doi:10.3969/j.issn.1000-436x.2014.08.021

适合 TCP 的网络编码重传机制

陈静[1]，宋学鹏[1]，刘芳[2]
（1. 科学技术部信息中心，北京 100862；2. 中国科学院 计算机网络信息中心，北京 100190）

摘 要：通过理论分析，看出基于反馈的重传方法比定量重传的方法有更低的解码延迟。提出了一种新型的基于反馈的网络编码(FNC)重传机制，利用 seen 机制中的隐含信息来获取接收方解码所需的重传分组个数，并改变了编码规则使部分分组可以提前解码。该机制不仅可以处理有固定误码率的随机分组丢失，还可以有效地应对大量突发性分组丢失。仿真结果显示，该机制在高误码率下也能保持较高的吞吐量，且极大地减少了解码延迟，传输过程基本不受分组丢失的影响，有效地对拥塞控制协议隐藏了链路错误。算法简单有效，更适于在实际系统中应用。

关键词：网络编码；TCP；解码延迟；重传
中图分类号：TP393.0 **文献标识码**：A **文章编号**：1000-436X(2014)08-0169-10

Effective retransmission in network coding for TCP

CHEN Jing[1], SONG Xue-peng[1], LIU Fang[2]
(1. Information Center of Ministry of Science and Technology, Beijing 100862, China;
2.Computer Network Information Center, Chinese Academy of Sciences, Beijing 100190, China)

Abstract: Theoretical analysis for evaluating the decoding delay and redundancy of the different retransmission schemes show that the feedback based retransmission scheme has a lower decoding delay than the scheme with a fixed retransmission rate, whereas these schemes which retransmit packets containing all packets in the coding window are deeply influenced by the loss rate. Therefore a novel network coding retransmission scheme which uses the implicit information of seen scheme to acquire the number of packets the receiver needs for decoding packets is proposed, the encoding rules of retransmission to let a part of packets decodable in advance are changed. Proposed scheme can work well on handling not only random losses but bursty losses in reducing decoding delay. More important, it isn't influenced by the loss rate, and thus can effectively mask losses. Simulation results show that the new scheme significantly outperforms the previous coding approach in reducing decoding delay and increasing throughput at the same time. It is more possible to be implemented in practical systems.

Key words: network coding; TCP; decoding delay; retransmission

1 引言

在无线网络环境中，传统的 TCP 已无法达到令人满意的性能。即使一个理想的拥塞控制协议可以在传输过程中不产生任何拥塞分组丢失，但非拥塞导致的分组丢失（包括固定误码率的随机分组丢失以及由于恶劣天气或信号被屏蔽等原因造成的突发性分组丢失）仍会严重影响传输性能。而且一个拥塞控制算法的链路利用率越高，该算法受到非拥塞分组丢失的影响也就越大[1]。因此，如何处理错误分组丢失是无线网络传输中的一个重要问题。

网络编码方法的核心思想是通过传输多个数据分组的编码组合来代替传输一个单独的分组，因

收稿日期：2013-06-15；修回日期：2013-08-29
基金项目：国防科技预研基金资助项目；中国科学院新型网络协议体系研究基金资助项目
Foundation Items: Chinese Defense Advance Research Program of Science and Technology; Foundation Project of Chinese Academy of Sciences

此，传输就是将一个原始数据分组添加到编码池中，收到确认后将数据从池中移走的过程，它有良好的应对传输错误的能力[2]。最简单的编码方式就是随机线性编码方式，通过随机产生一组系数向量，与待编码的分组向量进行矩阵乘法，就可以得到编码后的数据分组。编码后的数据传输相互独立，不受到任何其他分组丢失的影响。因此，将网络编码与 TCP 结合可以极大地提高 TCP 在无线信道中传输的健壮性和有效性。

文献[3]中提出的 ANC (ARQ for network coding)机制在多播环境中引入了 seen 分组确认的概念，使接收方可以确认尚未解出的编码分组，并将它从发送方缓冲区中移出。这使发送方的平均队列长度由传统的$\Omega((1-\varepsilon)^{-2})$减少为$\Omega((1-\varepsilon)^{-1})$，其中，$\varepsilon$为误码率。在不考虑随机算法导致系数向量线性相关的情况下，每个线性组合的到达都必然使一个新的数据分组被看到，因此，算法是吞吐量最优的。但是，由于没有对解码时间的限制要求，丢失较多的接收者可能很久都不能从分组丢失中恢复，解码时间越来越长。TCP/NC[4]将网络编码的方法与 TCP 协议进行了有效结合，在端节点的 TCP 和 IP 层之间增加了网络编码层，利用 seen 机制对拥塞控制算法隐藏了分组丢失，希望使有损信道对 TCP 来说看起来像无损信道一样，传输层可以不受处理错误分组丢失带来的影响，只要单纯解决拥塞问题即可。协议采用的是定量重传的方式，根据冗余因子 R（传输成功率的倒数）来重传数据，以此来补偿错误分组丢失并匹配 TCP 的发送速率。但这种方式可能导致大量冗余，不是吞吐量最优的。更重要的是它没有考虑解码延迟的问题，也不能很好地应对突发性分组丢失，因此并不能有效地应用于实际系统中。实际上，尽管网络编码可以带来吞吐量的提高，但大多数编码机制都没有考虑解码延迟，因此目的端需要等待较长时间才能解出一个分组，并且需保留很大的缓冲区来保存未解码的分组。文献[5]重新设计了 ANC 的编码规则并限制了解码时间，使它更适合于实时性要求较高的程序。但是，它是以牺牲一部分吞吐量为代价来换取解码延迟的减少，且由于需要保留数据在规定时间内重传，因此这种传输方式不能在分组被看到之后就将它从缓冲区中删除，队列平均长度由 ANC 的$\Omega((1-\varepsilon)^{-1})$又恢复到$\Omega((1-\varepsilon)^{-2})$。

因此，虽然通过 seen 可以在很大程度上将 TCP 感知的传输时延与接收端正确地解码出分组的时延成功解耦，但解决问题的途径有 2 个：重新设计 TCP 的返回信号，即 seen 信号，比如在发送端适当地根据重发的次数增加一些发还给 TCP 的附加 seen 信号；另一个就是本文所提出的算法，通过适当增加重传来改善这个时间间隔的波动幅度。

将已有的编码方法与 TCP 结合会面临以下问题：1）当前的编码规则会导致解码延迟的增加；2）网络编码可能会把可解码分组个数增加过程中的时间间隔波动叠加到网络拥塞造成的时延之上，从而影响 TCP 拥塞控制的正确实施，计算和仿真均表明，在信道误码率较高时，这种时间波动的影响是显著的（参见文献[5]中第 5 章的图 3，以及本文第 3 节的分析）；3）由人为设定的参数来引发的重传只能应对已知的、有固定分组丢失率的传输环境，不能很好地应对未知传输环境以及突发性分组丢失。

为了解决以上问题，提出了端到端的基于反馈的网络编码重传机制(feedback based network coding retransmission scheme)，通过适当增加重传来改善可解码分组个数增加过程中的时间间隔的波动幅度。利用 seen 机制中隐含的信息，可以得到接收端解码所需要的分组的准确数量以及时地进行重传，同时还改变了重传分组的编码规则使接收方可以提前解出部分分组。仿真结果显示，与以往的算法相比，本方法极大地提高了吞吐量并减少了解码延迟，算法不仅可以有效地处理固定速率的随机分组丢失，还可以良好地应对大量连续的突发性分组丢失，算法比以往的网络编码方法更有效地对拥塞协议隐藏了分组丢失。同时，该机制保持了 TCP 端到端的语义，编码操作仅在端节点实施。因此，该方法简单且有效，适合于在实际系统中执行。

2 背景介绍

在网络编码中，将分组视为大小为 q 的有限域 F_q 上的向量，令 $\boldsymbol{p}_k$ 为源端发送的第 k 个原始分组。当发送窗口移动使源端可以发送一个新分组时，它发送所有未确认分组的随机线性组合来代替发送一个原始数据分组。本章介绍 seen 机制以及如何将它应用于现存协议栈中。

文献[3]提出了与 ARQ 结合的网络编码机制 ANC，用于处理多播环境下的分组丢失。当接收方有足够信息能够计算出形如$(\boldsymbol{p}_k+q)$的线性表达式

时，称它看到(seen)了分组 $\boldsymbol{p}_k$，其中，$q=\sum_{l>k}\boldsymbol{\alpha}_l\cdot\boldsymbol{p}_l$，$\boldsymbol{\alpha}_l\in F_q$。接收方确认的是最久未被看到的分组，当一个分组被所有接收者都看到后，就可以从发送端的缓冲队列中删除。seen 分组确认提供了一种有效的手段使接收端即使没有解出分组也可以发送确认信息，因此发送方也可以保持较小的队列长度。文献[3]中已证明，在泊松到达速率、完美反馈、同一分组丢失率的情况下，该机制是吞吐量最优的，因为接收到的每个分组都是有效的。

文献[4]提出的 TCP/NC 协议，在文献[3]的基础上针对单个连接而设计。它将 seen 机制应用于传输控制，并在协议栈中在传输层和网络层之间引入了一个新的网络编码层，以期望达到对拥塞控制协议隐藏分组丢失的效果。

为了适应端到端的单连接传输，协议对 ANC 机制进行了调整。发送方发送的是当前窗口中所有分组的线性组合，而不是每个接收者最久未确认的分组的组合。由于每个到达分组都使得一个新的分组被看到，因此每个 ACK 都将使拥塞窗口向前推进，不会出现重复的 ACK，TCP-Reno 类的基于 3 次重复确认来判断分组丢失的快速重传、快速恢复算法不再适用。因而 TCP/NC 选择了 TCP-Vegas[6]来作为它的拥塞控制算法。并且，协议引入了一种新的 RTT 的估算方式，它将到达的 ACK 与发生在诱发上一个 ACK 之后的传输相匹配，而不是匹配引发确认的传输。如图 1(c)中，RTT_2 是从第 2 次传输开始计算，而不是从第 4 次传输开始。

为了补偿信道错误的损失以匹配 TCP 的发送速率，每发送一定数量的分组后，会发送一个冗余分组，冗余因子 R 的最佳取值为传输成功率的倒数。如图 1(a)所示，设分组丢失率为 20%，那么 R=1/0.8=1.25，即每发送 4 个编码分组后，进行一次重传，第 5 次传输即是一次重传。对 $\boldsymbol{p}_2$、$\boldsymbol{p}_3$ 和 $\boldsymbol{p}_4$ 编码，紧随第 4 次传输后发出。这次重传补偿了丢失的第 2 次传输，使 $\boldsymbol{p}_3$、$\boldsymbol{p}_4$ 被接收方看到，$\boldsymbol{p}_3$、$\boldsymbol{p}_4$ 及 $\boldsymbol{p}_5$ 变为可解。但是，这种定量重传的传输机制可能会导致如下 2 种情况：如图 1(b)所示，第 5 次重传是完全无用的，因为之前并没有分组丢失发生，不需要重传，称这是一个冗余分组；如图 1(c)所示，之前丢了 2 个编码分组，虽然第 5 次重传是有效的，但它仍不足以帮助接收端解码。因此，可以使用基于反馈的重传方式来代替定量重传。

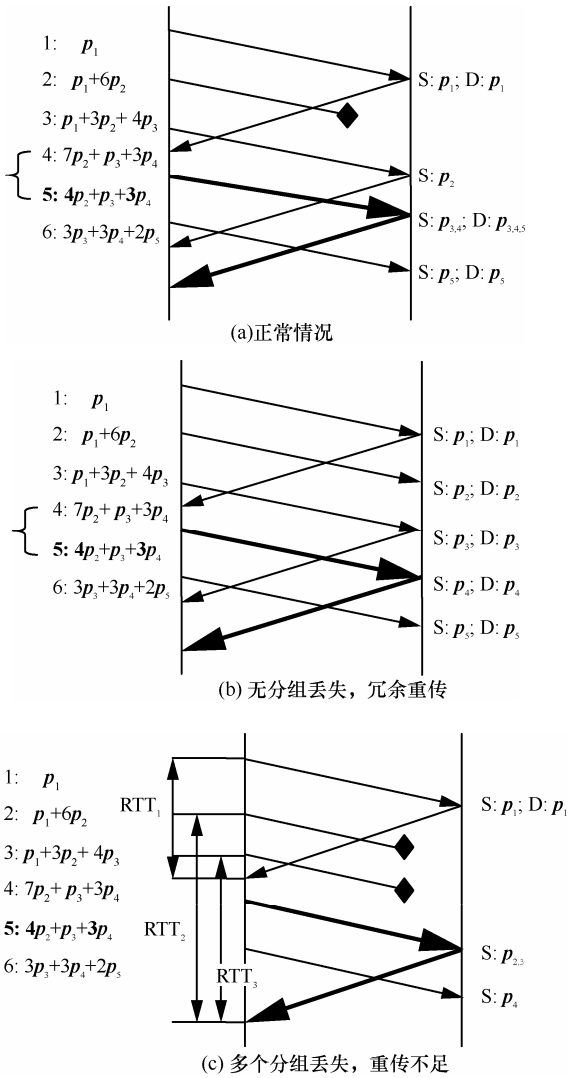


图 1　采用冗余因子重传示例

3　理论分析

在分析之前，首先阐明“解码延迟”和“冗余”这 2 个概念的含义。分组 $\boldsymbol{p}_k$ 的解码延迟是从它第一次在编码层中被编码算起，直到它在目的端被成功解出为止。因此它与传输延迟以及目的端的状态有关。这里主要关心的是如何重传使接收方可以有足够的信息解出全部的分组，图 1(b)中的第 5 次传输就是一个冗余重传。

系统模型　这里的理论分析是基于时间槽系统(slotted time system)。源端按顺序在每个时间槽内产生一个新分组（编号小的分组在前）。一个新到达的分组将引发一次传输，它由编码窗口中的所有

分组（包括这个新到达的分组）线性组合而成。由于文献[7]已证明了在大小为 q 的有限域 Fq 上，随机产生 2 个线性独立的向量的概率大于 $1-1/q$，因此，可以假设域的范围足够大使收到的每个组合都可以使新的分组被看到。

反馈 假设接收方到发送方之间有完美反馈（无错误、无丢失、无延迟）。发送方收到 $\boldsymbol{p}_k$ 被看到(seen)的确认信息后就将 $\boldsymbol{p}_k$ 从缓存队列中移出。

重传 重传不是由新的分组到达编码窗口中引发的，它发生在传输和收到反馈之后，在同一个时间槽内，以概率 r 发生。因此，重传是由编码窗口中所有的旧分组组合而成。

时间点 图 2 所示为在一个时间槽内各事件发生的先后顺序。假设包含一个新分组的一次新的传输在接收方发送反馈前到达，对该分组的反馈发生在同一时间槽内。反馈在重传发生前到达，重传发生在同一时间槽的结尾处，不占用新的时间。非拥塞分组丢失之间相互独立，发生概率为 e。

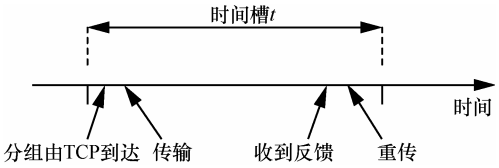


图 2 1 个时间槽内全部可能发生的事件

定义 1 （目的端的状态）目的端有 2 个状态：D（decodable）状态和 U（undecodable）状态。如果它能够解出当前所有的分组，那么目的端在 D 状态。如果有传输中分组丢失发生，目的端将不能解出后面到达的分组，称目的端在 U 状态。

当目的端在 D 状态时，在没有分组丢失发生而收到一个新的组合后，它将仍然处于可解状态，因为每次编码时只增加一个新的分组。如果目的端在 D 状态时源端发送了一个重传分组，则这是一次冗余传输。如果发生了分组丢失，那么目的端将变为 U 状态，只有当它收到足够的重传分组使组合的分组数大于等于向量维数时，才能变为可解状态再解出全部的分组。目的端的初始状态是 D 状态。因此，整个传输过程中，接收方的状态是由一系列的 D 链和 U 链组成的。在 D 状态下发生第一次分组丢失时，是一个 U 链的开始，直到分组丢失数小于等于重传数时，U 链结束。例如图 1(a)中，接收方的状态为 D,U,U,D, D,…，第一个 U 链的长度为 2。

令 x 表示当前接收方变为可解状态所需要收到的重传分组的数量。它是具有一个吸收壁的 Markov 链，其状态空间为 I：{0,1,2,…}，当到达 0 状态时，U 链结束。之后发生分组丢失视为下一个 U 链开始。因此，它的一步转移概率为

$$\begin{cases} P_{i,i+1} = e(1-r) \\ P_{i,i-1} = (1-e)r \\ P_{i,i} = er + (1-e)(1-r), j \neq i+1, i-1, j \in I, i \geqslant 1, i \in I \\ P_{i,j} = 0 \\ P_{0,0} = 1 \end{cases} \tag{1}$$

设链 U_i 的长度为 L，因此，这 L 个分组需要在 U 链结束时才能被解码，如图 3 所示，对总的延迟的贡献为 $L+(L-1)+(L-2)\cdots+1=(1+L)L/2$ 。因此，所有分组的解码延迟总和 $T=f(L_1)+f(L_2)+\cdots+f(L_k)+C$，其中，$k$ 是本次传输中 U 链的总个数，$f(x)=(1+x)x/2$，$C=N-L_1-L_2-\cdots-L_k$，N 是一次传输的总分组数，C 即是 D 链总长。所有 U 链服从同一分布。因此，T 与 U 链长度 L 有关，L 越短，则 T 越短。

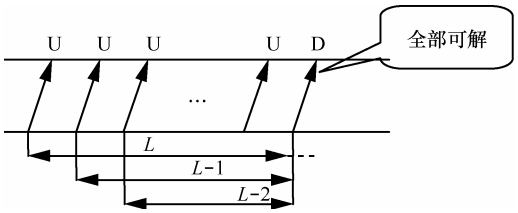


图 3 不可解链示意

命题 1 U 链长等于 l 的概率为

$$P(L=l) = \sum_{i=1}^{\lfloor l/2 \rfloor} \mathrm{C}_{l-2}^{l-2i} \mathrm{Catalan}(i-1) e^i r^i (1-e)^i (1-r)^i (1-e-r+2er)^{l-2i} \tag{2}$$

证明 U 链长度为 l 表示 x 在 l 步（时间槽）后会返回到 0 状态。换言之，在经过 l 个时间槽后，前进一步的总次数必须等于后退一步的总次数，而在 l 步之前，前进的次数必须一直大于后退的次数才能保证 U 链没有结束。令 F_i 为 l 次事件中前进总次数为 i 的概率。因此 l 步后后退事件的总数也为 i，而在原地不动的次数为 $l-2i$。第一次事件必然是前进一步，而最后一次事件必然是后退一步至 0 状态。因此，可以得到 $F_i = P_{i,i+1}{}^i P_{i,i-1}{}^i P_{i,i}^{(l-2i)} \mathrm{C}_{l-2}^{l-2i} \mathrm{Catalan}(i-1)$，其中，$0<i\leqslant\lfloor l/2 \rfloor$。而 $P(L=l)$就是所有 F_i 的总和，从而得到了式(2)。

显然，U 链的长度深受分组丢失率 e 的影响。对以固定速率重传的方法来说，可以调整上述分析，认为一个时间槽内可以发送 m 个新分组，然后再发送一个重传分组。因此，一步转移矩阵是对式(1)的扩展。

$$\begin{cases} P_{i,i-1}=\mathrm{C}_m^0(1-e)^m \\ P_{i,i}=\mathrm{C}_m^1 e(1-e)^{m-1} \\ P_{i,i+k}=\mathrm{C}_m^{k+1}e^{k+1}(1-e)^{m-k-1} \quad 0<k<m,k\in I, \\ P_{i,j}=0 \qquad j\geqslant i+m \ \text{ 或 } \ j<i-1,j\in I,i\geqslant 1,i\in I \\ P_{0,0}=1 \end{cases} \tag{3}$$

文献[4]中，通过实验证明了当 $m=(1-e)/e$ 时定量重传的方法可以获得最优吞吐量。因此，这种方式深受分组丢失率 e 的影响。并且，$P(L=l)$永远是大于零的，也就是说，在任何环境下，定量重传的方法都有可能会出现很长的 U 链。而如果加快重传的速度，即改变 m 的值，以减少解码延迟，但将发送大量无用的冗余分组会降低吞吐量。因此，定量重传机制无法有效地平衡吞吐量和解码延迟。

相比之下，对基于反馈的重传机制而言，所有分组丢失都将在一个 *RTT* 后重传。将一个 *RTT* 作为一个时间槽，那么该模型仍可以看作是对上述模型的一个扩展：在一个时间槽内先发送 m 个新分组，在该时间槽的后半部分重传所有分组丢失。因此，l 步内的所有事件的发生顺序应为：前 $l-1$ 步只能前进或停止不动，第 l 步时回退至 0 状态。因此，F_i 的系数变为 C_{l-2}^{l-2i}，远小于定量重传的因子 $\mathrm{C}_{l-2}^{l-2i}\mathrm{Catalan}(i-1)$。并且，U 链必将在一个 RTT 后结束，即如果当前发送窗口为 wnd，那么 $P(L>wnd)$的概率为 0。因此每个分组的解码延迟不超过 1.5*RTT*。

但是，以上讨论中一直没有考虑传输延迟以及反馈延迟所带来的影响。实际上，在接收方发回反馈后，仍然会有来自 TCP 的新的分组进入编码窗口，如果在发送确认以及重传的过程中，有新的分组丢失发生，那么即使算法按照反馈信息重传了相应数量的分组，接收方仍将处于不可解状态。因此，如果假设重传分组是由当前编码窗口中的所有分组组成的话，基于反馈的重传机制也同样会受到分组丢失率 e 的影响，解码延迟仍然很大。因此，可以改变了重传分组的编码规则来克服这个问题。理论上，忽略重传分组丢失（二次重传分组丢失率为 e^2），在不考虑编解码操作所花费的时间后，每个分组的解码延迟不超过 1.5*RTT*。

4　协议设计

为了缩短解码延迟和冗余的数量，提出了基于反馈的网络编码重传机制 FNC（feedback based network coding retransmission scheme），它以 TCP/NC 系统为背景，通过确认已发出的分组和已看到的分组的序号之差来获知接收方变为可解状态所需要重传的分组的数量。首先，源端的一个分组由 TCP 到达编码层后，编码层产生一个编码窗口中所有分组的随机线性组合，并将其发送到目的端。利用 seen 机制，接收方已收到的分组的最大序列号减去已看到的分组的数量，就是解码所需要的分组的数量。因此，在接收方需维护 2 个变量：已看到的编码分组的数量 *SEEN_CNT*；已收到的编码分组中包含的最大序列编号 *MAX_SEQ*。在每个 ACK 中，目的端不仅在分组头中嵌入最久未看到的分组的序列号，还要加入 *MAX_SEQ* 与 *SEEN_CNT* 的差值，称为 *DIFF*。举例说明，假设源端发送了以下 3 个线性组合：$\boldsymbol{x}=\boldsymbol{p}_1$，$\boldsymbol{y}=\boldsymbol{p}_1+\boldsymbol{p}_2$，$\boldsymbol{z}=\boldsymbol{p}_1+\boldsymbol{p}_2+\boldsymbol{p}_3$，但第 2 次传输的 $\boldsymbol{y}$ 在传输过程中丢失，目的端仅接收到组合 $\boldsymbol{x}$ 和 $\boldsymbol{z}$。因此，在接收端收到 $\boldsymbol{z}$ 后，$\boldsymbol{p}_1$ 和 $\boldsymbol{p}_2$ 可以被看到，*SEEN_CNT* 值为 2，而最大序列号 *MAX_SEQ* 为 3，所以 *DIFF* = 3−2 = 1。这表示目的端需要 1 个重传分组才能转为 D 状态。在发送端，如果收到的 ACK 的 *DIFF* 大于 0，那么发送方进行以下判断来决定是否重传。首先，检查当前时间与上次重传时间的差值，是否大于定时器超时值（设置为一个稍大于 RTT 的值）。如果大于，那么发送方重传 *DIFF* 数量的编码分组，该重传分组由编码窗口中的前 *DIFF* 个原始分组的线性组合组成，这样做可以避免一个 RTT 内的多次重传；如果不是，那么发送方比较当前收到的 *DIFF* 值是否大于上次的 *DIFF* 值(*LAST_DIFF*)，如果确实大于，表示有新的分组丢失发生，那么重传(*DIFF*−*LAST_DIFF*)个，重传分组由当前编码窗口中的前(*DIFF*−*LAST_DIFF*)个原始分组线性组合而成。

如果重传分组是由当前窗口中的所有分组组合而成的，就像 TCP/NC 协议中的设计一样，尽管解码延迟有所降低，但仍然远高于一个 RTT。这主要是由于传输是有延迟的。在一个分组的传输过程中，后面发送的编码分组仍然有可能丢失，而发回的确认却不包含之后的分组丢失数量，但编码窗口

中却包含了这些新产生的分组，因此，在源端收到确认准备重传时，实际的 *DIFF* 值已经大于确认分组中显示的值，重传分组的数量仍然不足以令接收端结束 U 链变为 D 状态。例如图 4 中，当源端收到确认后，实际上已经丢了 2 个分组，但发送方仍然只重传一个分组，如果重传的编码分组由当前编码窗口中的 $\boldsymbol{p}_2$、$\boldsymbol{p}_3$ 及 $\boldsymbol{p}_4$ 组成，那么接收方将仍然处于不可解的 U 状态，U 链长度 4，所有分组都是不可解的。而改进后的方法只重传编码窗口中的前 *DIFF*（或 *DIFF*−*LAST_DIFF*）个分组的线性组合，可以使部分分组得以解出。这样尽管 U 链还没有结束，但使最小不可解的分组向后移动了，也就减少了平均解码延迟。而且，对更少的分组进行编、解码操作也可以降低操作的复杂性和花费。因此，在图 4 的例子中，第一次确认收到的 *DIFF* 值为 1，所以重传的编码分组仅由 $\boldsymbol{p}_2$ 组成，这样成功解出了 $\boldsymbol{p}_1$ 和 $\boldsymbol{p}_2$。

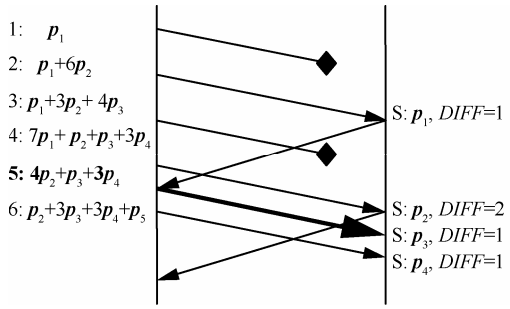


图 4 基于反馈的重传示例

为了更完整地描述 FNC 算法，给出其对 NC 算法改进部分的伪代码描述。

源端：

初始化：

令 *LAST_DIFF*, *Tlast* = 0；

从目的端收到 ACK 后:

源端从缓冲区删除被确认的分组，并获取 *DIFF* 值，如果这是第 4 次连续的 *DIFF* 大于 0，那么

1) If (*Tnow*−*Tlast* > RTT)
令 *LAST_DIFF* = 0;
goto 3);
2) If *DIFF* ≤*LAST_DIFF* goto 5)；
3) 以下步骤重复(*DIFF* − *LAST_DIFF*)次:
将编码窗口中的分组产生一个随机线性组合；
投递分组至 IP 层；
4) 更新 *Tlast* 为当前时间;
5) 更新 *LAST_DIFF*= *DIFF*;

接收端：

初始化：

令 *SEEN_CNT*, *MAX_SEQ* =0；

从源端收到分组后：

1) 进行高斯消除，更新能 seen 的分组序号；
2) 更新 *SEEN_CNT* 和 *MAX_SEQ*；
3) 将网络编码 ACK 头添加在 TCP 的 ACK 头之前，*DIFF* 为 *MAX_SEQ* 和 *SEEN_CNT* 的差值。

算法不仅避免了当接收方在可解状态下的冗余传输，且当它在不可解状态时，可以有效地重传合适数量的分组使接收方尽快地由 U 状态转为 D 状态，并提前解出部分分组。这极大地减少了了解码延迟和冗余，最重要的是它不是以牺牲吞吐量为代价的性能提高。理论上，不考虑系数向量的相关性，每个收到的分组都是有效可更新的，因此算法的吞吐量是最优的。实际上，该算法并不会受到系数产生算法和有限域大小的影响，即使编码算法会产生一些线性相关的系数向量，该方法仍能有效地工作。并且，不同于 SNC[5]，该方法并没有改变 seen 分组确认的本质，因此队列长度的期望仍然是 $\Omega((1-\varepsilon)^{-1})$，而不会因为提高了解码速度而导致队列长度的增加。

与传统的重传机制相比，算法关注分组丢失的总量，极大地减少了分组头的开销。与文献[4]中的方法相比，该方法更好地对拥塞算法隐藏了分组丢失。它不仅可以处理以常值速率丢失的随机分组丢失，还可以处理由于其他原因可能出现的突发性的连续分组丢失。同时该机制保留了 TCP 端到端的语义，编码操作仅在端节点进行，实现了对拥塞控制算法隐藏分组丢失的目的。

实际上，TCP/NC 粗略地通过以重传比例为传输成功率的倒数来补偿错误分组丢失，而拥塞分组丢失仍然是由拥塞算法重传。但由于在编码层的重传算法无法区分分组丢失原因，因此，以上的重传算法不适合可能产生很多拥塞的算法。对此，例如与 Vegas 结合时，可以在编码层限制重传分组的数量，使每一时刻已重传的总数不得超过传输总量×误码率，与 TCP/NC 均匀地将分组丢失分散地重传相比，该方法可以在需要重传的时间重传需要数量的分组，而又不会隐藏拥塞分组丢失。而对于像 VCP[8]、MLCP[9]这类基本零拥塞分组丢失的拥塞方法来说，则完全没有必要采用上述策略，这样可以更好地处理突发的以及并没有固定误码率环境下的未知分组丢失，具有更低的解码延迟[10]。

5　仿真结果

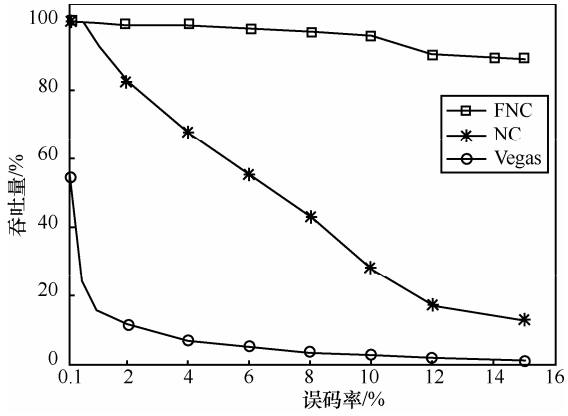
为了与现有的协议进行比较，采用目前流行的 NS-2[11]仿真软件，按照文献[4]提出的仿真拓扑结构进行仿真实验，分组传输经过 4 跳，分组长 1 000 byte。将 NC、FNC 这 2 种网络编码方法分别与 Vegas 结合，Vegas 的参数为 α=28，β=30，γ=2；NC 中的冗余因子 R 取值为分组丢失率的倒数，以达到 NC 的最优性能。衡量标准包括吞吐量、平均解码延迟以及最大解码延迟。结合了网络编码的方法的吞吐量是指仿真结束时目的端已经看到的分组的数量，而不是已解码的分组的数量。相同场景下，将有误码下的 Vegas、NC 以及 FNC 与没有误码下的 Vegas 得到的吞吐量进行比较，对比不同重传机制隐藏分组丢失的效果。所有实验的仿真时间为 200 s，统计吞吐量时忽略前 20 s。每个实验运行 50 次，结果取平均值。

5.1　随机误码对协议的性能影响

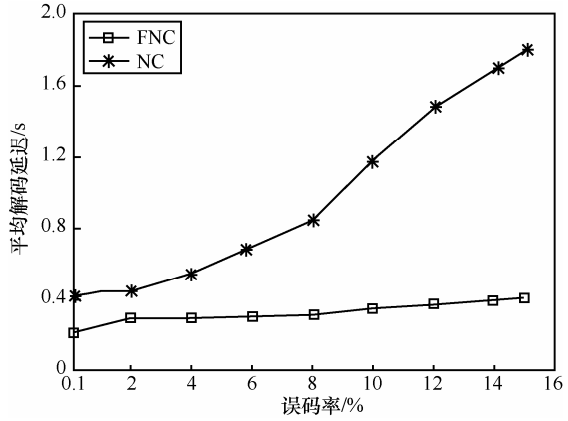
链路带宽定为 10 Mbit/s，往返延迟 80 ms，分组丢失率从 0%到 15%范围变化。图 5(a)显示出随着误码率的增加，Vegas 的吞吐量大幅下降。NC 的吞吐量比 Vegas 要好，但比 FNC 的性能差，因为它只成功隐藏了部分分组丢失。相比之下，FNC 可以很好地应对错误。虽然随着误码率增加造成了 2 次以上重传分组丢失的概率增加，吞吐量下降缓慢，远小于 NC 受到的影响，FNC 保持了 88%以上的吞吐量，有效地向 TCP 隐藏了错误。同时，本文方法还维持了相当低的解码延迟，从图 5(b)中可以看出，FNC 的平均解码延迟并没有随着误码率的增加而增加。这是因为本文重传方式是基于反馈的，因此主要由 *RTT* 和编解码操作所花费的时间决定，并不会受到误码率的影响。相比之下，NC 则深受误码率的影响，因为它的一次重传很可能并不足以补偿当前的分组丢失量。在有些情况下，NC 的平均解码延迟甚至高于 FNC 的最大解码延迟。

5.2　传输延迟对协议的性能影响

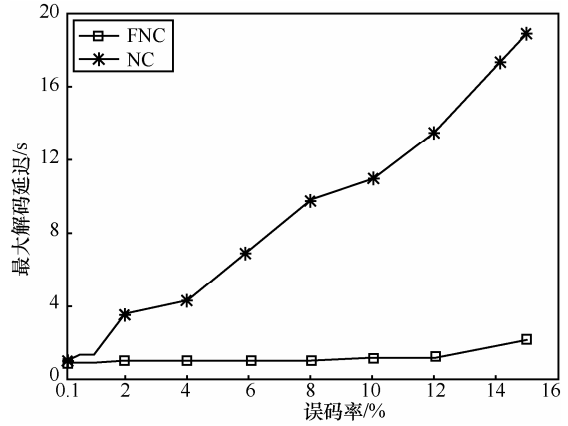
这一部分研究传输延时对不同算法的影响。本场景的链路带宽为 10 Mbit/s，误码率 5%，*RTT* 变化范围为 10 ms 至 1 000 ms。如图 6(a)所示，在每种情况下，FNC 的吞吐量都优于 NC 和 Vegas。FNC 的吞吐量基本都是 NC 的 2 倍以上。不过，随着 *RTT* 增加至 1 000 ms，FNC 的吞吐量大幅下降。这是由于本文方法是基于反馈来重传的。因此 FNC 并不适合应用于具有超长传输延迟的深空通信中（*RTT* 可能达到几十分钟）。不过在现实环境中，绝大多数情况的 *RTT* 都不会超过 1 000 ms。在图 6(b)中可以看到，NC 的平均解码延迟均高于 FNC，是 FNC 的 2 到 10 倍。而随着 *RTT* 的增加，超过 500 ms 后，2 种方法的解码延迟有着明显的增加：当往返延迟为 500 ms，FNC 的平均解码延迟为 1.001 65 s，当往返延迟为 1 000 ms，它的平均解码延迟为 2.353 03 s，



(a) 吞吐量

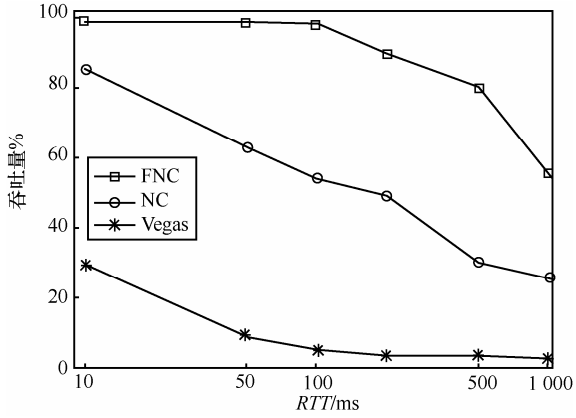


(b) 平均解码延迟

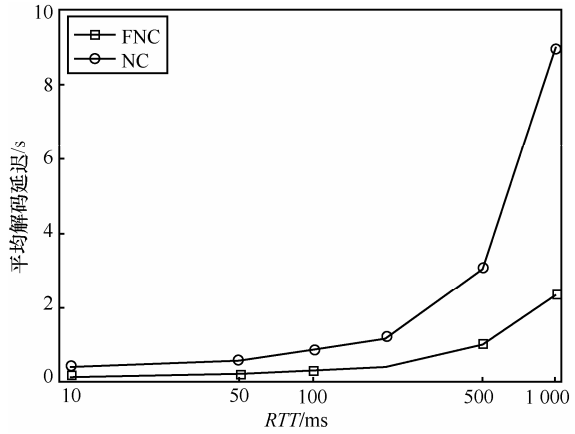


(c) 最大解码延迟

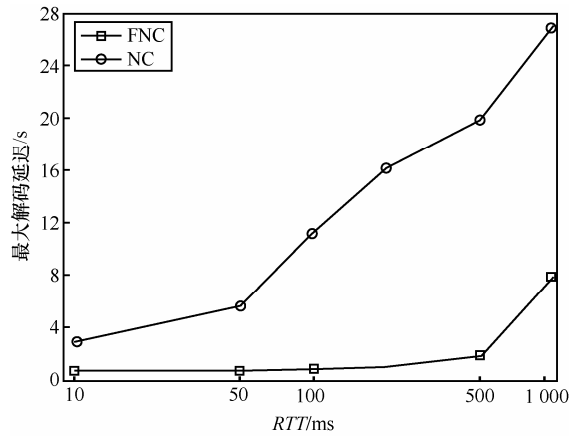
图 5　误码率的影响



(a) 吞吐量



(b) 平均解码延迟



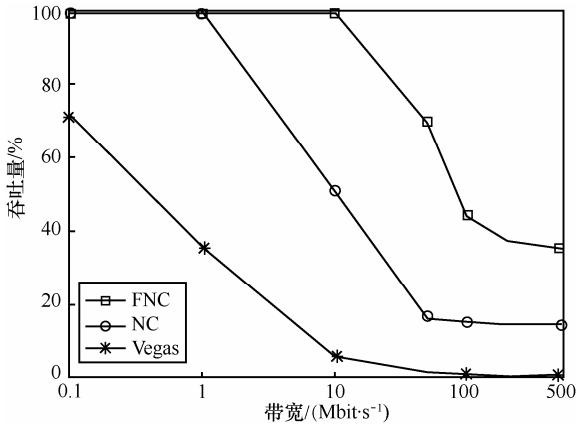
(c) 最大解码延迟

图 6　RTT 的影响

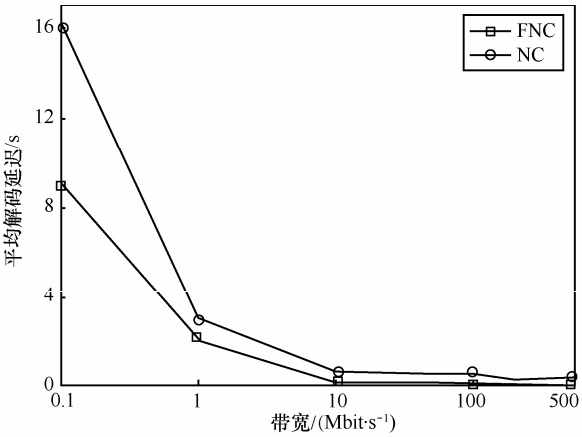
在 RTT 的 2 倍左右，符合预期范围。理论上，NC 的解码延迟不应随着 *RTT* 的增加有较大变化，但是网络编码的方法必须要与具体的拥塞控制算法相结合。Vegas 算法并不是一个适宜在长延时环境下传输的算法，随着 *RTT* 的增加将导致反馈减慢，拥塞窗口下降，进而发送速率下降，因此接收方并不能及时地收到补偿错误的重传分组，所以从实验结果中可以看出 NC 方法的解码延迟也会随着 *RTT* 的增加迅速增加，而不可能如理想中的保持不变。

5.3　链路带宽对协议的性能影响

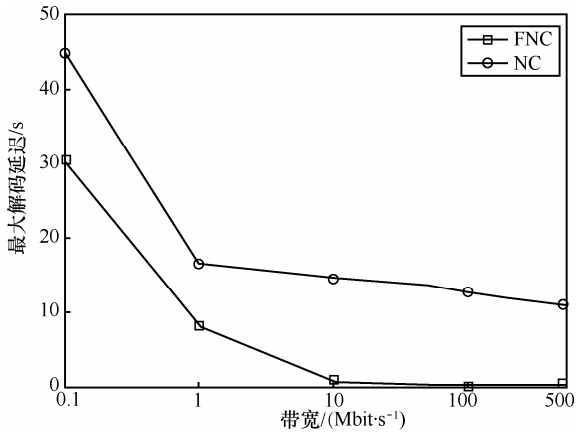
此场景中，设往返延时为 80 ms，误码率为 5%，研究链路带宽对协议的影响。如图 7 所示，与 NC



(a) 吞吐量



(b) 平均解码延迟



(c) 最大解码延迟

图 7　带宽的影响

结合的算法的最大解码延迟在各种带宽下都相当高，而 FNC 方法的最大和平均延迟都较低。在低带宽时，由于拥塞控制算法本身导致了大量拥塞分组丢失，因此，解码延迟会增加很多。在高带宽下，每个分组丢失都将严重影响 Vegas 的性能，因此，所有算法的性能下降都比较严重。而本文方法由于隐藏分组丢失的效果更好一些，所以性能好于其他 2 个协议。

5.4　突发性连续分组丢失对协议的性能影响

最后，测试突发性连续分组丢失场景下协议的性能。仍然设带宽 10 Mbit/s，延迟 80 ms。背景误码设为 0.1%，在第 40 s 时，误码率变为 50%，持续时间 5 s，在第 80 s 时误码率变为 100%，持续时间 2 s，也就是说，信号完全被障碍物遮蔽了 2 s，会产生大量连续分组丢失。被测试的 Vegas 和 FNC 并不知道分组丢失率的变化情况。图 8 显示了 FNC 能够快速且有效地应对突发性分组丢失。在 40 s 之前，FNC 基本上能够隐藏所有的随机分组丢失，而没有结合网络编码的 Vegas 的吞吐量则受到随机误码的严重影响。当分组丢失率增加至 50%后，Vegas 基本不能工作，而 FNC 仍然保持了较高的吞吐量。它之所以没有隐藏全部分组丢失是由于重传分组也有 50%发生错误的可能。在 62 s 后，FNC 有一次突然的吞吐量的增加，这是由于发送方一起重传了刚才所有没有发出去的分组。由于 Vegas 拥塞控制算法本身的原因，稳定后的发送速率没有恢复到突发分组丢失之前的水平。FNC 的分组的最大解码延迟为 3.330 53 s，这其中包含了在 60 s 处连接断开的 2 s，因此实际恢复时间仅为 1 s 多。分组平均解码延迟为 0.083 904 2 s，即一个 *RTT* 左右。

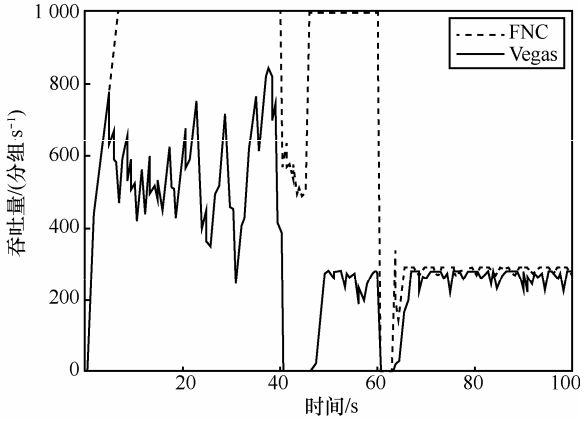


图 8　突发性分组丢失的影响

由于 NC 算法需要依赖于对误码率的估计算法，因此无法对其进行评判，但可以从 5.1 节的结果中看到 NC 隐藏分组丢失的效果不可能优于 FNC，同时，即时 NC 能够实时知道当前分组丢失率，那么在 60 s 至 62 s 之间，NC 每发送 1 个新分组后会发送 1 个重传分组（虽然重传分组也不能正确到达），而在 62 s 后，NC 获得的当前分组丢失率仍然是 0.1%，可以预料，NC 需要相当长的时间，甚至可能是在仿真结束时才可能从这次 burst 分组丢失中恢复。

6　结束语

网络编码是处理错误丢失的有力工具，但其解码延迟和传输冗余量都严重影响了传输性能，因此已有算法大都不适于在实际系统中应用。本文首先分析了使用不同重传机制的传输性能，可以看出，基于反馈的重传方法比定量重传的算法有着更低的解码延迟。但如果这些方法使用的重传方式是基于将当前编码窗口中的所有分组组合在一起的话，其性能将深受误码率的影响。由此提出了基于反馈的网络编码重传机制。它基于网络编码中使用的 seen 确认方式，根据可获得的隐含信息，准确地获知解码所需的重传分组的数量。另外，还改变了编码规则，使尽可能早地解出部分分组。该方法不仅可以应对随机分组丢失，还能有效地应付突发性的连续分组丢失。仿真实验显示，相比于定量重传的方法，本文方法极大地减少了解码延迟并更有效地隐藏了分组丢失。同时，本机制保持了 TCP 端到端的语义，编码操作仅在端节点实施。该方法简单且有效，适合在实际系统中执行。

参考文献：

[1] BARAKAT C, ANTIPOLIS S, ALTMAN E. On TCP performance in an heterogeneous network: a survey[J]. IEEE Communications Magazine, 2000, 38(1): 40-46.

[2] HO T. Networking from a Network Coding Perspective[D]. Cambridge: Massachusetts Institute of Technology, 2004.

[3] SUNDARARAJAN J K, SHAH D, MEDARD M. ARQ for network coding[A]. International Symposium on Information Theory, IEEE ISIT[C]. Toronto, Canada, 2008.1651-1655.

[4] SUNDARARAJAN J K, SHAH D, MEDARD M, *et al*. Network coding meets TCP: theory and implementation[J]. Proceedings of the

IEEE, 2011,99(3):490-512.

[5] BARROS J, COSTA R A, MUNUARETTO D, *et al*. Effective delay control in online network coding[A]. The 28th Conference on Computer Communications, IEEE INFOCOM[C]. Brazil, 2009. 208-216.

[6] BRAMKO L S, O'MALLEY S W, PETERSON L L. TCP vegas: new techniques for congestion detection and avoidance[A]. Proceedings of the SIGCOMM '94[C]. Symposium, 1994.

[7] LUN D S, MEDARD M, EFFROS M. On coding for reliable communication over packet networks[J]. Physical Communication, 2008,1(1): 3-20.

[8] XIA Y, SUBRAMANIAR L, STOICA I, *et al*. One more bit is enough[J]. IEEE Transactionin in IEEE/ACM Trans Networking, 2008, 16(6):1281-1294.

[9] QAZI I A, ZUAI T. On the design of load factor based congestion control protocols for next-generation networks[A]. The 27th Conference on Computer Communications[C]. Phoenix, AZ, 2008. 96-100.

[10] CHEN J, TAN W, LIU L X, *et al*. Towards zero loss for TCP in wireless networks[A]. Performance Computing and Communications Conference (IPCCC)[C]. Scottsdale, AZ, 2009. 65-70.

[11] NS-2 Network Simulator[EB/OL]. http://www.isi.edu/nsnam/ns/.

作者简介：

陈静（1983-），女，江西定南人，博士，科学技术部信息中心助理研究员，主要研究方向为计算机网络、电子政务。

宋学鹏（1977-），男，河北邢台人，博士，科学技术部信息中心高级工程师，主要研究方向为电子信息科学与技术。

刘芳（1982-），女，内蒙古鄂尔多斯人，博士，中国科学院副研究员，主要研究方向为基于GPU的通用计算和真实感绘制算法。

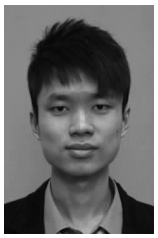
（上接第 168 页）

[26] Regular expression processor[EB/OL]. http://regex.wustl.edu/index.php/Main_Page, 2013-12-01.

[27] AGRAWAL B, SHERWOOD T. Modeling TCAM power for next generation network devices[A]. Proceeding of IEEE International Symposium on Performance Analysis of Systems and Software[C]. Austin, TX, USA, 2006.

黄昆（1978-），男，江西永丰人，博士，中国科学院助理研究员，主要研究方向为未来互联网、网络安全。

作者简介：



丁麟轩（1988-），男，山东烟台人，湖南大学硕士生，主要研究方向为网络安全。

张大方[通信作者]（1959-），男，上海人，博士，湖南大学教授、博士生导师，主要研究方向为可信系统与网络、网络安全、软件工程。E-mail:dfzhang@hnu.edu.cn。