doi:10.3969/j.issn.1000-436x.2014.07.009

有损网络下的高性能传输控制协议研究

潘凯，李挥
（北京大学 深圳研究生院 深圳市融合网络播控工程实验室 深圳市云计算关键技术和应用重点实验室，广东 深圳 518055）

摘 要：提出了一种基于网络编码的传输控制协议中冗余系数的更新算法，用来解决实际通信中冗余系数无法预先确定的问题。数据发送前采用简化系数的线性网络编码对数据进行编码，在方便解码的同时减小了系数在首部的开销。在对算法进行详细设计后，使用 NS2 软件对该算法在不同场景的有损网络下进行仿真测试。测试结果表明，使用该算法的协议在保持原有协议公平性的同时在有效性和适应性方面（尤其在低负载网络上）都远远优于现有协议及固定冗余系数的网络编码传输控制协议。

关键词：网络编码；冗余系数；有损网络
中图分类号：TN915 **文献标识码**：A **文章编号**：1000-436X(2014)07-0070-07

Research on the high-performance transmission control protocol under lossy networks

PAN Kai, LI Hui
(Shenzhen Key Lab of Cloud Computing Technology and Application, Shenzhen Engineering Laboratory of Converged Networking Technology, Peking University Shenzhen Graduate School, Shenzhen 518055, China)

Abstract: A novel redundancy parameter updating algorithm based on network coding combined TCP was proposed. Reduced coefficients linear network coding which could simplify decoding and simultaneously reduce the overhead in the header was involved in encoding process before data transmission. During the simulations, NS2 was used to test the communication under different lossy networks. Results showed that the new protocol which adopted this algorithm retained the fairness and outperformed the other protocols in effectiveness and adaptability especially under light-load networks.

Key words: network coding; redundancy parameter; lossy network

1 引言

众所周知，作为 IP 协议族核心协议之一的传输控制协议(TCP)为计算机端到端的通信提供了可靠保障。虽然其在因特网中扮演着如此重要的角色，但在有损网络中却由于外部干扰而表现得不尽如人意。导致这一问题的根源是其错误地将所有数据分组丢失归结为网络拥塞而盲目减小拥塞窗口，事实上这一做法似乎仅在有线网络中可行。于是，被强制减小的拥塞窗口由于必须经历拥塞避免阶段的保守增加而导致链路使用率低下。

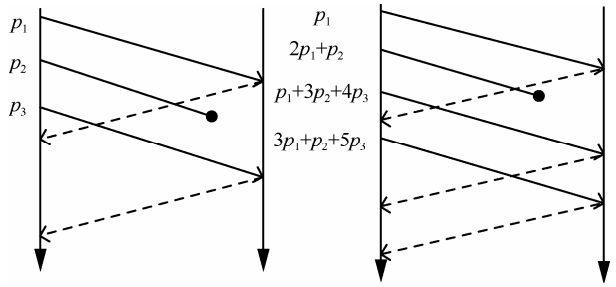
作为通信网络（尤其是无线网络）重要潜在技术之一的网络编码（network coding），其出现引起了世界范围内的极大关注。信息通常被看作是无法继续压缩的管道流水，而网络编码对其再次处理，从而得到了顽健性和有效性的进一步提升。

以顺序传输和顺序响应为特点的 TCP 协议在某些极端情况下可能导致“队头阻塞”问题，即接

收稿日期：2013-03-17；**修回日期**：2013-05-20
基金项目：国家重点基础研究计划（“973”计划）基金资助项目（2012CB315904）；国家自然科学基金资助项目(60872010)；深圳基础研究基金资助项目（JC201104210120A）；广东省自然科学基金资助项目（2011010000923）
Foundation Items: The National Basic Research Program of China (973 Program) (2012CB315904)；The National Natural Science Foundation of China (60872010); The Basic Research Program of Shenzhen (JC201104210120A)；The Natural Science Foundation of Guangdong Province (2011010000923)

收方已获得序号较大的数据分组但迟迟未见序号较小的“队头”数据分组。网络编码能够很好地解决这一问题，经过编码操作，编码分组（原始数据分组的线性组合）的数量将代替原来的序号成为值得关心的重点。

网络编码和 TCP 协议相结合最早由 MIT 提出[3]。通过在收发双方协议栈的传输层和网络层之间加入独立的网络编码层来实现上述结合思想。事实上，对传输层掩盖数据分组丢失这一思想的普遍解决方法是通过链路层进行重传[4]，但这样会使得链路层重传和传输层重传的交互过程变得异常复杂。新加入的网络编码层致力于通过冗余分组的方式向上层掩盖丢失。一旦接收方收到数量足够的编码分组，便能通过解码得到原始信息，上述思想可由图 1 说明。



(a) 传统 TCP 协议：接收方必须确切收到每一个数据分组

(b) 带有 NC 功能的 TCP 协议：接收方只需收到一定数量的编码数据分组便可恢复原始信息

图 1　传统 TCP 协议与带有 NC 功能的 TCP 协议

冗余对于传统 TCP 协议来说并不容易实现，因为发送方无法预先知晓将会丢失的数据分组。但如果发送的是经过编码的数据分组将会大不相同，因为每个数据分组均含有部分原始信息，一旦数量满足解码要求（编码系数假定线性无关），传输便可视为提前结束。从图 1（a）可以看到，假设有 3 个数据分组需要传输，采用传统 TCP 协议的发送方由于不能预知将会发生丢失的数据分组而无法轻易发送冗余分组，图 1（b）则仅需简单地将原始数据分组多进行一次编码发送即可达到掩盖丢失的目的。当然上述过程必须建立在网络状况事先确定的情况下，比如图中的丢失率就为 33%。不过新问题随之而来，即这个丢失率该如何获知，因为它将直接决定冗余分组的数量：太少起不到作用，太多又容易阻塞网络。

作为网络编码和 TCP 协议结合的原型 TCP-NC，其必然有诸多缺陷，于是文献[6]对缩短解码时延做了一些改进。不过，关于冗余该如何确定始终未在文献[3]和文献[6]中提及。为此，本文设计了一个计算冗余的算法，并且对可能存在的系数开销过大问题做了一些优化。本文将这一使用网络编码并能够动态调整冗余（dynamic redundancy）的新协议称为 TCP-NCDR。本文主要对比目前广泛使用的 Reno、Vegas[5]和 Westwood[11]版本。

2　TCP-NCDR 协议

与传统 TCP 协议不同，在引入网络编码功能后接收方需要收集一定数量的编码分组来恢复原始信息。因此，通常存放于网络编码层分组头中的编码系数[7,8]成了解码的关键信息。由于编码操作在大小为 256 的有限域上进行，故域中每个元素将占用 1 byte 的空间。因特网中传输的典型数据分组约为 1 400 byte，根据实验中的设置，尽管带宽并不大，参与编码的原始数据分组也能够很轻易地超过 60，而这一数字已经大大超过了 TCP 分组头和 IP 分组头的长度，因而势必会影响到有效载荷的使用率，并且随着带宽的增大这一数字还会进一步增加。

虽然即使是在大小为 256 的域内以随机数作为编码系数，其解码成功率也能超过 99.6%[13]，但直接这样做在数据分组较多的情况下将显得异常复杂。因此，事先选好编码系数在一定程度上可以降低解码的计算复杂度。考虑到如果参与编码的原始数据分组“维度”不同（数据分组 a、b 形成的编码分组和 a、b、c 形成的编码分组维度不同）可能会使得解码稍显容易的道理，本文提出的协议采用了预先选定的简单系数。只要收发双方深谙系数的使用规则，通信便可无障碍进行。这里使用 *cod_num* 替代常规系数，其可以“告知”接收方当前编码分组中所含原始数据分组的数量。用向量 $\boldsymbol{C}^n$ 表示第 n 次编码所使用的系数向量

$$\boldsymbol{C}^n=\left[c^n_{seen+1}\quad\cdots\quad c^n_{seen+i}\quad\cdots\quad c^n_{seen+cod\_num}\right] \tag{1}$$

这里

$$c^n_{seen+i}=\begin{cases}1,(cod\_num=1\,||\,(cod\_num>1\,\&\,\&\\ \qquad\qquad i<cod\_num))\\ 2,(cod\_num>1\,\&\,\&\,i=cod\_num)\end{cases} \tag{2}$$

假设 p_i 表示索引为 i 的原始数据分组，$\boldsymbol{P}^n$ 为第 n 次传输的原始数据分组列向量。当没有丢失发生时第 n 个发出的编码分组 p_{tx_n} 定义如下

$$p_{tx\_n}=\boldsymbol{C}^n\boldsymbol{P}^n=\sum_{j=seen+1}^{n}c^n_j p_j \tag{3}$$

上述变量 *seen* 代表具有 $\boldsymbol{p}_k + \boldsymbol{q}$ 形式的线性组合，这里 $\boldsymbol{q} = \sum_{l<k} \alpha_l \boldsymbol{p}_l$，$\alpha_l \in F_q$，并且对于所有的 $l > k$ 都成立。规定编码分组的索引等于所含原始数据分组的最大索引，易知该编码分组索引为 k。

简单来说，*seen* 能够反映数据分组当前是否可以被解码。当 *seen* 等于参与编码的原始数据分组的个数时，接收方便可以通过解码获得原始数据。当随机丢失事件发生时，由当前丢失引起的第 n 次重传定义如下

$$p_{rtx\_n} = \boldsymbol{C}^n \boldsymbol{P}^n = \sum_{j=seen+1}^{seen+loss} c_j^n p_j \tag{4}$$

这里假设 ACK 总能被发送方正确接收。显然长度为 1 byte 的 *cod_num* 能够表示 256（2^8）个原始数据分组。*loss* 的含义将在第 3 节介绍。

编码系数和编码方式如图 2 所示，其中图 2（a）表示无丢失的情况，其他代表有丢失。这里并没有将已被接收方确认的数据分组从图中删除，而是保留下来形成一个下三角矩阵方便观察，事实上可以通过高斯消元法得到符合现实情况的带状矩阵。图 2（b）反映了有一个数据分组丢失的场景，2 个或多个数据分组丢失的情况与之类似。从图中可以看出原本发送的第 4 个编码分组发生了丢失，故从接收方对第 5 个编码分组的响应可以知道，接收方已经“看见”4 个原始数据分组，所以索引为 5 的数据分组需要重发。由此

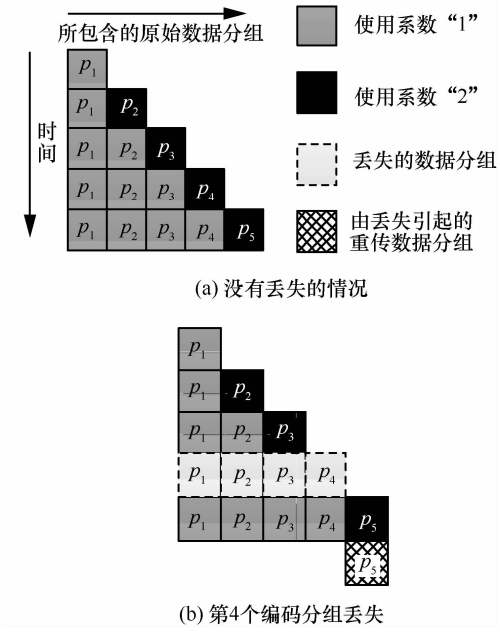


图 2　数据分组编码方式和系数使用

可见，如果 ACK 都能被正确接收，接收方就一定可以恢复出原始数据。

证明　事实上总可以找到如图 2 所示由正常传输分组 p_{tx_i} 和重传分组 p_{rtx_i} 构成的可解码块。假设每个可解码块中仅有一个重传分组 p_{rtx_i} 并且用 R_j 表示接收方收到的除去 p_{tx_j} 的系数矩阵

$$R_j = [\boldsymbol{C}^1 \cdots \boldsymbol{C}^{j-1} \boldsymbol{C}^{j+1} \cdots \boldsymbol{C}^n]$$

在上述假设下，当正常传输分组 p_{tx_j+1} 到达时，接收方已收到 j 个编码分组同时“看见”j 个原始分组，所以索引为 j+1 的编码分组需要重传。因此，接收方便能获得一个可解码块 R_j^+

$$R_j^+ = [\boldsymbol{C}^1 \cdots \boldsymbol{C}^{j-1} \boldsymbol{C}^{j+1} \cdots \boldsymbol{C}^n \boldsymbol{C}^{n+1}]'$$

这里 $\boldsymbol{C}^{n+1} = [0 \cdots c_{j+1}^{n+1} \cdots 0]'$。然后通过计算 $\prod_{k=j+1}^{n} \cdot E(k,n) R_j^+$ 的值，$E(k, n)$交换 k 行和 n 行的单位矩阵便可以获得满秩的小三角矩阵。若重传分组超过一个，也可以通过上述方法找到可解码块，因为 $[\boldsymbol{C}^m \cdots \boldsymbol{C}^l]'$ 总可以被映射到 $[\boldsymbol{C}^l \cdots \boldsymbol{C}^{l-m+1}]'$。

3　动态冗余系数更新算法

本节将详细描述 TCP-NCDR 中掩盖丢失的机制。

文献[3]首次将冗余系数 R 用来弥补信道中的随即丢失。显然，R 的值既不能太大也不能太小。因为太小将无法向传输层掩盖丢失致使重传导致吞吐量低下，但如果太大过多的冗余分组又容易阻塞网络。因此，理想的 R 值在理论上应该等于成功接收率的倒数。然而，文献[3]并没有提及如何计算并且使 R 尽可能适应网络状况。

文献[6]为了不使用冗余系数 R 而引入了一个称为 *loss* 的变量反映完成解码还需发送编码数据分组的个数，通过此变量，发送方可以较为准确地确定编码分组数量，但这种方法无法有效应对重传分组丢失的情况。

在每个编码分组的头部都有一个 *pktID* 的变量，这个变量与 TCP 头部的序列号无关。序列号被用来确认数据，并在最初的 3 次握手中由收发双方识别及交换。而 *pktID* 表示编码分组中所含原始数据分组的最大索引值。如一个编码分组含有 $\boldsymbol{p}_2$、$\boldsymbol{p}_3$ 和 $\boldsymbol{p}_4$，则 *pktID* 便为 4。传输过程中 *pktID* 每次增加

1 或者不变。当一个编码分组到达时，接收方首先检查其是否有用，然后记录下收到编码分组的数量 *pktNUM* 并计算 *loss* 值

$$loss = pktID - pktNUM \tag{5}$$

新的 *loss* 值将会附带在 ACK 中传给发送方。

注意到可解码的条件是 *pktID* 和 *pktNUM* 相等，因此若 *loss* 为负则表示没有意义，同时还必须注意是否存在线性相关的系数。

发送方收到 ACK 后开始计算 *diff_loss* 的值

$$diff\_loss = loss - loss\_old \tag{6}$$

这里 *loss_old* 是上一次 *loss* 的值（*loss* 初值为 0）。如果 *diff_loss* 小于 0，表示一个或多个重传分组已经被接收方确认，则 *R* 应减去相应的值。否则，在发送 *diff_loss* 个编码分组后 *R* 应当进行调整

$$R = R\_old + diff\_loss/n \tag{7}$$

这里 *n* 是在传输方向链路上数据分组的个数。设链路带宽为 *bw*，时延为 *ld*，*n* 表示为

$$n = bw \cdot ld/pktsize \tag{8}$$

这里，*R_old* 是前一次 *R* 的值（初值为 1）。引入 *diff/n* 为的是能起到前一次重传丢失后快速重传的作用，因为正常情况下在这期间对前一次重传响应的 ACK 应该已经到达发送方。图 3 是发送方更新冗余系数 *R* 的算法。

```
算法 更新冗余系数R
1) 收到接收方对数据的确认ACK，提取首部中loss
2) if loss==0 then
3)     R:=1，quit
4) else
5)     根据式(6)计算diff_loss;
6)     if diff_loss<0 then
7)         R=R+diff_loss;
8)         if R<1 then
9)             R:=1, quit;
10)        end if
11)    else if diff_loss==0 then
12)        quit;
13)    else
14)        发送diff_loss个编码分组，用式(7)更新R，quit;
15)    end if
16) end if
```

图 3　冗余系数 *R* 更新算法

图 4 是冗余系数 *R* 更新过程的一个例子。

正如上面所提到的，*loss* 隐含了为完成解码发送方仍需重传的编码分组数量。若 *loss* 为 0，则表示在收到上一个 ACK 时没有数据分组丢失，不需要重传。当 *loss* 大于 0，*diff_loss* 为 2 个相邻 *loss* 的差。若 *diff_loss* 大于 0，则表示有新的丢失发生，那么冗余分组必须立即发送；若 *diff_loss* 小于 0，表示 *diff_loss* 个冗余分组被接收方确认；*diff_loss* 等于 0 表示没有新的丢失同时也没有冗余分组被确认。在此过程中，很有可能发生冗余分组丢失的情况，所以每次叠加的 *diff_loss/n* 便为先于 RTO 进行重传提供了契机。

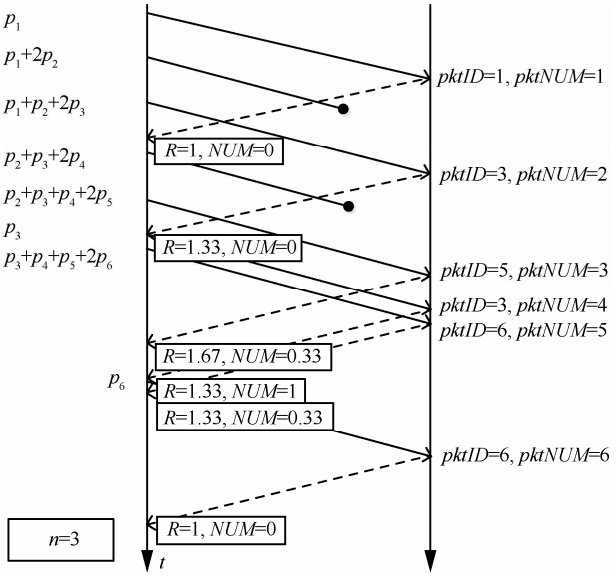


图 4　更新冗余系数 *R* 的例子

通过上述算法，*R* 在每次收到 ACK 时都会得到更新。因此，*R* 可以反映下层链路在 *RTT*/2（通常以 ms 为单位）之前的情况。这使得发送方可以更精确地控制冗余系数，相比文献[7]中的固定冗余系数是一个不小的改进。

上面提到编码分组的数量正比于带宽和 RTT。设带宽为 *bw*，信道丢失率为 *p*，则未解码的编码分组个数 *N* 在数量级上有如下正比关系

$$N \propto \min\left\{\frac{bw\cdot RTT}{pktsize}\times\left[1+\sum_{m\to\infty}\frac{(m+1)bw\cdot RTT}{pktsize}\cdot (1-p)^{\frac{(m+1)bw\cdot RTT}{pktsize}}\times p^{m+1}\right], \frac{bw(RTO+RTT/2)}{pktsize}\right\} \tag{9}$$

这里将其看做为伯努利试验。*bw*·*RTT*/*pktsize* 表示收到第一个 ACK 时，在传输方向上数据分组的个数。考虑最坏的情况，假设稳定传输状态下某一个数据分组丢失，则其将导致在收到重传数据分组以前的所有数据分组都无法解码。此次传输可以看作是丢失发生时的“第一批”数据分组，因为待

这批数据分组发送出去后第一个返回的 ACK 便会到达发送方，根据算法此时会进行一次重传。若此重传数据分组不幸丢失，则将导致“第二批”数据分组无法解码。不过此过程并不会一直持续下去，因为其同时还会受到 RTO 的限制。故 N 的大小与重传数据分组的接收情况有关，最坏的情况是重传分组全部丢失。但只要有重传分组到达接收方，就能使一部分编码分组实现解码。

4 仿真测试

4.1 TCP-NCDR 的公平性

仿真所使用的拓扑结构如图 5 所示，该结构中通信双方之间存在三段有线链路和一跳无线链路构成，仿真采用 NS2[12]软件。

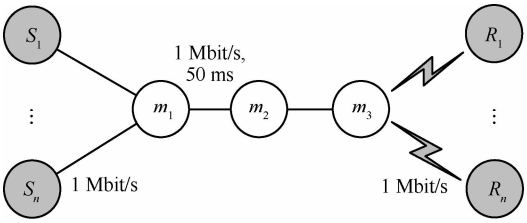


图 5 仿真拓扑

公平性是指某种相同的流应当能够共享带宽。仿真中分别建立了数量不等的（从 1 到 10 条）Reno 以及 TCP-NCDR 流。为了测试两者的公平性，使用 Jain 公平性指数[9]

$$f=\frac{\left(\sum_{i=1}^{n}x_i\right)^2}{n\left(\sum_{i=1}^{n}x_i^2\right)} \tag{10}$$

其中，n 是连接的数量，x_i 表示第 i 条连接的吞吐量。F 越接近 1 则说明公平性越好。图 6 为 Reno 和 TCP-NCDR 公平性的测试结果。可以看出，图中 TCP-NCDR 的 f 值在不同场景下均接近 1，说明其公平性较好。

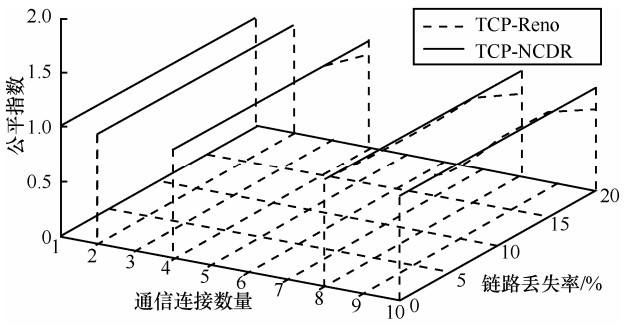


图 6 协议公平性对比

4.2 TCP-NCDR 的有效性

为了更好地凸显 TCP-NCDR 对吞吐量的提升，使用归一化吞吐量 T_n

$$T_n=\frac{T_{\mathrm{NCDR}}}{T_{\mathrm{Reno}}} \tag{11}$$

其中，T_{NCDR} 和 T_{Reno} 分别为 TCP-NCDR 和 Reno 的平均吞吐量。实验中数量不等的流（从 1~10 条）将共享网络带宽，显然，T_n 越大对吞吐量的提升就越显著。

从图 7 可以看出，TCP-NCDR 可以有效提升吞吐量，尤其在流数量比较少的情况下。随着流数量的增加，网络负载也变得更重，TCP-NCDR 对吞吐量的提升也逐渐变小。这表明 TCP-NCDR 尤其适用于负载较低的网络。

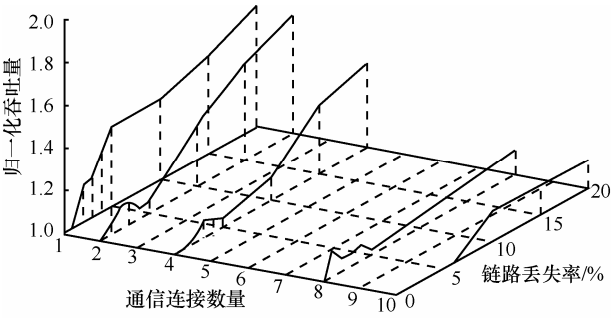


图 7 协议有效性对比

因此，下面将在负载较低的情况下（设 n=3）进一步研究 TCP-NCDR 的性能。将 S_1-R_1 这条路径作为目标流量，其他 2 条作为背景流量。整个仿真时间设为 3 000 s。背景流量在 5 s 时开始，背景流量在 10~15 s 间随机开始。实验重复 10 次，并计算出平均流量的 95%置信区间，如图 8 所示。从图中可以看出，当丢失率为 0 时，Reno 的吞吐量要高于 TCP-NCDR，这是受限于网络编码的编解码速度。但是当丢失率逐渐上升，Reno 的性能也随之下降，因为随机丢失使其拥塞窗口始终保持在相对较小的水平，而 TCP-NCDR 由于冗余的存在而仍然保持较高的吞吐量。

接下来观察在随机丢失率 20%的情况下 2 条流的完成时间。序列号和完成时间的关系如图 9 所示。

4.3 TCP-NCDR 的适应性

TCP-Westwood 不同于 Reno 的地方在于其在检测到丢失之后通过将拥塞窗口设置为匹配当前速率的大小，而非简单使用常规的乘性减小方式[11]。

因此，其相比 Reno 更适合用于无线网络。

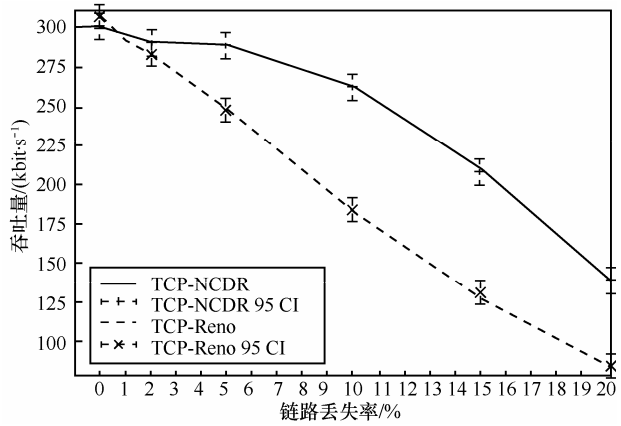


图 8　协议在低负载时的性能对比

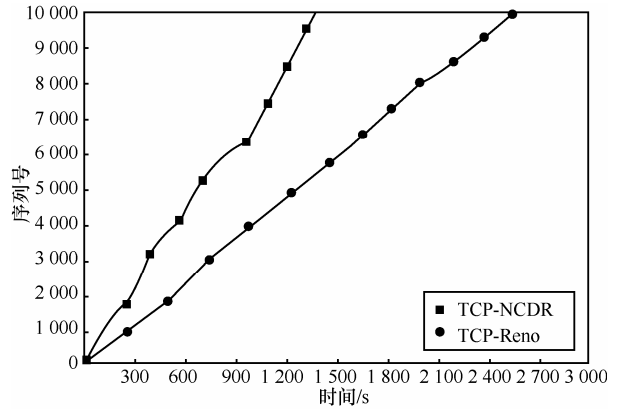


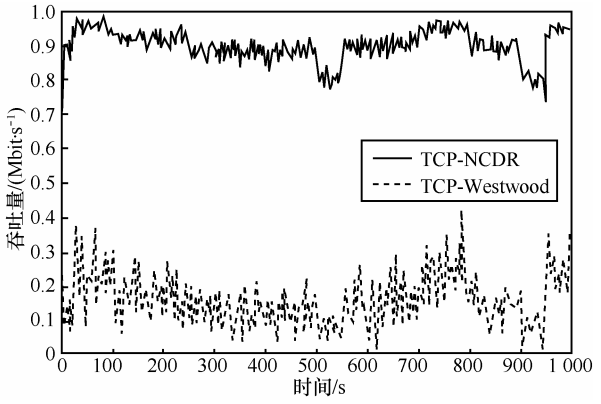
图 9　流完成时间对比

为了验证 TCP-NCDR 的适应性，人为将丢失率模仿现实环境做剧烈变化，如表 1 所示。吞吐量的值为每个 2.5 s 计算得出，仿真时间为 1 000 s，此实验中 n=1。为了清晰地显示个协议吞吐量随丢失率的变化，将图一分为二，如图 10(a)和图 10(b)所示。其中 10(a)显示了 TCP-NCDR 和 Westwood 吞吐量随丢失率的变化，可以看到 TCP-NCDR 可以对网络环境的变化做出快速反应，将吞吐量保持在较高的水平，提高信道利用率。而 Westwood 始终较低。

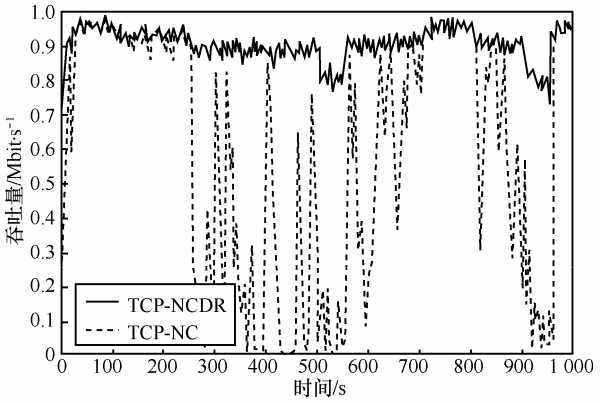
对于 TCP-NC，将 R 固定为 1.087（92%的倒数），也就是说，网络编码层可以在丢失率小于 8%时向传输层成功掩盖丢失。如图 10(b)所示，在 20~250 s 时，TCP-NC 的性能与 TCP-NCDR 相当，因为此时的 R 可满足网络需求，同样 700~800 s 和 950~100 s 也可满足需求。但在其他时候，R 由于太小而无法掩盖丢失，致使超时重传经常发生进而导致吞吐量较低。表 2 还给出了该环境下通过权重计算得出的吞吐量理论最大值。

表 1　　丢失率随时间变化关系

T/s	*Loss Rate*
1~20	10%
20~100	5%
100~250	8%
250~500	12%
500~550	20%
550~700	10%
700~800	5%
800~900	10%
900~950	20%
950~1000	5%



(a) Westwood 与 TCP-NCDR 吞吐量对比



(b) TCP-NC 与 TCP-NCDR 吞吐量对比

图 10　Westwood 和 TCP-NC 与 TCP-NCDR 吞吐量对比

表 2　　通过权重计算得出吞吐量

吞吐量	TCP-Westwood	TCP-NC	TCP-NCDR
吞吐量理论最大值 /(Mbit·s⁻¹)	0.147 1	0.58	0.843
吞吐量理论平均值 /(Mbit·s⁻¹)		0.899	

下面不再固定信道丢失率，让其在服从均值为

μ 的正态分布下随机取值。考虑到丢失率不会为负，σ 由 3σ 原理计算得出。TCP-NC 的冗系数 R 固定为 μ 的倒数。从图 11 中可以看出，由于丢失率不再固定，TCP-NC 并不能很好地掩盖随机丢失。随着 μ 的增大分布会变得更平，故 TCP-NC 的吞吐量也将下降得更快。

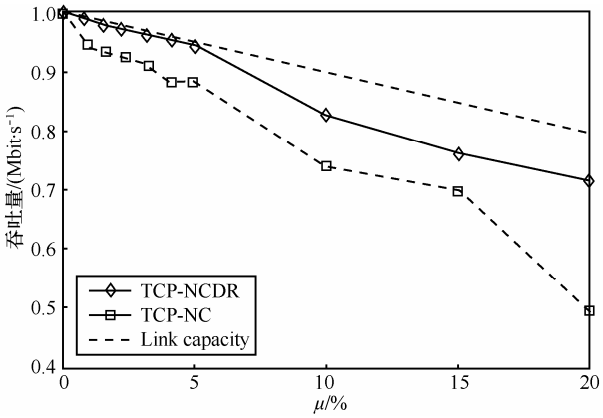


图 11 协议在丢失率服从正态分布下的性能对比

5 结束语

本文中提出了一种根据网络状况动态调整冗余系数 R 的方法，同时通过使用事先选定的系数来减少编码分组首部的开销。与固定冗余系数的 TCP-NC 相比，TCP-NCDR 所发送的冗余分组更加准确，同时能更好地向传输层掩盖丢失。考虑到随着参与编码的原始数据分组数量增多而带来的首部开销增大，使用简单系数并规定了系数使用原则。随后，仿真从公平性、有效性和适应性 3 方面对 TCP-NCDR 和其他协议做了对比测试，结果显示 TCP-NCDR 在不失公平性的前提下，对于固定丢失率和动态丢失率的情况均能较好地向传输层掩盖丢失。

下一步计划将 TCP-NCDR 的机制在 Linux 内核中修改并用实际 PC 搭建测试网络。此外，还打算将注意力放在传输对用户主观感受产生的影响，而不仅仅是客观性能数据的测量。

参考文献：

[1] AHLSWEDE R, CAI N, LI S Y, *et al*. Network information flow[J]. IEEE Trans on Information Theory, 2000, 46(4):1204-1216.

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

[3] SUNDARARAJAN J K, SHAH D, MEDARD M, *et al*. Network coding meets TCP[A]. Proceedings of IEEE INFOCOM[C]. 2009. 280-288.

[4] PAUL S, AYANOGLU E, PORTA T F L, *et al*. An asymmetric protocol for digital cellular communications[A]. Proceedings of IEEE INFOCOM[C]. 1995.1053-1062.

[5] BRAMKO L S, S. MALLEY W O, PETERSON L L. TCP Vegas: new techniques for congestion detection and avoidance[A]. Proceedings of SIGCOMM '94 Symposium[C]. 1994.24-35.

[6] CHEN J, TAN W, LIU L X. Towards zero loss for TCP in wireless networks[A]. Proceedings of IEEE IPCCC[C]. 2009.65-70.

[7] SUNDARARAJAN J K, SHAH D, MEDARD M, *et al.* Network coding meets TCP: theory and implementation[A]. Proceedings of IEEE[C]. 2011.490-512.

[8] CHOU P A, WU Y N, JAIN K. Practical network coding[A]. Proceedings of Allerton Conference on Communication, Control, and Computing[C]. 2003.

[9] JAIN R, The Art of Computer Systems Performance Analysis[M]. New York: Wiley, 1991.

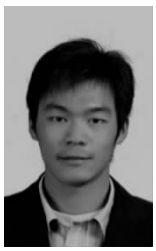
[10] NS-2 network simulator[EB/OL]. http://www.isi.edu/nsnam.

[11] UCLA computer science department high performance internet lab TCP WESTWOOD home page[EB/OL].http://www.cs.ucla.edu/NRL/hpi/tcpw.

[12] ZANELLA A, PROCISSI G, GERLA M, *et al.* TCP westwood: analytic model and performance evaluation[A]. Proceedings of IEEE GLOBECOM[C]. 2001.1703-1707.

[13] WANG D, ZHANG Q, LIU J C. Partial network coding: theory and application for continuous sensor data collection[A]. Proceedings of IEEE International Workshop on Quality of Service'06[C]. 2006. 93-101.

作者简介：



潘凯（1986-），男，江苏常州人，北京大学深圳研究生院博士生，主要研究方向为网络编码、网络协议。

李挥（1964-），男，广东汕头人，博士，北京大学深圳研究生院教授、博士生导师，主要研究方向为三网合一流媒体云计算网络视频关键技术、下一代网络体系结构、网络路由和宽带交换结构、网络编码理论及其应用、嵌入式系统开发。