(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 101882976 A
(43)申请公布日 2010.11.10

(21)申请号 201010225574.0

(22)申请日 2010.07.14

(71)申请人 北京邮电大学
地址 100876 北京市海淀区西土城路 10 号

(72)发明人 刘元安 林晓峰 谢刚 曹震
袁东明 黎淑兰 于翠屏 王坤明

(51)Int.Cl.
H04L 1/00(2006.01)
H04L 12/56(2006.01)

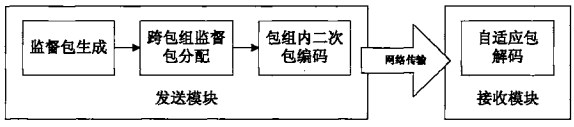
权利要求书 3 页 说明书 7 页 附图 3 页

(54)发明名称

一种数据包可靠传输方法

(57)摘要

本发明提出了一种数据包可靠传输方法，利用包组联合的方式，既保持了传统的包组内独立恢复丢包的特点，又在包组间建立丢包恢复的关联。根据实际的丢包数目，能够自适应的等待后续包组到来，扩展当前包组的丢包恢复能力。因而，本发明相对于传统方案，能够在几乎不增加时延的条件下，明显提升丢包恢复性能，在丢包具有较强相关性的实际网络中具有更好的性能。



CN 101882976 A

1. 一种数据包可靠传输的方法，该方法包括以下步骤：

(1) 利用 RS 码（里德－所罗门码）生成矩阵，构造其一个缺失子矩阵作为包编码的生成矩阵，并以此新生成矩阵产生当前数据包组的所有监督包；

(2) 将当前数据包组生成的所有监督包进行跨包组重新分配；

(3) 将分配后的编码包组进行包组内二次编码；

(4) 接收端采用自适应丢包恢复，根据当前编码包组丢包数目的不同，自适应的等待相应数量的后续关联包组，进行恢复。

2. 根据权利要求 1，采用的包编码的生成矩阵 G，其特征在于：

设 $G_{g\times h}$ 为一个 g 行 h 列的 RS 码的系统生成矩阵，并将其表示成子矩阵形式：

$$G_{g\times h}=\left[I_g \quad W_{g\times(g+r)}\right]=$$

$$\begin{bmatrix} I_k & O_{k\times N_{out}} & W^{in}{}_{k\times N_{in}} & W^{out}{}_{k\times(r-N_{in})} \\ O_{N_{out}\times k} & I_{N_{out}} & Y^{in}{}_{N_{out}\times N_{in}} & Y^{out}{}_{N_{out}\times(r-N_{in})} \end{bmatrix}$$

I 表示单位阵，0 表示零矩阵，N_{in} 为一个包组的包组内监督包数目，N_{out} 为一个包组的跨包组监督包数目，$W_{k\times N_{in}}$ 为 k 行 N_{in} 列的包组内包组内子监督矩阵，$Y^{in}{}_{N_{out}\times N_{in}}$ 为 N_{out} 行 N_{in} 列的包组内子监督矩阵，$W^{out}{}_{k\times(r-N_{in})}$ 为 k 行 $r-N_{in}$ 列的跨包组子监督矩阵，$Y^{out}{}_{N_{out}\times(r-N_{in})}$ 为 N_{out} 行 $r-N_{in}$ 列的跨包组子监督矩阵。$g = k+N_{out}$，$h = k+N_{out}+r$，k 为包组中数据包的个数，r 为这些数据包通过包编码生成监督包的个数。构造一个 k 行 k+r 列的 $G_{g\times h}$ 的缺失子矩阵，并将其作为监督包生成的生成矩阵：

$$G=\left[I_k \quad W^{in}{}_{k\times N_{in}} \quad W^{out}{}_{k\times(r-N_{in})}\right]=\left[I_k \quad W_{k\times r}\right]$$

3. 根据权利要求 1，将当前包组的数据包生成的所有监督包进行跨包组分配，并形成新的编码包组，其特征在于：

设当前包组为第 u 个包组，当前包组所有监督包为 $p_u = [p_{u,1}\ p_{u,2}\cdots p_{u,r}]$。通过这种监督包的分配方法，将当前编码包组分成三部分：数据包部分 d_u，包组内监督部分 $V_{u,in}$ 和跨包组监督部分 $V_{u,out}$。

$d_u = [d_{u,1}\ d_{u,2}\cdots d_{u,k}]$

$$V_{u,in}=\left[p_{u,1} \quad p_{u,2} \quad \cdots \quad p_{u,N_{in}}\right]$$

$$V_{u,out}=\left[S_{u,1} \quad S_{u,2} \quad \cdots \quad S_{u,N_{out}}\right]$$

其中，$d_{u,i}$ 为第 u 个包组中的第 i 个数据包，$S_{u,i}$ 为第 u 个包组中第 i 个跨包组监督包，$S_{u,i}$ 表达式如下：

$$S_{u,i}=\sum_{j=0}^{l-1} p_{u-j,N_{in}+j\cdot N_{out}+i}$$

其中 l 为相关联的包组数。

4. 根据权利要求 1，将编码包组进行包组内二次编码，其特征在于：

将 $S_{u,i}$ ($i = 1,2, \cdots N_{out}$) 与 $p_{u,i}$ ($i = 1,2, \cdots N_{in}$) 建立下面的监督关系：

$$R_{u,j}=p_{u,j}+\sum_{j=1}^{N_{out}} y_{j,i}\cdot S_{u,j}$$

$$\begin{bmatrix} y_{1,1} & y_{1,2} & \cdots & y_{1,N_{in}} \\ y_{2,1} & y_{2,2} & \cdots & y_{2,N_{in}} \\ \vdots & \vdots & \ddots & \vdots \\ y_{N_{out},1} & y_{N_{out},2} & \cdots & y_{N_{out},N_{int}} \end{bmatrix} = Y^{in}{}_{N_{out}\times N_{in}}$$

其中，$i = 1,2, \cdots N_{in}$，$Y^{in}{}_{N_{out}\times N_{in}}$为 $G_{g\times h}$ 中的子矩阵。然后用 $R_{u,i}$ 替换包组中的 $p_{u,i}$，组成新的 $V_{u,in}$，并形成最终的编码包组，且满足下面的关系。

$D_u \cdot G_{in} = R_u$

其中，$D_u = \begin{bmatrix} d_{u,1} & d_{u,2} & \cdots & d_{u,k} & S_{u,1} & S_{u,2} & \cdots & S_{u,N_{out}} \end{bmatrix}$，

$$R_u = \begin{bmatrix} d_{u,1} & d_{u,2} & \cdots & d_{u,k} & S_{u,1} & S_{u,2} & \cdots & S_{u,N_{out}} & R_{u,1} & R_{u,2} & \cdots & R_{u,N_{in}} \end{bmatrix}$$

$$G_{in} = \begin{bmatrix} I_k & O_{k\times N_{out}} & W^{in}{}_{k\times N_{in}} \\ O_{N_{out}\times k} & I_{N_{out}} & Y^{in}{}_{N_{out}\times N_{in}} \end{bmatrix}$$

5. 根据权利要求 1，采用自适应的包解码方法，其特征在于：

设 N_l 为当前编码包组的丢包数，假设与此包组相关联的前 l-1 个包组的数据包是完整的，即没有丢失或已经顺利恢复丢包，且按照上述的包编码方法，每获得一个监督包，就可以多恢复出一个丢包。

(1) 若 $N_l \leqslant N_{in}$，则利用解线性方程组方法，不需要任何其他编码包组的协助，就能够进行恢复丢包。

(2) 若 $N_{in} < N_l \leqslant N_{in}+N_{out}$，假设之前的 l-1 包组是完整的，即包组内没有丢包或者已经恢复所有数据包，首先计算出 $p_{a,b}$($a \in \{u-l-1, u-l, \cdots, u-1\}$，$b \in \{1,2, \cdots, m\}$)，再还原出隐藏在 $S_{u,i}$($i = 1,2, \cdots, N_{out}$) 中的 N_{out} 个第 u 个包组的监督包：

$$p_{u,i} = S_{u,I} - \sum_{j=1}^{l-1} p_{u-j,N_{in}+j\cdot N_{out}+I}$$

其中，$i \in \{N_{in}+1, N_{in}+2, \cdots, N_{in}+N_{out}\}$，$I = i-N_{in}$，为第 u 个包组的第 i 个监督包累加在跨包组监督部分的位置下标。将还原出的监督包和包组内监督部分 $V_{u,out}$ 联合起来，便得到了 $N_{in}+N_{out}$ 个第 u 个包组的监督包，这相当于将当前包组生成矩阵 G_{in} 扩展了 N_{out} 列：

$$G_{in}' = \begin{bmatrix} I_k & O_{k\times N_{out}} & W^{in}{}_{k\times N_{in}} & W^{out,1}{}_{k\times N_{out}} \\ O_{N_{out}\times k} & I_{N_{out}} & Y^{in}{}_{N_{out}\times N_{in}} & O_{N_{out}\times N_{out}} \end{bmatrix}$$

其中 $W^{out,1}{}_{k\times N_{out}}$ 是 $G_{g\times h}$ 中 $W^{out}{}_{k\times(m-N_{in})}$ 的前 N_{out} 列组成的子矩阵。任取 G'_{in} 的 k 列，对这 k 维的列向量组，截取其前 k 行得到的方阵一定是 $G_{g\times h}$ 的一个缺失子方阵。此方阵的列向量是线性无关的，于是这任取 G'_{in} 的 k 列也是线性无关的，可以恢复当前包组中所有的数据包。

(3) 如果 $N_l > N_{in}+N_{out}$，现有的恢复能力不能满足需求，则等待第 u+1 个包组的到来。如果第 u-l+2 个包组到第 u+1 个包组这 l 个包组中，除第 u 个包组，都没有丢包，或者通过包组内监督部分恢复出包组内全部数据包，进而还原出压缩在 $S_{u+1,i}$($i = 1,2, \cdots, N_{out}$) 中的 N_{out} 个第 u 个包组的监督包：

$$p_{u,i} = S_{u+1,I} - \sum_{j=0, j\neq 1}^{l-1} p_{u-j, N_{in}+j\cdot N_{out}+I}$$

其中，i ∈ {N_{in}+N_{out}+1，N_{in}+N_{out}+2，…，N_{in}+2 • N_{out}}，I = i-N_{in}-N_{out}。这样便总共得到了第 u 个包组的 N_{in}+2 • N_{out} 个监督包，与步骤二中，可以得到新的增加了 N_{out} 列的当前包组生成矩阵。于是，如果 $N_l \leq N_{in}$+2 • N_{out}，则可以恢复第 u 个包组中的所有数据包。通过第 u 个包组后面的 l-1 个包组的到来，至多可还原出第 u 个包组的 (l-1) • N_{out} 个额外的监督包，这样至多可以恢复出第 u 个包组中的 N_{in}+1 • N_{out} 个丢包。

(4)、若 $N_l > N_{in}$+1 • N_{out}，则当前包组丢失情况超出恢复上限，放弃恢复当前包组的丢包。

6. 根据权利要求 5，在自适应包解码过程中，当 $N_l > N_{in}$+N_{out} 时，可以设置包解码的最大等待包组数 M，即为了恢复第 u 个包组的丢包，已经等待了 M 个包组，如果仍然没有得到足够的第 u 个包组的监督包，则不再继续等待后续的包组，放弃对第 u 个包组的恢复。

一种数据包可靠传输方法

技术领域

[0001] 本发明涉及网络数据传输领域，是一种对数据包丢失进行恢复的有效方法。尤其适用于实时通信和单向通信中的数据包丢失恢复。

背景技术

[0002] 随着计算机网络及其技术的迅速发展，网络环境在不断改善，然而数据包丢失问题仍然是影响网络通信质量的最大因素之一。对于数据包丢失，主要有两种解决方法：一种是 ARQ 技术：通过反馈信道，将丢失的数据段信息反馈给发送方，发送方重新发送丢失的数据段；另一种是 FEC 技术：通过对数据包分组添加冗余包，使得发送的数据包组具有一定的丢包恢复能力。对传输时延敏感的业务或只有单向信道的应用场景中，采用 FEC 技术则是一种有效且实用的选择。

[0003] 常见的 FEC 技术通常是基于奇偶校验码、RS 码（里德－所罗门码）等线性分组码的包编码方法。基于奇偶校验码的包编码，利用奇偶校验码的编码方法，产生一个冗余的监督包，并添加在数据包后，构成一组编码包组，这种方法可以恢复包组内任意丢失的一个编码包；类似的，基于 RS 码的包编码，利用 RS 码的编码方法，产生 m 个冗余的监督包，并添加在数据包后，构成一组编码包组，这种方法可以恢复包组内任意丢失的 m 个编码包。

[0004] 经过研究和统计，Internet 的丢包模式：大部分丢包是一种随机丢包，少数丢包表现为突发性的较长连续丢包，即少数丢包表现出较强相关性 [6]。传统的基于线性分组码的丢包恢复方案，虽然在处理分布均匀的随机丢包时可获得较好的效果，但是如果网络中出现了突发的较多个密集丢包，无差别的对每个包组添加更多的冗余包，将极大增加网络的负担。且每个包组内孤立的进行丢包恢复，使得部分包组的恢复能力过剩，部分包组的恢复能力不足，无法动态的合理调度恢复资源。

发明内容

[0005] 本发明提出了一种数据包可靠传输的方法，打破了线性分组码的包组间孤立地恢复丢包的弊端，通过对 RS 码生成矩阵的重新设计，在包组间引入动态关联，通过后续包组的到来选择性的扩展当前包组的丢包恢复能力。

[0006] 一种数据包可靠传输恢复方法，其特征在于，包含以下步骤：

[0007] (1) 从 RS 码生成矩阵中，构造其一个缺失子矩阵作为包编码的生成矩阵，并以此新生成矩阵产生当前数据包组的所有监督包；

[0008] (2) 将当前数据包组生成的所有监督包进行跨包组重新分配；

[0009] (3) 将分配后的编码包组进行包组内二次编码；

[0010] (4) 在接收端采用自适应丢包恢复，根据当前编码包组丢包多少的不同，自适应的等待相应数量的后续关联包组，进行恢复。

[0011] 基于上述技术方案，采用的包编码的生成矩阵 G，其特征在于：

[0012] 设 $G_{g\times h}$ 为一个 g 行 h 列的 RS 码的系统生成矩阵，并将其表示成子矩阵形式：

[0013] $$G_{g\times h}=\left[I_g \quad W_{g\times(g+r)}\right]=$$

[0014] $$\begin{bmatrix} I_k & O_{k\times N_{out}} & W^{in}{}_{k\times N_{in}} & W^{out}{}_{k\times(r-N_{in})} \\ O_{N_{out}\times k} & I_{N_{out}} & Y^{in}{}_{N_{out}\times N_{in}} & Y^{out}{}_{N_{out}\times(r-N_{in})} \end{bmatrix}$$

[0015] I 表示单位阵，0 表示零矩阵，N_{in} 为一个包组的包组内监督包数目，N_{out} 为一个包组的跨包组监督包数目，$W_{k\times N_{in}}$ 为 k 行 N_{in} 列的包组内包组内子监督矩阵，$Y^{in}{}_{N_{out}\times N_{in}}$ 为 N_{out} 行 N_{in} 列的包组内子监督矩阵，$W^{out}{}_{k\times(r-N_{in})}$ 为 k 行 $r-N_{in}$ 列的跨包组子监督矩阵，$Y^{out}{}_{N_{out}\times(r-N_{in})}$ 为 N_{out} 行 $r-N_{in}$ 列的跨包组子监督矩阵。$g = k+N_{out}$，$h = k+N_{out}+r$，k 为包组中数据包的个数，r 为这些数据包通过包编码生成监督包的个数。构造一个 k 行 k+r 列的 $G_{g\times h}$ 的缺失子矩阵，并将其作为监督包生成的生成矩阵：

[0016] $$G=\left[I_k \quad W^{in}{}_{k\times N_{in}} \quad W^{out}{}_{k\times(r-N_{in})}\right]=\left[I_k \quad W_{k\times r}\right]$$

[0017] 基于上述技术方案，将当前包组的数据包生成的所有监督包进行跨包组分配，并形成新的编码包组，其特征在于：

[0018] 设当前包组为第 u 个包组，当前包组所有监督包为 $p_u = [p_{u,1}\ p_{u,2}\cdots p_{u,r}]$。通过这种监督包的分配方法，将当前编码包组分成三部分：数据包部分 d_u，包组内监督部分 $V_{u,in}$ 和跨包组监督部分 $V_{u,out}$。

[0019] $d_u = [d_{u,1}\ d_{u,2}\cdots d_{u,k}]$

[0020] $$V_{u,in}=\left[p_{u,1} \quad p_{u,2} \quad \cdots \quad p_{u,N_{in}}\right]$$

[0021] $$V_{u,out}=\left[S_{u,1} \quad S_{u,2} \quad \cdots \quad S_{u,N_{out}}\right]$$

[0022] 其中，$d_{u,i}$ 为第 u 个包组中的第 i 个数据包，$S_{u,i}$ 为第 u 个包组中第 i 个跨包组监督包，$S_{u,i}$ 表达式如下：

[0023] $$S_{u,i}=\sum_{j=0}^{l-1} p_{u-j,N_{in}+j\cdot N_{out}+i}$$

[0024] 其中 l 为相关联的包组数。

[0025] 基于上述技术方案，将编码包组进行包组内二次编码，其特征在于：将 $S_{u,i}$（$i = 1, 2, \cdots N_{out}$）与 $p_{u,i}$（$i = 1, 2, \cdots N_{in}$）建立下面的监督关系：

[0026] $$R_{u,i}=p_{u,i}+\sum_{j=1}^{N_{out}} y_{j,i}\cdot S_{u,j}$$

[0027] $$\begin{bmatrix} y_{1,1} & y_{1,2} & \cdots & y_{1,N_{in}} \\ y_{2,1} & y_{2,2} & \cdots & y_{2,N_{in}} \\ \vdots & \vdots & \ddots & \vdots \\ y_{N_{out},1} & y_{N_{out},2} & \cdots & y_{N_{out},N_{\mathrm{int}}} \end{bmatrix}=Y^{in}{}_{N_{out}\times N_{in}}$$

[0028] 其中，$i = 1, 2, \cdots N_{in}$，$Y^{in}{}_{N_{out}\times N_{in}}$ 为 $G_{g\times h}$ 中的子矩阵。然后用 $R_{u,i}$ 替换包组中的 $p_{u,i}$，组成新的 $V_{u,in}$，并形成最终的编码包组，且满足下面的方程关系。

[0029] $D_u \cdot G_{in} = R_u$

[0030] 其中，$D_u=\left[d_{u,1}\quad d_{u,2}\quad\cdots\quad d_{u,k}\quad S_{u,1}\quad S_{u,2}\quad\cdots\quad S_{u,N_{out}}\right]$，

[0031] $R_u=\left[d_{u,1}\quad d_{u,2}\quad\cdots\quad d_{u,k}\quad S_{u,1}\quad S_{u,2}\quad\cdots\quad S_{u,N_{out}}\ R_{u,1}\quad R_{u,2}\quad\cdots\quad R_{u,N_{in}}\right]$

[0032] $$G_{in}=\begin{bmatrix} I_k & O_{k\times N_{out}} & {W^{in}}_{k\times N_{in}} \\ O_{N_{out}\times k} & I_{N_{out}} & {Y^{in}}_{N_{out}\times N_{in}} \end{bmatrix}$$

[0033] 基于上述技术方案，采用自适应的包解码方法，其特征在于：

[0034] 设 N_l 为当前编码包组的丢包数，假设与此包组相关联的前 l-1 个包组的数据包是完整的，即没有丢失或已经顺利恢复丢包，且按照上述的包编码方法，每获得一个监督包，就可以多恢复出一个丢包。

[0035] (1) 若 $N_l \leqslant N_{in}$，则利用解线性方程组方法，不需要任何其他编码包组的协助，就能够进行恢复丢包。

[0036] (2) 若 $N_{in} < N_l \leqslant N_{in}+N_{out}$，假设之前的 l-1 包组是完整的，即包组内没有丢包或者已经恢复所有数据包，首先计算出 $p_{a,b}$($a \in \{u-l-1, u-l, \cdots, u-1\}$，$b \in \{1,2,\cdots,m\}$)，再还原出隐藏在 $S_{u,i}$($i = 1,2,\cdots,N_{out}$) 中的 N_{out} 个第 u 个包组的监督包：

[0037] $$p_{u,i}=S_{u,l}-\sum_{j=1}^{l-1}p_{u-j,N_{in}+j\cdot N_{out}+l}$$

[0038] 其中，$i \in \{N_{in}+1, N_{in}+2, \cdots, N_{in}+N_{out}\}$，$I = i-N_{in}$，为第 u 个包组的第 i 个监督包累加在跨包组监督部分的位置下标。将还原出的监督包和包组内监督部分 $V_{u,out}$ 联合起来，便得到了 $N_{in}+N_{out}$ 个第 u 个包组的监督包，相当于将当前包组生成矩阵 G_{in} 扩展了 N_{out} 列：

[0039] $$G_{in}'=\begin{bmatrix} I_k & O_{k\times N_{out}} & {W^{in}}_{k\times N_{in}} & {W^{out,1}}_{k\times N_{out}} \\ O_{N_{out}\times k} & I_{N_{out}} & {Y^{in}}_{N_{out}\times N_{in}} & O_{N_{out}\times N_{out}} \end{bmatrix}$$

[0040] 其中${W^{out,1}}_{k\times N_{out}}$是 $G_{g\times h}$ 中${W^{out}}_{k\times(m-N_{in})}$的前 N_{out} 列组成的子矩阵。任取 G'_{in} 的 k 列，对这 k 维的列向量组，截取其前 k 行得到的方阵一定是 $G_{g\times h}$ 的一个缺失子方阵。此方阵的列向量是线性无关的，于是这任取 G'_{in} 的 k 列也是线性无关的，可以恢复当前包组中所有的数据包。

[0041] (3) 如果 $N_l > N_{in}+N_{out}$，现有的恢复能力不能满足需求，则等待第 u+1 个包组的到来。如果第 u-l+2 个包组到第 u+1 个包组这 l 个包组中，除第 u 个包组，都没有丢包，或者通过包组内监督部分恢复出包组内全部数据包，进而还原出压缩在 $S_{u+1,i}$($i = 1,2,\cdots,N_{out}$) 中的 N_{out} 个第 u 个包组的监督包：

[0042] $$p_{u,i}=S_{u+1,l}-\sum_{j=0,j\neq 1}^{l-1}p_{u-j,N_{in}+j\cdot N_{out}+l}$$

[0043] 其中，$i \in \{N_{in}+N_{out}+1, N_{in}+N_{out}+2, \cdots, N_{in}+2\cdot N_{out}\}$，$I = i-N_{in}-N_{out}$。这样便总共得到了第 u 个包组的 $N_{in}+2\cdot N_{out}$ 个监督包，与步骤二中，可以得到新的增加了 N_{out} 列的当前包组生成矩阵。于是，如果 $N_l \leqslant N_{in}+2\cdot N_{out}$，则可以恢复第 u 个包组中的所有数据包。通过第 u 个包组后面的 l-1 个包组的到来，至多可还原出第 u 个包组的 $(l-1)\cdot N_{out}$ 个额外的监督包，这样至多可以恢复出第 u 个包组中的 $N_{in}+1\cdot N_{out}$ 个丢包。

[0044] (4)、若 $N_l > N_{in}+1\cdot N_{out}$，则当前包组丢失情况超出恢复上限，放弃恢复当前包组的丢包。

[0045] 基于上述方案，在自适应包解码过程中，当 $N_1 > N_{in}+N_{out}$ 时，可以设置包解码的最大等待包组数M，即为了恢复第u个包组的丢包，已经等待了M个包组，如果仍然没有得到足够的第u个包组的监督包，则不再继续等待后续的包组，放弃对第u个包组的恢复。

[0046] 本发明一种数据包可靠传输方法，具有以下的优点：

[0047] 1、打破了线性分组码的包组间孤立地恢复丢包的弊端，通过对生成矩阵的重新设计，在包组间引入了丢包恢复的动态关联，通过后续包组的到来选择性的扩展当前包组的丢包恢复能力。

[0048] 2、相对于传统的基于RS码的方案，在相同冗余度下，明显提升恢复性能；在相当的恢复能力下，提高了传输效率。

[0049] 3、能够自适应的针对当前包组丢包情况调整恢复能力和解码时延，从而提升系统的整体性能。

附图说明

[0050] 图1为本发明的系统框图；

[0051] 图2为本发明的第u个编码包组的结构图；

[0052] 图3为本发明的第u个包组的监督包分配方式图；

[0053] 图4为本发明的第u个包组的监督包部分（包组内监督部分和跨包组监督部分）的构成图；

[0054] 图5为本发明的接收端自适应包解码的流程图；

[0055] 图6为本发明的相同编码效率下，本发明和其他方法的丢包恢复性能比较图；

[0056] 图7为本发明的相当的丢包恢复性能下，本发明和其他方法的编码效率比较图。

具体实施方式

[0057] 下面结合附图对本发明做详细说明。

[0058] 图1示例了本发明的系统框架，该系统包含了发送端和接收端两部分。发送端进一步包括监督包生成模块，跨包组监督包分配模块和包组内二次编码模块。接收端进一步包括自适应包解码模块。

[0059] 在详细说明本发明前，首先做一些声明和定义。

[0060] 设将k个数据包划为一组，并生成n-k＝r个监督包。将数据包和监督包都称为编码包，从而形成n个包组成的编码包组。每个编码包都有s个比特，且有一个包序列号与之对应，用来发现哪些编码包在传输中丢失了。由于本发明采用RS码作为编码方案，所以文中所述的运算都是指GF域（有限域）的运算。设当前编码包组为第u包组。$d_{u,i}$ 为第u个包组的第i个数据包，i＝1,2,...k，$d_{u,i}$ 又可以表示为s维的列向量 $[d_{u,i,1}\ d_{u,i,2}\cdots d_{u,i,s}]$，其中 $d_{u,i,1}$ 为第u个包组的第i个数据包的第1个比特，其余依次类推。$p_{u,i}$ 表示第u个包组的第i个监督包，i＝1,2,...r，$p_{u,i}$ 又可以表示为s维的列向量 $[p_{u,i,1}\ p_{u,i,2}\cdots p_{u,i,s}]$，其中 $p_{u,i,1}$ 为第u个包组的第i个数据包的第1个比特，其余依次类推。

[0061] 发送模块步骤101：监督包生成。

[0062] 设 $G_{g\times h}$ 为一个g行h列的RS码的系统生成矩阵，并将其表示成子矩阵形式：

[0063] $$G_{g\times h}=\left[I_g \quad W_{g\times(g+r)}\right]=$$

[0064] $$\begin{bmatrix} I_k & O_{k\times N_{out}} & W^{in}{}_{k\times N_{in}} & W^{out}{}_{k\times(r-N_{in})} \\ O_{N_{out}\times k} & I_{N_{out}} & Y^{in}{}_{N_{out}\times N_{in}} & Y^{out}{}_{N_{out}\times(r-N_{in})} \end{bmatrix}$$

[0065] I 表示单位阵，0 表示零矩阵。构造一个 k 行 k+r 列的 $G_{g\times h}$ 的缺失子矩阵，并将其作为监督包生成的生成矩阵：

[0066] $$G=\left[I_k \quad W^{in}{}_{k\times N_{in}} \quad W^{out}{}_{k\times(r-N_{in})}\right]=\left[I_k \quad W_{k\times r}\right]$$

[0067] 通过每个包组的 k 个数据包的 RS 码编码运算，得到这个包组的 r 个

[0068] 监督包 $p_u = d_u \cdot W_{k\times r}$ (1)

[0069] 其中，p_u 为监督包行向量 $[p_{u,1}\ p_{u,2}\cdots p_{u,r}]$，$d_u$ 为数据包行向量 $[d_{u,1}\ d_{u,2}\ \cdots d_{u,k}]$。

[0070] 发送模块步骤 102：跨包组监督包分配。

[0071] 将发送步骤 1 中生成的监督包分成两部分：其中一部分保存当前编码包组内的包组内监督区（$V_{u,in}$）中，使当前编码包组在不需要任何其他包组的协助下，具有一定的丢包恢复能力；另一部分累加到当前包组和后续关联的 l-1 个包组的跨包组监督区（$V_{u,out}$）中，使当前包组在必要的时候可以等待后续关联包组的到来，从而提升当前包组的丢包恢复能力。

[0072] 图 2 示例了本发明的第 u 个编码包组的结构图，通过这种监督包的分配方法，实际上将编码包组分成三部分：数据包部分 d_u，包组内监督部分 $V_{u,in}$ 和跨包组监督部分 $V_{u,out}$。

[0073] $d_u = [d_{u,1}\ d_{u,2}\cdots d_{u,k}]$

[0074] $$V_{u,in}=\left[p_{u,1} \quad p_{u,2} \quad \cdots \quad p_{u,N_{in}}\right]$$

[0075] $$V_{u,out}=\left[S_{u,1} \quad S_{u,2} \quad \cdots \quad S_{u,N_{out}}\right]$$

[0076] 其中，N_{in} 为一个包组的包组内监督包数目，$S_{u,i}$ 为第 u 个包组中第 i 个跨包组监督包，N_{out} 为一个包组的跨包组监督包数目，$S_{u,i}$ 表达式如下：

[0077] $$S_{u,i}=\sum_{j=0}^{l-1} p_{u-j,N_{in}+j\cdot N_{out}+i} \qquad (2)$$

[0078] 其中 l 为相关联的包组数。

[0079] 图 3 示例了第 u 个包组的监督包分配，图 4 示例了第 u 个包组监督包部分（包组内监督部分和跨包组监督部分）的构成。为简单起见，两图中的编码选取较小的参数：r = 3，N_{int} = 1，N_{out} = 1，l = 2。

[0080] 发送模块步骤 103：包组内二次编码。

[0081] 为了使 $S_{u,i}$ 在丢失的情况下也能够在当前包组恢复，将 $S_{u,i}$（i = 1,2，$\cdots N_{out}$）与 $P_{u,i}$（i = 1,2，$\cdots N_{in}$）建立下面的监督关系：

[0082] $$R_{u,i}=p_{u,i}+\sum_{j=1}^{N_{out}} y_{j,i}\cdot S_{u,j}$$

[0083] $$\begin{bmatrix} y_{1,1} & y_{1,2} & \cdots & y_{1,N_{in}} \\ y_{2,1} & y_{2,2} & \cdots & y_{2,N_{in}} \\ \vdots & \vdots & \ddots & \vdots \\ y_{N_{out},1} & y_{N_{out},2} & \cdots & y_{N_{out},N_{int}} \end{bmatrix}=Y^{in}{}_{N_{out}\times N_{in}}$$

[0084] 其中，i = 1,2，…N_{in}，$Y^{in}_{N_{out}\times N_{in}}$为 $G_{g\times h}$ 中的子矩阵。然后用 $R_{u,i}$ 替换包组中的 $p_{u,i}$，组成新的 $V_{u,in}$。

[0085] 图 5 示例了本发明的接收端自适应包解码的流程图，其中 N_l 表示当前包组的丢包数，k 的初值为 1，为了图示简洁起见，省略的部分操作会在下文阐述。

[0086] 每接收一个编码包组后，通过包序号来判断哪些编码包发生丢失，并计算此包组丢包数 N_l，并假设与此包组相关联的前 1-1 个包组的数据包是完整的，即没有丢失或者已经顺利恢复丢包。如果 $N_l = 0$ 或者仅仅是非数据包发生了丢失，则直接提取出当前包组的数据包部分 d_u；否则，采用下面的策略进行恢复。

[0087] 在步骤 201，判断 N_l 是否小于 $N_{in}+1 \cdot N_{out}$，如果判断为假，则进入步骤 202。

[0088] 在步骤 202，由于当前包组丢失情况超出恢复上限，放弃恢复本包组的丢包。

[0089] 如果步骤 201 的判断为真，则进入步骤 203。

[0090] 在步骤 203，判断 N_l 是否小于 N_{in}，如果判断为真，则进入步骤 204。

[0091] 在步骤 204，利用下面的矩阵形式线性方程组来恢复任意丢失小于 N_{in} 个编码包（包括数据包和跨包组监督包）：

[0092] $D_u \cdot G_{in} = R_u$

[0093] 其中，$D_u = \left[d_{u,1} \quad d_{u,2} \quad \cdots \quad d_{u,k} \quad S_{u,1} \quad S_{u,2} \quad \cdots \quad S_{u,N_{out}} \right]$，

[0094] $R_u = \left[d_{u,1} \quad d_{u,2} \quad \cdots \quad d_{u,k} \quad S_{u,1} \quad S_{u,2} \quad \cdots \quad S_{u,N_{out}} \; R_{u,1} \quad R_{u,2} \quad \cdots \quad R_{u,N_{in}} \right]$

[0095] $$G_{in} = \begin{bmatrix} I_k & O_{k\times N_{out}} & W^{in}{}_{k\times N_{in}} \\ O_{N_{out}\times k} & I_{N_{out}} & Y \end{bmatrix}$$

[0096] 如果步骤 203 的判断为假，则进入步骤 205。

[0097] 在步骤 205，利用式 2，还原出隐藏在 $S_{u,i}$（i = 1,2，…，N_{out}）中的 N_{out} 个第 u 个包组的监督包：

[0098] $$p_{u,i} = S_{u,I} - \sum_{j=1}^{I-1} p_{u-j,N_{in}+j\cdot N_{out}+I}$$

[0099] 其中，i = N_{in}+1，N_{in}+2，…，N_{in}+N_{out}，I = i-N_{in}，为第 u 个包组的第 i 个监督包累加在跨包组监督部分的位置下标。又根据假设，前 1-1 包组的数据包是完整的，所以式中的 $p_{u-j,N_{in}+j\cdot N_{out}+I}$ 通过式 1 是可求出的。然后进入步骤 206。

[0100] 在步骤 206，判断 N_l 是否小于 N_{in}+N_{out}，如果判断为真，则进入步骤 207。

[0101] 在步骤 207，将在步骤 4 中还原出的监督包和包组内监督部分 $V_{u,out}$ 联合起来，便得到了 N_{in}+N_{out} 个当前包组的监督包，这相当于将步骤 3 中的 G_{in} 扩展了 N_{out} 列：

[0102] $$G_{in}' = \begin{bmatrix} I_k & O_{k\times N_{out}} & W^{in}{}_{k\times N_{in}} & W^{out,1}{}_{k\times N_{out}} \\ O_{N_{out}\times k} & I_{N_{out}} & Y^{in}{}_{N_{out}\times N_{in}} & O_{N_{out}\times N_{out}} \end{bmatrix}$$

[0103] 其中，其中 $W^{out,1}{}_{k\times N_{out}}$ 是 $G_{g\times h}$ 中 $W^{out}{}_{k\times(m-N_{in})}$ 的前 N_{out} 列组成的子矩阵。于是和步骤 3 中类似的，利用下面的方程组可以恢复出任意丢失的小于 N_{in}+N_{out} 个编码包：

[0104] $D_u \cdot G'_{in} = R_u$

[0105] 如果步骤 206 的判断为假，则进入步骤 208。

[0106] 在步骤 208，等待第 u+1 个包组的到来。如果第 u-1+2 个包组到第 u+1 个包组这 1

个包组中，除第 u 个包组，都没有丢失数据包，或者通过步骤 3 恢复出全部数据包，则与步骤 4 相似的，可还原出压缩在 $S_{u+1,i}$（i = 1，2，…，N_{out}）中的 N_{out} 个第 u 个包组的监督包：

[0107] $$p_{u,i} = S_{u+1,I} - \sum_{j=0,j\neq 1}^{I-1} p_{u-j,N_{in}+j\cdot N_{out}+I}$$

[0108] 其中，i = N_{in}+N_{out}+1，N_{in}+N_{out}+2，…，N_{in}+2 • N_{out}，I = i-N_{in}-N_{out}。这样便总共得到了第 u 个包组的 N_{in}+2 • N_{out} 个监督包。并使 k 值加 1（k 的初值为 1），然后进入步骤 8。

[0109] 在步骤 209，判断 N_1 是否小于 N_{in}+k×N_{out}，如果判断为真，则进入步骤 9。

[0110] 在步骤 210，利用在步骤 7 中新获得的 N_{out} 个监督包，便得到了 N_{in}+k×N_{out} 个当前包组的监督包，这相当于将步骤 3 中的 G_{in} 扩展了 k×N_{out} 列，于是和步骤 6 类似的，可以恢复出第 u 个包组的任意丢失小于 N_{in}+k×N_{out} 个编码包。

[0111] 如果步骤 209 中的判断为假，则回到步骤 208，继续等待后续编码包组的到来，和步骤 208 进行相同的操作，提取出第 u 个包组的监督包，之后进入步骤 209 判断是否收集到足够的监督包，否则继续循环，直到收集到足够的监督包，并最终恢复出第 u 包组的数据包。

[0112] 性能分新

[0113] 对本文发明提出的一种数据包可靠传输的方法进行了仿真。

[0114] 仿真条件：选取 Gilbert 模型作为仿真的网络模型，从而更接近丢包具有相关性的实际网络状况，以便更有效的示例本发明的实际应用性能，其中平均丢包率从 0.01 步进到 0.1，条件丢包率为 0.3，突发丢包长度的期望为 1.427。将本发明的方案记为 C_{CRS}，传统的基于 RS 编码和奇偶校验编码的包编码方案分别记为 C_{RS} 和 C_{PC}。

[0115] 仿真一：使上述的三种方案具有相同编码效率，比较其丢包恢复性能。三种方案的编码参数分别为 C_{CRS}(15,2,1,3)，C_{RS}(15,12) 和 C_{PC}(5,4)，其中 C_{CRS}(15,2,1,3) 表示 n = 15，N_{in} = 2，N_{out} = 1，1 = 3。显然三种方案的编码效率均为 4/5。从图 6 中可看出本发明 C_{CRS} 方案的丢包恢复性能要明显好于其他两种方案。

[0116] 仿真二：使不同的方案具有相当的丢包恢复性能，比较其编码效率。图 7 为 C_{CRS}(15,2,1,3) 与 C_{RS}(15,11) 的丢包恢复性能比较，可以看出两者在丢包率适中的条件下，恢复性能比较接近。两种方法的传输效率参数如表 1 所示。从表 1 可以看出，在恢复性能不损失的条件下，本文提出的算法可以很大程度提升传输效率。

[0117]

	组内数据包数	组内冗余包数	单位数据包的冗余包数	单位数据包的冗余包数之比
CRS	12	3	3/12	(3/12) / (4/11) =11/16
RS	11	4	4/11	

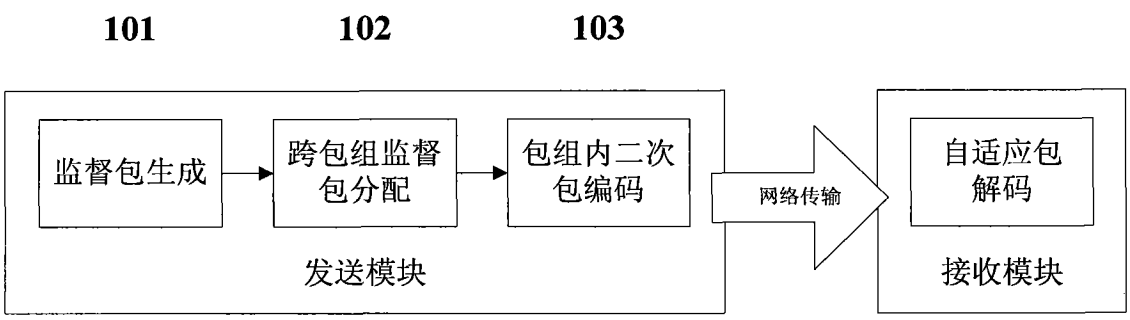

101
102
103
监督包生成
跨包组监督包分配
包组内二次包编码
网络传输
自适应包解码
发送模块
接收模块

图 1

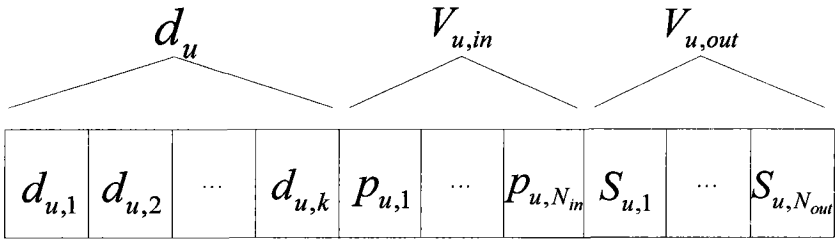

d_u
$V_{u,in}$
$V_{u,out}$
$d_{u,1}$ $d_{u,2}$ … $d_{u,k}$ $p_{u,1}$ … $p_{u,N_{in}}$ $S_{u,1}$ … $S_{u,N_{out}}$

图 2

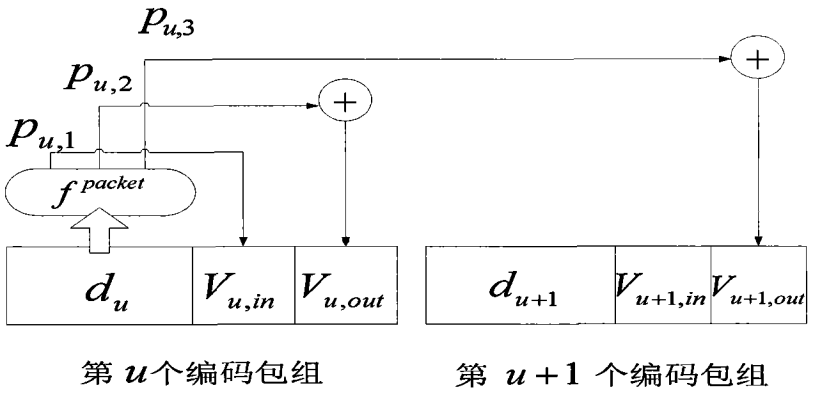

$p_{u,3}$
$p_{u,2}$
$p_{u,1}$
f^{packet}
d_u $V_{u,in}$ $V_{u,out}$
d_{u+1} $V_{u+1,in}$ $V_{u+1,out}$
第 u 个编码包组
第 $u+1$ 个编码包组

图 3

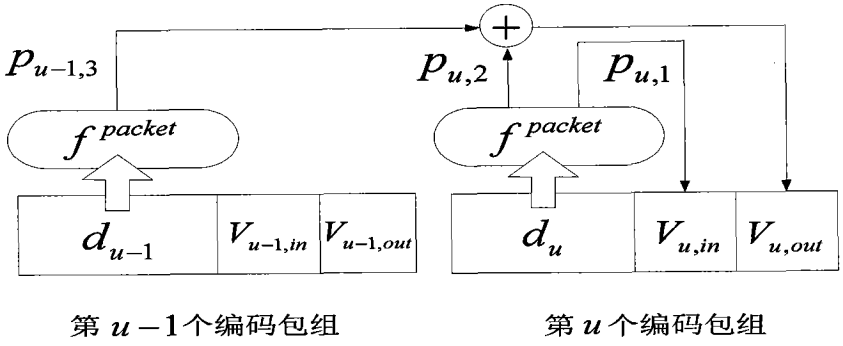

$p_{u-1,3}$
$p_{u,2}$
$p_{u,1}$
f^{packet}
f^{packet}
d_{u-1} $V_{u-1,in}$ $V_{u-1,out}$
d_u $V_{u,in}$ $V_{u,out}$
第 $u-1$ 个编码包组
第 u 个编码包组

图 4

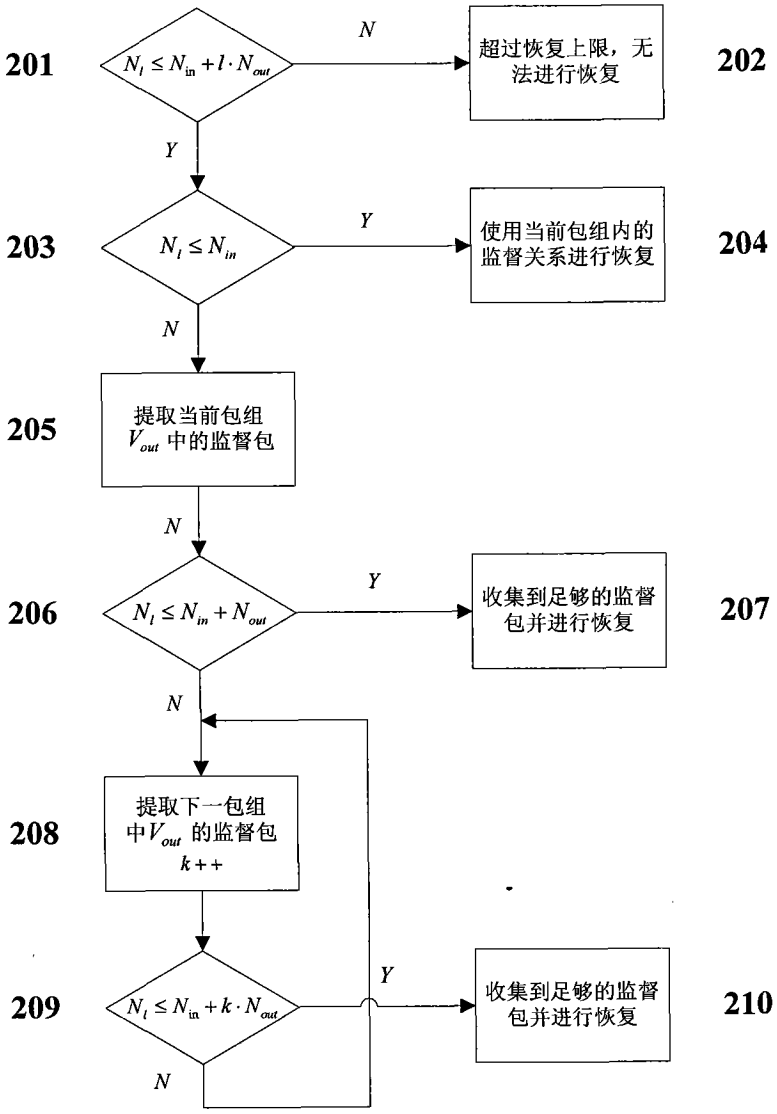

201
$N_l \le N_{in} + l \cdot N_{out}$
N
超过恢复上限，无法进行恢复
202
Y
203
$N_l \le N_{in}$
Y
使用当前包组内的监督关系进行恢复
204
N
205
提取当前包组 V_{out} 中的监督包
N
206
$N_l \le N_{in} + N_{out}$
Y
收集到足够的监督包并进行恢复
207
N
208
提取下一包组中 V_{out} 的监督包
$k++$
209
$N_l \le N_{in} + k \cdot N_{out}$
Y
收集到足够的监督包并进行恢复
210
N

图 5

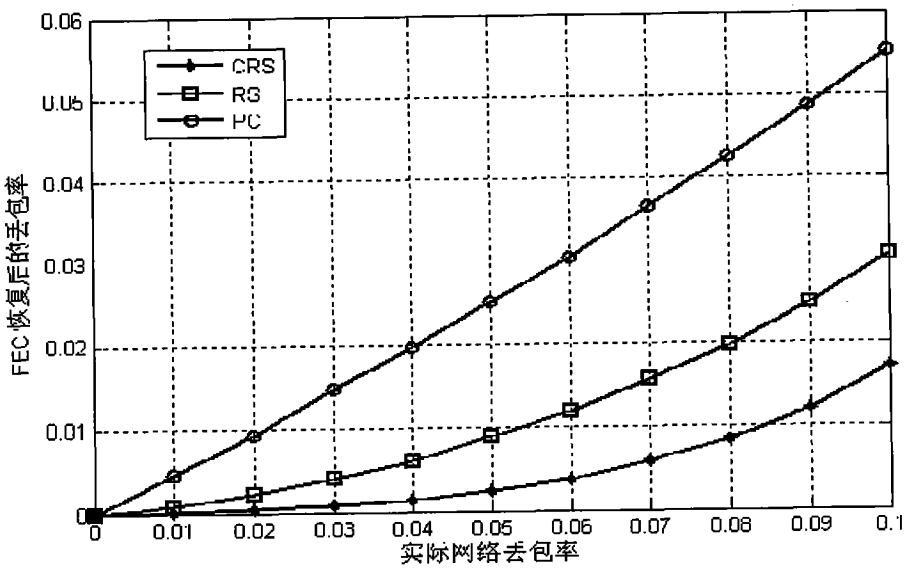

CRS
RS
PC
FEC恢复后的丢包率
实际网络丢包率
0.06
0.05
0.04
0.03
0.02
0.01
0
0.01
0.02
0.03
0.04
0.05
0.06
0.07
0.08
0.09
0.1

图 6

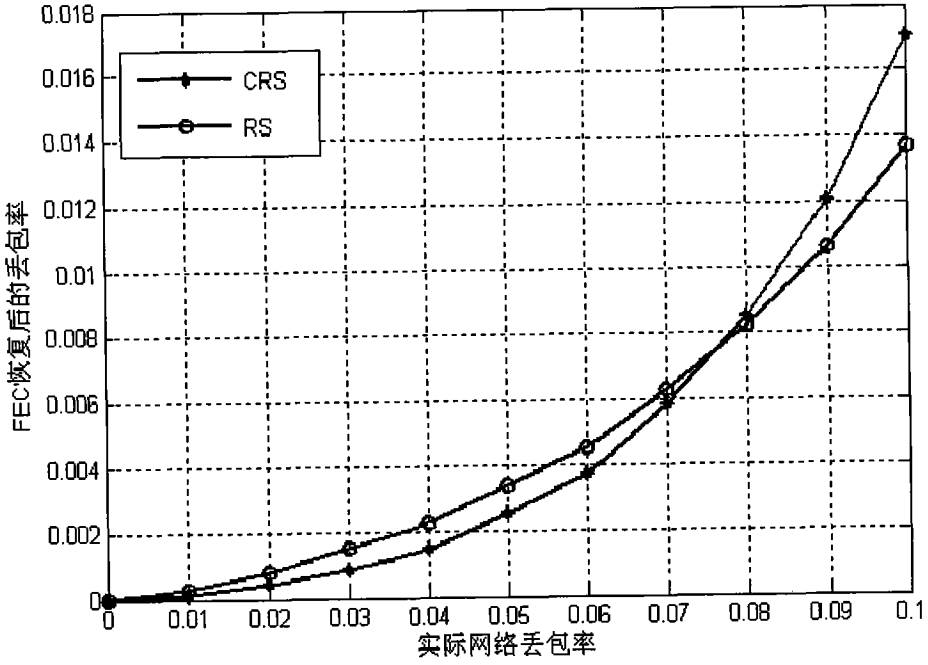

CRS
RS
FEC恢复后的丢包率
实际网络丢包率
0.018
0.016
0.014
0.012
0.01
0.008
0.006
0.004
0.002
0
0.01
0.02
0.03
0.04
0.05
0.06
0.07
0.08
0.09
0.1

图 7