第 i 部分的 FEC OTI 中，其可以如下所述地使用：FEC 发送方和

接收机针对于一个分组中所包含的第 i 部分的经编码的符号而使用的 UFSI 是 $U+BU_I$，其中 U 是在携带该经编码的符号的分组中所携带的 UFSI。因此，例如，如果一个分组携带 U = 1,045，在针对第 i 部分的 FEC OTI 中的基 UFSI 是 BU_i = 2,000,000，那么该经编码的符号是 2,001,045。

[0143] 基 UFSI 的使用具有一些优点。例如，在针对于不同的部分，仅发送一个修复符号的情况下（由于稍后描述的原因），设置 $BU_i = KT_i$ 是有利的，其中 KT_i 是第 i 部分中的文件符号的数量。在该情况下，在发送的分组序列中的 UFSI 序列可以是 0、1、2、3 等，而不管对于每一个部分来说，在分组中只发送根据该部分所生成的修复符号。

[0144] 本段描述基 UFSI 的使用的另一种示例性优点。在 [FLUTE]、[ALC]、[LCT]、[FEC BB] 中描述的协议簇将传输对象标识符（其还称为 TOI）与要传输的文件或对象的 FEC OTI 进行关联。可以在不同的时间，或者在不同的会话中，发送针对相同部分的经编码的符号，这些经编码的符号可以与不同的 TOI 相关联。此外，有利的是，针对与每一个不同的 TOI 相关联的分组，能够发送以 UFSI = 0 起始的编码分组。通过具有针对每一个部分，能够独立地将基 UFSI 指定成 FEC OTI 的一部分的能力，针对各个部分的不同的基 UFSI 可以与针对该文件将发送的经编码的符号的每一个 TOI 相关联，而无需针对不同的 TOI，发送重复的经编码的符号。例如，可以在与 TOI = 1 相关联的分组和与 TOI = 2 相关联的分组中，发送针对相同部分的经编码的符号，其中与 TOI = 1 相关联的针对该部分的基 UFSI 被设置为 0，与 TOI = 2 相关联的针对相同部分的基 UFSI 被设置为 1,000,000。随后，TOI = 1 和 TOI = 2 的编码分组，可以包含 UFSI0、1、2 等的序列，只要针对具有 TOI = 1 的该部分，发送少于 1,000,000 个经编码的符号，那么在与这两个 TOI 相关联的发送的经编码的符号之中，不存在该部分的重复的经编码的符号。此外，还有利的是，具有将被 FEC OTI 中的所有部分都使用的基 UFSI，而不是针对每一个部分，指定不同的基 UFSI，这是由于这可以减少传输 FEC OTI 所需要的字节数量，同时共享针对该 FEC OTI 中的每一个部分，指定单独的基 UFSI 的多种优点，特别是当结合该方法使用的 FEC 编码是信息增加的 FEC 编码（例如，在 Luby I 中所描述的 FEC 编码）时，这是由于在该情况下，针对所有部分的 UFSI 的有效范围可以是非常的大。

[0145] 将基本 UFD-UEP 方法与在 [RaptorQ-RFC-6330] 中描述的技术进行组合，以确定源块和子块结构提供很多种利益。具体而言，基本 UFD 方法的所有优点，在 UFD-UEP 方法中仍然持有。例如，先前的方法所称的源符号（为了发送文件的 UEP 部分中的一个，其通过 SBN 和 ESI 的组合来进行标识），可以被视作为通过 UFSI 来标识的文件符号（当使用 UFD-UEP 方法时）。应当注意，KT = ceil(F/T) 是该文件的第 i 部分中的文件符号的总数。当针对文件的每一个部分，确定源块结构和子块结构时（例如，如在 [RaptorQ-RFC-6330] 中所描述的），使用上面所描述的 UFD-UEP 方法，将针对于文件的一个部分的符号的标识从 (SBN,ESI) 格式转换成 UFSI 格式和从 UFSI 格式转换成 (SBN,ESI) 格式，用于文件符号的 UFSI 的范围是 0,1,2,…，KT_i-1，根据该文件生成的任何修复符号将具有范围 KT_i, KT_i+1, KT_i+2 等中的 UFSI。这种属性允许通过简单地将符号的 UFSI 与 KT_i 的值进行比较，来确定该符号是文件的原始第 i 部分的一部分，还是根据该文件的第 i 部分所生成的修复符号。例如，这可以用于使不支持 FEC 解码的接收机，能够基于在分组中携带的 UFSI 值，以及基于用于该文件的每一个部分 i 的 T_i 和 KT_i 的值，来确定分组的哪些部分包含原始文

件的部分（以及其在该文件中的位置），分组的哪些部分包含可以忽略的修复符号。

[0146] 图 8A 和图 8B 示出了一种示例，在该情况下，该文件包括两个部分。第一部分被划分成 5 个源块，其中这些源块中的前 3 个的每一个都具有 6 个源符号，剩余的 2 个源块中的每一个具有 5 个源符号，其中这些符号中的每一个的大小是例如 48 个字节，因此第一部分的大小是 28*48 = 1,344 个字节。第二部分被划分成 4 个源块，其中这些源块中的前 3 个的每一个都具有 4 个源符号，剩余的 1 个源块具有 3 个源符号，其中这些符号中的每一个的大小是例如 256 个字节，因此第二部分的大小是 15*256 = 3,840 个字节。

[0147] 图 9 示出了用于图 8A 和图 8B 中所示出的文件结构的一种可能信息分组。在该示例中，每一个分组都包括 UFSI C、基于 C 来根据图 8A 和图 8B 中所示的文件结构的第一部分所生成的大小为 T_1 = 48 个字节的第一经编码的符号（如先前参照图 6 所描述的）、以及基于 C 来根据图 8A 和图 8B 中所示的文件结构的第二部分所生成的大小为 T_2 = 256 个字节的第二经编码的符号（如先前参照图 6 所描述的）。这些分组的无阴影部分携带该文件的相应部分的源符号，而有阴影部分携带根据该文件的相应部分所生成的修复符号。在该示例中，为了恢复该文件的第一部分所需要的最小数量的分组是 28，而为了恢复该文件的第二部分所需要的最小数量的分组是 15。

[0148] 图 9 描述了具有 UFSIs0,…,27 的 28 个分组，因此在这些分组中携带的针对文件的第一部分的所有经编码的符号是源符号。使用至少 28 的 UFSI 值所生成的任何其它分组，携带针对该文件的第一部分的修复符号。

[0149] 基于 UFD-UEP 方法的另一种优点在于：如果按照其 UFSI 的顺序（即，以 UFSI 顺序 0、1、2、3、4、…、）来发送经编码的符号，那么用于文件的第 i 部分的 Z_i 个源块的经编码的符号是以交织顺序来发送的，即，首先发送这 Z_i 个源块中的每一个里具有 ESI0 的 Z_i 个经编码的符号，接着发送这 Z_i 个源块中的每一个里具有 ESI1 的 Z_i 个经编码的符号等。这种属性对于文件的所有部分来说都是成立的，即使每一个部分具有独立的源块结构。对于大部分传输来说，这种简单的发送顺序是足够的和优选的。但是，如果分组丢失经历某种周期性（其可能与源块的数量 Z_i 相同步），则潜在的更佳发送顺序是在发送这些符号之前，对每一组的 Z 个 UFSI 连续经编码的符号进行随机地置换（其中，Z 是所有的 Z_i 值之中的最小值，i = 0,…,d-1），即，以随机置换的顺序来发送具有 UFSI0,…,Z-1 的前 Z 个经编码的符号，随后，以随机置换的顺序来发送具有 UFSI Z,…,2*Z-1 的下面 Z 个经编码的符号等。另一种潜在发送顺序是：在发送经编码的符号之前，对要发送的所有经编码的符号进行随机地置换。

[0150] 相比于先前已知的方法或者先前方法的简单扩展，使用 UFD-UEP 方法能提供多种其它的优势。例如，使用包括 UFSI 字段的 FEC 有效载荷 ID 字段的 UFD-UEP 方法，针对文件的每一个部分的可能经编码的符号的总数仅受到 UFSI 字段的大小的限制，其独立于该文件的各个部分的源块结构。此外，UFSI 字段的使用提供了一种通用和简洁的 FEC 有效载荷 ID，其允许对根据文件的各个部分的完全不相同的源块结构所生成的符号进行同时地标识。例如，当使用导致 32 比特 FEC 有效载荷 ID 的 32 比特 UFSI 时，可以针对一个文件所生成的经编码的符号的总数是 4,294,967,296，独立于该文件被划分成了多少个源块，并且独立于针对该文件的各个部分的该文件的子块结构（如果使用子块划分的话）。因此，如果针对文件的第一部分的符号的大小是 256 个字节，针对文件的第二部分的符号的大小是

1,024 个字节，那么在该示例中，文件的第一部分的大小可以是 1GB，文件的第二部分的大小是 4GB，转而经编码的符号的数量可以是 1,024 乘以每一个文件部分的源符号的数量，其独立于各个文件部分是被划分成一个源块、16,384 个源块、还是 4,194,304 个源块。

[0151] 图 10 示出了使用 [RaptorQ-RFC-6330] 的简单 UEP 文件传输方法的性能。在该示例中，UEP 文件传输是针对于 30 个源分组（其使用 100 个修复分组进行保护）和 970 个源分组（其使用 900 个修复分组进行保护）。

[0152] 图 11 示出了相对于图 10 的简单 UEP 文件传输方法，UFD-UEP 文件传输方法提供的改进的示例。在该示例中，将 1MB 的文件划分成 32KB 的第一部分和 992KB 的第二部分。对于这两种方法，使用在 [RaptorQ-RFC-6330] 中所指定的 FEC 编码，用于携带经编码的符号的每一个分组中的大小是 1,024 个字节，发送了总共 2,048 个分组。

[0153] 对于图 11 中所示的简单 UEP 文件传输方法示例，对文件的第一部分和文件的第二部分进行独立地处理和传输，其中在这两种情况下，在每一个分组中携带大小为 1,024 个字节的准确地一个经编码的符号。用于该文件的第一部分的源在 32 个分组中进行携带，发送包含经编码的符号的总共 128 个分组。用于该文件的第二部分的源在 992 个分组中进行携带，发送包含经编码的符号的总共 1,920 个分组。

[0154] 对于图 11 中所示出的 UFD-UEP 文件传输方法，以组合的方式对文件的第一部分和文件的第二部分进行处理和传输，即，发送的每一个分组包含针对这两个部分中的每一个的经编码的符号，其中对于第一部分来说，经编码的符号的大小是 64 个字节，对于第二部分来说，经编码的符号的大小是 960 个字节。用于该文件的第一部分的源在 512 个分组中进行携带，所有 2,048 个分组都包含针对第一部分的经编码的符号。用于该文件的第二部分的源在 1059 个分组中进行携带（在源的最后一个分组中的经编码的符号，使用零来填充以达到全部符号大小为 992 个字节的第二部分的经编码的符号），所有 2,048 个分组都包含针对第二部分的经编码的符号。

[0155] 如图 11 中所观察到的，根据丢包率而言，对于文件的第二部分来说，简单 UEP 文件传输方法和 UFD-UEP 文件传输方法的恢复性能实际上是相同的，即，在两种情况下，对文件的第二部分进行公平地恢复，以一致地达到接近 48％的丢包率。另一方面，对于文件的第一部分而言，UFD-UEP 文件传输方法的恢复性能显著地比简单 UEP 文件传输方法更佳：简单 UEP 文件传输方法可以以小于 65％的丢包率来一贯地恢复文件的第一部分，而 UFD-UEP 文件传输方法可以以接近 75％的丢包率来一贯地恢复文件的第一部分。

[0156] 用于绑定的文件传输服务的通用文件传输方法和系统

[0157] 大部分的先前文件传输方法不支持绑定的文件传输，即，将多个文件传输成单一的绑定文件。传输几个文件的一种简单明了方法就是独立地传送每一个文件。但是，这种简单明了的方法具有一些缺点。例如，如果这些文件很小，则提供的保护与理想差距很远，这是由于：当包含针对每一个文件的经编码的符号的分组的数量很小时，丢包率统计存在较大的方差。

[0158] 图 12 示出了这种问题。在图 12 中，根据在传输期间，网络中的丢包的百分比，来示出 32KB 文件的文件传输的可靠性。在该文件传输示例中，符号的大小是 1,024，使用 [RaptorQ-RFC-6330] 中所指定的 FEC 编码，将该文件的 32 个源符号编码成 64 个经编码的符号，每一个经编码的符号使用一个单独的分组来发送。如在图 12 中所观察到的，由于这

种变化，在可以实现文件的可靠传输情况下的丢包百分比远远小于 50％。

[0159] 此外，如果对很多较小的文件进行独立地编码和发送，那么在可靠地接收所有文件的情况下的丢包百分比甚至更小。图 12 示出了用于 32 个文件的传输的这种行为，其中每一个文件的大小为 32KB，使用与在先前的段落中所描述的相同参数，独立于其它文件来执行各个文件的编码和传输。如可以观察到的，当丢包率低于大约 25％（其远远小于 50％）时，才可以可靠地实现全部 32 个文件的传输。

[0160] 可以如下所述地对 UFD-UEP 文件传输方法进行扩展，以便提供一种 UFD 绑定的文件传输方法。该 UFD 绑定的文件传输方法可以使用与 UFD-UEP 文件传输方法相同的方法，但不是发送同一文件的 d 个部分的传输，而是替代地发送：每一个部分是一个单独的文件，将 d 个文件传输成一个绑定。假定发送机希望提供大小分别为 $F_0, F_1, \cdots, F_{d-1}$ 的 d 个文件的绑定传输。该发送机 UFD 绑定的方法将分组大小 T 划分成大小为 $T_0, T_1, \cdots, T_{d-1}$ 的 d 个部分，因此 T 等于关于 i 的 T_i 之和。T 的这种划分是基于 $F_0, F_1, \cdots, F_{d-1}$，以及相应文件的优先级。

[0161] 比率 F_i/T_i 确定为了恢复第 i 文件，需要接收多少个分组（其假定使用理想的 FEC 编码来将该文件的第 i 部分保护成一个源块），因此比率 F_i/T_i 越小，则文件的第 i 部分的优先级越高。在实现时，可能需要比 F_i/T_i 稍微更多的分组，来恢复该文件的第 i 部分，例如，由于 FEC 编码是不完美的，并呈现某种接收开销，或者由于该文件的该部分被划分成多个源块，用于一些源块的经编码的符号按照与其它源块相比更高的比率丢失，或者由于 F_i/T_i 不是整数。如果期望将要传输的所有文件的优先级是相同的，那么设置 T_i，使得 $F_i/T_i \approx F/T$。对于发送方和接收机来说，该 UFD 绑定的文件传输方法的很多细节几乎与 UFD-UEP 文件传输方法相同，故进行了省略。

[0162] 可以对该 UFD 绑定的文件传输方法进行扩展，以便同时地提供绑定的文件传输和 UEP 文件传输，即，可以差别化地设置该绑定中的每一个文件的优先级。此外，该 UFD 绑定的文件传输方法可以在具有适当的信令的基础上，支持多个文件的具有优先级的传输和一个文件的一部分具有优先级的传输。例如，如果使用 UFD 绑定的文件传输方法，对三个对象进行编码和发送，那么前两个对象可以是具有不同优先级的同一文件的不同部分，第三对象可以是具有不同优先级的不同文件。接收机可以基于多种因素，来决定其对于恢复该绑定的文件中的哪个文件或者文件部分感兴趣，并且独立于其它文件或者文件的其它部分，只对这些文件或者文件的部分进行恢复。如本领域普通技术人员所应当认识到的，在阅读本发明公开内容之后，存在该 UFD 绑定的文件传输方法的多种可能的替代版本。

[0163] 举一个 UFD 绑定的文件传输的简单示例，假定要传输 32 个文件的绑定，每一个文件的大小为 32KB，各个文件具有相同的优先级。假定 T ＝ 1,024 字节。在该情况下，对于每一个 $i = 0, \cdots, 31$，T_i 的值＝ 32 个字节。每一个分组包含：针对这 32 个文件中的每一个文件的 32 字节经编码的符号，以及用于标识该分组中的 32 个经编码的符号的 UFSI。在该示例中，存在 1,024 个分组，其包含来自于 32 个相同大小的文件中的每一个文件的源符号，这些是 UFSI 范围为 0 到 1,023 的分组。假定在该示例中，针对每一个文件，生成 1,024 个另外的修复符号，并在 UFSI 范围为 1,024 到 2,047 的另外 1,024 个分组中进行发送。

[0164] 图 12 根据丢包率，示出了该 UFD 绑定的文件传输示例的恢复属性。在该示例中，可以以丢包率接近 50％，使用该 UFD 绑定的文件传输方法来可靠地恢复所有 32 个文件，其相对于近似 25％的丢包率，实现了一种本质的改进，这允许使用每一个文件的单独编码和

发送，来进行所有 32 个文件的可靠传输。

[0165] UFD 和 UEP 文件传输方法可以具有多种其它应用。例如，可以通过多播或者广播网络（例如，根据 3GPP eMBMS 标准的网络），来传输 DASH 格式的内容的分段。在该情况下，DASH 格式的内容的每一个分段可以包括：按照 1Mbps 进行编码的 8 秒的视频内容，即，每一个分段的大小是 1MB。

[0166] 当该 DASH 格式的内容最终由移动设备上的终端用户进行播放时，期望全保真度的这种播放（即，不具有伪影或者跳过或者缓冲），终端用户可能仅期望播放被传输的内容的一部分。为了在传输和播放时获得效率和灵活性（例如，能够将内容一次格式化成 DASH 格式，在不进行重新格式化的基础上，通过多种传输方式进行传输），期望将每一个分段传输成一个单独的文件。

[0167] 对于网络效率而言，期望的是，每一个 DASH 分段都是 FEC 保护的，并与其它分段进行完全交织地传输，其中，优选地可以使用 UFD 和 UEP 传输方法在分组中所提供的交织。例如，假定要传输 150 个分段，因此要传输的 DASH 内容的大小是包括 1200 秒的 150MB，或者等同地 20 分钟的播放呈现时间。假定在各分组的有效载荷为 1200 字节的分组中，发送该内容。那么，使用本申请所描述的 UFD 和 UEP 传输方法，可以将每一个分组的 8 字节分配给 150 个分段中的每一个，即，每一个分组携带这 150 个 DASH 分段中的每一个分段的一个 8 字节符号。那么，独立于丢包模式，可以在到达移动接收设备近似 131,072 个分组之后，即，这 150 个分段中的每一个分段到达 1MB 的数据之后，恢复各个 DASH 分段。在该示例中，在接收移动设备处，可以在无需对其它 DASH 分段进行解码的情况下，对这 150 个 DASH 分段中的每一个进行解码并进行可选地播放。

[0168] 再举一个本申请所描述的 UFD 和 UEP 文件传输方法的示例，假定传输通过多播或者广播网络（例如，根据 3GPP eMBMS 标准的网络）传送的 DASH 格式的内容，其中 DASH 内容包括视频分段和音频分段序列，每一个视频分段包括按照 1Mbps 编码的 8 秒视频内容（即，每一个视频分段的大小是 1MB），每一个音频分段包括按照 32Kbps 编码的 8 秒的相应音频内容（即，每一个音频分段的大小是 32KB），其中视频分段和相应的音频分段包含主要用于相同 8 秒的时间间隔的呈现时间的内容。

[0169] 在该示例中，DASH 内容将被流水化，即，每一个视频分段和相应的音频分段将按照其呈现时间的顺序，来按顺序地传输。假定在各分组的有效载荷为 1200 字节的分组中发送该内容。那么，使用本申请所描述的 UFD 和 UEP 传输方法，可以将每一个分组的 1160 字节分配给一个视频分段，将每一个分组的剩余 40 字节分配给相应的音频分段，即，每一个分组携带视频分段的一个 1160 字节符号和相应的音频分段的 40 字节符号。那么，独立于丢包模式，可以通过接收包含用于视频分段的近似 905 个分组，来恢复该视频分段，可以从包含用于相应的音频分段的符号的近似 820 个分组中，恢复该相应的音频分段。

[0170] 因此，整体上，与 8 秒时段的呈现时间相对应的任何 820 个分组（其存在相应的视频分段和音频分段）的接收，允许移动接收设备恢复该音频分段，另外 85 个分组的接收（或者总共 905 个分组）还允许相应的视频分段的恢复。在该示例中，与视频相比，对音频进行更多的保护，这在大多情况下是优选的，例如，其是由于视频播放的定时是通过音频播放的定时来指示的，在大多情况下，优选的是，如果没有接收到足够的数据来播放音频和视频二者，则至少能对音频进行播放。

[0171] 用于单播修复请求的本地 HTTP 支持的方法

[0172] 可以对文件或者文件部分进行组织并可用成具有相关联的 URL 的“HTTP 文件”，标准 HTTP web 高速缓存服务器可以使用该 URL 来存储和提供接收机对于该文件或者文件部分的访问。这种文件或者文件部分按照其原始顺序可用成 HTTP 文件，称为“原始顺序 HTTP 文件”。通常，一种用于单播修复请求的本地 HTTP 支持的方法，可以基于 SBN 和 ESI，将用于 HTTP 文件的源块的符号和子符号的修复请求，在 HTTP 文件中转换成标准 HTTP（例如，本地 HTTP/1.1）字节范围请求（其通常称为具有指定的字节范围的 HTTP 的 HTTP 部分 GET 请求）。这使标准 HTTP web 高速缓存服务器能够服务这些修复请求，避免大规模地部署专用的 HTTP web 高速缓存服务器的需要，其中这些专用的 HTTP web 高速缓存服务器理解如何将一个 HTTP 文件划分成源块和源块中的源符号。

[0173] 在这些场景中，当系统 FEC 编码在使用时，通常有利的是，仅在这些会话中的一些会话（例如，广播 / 多播会话）里发送修复符号，由于这允许接收机在单播修复请求中只请求源符号。这具有下面的优点 :只需要 HTTP 文件，或者该 HTTP 文件的简单重新排序转换（如下面所更详细描述的）被提供给单播修复服务器（其可以是标准 HTTP web 高速缓存服务器），使规定 HTTP 文件和分发该 HTTP 文件更简单的逻辑，由于这些逻辑独立于 FEC 编码。另一种优点在于 :如果在会话中没有发送源符号，则由于所有的源符号都“丢失”，因此接收机可以在单播修复请求中请求任何顺序的源符号，因为这允许针对连续序列的源符号的修复请求，因此其是优选的。例如，假定使用具有理想的恢复属性的 FEC 编码，一个部分的源块包括 1,000 个源符号，接收到该源块的 700 个修复符号（即使这些修复符号中的一些在传输过程中丢失，因此接收的修复符号的 ESI 的模式可能远远不是连续的）。随后，接收机可以在单播修复请求中，请求该源块的前 300 个连续源符号，对这些源符号与所述 700 个修复符号进行组合，以恢复该源块。如果这些源符号在 HTTP 文件中是连续的，那么可以使用单一 HTTP 字节范围请求来请求和接收所有 300 个连续的源符号。

[0174] 接收机不需要执行前缀请求（即，针对文件的以第一字节起始的、设置数量的字节的请求），但需要所有 UE 执行前缀请求，随着从 UE 所获得的请求发生极大地重叠，显著地提高 HTTP 服务器中的高速缓存效率，即使当存在很多接收设备，不同的接收设备经历任意不同的丢包模式（当通过广播或多播会话来获得分组）。

[0175] 基于所描述的这些利益中的一些，上面所描述的技术可以有利于网络或者网络设备向接收机说明仅正在广播修复符号。当发送该指示时，接收机不需要跟踪其接收到什么源符号，只需要跟踪其接收的符号的数量。例如，基于接收的符号总数，接收机可以随后确定要从修复服务器请求多少更多的符号。为了提高高速缓存效率，接收机可以被配置为 :基于其对于只有修复符号被广播的指示的了解，针对丢失数量的源符号，进行前缀请求。

[0176] 替代地，可以向进行前缀请求的接收机给予单独的信令指示。在另一个实施例中，可以对接收机进行配置，使得只要接收机具有在丢失的源块的不同子集之间进行选择的灵活性，其就始终选择具有最低符号索引值的丢失的源符号（即，最靠近源块或者子块的开始的源符号）。这可以是通过广播信道来发送源符号和修复符号的情况。

[0177] 在另一个实施例中，接收设备跟踪接收的符号标识符、ESI，并基于例如 FEC OTI 或者在接收分组中携带的 FEC 有效载荷 ID，来确定其是否接收与仅修复符号相对应的 ESI。如果是这种情况，则这些接收机可以进行前缀请求。在该实施例中，无需使用显式信令，接

收机将在修复请求中，发送针对源块和子块的前缀的请求（如果其仅接收修复符号的话）。即使在接收设备从广播信道接收修复符号的情况下，根据接收的符号的模式，以及进行简单请求和潜在地接收已经通过广播信道接收的冗余数据之间的平衡，这些修复请求仍然是很大或者完全的前缀请求。

[0178] 当发送仅修复符号被广播的信息，并且发送接收机进行前缀请求的信息时，在使用 HTTP 字节范围文件修复请求或者基于符号的文件修复请求时，存在一些优点。因此，这些方法可以应用于支持任意一种、或者这两种请求格式的接收机和网络。

[0179] 用于指示仅修复符号被广播的信令，可以在带外发送（通过单播）或者带内发送（通过广播）。根据期望的粒度级别，该指示可应用于服务级别、会话级别、媒体级别或者甚至文件级别。例如，可以将该信令增加在用户服务描述（“USD”）中。USD 可以指示：为了传输所有的识别的服务，仅修复符号被广播。此外，还可以将该信令增加在媒体呈现描述（“MPD”）中，其中 MPD 指示使用仅修复符号来广播哪个媒体。可以将该信令增加在会话描述协议（“SDP”）中，以指示其应用于整个会话的传输，还是应用于该会话中的媒体组成部分。此外，还可以将该信令增加在文件描述表（“FDT”）中，以指示修复符号的广播仅在文件级别上应用。

[0180] 用于需要接收机进行针对修复数据的前缀请求的信令，还可以通过带内和带外传输来进行传送。此外，根据期望的粒度级别，该指示还可以应用于服务级别、会话级别、媒体级别或者甚至文件级别。

[0181] 因此，存在着用于提供这种指示的多种方法和装置，接收机可以被编程和 / 或配置为：在接收到这种指示和没有接收到这种指示的情况下，进行差别化响应。

[0182] 该请求可以是针对于特定数量的字节或者符号。在一些情况下，接收机将基于接收的修复符号的数量，以及预期在将要进行接收的文件或对象中包含的源符号的数量，来确定要请求的源符号的数量。在其它情况下，接收机可以执行调度操作，其中接收机确定如何使用其已经接收到的仅修复符号来进行解码，并因此可以注意到需要的更具体数量的源符号。例如，修复符号中的一些是其它修复符号的冗余，在该情况下，可能需要更多的源符号。在其它实例中，也许会发现需要更少的源符号。

[0183] 接收机可以基于 FEC OTI，将针对与特定的 (SBN,ESI) 对相对应的原始顺序 HTTP 文件的源符号的请求，转换成 HTTP 字节范围请求。例如，假定针对一个文件的 FEC OTI 是 (F,Al,T,Z,N)，要请求的源符号的 SBN 是 A，针对源符号的 ESI 是 B，为了简单起见，假定 N = 1，即在该示例的文件结构中，不再将源块划分成子块。随后，接收机可以使用例如在 [RaptorQ-RFC-6330] 的第 4.4 节中所描述的方法来确定 (KL,KS,ZL,ZS)，其中第一 ZL 源块具有 KL 个源符号，剩余的 ZS 个源块具有 KS 个源符号。随后，基于 A、B 和符号大小 T，接收机可以确定该文件中的源符号的起始字节索引 SI 是如式 1 中所示，该文件中的源符号的结束字节索引是 EI = SI+T-1。随后，接收机可以使用针对于源符号的标准 HTTP 字节范围请求，来请求字节 SI 到 EI。

[0184] $$SI = T*(KL*\min\{A,ZL\}+KS*\max\{A-ZL,0\}+B) \quad (式\ 1)$$

[0185] 存在对于这些方法的多种扩展和改进，如本领域普通技术人员所认识到的。例如，如果在没有使用子块划分的情况下，从原始顺序 HTTP 文件请求多个连续的源符号，则可以进行单一 HTTP 字节范围请求。再举一个例子，HTTP web 高速缓存服务器除了具有或者替代

原始顺序HTTP文件之外，还可以具有包含修复符号的文件，或者原始顺序HTTP文件可以已经被扩展到包含修复符号，接收机可以使用类似于本申请所描述的那些的方法，来进行针对修复符号的HTTP字节范围请求。再举一个增强的示例，可以将这些方法扩展到使用子块划分的情况，使用类似的方法，如本领域普通技术人员所认识到的。例如，接收机可以使用例如在[RaptorQ-RFC-6330]的第4.4节中所描述的方法来确定(TL,TS,NL,NS)，其中源块的前NL子块的大小为TL*A1，源块的剩余NS个子块的大小为TS*A1。那么，基于A、B和符号大小T，接收机可以确定该文件中的与源符号相对应的N个子符号的起始和结束字节索引，使用标准HTTP字节范围请求，进行针对这些字节的请求。

[0186] 再举一个例子，接收机可以基于FEC OTI，将针对于与特定的UFSI相对应的文件或者文件部分的源符号的请求，转换成HTTP字节范围请求。

[0187] 下面的方法可以通过接收机恢复HTTP文件的符号，同时使用标准HTTP web高速缓存服务器来向接收机传输所请求的字节范围请求，来极大地增强使用标准HTTP字节范围请求的效率。

[0188] 一种用于支持如上所述的HTTP字节范围请求的简单明了方法，是使用原始顺序HTTP文件。该方法具有下面的优点：不需要原始文件或者文件部分的转换，来生成针对HTTP web高速缓存服务器的原始顺序HTTP文件，但可能需要多种不同的HTTP字节范围请求来请求源符号，即使当针对每一个源块，仅请求连续的源符号。为此，存在至少两个原因：(1)可能存在多个源块，存在来自于各个源块的丢失的源符号，在该情况下，针对每一个源块，可能需要单独的字节范围请求；(2)可能每一个源块存在多个子块，因此甚至从一个源块请求单一的源符号，也需要针对包括源符号的所述多个子符号的多个HTTP字节范围请求，这是由于一个源符号的子符号在原始顺序HTTP文件中是非连续的。

[0189] 一种用于支持上面所描述的HTTP字节范围请求的有利方法，首先基于原始文件或者文件部分的FEC OTI，将该文件或者文件部分重新组织成一种新格式(本申请称为“UFSI顺序HTTP文件”)。当存在多个源块和/或每一个源块存在多个子块时，该方法是有用的。UFSI顺序HTTP文件中的数据的顺序是：UFSI0的文件符号、UFSI1的文件符号、UFSI2的文件符号等，即，将UFSI顺序HTTP文件中的数据的顺序组织成，根据增加的连续UFSI进行排序的文件符号，如FEC OTI所确定的。URL可以与UFSI顺序HTTP文件相关联，可以将URL提供给HTTP web高速缓存系统。随后，接收机可以根据需要，使用HTTP字节范围请求，来请求UFSI顺序HTTP文件的一部分。UFSI顺序HTTP文件的一种优点在于：如果接收机从每一个源块接收到近似相同数量的修复符号，那么获得足够的源符号来恢复原始文件或者文件部分而所需要的HTTP字节范围请求的数量可以是最小的，例如，如果接收到每一个源块的准确相同数量的修复符号，那么一个HTTP字节范围请求可以是足够的。例如，针对UFSI顺序HTTP文件的前L*T*Z个连续字节的请求，足够用于从原始文件或者文件部分的Z个源块中的每一个源块接收前L个大小为T的源符号。如果在针对Z个源符号的每一个的一个或多个会话中接收到K-L个修复符号，并且FEC编码具有理想的恢复属性，那么从该HTTP字节范围请求接收的L*T*Z个字节，足够用于对该HTTP文件进行FEC解码，其中在该示例中，推测K是在Z个源块中的每一个源块里的源符号的数量。

[0190] 另一种用于支持上面所描述的HTTP字节范围请求的有利方法，首先基于原始文件或者文件部分的FEC OTI，将该文件或者文件部分重新组织成一种不同的新格式(本申请

称为“SS 顺序 HTTP 文件”)。当每一个源块存在多个子块时,该方法是有用的,这是由于在该情况下,每一个源符号可能不是该原始文件或者文件部分的一个连续部分,即,一个源符号的子符号可以按照原始文件排序,分散在整个源块之中。SS 顺序 HTTP 文件中的数据的顺序是具有下面的顺序:第一源块的所有源符号、接着是第二源块的所有源符号、接着是第三源块的所有源符号等,即,将 SS 顺序 HTTP 文件中的数据的顺序组织成,根据在源块中的增加的连续 ESI 进行排序的源符号,其是按照增加的连续 SBN 顺序的源块的串联,如 FEC OTI 所确定的。URL 可以与 SS 顺序 HTTP 文件相关联,可以将 URL 提供给 HTTP web 高速缓存系统。随后,接收机可以根据需要,使用 HTTP 字节范围请求,来请求 SS 顺序 HTTP 文件的一部分。如果接收机从每一个源块接收到不同数量的修复符号,则 SS 顺序 HTTP 文件是特别有利的。例如,如果存在两个源块,每一个源块具有 1,000 个源符号,如果接收机从一个或多个会话接收到第一源块的 900 个修复符号,第二源块的 100 个修复符号,那么接收机可以执行一个针对 SS 顺序 HTTP 文件的前 T*100 个字节的请求,并接收第一源块的 100 个源符号,接收机可以进行另一个针对 SS 顺序 HTTP 文件的字节 T*100 到 T*1,900 - 1 的请求,接收第二源块的 900 个源符号,其中在该示例中,T 是用于这两个源块的字节的符号大小。

[0191] 此外,还可以使用刚刚所描述的两个方法的组合,即,通过针对接收机的不同 URL,UFSI 顺序 HTTP 文件和 SS 顺序 HTTP 文件是可用的,转而接收机可以向两个不同的格式的 HTTP 文件,使用 HTTP 字节请求的组合。例如,如果存在 10 个源块,每一个源块具有 1,000 个源符号,并且如果在一个或多个会话中,接收到前 8 个源块中的每一个的 800 个修复符号,接收到第 9 个源块的 820 个修复符号,接收到第 10 个源块的 500 个修复符号,那么接收机可以向 UFSI 顺序 HTTP 文件进行 HTTP 字节范围请求,以便接收 10 个源块中的每一个源块的 200 个源符号,向 SS 顺序 HTTP 文件进行另外的 HTTP 字节范围请求,以便接收第 10 个源块的另外 300 个源符号。在该示例中,接收机可以接收第 9 个源块的不需要的 20 个源符号(其假定具有理想恢复属性的 FEC 编码),但在一些情况下,与准确地请求每一个源块的所需要的数量的源符号(其需要更大数量的不同的 HTTP 字节范围请求)相比,使用更少数量的 HTTP 字节范围请求来请求一些源块的多于最小需要的字节,可以是更高效的。

[0192] 有利的是,发送接收机可以使用该 HTTP 字节范围请求来修复文件的时间。这实现将该特征逐渐引入到当前针对文件修复使用其它请求消息的已部署网络中。计划将其网络操作转换到传统的基于 HTTP 的文件修复服务器的系统运营商,可以在该特征在其整个网络或者漫游的合作伙伴网络上被完全地支持之前,开始部署能够进行 HTTP 字节范围请求的接收机。随后,随着部署的具有 HTTP 能力接收机的数量增加,运营商可以开始部署 HTTP 服务器,以便与来自这些终端的估计的请求有效载荷相匹配。在特定的网络中可以使用信令来向接收机指示:其可以在该网络中使用 HTTP 字节范围请求。

[0193] 用于向网络发送 HTTP 字节范围请求的支持的另一种方法,是除了提供修复服务器 URI 之外,还提供指示该服务器只支持 HTTP 字节范围请求的描述符(例如,将服务器 URI 标签成“byteRangeServiceURI”类型对象)。在已经部署了仅接受基于符号的请求的修复服务器的网络中,这种方法是有用的,其通常用“serviceURI”描述符来标识。这种新的“byteRangeServiceURI”的规定,使新的接收机能够使用与这些服务器的字节范围请求,同时防止不认识该描述符的传统接收机从这些 HTTP 服务器错误地进行基于符号的请求。

[0194] 用于向网络发送 HTTP 字节范围请求的支持的另一个实施例,是除了提供修复服

务器URI之外，还提供指示该服务器是只支持HTTP字节范围请求，还是支持HTTP字节范围请求和其它请求格式（例如，基于符号的请求）的描述符。在另一个实施例中，该描述符指示三种可能中的一种：(1)该修复服务器是否只支持HTTP字节范围请求；(2)该修复服务器是否只支持另一种类型的请求格式（例如，基于符号的请求）；(3)该修复服务器是否同时支持HTTP字节范围请求和其它请求格式。此外，该描述符还指示这些请求格式中的哪一种是优选的，或者需要进行使用（如果该接收机支持的话）。可以将服务器所支持的请求格式的这种描述符，连同修复服务器URI的列表一起发送。

[0195] 此外，还可以向接收机发送另一个描述符（例如，“useFileURI”元素），其指示其可以使用HTTP字节范围请求，直接从存储文件的内容服务器（例如，互联网上的位置）获取该文件的一部分或者该文件的全部。这实现了一种体系结构，在该体系结构中，网络运营商（例如，移动网络运营商）不需要部署专用的文件修复服务器，而可以依赖于这些文件的内容服务器来提供修复过程的源数据。当通过广播信道下载文件时，可以使用包括文件的URI(例如，www.<news providersite>.com/latest_news.3gp)的文件描述符表来发送这些文件。“useFileURI”元素的存在将向终端指示其可以使用HTTP字节范围请求，来从www.<news providersite>.com/latest_news.3gp获取源数据。因此，不是向接收机指出www.<mobile-oper-cached-file-server-of-news>.com/newsfile_003.3gp，而是接收机可以直接进入该内容已经可用的站点，其通常用于使用传统的HTTP协议来访问该内容的浏览器。可以存在发送文件格式的另一种元素，例如原始顺序或者UFSI顺序或者SS顺序或者扩展的原始顺序HTTP文件格式。

[0196] 独立于专用修复过程，除了广播的内容之外，内容传输系统可以包括用于向接收机发送信号的装置，文件的内容的一些或者全部是单播可用的。这种信令可以是提供成广播的一部分的数据，其中该数据指向单播位置。例如，该广播数据可以包括文件可用性的指示和用于该文件的URL。不同的实施例可以差别化地使用这些可用的文件。例如，被广播的URL所指示的文件，可以是用于修复过程的文件，其用于加速下载和/或支持更快速的初始播出。

[0197] 在一些实施例中，文件存在于一个以上的网络或者物理位置。为此，广播信道上的信令可以包括：指示可访问该文件的多个位置的URL列表。在一些实施例中，存在实际指向同一物理位置，因此指向相同的文件或对象的多个URL。

[0198] URL所指示的源可以是受到限制的，例如，只可用于有限数量的时间。有益的是，实现对象（其在广播分发中通过标识符来指定）到可能的多个URL的映射。对于这些URL中的每一个，可以提供另外的信息，例如，一个文件在一个位置可使用HTTP协议来物理地访问，在其它位置或者相同的位置可使用其它协议来访问的指示。这种另外信息的其它示例包括：该资源可在特定的URL处访问的时间、和/或关于在访问该资源的多个URL之间选择一个URL时的优先级的规则（例如，优先级列表、或者取决于接入网络可用性的优先级列表）等。

[0199] 还可以连同服务器URI列表来发送另一种描述符，以便提供当接收机请求修复数据时，其应当首先尝试哪个服务器或者域的优先级。在一个优选的实施例中，“priorityURI”可以与某些URI相关联，以便指示：接收机在选择服务器时，其应当具有优先级。这实现了两种层级的优先级划分，从而接收机首先尝试具有“priorityURI”

的服务器，仅当具有优先级的服务器无响应时，才向其它服务器发送请求。在另一个实施例中，可以将服务器URI的组结构化成具有按照优先级下降的顺序所列出的组的“priortyGroups”。这向接收机指出其应当首先从最高优先级组中进行选择。如果针对所有这些服务器的请求都失败，则接收机转到从下一组的服务器中进行选择等，从而实现多个层级的优先级划分。

[0200] 现在描述可以向接收机发送描述符和服务器URI列表的两种通用方法。在一种方法中，该信令是通过针对于所有接收机的广播信道（带内），在另一种方法中，该信令是通过针对每一个接收机的单播信道（带外）。可以将该信息增加到在文件修复特征中使用的多种现有消息类型，例如，关联过程描述（配置文件）、用户服务描述（“USD”）、媒体呈现描述（“MPD”）、会话描述协议（“SDP”）或者文件描述表（“FDT”）。

[0201] 当服务器被配置为发送FDT和USD，以便分发该信息，接收机被配置为读取和理解这些FDT和USD时，这提供了用于发送各种有用信息的工具。

[0202] 例如，该信息可以在FLUTE中的文件元素中包括“替代内容位置”，其可以包括用于相同文件的多个替代URL。可以对该元素进行扩展，以便还增加其它参数（例如，资源可用性的指示符、其可用的格式、其可用的时间、优先级等）。

[0203] 再举一个例子，可以将“BaseURL”元素包括在FDT元素中（并因此在文件元素中不需要），以便向接收机指示可以根据这些BaseURL元素中的任何一个来解析FDT中的每一个文件。也可以存在如上的标志和定时。这可以覆盖多种部署选项。在一种变型中，在USD层级而不是FLUTE上，呈现该BaseURL元素。因此，可以将USD提供成用于进行MBMS的BaseURL解析，同时与FLUTE相兼容的方式。

[0204] 在另一个实施例中，将一些可选元素（例如，“替代内容位置1”元素和“替代内容位置2”元素）引入到文件传输表(FDT)的文件元素中。这些元素中的每一个元素可以用于在内容服务器上或者在公共专用服务器上提供文件的URL。在一个实施例中，当存在这些元素中的任意一个时，在从基于符号的文件修复服务器进行请求之前，终端对这些URL划分优先级。此外，还可能的是，当这两个元素存在时，在“替代内容位置2”元素之前，选择“替代内容位置1”元素。这些URL是绝对URL或者如RFC3986中所描述的相对引用。

[0205] 在另一个实施例中，可选的“可用时”元素与上面的“替代内容位置x”属性中的每一个相关联，以便指示可用时间（如果知道的话）。

[0206] 在另一个实施例中，可以在文件传输表中使用可选的元素“Base-URL-1”和“Base-URL-2”。当存在时，“Base-URL-1”提供用于对FDT文件元素的任何“替代内容位置1”元素中包括的相对引用进行解析的基URL。当存在时，“Base-URL-2”提供用于对FDT文件元素的任何“替代内容位置2”元素中包括的相对引用进行解析的基URL。

[0207] 可以考虑其它选项，也可以考虑上面选项的组合。例如，将文件格式类型嵌入在根据可用的文件格式来组合的其它URL中。

[0208] 随着运营商的网络中的终端自动产生，更多的接收机能够进行HTTP字节范围请求，运营商可以将文件修复容量的大部分转移到传统HTTP服务器，并通过该信令方法来指示其能力。

[0209] 在接收到某些修复服务器支持HTTP字节范围请求的指示之后，支持HTTP字节范围请求的终端可以使用该字节范围消息，来从支持服务器中的一个请求数据。

[0210] 在另一个实施例中（在该实施例中，描述符指示要使用的请求格式的优选或者要求），接收机遵循所指示的该优选或者要求。

[0211] 另一种方法是向接收机发送：修复服务器支持 HTTP 字节范围请求，而无需标识哪个具体的服务器具有该能力。这种信令指示所有修复服务器都能够只进行字节范围请求、或者所有都能够进行 HTTP 字节范围请求和其它请求消息格式（例如，基于符号的请求）。

[0212] 在另一个实施例中，该信号指示如上面所列出的三种可能中的一种：(1) 所有修复服务器是否只支持 HTTP 字节范围请求；(2) 该修复服务器是否只支持另一种类型的请求格式（例如，基于符号的请求）；(3) 该修复服务器是否同时支持 HTTP 字节范围请求和其它请求格式。当使用这种简化的信令时，运营商可以在向接收机发送该信息之前，对所有的文件修复服务器进行升级，以支持所指示的请求格式。此外，还可以发送文件格式类型，例如，原始顺序、UFSI 顺序、SS 顺序、扩展的原始顺序或者其它类型的 HTTP 文件格式。如果没有显式地发送文件格式类型，则可以存在缺省的文件格式类型，例如，在没有发送文件格式类型的情况下，原始顺序 HTTP 文件格式可以是缺省文件格式类型。

[0213] 类似于描述符和服务器 URI 信息，也可以使用两种通用方法来向接收机发送该信令。在一种方法中，该信令是通过针对于所有接收机的广播信道（带内），在另一种方法中，该信令是通过针对每一个接收机的单播信道（带外）。可以将该指示增加到在文件修复特征中使用的多种现有消息类型中，例如，关联过程描述（配置文件）、用户服务描述（“USD”）、媒体呈现描述（“MPD”）、会话描述协议（“SDP”）或者文件描述表（“FDT”）。

[0214] 现在描述可以向接收机发送描述符和服务器 URI 的列表的两种通用方法。

[0215] 如本领域普通技术人员所认识到的，存在这些方法的多种变型。例如，原始顺序 HTTP 文件、UFSI 顺序文件和 SS 顺序 HTTP 文件全部都可用于请求。再举一个例子，可以对 SS 顺序 HTTP 文件进行分割，并可用成多个 HTTP 文件，一个这种 HTTP 文件对应于每一个源块。举一些变型的其它示例，可以使用不同于基于 HTTP 的方法和协议的方法和协议（例如，基于 RTP/UDP 的方法），或者可以使用建立在 UDP 之上的专有协议。

[0216] 硬件系统和示例

[0217] 上面所描述的方法和系统可以用硬件、软件和 / 或包含程序代码或者其它指令的适当组织的计算机可读介质来实现。程序代码可以提供在非临时性介质上，或者发送成下载等，以便在适当的硬件平台上执行。本申请提供了可以使用上面的内容的系统的一些示例。

[0218] 图 13 是使用多阶段编码的通信系统 1300 的框图，如可以用于作为文件传输系统的一部分来发送分组，如本申请所描述的。当然，也可以使用其它编码和 / 或硬件。

[0219] 在通信系统 1300 中，将输入文件 1301 或者输入流 1305 提供给输入符号生成器 1310。输入符号生成器 1310 根据该输入文件或者流，生成一个或多个输入符号的序列 (IS(0)、IS(1)、IS(2)、…)，其中每一个输入符号具有一个值和一个位置（在图 13 中描述成用括号括起来的整数）。如上面所解释的，用于输入符号的可能值（即，其字母表）通常是 2^M 个符号的字母表，使得每一个输入符号编码对应于输入文件的 M 个比特。通常，M 的值通过使用通信系统 1300 来确定，但通用系统可以包括针对输入符号生成器 1310 的符号大小输入，使得 M 可以在使用之间发生变化。将输入符号生成器 1310 的输出提供给编码器 1315。

[0220] 静态密钥生成器 1330 产生静态密钥的流 S_0、S_1、…。生成的静态密钥的数量通常是受限的，其取决于编码器 1315 的具体实施例。随后，将更详细地描述静态密钥的生成。动态密钥生成器 1320 针对要由编码器 1315 生成的每一个输出符号，生成动态密钥。对每一个动态密钥进行动态生成，使得用于相同输入文件的动态密钥的很大部分是唯一的。随机数生成器 1335 可以向生成器 1320 和 / 或生成器 1330 提供输入。例如，Luby I 描述了可以使用的密钥生成器的实施例。将动态密钥生成器 1320 和静态密钥生成器 1330 的输出提供给编码器 1315。

[0221] 根据动态密钥生成器 1320 所提供的每一个密钥 I，编码器 1315 从输入符号生成器所提供的输入符号，生成具有值 B(I) 的输出符号。下面将更详细地描述编码器 1315 的操作。每一个输出符号的值是基于其密钥、基于输入符号中的一个或多个的某种函数、以及根据输入符号所计算的可能的一个或多个冗余符号来生成的。导致特定的输出符号的输入符号和冗余符号的集合，本申请称为输出符号的“关联符号”或者仅其的“关联”。该函数（“值函数”）和关联的选择，是根据下面所更详细描述的处理来进行的。通常（但不是始终），M 对于输入符号和输出符号来说是相同的，即，其均对应于相同数量的比特的编码。

[0222] 在一些实施例中，编码器 1315 使用输入符号的数量 K 来选择所述关联。如果 K 是提前未知的，例如当该输入是流媒体文件时，K 可以只是某个估计量。此外，编码器 1315 还可以使用值 K，来为输入符号和编码器 1315 所生成的任何中间符号来分配存储空间。

[0223] 编码器 1315 向发送模块 1340 提供输出符号。此外，还从动态密钥生成器 1320 向发送模块 1340 提供每一个这种输出符号的密钥。发送模块 1340 发送输出符号，并根据使用的密钥方法，发送模块 1340 还可以通过信道 1345 向接收模块 1350 发送关于所发送的输出符号的密钥的某种数据。假定信道 1345 是擦除型信道，但对于通信系统 1300 的适当操作来说，这并不是必须的。

[0224] 模块 1340、1345 和 1350 可以是任何适当的硬件组件、软件组件、物理介质或者其任意组合，只要发送模块 1340 用于向信道 1345 发送输出符号和关于其密钥的任何必需数据，接收模块 1350 用于从信道 1345 接收符号和关于其密钥的潜在某种数据。K 的值（如果使用其来确定所述关联的话）可以通过信道 1345 来发送，或者其可以通过编码器 1315 和解码器 1355 的协商来提前设置。在图 13（和本申请的其它地方）所示出的其它元素（无论其被描述成模块、步骤、处理等），也可以使用硬件、软件和 / 或在电子可读介质上存储的程序代码来实现。

[0225] 如上面所解释的，信道 1345 可以是实时信道，例如，通过互联网或者广播链路从电视发射机到电视接收者的路径或者从一个点到另一个点的电话连接，或者信道 1345 可以是诸如 CD-ROM、磁盘驱动器、Web 站点等之类的存储信道。信道 1345 甚至可以是实时信道和存储信道的组合，例如，当一个人通过电话线从一个个人计算机向互联网服务提供商 (ISP) 发送输入文件时所形成的信道，该输入文件存储在 Web 服务器上，并随后通过互联网发送给接收者。

[0226] 由于假定信道 1345 是擦除型信道，因此通信系统 1300 不假定离开接收模块 1350 的输出符号和进入发送模块 1340 的输出符号之间的一对一对应关系。事实上，当信道 1345 包括分组网络时，通信系统 1300 甚至不能够假定在通过信道 1345 来发送时，保持任何两个或更多分组的相对顺序。因此，输出符号的密钥通过使用上面所描述的密钥方案中的一个

或多个来确定，不需要通过这些输出符号离开接收模块 1350 的顺序来确定。

[0227] 接收模块 1350 向解码器 1355 提供输出符号，任何数据接收模块 1350 接收关于向动态密钥重新生成器 1360 提供的这些输出符号的密钥。动态密钥重新生成器 1360 重新生成用于所接收的输出符号的动态密钥，将这些动态密钥提供给解码器 1355。静态密钥生成器 1363 重新生成静态密钥 S_0、S_1、…，并将其提供给解码器 1355。静态密钥生成器访问在编码和解码处理期间使用的随机数生成器 1335。这可以具有访问相同的物理设备的形式（如果随机数是在该设备上生成的话），或者具有访问用于生成随机数以实现相同行为的相同算法的形式。解码器 1355 使用动态密钥重新生成器 1360 和静态密钥生成器 1363 所提供的密钥以及相应的输出符号，来恢复输入符号（同样的 IS(0)、IS(1)、IS(2)、…）。解码器向输入文件重组器 1365 提供所恢复的输入符号，重组器 1365 生成输入文件 1301 或者输入流 1305 的拷贝 1370。

[0228] 可以使用多个接收机和 / 或多个发送方来进行文件传输。在图 14-15 中示出这些概念。

[0229] 图 14 示出了一个接收机 2402 通过三个信道 2406，从三个发射机 2404（其分别地表示为“A”、“B”和“C”）接收数据的布置。可以使用该布置对可用于接收机的带宽增至三倍，或者处理发射机没有足够长的可用时间来从任何一个发射机获得完整的文件。如上所述，每一个发射机 2404 发送值 S() 的流。每一个 S() 值代表输出符号 B(I) 和密钥 I，上面解释了其的使用。例如，值 $S(A,n_A)$ 是在发射机 2402(A) 处所生成的输出符号的序列中的第“n_A”个输出符号和第“n_A”个密钥。来自于一个发射机的密钥序列，优选地与来自其它发射机的密钥序列不相同，使得发射机不会重复工作。根据序列 S() 是发射机的函数的事实，在图 14 中示出了该内容。

[0230] 应当注意，发射机 2402 不需要为了不进行重复工作，而进行同步或者协调。事实上，在没有协调的基础上，每一个发射机可能处于其序列中的不同位置（即，$n_A \neq n_B \neq n_C$）。

[0231] 在图 15 中，将一个输入文件 2502 的拷贝提供给多个发射机 2504（在图中示出了其中的两个）。发射机 2504 通过信道 2506 向接收机 2508 独立地发送根据输入文件 2502 的内容所生成的输出符号。在接收机的解码器能够恢复整个输入文件之前，所示出的两个发射机中的每一个发射机可以只需要发送 (K+A)/2 个输出符号。

[0232] 使用两个接收机和两个发射机，接收机单元 2510 所接收的信息的总量可以是在一个信道 2506 上可用的信息的四倍。该信息量可以小于单一信道信息的四倍（例如，如果发射机向两个接收机广播相同的数据的话）。在该情况下，接收机单元 2510 处的信息的量至少是两倍（通常是更多倍数，如果在任何信道中丢失有数据的话）。应当注意的是，即使这些发射机只广播一个信号，但接收机处于不同的时间的视角，有利的是一个以上的接收机对各个发射机进行监听。在图 15 中，接收机单元 2510 执行与图 13 中所示出的接收模块 1350、解码器 1355、动态密钥重新生成器 1360 和输入文件重组器 1365 的功能相类似的功能。

[0233] 在一些实施例中，在具有两个编码器的一个计算设备中，对输入文件 2502 进行编码，使得计算设备可以为一个发射机提供一个输出，为另一个发射机提供另一个输出。在阅读了本发明之后，这些示例的其它变型应当是显而易见的。

[0234] 应当理解的是，本申请所描述的编码装置和方法也可以用于其它通信情形，并且

不限于诸如互联网之类的通信网络。例如，光盘技术还使用擦除型和纠错编码来处理划伤磁盘的问题，在其上存储信息时，将通过使用链式反应码来获益。再举一个例子，卫星系统可以使用擦除型编码，以便平衡用于传输的功率需求，通过减少功率来有目的地允许更多的差错，链式反应编码在该应用情形下是有用的。此外，已使用擦除型编码来开发 RAID（独立磁盘冗余阵列）系统，以便可靠地进行信息存储。因此，本发明证明在诸如上面的示例之类的其它应用中是有用的，其中在这些情形下，使用编码来处理潜在有损数据或错误数据的问题。

[0235] 在一些优选的实施例中，向两个或更多的多目的计算机器提供用于执行上面所描述的通信过程的指令集（或者软件），其中这些计算机器通过可能的有损通信介质进行通信。机器的数量可以从一个发送方和一个接收者到任意数量的发送和 / 或接收的机器进行变化。连接这些机器的通信介质可以是有线、光纤、无线等。上面所描述的通信系统具有多种用途，这些根据本申请的描述将是显而易见的。

[0236] 使用 HTTP 流服务器的块请求流媒体系统，可以如上所述地传输文件。下面，将描述该系统的一个示例性实现。使用 HTTP 流媒体，源可以是标准的 web 服务器和内容传输网络 (CDN)，可以使用标准的 HTTP。

[0237] 在客户端，可以使用 HTTP，针对各个分段进行请求，客户端可以将这些分段无缝地拼接在一起。HTTP 流媒体的优点包括 ：使用标准化和现有的基础设施。

[0238] 图 16 示出了可以传送文件的块请求流媒体系统的简化图。在图 16 中，示出了块流系统 1600，其包括块服务基础设施（“BST”）1601，该块服务基础设施（“BST”）1601 转而包括摄取系统 1603，该摄取系统 1603 用于摄取内容 1602、准备该内容，通过将该内容存储到可由摄取系统 1603 和 HTTP 流服务器 1604 访问的内容存储器 1610，来对该内容进行封装以便由 HTTP 流服务器 1604 进行服务。如图所示，系统 1600 还可以包括 HTTP 高速缓存 1606。在操作中，诸如 HTTP 流客户端之类的客户端 1608 向 HTTP 流服务器 1604 发送请求 1612 并且从 HTTP 流服务器 1604 或 HTTP 高速缓存 1606 接收响应 1614。在各种情况下，图 16 中所示的元件可以至少部分地用软件来实现，其中所述软件包括在处理器或者其它电子器件上执行的程序代码。

[0239] 如图 17 中所示，可以将媒体块存储在块服务基础设施 1601(1) 中，该块服务基础设施 1601(1) 可以是例如 HTTP 服务器、内容传送网络设备、HTTP 代理、FTP 代理或者服务器、或者某种其它媒体服务器或系统。块服务基础设施 1601(1) 连接到网络 1722，该网络 1722 可以是例如互联网协议（“IP”）网络（例如，互联网）。将块请求流系统客户端示出为具有六个功能组件，也叫做 ：块选择器 1723，其被提供有上面所描述的元数据，并且执行要从由元数据所指示的多个可用块中选择要请求的块或者部分块的功能 ；块请求器 1724，其从块选择器 1723 接收请求指令，并且通过网络 1722 执行向块服务基础设施 1601(1) 发送针对所规定的块、一个块的一部分或者多个块的请求所必需的操作，以及转而接收包括块的数据 ；以及块缓冲器 1725、缓冲器监测器 1726、媒体解码器 1727、以及有助于媒体消费的一个或多个媒体转换器 1728。

[0240] 将块请求器 1724 所接收的块数据传送到用于存储媒体数据的块缓冲器 1725，以进行临时存储。替代地，可以直接将所接收的块数据存储到块缓冲器 1725。块缓冲器 1725 向媒体解码器 1727 提供媒体数据，媒体解码器 1727 对该数据执行为了向媒体转换器 1728

提供适当输入所必需的这些转换，该媒体转换器 1728 以适当的形式将媒体呈现给用户或者其它消费。媒体转换器的示例包括视觉显示设备（例如，在移动电话、计算机系统或者电视中所建立的那些），并且还可以包括诸如扬声器或耳机之类的音频呈现设备。

[0241] 缓冲器监测器 1726 接收涉及块缓冲器 1725 的内容的信息，并基于该信息和可能的其它信息，提供对块选择器 1723 的输入，该块选择器 1723 用于确定要请求的块的选择，如本申请所描述的。

[0242] 块请求流系统 (BRSS) 包括对一个或多个内容服务器（例如，HTTP 服务器、FTP 服务器等）进行请求的一个或多个客户端。摄取系统包括一个或多个摄取处理器，其中摄取处理器接收内容（实时地或非实时地），对于由 BRSS 所使用的内容进行处理，并且将该内容以及还有可能连同由摄取处理器所生成的元数据存储到可由内容服务器访问的存储器中。

[0243] BRSS 还可以包含与内容服务器进行协调的内容高速缓存。内容服务器和内容高速缓存可以是接收针对具有 HTTP 请求形式的文件或分段的请求的传统 HTTP 服务器和 HTTP 高速缓存，其中所述 HTTP 请求包括 URL 并且还可以包括字节范围，以便请求与该 URL 所指示的所有文件或分段相比更少的内容。客户端可以包括传统 HTTP 客户端，该传统 HTTP 客户端进行 HTTP 服务器的请求并且处理对这些请求的响应，其中 HTTP 客户端是由新型客户端系统驱动的，该新型客户端系统形成请求，将这些请求传送到 HTTP 客户端，从 HTTP 客户端获得响应，对这些响应进行处理（或者存储、转换等）以便将这些响应提供给呈现播放器，以便由客户端设备进行播放。一般情况下，客户端系统事先不知道将需要哪些媒体（这是因为这些需求依赖于用户输入、用户输入的变化等），所以其被称为“流”系统，在所述“流”系统中，一接收到媒体或者在接收到媒体后不久，就“消费”该媒体。结果，响应延迟和带宽约束可能造成呈现的延迟，例如，造成呈现的暂停（由于流赶上用户在消费该呈现所处的地方）。

[0244] 为了提供能觉察到具有良好质量的呈现，可以在 BRSS 中（在客户端终端处、在摄取终端处、或者二者）实现多个细节。在一些情况下，实现的细节是在考虑或处理位于网络的客户端 - 服务器接口的情况下完成的。在一些实施例中，客户端系统和摄取系统都意识到增强，而在其它实施例中，仅一方意识到增强。在这些情况下，整个系统从增强中获益（即使一方没有意识到该情况），而在其它情况下，只有双方都意识到时才产生该益处，但是当一方没有意识到时，其仍然进行操作而不会失败。

[0245] 如图 18 中所示，可以根据各种实施例，将摄取系统实现成硬件和软件组件的组合。该摄取系统可以包括能被执行以使系统执行本申请所讨论的方法中的任何一个或多个的指令集。可以将该系统实现成具有计算机形式的特定机器。该系统可以是服务器计算机、个人计算机 (PC)、或者能够执行指令集（串行或其它方式）的任何系统，其中该指令集规定了要由该系统采取的动作。此外，虽然仅示出了单个系统，但是术语“系统”应当还包括 :单独地或者联合地执行一个（或多个）指令集以执行本申请所讨论的方法中的任何一个或多个方法的任何系统集合。

[0246] 摄取系统可以包括摄取处理器 1802（例如，中央处理单元 (CPU)）、存储器 1804 以及磁盘存储器 1806，其中存储器 1804 可以在执行期间存储程序代码，所有这些部件通过总线 1800 与进行相互通信。该系统还可以包括视频显示单元 1808（例如，液晶显示器 (LCD) 或阴极射线管 (CRT)）。该系统还可以包括字母数字输入设备 1810（例如，键盘）、以及用于

接收内容源和传输内容存储的网络接口设备 1812。

[0247] 磁盘存储器 1806 可以包括机器可读介质，其中在该机器可读介质上，可以存储体现本申请所描述的方法或功能中的任何一个或多个的一个或多个指令集（例如，软件）。这些指令还可以完全地或者至少部分地位于存储器 1804 中、和 / 或在系统执行期间位于摄取处理器 1802 中，其中存储器 1804 和摄取处理器 1802 也构成机器可读介质。

[0248] 如图 19 中所示，可以根据各种实施例，将客户端系统实现成硬件和软件组件的组合。客户端系统可以包括能被执行以使系统执行本申请所讨论的方法中的任何一个或多个的指令集。可以将该系统实现成具有计算机形式的特定机器。该系统可以是服务器计算机、个人计算机 (PC)、或者能够执行指令集（顺序的或者其它方式）的任何系统，其中该指令集规定要由该系统采取的动作。此外，虽然仅示出了单个系统，但是术语“系统”还应当包括：单独或者联合地执行一个（或多个）指令集以执行本申请所描述的方法中的任何一个或多个的任何系统集合。

[0249] 客户端系统可以包括客户端处理器 1902（例如，中央处理单元 (CPU)）、存储器 1904 以及磁盘存储 1906，其中存储器 1904 可以在执行期间存储程序代码，所有这些部件通过总线 1900 进行相互通信。该系统还可以包括视频显示单元 1908（例如，液晶显示器 (LCD) 或者阴极射线管 (CRT)）。该系统还可以包括字母数字输入设备 1910（例如，键盘）、以及用于发送请求和接收响应的网络接口设备 1912。

[0250] 磁盘存储 1906 可以包括机器可读介质，其中在该机器可读介质上可以存储体现本申请所描述的方法或功能中的任何一个或多个的一个或多个指令集（例如，软件）。这些指令还可以完全地或者至少部分地位于存储器 1904 中、和 / 或在由系统对其执行期间位于客户端处理器 1902 中，其中存储器 1904 和客户端处理器 1902 也构成机器可读介质。

[0251] 特定实施例的示例

[0252] 使用在 [RaptorQ-RFC-6330] 中详细说明的 RaptorQ FEC 方案，在该节的针对通用对象传输的详细说明的 FEC 方案中，示出了一个特定的实施例。可以将 UOSI FEC 有效载荷 ID 与 [RaptorQ-RFC-6330] 中的 RaptorQ FEC 方案一起使用（本申请称为“UOD-RaptorQ FEC 方案”），以便提供简化的和增强的对象传输能力。具体而言，提供了用于基本对象传输的更灵活和更简单支持，此外还提供了针对不等错误保护 (UEP) 对象传输、针对绑定的对象传输和针对 UEP 和绑定的对象传输的组合的支持。应当理解的是，可以使用适当的硬件和 / 或软件来实现通信设备之间的“UOD-RaptorQ FEC 方案”。

[0253] FEC 有效载荷 ID 格式是如图 2 中所示的 4 字节字段，其中在该特定的实现中，32 比特、无符号整数的通用文件符号标识符 (UFSI)，通过也是 32 比特、无符号整数的通用对象符号标识符 (UOSI) 来概括。该 UOSI 是非负数整数，其结合 FEC OTI 来用于标识在携带该有效载荷 ID 的分组中包含的经编码的符号。

[0254] 为了传输单一对象、或者多个对象、或者被划分成具有不同优先级的部分的单一对象、或者这些的任意组合，FEC OTI 是如下所述的。应当注意的是，对于传输的每一个对象，FEC OTI 可以与 RaptorQ FEC 方案所指定的相同。一种传输可以包括 d 个对象，其中 d 是某个正整数。每一个对象可以包括同一文件的不同部分、或者不同的文件、或者其的组合。对象 i 的大小 F_i 和用于对象 i 的经编码的符号的大小 T_i 之间的关系，可以用于确定对象 i 在该传输中的优先级。

[0255] 图 20 示出了一种通用 FEC OTI 字段的示例。如本申请所使用的，对于传送的多个（d 个）对象，存在 8 比特、无符号的整数，所示出的示例对应于 d = 2。缺省值可以是 d = 1。其它子字段包括：对于这 d 个对象中的每一个（即，i = 1,...,d），特定于对象 i 的通用 FEC OTI 元素包括：用于符号大小 T_i 的 16 比特无符号整数，其是小于 2^{16} 的正整数（其以字节为单位来指示用于对象 i 的符号的大小）；用于对象 i 的传送长度 F_i 的 40 比特无符号整数，其是至多的非负整数，以字节为单位来指示对象 i 的传送长度。提供了适当的填充。

[0256] 与通用 FEC OTI 字段相比，可以存在特定于方案的 FEC OTI 元素，例如，如图 21 中所示。如图所示，在该示例中，特定于方案的 FEC OTI 元素包括：符号对齐参数 (Al)(8 比特，无符号整数）、用于每一个对象的特定于方案的 FEC OTI 元素（其包括用于对象 i 的源块的数量 Z_i(12 比特，无符号整数））、以及用于对象 i 的子块的数量 N_i。该编码的特定于方案的对象传输信息是 (1+3*d) 字节字段。图 21 中的示例对应于 d = 2。转而该编码的 FEC OTI 可以是 (2+10*d) 字节字段，其包括编码的通用 FEC OTI 和编码的特定于方案的 FEC OTI 元素的串联。

[0257] 使用编码的 FEC OTI 的内容传输可以涉及：使用 UOD-RaptorQ FEC 方案和利用 UOD-RaptorQ FEC 方案进行对象传输的内容传输协议（“CDP”），在系统的设备、计算机、程序之间进行信息的交换。CDP 应当向编码器设备和解码器设备提供 d、Al，以及对于每一个对象，F_i、T_i(Al 的倍数）、Z_i 和 N_i。向编码器提供通过 i 进行索引的各个对象。使用 UOD-RaptorQ 编码器方案的编码器设备将提供 CDP，针对要发送的每一个分组，该分组的 UOSI 和用于这 d 个对象的经编码的符号。

[0258] 参数选择的示例

[0259] 在一个示例中，可以使用在 [RaptorQ-RFC-6330] 的第 4.3 节中所描述的示例性参数推导，其独立地应用于 d 个对象中的每一个。举例而言，Al = 4 字节，SS = 8（其意味着每一个子符号至少是 SS*Al = 32 字节），对象 i 的 Ti 优选地是至少 SS*Al 字节，其中 T_i 是 Al 的倍数，而每一个编码分组的有效载荷大小优选地具有至少 T 的大小，其中 T 是 T_i 的关于 i = 1,...,d 之和。

[0260] 对于源块构造来说，在 [RaptorQ-RFC-6330] 的第 4.4.1 节点中的过程可以独立地应用于 d 个对象中的每一个。

[0261] 对于编码分组构造来说，每一个编码分组都包含 UOSI 和用于 d 个对象的经编码的符号。为了实现 [RaptorQ-RFC-6330] 的 RaptorQ FEC 方案所使用的 FEC 有效载荷 ID 的 (SBN,ESI) 格式，和 UOD-RaptorQ FEC 方案所使用的 UOSI 格式之间的兼容性，其可以是特定的格式。例如，对于每一个对象，从 UOSI 值 C 到针对对象 i 的相应 (SBN,ESI) 值 (A,B) 的映射，可以是 $B = floor(C/Z_i)$，$A = C - B*Z_i$。类似地，对于每一个对象，从对象 i 的 (SBN,ESI) 值 (A,B) 到相应的 UOSI 值 C 的映射，将是 $C = A+B*Z_i$。

[0262] 对于每一个对象 i = 1,...,d 而言，从 0 到 KT_i-1 的 UOSI 值以源块交织的顺序来标识源块中的对象 i 的源符号，其中 $KT_i = ceil(F_i/T_i)$。从 KT_i 向上的 UOSI 值标识使用 RaptorQ 编码器，根据对象 i 的源符号所生成的修复符号。

[0263] 编码分组可以包含源符号、修复符号或者源符号和修复符号的组合。一个分组可以包含来自于对象 i 的同一源块的任意数量的符号。当用于对象 i 的源分组中的最后源符号包括为了 FEC 编码目的所增加的填充字节时，这些字节应当包括在该分组中，使得在每

一个分组中只包括完整的符号。

[0264] 在每一个编码分组中携带的通用对象符号标识符 (C)，是在该分组中携带的针对每一个对象的第一经编码的符号的 UOSI。该分组中针对各个对象的后续经编码的符号，具有串行顺序的、编号为从 C+1 到 C+G-1 的 UOSI，其中是该分组中针对各个对象的经编码的符号的数量。在其它实施例中，可以针对每一个对象 i，存在被指定成 FEC OTI 信息的一部分的基 UOSI U_i，在该情况下，在该分组中用于对象 i 的第一符号的 UOSI 是 $C+U_i$。

[0265] 在优选的实现中，在每一个编码分组中，对应于 d 个对象中的每一个，存在一个经编码的符号。在优选的实现中，根据下面的过程来生成和发送编码分组。对于每一个 UOSI 值 C = 0,1,2,3,...，编码器如下所述地生成和发送编码分组，其中该编码分组的 FEC 有效载荷 ID 的值被设置为 UOSI 值 (C)，对于每一个对象 i（其中 i = 1,...,d)，编码器确定与 UOSI 值 C 相对应的 (SBN,ESI) 值 (A_i,B_i)，根据 RaptorQ FEC 方案 [RaptorQ-RFC-6330] 的过程，生成与对象 i 的 (SBN,ESI) 值 (A_i,B_i) 相对应的大小为 T_i 的经编码的符号 E_i，将经编码的符号 E_i 增加到编码分组的有效载荷中，并发送该编码分组。应当注意，不需要接收机知道编码分组的总数。

[0266] 单播源服务器和广播修复服务器

[0267] 在本节所描述的大部分实施例中，以广播方式来发送修复符号，响应于针对源符号的某些集合的 UE 请求，以单播方式来发送源符号。存在一些其它实施例，在该节所描述的一些以及在本申请的其它地方所描述的一些（其中至少一些源符号以广播方式进行发送），还存在响应于 UE 请求，以单播方式来发送至少一些修复符号的一些实施例。

[0268] 一种示例性广播文件传输系统是 eMBMS 文件传输系统。一种示例性单播文件传输系统是具备 HTTP 字节范围请求能力的系统。但是，可以使用只能请求并提供整个文件的单播系统，但将会丧失很多利益。本申请的更早部分示出了一些示例系统。

[0269] 在该节的示例中，在广播会话中发送修复数据，在单播修复会话中只发送源数据。单播修复请求是针对于原始文件的一部分的标准 HTTP1.1 字节范围请求。由于广播会话只具有修复符号，因此在两个会话之间不具有重叠的数据。应当注意的是，在一些实现中，先前描述的系统和该节的系统可以共存和 / 或进行组合。

[0270] 如上面所解释的，UE（例如，诸如移动用户设备之类的终端用户设备）可以被配置为：确定该 UE 丢失了哪些源符号，其转换自源块编号，将丢失的源符号的符号 ID 编码成标准 HTTP1.1 字节范围请求，并进行这些请求。

[0271] 通过在广播会话中发送修复符号，单播修复服务器只需要该文件的原始拷贝，这些请求是针对于连续范围的源符号的简单请求，或者是针对于连续的更大范围的连续字节的请求，而不是针对于各个符号的请求或者针对于多个较小范围的字节的请求。UE 可以基于简单地统计接收到多少符号和需要多少符号，或者基于运行解码调度来确定例如源符号的前缀数量（其允许源块的解码）、或者允许源子块的解码的源子符号的前缀数量、或者允许子块或者源块的解码的前缀字节范围大小，来确定要请求多少符号。应当注意，在该上下文和本申请的其它上下文中，“前缀”意味着指定“与一个子块相对应的文件的部分的前缀”或者“与一个源块相对应的文件的部分的前缀”，即，“前缀”可以不指代整个文件的前缀，而是指代该文件的有关部分的前缀，其中该有关部分可以根据使用“前缀”的上下文来进行推断。

[0272] 当不使用子块划分时，可以将针对一个源块的连续源符号的请求，转换成针对一个连续的字节序列的请求。当使用子块划分时，可以将针对一个子块的连续数量的源子符号的请求，转换成针对该文件的连续的字节序列的请求。

[0273] 如本申请所解释的，UE 可以根据针对一个子块的连续数量的源子符号的请求，来导出在该文件中的字节范围请求。

[0274] 在一些实现中，不同的广播服务器具有不同的修复符号集，其或许是基于不同的地理。这对于在广播传输期间从一个地理位置移动到另一个地理位置，以便确保不存在针对同一设备的重复接收的 UE 来说是有益的。

[0275] UE 在 HTTP 请求中请求一个子块（或者块，在不使用子块的情况下）的字节范围前缀，极大地独立于其在广播会话中接收的修复子符号（符号，在不使用子块的情况下；下面的括号也应用此含义），其仅取决于其在广播会话中接收到多少修复符号。对于一些编码（例如，在 IETF RFC5510 中详细说明的 Reed-Solomon 编码），所请求的字节范围前缀可以完全独立于在广播会话中接收的修复符号。对于其它编码（例如，在 IETF RFC6330 中详细说明的 RaptorQ 编码），字节范围前缀的大小只很弱地取决于在广播会话中接收到哪些修复符号。例如，如果在一个源子块中存在 K 个源子符号，并且接收到该子块的 K-L 个修复子符号，则请求与前 L 个源子符号相对应的字节范围前缀，将允许以至少 99％的概率进行解码，请求 L+1 个子符号，将允许以至少 99.99％的概率进行解码，请求 L+2 个子符号，将允许以至少 99.9999％的概率进行解码。是否进行解码可以是取决于用于解码的子符号的模式，接收设备在进行该请求之前，确定具体的接收子符号序列是否允许进行解码，因此确定要进行哪些请求以保证解码。因此，针对修复所请求的前缀的大小，可能只很弱地取决于在广播会话中的接收子符号的模式。

[0276] 但是，为了简单起见，接收设备可以针对 L+2 个子符号进行请求，在该情况下，请求大小独立于在广播会话中接收到哪些子符号，以接收潜在的 2 个冗余子符号的开销为代价，以及以很小概率的不能进行解码为代价，即，为了恢复 K 个源子符号，总共接收 K+2 个子符号，其中 K-L 个子符号来自于广播会话，L+2 个子符号来自于该修复请求。独立于接收到哪些数据，进行该大小和模式的请求的一些原因，仅取决于在广播会话中接收到多少数据，其包括：

[0277] (a)UE 请求可以是简单的，始终从源子块（块）的开始进行请求，每一个源子块（块）至多一个字节范围请求（在一些情况下，如果在广播会话中接收到足够的修复子符号（符号），那么不需要另外的 HTTP 请求）。

[0278] (b)UE 可以在通过 HTTP 从修复服务器下载的同时，开始播出该子块（块）的内容，转而一旦其下载了足够的内容，就可以使用通过广播已经接收的修复子符号来恢复该子块的剩余部分（剩余子块的大小本质上是在广播会话中接收的针对该子块（块）的修复子符号（符号）的大小）。

[0279] 甚至可能的情况是，接收机同时通过广播接收修复数据时，在同一时间尝试播放该内容，通过 HTTP 下载足够的各个子块，直到与通过广播已经接收的内容相组合后，足够用于对该子块的剩余部分进行解码和恢复。在该情况下，随着广播继续进行，随着 UE 通过广播接收到针对所有子块（块）的越来越多的修复子符号（符号），针对每一个子块（块）所需要下载的 HTTP 的量开始变得更小。

[0280] (c) 由于 UE 正在请求源子块的前缀，针对于文件的一部分的 UE 请求之间的重叠尽可能地高，例如，接收针对于一个源子块（块）的相同数量的修复子符号（符号）的两个 UE，针对该源子块（源块）准确地进行相同的字节范围请求，即使当接收到的修复符号的模式是独立的，并且可以是完全不同的。

[0281] 当不同的 UE 从针对一个子块的广播会话接收到不同数量的修复子符号时，接收到最小数量的修复子符号的 UE，通常针对该源子块进行最大的前缀字节范围请求，来自于其它 UE 的所有请求将是该请求的一个子集。因此，如果 HTTP 服务器已经取得和服务一个 UE 针对一个子块（块）的字节范围请求，那么该 HTTP 服务器可以对该响应进行高速缓存，使其可用于针对相同的子块（块）进行修复请求的下一个 UE。与 UE 所请求的相比，HTTP 高速缓存的服务器可以主动地从上游服务器请求更大的字节范围，因此减少在该请求的大小之外的数据变得可用的时间（如果存在来自相同 UE 或者其它 UE 针对超出当前请求之外的数据部分的另外请求）。因此，如果一个 UE 从文件的有关部分，进行针对数据的前缀的请求，那么当后续的 UE 从文件的相同有关部分，进行针对数据的更大前缀的请求时，HTTP 高速缓存服务器可能已经具有了该高速缓存，并能够立即使用数据的相应前缀对该请求进行响应。

[0282] 如果 HTTP 修复服务器接收到针对重叠的字节范围的多个请求（例如，超过某个门限），那么服务器还可以被配置为：决定将一个请求转发给请求该字节范围的数据的广播的 BMSC，以避免将这些单播信道冗余地装载到这些接收机。修复服务器只需要向每一个接收机单播将不会由 BMSC 进行广播的源数据。BMSC 可以将该字节范围请求转换到所需要的必需的源符号，随后广播这些源符号，或者更佳的是，广播这些更多的新修复符号。这使终端能够接收足够的非冗余符号，以便在很少装载到单播信道的情况下，对其文件进行修复。

[0283] 在一些情况下，有利的是，当 UE 准备播放与一个源块（子块）相关联的内容，UE 只针对该源块（子块）的源符号（子符号）进行单播请求，即，使用“流模式”进行请求。一种原因在于：UE 只请求其实际将要播放的内容的修复数据，如果由于某种原因，其不再播放该内容中的一些，那么其不再请求针对该块（子块）的数据，以避免浪费该单播网络资源。

[0284] 例如，UE 可以从中间位置处开始播放一个内容（文件），仅播放该内容的四分之一。在该示例中，仅当该播放临近时，才进行这些请求，将单播网络资源减少到：如果在播放之前为了恢复整个内容文件而进行单播修复请求时所需要的内容的大约四分之一。当然，一旦接收到一个源块（子块）的足够的源符号（子符号），那么不需要该 UE 针对该源块（子块），从单播修复服务器请求另外的数据。

[0285] 系统运营商可以从这些方法中受益。例如，可以使单播修复服务器的部署简化，并且使费用更加的合理。此外，还消除了需要购买、部署和操作专用的修复服务器，而不是更多的更便宜和更容易地依赖于部署和管理 HTTP web 服务器的需求。再举一个例子，运营商可以部署一种服务，通过该服务，透明地捕获互联网内容文件，将针对其 FEC 修复数据广播给 UE，随后 UE 包括一些方法以便使用向其所广播的所接收的 FEC 修复数据，确定结合通过广播所接收的 FEC 修复数据进行什么 HTTP1.1 字节范围请求，以便使其能够恢复将要进行播放的该文件的一部分，针对已经在现有的 CDN 中高速缓存的互联网可用的内容文件的一部分（其不需要由运营商进行操作，也不是由运营商所必须产生的内容），通过互联网进行适当的 HTTP1.1 字节范围请求。

[0286] 通过这样操作，运营商可以提供有价值的服务，提供更佳的用户体验（这是由于文件能更加可靠和更加快速地可用），同时减少传输这些文件所需要的在其网络上的整体网络业务量（这是由于广播数据对于所有 UE 都是有用的，与通过针对各个 UE 的很多个别单播连接来发送数据相比，在发送数据时占用更少的网络资源），同时运营商可以避免部署任何单播修复服务器来支持其服务的费用，这是由于该内容已经可以从其它方所提供的标准 HTTP web 服务器处可用，并且其用于支持单播修复。

[0287] 系统运营商还可以重新设定现有的 HTTP web 服务器的用途，生成新的服务机会，例如截获和广播用于通过 HTTP web 服务器从其它内容提供商向 UE 传输的文件的修复符号。

[0288] 另外，该方法可以保存运营商网络带宽容量。该成本是在网络上发送的广播数据的量，而利益则是所有 UE 的效益之和，其中这些 UE 都播放在本 UE 上接收的广播数据的量的内容。这可以在运营商网络上提供更佳的用户体验，以及内容的更快速和更可靠传输。

[0289] 会话描述和 MBMS 下载传输协议 (FLUTE) 向客户端（即，UE) 提供足够的信息，来确定每一个文件的源块和经编码的符号结构。据此，客户端能够确定请求和使用哪些源符号来恢复该文件。因此，MBMS 客户端能够识别完成（文件的）源块的重建而所需要的源符号的数量。

[0290] 当使用 MBMS FEC 方案时，如 3GPP TS26.346 中所详细说明的，MBMS 客户端可以在确定所需要的另外的源符号时，考虑已经接收的修复符号。在该情况下，客户端应当识别（该文件的）每一个源块数量 (r)，针对允许 MBMS FEC 解码器恢复该文件的该源块的前 r 个源符号，进行修复请求。

[0291] 一旦识别出丢失的文件数据，MBMS 客户端就向文件修复服务器发送一个或多个消息，这些消息请求传输能恢复丢失的文件数据的数据。针对特定的 MBMS 传输的所有文件修复请求和修复响应，可以使用 HTTP 协议在单一 TCP 会话中发生。替代地，可以将字节范围请求划分成不同的潜在重叠的 HTTP/TCP 连接、或者 FTP 连接、或者 RTP 连接等上的多个请求。可以根据网络可用性，以及接收设备的繁忙程度以及其能力，按照非常慢或者高的比特速率，来传输针对修复请求的响应。可以将修复请求路由到根据所选定的“serviceURI”来解析得到的文件修复服务器 IP 地址。打开与服务器的 TCP 连接的时间，以及特定的 MBMS 客户端的第一次修复请求，可以随机化在某个时间窗上，使得不是所有 UE 都同时地进行其单播请求。

[0292] 当 MBMS UE 识别在修复请求中要请求的符号时，答复可以是所识别的相应符号，其应当包括在该请求中所标识的有关源块的所有符号。当使用子块划分时，MBMS UE 可以替代地请求源子块的子符号。替代地，如本申请所描述的，可以进行字节范围请求，以请求符号或者子符号。

[0293] MBMS UE 可以将该方法与其它方法进行组合。例如，MBMS UE 可以在广播会话中接收数据，在不对原始源文件进行解交织或解码的情况下，将其存储在长期存贮器中，如在 Luby VI 中所描述的。响应于终端用户请求播放该源文件的一部分（例如，当源文件是视频文件时），MBMS UE 可以触发原始源文件的这些部分的解交织和解码。如果 MBMS UE 没有从广播会话中接收到足够的数据，来恢复该原始源文件的被请求进行播放的部分，那么 MBMS UE 可以针对终端用户请求进行播放的该源文件的有关部分，进行如本申请所描述的修复请求。因此，仅当在广播会话中没有接收到足够的数据来恢复该部分时，MBMS UE 才可以进行请求，

以接收针对某些子块或者源块的额外数据，按照或者接近终端用户期望播放源文件的一部分（其与请求的子块或源块相交叉）的时间，才进行这些请求。该实施例将网络资源减到最小，这是由于仅当终端用户请求播放这些部分时，才请求该源文件的这些部分的修复数据，因此如果文件的一部分永远不被播放，那么就从不进行针对这些部分的修复请求。此外，该实施例将从长期存贮器中读取数据，以及对该数据进行解交织和解码以便恢复子块或者源块，而所需要的设备资源减到最小，这是由于仅当终端用户请求播放源文件的这些部分时，才执行这些操作。

[0294] 替代地，当 MBMS UE 具有适当的网络连接和空闲容量来执行修复请求时，MBMS UE 可以针对没有从广播会话中接收到足够的数据来实现恢复的子块或者源块，执行修复请求，随后 MBMS UE 除了存储从广播会话中接收的针对这些子块或源块的数据之外，还将该响应数据存储到长期存贮器以便用于这些子块或源块的稍后恢复。当用户请求播放原始文件的一部分时，MBMS UE 可以从长期存贮器中读回从广播会话接收的被交织和经编码的数据与来自于针对这些子块或源块的数据（其与用户请求的该文件的部分相交叉）的组合，恢复这些子块或者源块，将其直接提供给视频播放器以用于播出。这种替代方式具有下面的优点：当 MBMS UE 连接到适当的网络来进行修复请求和接收响应时，可以请求修复数据，而当用户请求播放时，MBMS UE 可能没有连接到适当的网络。此外，针对于用户对于播放的请求的播放响应时间更小，这是由于在该设备上进行用户请求的时间，用于播放所需要的所有数据都是本地存储的，而不是需要通过网络进行传输。

[0295] 再举一个例子，MBMS UE 可以针对需要另外的数据以便进行恢复的所有子块或者源块，来请求修复数据，并且恢复该文件的全部，将其存储在长期存贮器中以便稍后播放或者使用。MBMS UE 可以按照任何顺序，在任何时间点执行这些操作，即，在一些情况下，可以在从广播会话接收到数据之前，对修复数据进行请求，例如，如果已知没有通过广播会话发送足够的数据来完整地恢复该文件的这些子块或者源块，或者如果终端用户期望在广播会话结束之前（并潜在地在该广播会话开始之前），就开始播放内容。

[0296] 在一些实施例中，存在着不执行 FEC 的 UE（因此宁愿所有的源符号，而不需要修复符号）。在这些情况下，UE 将使用“无编码 FEC”，在广播会话和任何修复请求响应中接收源符号，但由于多个不同的 UE 经历不同的丢失模式，所以进行不同的源符号修复请求，因此这种方式不是可扩展的。

[0297] 在一些实施例中，修复广播会话在 UE 处使用如 Luby VI 中所示出的广播数据的部分重新排序。这里，可以将使用如上所述的 HTTP 的单播修复服务的组合，与 Luby VI 中描述的方法进行组合。可以将在广播传输中接收的符号 / 子符号，存储到如 Luby VI 中所描述的闪存或其它存储器中。在一些情况下，优选的是，仅在广播会话中才发送修复符号，但在其它情况下，优选的是，在广播会话中发送源符号中的至少一些（如果不是全部的话）。当通过具有字节范围请求或者具有其它 MBMS 方法的 HTTP，在修复服务中请求符号或者子符号时，出现两种不同的可能性（这两者可以同时地使用）：

[0298] (1) 可以使用在 Luby VI 中描述的方法，将从广播会话中接收的符号或者子符号写入到闪存中，随后当要恢复源块或者子块时，可以使用在 LubyVI 中描述的方法，将来自于广播会话的在闪存中存储的针对该源块或子块的符号或者子符号读入到 RAM 中。随后，可以将其与通过 HTTP 接收并直接存储到 RAM 中的符号或子符号进行组合，使用这些符号或子

符号的组合，在 RAM 中对该源块或子块进行解码，转而可以直接提供所恢复的源块或子块以用于播放。

[0299] (2) 可以使用在 Luby VI 中描述的一些方法，将通过 HTTP 接收的符号或者子符号写入到闪存中，对于通过广播会话接收的符号或者子符号执行相同的操作，随后在任何时间点，可以通过将基于 HTTP 请求而存储在闪存中的源块或子块的符号或子符号，与来自于广播会话的在闪存中存储的针对该源块或子块的符号或者子符号的组合，从闪存读入到 RAM 中，来及时地恢复源块或者子块。

[0300] 在上面的实施例中，优选的是，通过广播会话接收的符号或者子符号，以及在修复会话中接收的符号或者子符号，是极其不同的符号或者子符号集。

[0301] 图 22 示出了基本 FLUTE 文件传输。应当注意，FEC 有效载荷 ID 可以替代地包括本申请所描述的 UOSI 或 UOFI。

[0302] 图 23 示出了基本 FLUTE 分组格式的一些部分。

[0303] 图 24 示出了在使用子块的发送方处发生的子块编码。如上所述，可以将一个文件组织成一个或多个源块（在该示例中，组织成一个源块），并划分成一些子块，其中对这些子块进行布置以便根据子符号来形成符号。

[0304] 图 25 示出了在使用子块的接收机处发生的子块解码。在一些情况下，在接收机处进行存储之前，存在部分的重新排序。这可能是用的，例如，当在读取和写入之间的存储单元中，存在不对称时。

[0305] 图 26 示出了使用子块的文件传输。在典型的情况下，接收机在每一个子块中，观察到相同的丢失模式。从传输的角度来看，一个文件可以是一个源块，每一个分组可以包含一个符号。从解码的角度来看，对于所有子块来说，解码操作的调度可以是相同的，每一个子块的子符号具有相同的丢失或者接收模式，接收机一次针对所有的子块来计算该调度。随后，将解码操作的调度单独地应用于每一个子块，对子符号进行操作，并向所有子块应用相同的调度。即使对很大的源块进行操作，解码器的 CPU 也可以很好地执行，其只使用很少量的划伤 RAM 存储器，允许媒体的子块的访问和解码（当其被播放时）。

[0306] 图 27 示出了多个源块的处理的示例。在一些情况下，为了实现良好的网络效率，以循环方式来发送这些符号，其中每一轮发送来自每一个源块的一个符号。在其它情况下，为了更容易地客户端恢复或者端到端时延的原因，以顺序方式来发送这些符号，即，对于每一个源块来说，以连续的顺序来发送所有符号，而不插入来自其它源块的符号。在所有情况下，都可以使用子块划分。

[0307] 图 28 示出了使用 FEC 和 FLUTE 的工作流。在步骤 1，对文件进行规定，以便在 FLUTE 会话中进行携带。在步骤 2，可以将该文件分割成一些块。根据 FEC 方案可以处理的最大尺寸，来导出用于分段的需求。

[0308] 在步骤 3，将 FEC 应用于一个块，每一个块被划分成 K 个相同大小的系统符号，生成 R_1+R_2 个修复 FEC 符号。在步骤 4，$K+R_1$ 个符号在广播传输上，通过 FLUTE 来发送，或者替代地，使用 UDP 通过单播来发送，或者使用 HTTP 通过单播来发送。根据符号大小，每一个 IP/UDP/LCT 分组携带一个或多个符号。在步骤 5，接收机收集足够的分组，使得有 $K+\delta$ 个符号可用于 FEC 解码。在一些实施例中，使用 RaptorQ，$\delta = 2$。但是，其它值也是可以的，例如，$\delta = 0$ 或者 $\delta = 0.01*K$。在步骤 3 中所生成的另外的 R_2 个符号，可以提供给 HTTP 服

务器，以便为没有从步骤 4 中的传输接收到 K+δ 个符号的接收机，提供基于 HTTP 的修复服务。

[0309] 图 29 示出了上面所描述的广播 / 修复、单播 / 源配置。可以生成很大数量的修复符号。为了成功地进行 FEC 解码，UE 需要这些符号中的任何 K+δ 个。在所示出的实施例中，每一个广播 BM-SC 发送一个文件的 N 个不同的 Raptor 修复符号，每一个 BM-SC 只发送修复符号，即，ESI 在范围 {K,K+1,…} 之中的那些符号。

[0310] 上面的实施例的一种优点在于：当 UE 在不同的 BM-SC 的覆盖区域之间漫游时，UE 可以使用来自不同 BS-SC 的符号，不用担心符号的重复接收，这允许更高效的 FEC 解码。UE 可以针对源符号，进行简单的修复请求（但这可能需要专用的修复请求服务器），但还可以使用传统的 HTTP web 服务器（其中传统的 HTTP web 服务器只具有如在本申请的其它地方所更详细描述的原始源文件），进行 HTTP 字节范围请求。这允许更加成本有效融合的互联网 /MBMS 解决方案。

[0311] 图 30 示出了系统如何在 MBMS 承载上只广播修复符号，以及 UE 只请求一组连续的源符号。通常，UE 需要 K+δ 个符号来成功地对源块进行解码，但其通过 MBMS 承载只接收到 K+δ-R 个修复符号。为了恢复该源块，UE 请求该源块的前 R（或者 K，当 R>K 时）个源符号。它可以请求某些其它的 R 个源符号，但如果所有 UE 请求重叠的源符号集合，那么下游高速缓存更可能能够将所请求的源符号从其本地的高速缓存传输到更多的接收机。

[0312] 这提供了在修复服务器处，实现简单和高效的高速缓存。由于通过广播信道只发送修复符号，源块的源符号与 UE 已经接收的符号完全地不同。因此，修复服务器只需要存储源块的源符号，不需要存储或者生成新的修复符号。这允许更简单的修复服务器，这种更简单的修复服务器不需要实现 FEC 方案或者存储新的修复符号。

[0313] 可以对接收的符号进行简单的 UE 处理。UE 只需要跟踪其接收到的修复符号的数量，转而计算其需要的源符号的数量 (R，当 R 小于或等于 K 时，或者 K，当 R 大于 K 时）。UE 不需要跟踪哪些具体的源符号丢失了，并随后针对这些丢失的源符号来生成请求。

[0314] 另外，UE 可以使用初始的 ESI+(R 的值），从修复服务器请求一组连续的源符号。这使得这些请求保持非常简单和简短。此外，这还可以简化服务器响应构造，这是由于服务器不需要获取和添加非连续的源符号集合。由于服务器只需要处理连续的源符号请求，因此这在不使用“无编码”FEC 方案的部署中非常有利。

[0315] 图 31 示出了当使用子块划分时，通过 HTTP 字节范围请求，使用单播修复。

[0316] 使用 HTTP 字节范围修复请求，UE 可以针对同一源块的每一个子块，请求相同数量的字节。当在广播会话中没有发送源符号时，可以在大部分环境中，使用每一个子块对应于一个字节范围的请求。

[0317] 可以在“按需”的基础上，进行 HTTP 请求。当到达播放各个子块中的内容的时间时，UE 可以进行针对该子块的请求。UE 可以在 HTTP 上请求和播放针对子块的内容，直到结合接收机通过其它传输途径（例如，广播会话）接收的子符号，具有足够的子符号来解码和恢复该子块为上。一旦对足够的子符号进行了接收和解码，UE 就可以从本地恢复的子块开始，无缝地继续进行播放。UE 不需要针对从不被播放的子块，进行 HTTP 请求。

[0318] 字节范围计算的示例

[0319] 接收机可以执行在本节中所解释的计算，以便确定要请求什么字节范围。这种计

算可以依赖于FEC结构。随后，接收机可以向服务器发送请求，以便获得该接收机基于字节范围请求所需要的符号。

[0320] 在优选的实施例中，接收机使用FEC对象传输信息（“FEC OTI”）来认识其没有接收到哪些符号。FEC OTI确定文件的源块、子块、符号和子符号结构，其包括对象传输大小F、对齐因子Al、符号大小T、源块的数量Z、以及每一个源块的子块的数量N。接收机使用所有这些信息以及FEC编码自身的属性，来准确地确定其需要多少符号和需要什么符号，其中每一个符号或者子符号通过SBN、ESI和SuBN（分别为源块数量、经编码的符号标识符、子块数量）来标识。在该实施例中，对源符号或者源子符号进行请求。接收机使用FEC OTI信息来计算该文件中与这些符号或子符号相对应的字节范围，如下面所描述的。

[0321] 当不使用子块划分时（即，当N＝1时），对于给定的SBN和ESI，接收机可以计算(SBN,ESI)对所标识的相应源符号在该文件中的起始字节编号和结束字节编号。

[0322] 现在描述计算过程。这可以通过接收机中的一个程序或者接收机中的逻辑单元来执行。使KT＝ceil(F/T)是文件中的源符号的数量（向上取整到最近的整数，其中最后的源符号可以被认为是用零填充成的一个完整源符号）。

[0323] 使KL＝ceil(KT/Z)并且KS＝floor(KT/Z)，ZL＝KT-KS*Z并且ZS＝Z－ZL，其中KL是前ZL个源块中的源符号的数量，KS是剩余的ZS个源块中的源符号的数量。

[0324] 对于每一个i＝0,…,Z-1，使K_i是源块i中的源符号的数量，即，K_i＝KL，其中i＝0,…,ZL-1，K_i＝KS，其中i＝ZL,…,Z-1。

[0325] 随后，与SBN和ESI所标识的符号相关联的起始字节，可以如式2中所示地进行计算。可以通过将T-1增加到起始字节位置，来计算该子符号的结束字节位置。

[0326] $$\sum_{i<\mathrm{SBN}} K_i \cdot T + \mathrm{ESI} \cdot T \quad （式2）$$

[0327] 当使用子块划分时（即，当N>1时），对于给定的SBN、SuBN和ESI，接收机可以计算该三元组所标识的相应源子符号在该文件中的起始字节编号和结束字节编号。针对KT、KL、KS、ZL、ZS和K_i的计算，与不使用子块划分(N＝1)时的情况的计算相同，其中i＝0,…,Z-1。

[0328] 用于识别子块字节位置的其它计算，可以如下所述地执行。

[0329] 使TL＝ceil(T/(N*Al))并且TS＝floor(T/(N*Al))，NL＝T/Al-TS*N并且NS＝N-NL，其中TL*Al是前NL个子块中的子符号的大小，TS*Al是剩余的NS个子块中的子块的大小。

[0330] 对于每一个i＝0,…,Z-1，使T_j是子块j中的子符号的大小，即，T_j＝TL*Al，其中j＝0,…,NL-1，T_j＝TS*Al，其中j＝NL,…,N-1。

[0331] 随后，与通过SBN、ESI和SuBN所标识的符号相关联的起始字节，可以如式3中所示地进行计算。该子符号的结束字节位置可以通过将T_{SuBN}-1增加到起始字节位置来计算。

[0332] $$\sum_{i<\mathrm{SBN}} K_i \cdot T + \sum_{j<\mathrm{SuBN}} T_j \cdot K_{SBN} + \mathrm{ESI} \cdot T_{SuBN} \quad （式3）$$

[0333] 基于针对请求的符号或者子符号的起始和结束字节位置的这些计算，接收机可以确定其需要向服务器发送的字节范围请求。在存在连续的丢失符号或者子符号的情况下，接收机可以从第一丢失的符号或者子符号的起始字节位置，开始该字节范围请求，在最后

丢失的符号或者子符号的结束字节位置处，结束该范围。

[0334] 为了示出在使用子块划分时，如何进行上面的计算，假定接收机具有一个广播下载会话，尝试获得将使用对象大小 F = 20,000,000 字节、对齐因子 Al = 4 字节、符号大小 T = 1320 字节、源块的数量 Z = 2、以及每一个源块的子块的数量为 N = 12 的 FEC OTI 来传输的 URI 为 www.<mobile-operator>.com/news/weather.3gp 的 3gp 文件。在该示例中，该文件中的源符号的总数是 KT = 15,152，SBN = 0 的第一源块和 SBN = 1 的第二源块均具有 7,576 个源符号。在该情况下，进一步将各个源块划分成 12 个子块，每一个具有 7,576 个源子符号，其中前 6 个子块具有大小为 112 个字节的子符号，剩余的 6 个子块具有大小为 108 个字节的子符号。在该广播下载会话之后，假定接收机认识到其没有接收到来自于 SBN = 0 的第一源块的 ESI 为 12-29 的 18 个连续的源符号，以及来自于 SBN = 1 的第二源块的 ESI 为 12-29 的 18 个连续的源符号，为了对这些源块进行解码，其至少需要来自这些源块中的每一个源块的更多符号。

[0335] 对于 SBN = 0 的第一源块来说，构成 ESI = 12 的源符号的 12 个子符号的起始字节位置 (18 个丢失的源符号中的第一个)，可以如图 32 中所示地进行计算。此外，图 32 还示出了如何对构成 ESI = 29 的源符号的 12 个子符号的结束字节位置 (18 个丢失的源符号中的最后一个) 进行计算。

[0336] 类似地，图 33 示出了：如何针对于 SBN = 1 的第二源块，对构成 ESI = 12 的符号的 12 个子符号的起始字节位置进行计算。此外，图 33 还示出了如何对构成 ESI = 29 的源符号的 12 个子符号的结束字节位置进行计算。

[0337] 如果所选定的修复服务器 URI 是 http://mbmsrepair2.<mobile-operator>.com，则 HTTP GET 请求可以是如下所述：

[0338] GET www.<mobile-operator>.com/news/weather.3gp HTTP/1.1

[0339] Range:

[0340] bytes = 1344-3359,849856-851871,1698368-1700383,2546880-2548895,3395392-3397407,4243904-4245919,5092368-5094311,5910576-5912519,6728784-6730727,7546992-7548935,8365200-8367143,9183408-9185351,10001664-10003679,10850176-10852191,11698688-11700703,12547200-12549215,13395712-13397727,14244224-14246239,15092688-15094631,15910896-15912839,16729104-16731047,17547312-17549255,18365520-18367463,19183728-19185671

[0341] Host:mbmsrepair2.<mobile-operator>.com

[0342] 头部：统计信息收集

[0343] 在一些实施例中，对统计信息进行收集，这是由于其与针对内容所进行的各种请求有关。这些统计可以是有用的，例如，用于系统运营商能够告知有多少文件不是完全地源自于内容的广播。这可以通过仅查看修复服务器的 HTTP 日志来执行，其中修复服务器以单播方式来提供修复数据。但是，如果修复服务器也服务成内容的直接源 (例如，根据来自于不尝试接收广播符号，或者简单地将修复服务器视作为该内容的主 HTTP 源的设备的请求)，则统计信息将在数目上超过这些请求。系统运营商可以具有其自身报告广播 / 单播统计 (并且不报告仅请求的单播) 的设备，但这依赖于这些设备，并且是令人麻烦的。在一种解决方案中，HTTP 修复服务器收集这些统计信息，并能够对由于广播丢失而针对修复数据

的请求，和其是针对内容的直接单播请求进行区分。

[0344] 在一种特定的方法中，修复服务器确定请求的类型（修复请求对比直接请求），以及哪个终端在进行该请求。第一部分用于区分下面两种请求：其正在接收的用于修复通过广播信道所接收的文件的 HTTP 请求；来自于仅请求或者修复其通过单播所接收的文件的终端的请求。终端标识使服务器能够对来自同一终端的请求进行映射，还提供针对每一个终端的详细接收统计。这可以用于减少双重统计，还用于更佳地映射修复模式。

[0345] 为了实现服务器确定所述类型和终端的这种能力，存在着一些这样做的方式。

[0346] 服务器 URL 或者文件的 URL，可以是用于区分直接单播类型请求的特定于修复类型请求。例如，这些 URL 可以仅在广播服务的关联传输过程中提供，也可以在被广播的文件的文件传输表中提供。不向终端提供这些 URL，来直接地下载没有被广播的文件。服务器可以确定针对特定 URL 的请求是用于修复广播的文件，并基于这些请求来收集修复统计信息。为了识别来自于同一终端的请求，服务器可以使用 HTTP GET 请求的源 IP 地址。

[0347] 服务器可以使用终端的 IP 地址，以识别哪个终端正在进行该请求。此外，如果 HTTP 服务器知道 IP 地址属于广播终端，则服务器还可以使用该 IP 地址来确定该请求是否来自于广播终端。应当注意，如果允许终端请求其没有通过广播接收的文件的数据，那么对于收集广播统计信息来说，该方法是不可靠的。所以可以对于直接从这些服务器请求文件的广播终端，设置一些限制条件。

[0348] 在另一种方法中，可以对终端所使用的 HTTP GET 请求进行扩展，以便从 HTTP 服务器请求数据。该扩展将识别该请求是用于修复广播的文件。此外，该扩展还可以识别终端的唯一 ID，例如，移动目录号码或者国际移动用户标识（IMSI）。扩展头部的另一个示例是使用如 3GPP TS24.109 中所规定的“X-3GPP-Intended-Identity extension-header”。如果为了私密性或者安全目的，运营商不希望直接地发送这些 ID，则可以向这些 ID 应用哈希或者加密。

[0349] 所有上面的方法都需要修复服务器的定制，并因此很难使用标准的 HTTP 服务器。支持上面的过程的服务器可以是可选的特征。用此方式，愿意依赖于来自 UE（即，终端）的接收报告的系统运营商，仍然使用标准的 HTTP 服务器。

[0350] 源符号和修复符号的广义分配

[0351] 在上面所描述的示例中，内容传输系统可以包括诸如 UE 之类的客户端和诸如 MBMS 广播服务器和 / 或 HTTP 服务器之类的服务器，优选地，其除了支持 URL 文件请求之外，还支持字节范围请求。在该方法中，可以仅对修复符号进行广播，仅源符号可通过针对于 HTTP 服务器的请求而获得。但是，在更广义的情况下，可以对修复和 / 或源符号进行广播，修复和 / 或源符号可从 HTTP 服务器获得。

[0352] 在一个示例中，将可从 HTTP 服务器获得的本质表达和 / 或发送成一个范围的 UOSI 值（替代地，其称为 UOFI 值）。当 UOSI 值的范围和与原始文件相对应的 UOSI 值相同时（即，UOSI 值 0 到 ceil(F/T)-1，其中 F 是字节表示的文件大小，T 是字节表示的符号大小），则其是可从 HTTP 修复服务器获得的原始格式（即，原始顺序 HTTP 文件格式）的原始文件。当 UOSI 值的范围在修复符号或者修复和源符号的 UOSI 上变化时，那么这些符号也是可获得的（其具有原始顺序 HTTP 文件格式的自然扩展的格式，下文对此进行了进一步描述，下文称为“扩展的原始顺序 HTTP 文件格式”）。这允许一种类型的请求来处理各种类型的符

号，并进行无缝地执行。例如，HTTP 服务器可以只通过发送 UOSI 范围或者某种其它信令，等同于只具有修复符号的修复文件或者具有修复符号和源符号的修复文件（混合格式），来对待只具有源符号的修复文件。应当注意，即使当使用子块划分时，当块大小在文件的不同源块之间发生变化时，该方案也能进行工作。

[0353] 在符号的 UOSI 排序中，只包含源符号的修复文件将具有 0 到 KT-1 的 UOSI 范围，其中 KT 是原始文件中的源符号的总数。这是可以发送的范围。例如，如果一个文件是 20,000,000 字节，符号大小是 1,000 字节，转而 KT = 20,000，因此发送 [0,19,999] 的 UOSI 范围，指示该修复文件是原始源文件（其独立于原始文件中的源块的数量），即，在该情况下，扩展的原始顺序 HTTP 文件与原始顺序 HTTP 文件相同。

[0354] 假定内容传输系统想要发送该修复文件只携带修复符号的信号。那么可以使用 [20,000,X] 的 UOSI 范围，其中 x 大于或等于 20,000。另外假定该源文件被划分成 Z = 19 个源块，如在 FEC 对象传输信息（“OTI”）中所发送的，因此前 12 个源块具有 1053 个源符号，剩余的 7 个源块具有 1052 个源符号。转而，为了发送该修复文件只携带修复符号，与每一个源块中的源符号的数量相比，针对该源块的修复符号的数量至少更多 3 个，那么可以将 UOSI 范围发送成 [20,000,40,063]，即，在该修复文件中，存在针对 19 个源块中的每一个源块的 1056 个修复符号。

[0355] 此外，如果使用子块划分，那么每一个子块的子符号具有在每一个源块中的顺序，即，在针对一个源块的修复数据中，首先是针对第一子块的 1056 个子符号，接着是针对第二子块的 1056 个子符号等，即其不需要是下面的情形：修复符号在针对该源块的修复数据中是连续的，并因此与原始源文件中的源数据具有相同的组织（即，扩展的原始顺序 HTTP 文件格式是原始顺序 HTTP 文件格式的自然扩展）。例如，假定 19 个源块中的每一个被划分成 N = 3 个子块，用于这些子块的子符号大小分别是 336、332 和 332 个字节。使 (J,L) 是用于该文件中的子块的标识符，其中 J 是源块编号，L 是该源块中的子块编号。那么，该文件将包括：针对子块 (0,0) 的 1056 个子符号，每一个子符号的大小为 336 字节，接着是针对子块 (0,1) 的 1056 个子符号，每一个子符号的大小为 332 字节，接着是针对子块 (0,2) 的 1056 个子符号，每一个子符号的大小为 332 字节，接着是针对子块 (1,0) 的 1056 个子符号，每一个子符号的大小为 336 字节，接着是针对子块 (1,1) 的 1056 个子符号，每一个子符号的大小为 332 字节，接着是针对子块 (1,2) 的 1056 个子符号，每一个子符号的大小为 332 字节等。在该示例中，源块 J = 0、…、11 中的子块以 ESI1053 开始，以 ESI2108 结束，源块 J = 12、…、18 中的子块以 ESI1052 开始，以 ESI2107 结束。

[0356] 再举一个例子，假定该服务意味着：当丢失了为了恢复该文件所需要的至多 20％的数据时，只支持接收机返回到从 HTTP 修复服务器寻求修复数据。那么，包含修复数据的修复文件可以更小，例如，对应于 [20,000,24,027] 的 UOSI 范围，即，针对于该修复文件中的 19 个源块里的每一个，存在 212 个可用的修复符号。在该示例中，针对位于前 11 个源块中的子块的子符号的 ESI 范围是 1053 到 1264，针对位于剩余的 8 个源块中的子块的子符号的 ESI 范围是 1052 到 1263。

[0357] 通常，原始源文件格式是顺序的，例如，数据处于下面的顺序：为了从开始到结尾地播放内容，而要消费的该文件的字节的顺序，即其是原始顺序 HTTP 文件格式。由于多种原因，这是一种便利的格式，其包括用于在 HTTP web 服务器上持有以便进行传输、以便存储

在文件系统中等的自然格式。这可以是用于源文件的格式，无论其是通过 eMBMS 来传输，还是通过单播来传输。如果通过 eMBMS 信道只发送修复符号，那么不发送顺序格式的源文件的任何部分，所以来自 HTTP web 服务器的请求是不重复的，可以进行针对源块（或者子块）的连续初始部分的请求，以便与通过 eMBMS 接收的内容进行组合，从而以有用的方式来恢复该文件。

[0358] 在期望通过 eMBMS 信道向接收机发送源符号和修复符号的情况下，使仅原始源文件在 HTTP web 服务器上可用也不能进行工作，这是由于已经通过 eMBMS 接收到源文件的一些部分，因此用于请求源文件的一些部分的针对 HTTP web 服务器的请求模式，可以结束对于源文件的很多较小部分的请求（跳过通过 eMBMS 接收的部分），从而就 HTTP 字节范围请求的容量而言，潜在地造成相当大的低效率，访问 HTTP web 服务器中的非连续数据部分，减少来自于不同的接收机（其中这些接收机经历 eMBMS 信道的不同的丢包模式）的请求不同的数据的高速缓存效率。

[0359] 如本申请所解释的，这可以通过使用扩展的原始顺序 HTTP 文件格式来解决，其中扩展的原始顺序 HTTP 文件格式将原始源文件格式扩展为包括修复符号，或者以自然的方式，在一些情况下只包括修复符号。使用这种新的文件格式，该文件可在传统的 HTTP web 服务器上获得，并用于提供融合的服务，其中在该服务中，可以使用字节范围请求，结合通过 eMBMS 来传输源符号和修复符号，进行针对传统的 HTTP web 服务器的修复请求，使得高速缓存效率较高，HTTP 字节范围请求减到最小，并是独立于不同的接收机在 eMBMS 会话中的丢失模式经历，针对于连续部分的数据（其中该数据以文件中的相同点起始）的请求。

[0360] 技术上，文件 X 可以具有相关联的 FEC OTI 参数 (F,Al,T,Z,N)。在扩展的原始顺序 HTTP 文件格式中，文件中的数据的范围可以依据通用对象符号标识符（“UOSI”）来表示，其替代地称为 UOSI。一个符号的 UOSI 与该符号的源块编号（“SBN”）和其经编码的符号标识符（“ESI”）相对应，即，对于 SBN 为 A 和 ESI 为 B 的符号，UOSI 可以是 C = A+B*Z。逆映射是：对于具有 C 的 UOSI 值的符号，ESI 值是 B = floor(C/Z)，SBN 是 A = C-B*Z。在发送 UOSI 范围时，可以用 [SU,EU] 形式来表达该范围，其中 SU 是该文件的起始 UOSI 值，EU 是该文件的结束 UOSI 值。应当注意，SU 小于或等于 EU。转而，在文件中包括的符号的范围是 UOSI 范围从 SU 到 EU 的那些符号，因此文件中的符号的总数是 EU-SU+1。这种新文件格式的一个特殊情形是 SU = 0，EU = ceil(F/T)-1，在该情况下，该文件准确地与原始文件相同，即，在该情况下，扩展的原始顺序 HTTP 文件格式与原始顺序 HTTP 文件格式相冲突（如果文件大小 F 不是符号大小 T 的倍数，则取模可能使用零填充来填充最后的符号）。

[0361] 该文件可以通过包括FEC OTI参数的URL来标识，通过该文件中的符号的起始和结束 UOSI 来标识。在这种新的扩展的原始顺序 HTTP 文件格式中，文件中的数据是：与 SBN = 0 的源块的 ESI 相关联的所有符号、接着是与 SBN = 1 的源块的 ESI 相关联的所有符号等，直到与 SBN = Z-1 的源块的 ESI 相关联的所有符号。在源块中，组织具有子块的顺序，即，与该源块的第一子块相关联的所有子符号、接着是与该源块的第二子块相关联的所有子符号等、直到与该源块的最后 (N-1) 子块相关联的所有子符号。

[0362] 位于 UOSI 范围 SU 到 EU 的 SBN = J 的源块中的源符号的 ESI，可以如下所述地进行计算。使 SESI(J) 是 UOSI 范围 (SU,…,EU) 中的符号之间，用于源块 J 的起始 ESI，使 EESI(J) 是 UOSI 范围 (SU,…,EU) 中的符号之间，用于源块 J 的结束 ESI。转而，可以如下

所述地计算 SESI(J)：

[0363] SESI(J) = floor(SU/Z)

[0364] If((SU-Z*floor(SU/Z))>J)then SESI(J) = SESI(J)+1.

[0365] 类似地，可以如下所述地计算 EESI(J)：

[0366] EESI(J) = floor(EU/Z)

[0367] If((EU-Z*floor(EU/Z))<J)then EESI(J) = EESI(J)-1.

[0368] 转而，来自于 UOSI 范围 (SU,…,EU) 中的源块 J 的符号的总数是 NK(J) = EESI(J) - SESI(J)+1。应当注意，针对该文件的不同源块的多个符号，独立于值 (SU,EU) 的范围，全部都落入彼此中的一个之中。

[0369] 可以如本申请所示出地，计算子块 L 中的子符号的大小 TS(L)。在下面的公式中，TS(L′) 是子块 L′ 中的子符号的大小。

[0370] 通过文件中的 (SBN,SuBN,ESI) 三元组 (J,L,I) 来索引的子符号的起始字节位置，可以如下所述地进行计算，其中，SESI(J)<=I<=EESI(J)：

[0371] T*(Sum{J′ = 0,…,J-1}NK(J′))+NK(J)*(Sum{L′ = 0,…,L-1}TS(L′))+TS(L)*(I-SESI(J)).

[0372] 在文件中通过 (J,L,I) 来索引的子符号的结束字节位置，是起始字节位置加 TS(L) - 1。

[0373] 举例而言，假定通过设置 SU = ceil(F/T) 和 EU = SU+Z*M-1 来形成文件，其中 M 是任何接收客户端所预期的每一个源块的最大符号丢失量，因此在该情况下，针对 Z 个源块中的每一个，在文件中存在精确地 M 个符号，该文件单独地由修复符号构成，在该文件中具有最小 ESI 值的修复符号在修复符号之中，具有最小的可能 ESI 值。在原始的 eMBMS 广播中，如果只发送源符号，那么单独地由修复符号构成的该文件可以上载到 HTTP web 服务器上，并由发出字节范围请求的客户端使用，以便提供针对 eMBMS 的修复服务。在某种更通用的意义上，如果在 eMBMS 会话中发送的最大 UOSI 是 X，那么用于修复的文件可以通过 UOSI 范围 (X′,Y′) 来指定，其中 X′>X,(Y′-X′)/Z 是任何客户端为了恢复一个源块或者源子块所需要请求的符号或者子符号的数量的上限。

[0374] 图 34 示出了处于其原始顺序的文件，其中在该示例中，将该文件划分成个源块，每一个源块被进一步划分成 N = 2 个子块，在前两个源块中的每一个里存在 4 个源符号，在第三源块中存在 3 个源符号。在图 34 中，每一个源符号使用 (J,L,I) 来标记，其中 J 是 SBN，L 是 SuBN，I 是该子符号的 ESI。

[0375] 图 35 示出了用于与图 34 相对应的文件的源符号，其中以 UOSI 顺序来列出这些源符号。因此，UOSI = 0 的源符号包括标记为 (0,0,0) 和 (0,1,0) 的两个子符号，UOSI = 5 的源符号包括标记为 (2,0,1) 和 (2,1,1) 的两个子符号。

[0376] 图 36 针对图 34 中所示出的文件，示出了 UOSI 范围从 11 到 19 的修复符号。可以根据原始文件（即，根据在图 35 中所示出的源符号）来生成这些修复符号。每一个修复符号包括：使用相同的 ESI，根据给定的源块的两个子块中的每一个来生成的两个子符号，例如，UOSI = 16 的修复符号包括标记为 (1,0,5) 和 (1,1,5) 的两个子符号。

[0377] 图 37 针对图 34 中所示出的文件，示出了与 UOSI 范围 11 到 19 相对应的扩展原始顺序 HTTP 文件格式。在该排序中，包括所指定的 UOSI 范围中的符号的子符号，首先根据 SBN

进行排序，接着根据 SuBN 进行排序，接着根据 ESI 进行排序。因此，针对给定的子块的所有子符号在扩展的原始顺序 HTTP 文件格式内是连续的。应当注意，在该示例中，对于包括 SBN ＝ 0 和 SBN ＝ 1 的源块的子块的子符号，ESI 范围是 4 到 6，而对于包括 SBN ＝ 2 的源块的子块的子符号，ESI 范围是 3 到 5。应当注意，扩展的原始顺序 HTTP 文件格式支持使用单一字节范围，从单一子块请求任意数量的子符号，直到文件中针对该子块的子符号的数量。

[0378] 此外，扩展的原始顺序 HTTP 文件格式还可以用于生成包括源符号和修复符号的混合的文件。例如，图 38 针对于图 34 中所示出的文件，示出了 UOSI 范围 8 到 12 的扩展原始顺序 HTTP 文件格式。应当注意，在该示例中，UOSI8、9 和 10 对应于源符号，UOSI11 和 12 对应于修复符号。此外，还应当注意，在该示例中，针对源块中的一些，存在不同数量的符号，例如，对于 SBN ＝ 0 和 SBN ＝ 2 的源块，存在 2 个符号，而对于 SBN ＝ 1 的源块，存在 1 个符号。

[0379] 在一些情况下，可能期望为给定的源文件，提供一个以上的扩展顺序 HTTP 文件。例如，在不同的 CDN 或者在不同的区域，有不同的文件可离线使用。例如，可以存在与原始源文件相对应的 UOSI 范围为 [0,19,999] 的一个文件，可以存在 UOSI 范围分别为 [20,000,22,000]、[22,001,26,354] 和 [45,651,64,356] 的三个其它文件。替代地，可以存在与原始源文件相对应的多个文件，例如，针对具有个源符号的源文件，UOSI 范围分别为 [0,7,000]、[7,001,14,000]、[14,000,19,999] 的三个文件。再举一个例子，可以存在具有可用的重叠 UOSI 范围的 2 个文件，例如，UOSI 范围分别为 [0,15,000] 和 [10,000,30,000]。在一些情况下，扩展的原始顺序 HTTP 文件中的符号可以是源符号和修复符号的混合，例如，UOSI 范围是 [10,000,30,000]，而源文件中的源符号的数量是 20,000。

[0380] 针对原始顺序 HTTP 文件的 HTTP 字节范围请求的示例

[0381] MBMS UE 可以使用如 RFC2616 中所规定的传统 HTTP/1.1GET 或者部分 GET 请求，来分别请求引用的资源的源符号或者子符号的全部或者一个子集。如果 MBMS UE 从支持字节范围请求的文件修复服务器请求符号，则使用这些消息格式，即，将其 URI 列成为“byteRangeServiceURI”或者“priorityByteRangeServiceURI”，或者当直接从 FLUTE FDT 实例中的“Content-Location（内容－位置）”属性的服务器 URI 部分，请求源数据时。

[0382] 对于原始顺序 HTTP 文件，当 MBMS UE 请求要发送的资源的所有源符号时，其使用 HTTP GET 请求。如果 MBMS UE 只请求源符号或者子符号的一个子集的传输时，该 UE 可以使用具有范围请求头部的 HTTP 部分 GET 请求，如 RFC2626 的 14.35.2 中所规定的。MBMS UE 将这些特定的源符号或者子符号指示成如 RFC2616 的 14.35.1 中所规定的字节范围规范 (byte-range-spec)。

[0383] 对于消息效率而言，HTTP GET 方法使 UE 能够在单一的部分 GET 请求中包括多个字节范围请求。如果 UE 在单一请求中包括多个字节范围，则 HTTP GET 请求在长度上应当不超过 2048 字节，以避免被 HTTP 服务器截短。

[0384] 如果 MBMS UE 确定其可以在源符号或者子符号的多个子集之中进行选择，MBMS UE 应当请求具有可用的最低 ESI 值的子集，即，分别从源块或者源子块的开始处，选择丢失的源符号或者子符号。这提高了 HTTP 文件修复服务器的高速缓存效率。其它策略也是可以的，例如，MBMS UE 请求具有可用的最高 ESI 值的子集。

[0385] 如果在特定的 MBMS 下载会话中下载一个以上的文件，并且如果 MBMS 客户端需要

在该会话中接收的一个以上的文件的修复数据，那么 MBMS 客户端可以发送针对每一个文件的单独 HTTP GET 请求。

[0386] 例如，假定在 MBMS 下载会话中，FLUTE FDT 实例中的“Content-Location”属性被设置为 www.example.com/news/sports.3gp 的 3gp 文件，将使用对象大小 F = 20,000,000 字节、对齐因子 Al = 4 字节、符号大小 T = 1320 字节、源块的数量 Z = 2、以及每一个源块的子块的数量为 N = 1 的 FEC OTI 来传输。在该示例中，该文件中的源符号的总数是 KT = 15,152，SBN = 0-11 的前 12 个源块具有 758 个源符号，SBN12-19 的剩余 8 个源块具有 757 个源符号。

[0387] 在该 MBMS 下载会话之后，假定 MBMS 客户端认识到其没有接收到来自于 SBN = 5 的源块的 ESI 为 12-29 的 18 个连续的源符号，以及来自于 SBN = 19 的最后源块的 ESI 为 27-36 的 10 个连续的源符号，为了对这些源块进行解码，其至少需要来自这些源块中的每一个源块的更多符号。对于 SBN = 5 的源块来说，ESI = 12 的第一丢失的源符号的起始字节位置，可以如图 39 中所示地进行计算。此外，图 39 还示出了如何对 ESI = 29 的最后的丢失符号的结束字节位置进行计算。在图 39 的表中的最后一行，还示出了针对 SBN = 19 的源块的丢失符号的起始和结束字节位置。

[0388] 如果所选定的修复服务器接受字节范围请求（例如，服务器 URI 使用相关联的传输过程元数据分段中的“byteRangeServiceURI”元素来标识），其服务器 URI 是 http://httprepair1.example.com，则针对修复服务器的 HTTP GET 请求是如下所述：

[0389] GET/www.example.com/news/sports.3gp HTTP/1.1

[0390] Range:bytes = 5018640-5042399,19037040-19050239

[0391] Host:httprepair1.example.com

[0392] 可以通过多种不同的方法或者实施例，将扩展的原始顺序 HTTP 文件格式的该文件的 UOSI 范围发送给 UE。在一个实施例中，例如，通过增加新的属性“Start-UOSI”和“End-UOSI”，可以将其发送成与该文件相关联的 FLUTE 实例中的文件描述表的一部分。在另一个实施例中，可以将 UOSI 范围直接编码在文件的 URL 中，例如，通过向 URL 插入 / 附加“start_uosi = xxx”和“end_uosi = yyy”属性。替代地，可以通过与文件的传输会话相关联的会话描述协议 (SDP)、媒体呈现描述 (MPD) 或者用户服务描述 (USD)，来发送文件的 UOSI 范围，例如，通过将“Start-UOSI”和“End-UOSI”的新属性增加到这些描述中的一个里。

[0393] 除了 UOSI 范围之外，还可以发送 UE 为了对该文件格式进行适当地解码所需要的辅助文件格式参数。这些另外的参数的示例包括一些 FEC 对象传输信息。此外，还可以通过规定用于这些辅助文件格式参数的属性，在 FLUTE 实例的文件描述表 (FDT) 中发送这些参数。在另一个实施例中，可以直接将辅助文件格式参数编码在文件的 URL 中。替代地，可以通过与文件的传输会话相关联的会话描述协议 (SDP)、媒体呈现描述 (MPD) 或者用户服务描述 (USD)，来发送辅助文件格式参数，例如，通过将用于这些参数的新属性增加到这些描述中的一个里。

[0394] 在一个实施例中，可以将 FEC OTI 信息、UOSI 范围或者任何通用辅助文件格式信息发送成 MIME 类型。例如，可以根据 RFC6381，使用字符串“application/mp4profiles = ‘flut’ fec-oti = ‘XYZ’”来发送 FEC OTI 参数，来指示 FLUTE 符号已被包括到具有指定的

FEC OTI 值的 MP4 文件中。在另一个实施例中，规定一种新的文件格式，将诸如“fec-oti”的参数增加到 MIME 类型注册中。该信令的一个例子是字符串“application/flut fec-oti ＝‘XYZ’”。

[0395] 在一些实施例中，虽然并不要求，但存储在 HTTP 服务器上的文件的辅助文件格式参数，与通过广播所发送的文件的辅助文件格式参数相同。在替代的实施例中，对于被广播的文件和存储在 HTTP 服务器上以用于实现修复的文件来说，这些辅助文件格式参数中的一些可以是不同的。

[0396] 例如，针对于 HTTP 服务器上存储的文件格式的 FEC OTI，可以与用于对广播进行格式化的 FEC OTI 不相同。一种实际示例是：当在不同的区域中，使用不同的 FEC OTI 参数来对广播数据进行编码时，在不同的覆盖区域之间移动的设备，可以具有被路由到不同的服务器的自己的单播修复请求，其中这些不同的服务器根据该设备的当前位置来服务差别化的编码文件。为了区分这些不同的参数值集合，可以直接将参数值编码到在修复服务器上存储的文件名或者 URI 中，如上所述。替代地，当作为 FDT、SDP、MPD 或 USD 的一部分来发送给终端时，可以使用不同的名称（例如，“FEC_OTI_bcast”和“FEC_OTI_file”）来规定其属性。随后，终端（例如，UE 终端）可以确定用于对从广播或者 HTTP 服务器获取的文件进行解码的适当过程。在一个实施例中，当使用不同的文件格式配置来呈现多个文件时，终端可以使用辅助文件参数来决定从 HTTP 服务器上的哪些文件获取数据。在其它实施例中，针对文件的 HTTP 请求可以包含用于指示所需要的 OTI 参数的实际指令；随后，HTTP 服务器可以使用嵌入的指令，对来自于正确编码的文件的请求进行服务，或者拒绝该请求（如果没有可用的该编码文件的话）。

[0397] 在另一个实施例中，可以需要终端根据 URL 来请求修复数据，其中该 URL 不会改变或者被格式化成发送辅助文件格式参数，例如，源文件位于具有互联网 URL 的服务器上。另一种方法是具有在 URL 中发送可选的参数的能力，并且当没有在 URL 中显式地发送这些参数时，假定缺省的参数值。例如，假定发送的潜在参数是格式类型（例如，SS 顺序 HTTP 文件格式、或者扩展的原始顺序 HTTP 文件格式）、FEC OTI 参数和 UOSI 范围。

[0398] 假定存在 URL 为 www.example.com 的单播文件，即，在 URL 中没有发送这些参数中的任何一个。在该实例中，客户端假定的缺省参数将是扩展的原始顺序 HTTP 文件格式、UOSI 范围＝ 0、…、FEC OTI ＝广播 FEC OTI，即，其仅是具有原始顺序的原始源文件。

[0399] 相反，如果发送了 UOSI 范围，例如，向客户端提供的 URL 是 www.example.UOSI ＝ X_to_Y.com，那么该客户端将观察到 UOSI 范围是 X 到 Y，并使用该范围，由于没有发送 FEC OTI，因此其缺省地与该文件的广播版本的 FEC OTI 相同，并假定扩展的原始顺序 HTTP 文件格式。

[0400] 在第三示例中，如果在 URL 中发送了 UOSI 范围和 FEC OTI，例如 www.example.FEC_OTI ＝ (F,A1,T,Z,N).UOSI ＝ X_to_Y.com，那么在具有给定的 FEC OTI 和 UOSI 范围的情况下，假定扩展的原始顺序 HTTP 文件格式。

[0401] 在第四示例中，如果在 URL 中发送了该格式，即 www.example.Format ＝ SS.FEC_OTI ＝ (F,A1,T,Z,N).com，那么这向客户端说明该格式是 SS 顺序的 HTTP 文件格式，并提供了 FEC OTI（如果没有提供 FEC OTI 的话，则在该示例中，假定其与广播 FEC OTI 相同）。

[0402] 因此，通过包含可以在 URL 中发送的这些可选参数，人们可以实现下面操作：允许

已经具有与其相关联的 URL 的文件不修改其 URL(确保用于未发送的参数的缺省值与该文件的格式相匹配),或者允许发送这些参数中的一些或者全部的 URL。一种关键优点在于:未修改的 URL 可以用于通过互联网或者 Over-The-Top 的已经存在的可用文件,这是由于在该情况下,未发送的参数将缺省地指示该文件是原始源文件。转而,对于具有发送这些参数的 URL 的文件来说,这些文件很可能是专门形成的(例如,产生该文件的一部分的修复数据,或者对该文件的格式进行重新组织)。因为这些文件不是原始源文件,因此在 URL 中发送另外的参数是可接受的,这是由于当这些文件是专门形成的时,需要形成用于这些文件的 URL。

[0403] 对于扩展的原始顺序 HTTP 文件,当 MBMS UE 需要要发送的资源的所有符号时,其使用 HTTP GET 请求。如果 MBMS UE 只请求符号或者子符号的一个子集的传输,该 UE 可以使用具有范围请求头部的 HTTP 部分 GET 请求,如 RFC2626 的 14.35.2 中所规定的。MBMS UE 将这些特定的符号或者子符号指示成如 RFC2616 的 14.35.1 中所规定的字节范围规范(byte-range-spec)。

[0404] 例如,假定在 MBMS 下载会话中,FLUTE FDT 实例中的“Content-Location”属性被设置为 www.example.com/news/sports.3gp 的 3gp 文件,将使用对象大小 F = 20,000,000 字节、对齐因子 Al = 4 字节、符号大小 T = 1320 字节、源块的数量 Z = 20、以及每一个源块的子块的数量为 N = 1 的 FEC OTI 来传输。在该示例中,该文件中的源符号的总数是 KT = 15,152,SBN = 0-11 的前 12 个源块具有 758 个源符号,SBN12-19 的剩余 8 个源块具有 757 个源符号。假定在该示例中,在 MBMS 下载会话中发送源符号,与通过 HTTP 可用的修复文件相关联的 UOSI 范围是 [15,152,17,151],即,该修复文件包括:针对扩展原始顺序 HTTP 文件格式的 20 个源块中的每一个的前 100 个修复符号。

[0405] 在该 MBMS 下载会话之后,假定 MBMS 客户端认识到为了对这些源块进行解码,其需要 SBN = 5 的源块的另外 18 个符号,以及 SBN = 19 的最后源块的另外 10 个符号。对于 SBN = 5 的源块来说,文件中的 ESI = 758 的第一符号的起始字节位置,可以如图 40 中所示地进行计算。此外,图 40 还示出了如何对 ESI = 775 的最后符号的结束字节位置进行计算。在图 40 的表中的最后一行,还示出了针对 SBN = 19 的源块的符号的起始和结束字节位置。

[0406] 如果所选定的修复服务器接受字节范围请求(例如,服务器 URI 使用相关联的传输过程元数据分段中的“byteRangeServiceURI”元素来标识),其服务器 URI 是 http://httprepair1.example.com,则针对修复服务器的 HTTP GET 请求是如下所述:

[0407] GET/www.example.com/news/sports.3gp HTTP/1.1

[0408] Range:bytes = 5002800-5026559,19001400-19014599

[0409] Host:httprepair1.example.com

[0410] 在使用子块划分的另一个示例中,假定在 MBMS 下载会话中,FLUTE FDT 实例中的“Content-Location”属性被设置为 www.example.com/news/international.3gp 的 3gp 文件,将使用对象大小 F = 20,000,000 字节、对齐因子 Al = 4 字节、符号大小 T = 1320 字节、源块的数量 Z = 2、以及每一个源块的子块的数量为 N = 12 的 FEC OTI 来传输。在该示例中,该文件中的源符号的总数是 KT = 15,152, SBN = 0 的第一源块和 SBN = 1 的第二源块具有 7,576 个源符号。在该示例中,每一个源块被进一步划分成 12 个子块,每一个子块具有 7,576 个源子符号,其中前 6 个子块具有大小为 112 字节的子符号,剩余的 6 个子块具有

大小为 108 字节的子符号。假定在该示例中，在 MBMS 下载会话中发送源符号，与通过 HTTP 可用的修复文件相关联的 UOSI 范围是 [15,152,17,151]，即，该修复文件包括：针对扩展原始顺序 HTTP 文件格式的 2 个源块中的每一个的前 1000 个修复符号。

[0411] 假定在该 MBMS 下载会话之后，MBMS 客户端认识到为了对这些源块进行解码，其需要 SBN = 0 的第一源块的另外 18 个符号，以及其需要 SBN = 1 的第二源块的另外 18 个符号。

[0412] 对于 SBN = 0 的第一源块来说，构成具有 ESI = 7576 的符号（在该修复文件中可用的第一符号）的 12 个子符号的起始字节位置，可以如图 41 中所示地进行计算。此外，图 41 还示出了如何对构成 ESI = 7593 的符号（18 符号中的最后一个）的 12 个子符号的结束字节位置进行计算。在图 41 的表中，“SuBN”指代子块编号。

[0413] 存在上面所描述的实施例的多种其它实施例和变型。作为本申请所描述的所有格式的一种变型的示例，除下面情形之外，可以使用相同的格式：以 ESI 的反向顺序来排序源块的符号和子符号。例如，如果一个文件被划分成两个源块，每一个具有两个子块，每一个子块具有三个子符号，那么该文件的原始顺序 HTTP 格式可以表示成 (0,0,0)、(0,0,1)、(0,0,2)、(0,1,0)、(0,1,1)、(0,1,2)、(1,0,0)、(1,0,1)、(1,0,2)、(1,1,0)、(1,1,1)、(1,1,2)，其中每一个三元组指示每一个子符号的 SBN、SuBN 和 ESI。处于反向顺序的该文件，可以表示成 (0,0,2)、(0,0,1)、(0,0,0)、(0,1,2)、(0,1,1)、(0,1,0)、(1,0,2)、(1,0,1)、(1,0,0)、(1,1,2)、(1,1,1)、(1,1,0)。类似的反向变型保持本申请所描述的所有其它文件格式。如果客户端正在下载能用于对来自两个不同服务器的内容进行重建的数据，则该文件的反向变型可以是有用的，其中该客户端正在下载或者接收能用于对来自第一服务器的内容（其是以一种文件格式存储的）进行重建的数据，并且该客户端正在下载或者接收能用于对来自第二服务器的内容（其是以一种反向格式存储的）进行重建的文件。如果来自两个服务器的数据的传输速度不同，或者是不可预测的，那么客户端可以简单地等待，直到组合来自这两个服务器的数据足够达到对内容进行重建，随后客户端可以终止该连接，或者简单地停止从这两个服务器接收另外的数据。由于一种格式是另一种格式的反向顺序，因此独立于客户端从每一个服务器接收到多少的该数据，客户端接收到允许该客户端对内容进行重建的很少冗余数据。

[0414] 存在指定扩展的原始顺序 HTTP 格式文件的格式（或者本申请所描述的多种其它格式）的多种其它变型。例如，可以使用字节范围来将数据指定为扩展的原始顺序 HTTP 格式文件，而不是 UOSI 范围。例如，如果 UOSI 范围是 (X,Y)，符号大小是 T，那么指定相同格式的字节范围，可以是在字节 X*T 处起始并在字节 (Y+1)*T-1 处结束的字节范围。指定该格式的其它变型包括：针对文件的每一个源块，指定 ESI 范围的显式列表。例如，如果一个文件具有 5 个源符号，那么该文件的格式的详细说明可以是 SBN = 0、ESI = 20-50、SBN = 1、ESI = 30-40、SBN = 2、ESI = 10-19、SBN = 3、ESI = 0-50、SBN = 4、ESI = 17-37。在该示例中，相同的源块可以用 ESI 的不同范围被列出多次，例如，可以将 SBN = 4、ESI = 55-65 增加到上面的列表中，随后在该格式的文件中，针对 SBN = 4 的源块，存在两个范围的 ESI。替代地，ESI 范围可以用字节范围来替代，其中在该情况下，各个字节范围都是相对于源块的。例如，在上面的示例中，如果符号大小是 1,000 字节，那么字节范围 30,000 - 40,999 将等同于 ESI 范围 30-40。可以将这些格式的规范的变型与本申请先前所描述的所有格式进

行组合。

[0415] 如上面所描述的，存在一些特定的示例。本领域普通技术人员在阅读本申请公开内容后，还可以想到另外的实施例。在其它实施例中，有利地，可以实现上面所公开的发明的组合或者子组合。为了说明目的，示出了组成部分的示例性排列，应当理解的是，在本发明的替代实施例中，可以预期对这些组成部分的组合、补充、重新排列等。因此，虽然已经针对示例性实施例描述了本发明，但本领域普通技术人员应当认识到，众多修改是有可能的。

[0416] 例如，本申请所描述的过程可以使用硬件组件、软件组件和／或其任意组合来实现。因此，应当将说明书和附图视作为示例性的，而不是限制意义的。但是，显然，可以在不脱离如权利要求书所阐述的本发明的更广精神和保护范围的基础上，对本发明做出各种修改和改变，本发明旨在覆盖落入所附权利要求书的保护范围之内的所有修改和等同物。

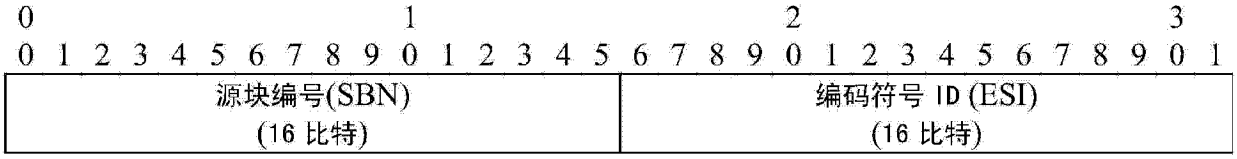


图 1A

Raptor-RFC5053 有效载荷 ID(现有技术)

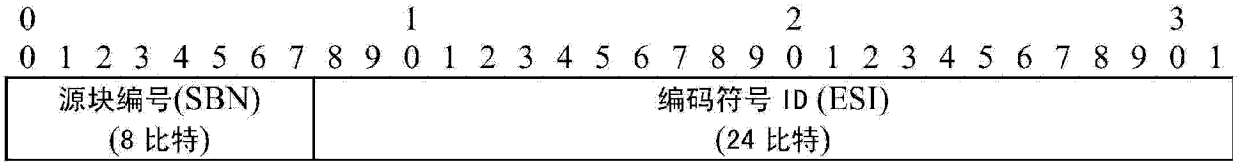


图 1B

Raptor 特定有效载荷 ID(现有技术)

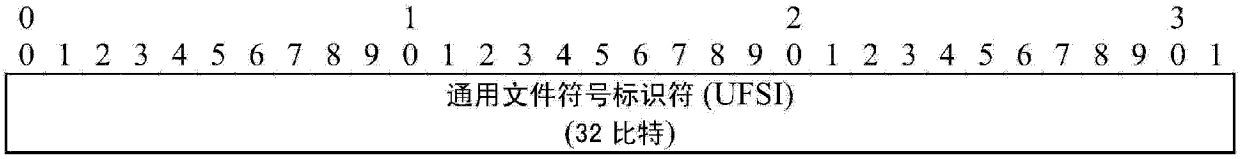


图 2

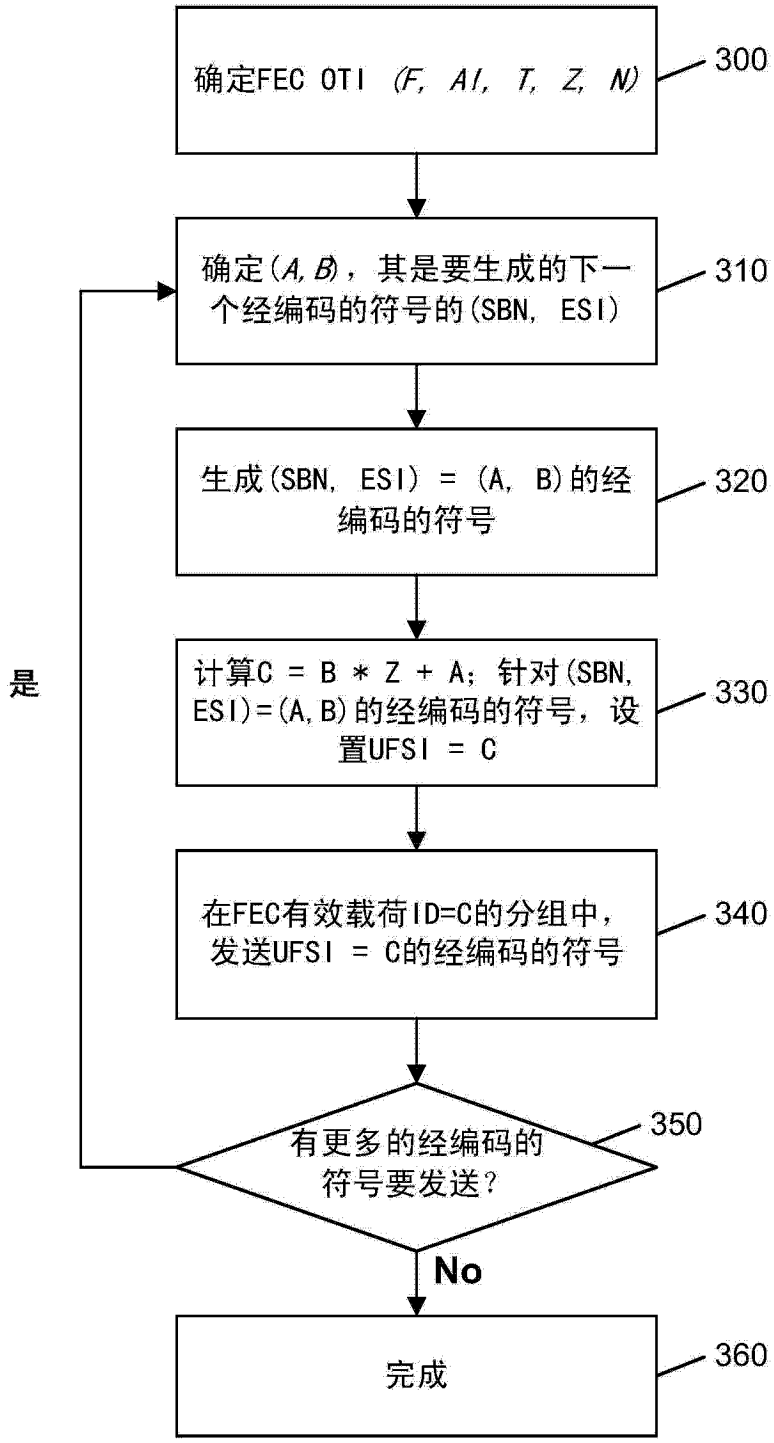

确定FEC OTI (F, AI, T, Z, N)
300
确定(A, B)，其是要生成的下一个经编码的符号的(SBN, ESI)
310
生成(SBN, ESI) = (A, B)的经编码的符号
320
计算C = B * Z + A；针对(SBN, ESI)=(A,B)的经编码的符号，设置UFSI = C
330
在FEC有效载荷ID=C的分组中，发送UFSI = C的经编码的符号
340
有更多的经编码的符号要发送？
350
是
No
完成
360

图 3

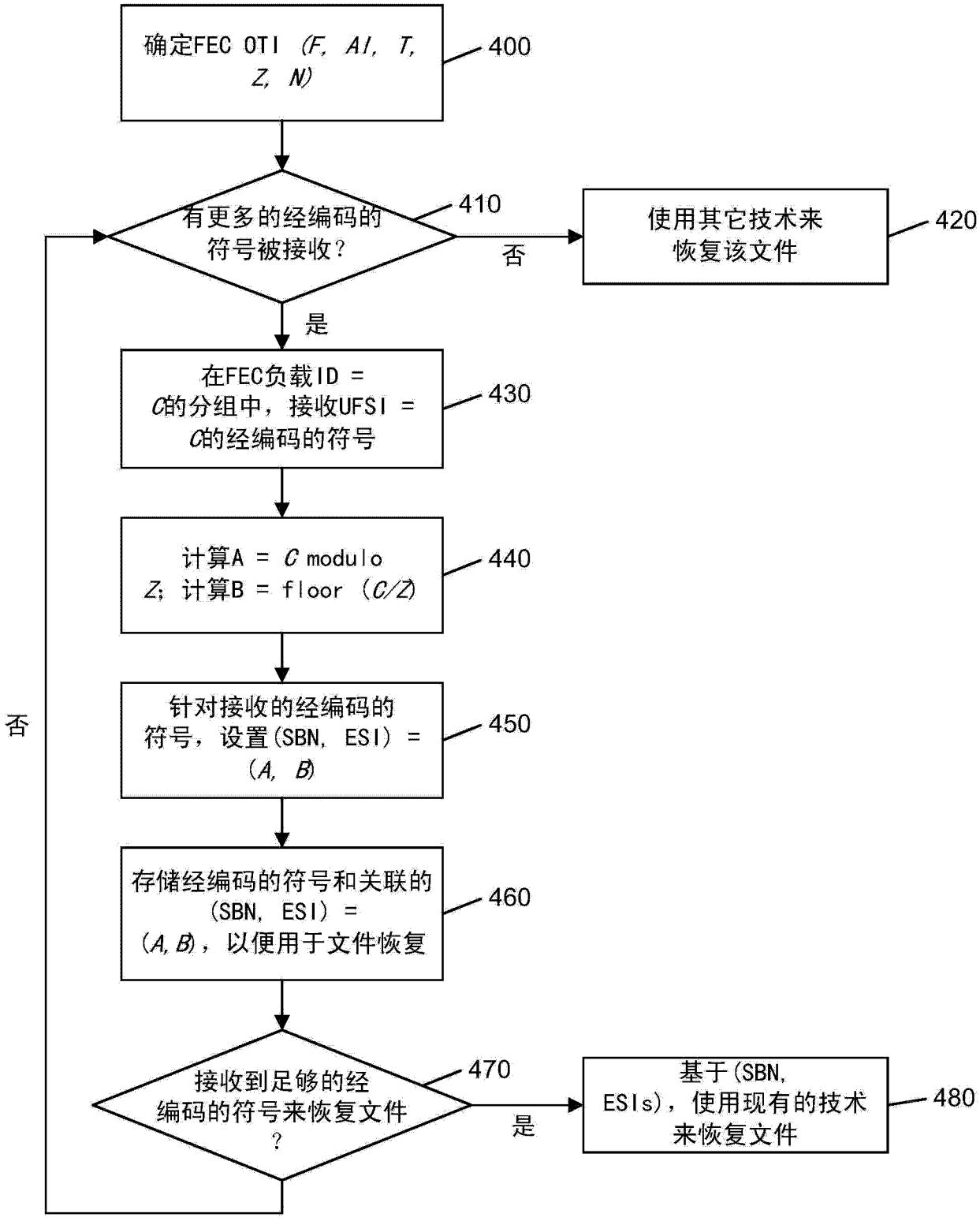

确定FEC OTI (F, AI, T, Z, N)
400
有更多的经编码的符号被接收？
410
否
使用其它技术来恢复该文件
420
是
在FEC负载ID = C的分组中，接收UFSI = C的经编码的符号
430
计算A = C modulo Z；计算B = floor (C/Z)
440
针对接收的经编码的符号，设置(SBN, ESI) = (A, B)
450
存储经编码的符号和关联的(SBN, ESI) = (A, B)，以便用于文件恢复
460
接收到足够的经编码的符号来恢复文件？
470
是
基于(SBN, ESIs)，使用现有的技术来恢复文件
480
否

图 4

ESI 0	ESI 1	ESI 2	ESI 3	ESI 4	ESI 5	
(0,0)	(0,1)	(0,2)	(0,3)	(0,4)	(0,5)	SBN 0
(1,0)	(1,1)	(1,2)	(1,3)	(1,4)	(1,5)	SBN 1
(2,0)	(2,1)	(2,2)	(2,3)	(2,4)	(2,5)	SBN 2
(3,0)	(3,1)	(3,2)	(3,3)	(3,4)		SBN 3
(4,0)	(4,1)	(4,2)	(4,3)	(4,4)		SBN 4

图 5A

SBN 0	UFSI 0	UFSI 5	UFSI 10	UFSI 15	UFSI 20	UFSI 25
SBN 1	UFSI 1	UFSI 6	UFSI 11	UFSI 16	UFSI 21	UFSI 26
SBN 2	UFSI 2	UFSI 7	UFSI 12	UFSI 17	UFSI 22	UFSI 27
SBN 3	UFSI 3	UFSI 8	UFSI 13	UFSI 18	UFSI 23	
SBN 4	UFSI 4	UFSI 9	UFSI 14	UFSI 19	UFSI 24	

图 5B

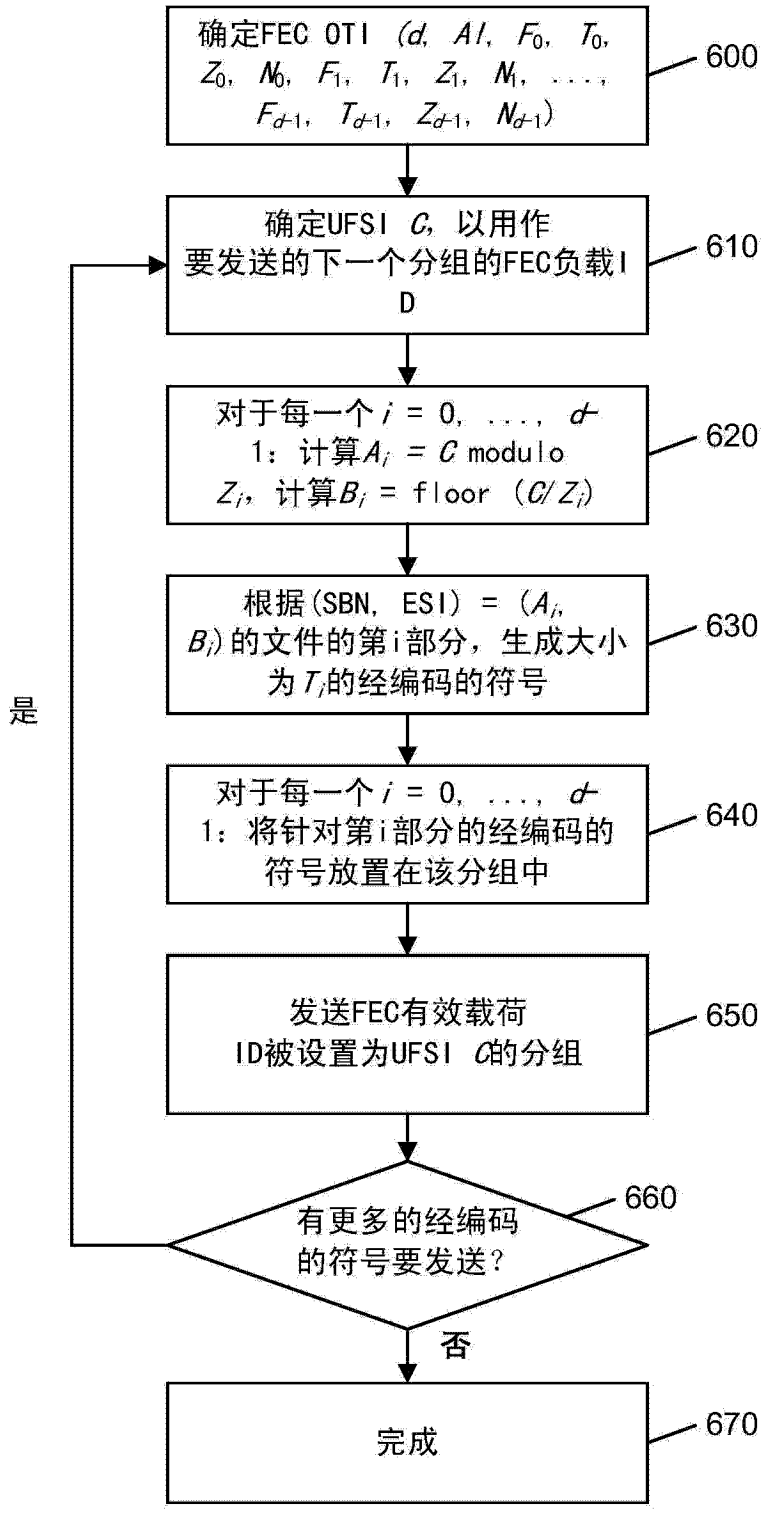

确定FEC OTI (d, Al, F_0, T_0, Z_0, N_0, F_1, T_1, Z_1, N_1, ..., F_{d-1}, T_{d-1}, Z_{d-1}, N_{d-1})
600
确定UFSI C，以用作要发送的下一个分组的FEC负载ID
610
对于每一个 i = 0, ..., d-1：计算A_i = C modulo Z_i，计算B_i = floor (C/Z_i)
620
根据(SBN, ESI) = (A_i, B_i)的文件的第i部分，生成大小为T_i的经编码的符号
630
对于每一个 i = 0, ..., d-1：将针对第i部分的经编码的符号放置在该分组中
640
发送FEC有效载荷ID被设置为UFSI C的分组
650
有更多的经编码的符号要发送？
660
是
否
完成
670

图 6

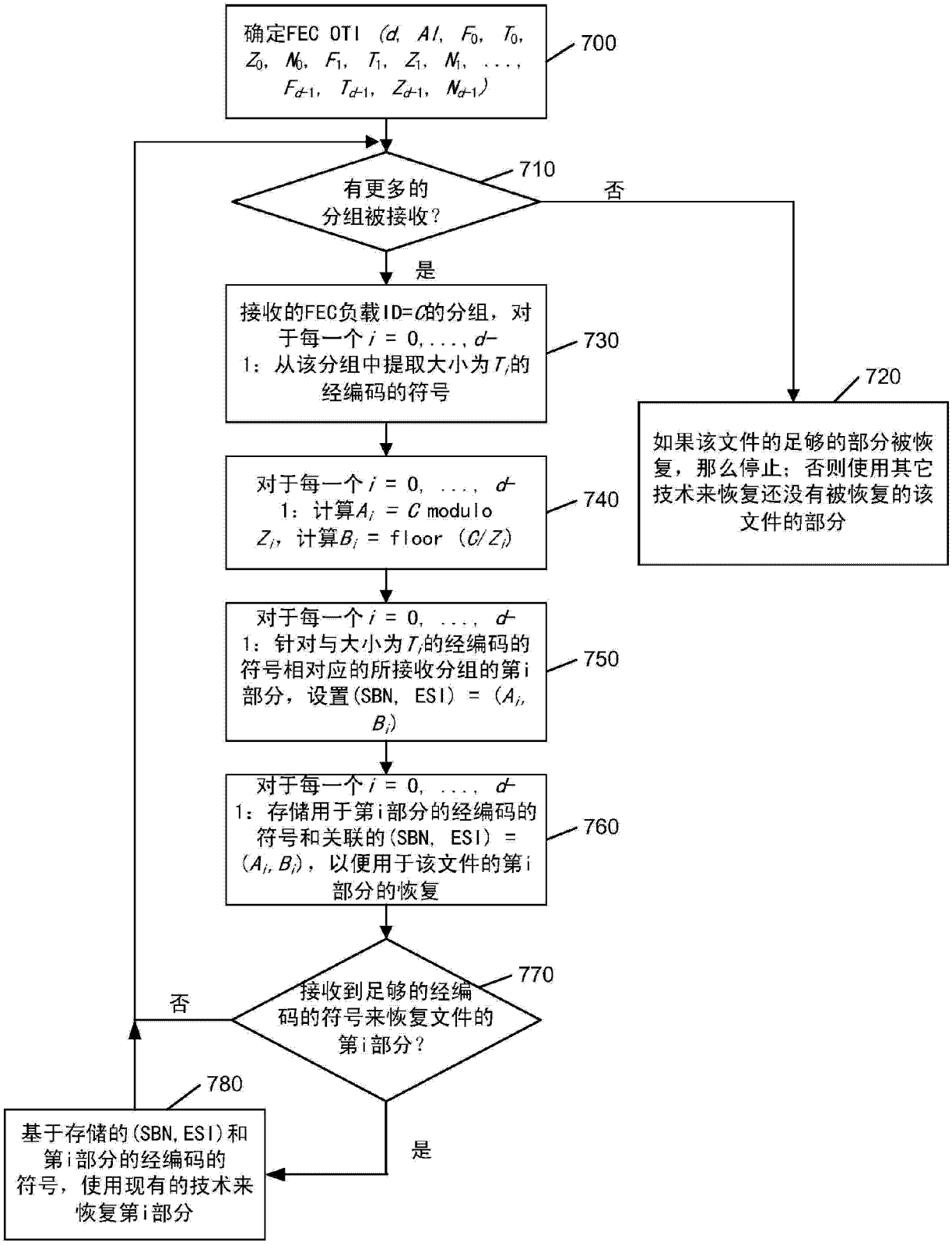

确定FEC OTI (d, Al, F_0, T_0, Z_0, N_0, F_1, T_1, Z_1, N_1, ..., F_{d-1}, T_{d-1}, Z_{d-1}, N_{d-1})
700
有更多的分组被接收?
710
否
是
接收的FEC负载ID=C的分组，对于每一个 i = 0, ..., d-1：从该分组中提取大小为 T_i 的经编码的符号
730
如果该文件的足够的部分被恢复，那么停止；否则使用其它技术来恢复还没有被恢复的该文件的部分
720
对于每一个 i = 0, ..., d-1：计算 A_i = C modulo Z_i，计算 B_i = floor (C/Z_i)
740
对于每一个 i = 0, ..., d-1：针对与大小为 T_i 的经编码的符号相对应的所接收分组的第i部分，设置(SBN, ESI) = (A_i, B_i)
750
对于每一个 i = 0, ..., d-1：存储用于第i部分的经编码的符号和关联的(SBN, ESI) = (A_i, B_i)，以便用于该文件的第i部分的恢复
760
接收到足够的经编码的符号来恢复文件的第i部分?
770
否
是
基于存储的(SBN, ESI)和第i部分的经编码的符号，使用现有的技术来恢复第i部分
780

图 7

文件的第1部分的结构

(0,0)	(0,1)	(0,2)	(0,3)	(0,4)	(0,5)	SBN 0
(1,0)	(1,1)	(1,2)	(1,3)	(1,4)	(1,5)	SBN 1
(2,0)	(2,1)	(2,2)	(2,3)	(2,4)	(2,5)	SBN 2
(3,0)	(3,1)	(3,2)	(3,3)	(3,4)		SBN 3
(4,0)	(4,1)	(4,2)	(4,3)	(4,4)		SBN 4
ESI 0	ESI 1	ESI 2	ESI 3	ESI 4	ESI 5	

图 8A

文件的第2部分的结构

(0,0)	(0,1)	(0,2)	(0,3)	SBN 0
(1,0)	(1,1)	(1,2)	(1,3)	SBN 1
(2,0)	(2,1)	(2,2)	(2,3)	SBN 2
(3,0)	(3,1)	(3,2)		SBN 3
ESI 0	ESI 1	ESI 2	ESI 3	

图 8B

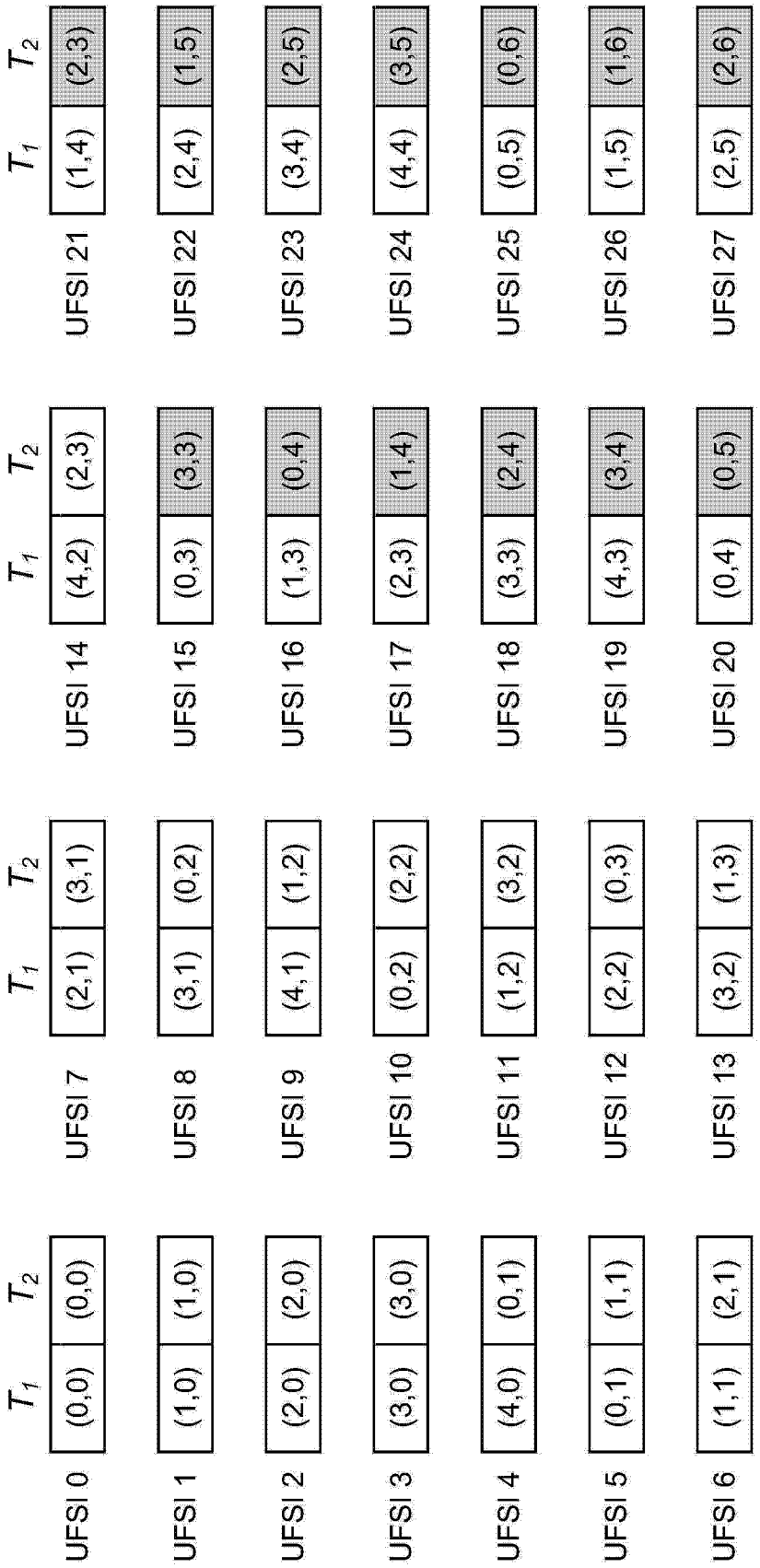

UFSI 0 (0,0) (0,0)
UFSI 1 (1,0) (1,0)
UFSI 2 (2,0) (2,0)
UFSI 3 (3,0) (3,0)
UFSI 4 (4,0) (0,1)
UFSI 5 (0,1) (1,1)
UFSI 6 (1,1) (2,1)
UFSI 7 (2,1) (3,1)
UFSI 8 (3,1) (0,2)
UFSI 9 (4,1) (1,2)
UFSI 10 (0,2) (2,2)
UFSI 11 (1,2) (3,2)
UFSI 12 (2,2) (0,3)
UFSI 13 (3,2) (1,3)
UFSI 14 (4,2) (2,3)
UFSI 15 (0,3) (3,3)
UFSI 16 (1,3) (0,4)
UFSI 17 (2,3) (1,4)
UFSI 18 (3,3) (2,4)
UFSI 19 (4,3) (3,4)
UFSI 20 (0,4) (0,5)
UFSI 21 (1,4) (2,3)
UFSI 22 (2,4) (1,5)
UFSI 23 (3,4) (2,5)
UFSI 24 (4,4) (3,5)
UFSI 25 (0,5) (0,6)
UFSI 26 (1,5) (1,6)
UFSI 27 (2,5) (2,6)
T1 T2

图 9

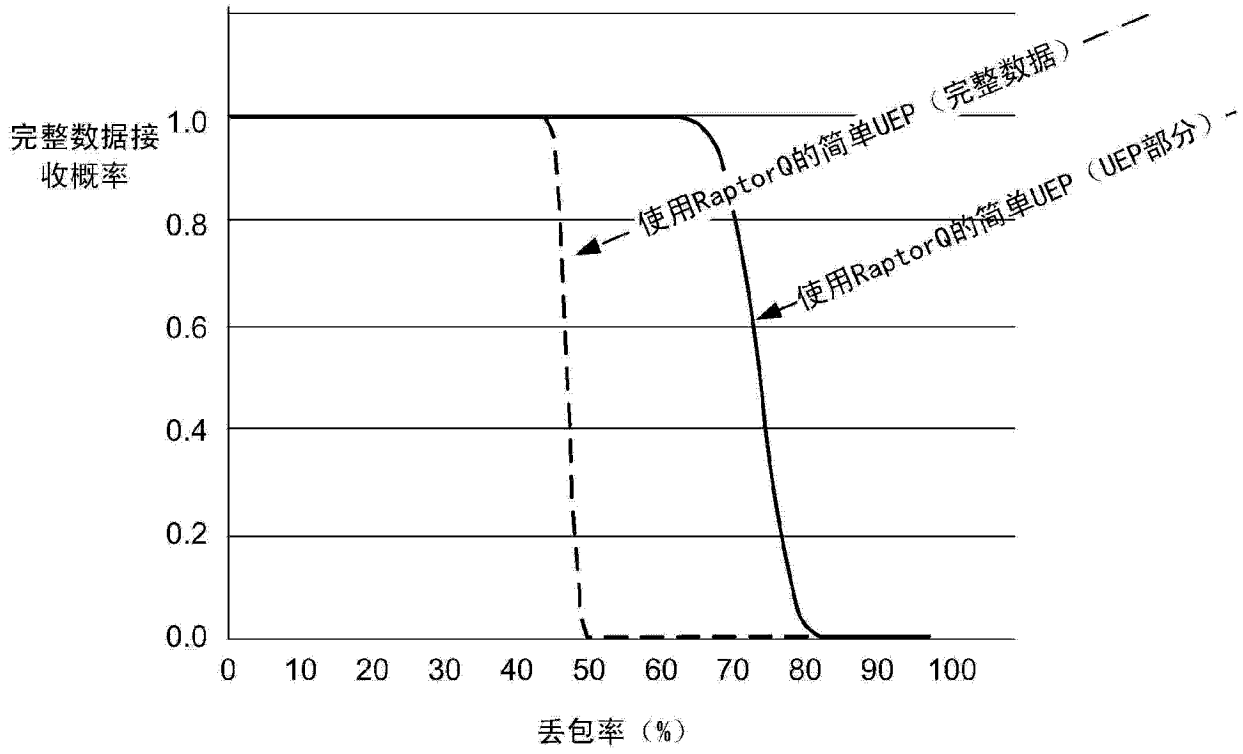

完整数据接收概率
1.0
0.8
0.6
0.4
0.2
0.0
0
10
20
30
40
50
60
70
80
90
100
丢包率（%）
使用RaptorQ的简单UEP（完整数据）
使用RaptorQ的简单UEP（UEP部分）

图 10

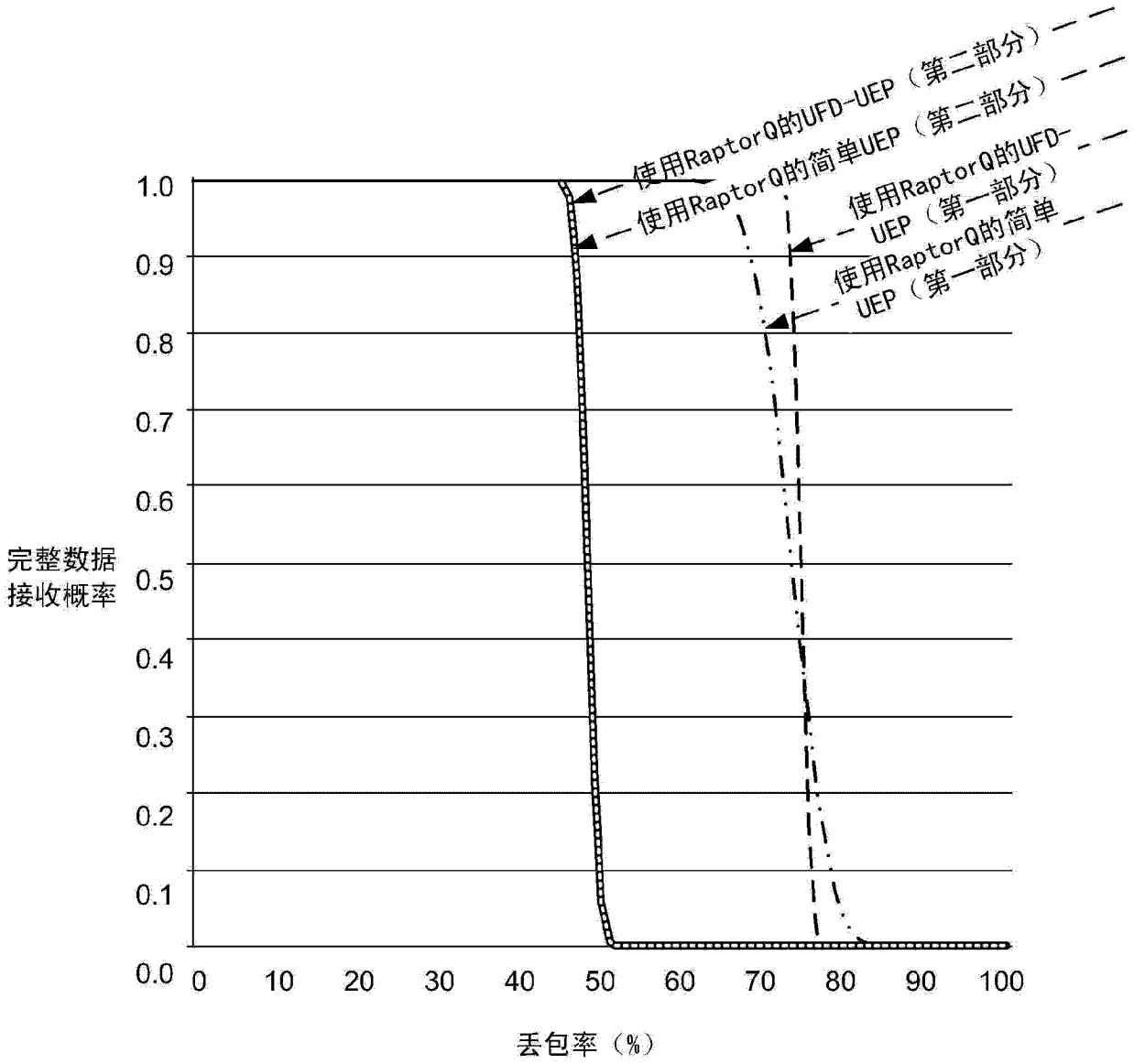

使用RaptorQ的UFD-UEP（第二部分）
使用RaptorQ的简单UEP（第二部分）
使用RaptorQ的UFD-UEP（第一部分）
使用RaptorQ的简单UEP（第一部分）
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
完整数据接收概率
0
10
20
30
40
50
60
70
80
90
100
丢包率（%）

图 11

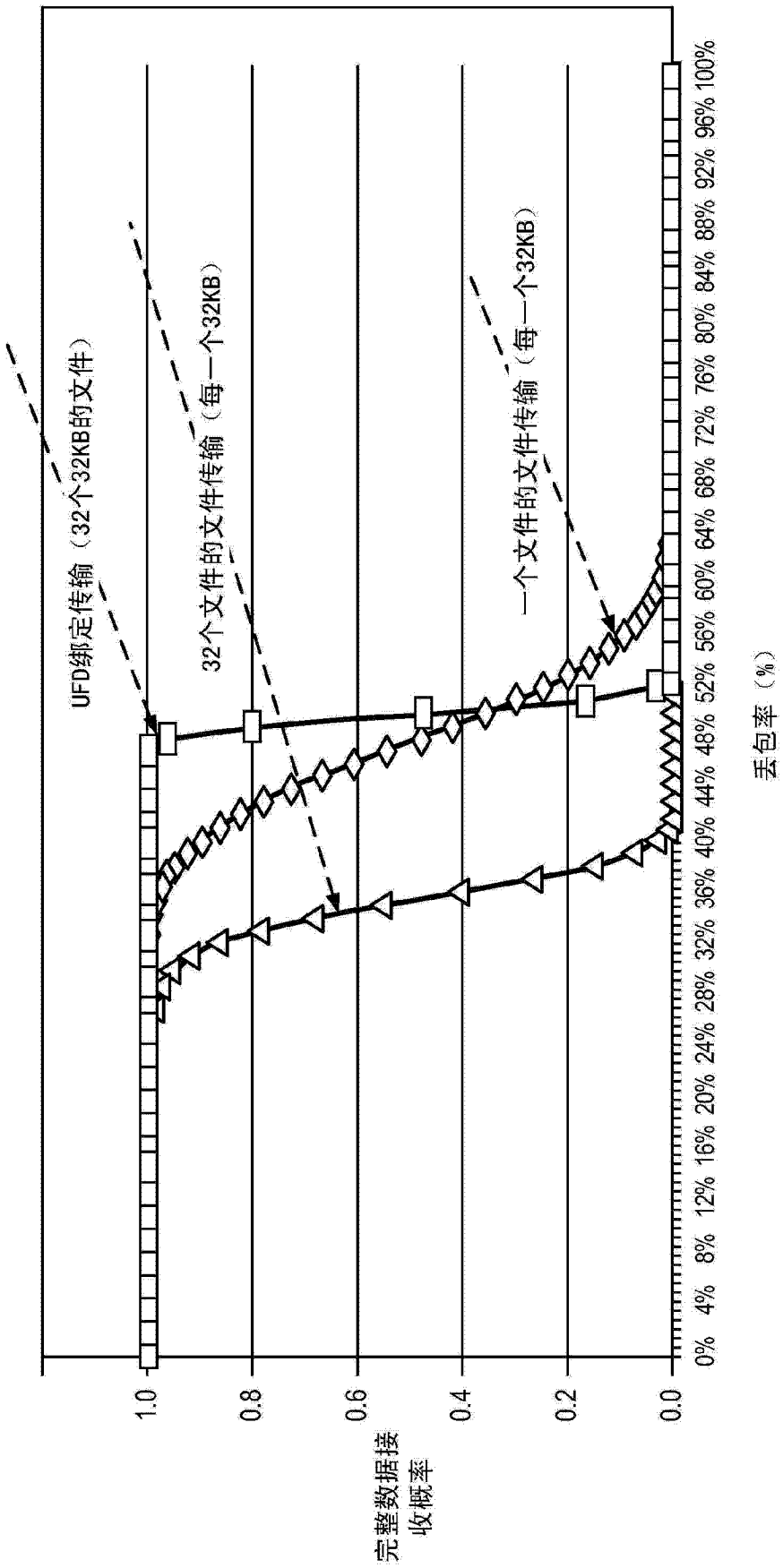

UFD绑定传输（32个32KB的文件）
32个文件的文件传输（每一个32KB）
一个文件的文件传输（每一个32KB）
完整数据接收概率
1.0
0.8
0.6
0.4
0.2
0.0
0% 4% 8% 12% 16% 20% 24% 28% 32% 36% 40% 44% 48% 52% 56% 60% 64% 68% 72% 76% 80% 84% 88% 92% 96% 100%
丢包率（%）

图 12

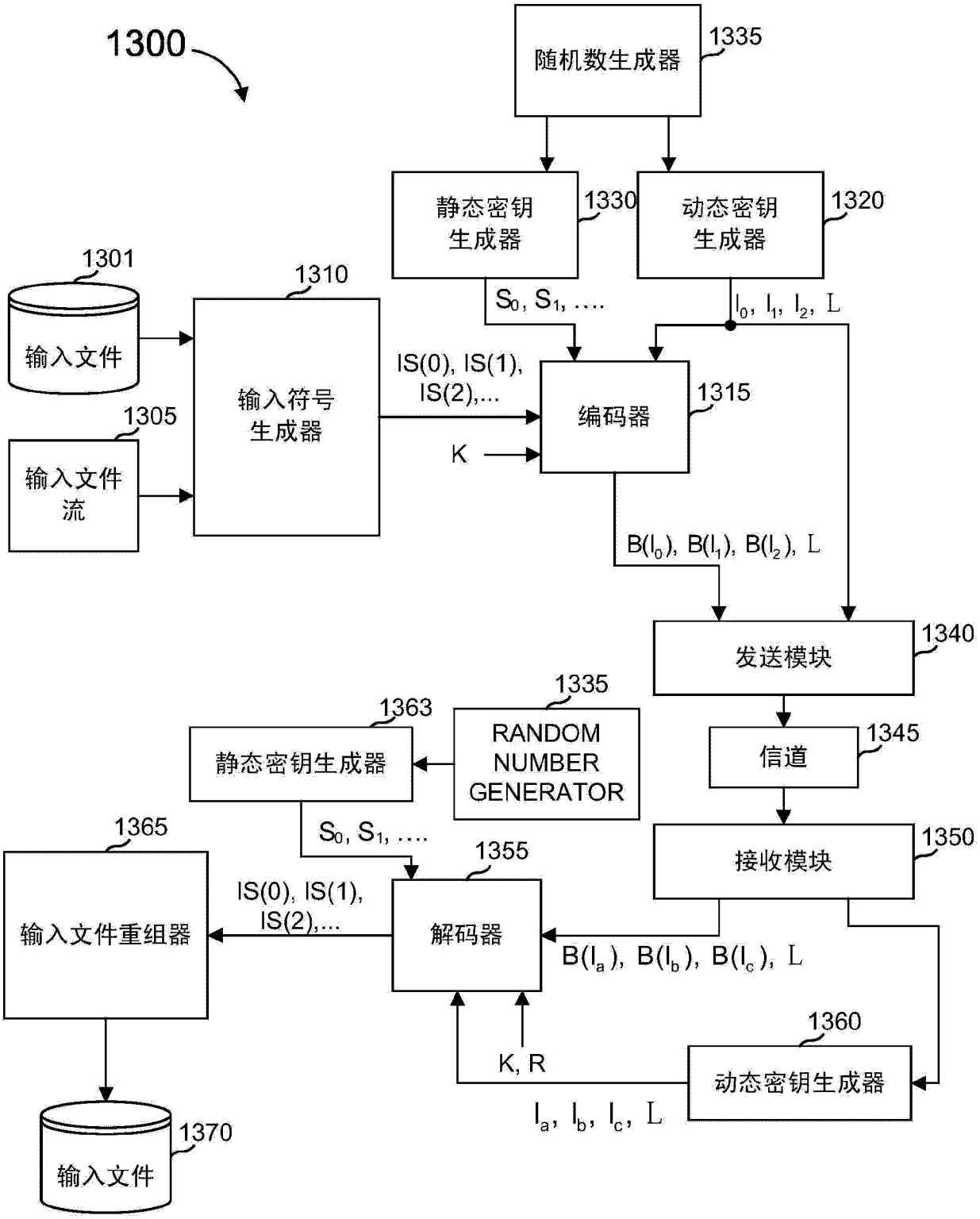

1300
随机数生成器
1335
静态密钥生成器
1330
动态密钥生成器
1320
1301
输入文件
1310
输入符号生成器
1305
输入文件流
S0, S1, ...
I0, I1, I2, L
IS(0), IS(1), IS(2),...
编码器
1315
K
B(I0), B(I1), B(I2), L
发送模块
1340
1335
RANDOM NUMBER GENERATOR
1363
静态密钥生成器
信道
1345
1365
S0, S1, ...
1355
接收模块
1350
输入文件重组器
IS(0), IS(1), IS(2),...
解码器
B(Ia), B(Ib), B(Ic), L
1360
K, R
动态密钥生成器
Ia, Ib, Ic, L
1370
输入文件

图 13

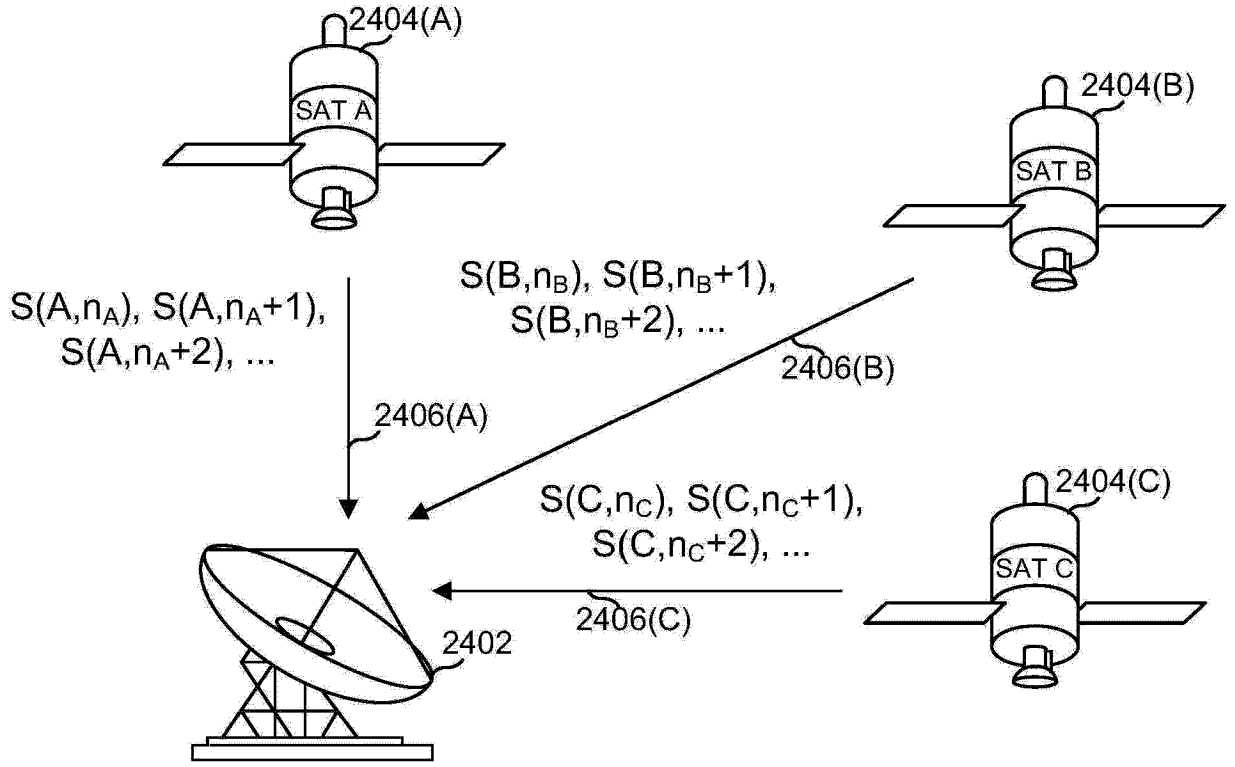

2404(A)
SAT A
2404(B)
SAT B
S(A,nA), S(A,nA+1),
S(A,nA+2), ...
S(B,nB), S(B,nB+1),
S(B,nB+2), ...
2406(B)
2406(A)
S(C,nC), S(C,nC+1),
S(C,nC+2), ...
2404(C)
SAT C
2406(C)
2402

图 14

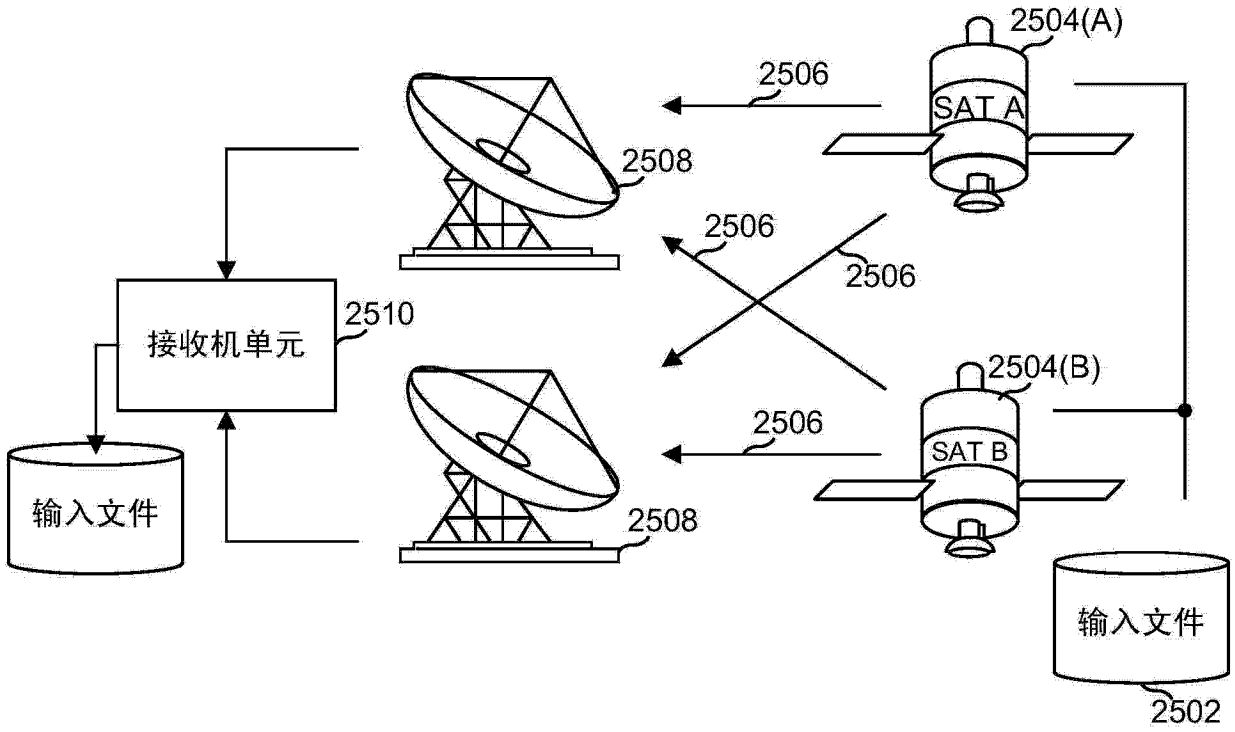

2504(A)
SAT A
2506
2508
2506
2506
2510
接收机单元
2504(B)
SAT B
2506
2508
输入文件
输入文件
2502

图 15

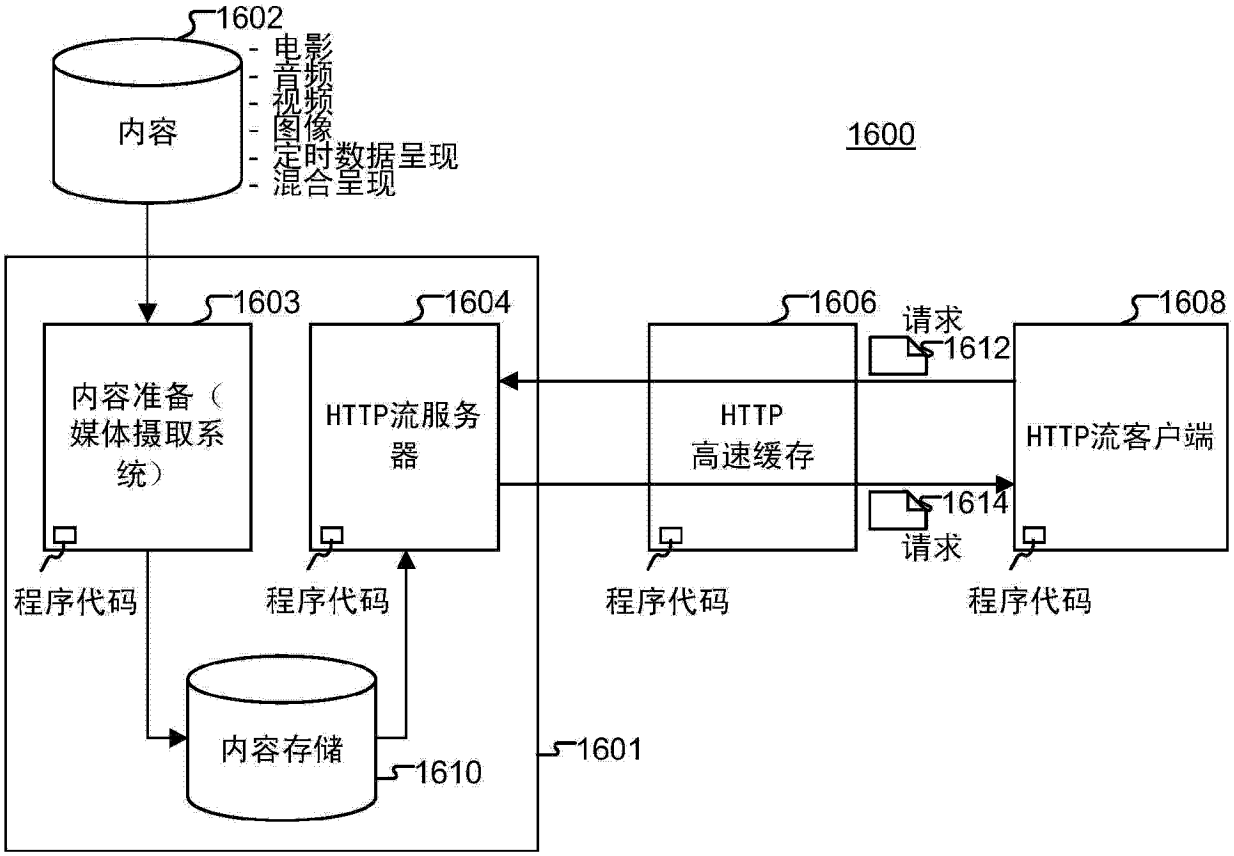

1602
内容
- 电影
- 音频
- 视频
- 图像
- 定时数据呈现
- 混合呈现
1600
1603
内容准备（媒体摄取系统）
1604
HTTP流服务器
1606
HTTP高速缓存
请求
1612
1608
HTTP流客户端
1614
请求
程序代码
程序代码
程序代码
程序代码
内容存储
1610
1601

图 16

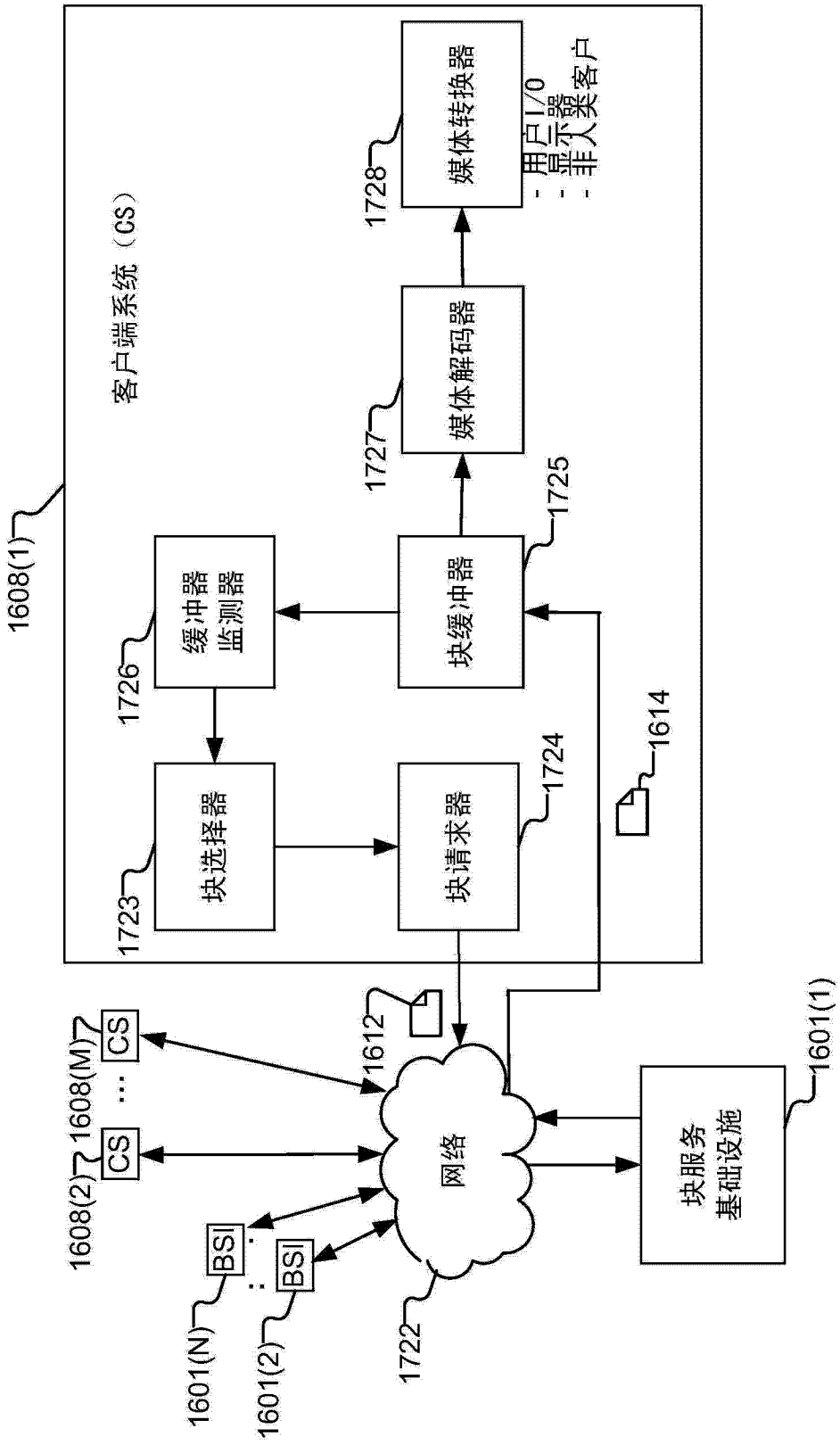

客户端系统（CS）
1608(1)
媒体转换器
1728
- 用户I/O
- 显示器
- 非人类客户
媒体解码器
1727
块缓冲器
1725
缓冲器
监测器
1726
块选择器
1723
块请求器
1724
1614
1612
网络
1722
块服务
基础设施
1601(1)
1608(M)
CS
1608(2)
CS
BSI
1601(N)
BSI
1601(2)

图 17

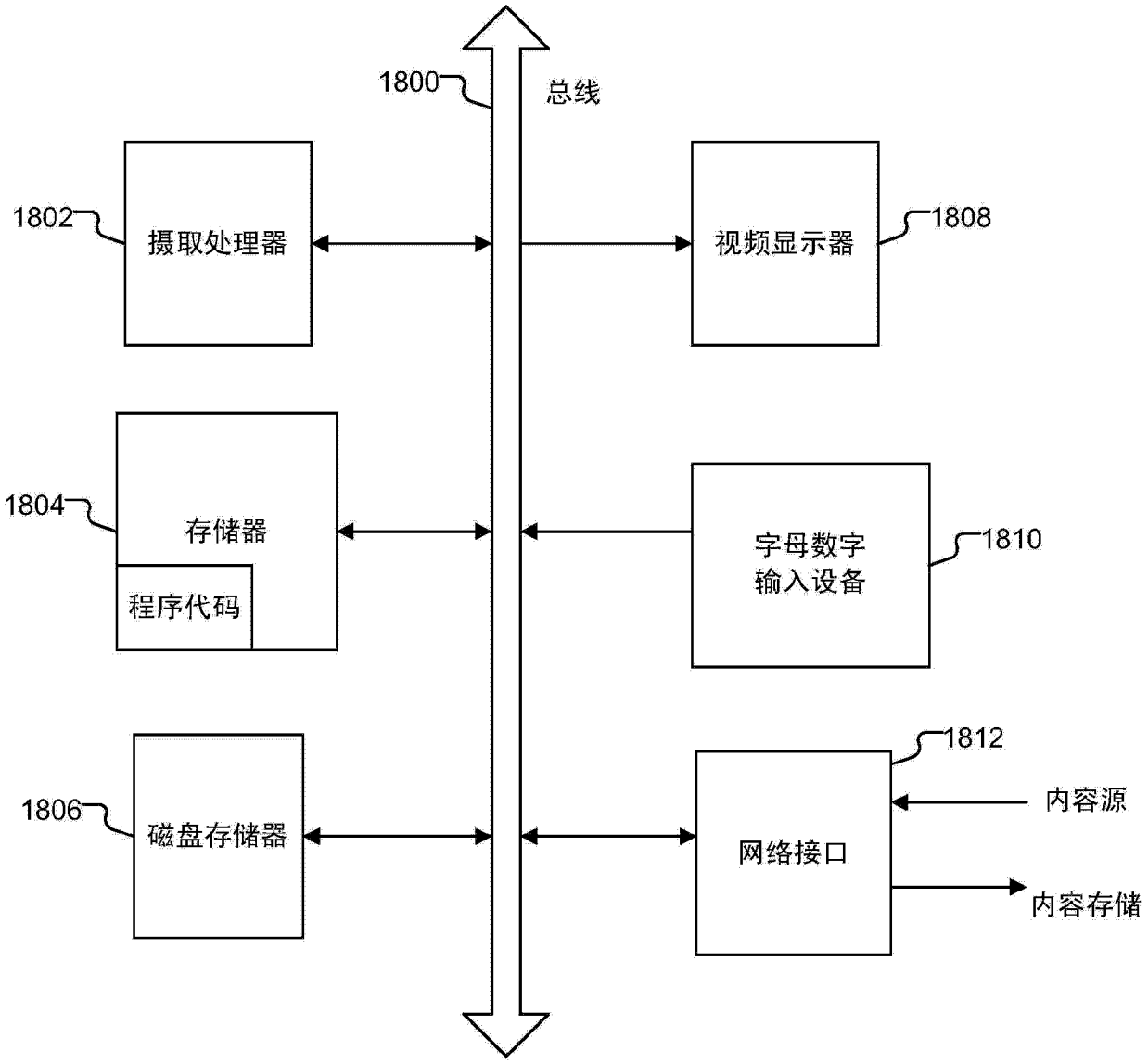

1800
总线
1802
摄取处理器
1808
视频显示器
1804
存储器
程序代码
字母数字
输入设备
1810
1812
1806
磁盘存储器
网络接口
内容源
内容存储

图 18

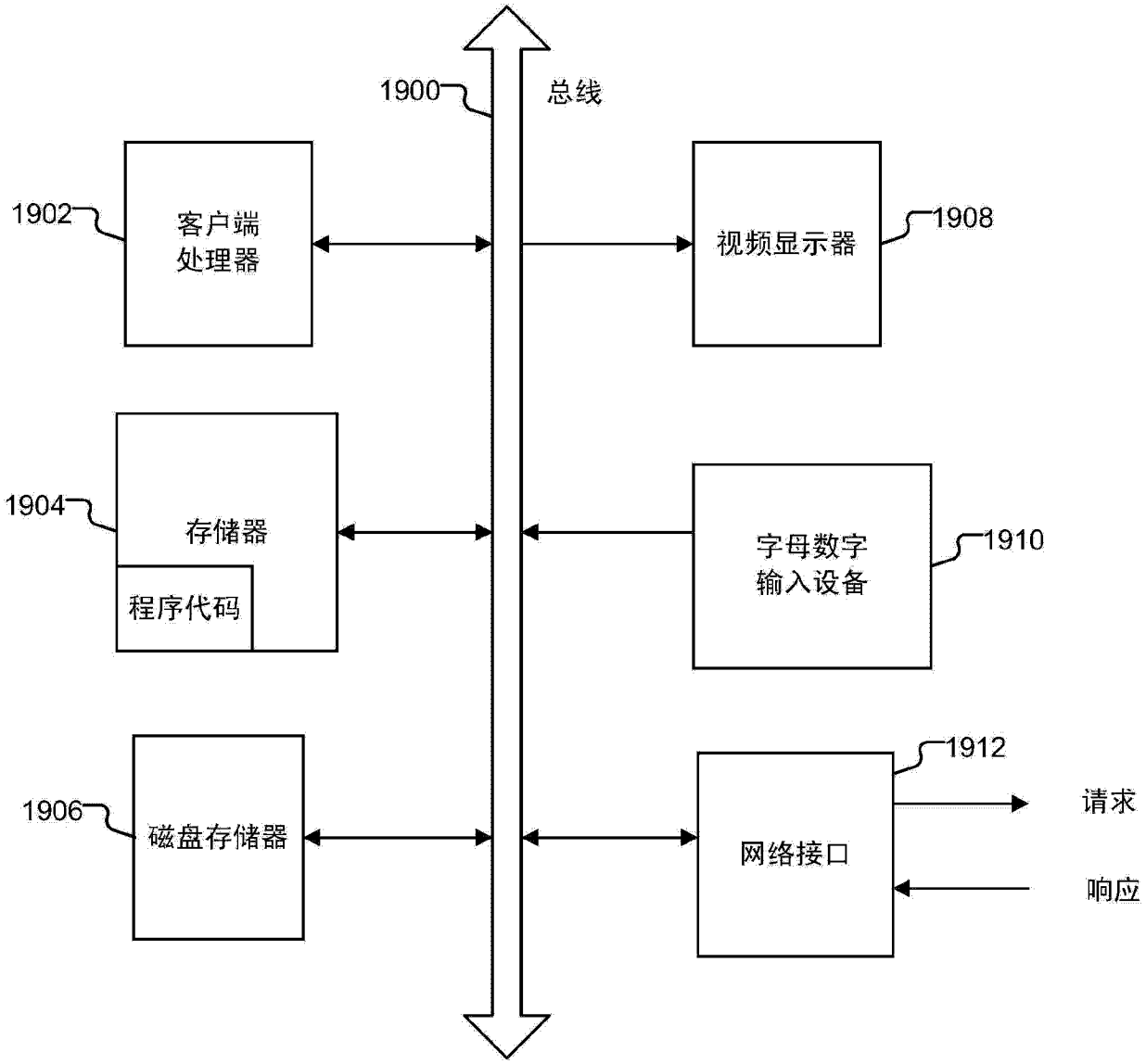

1900
总线
1902
客户端
处理器
1908
视频显示器
1904
存储器
程序代码
1910
字母数字
输入设备
1906
磁盘存储器
1912
网络接口
请求
响应

图 19


0 1 2 3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
对象的数量
(d=2) (8 比特)
对象 1 符号大小(T1)
F1
对象 1 传输长度(cont.) (F1)
对象 2 符号大小(T2)
对象 2 传输长度(F2)
对象 2 传输长度(cont.) (F1)
填充

图 20

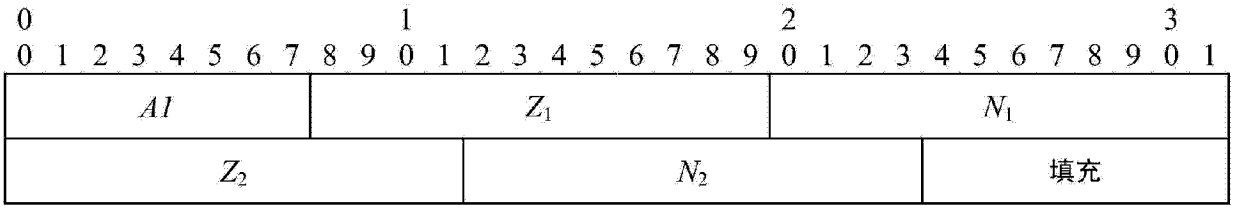


图 21

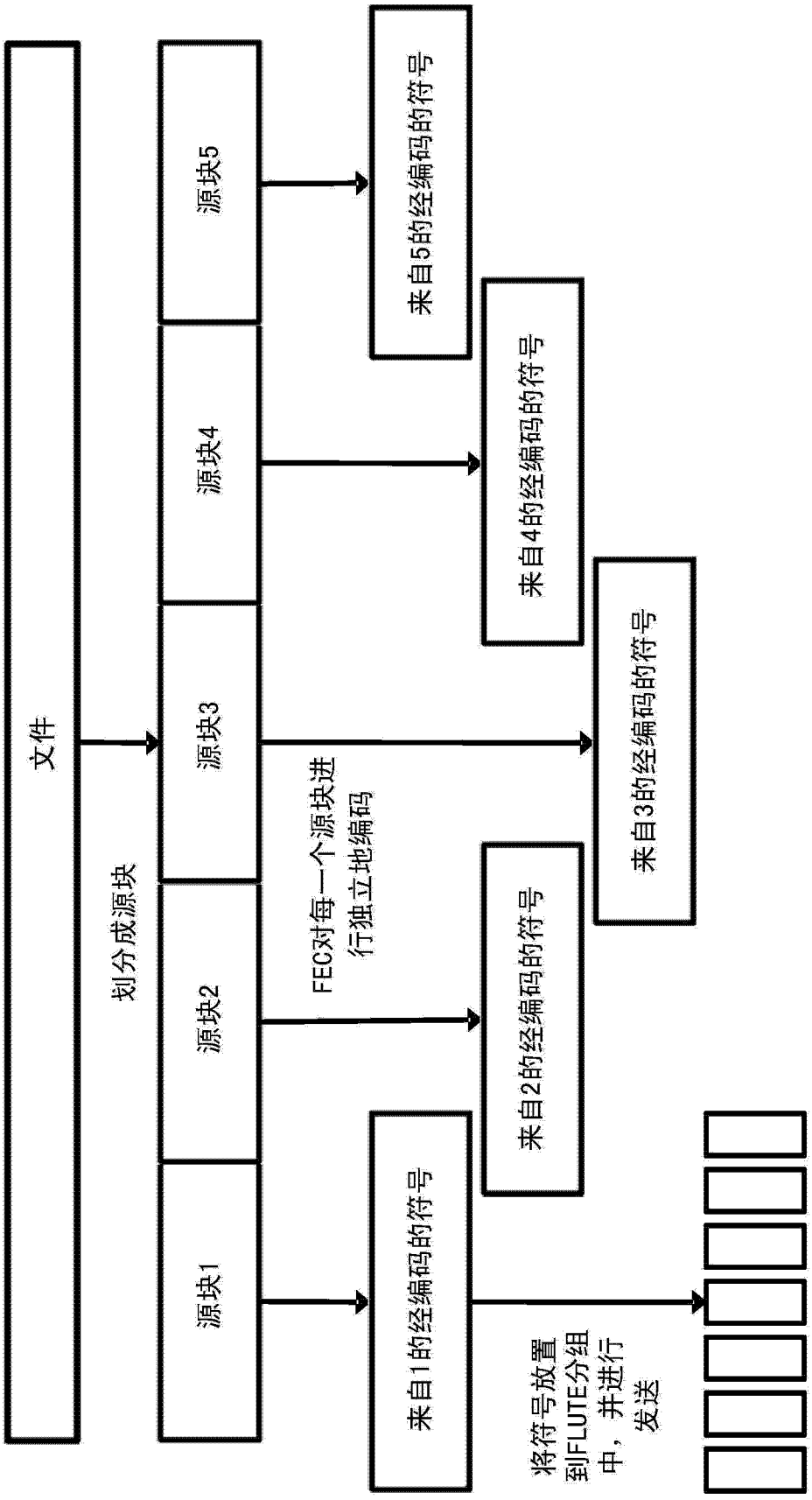

文件
划分成源块
源块1
源块2
源块3
源块4
源块5
FEC对每一个源块进行独立地编码
来自1的经编码的符号
来自2的经编码的符号
来自3的经编码的符号
来自4的经编码的符号
来自5的经编码的符号
将符号放置到FLUTE分组中，并进行发送

图 22

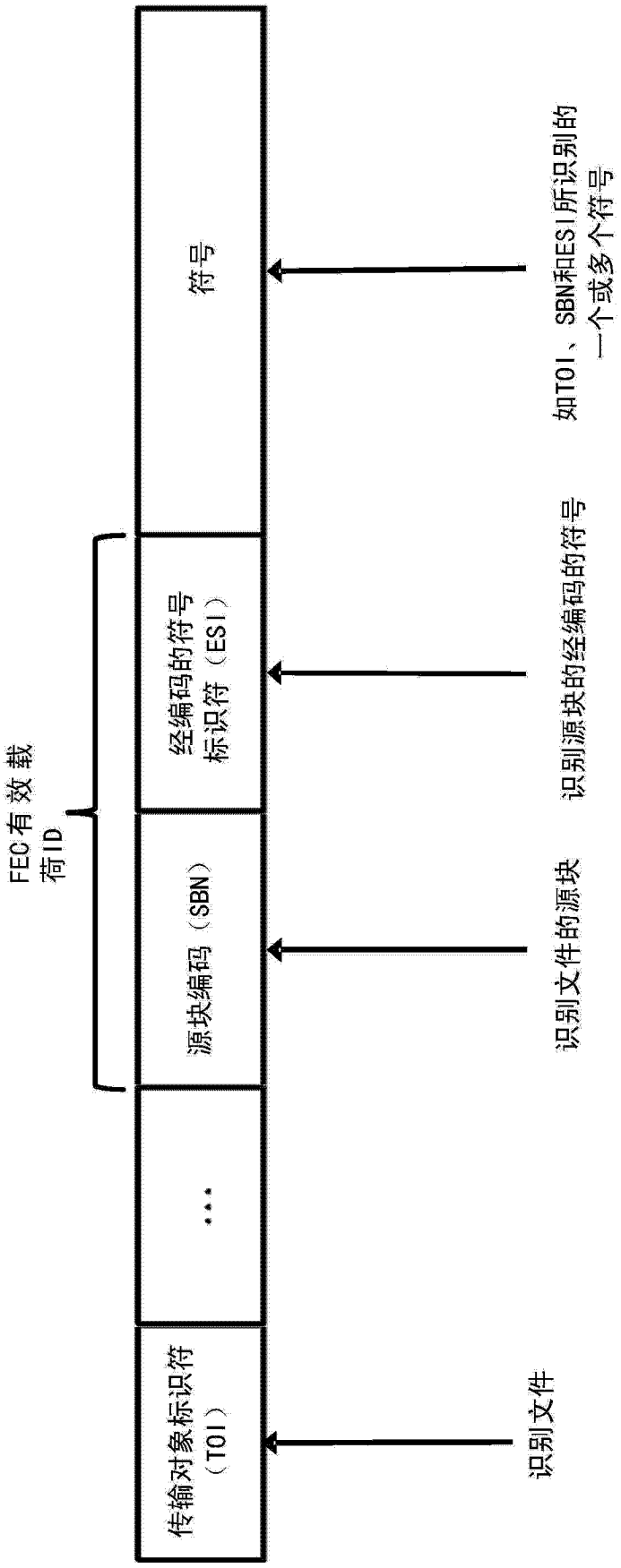

传输对象标识符（TOI）
...
源块编码（SBN）
经编码的符号标识符（ESI）
符号
FEC有效载荷ID
识别文件
识别文件的源块
识别源块的经编码的符号
如TOI、SBN和ESI所识别的一个或多个符号

图 23

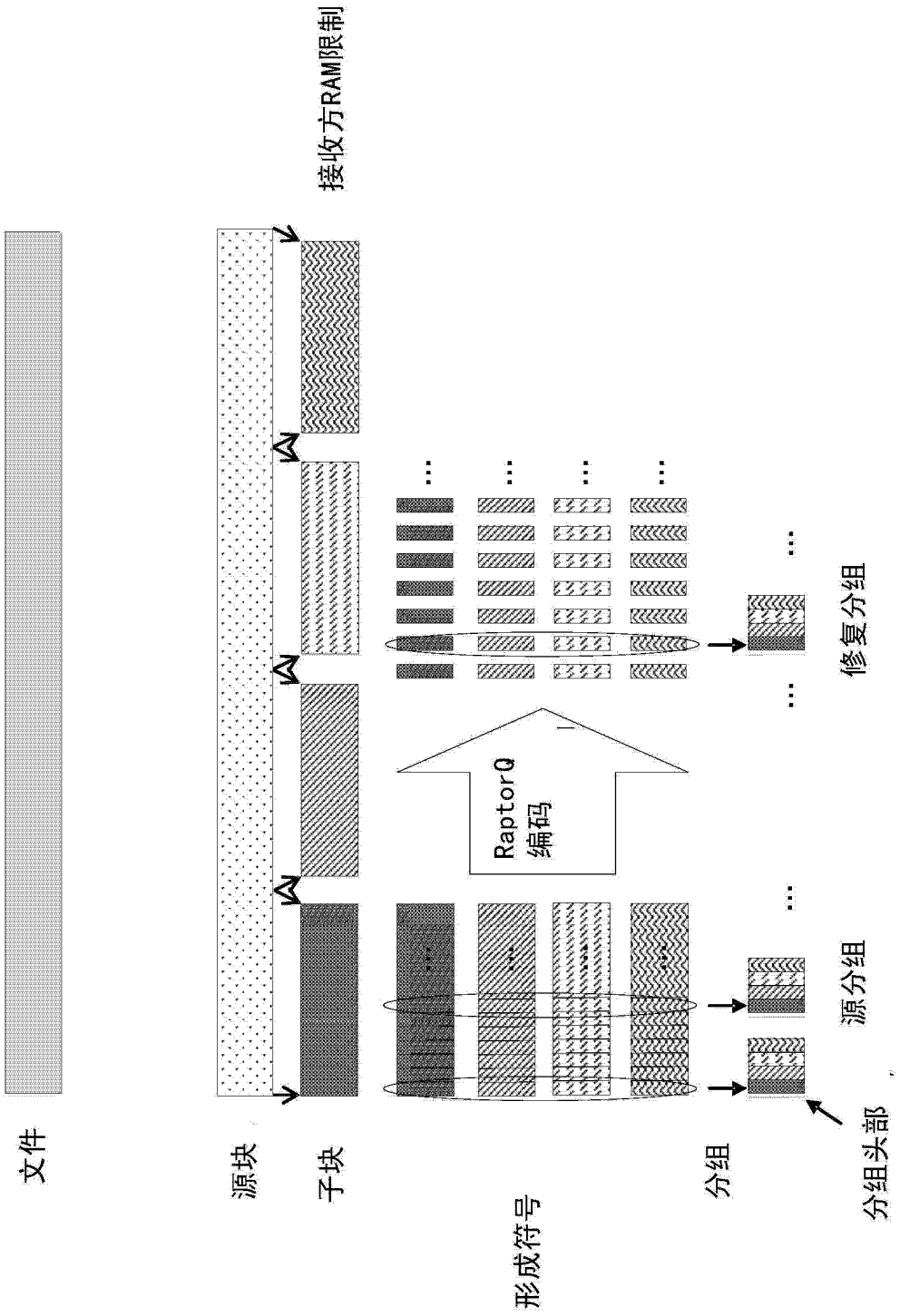


图 24

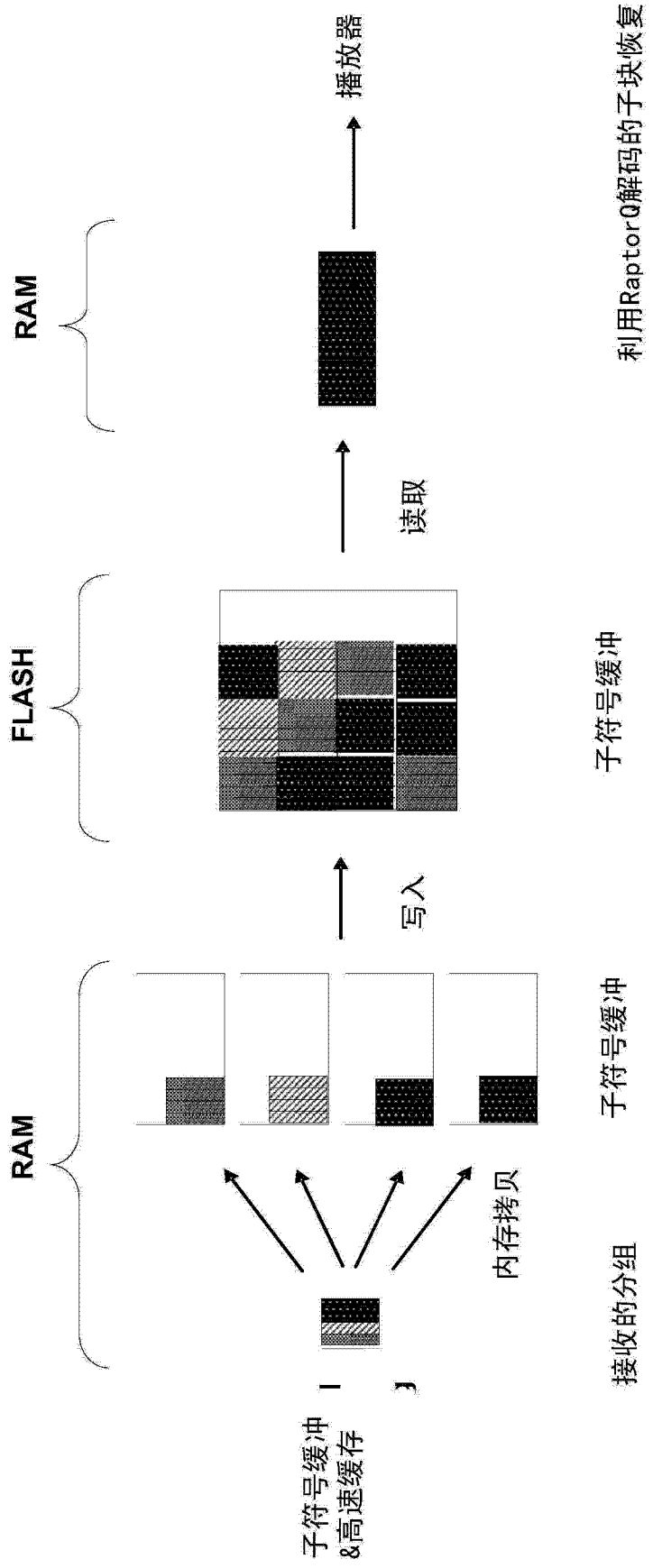

RAM
FLASH
RAM
播放器
读取
写入
内存拷贝
利用RaptorQ解码的子块恢复
子符号缓冲
子符号缓冲
接收的分组
子符号缓冲
&高速缓存

图 25

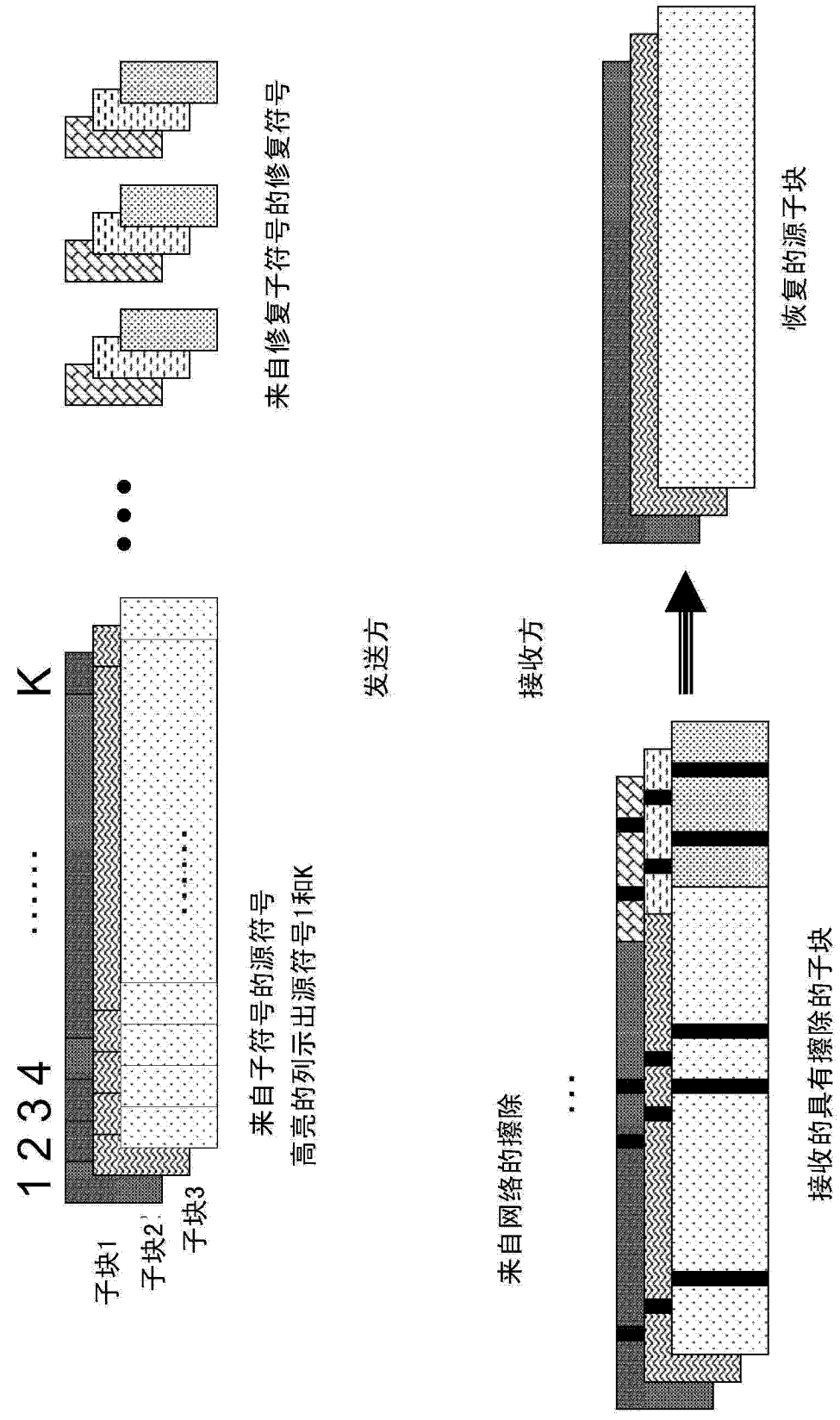

1 2 3 4 ..... K
子块1
子块2'
子块3
来自子符号的源符号
高亮的列示出源符号1和K
来自修复子符号的修复符号
发送方
接收方
来自网络的擦除
接收的具有擦除的子块
恢复的源子块

图 26

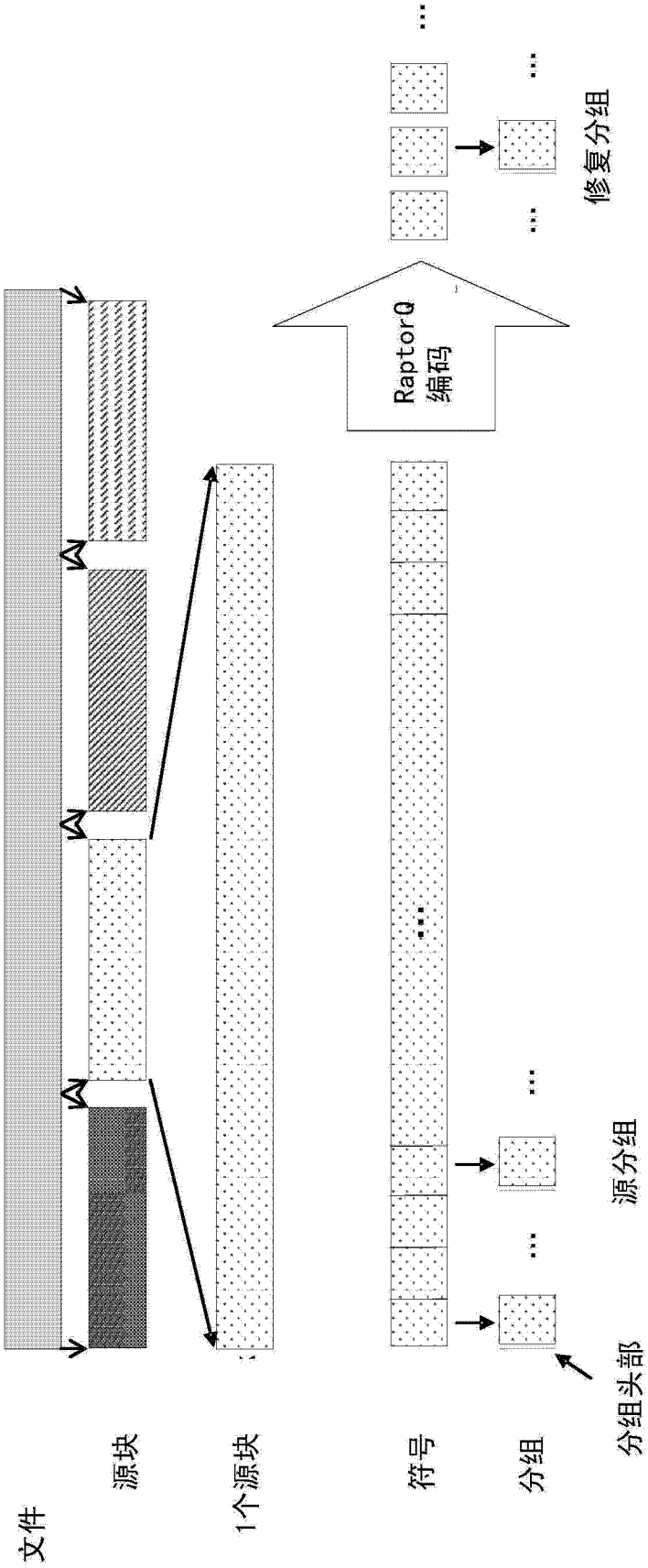


图 27

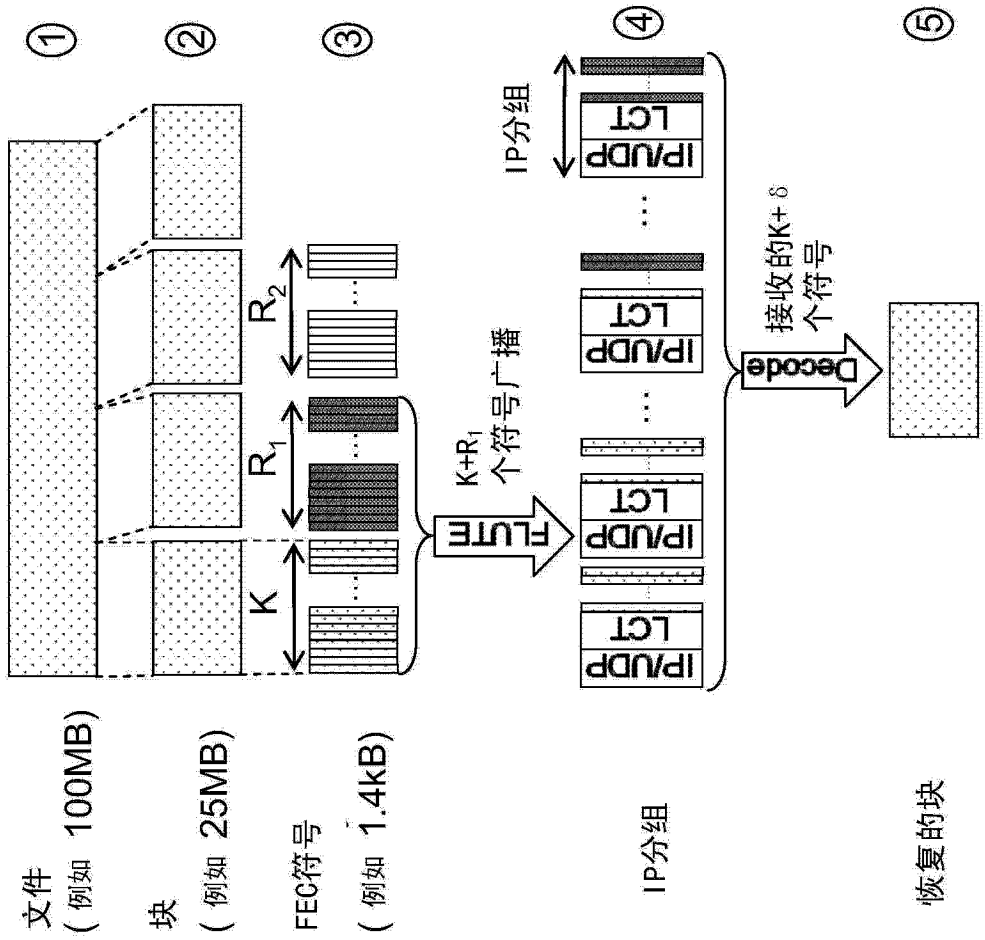

①
②
③
④
⑤
文件
（例如 100MB）
块
（例如 25MB）
FEC符号
（例如 1.4kB）
IP分组
恢复的块
K
R1
R2
FLUTE
K+R1
个符号广播
IP/UDP
LCT
IP分组
Decode
接收的K+δ
个符号

图 28

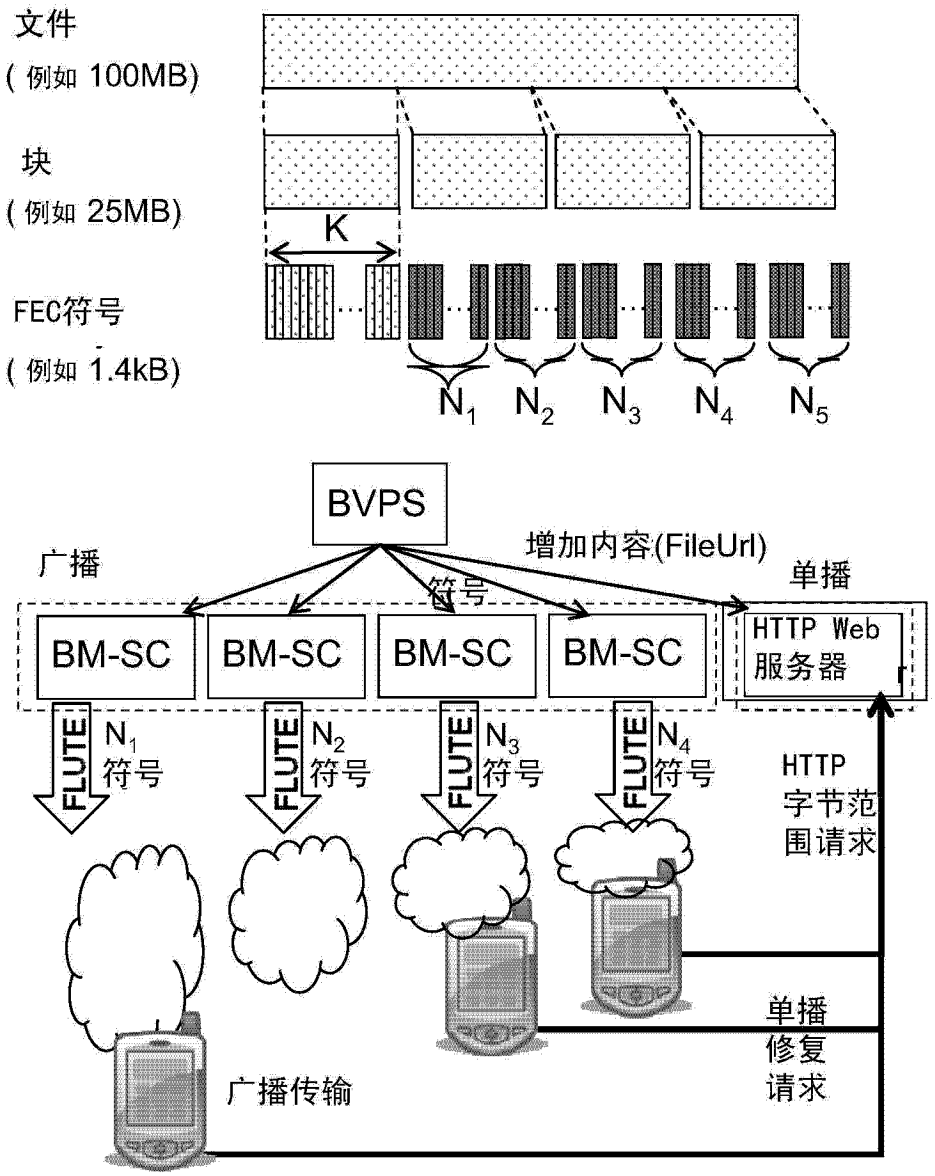

文件
(例如 100MB)
块
(例如 25MB)
K
FEC符号
(例如 1.4kB)
N1
N2
N3
N4
N5
BVPS
广播
增加内容(FileUrl)
单播
符号
BM-SC
BM-SC
BM-SC
BM-SC
HTTP Web
服务器
FLUTE
N1
符号
FLUTE
N2
符号
FLUTE
N3
符号
FLUTE
N4
符号
HTTP
字节范
围请求
单播
修复
请求
广播传输

图 29

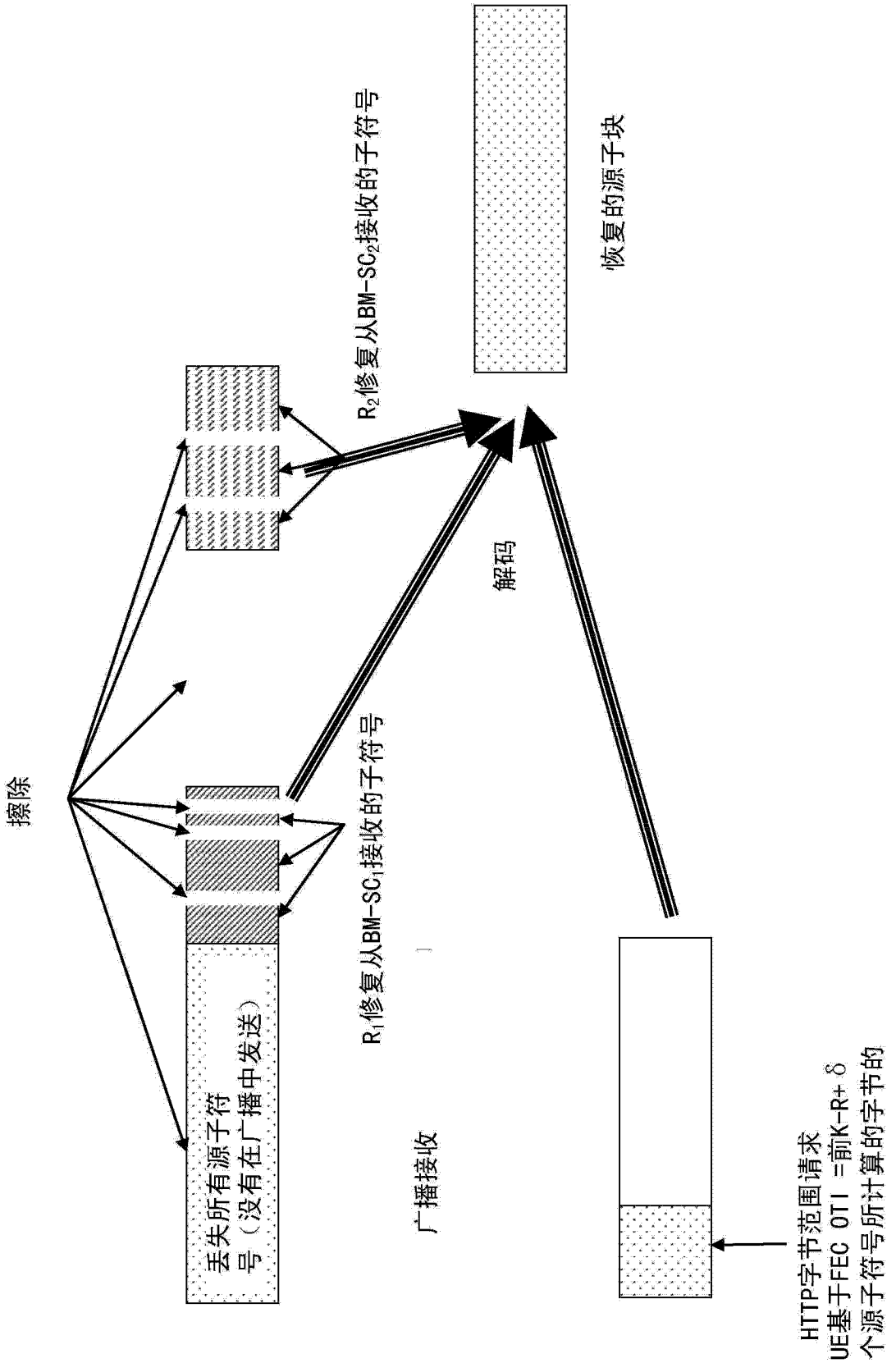

擦除
丢失所有源子符号（没有在广播中发送）
R_1修复从BM-SC_1接收的子符号
R_2修复从BM-SC_2接收的子符号
广播接收
解码
恢复的源子块
HTTP字节范围请求
UE基于FEC OTI =前K-R+ δ
个源子符号所计算的字节的

图 30

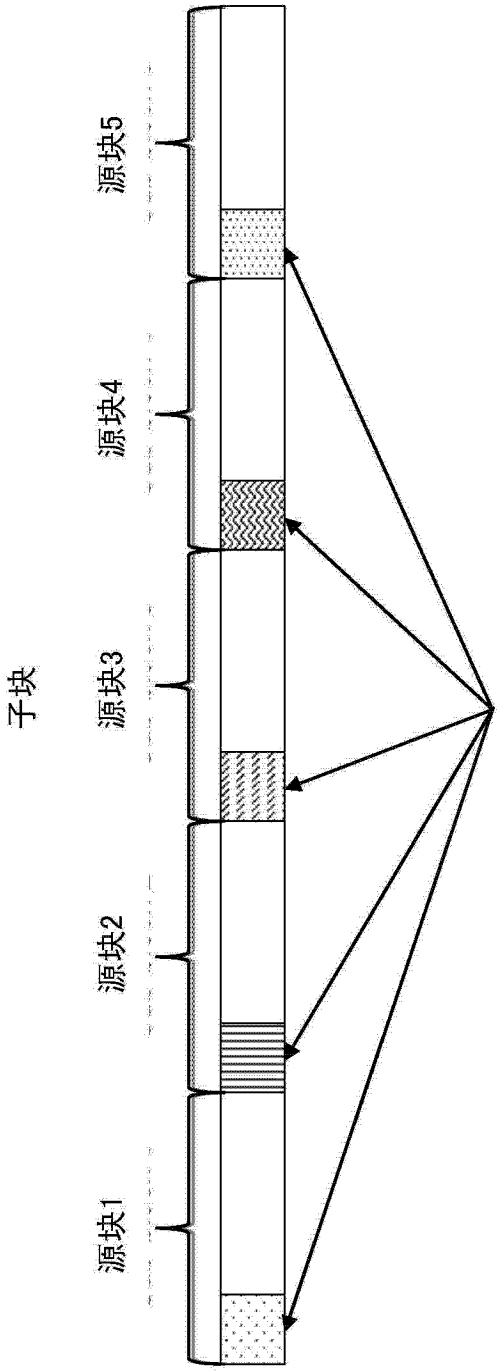


图 31

子块编号 (SuBN)	第一 ESI	起始字节（公式）	起始字节（值）	最后 ESI	结束字节（公式）	结束字节（值）
0	12	12 * 112	1,344	29	1,344 + 18 * 112 - 1	3,359
1	12	7,576 * 12 + 12 * 112	849,856	29	849,856 + 18 * 112 - 1	851,871
2	12	7,576 * 2 * 112 + 12 * 112	1,698,368	29	1,698,368 + 18 * 112 - 1	1,700,383
3	12	7,576 * 3 * 112 + 12 * 112	2,546,880	29	2,546,880 + 18 * 112 - 1	2,548,895
4	12	7,576 * 4 * 112 + 12 * 112	3,395,392	29	3,395,392 + 18 * 112 - 1	3,397,407
5	12	7,576 * 5 * 112 + 12 * 112	4,243,904	29	4,243,904 + 18 * 112 - 1	4,245,919
6	12	7,576 * 6 * 112 + 12 * 108	5,092,368	29	5,092,368 + 18 * 108 - 1	5,094,311
7	12	7,576 * (6 * 112 + 108) + 12 * 108	5,910,576	29	5,910,576 + 18 * 108 - 1	5,912,519
8	12	7,576 * (6 * 112 + 2 * 108) + 12 * 108	6,728,784	29	6,728,784 + 18 * 108 - 1	6,730,727
9	12	7,576 * (6 * 112 + 3 * 108) + 12 * 108	7,546,992	29	7,546,992 + 18 * 108 - 1	7,548,935
10	12	7,576 * (6 * 112 + 4 * 108) + 12 * 108	8,365,200	29	8,365,200 + 18 * 108 - 1	8,367,143
11	12	7,576 * (6 * 112 + 5 * 108) + 12 * 108	9,183,408	29	9,183,408 + 18 * 108 - 1	9,185,351

图 32

子块编号 (SuBN)	第一 ESI	起始字节（公式）	起始字节（值）	最后 ESI	结束字节（公式）	结束字节（值）
0	12	7,576 * 1320 + 12 * 112	10,001,664	29	10,001,664 + 18 * 112 - 1	10,003,679
1	12	7,576 * (1320 + 112) + 12 * 112	10,850,176	29	10,850,176 + 18 * 112 - 1	10,852,191
2	12	7,576 * (1320 + 2 * 112) + 12 * 112	11,698,688	29	11,698,688 + 18 * 112 - 1	11,700,703
3	12	7,576 * (1320 + 3 * 112) + 12 * 112	12,547,200	29	12,547,200 + 18 * 112 - 1	12,549,215
4	12	7,576 * (1320 + 4 * 112) + 12 * 112	13,395,712	29	13,395,712 + 18 * 112 - 1	13,397,727
5	12	7,576 * (1320 + 5 * 112) + 12 * 112	14,244,224	29	14,244,224 + 18 * 112 - 1	14,246,239
6	12	7,576 * (1320 + 6 * 112) + 12 * 108	15,092,688	29	15,092,688 + 18 * 108 - 1	15,094,631
7	12	7,576 * (1320 + 6 * 112 + 108) + 12 * 108	15,910,896	29	15,910,896 + 18 * 108 - 1	15,912,839
8	12	7,576 * (1320 + 6 * 112 + 2 * 108) + 12 * 108	16,729,104	29	16,729,104 + 18 * 108 - 1	16,731,047
9	12	7,576 * (1320 + 6 * 112 + 3 * 108) + 12 * 108	17,547,312	29	17,547,312 + 18 * 108 - 1	17,549,255
10	12	7,576 * (1320 + 6 * 112 + 4 * 108) + 12 * 108	18,365,520	29	18,365,520 + 18 * 108 - 1	18,367,463
11	12	7,576 * (1320 + 6 * 112 + 5 * 108) + 12 * 108	19,183,728	29	19,183,728 + 18 * 108 - 1	19,185,671

图 33

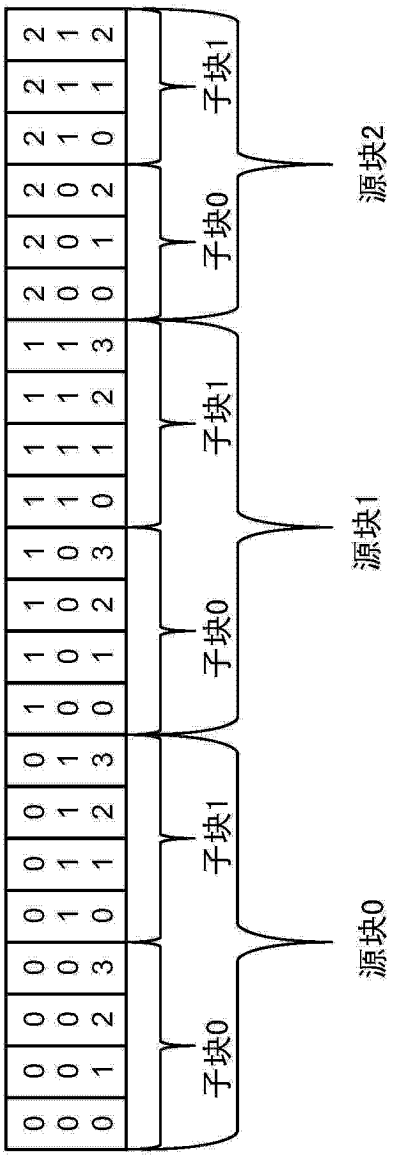


图 34

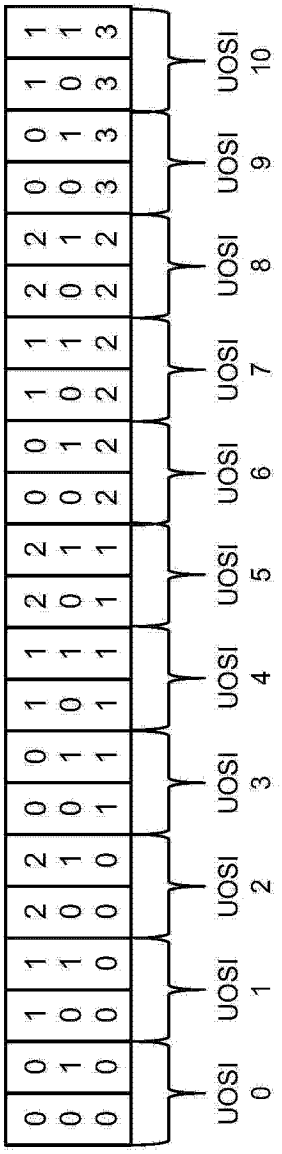

UOSI 0
UOSI 1
UOSI 2
UOSI 3
UOSI 4
UOSI 5
UOSI 6
UOSI 7
UOSI 8
UOSI 9
UOSI 10

图 35

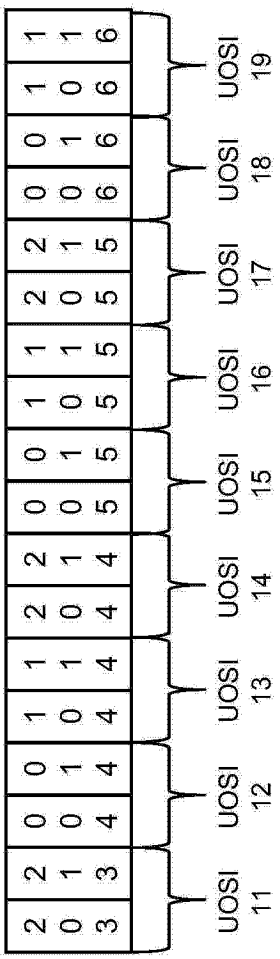

UOSI 11
UOSI 12
UOSI 13
UOSI 14
UOSI 15
UOSI 16
UOSI 17
UOSI 18
UOSI 19

图 36

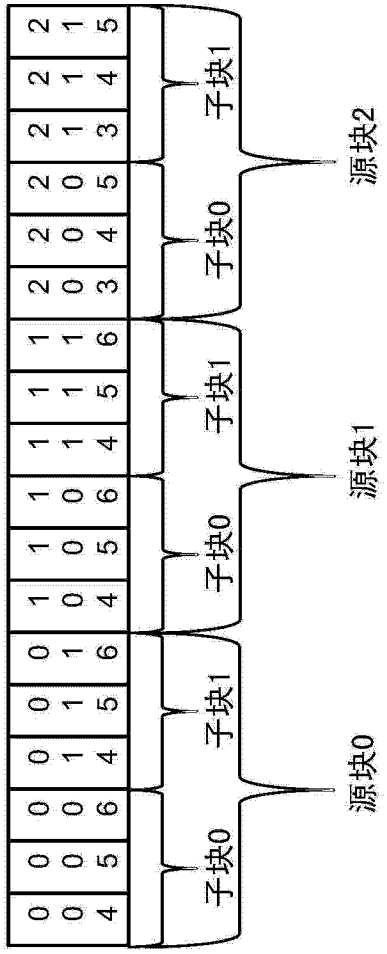

子块0
子块1
子块0
子块1
子块0
子块1
源块0
源块1
源块2

图 37

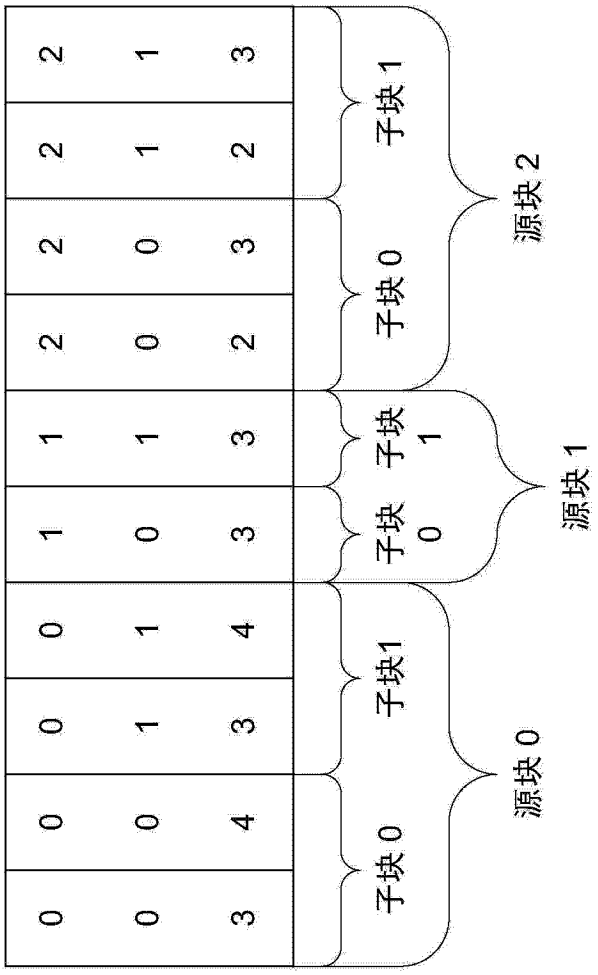


图 38

子块编号 (SuBN)	第一 ESI	起始字节（公式）	起始字节（值）	最后 ESI	结束字节（公式）	结束字节（值）
5	12	(5 * 758 + 12) * 1320	5,018,640	29	5,018,640 + 1320 * 18 - 1	5,042,399
19	27	(12 * 758 + 7 * 757 + 27) * 1320	19,037,040	36	19,037,040 + 1320 * 10 - 1	19,050,239

图 39

子块编号 (SuBN)	第一 ESI	起始字节（公式）	起始字节（值）	最后 ESI	结束字节（公式）	结束字节（值）
5	758	5 * 758 * 1320	5,002,800	775	5,002,800 + 1320 * 18 - 1	5,026,559
19	757	(12 * 758 + 7 * 757) * 1320	19,001,400	766	19,001,400 + 1320 * 10 - 1	19,014,599

图 40

子块编号 (SuBN)	第一 ESI	起始字节（公式）	起始字节（值）	最后 ESI	结束字节（公式）	结束字节（值）
0	7576	0	0	7593	0 + 18 * 112 - 1	2,015
1	7576	7,576 * 112	848,512	7593	848,512 + 18 * 112 - 1	850,527
2	7576	7,576 * 2 * 112	1,697,204	7593	1,697,204 + 18 * 112 - 1	1,699,219
3	7576	7,576 * 3 * 112	2,545,536	7593	2,545,536 + 18 * 112 - 1	2,547,551
4	7576	7,576 * 4 * 112	3,394,048	7593	3,394,048 + 18 * 112 - 1	3,396,063
5	7576	7,576 * 5 * 112	4,242,560	7593	4,242,560 + 18 * 112 - 1	4,244,575
6	7576	7,576 * 6 * 112	5,091,072	7593	5,091,072 + 18 * 108 - 1	5,093,015
7	7576	7,576 * (6 * 112 + 108)	5,909,280	7593	5,909,280 + 18 * 108 - 1	5,911,223
8	7576	7,576 * (6 * 112 + 2 * 108)	6,727,488	7593	6,727,488 + 18 * 108 - 1	6,729,431
9	7576	7,576 * (6 * 112 + 3 * 108)	7,545,696	7593	7,545,696 + 18 * 108 - 1	7,547,639
10	7576	7,576 * (6 * 112 + 4 * 108)	8,363,904	7593	8,363,904 + 18 * 108 - 1	8,365,847
11	7576	7,576 * (6 * 112 + 5 * 108)	9,182,112	7593	9,182,112 + 18 * 108 - 1	9,184,055

图 41