国内图书分类号：TM30　　密级：公开
国际图书分类号：621.3

西南交通大学
研究生学位论文

面向无线网络的新型传输层协议设计与实现

年　　级　2014
姓　　名　董泽锋
申请学位级别　工学硕士
专　　业　通信与信息系统
指 导 教 师　陈庆春　教授/博导

二〇一七年 四月 十三日

Classified Index: TM30
U.D.C: 621.3

Southwest Jiaotong University

Master Degree Thesis

Design and Implementation of New Transport Layer Protocol for Wireless Networks

Grade: 2014

Candidate: Dong Zefeng

Academic Degree Applied for: Master of Engineering

Specialty: Communication and Information System

Supervisor: Prof. Chen Qingchun

April 13, 2017

西南交通大学
学位论文版权使用授权书

本学位论文作者完全了解学校有关保留、使用学位论文的规定，同意学校保留并向国家有关部门或机构送交论文的复印件和电子版，允许论文被查阅和借阅。本人授权西南交通大学可以将本论文的全部或部分内容编入有关数据库进行检索，可以采用影印、缩印或扫描等复印手段保存和汇编本学位论文。

本学位论文属于

1．保密☐，在　　年解密后适用本授权书

2．不保密☐，使用本授权书。

（请在以上方框内打“√”）

学位论文作者签名：　　指导老师签名：

日期：　　日期：

西南交通大学硕士学位论文主要工作（贡献）声明

本人在学位论文中所做的主要工作或贡献如下：

（略）

本人郑重声明：所呈交的学位论文，是在导师指导下独立进行研究工作所得的成果。除文中已经注明引用的内容外，本论文不包含任何其他个人或集体已经发表或撰写过的研究成果。对本文的研究做出贡献的个人和集体，均已在文中作了明确说明。本人完全了解违反上述声明所引起的一切法律责任将由本人承担。

学位论文作者签名：

日期：

摘　要

传输层协议（TCP）将丢包看做是链路出现拥塞的标志，然后启动相应的拥塞控制算法，降低传输速率。考虑到有线网络中极低的误码率，这一机制是适用的。

然而在无线网络中，由于天气、障碍物、多径干扰等各种因素，其丢包率比在有线网络中大得多，且丢包多由传输错误导致。传统 TCP 无法鉴别拥塞丢包和非拥塞丢包，不分区别地频繁启动拥塞控制机制，降低数据发送速率，将导致 TCP 无法充分利用可用带宽。网络编码的出现对于解决这一问题给出了新的思路。通过在发送端冗余编码，可以掩盖链路中出现的随机丢包，让 TCP 只会检测到拥塞丢包，从而提高网络的吞吐率。

本文分析了传统 TCP 协议在无线网络中所面临的问题，对近些年来改善无线网络中 TCP 性能的相关工作进行了梳理。介绍了网络编码原理，尤其是随机线性网络编码。对于 batch-coding 和 pipeline-coding 进行了比较，分析了其应用场景和优缺点。在以上基础上，结合 TCP 协议特点，如滑动窗口，设计实现了编码 TCP 协议，并将其移植入嵌入式设备。在真实丢包网络中，对编码 TCP 进行性能测试，与 TCP-vegas 协议进行性能对比，分析影响其性能的关键因素。在了解现有编码 TCP 协议不足的情形下，对其做了进一步改进。提出了一种新的自适应冗余算法，用于更好地应对网络状况的波动，适应不同丢包率的网络；借鉴 MAC 层的 ARQ 协议，设计了编码层的反馈重传机制，可以有效减少解码时延，减小编码端缓存队列长度，抵抗突发丢包。

关键词：网络编码；传输层协议；重传；自适应冗余；丢包

Abstract

Here is for you to write the English abstract.

Key words: Network Coding; Transport Protocol; Retransmission ; Adaptive Redundancy; Packet Loss

目　　录

第 1 章　绪论 1
1.1　论文的研究背景和意义 1
1.2　无线网络中传输层协议研究现状 2
1.3　论文主要工作及内容安排 5
第 2 章　网络编码和 TCP/NC 协议 6
2.1　网络编码理论基础 6
2.1.1　Network Coding 基本概念 6
2.1.2　网络编码的图论模型 7
2.1.3　线性网络编码 8
2.1.4　Network Coding 编码机制 9
2.2　TCP/NC 协议 15
2.2.1　ACK on degree of freedom 15
2.2.2　TCP/NC 原理 17
2.3　本章小结 19
第 3 章　TCP/NC 协议实现 20
3.1　数据包编码 20
3.1.1　有限域与扩域 20
3.1.2　编码实现 21
第 4 章　TCP/NC 的改进及仿真结果 23
第 5 章　论文工作总结与展望 24
结论 25
致谢 26
参考文献 30
附录 A 31
攻读硕士学位期间发表的论文及科研成果 32

第 1 章 绪论

本章首先介绍传输层协议在现今互联网的广泛应用，对 TCP-Reno 协议进行了粗略的描述，重点指出了 TCP 在无线网络环境下的劣势。阐述了改进 TCP 协议的必要性，对国内外学者在改进无线网络环境下 TCP 性能的研究现状进行分析和比较。最后介绍了本论文的工作及论文的相关安排。

1.1 论文的研究背景和意义

互联网为人们提供了一个快速、实时交换信息的平台。传输层协议（TCP）和网络层协议（IP）在设计之初就被紧密地联系在一起，定义了不同终端之间的通信规则，也是在互联网中数据传输方面应用最为广泛的协议族。从主干网到各种异构网络，由于简单、可靠，TCP/IP 的组合主导了今天的通信。从交互式的应用，如 Telnet 和 HTTP，到类似 FTP 协议这样的大块数据传输，TCP 已经成为了事实上的标准。TCP 最初是针对有线网络设计的。在有线网络中，很少出现比特传输错误，拥塞是导致丢包的主要原因。每个 TCP 报文都有一个序号，只有接收端按序收到正确的报文，发送端才会收到接收端回复的相应报文的确认（Acknowledge，ACK ）。另一方面，丢包或者乱序包表明传输出错。基于滑动窗口和加法增乘法减（Additive increase multiplicative decrease，AIMD）[1]，TCP 实现了流控和拥塞控制算法来解决这个问题。AIMD 体现了 TCP 协议在带宽利用和公平性方面的有效性

TCP-Reno 是目前应用最为广泛的 TCP 协议之一。它有四个传输阶段：慢启动，拥塞避免，快恢复和快重传。TCP 维护了两个变量，拥塞窗口大小（cwnd）和慢启动门限（ssthresh）。cwnd 的初始值设为最大报文段大小（MSS）。一个 TCP 连接建立起来后，首先进入慢启动阶段。对于每一个 ACK，cwnd 增长一个 MSS 大小；因此，cwnd 随着往返时延（Round-Trip Times，RTT）进行指数增长。当 cwnd 超过了 ssthresh，TCP 发送端进入拥塞避免阶段。当发送窗口内所有的报文都被确认后，cwnd 才会增长一个 MSS 大小，也就是说 cwnd 随着 RTT 线性增长。如果传输链路中的某个节点因为拥塞而导致了丢包，或者发送端收到 3 个重复 ACK，或者发送端的重传定时器超时，发送端都会减小 cwnd 的值，进而减小发送速度，以便缓解链路的拥塞状况。

随着无线技术的发展和不断增长的用户需求，IP 协议族已经扩展到无线网络环境下的应用中去。实际上，基于 IP 协议的网络有很大部分是异构网络。这意味着从一个终端用户到另一个终端用户的路径将会包含有线链路和无线链路。然而，在有线网络

中我们所信赖的 TCP 协议在无线网络中却表现不佳。这个问题来自于有线网络和无线网络不同的链路特征，以及传统 TCP 协议所设计的丢包模型。其主要表现为网络吞吐率的下降、网络资源无法充分利用和对数据传输的频繁打断。

不像光纤骨干网，无线链路使用开放的空间作为传输介质，也因此受限于许多不可控的因素，如天气、障碍物、多径干扰等。在应用场景方面，无线网络环境下的用户多为移动终端。因此，相比有线网络，无线网络有很高的误码率。同时，受限于无线信号覆盖范围，移动的终端用户经常需要在不同的 AP 之间切换，这会导致在一个通信业务时段内频繁出现断开和重连的事件。在断开期间，数据报文和 ACK 都会丢失，导致 TCP 产生重传。而断开期间的重传都会失败，这些失败的重传会让 TCP 的重传超时时间（Retransmission Time Out，RTO）出现指数的增长，拥塞窗口乘法减少（Multiplicative Decrease），直接导致 TCP 在相当长一段时间内（可以长达 50 RTTs）无法正常工作[2]。

标准 TCP，如 TCP-Reno，无法很好处理这种高误码率和频繁断开连接的情况。由于所有的丢包都被视为网络拥塞的结果，无线链路的高误码率导致的随机丢包会让标准 TCP 错误地启动拥塞控制机制，降低发送速率。如果一个 RTT 内仅仅出现一个丢包的话，TCP-Reno 协议引入的快速重传和快速恢复算法能够从零星的随机丢包中很快恢复过来。然而，噪声以及无线网络中的其他因素，导致经常出现突发的一连串的随机比特错误，也因此有很高概率在一个 RTT 内出现多个报文丢失的情况。再一次的，一个 RTT 内多次失败的重传会导致 TCP 的 RTO 快速增大，拥塞窗口乘法减小。可以看到，因为无法辨别链路的拥塞丢包和非拥塞丢包，标准 TCP 吞吐率出现了急剧下降。

通信网络在过去几十年得到了极大的发展。分组交换技术将语音和数据网络融合成了多媒体网络。不断涌现的无线应用，如高速多媒体服务，要求对现有的 TCP 协议作出改进，以适应无线网络环境下的数据传输。考虑到现有的互联网架构在 TCP/IP 协议族上，考虑到兼容性和成本，另建一套协议体系不现实。问题集中在如何对现有 TCP 协议进行改进，以适应无线网络环境下的应用。国内外有许多学者已经做了很多有益的工作，其中 Sundararajan 等人关于编码 TCP[3] 的工作引起了很大的关注。作者在 TCP 层和 IP 层之间引入一个网络编码层，提出了编码 TCP 协议（TCP/NC ），在不对现有 TCP 协议作出修改的前提下，提高 TCP 在无线网络环境下的吞吐率。本文在文献 [3] 的基础上，设计并实现了 TCP/NC 协议，并作出改进。

1.2 无线网络中传输层协议研究现状

关于标准 TCP 在无线网络中的缺点, 很多学者做了研究[2,4–7]。针对 TCP 在面对无线网络中随机丢包这一问题上的无力，目前解决的方法主要分为两类。第一类着眼于

掩盖链路中出现的丢包，这样发送端就只会检测到拥塞丢包。其背后的思想是，既然丢包是在局部链路发生的，那么应该在局部链路里就地解决。TCP 层不需要了解某段局部链路的情况。采用此种方法的协议使得整个链路看起来是一条高质量的链路，只不过有效带宽减小了而已。第二类则通过改进现有 TCP 协议的一些机制，使 TCP 可以辨别出拥塞丢包和非拥塞丢包。这样只有当出现拥塞丢包时，TCP 才会启动相应的拥塞控制算法。

对于第一类方法来说，掩盖丢包意味着不需要发送端的干涉，非拥塞丢包问题就可以解决。可以在链路层或者 TCP 层来达到这一目的。

在链路层上，两个著名的机制就是自动请求重传（Automatic Repeat Request，ARQ）和前向纠错码（Forward Error Correction，FEC），可以在局部链路上提供可靠传输[8–10]。

当丢包不是很频繁，传播时延不重要时，ARQ 很有效。只有当包重传时，才会耗费额外的带宽。然而，ARQ 可能会和 TCP 原有的机制相冲突[4]。出现丢包后，如果链路层不提供按序的报文交付，从 TCP 层下来的新的报文陆续抵达接收端，会触发接收端那边产生重复 ACK。当链路层重传这些丢失的报文时，这些重复 ACK 会抵达发送端。这又会让发送端的发送窗口值变小，而这是我们急需避免的。不仅如此，当链路层在重传报文时，TCP 的重传定时器还可能超时。文献 [11] 设计了一个跨层的算法，通过获取 TCP 层的端到端的丢包率来优化链路层的重传次数。文献 [12] 提出了一种 TCP-aware dynamic ARQ 算法。不需要修改 TCP 的状态机，利用 TCP 层采样得到的 RTT 值及目前为止重传报文的个数来动态调整链路层允许的最大重传次数和重传优先级。

FEC 则通过发送冗余信息来重构错误的报文。缺点这是当链路质量较好时，浪费了可用带宽。同时，采用 FEC 也会耗费额外的 CPU 处理时间、内存，增加时延。对于长时延的链路来说，重传代价是很高的，采用 FEC 是一个不错的方法。文献 [13] 将 FEC 应用于卫星链路上，以让 TCP 适应卫星链路大时延、高丢包的特征。

在 TCP 层掩盖丢包意味着我们需要在 TCP 层重传报文，但不能让数据发送源端发现。处在 lossy 信道的入口处的路由器部署一个 TCP agent，它保存了每一个经过它去往 lossy 信道的报文。当它看到某个报文的 ACK 时，才会丢弃该报文。文献 [14] 设计了 Indirect-TCP 协议。Indirect-TCP 在 lossy 信道入口处的路由器那里终止了原始的 TCP 连接，TCP agent 接管了原来连接的报文，然后将其送往目的站点。这种方法破坏了互联网端到端的语义，而且在 TCP agent 处需要保存大量的信息。文献 [15] 提出了 Snoop 协议，保持了端到端的语义。中间代理不会终止原来的 TCP 连接，不自己构建 ACK 报文，仅仅保存了经过它的数据报文的拷贝。来自目的站点的重复 ACK 会被丢弃，不会发往 TCP 的数据发送源端。当 agent 收到三个以上重复 ACK 或者本地的 RTO

超时，报文会在 agent 处被重传。实际上这种处理策略和链路层的 ARQ 没有本质不同，可能会和 Source 端的 TCP 原有机制冲突。

第二类方法致力于让 TCP 层分辨拥塞丢包和非拥塞丢包。文献 [16] 提出了 Explicit Loss Notification（ELN）。ELN 的基本思想是接收端的 MAC 层可以检测出错包。传统的 MAC 层在收到错包的时候会直接将其丢弃，在超过一定时间后，TCP 发送端会认为由于拥塞，这个包丢失了。如果能够告知 TCP 的发送端关于错包的信息，TCP 就可以辨别拥塞丢包和非拥塞丢包。接收端的 MAC 层在收到错包时，会将这一信息告知上面的 TCP 层，然后 TCP 层会向对方回送一个报文，告知某个包传输出错。ELN 的优点是只需要在通信的双方作出修改，而无需修改网络中的内部节点。其优点同样是缺点，我们需要修改现有的 TCP 协议和 MAC 协议，与现有体系兼容性不好。TCP 协议的一个版本 TCP-Vegas[17] 则直接改进了原有的拥塞控制协议，直接让拥塞检测和丢包解耦。换句话说，TCP-vegas 判断拥塞的方法和丢包没有关系。在 TCP-vegas 协议中，使用发送窗口值和采样得到的 RTT 值来计算网络中的报文。然后根据最近一个 RTT 的发送速度来判断网络中的报文是否过多，进而决定是否减小拥塞窗口。TCP-vegas 并没有得到广泛应用是因为其在与 TCP-Reno 共存时的劣势。当网络中的所有主机全部采用 TCP-vegas 时，整体的效果明显优于 TCP-Reno。但当网络中 TCP-vegas 与 TCP-Reno 共存时，TCP-vegas 没有办法和 TCP-Reno 公平竞争带宽。产生这种现象的原因是 TCP-Reno 使用了较具侵略性的拥塞控制算法，其传送端会不断地将数据送到网络上，直到拥塞发生。相比之下，TCP-vegas 的发送端在网络开始拥塞时就将传送速率降低，以避免拥塞发生。

RFC3168 所引入的 Explicit Congestion Notification（ECN) 机制[18] 是另一种显示告知 TCP 链路发生了拥塞的方法。在 IP 首部的 TOS 字段的第 7 和 8 比特被重新定义为 ECN 字段。网络中的路由器通过 ECN 字段来指示网络出现拥塞。ECN 机制是后期加入标准的，需要网络中的路由器支持才行。

网络编码思想创于千年更替之际，由 Ahlswede、蔡和杨在其开创性论文 [19] 中提出后迅速吸引了世界各地的研究者和实践者。网络编码（Network Coding）的出现对于传统 TCP 协议在无线网络环境下的改进给出了新的方向。文献 [20] 首次提出了将网络编码应用于真实网络的方法。通过在每个报文的头部添加相关信息，达到了去中心化的目标，形成了一个实质上的分布式系统，可以抵抗丢包，同时也能适应网络拓扑的变化。文献 [21] 提出了一种可靠传输的机制，主要针对大时延及反馈链路信道质量很差的网络。如果网络时延很大，重传报文的代价就很大；如果反馈信道质量很差，经常出现丢包，TCP 的发送端收不到 ACK，误以为数据报文出现丢失，减小传输速率。

在发送端对数据报文进行线性冗余编码，编码窗口根据收到的 ACK 来动态变化，作者采用了一个隐式 ACK 的策略，在降低对 ACK 的依赖的情况下，实现了高速的可靠传输。Sundararajan 等人在文献 [22] 中首次基于网络编码提出了“acknowledge degrees of freedom”的概念，并将其应用到链路层 ARQ 上面，减小了网络编码应用于多播协议上后编码端的编码队列长度。期望队列长度由 $\Omega\left(\frac{1}{(1-\rho)^2}\right)$ 变为 $\mathrm{O}\left(\frac{1}{1-\rho}\right)$，其中 ρ 为丢包率。再次基础上，作者又于文献 [3] 中将 Network Coding（NC）应用于 TCP/IP 协议栈，提出了 TCP/NC 协议，增强传统 TCP 协议在无线网络的吞吐率。通过在 TCP 层和 IP 层之间增加一个网络编码层，TCP/NC 可以抵抗 lossy 网络的随机丢包。此篇文章引起了极大关注，原因在于 NC 编码层可以很友好地和 TCP 层共存，不需要对现有 TCP/IP 协议栈做任何修改，适合大规模地部署。本文的工作也是基于文献 [3, 23] 设计、实现 TCP/NC 协议，并对 TCP/NC 协议进行了改进，将其移植入嵌入式开发板中，在无人机等真实丢包网络中与 TCP-vegas 进行性能对比。

1.3 论文主要工作及内容安排

本论文以 TCP/NC [3] 为基础, 设计实现 TCP/NC 协议，并将其移植入嵌入式板子中。搭建无人机测试环境，在真实丢包环境中测试 TCP/NC 的性能，分析影响其性能的关键因素。针对丢包率变化的网络，设计自适应冗余度算法。针对解码时延过大、编码队列长度过大和突发丢包，设计编码层重传算法。测试分析这两种改进的优劣，给出各自的应用场景。

本论文的章节安排如下：

第一章对标准 TCP 协议在无线网络中存在的问题进行了分析，重点分析了其拥塞控制机制和无线链路的特征之间的对立。回顾了目前一些学者在改进 TCP 协议在无线网络环境下的性能的一些工作，总结了各自的优缺点。最后对本论文的主要工作和内容安排做了介绍。

第二章首先介绍了网络编码的相关理论，对其图论模型做了描述，并在此基础上阐述了线性网络编码的相关知识。然后介绍了目前在网络编码领域应用最为广泛的 Batch Coding 编码机制，分析了其优缺点，引出了流水线编码机制，给出了编解码算法。最后介绍了 TCP/NC 协议的原理架构，重点提及了其引入的几个新概念，如“seen packet”，分析其是如何抵抗网络丢包的。第二章主要是为后面的 TCP/NC 协议的实现和改进打下基础。

第三章对

第四章对

第 2 章　网络编码和 TCP/NC 协议

本章主要介绍网络编码基础原理，重点分析了线性网络编码。详细说明了 TCP/NC 协议的设计原理。

2.1　网络编码理论基础

2000 年，R. Ahlswede 等在论文“Network Information Flow”[19] 中创造性地提出了“网络编码”新概念，首次将编码和路由有机地结合在一起，建立了一种全新的网络体系结构，不仅解决了广播路由这一信息论中的经典难题，而且使得达到组播网络容量的理论上限成为可能。网络编码建立在一个简单而广泛的概念的基础上：在包交换网络中，中间节点不仅仅是简单地路由转发接收到的数据包，而是对它们进行一些函数操作并计算、转发操作结果[24]。运用网络编码可以提高网络吞吐率、均衡网络负载和提高网络带宽利用率等。

2.1.1　Network Coding 基本概念

我们以文献 [19] 中著名的蝶形网络，即图2-1，为例来说明 Network Coding 的基本原理。

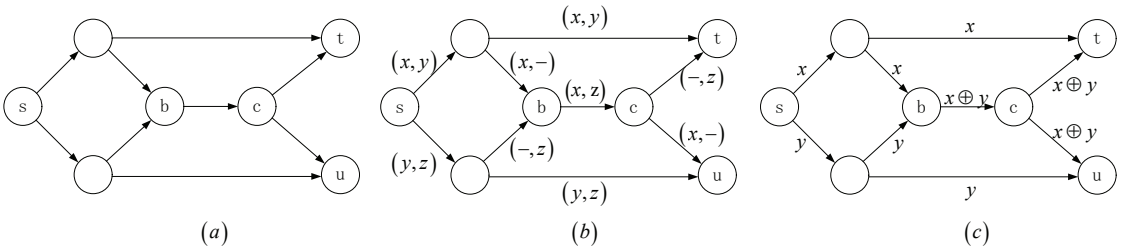

图 2-1 蝶形网络

图2-1为一个包交换模型。在该网络中，源节点 s 需要将信息多播给两个目的的节点 t 和 u。有向图的每一条有向边代表一条无差错包传输信道，每个 *channel use* 可以传输 1 个长为 m 比特的数据包。源节点希望以尽可能高的速率与两个目的节点通信。

解决该问题，传统的路由方法如图2-1(b) 为例。在第一个时隙内，源节点 s 将 x 发送给 t 和 u，将 y 发送给 u。在第二个时隙内，源节点 s 将 y 传给了 t，将 z 传给了 t 和 u。两个时隙结束后，两个目的节点 t 和 u 都收到了 x，y，z 这三个数据包。可以看到，

使用传统的路由方法，在两个时隙内可以传输 3 个不同的报文给 t 和 u，因此传统路由方法的多播吞吐量为 $\frac{3}{2} packet/channel\ use$。该多播吞吐量已被证明是使用路由方法能够实现的最大吞吐量。

然而使用图2-1(c) 所示的网络编码方法，能够实现 $2\ packet/perchannel\ use$。该方法中，第一个时隙源节点 s 仍分发两个数据包 x 和 y，与路由方法不同的是，节点 b 将对 x 和 y 进行异或运算，并转发运算结果。目的节点 t 收到数据包 x 和 $x \oplus y$，并根据它们恢复出 x 和 y。同样地，目的节点 u 也可以恢复出 x 和 y。网络编码以网络中间节点的编码操作和目的节点的解码操作为代价，提升了网络的多播吞吐量，并突破了路由方法所能实现的吞吐量上限。是否有能比图2-1(c) 更好的方案？答案是没有。一个网络的最大多播吞吐量取决于分割源节点和目的节点的最小割集[19]。图2-1(a) 的最小割集为 2，故图2-1(c) 已经是最优方案。

从上述例子中我们可以明白，要想充分利用通信网络的信息传输容量，光靠改进路由算法是不够的。在网络中，多个信息数据包可以通过多种方式有效地结合在一起，目的节点再从这些结合后的信息中恢复出原来数据包，达到提高网络的吞吐率的目的。

2.1.2 网络编码的图论模型

一个组合包网络 $N = (V, E, S, T, A)$ 包括：

(1) 有限有向无环多重图 $G = (V, E)$，其中 V 表示图 G 的顶点集合，E 表示图 G 的有向边的多重集合。

(2) 无重复源节点集合 $S \subset V$，$S = \{s_1, s_2, \cdots, s_{|S|}\}$。

(3) 无重复目的节点集合 $T \subset V$，$T = \{t_1, t_2, \cdots, t_{|T|}\}$。

(4) 有限的数据包集合 A，$|A| \geq 2$。

图 G 的顶点代表包交换网络中的通信节点，有向边代表通信节点之间的无差错传输信道。有向边 (u, v) 具有单位容量，即每条边每次只能将一个数据包（从符号集 A 中选取的一个符号）从 u 传送给点 v。如果要进行更大容量的传输，可以在 u 和 v 之间建立多条平行边，单位容量都为 1。以图2-2单源多宿网络为例，图2-2(b) 与图2-2本质上是一样的，仅仅是将所有的路径都转换为单位容量。对于顶点 $u, v \in V$，如果 $v = u$ 或者图 G 中存在一条从 u 到 v 的有向路径，就称 u 可达 v。在有向边 $(u, v) \in E$ 上进行的操作是将点 u 发出的数据包 $p \in A$ 无差错地交付给点 v。

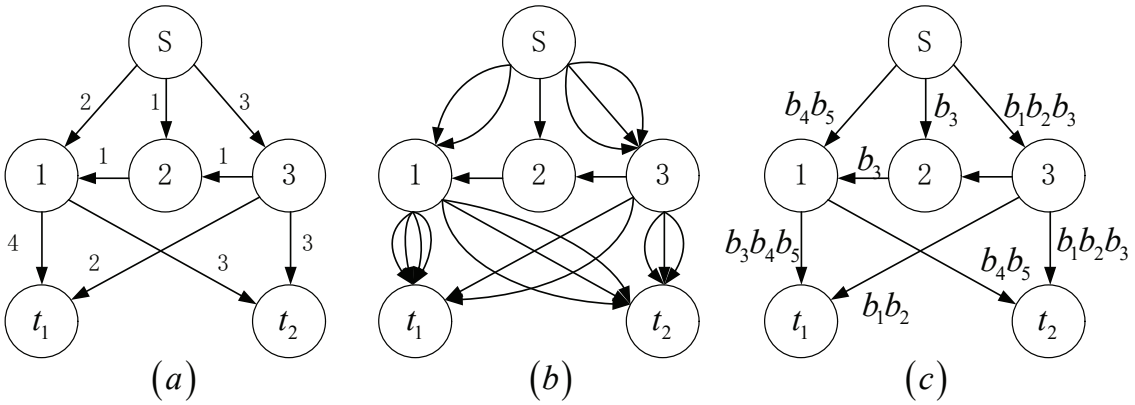

图 2-2 单源多宿网络（没有网络编码）

对于顶点 $v \in V$，我们用 $I(v)$ 表示所有进入点 v 的边的集合，用 $O(v)$ 表示所有从点 v 出发的边的集合。那么，组合包网络中的组合逻辑指的是：顶点 v 的出边集合 $O(v)$ 中的任一非空闲边上传输的数据包是来自其入边集合 $I(v)$ 中的所有非空闲数据包经过 v 的编码函数处理后得到的数据包集合。对于一个实际网络而言，在节点 v 通过某个数据包函数对入边集合的数据包进行处理，生成输出数据包之前，进入节点的数据包是需要被缓存的。

节点 v 使用的函数被称为 v 的本地编码函数。当 v 只有路由功能时，它的出边上发送的数据包可以看做是对入边上收到的数据包的复制，即图2-2(c) 所示的情况。图2-2(c) 中，网络内部节点 $1,2,3$ 仅仅通过存储转发的方法就达到了多播最大吞吐率，即源节点 s 可以同时发送 5 个数据包 $b_1b_2b_3b_4b_5$ 给 t_1 和 t_2。当 v 具有对入边数据包进行编码的功能后，如图2-1(c) 中节点 b 所示，网络多播吞吐率突破了仅仅使用路由方法的吞吐率上限。

对于组合包网络，一次 *channel use* 指的是为 E 中的每一条非空闲边都分配一个具体的数据包（取自集合 A ），也可以看做是网路中的节点对特定编码函数的一次实现。换句话说，一次 *channel use* 中的本地编码函数是固定的，且网络中的每条边最多被使用一次

2.1.3 线性网络编码

我们用列向量 $(p_1, p_2, \ldots, p_r)^T$ 表示进入源节点 s 的数据包 $\mathrm{X}_{\mathrm{I}(s)}$。每个数据包 p_i 都是 $\mathbb{F}_q$ 上的标量。对于更一般的多个数据包的情况，p_i 可以看做是 $\mathbb{F}_q$ 上长度为 m 的矢量。这样，我们可以用一个定义在 $\mathbb{F}_q$ 上的 $r \times m$ 矩阵来表示 $\mathrm{X}_{\mathrm{I}(s)}$，该矩阵的第 i 行是 p_i。

本地编码函数是 $\mathbb{F}_q$ 上的线性函数，即任意中间节点 v 输出的数据包列向量 $\mathrm{X}_{\mathrm{O}(v)}$

与其接收到的数据包列向量 $\mathrm{X}_{\mathrm{I}(v)}$ 之间的关系可以用以下线性方程组表示：

$$\mathrm{X}_{\mathrm{O}(v)} = \mathrm{L}_v \mathrm{X}_{\mathrm{I}_{(v)}} \tag{2–1}$$

其中，L_v 是定义在 $\mathbb{F}_q$ 上的系数矩阵，也称为节点 v 的本地转移矩阵。换句话说，v 输出的每个数据包（$\mathrm{X}_{\mathrm{O}(v)}$ 的一个分量）都可以看做是进入 v 的多个数据包在 $\mathbb{F}_q$ 上的线性组合。L_v 的每一行都对应一条边 $e \in \mathrm{O}(v)$，称为边 e 的本地编码向量。

考虑到网络中只允许线性操作，因此任意边上传输的数据包都是原数据包 $p_1, p_2, \ldots, p_r$ 的线性组合。也就是 $\forall v \in V$，有

$$\mathrm{X}_{\mathrm{I}(v)} = G_v \begin{bmatrix} p_1 \\ \vdots \\ p_r \end{bmatrix} \tag{2–2}$$

其中，G_v 是定义在 $\mathbb{F}_q$ 上的系数矩阵，也称为 v 的全局转移矩阵。G_v 的每一行都对应一条边 $e \in \mathrm{I}_{(v)}$，称为边 e 的全局编码向量。

图2-3为标注了本地转移矩阵的蝶形网络。我们可以得到在目的节点 t 和 u 的全局转移矩阵为：

$$\begin{aligned} G_t &= \begin{bmatrix} \alpha_1\alpha_5 & \alpha_2\alpha_5 \\ \alpha_1\alpha_6\alpha_9\alpha_{11} + \alpha_3\alpha_7\alpha_{10}\alpha_{11} & \alpha_2\alpha_6\alpha_9\alpha_{11} + \alpha_4\alpha_7\alpha_{10}\alpha_{11} \end{bmatrix} \\ G_u &= \begin{bmatrix} \alpha_1\alpha_6\alpha_9\alpha_{12} + \alpha_3\alpha_7\alpha_{10}\alpha_{12} & \alpha_2\alpha_6\alpha_9\alpha_{12} + \alpha_4\alpha_7\alpha_{10}\alpha_{12} \\ \alpha_3\alpha_8 & \alpha_4\alpha_8 \end{bmatrix} \end{aligned} \tag{2–3}$$

对于目的节点 t 来说，当且仅当 $|\mathrm{I}(t)| \times r$ 阶全局转移矩阵 G_t 的秩为 r 时，t 才能恢复出 $(p_1, p_2, \ldots, p_r)$ 中的所有元素。因为只有 G_t 满秩时，才存在满足 ${G_t}^{-1} G_t = \mathrm{I}_r$ 的左逆矩阵，其中 I_r 是 $r \times r$ 阶单位矩阵。只需 G_t 满秩，而不要求 G_t 是可求逆的方阵。目的节点 t 可以通过计算 ${G_t}^{-1}\mathrm{X}_{\mathrm{I}(t)}$ 得到 $(p_1, p_2, \ldots, p_r)^T$。

2.1.4 Network Coding 编码机制

由于多径干扰、障碍物阻挡等原因，无线信道传输经常容易出错。链路层上比较著名的解决方法有前向纠错码（Forward Error Correction，FEC ）和自动重传请求（Automatic Repeat Quest，ARQ ）[25]。当错误是随机发生时，FEC 可以有效地纠正报文的错误部分。FEC 的优势是它不需要重传错误报文，也不会和 TCP 的原有机制产生冲突。

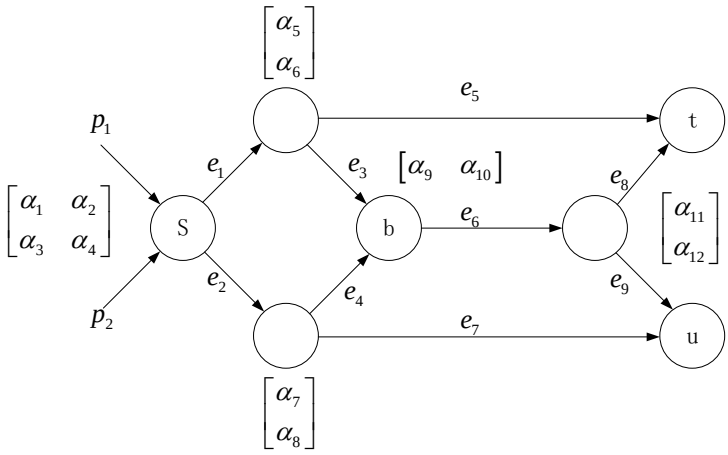

图 2-3 标注了本地转移矩阵的蝶形网络

FEC 的缺点则是当链路质量很好地时候，会浪费带宽，而且会增加接收端 CPU 的计算和内存负担。同时，对于由于链路短暂断开导致的丢包，FEC 也无能为力。当链路的丢包不是很频繁，还有传播时延不是很重要的时候，ARQ 就比较有优势。然而，ARQ 可能会与 TCP 的原有机制冲突[25]。为了有效控制丢包，有关学者将目光放在了基于 FEC 的编码机制上面，如纠删码（Erasure Coding ）[26] 和网络编码[19,20,27]。

网络中的数据包一般被看做是字节流。若干个数据包组成一个字节流，这个字节流被切分为 k 段，称为一个“*generation*”[20]。对于每一个 *generation*，编码端使用随机线性编码产生 n 个经过编码的数据包。在目的节点，n 个编码报文的任意 k 个独立数据包就足够恢复出原始数据包。换句话说，我们可以容忍网络中丢失 $n-k$ 个报文。纠删码和网络编码的区别是前者只在源节点进行编码，而后者则在网络中的各个节点进行编码。实际上，纠删码可以看做是网络编码的特例。

在恶劣的网络环境下，端到端的纠删码不足以实现可靠报文传输[28]。一些学者将网络编码应用于这些极具挑战性的网络中。和纠删码不同，网路的中间节点也会参与到编码中。在一些网络编码的实际应用中，“Batch Network Coding”是应用最为广泛的一种[20,27,29–32]。先介绍 Batch Coding 的相关原理。表3-1是后面会用到的一些术语。

Batch Coding

假定在源节点有一个应用在不断产生同样大小的数据包$p_1, p_2, p_3, \ldots,$。*generation* 的大小为 k。第 i^{th} 个 *generation* 产生的一个编码包可以被表示为：

$$c = \sum_{j=1}^{k} e_j p_{i\times k+j} \tag{2–4}$$

表 2-1 相关术语

名词	定义
Batch Coding	每一个编码包会对同一个 *generation* 里面的所有原始报文进行编码，只有 *generation* 满秩之后才会进行编码或者解码
流水线编码	每当一个数据报文来之后，就会产生一个编码包，目的节点会逐步解出原始报文
generation	数据包的集合，作为编码和解码的整体
编码向量	反应组成某个编码数据包中各个原始数据包的系数
秩	线性独立的编码包的个数
编码冗余	*generation* 的大小去除一个 *generation* 生成的编码报文的个数得到的商
“Innovative”报文	一个可以增加秩的报文
Delay	目的站点的应用收到包的时间和包从源节点的应用产生的时间的差

这里 e_k 是某个有限域 $\mathbb{F}$ 中的一个元素，$i \times k$ 是在第 i 个 *generation* 之前传输的的包的个数。每个数据包都被看做是在有限域 $\mathbb{F}$ 上的一个向量，所有操作都是在有限域 $\mathbb{F}$ 上进行。令 r 表示编码冗余度，$r \geq 1$。对于每一个 *generation*，源节点会产生 $k \times r$ 个编码包。在目的节点，收到 k 个线性独立的包后，目的节点通过高斯消元解公式2–5，就可以恢复出原来 k 个数据包。

$$\begin{bmatrix} c_1 \\ \vdots \\ c_k \end{bmatrix} = \begin{bmatrix} e_1^{(1)} & \cdots & e_k^{(1)} \\ \vdots & \ddots & \vdots \\ e_1^{(k)} & \cdots & e_k^{(k)} \end{bmatrix} \begin{bmatrix} p_1 \\ \vdots \\ p_k \end{bmatrix} \tag{2–5}$$

由于每个编码数据包都是同一个 *generation* 的数据包的线性组合，在源节点就引入了一个延时。换句话说，同一个 *generation* 里面前面的报文需要等后面的报文，直到这个 *generation* 满秩后，才可进行编码。在目的节点处，同一个 *generation* 的报文，要么都没解码，要么一个都解不出。

为了刻画 Batch Coding 整个的时延，我们定义“delay”如下：接收端的应用收到报文的时间和源节点的应用产生报文的时间的差值。t_i 表示源节点产生 **data$_i$** 的时间，t_{ci} 表示第 i 个编码包被发出去的时间。假定报文传输过程中的处理时延、传输时延、传播时延和排队时延都是常数，d_n 表示网络中出现的时延的总和，d_c 表示接收端的解码过程的时延，也为一个常数。假定一个 *generation* 还有 k 个数据包，那么在无损信道中，

传送数据包 $\mathbf{data_i}$ 的时延为：

$$t_{c\lceil i/k\rceil * k} - t_i + d_n + d_c \tag{2–6}$$

例如，在图2-4中，传送 $data_1$ 的时延为 $t_{c4} - t_1 + d_n + d_c$。这里 *generation* 的大小为 4，

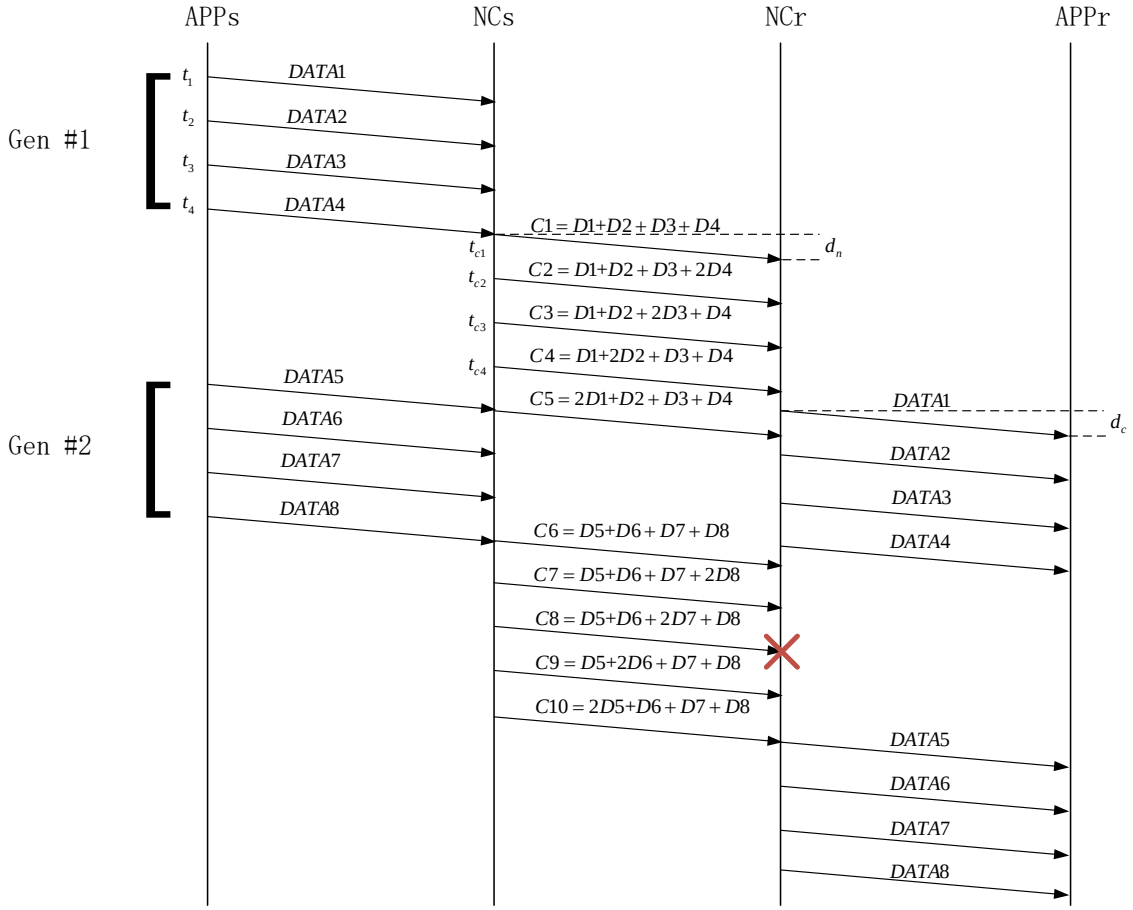

图 2-4 Batch Coding 例子

编码冗余为 1.25. 因此对于每个 *generation* 来说，都会发送 $k * r = 5$ 个编码包。

如果链路中出现丢包，接收端只需要收到 4 个线性独立的编码包就可以解出对应的 *generation* 的所有原始数据报文。图2-4中的“generation # 2”描述了一个丢包被恢复，解码端成功解码的例子。然而如果冗余度不够补偿丢失的话，那么这个 *generation* 就没有报文被解出来。如图2-5，丢了 $C4$ 和 $C5$，接收端无法解码。

“Batch Network Coding”指的是这样一类编码：源节点和网络中的中间节点在同一个 *generation* 上进行编码，目的节点只有在收到足够多的编码报文后，才能一次性解出这一个“*generation*”中的所有原始报文。

Batch Network Coding 有两个显著的缺点：首先，它引进了和 *generation* 大小成比

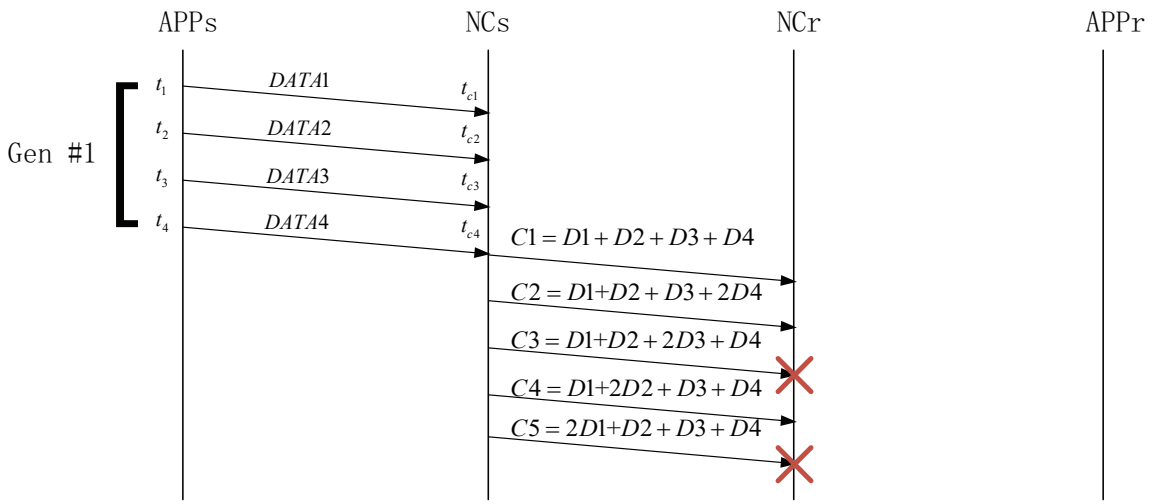

图 2-5 Batch Coding 例子

例的编解码时延，对于实时性要求比较高的应用来说，会限制 *generation* 的大小，也就会降低编码的弹性；其次，只有当接收到足够多的线性独立的编码报文后，才可以解出一个 *generation* 的所有报文，否则，整个 *generation* 会被丢弃。

文献 [33] 的作者针对 Batch Network Coding 的弱点提出了流水线网络编码（Pipeline Network Coding ）。

流水线编码

流水线编码旨在减少编解码时延，进而提高吞吐率。和 Batch Coding 不一样的是，流水线编码不需要等集满一个 *generation* 大小的报文后才进行编码，与公式2–4稍微不同，流水线编码的编码函数为：

$$c = \sum_{j=1}^{m} e_j p_{i \times k + j} \tag{2–7}$$

其中，m 为目前在编码缓存中的数据包个数。换句话说，如果收到一个原始报文，发送端基于目前收到的数据包立马执行一个编码函数。如果所有的编码包都依次正确地送到目的节点，那么目的节点可以构建如下的下三角矩阵：

$$\begin{bmatrix} c_1 \\ \vdots \\ c_k \end{bmatrix} = \begin{bmatrix} e_1^{(1)} & 0 & \cdots & 0 \\ e_1^{(2)} & e_2^{(2)} & & \vdots \\ \vdots & & \ddots & 0 \\ e_1^{(k)} & e_2^{(2)} & \cdots & e_k^{(k)} \end{bmatrix} \begin{bmatrix} p_1 \\ \vdots \\ p_k \end{bmatrix} \tag{2–8}$$

上述方程可以逐步求解，而无需等待整个的 *generation* 报文都集齐。例如，收到 c_1，目的节点可以解码出 p_1，以此类推。编码冗余的设计与 Batch Coding 略有不同。令 r 为编码冗余度，$r \geq 1$。对于每个数据包来说，$r-\lfloor r \rfloor$ 的概率产生 $\lfloor r \rfloor+1$ 个编码包，$1-(r-\lfloor r \rfloor)$ 的概率产生 r 个编码包。

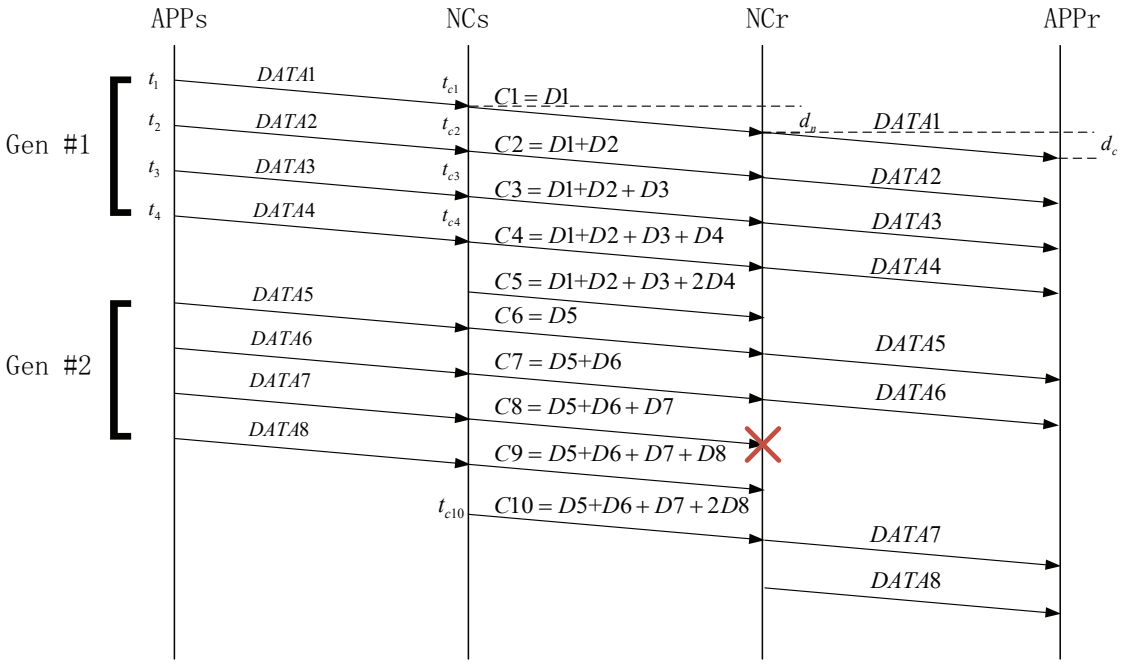

图 2-6 流水线编码

与前面分析 Batch Coding 时一样，我们让 t_i 表示源节点产生数据 **data**$_i$ 的时间，t_{ci} 表示第 i 个编码数据包被发出去的时间。那么传送数据包 **data**$_i$ 的时延为 $t_{ci}-t_i+d_n+d_c$。在大多数情况下，这比公式2–6所表示的时延小多了。

图2-6展示了一个流水线编码的例子。这里 *generation* 的大小为 4，冗余度为 1.25。当有一个原始数据包产生时，立马就编码得到一个编码包。同样地，在解码端，每当有一个“innovative”报文到来的时候，就可以立马解出一个原始报文。

对于丢包的情况，图2-6中 *Gen # 2* 中的 $C8$ 丢失，目的节点在收到 $C9$ 后，无法解出 $DATA7$。这时候需要将 undecoded packets 存起来，等到冗余报文过来才可以解出原始报文，如图2-6中的 $C10$。Batch Coding 的编解码流程如 Algorithm1和 Algorithm2。

采用流水线编码的优点主要有：

(1) 更低的解码时延

(2) 更高的吞吐率

(3) 对上层来说是透明的（UDP，TCP，或者其他协议）

本文所实现的 TCP/NC 协议也是基于流水线网络编码。

Algorithm 1: Encoding Function

Input: coding redundancy r，data packet *data* from up-layer，$generation\_buffer\,[0,\ldots,m,\ldots]$，number of packets currently in *generation buffer* $m-1$

Output: NONE

```
1  Store data to generation buffer;
2  i ← 0;
3  while i < num do
4      if (num − i < 1 ∧ rand()%100 > num − i) then
5          break;
6      Generate Coding Vector;
7      for j = 0; j ≤ m − 1; j + + do
8          if generation_buffer[j] == NULL then
9              coding_vector[j] = 0
10     Encode packet;
11     Send Coded Packet to MAC layer;
12     i ← i + 1;
```

Algorithm 2: Decoding Function

Input: data packet *data*，$generation\_buffer\,[0,\ldots,m,\ldots]$

Output: NONE

```
1  if (not_innovative(data)) then
2      Free_data(data);
3      return;
4  Store data to generation buffer;
5  Gaussian_Elimination(generation);
6  if (decodable packets cause no reordering) then
7      Deliver it to Transport Layer;
```

2.2 TCP/NC 协议

在详细阐述 TCP/NC 协议原理之前，先引入 *ACK on degree of freedom* 这个概念。

2.2.1 ACK on degree of freedom

现有的很多可靠传输协议都有 ACK 这个机制，如 TCP，ARQ 等。接收方通过回复发送方 ACK 来让发送方了解自己收到了哪些数据。如果在没有应用编码的网络中，那么接收端在收到原始报文后就可以回给发送方 ACK；如果在应用了网络编码的网络

中，如文献 [34]，接收端需要在收到若干个数据包后，对整体进行解码恢复原始数据包，才会回复 ACK，亦如第二章介绍的 Batch Coding，其缺点是传送数据包的时延较长。针对 Batch Coding 的缺点，文献 [33] 提出了流水线编码（Pipeline Coding ），但是在丢包网络中也会存在 ACK 延迟的问题。图2-6中的 $DATA7$ 和 $DATA8$ 需要等到收到 $C10$ 后才可以解出，进而回给发送端 ACK。ACK 的延迟不仅会影响吞吐率，还会影响发送端的编码队列的长度。直觉来说，编码端只有在收到某个数据包的 ACK 后，才会将其从编码缓存中删除。例如，源节点有 n 个数据包（每个数据包看做是有限域 $\mathbb{F}$ 上的向量），分别是 p_1，p_2，$\ldots$，p_n。经过编码，依次发出去 n 个编码包，分别是 p_1，(p_1+p_2)，(p_2+p_3)，$\ldots$，$(p_{n-1}+p_n)$。由于网路质量差，第一个包 p_1 丢了，接收端只收到 (p_1+p_2)，(p_2+p_3)，$\ldots$，$(p_{n-1}+p_n)$。接收端尽管收到了 $n-1$ 个编码包，但一个也无法解出来，也就无法给源节点确认 ACK。可靠传输要求发送端在没收到接收端确认 ACK 时必须把 p_1，p_2，$\ldots$，p_n 都放入缓存队列。我们可以看到这个队列此时的长度长达 n。

针对以上问题，文献 [22] 首次提出了“ACK on degree of freedom”的概念。在介绍“ACK on degree of freedom”之前，我们先引入一个定义。

定义 2.1 (Seeing a packet) 如果一个节点根据现有的信息可以计算出如$(p+q)$ 形式的线性组合，那么我们就说这个节点“***see packet*** p”。其中q 本身就是只包含序号比 p 大的报文的线性组合。解码出某个报文也算作是“*seeing a packet*”，此时$q=0$。

直觉告诉我们，如果目的节点已经 *see packet* p，那么发送端那边在产生编码包的时候，就不必将数据包 p 再囊括进去了。换句话说，发送端可以将数据包 p 从编码缓存中移除。接收端如果后面接收到足够的信息，将 q 解出来，那么 p 自然就解出来了。因此，接收端如果已经 *see packet* p，那么其可以直接向发送端回复 ACK，表示自己已经收到了报文 p，通知发送端那边将报文 p 从编码缓存中移除。

表2-2展示的就是 *ACK on degree of freedom* 的一个例子。在时隙 1 时，发送端发送了报文 p_1，p_1 在链路中丢失了，目的节点 A 没有收到；在时隙 2 时，由于 A 没有发送对报文 p_1 的 ACK，所以源节点发送缓存为 p_1 和 p_2，接着发送端对 p_1 和 p_2 进行异或，得到编码包，发出去。目的节点收到了编码包，虽然无法解出 p_1 和 p_2，但根据定义2.1，A 看到了 p_1。因此，A 回复源节点一个 ACK，让源节点将 p_1 从编码缓存中移除。我们可以看到，在时隙 3 时，源节点的发送缓存只有 p_2 和 p_3，和前面一样，源节点发送编码包 $p_2 \oplus p_3$。这一次 A 正确地收到了，A 看到了 p_1 和 p_2，但还是无法解出它们。我们看到，一直到时隙 5，A 都没有解出一个报文，但看到的报文（seen packet ）有 p_1，p_2，p_3。源节点在时隙 6 发送了 p_4，导致 p_5 被看到，然后在时隙 7 发送了 p_5，让 A 解出了

所有报文。

表 2-2 *ACK on degree of freedom* 例子

Time	Sender's queue	Transmitted packet	Channel state	Destination Node A	
				Decoded	Seen but not decoded
1	p_1	p_1	$\nrightarrow A$		
2	p_1, p_2	$p_1 \oplus p_2$	$\rightarrow A$		p_1
3	p_2, p_3	$p_2 \oplus p_3$	$\rightarrow A$		p_1, p_2
4	p_3, p_4	$p_3 \oplus p_4$	$\nrightarrow A$		p_1, p_2
5	p_3, p_4, p_5	$p_3 \oplus p_4 \oplus p_5$	$\rightarrow A$		p_1, p_2, p_3
6	p_4	p_4	$\rightarrow A$	p_4	p_1, p_2, p_3, p_5
7	p_5	p_5	$\rightarrow A$	p_1, p_2, p_3, p_4, p_5	

总结一下，*ACK on degree of freedom* 就是说我不再对原始数据包进行 ACK 确认，而是确认自由度。只要接收端看到了某个报文（see a packet)，即使没有解出来，依然可以恢复发送端对 seen packet 的确认。

2.2.2 TCP/NC 原理

网络编码自出现以来就作为一种重要的改善网络性能，尤其是无线网络性能的工具。它的主要优势在于能够跨时间、跨流地将数据混合起来。这可以让在 lossy 信道中的数据传输更为健壮和高效。尽管如此，我们还是很少看见应用网络编码的实际例子，一个主要原因就是如何将网络编码很自然地加入到现有的网络体系中，而不与其冲突。文献 [35] 在对网络编码与现有协议结合中可能出现的问题作出过分析，具体到 TCP 来说，就是寻找一种能够与现有的 TCP 的滑动窗口兼容的编码机制。

Sundararajan 等人在“Network Coding Meets TCP”[3] 一文中提出编码 TCP 协议（TCP/NC），用于提高标准 TCP 协议在无线网络环境下的性能。其通过在 TCP 层和 IP 层之间添加一个网络编码层来掩盖 lossy 信道中出现的随机丢包。与以往一些学者在改进 TCP 在无线网络下性能的工作不同的是，TCP/NC 不需要对原有的 OSI 协议栈做出修改。TCP/NC 可以很优雅地与标准 TCP 协议，IP 协议，链路层的协议一起工作，不会与原有协议的机制冲突，适合大规模地部署在现有互联网体系中。

TCP 可靠传输的背后思想是使用 Acknowledgements 对按序到来的数据包进行确认，同时发送端使用 ACK 来进行拥塞控制。对于使用网络编码来说，这个机制需要做点改变。其原因在于使用网络编码后，接收端收到的不再是原始数据包，而是原始数据包的线性组合（这里仅针对线性网络编码）。一旦收到足够多的线性组合，就可以解码得到原始数据包。标准 TCP 中“按序到达的报文”这个概念缺失了。现有的 ACK 机制不允许我们确认还未解码的数据包。上一小节所讲到的 *ACK on degree of freedom* 概念值得借鉴。绝大多数情况，接收端收到的线性组合报文都会给接收端带来新信息（在

有限域 $\mathbb{F}$ 下有可能出现非线性独立的编码包），也就是自由度（将编码数据包看做是向量）。TCP/NC 对于收到的编码包产生的自由度回复 ACK，表示自己看到了某个包 p[1]，而不管这个新来的包是否让接收端解出了一个包。

图2-7为 TCP/NC 的系统框架。在 TCP 层和 IP 层之间引入了一个 NC 编码层。NC 层缓存从 TCP 层下来的数据包，并放入编码缓存，然后生成处于编码缓存中所有数据包的线性组合并发出去。当发送端的 NC 层收到接收端的 ACK 后，才会将对应的数据包从编码缓存中删除，并将相应 ACK 传给上层 TCP。编码包在到达接收端的 NC 层之后会被放入解码缓存，并进行高斯消元等解码操作。如若有新的包被看到，则回复发送端相应的 ACK；如若有数据包被解出，将其交给上层 TCP。

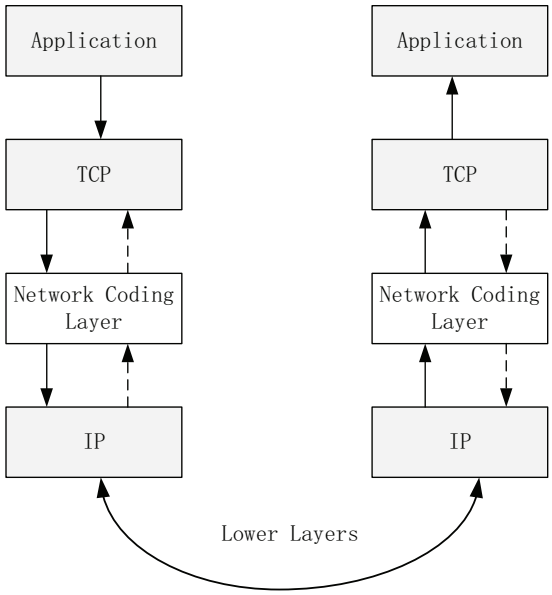

图 2-7 TCP/NC 协议架构

TCP/NC 能够抵抗链路丢包的关键就是发送端进行冗余编码。通过“seen”这个概念，将链路丢包这个问题往后推。如果后期的冗余包能够补偿丢失的报文，那么 TCP 层就不会发现丢包。如图2-8所示，虽然 $SEQ = 2$ 和 $SEQ = 3$ 这两个报文丢失，但发送端依然可以收到对 p_2 和 p_3 的 ACK。只要后期传输过程中的冗余包足够补偿掉 $SEQ = 2$ 和 $SEQ = 3$ 的丢失，那么上层 TCP 就根本发现不了丢包。可以看见，TCP/NC 可以一定程度掩盖链路的丢包。

[1]seen packet 的概念见2.2.1小节的定义2.1

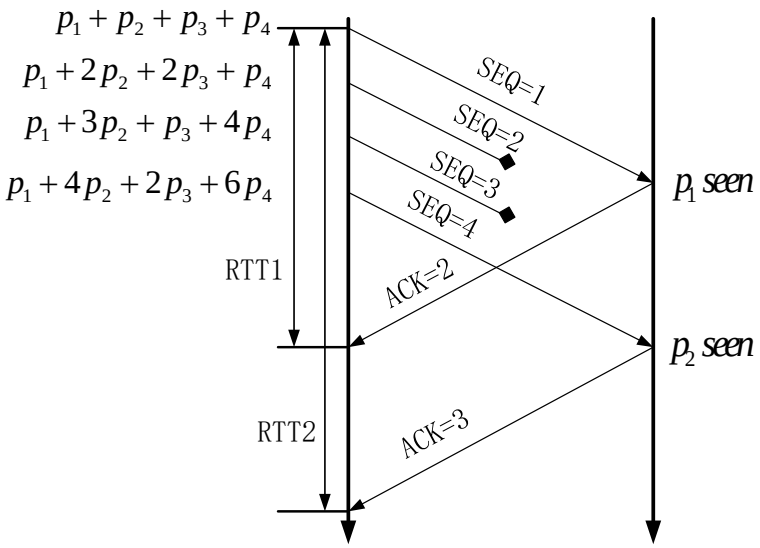

图 2-8 编码和 ACK 示例

2.3 本章小结

本章首先介绍了网络编码的相关理论，对其图论模型做了描述，并在此基础上阐述了线性网络编码的相关知识。然后介绍了目前在网络编码领域应用最为广泛的 Batch Coding 编码机制，分析了其优缺点，引出了流水线编码机制，给出了编解码算法。最后介绍了 TCP/NC 协议的原理架构，重点提及了其引入的几个新概念，如“seen packet”，分析其是如何抵抗网络丢包的。本章主要是为后面的 TCP/NC 协议的实现和改进打下基础。

第 3 章 TCP/NC 协议实现

本章主要讲 TCP/NC 的具体实现，在无线 lossy 信道下测试 TCP/NC 的性能，并与标准的 TCP-vegas 进行分析对比。

3.1 数据包编码

在阐述数据包编码之前，先简单介绍近世代数的相关概念。

3.1.1 有限域与扩域

定义 3.1 (阿贝尔群) 对于一个非空元素集合 G 以及定义在 G 上的一种运算“$*$”（这里的 $*$ 泛指一种代数运算，如 $+$，$-$，$\times$，$\div$，模 m 加 $\oplus$，模 m 乘 $\odot$ 等）。若满足以下四个条件：

(1) 封闭性，即 $\forall a,b \in G$，$\exists (a*b)=c \in G$。

(2) 结合性，即 $\forall a,b \in G$，$\exists a*(b*c)=(a*b)*c$。

(3) 存在唯一一个单位元 e，即 $\forall a \in G$，$\exists a*e=e*a=a$。

(4) G 中的每个元素各自存在唯一的逆元，即 $\forall a \in G$，$\exists a^{-1} \in G$，使得 $a*a^{-1}=a^{-1}*a=e$。这里 a^{-1} 泛指逆元。

(5) $\forall a,b \in G$，$\exists a*b=b*a$

则称这样的代数系统为阿贝尔群，记做 $(G,*)$[36]。

定义 3.2 (有限域) 非空集合 F 含有有限个元素，其中定义了加和乘两种运算，且满足

(1) F 关于加法构成阿贝尔群，加法恒等元记为 *0*

(2) F 中所有非零元素对乘法构成阿贝尔群，乘法恒等元记为 *1*

(3) F 加法和乘法满足分配律

则 F 与这两种运算构成有限域[36]。

表 3-1 本原多项式 $P(x)=x^4+x+1$ 的根生成的循环群

各次幂 α_k	α 的多项式	多项式系数 m 重
α^0	1	(0001)
α^1	α	(0010)
α^2	α^2	(0100)
α^3	α^3	(1000)
α^4	$\alpha+1$	(0011)
α^5	$\alpha^2+\alpha$	(0110)
α^6	$\alpha^3+\alpha^2$	(1100)
α^7	$\alpha^3+\alpha+1$	(1011)
α^8	α^2+1	(0101)
α^9	$\alpha^3+\alpha$	(1010)
α^{10}	$\alpha^2+\alpha+1$	(0111)
α^{11}	$\alpha^3+\alpha^2+\alpha$	(1110)
α^{12}	$\alpha^3+\alpha^2+\alpha+1$	(1111)
α^{13}	$\alpha^3+\alpha^2+1$	(1101)
α^{14}	α^3+1	(1001)

有限域提供了一个有限集，在该有限集上明确地定义且有效地实现了加法、减法、乘法及除法运算（减法可以转换为加法，除法可以转换为乘法），并允许系统使用矩阵、行列式、高斯消元等线性代数中常见的运算工具来解决该域上的联立线性方程组问题。

我们在讲到 $GF(q)$ 时，经常顺带会讲到 $GF(q^m)$，表示 $GF(q)$ 的扩域。扩域里至少存在一个本原元 α，它的各次幂 α^0，α^1，α^2，…，α^{q^m-2} 构成了扩域 $GF(q^m)$ 的全部非零域元素。表3-1展示的就是以本原多项式 $P(x)=x^4+x+1$ 对应的本原元的各次幂生成的 $GF(2^4)$ 的全部非零元素。

3.1.2 编码实现

由于从 TCP 层下来的数据包都是字节流，如何抽象出我们在第2.1.3小节中谈到的报文概念呢？进一步地，我们如何对这些报文进行各种操作，如加减乘除呢？

将数据包分解成字节，先对字节进行加减乘除的操作，拼接得到的结果是可行的。这解决了以何种角度刻画数据包的问题。对于数据包的运算，如果是在实数域上进行各种代数运算，则会涉及到进位的问题，不好处理。例如，p_1 和 p_2 这两个数据包的二进制形式为 $p_1=(1010\ 1010)$ 和 $p_2=(1111\ 0000)$。在实数域上计算得到 $p_3=p_1+p_2$，则 $p_3=(1\ 1001\ 1010)$，p_1 和 p_2 都是 8 个比特，但是 p_3 却是 9 比特，出现了进位，导致我们的处理很麻烦。

上一小节中提到的有限域和扩域是解决数据包运算问题的关键。有限域是一种代

数结构，也存在着和实数域中一样的加减乘除等运算，并且各个元素进行加减乘除得到的结果也在有限域中。具体实现过程如下。

首先，确定有限域的大小。由于在计算机中按字节操作最方便，一个字节 8 个比特，刚好可以看做是一个 8 位向量。因此，选定扩域 $GF(2^8)$。然后需要选定一个在 $GF(2)$ 下的本原多项式。

第 4 章　TCP/NC 的改进及仿真结果

第 5 章　论文工作总结与展望

结　　论

恭喜已经到达学位论文的结论部分！

祝贺您在推动人类科学发展的道路上又迈进了七十亿分之一。

致　　谢

在西南交通大学研究生学位论文 LaTeX 模板 swjtuThesis 的开发制作过程中，主要参考了以下的国内高校学位论文 LaTeX 模板，在此对所有的模板作者表示由衷的感谢：

- 中国科大学位论文通用 LaTeX 模板 (目前为数不多经过校方认证发布的模板)[1]
- 清华大学学位论文 LaTeX 模板 thuthesis[2]
- 北京大学学位论文 LaTeX 模板 pkuthss[3]
- 浙江大学研究生硕士 (博士) 学位论文 LaTeX 模板[4]
- 哈工大硕博士毕业论文 XeLaTeX 模版 PlutoThesis[5]
- 南京理工大学学位论文 LaTeX 模版[6]
- 北京交通大学毕设 LaTeX 模板 bjtuThesis[7]

最后，感谢所有在 swjtuThesis 模板开发过程中予以了各种无私帮助的老师和同学。其中特别感谢：西南交通大学电气工程学院智能牵引供电团队胡海涛副教授在模板开发过程中提出的若干意见，博士研究生杨鸣凯对模板基于 CTeX 端的调试；交通运输与物流学院 14 级硕士研究生甘婧对模板存在的一些问题的指正；此外还有西南交通大学博士研究生冯玎、王玘、王湘、李勇，硕士研究生张佳怡等，在此一并感谢。

希望未来能有更多的同学加入学位论文文 LaTeX 模板的开发和完善工作，推广 LaTeX 在国内青年学生学者圈子中使用。

原始作者：Limin HUANG

@ Studio0513

[1] 下载地址：http://gradschool.ustc.edu.cn/ylb/material/xw/wdxz.html
[2] 下载地址：https://github.com/xueruini/thuthesis
[3] 下载地址：https://github.com/CasperVector/pkuthss
[4] 下载地址：https://github.com/ZJU-Awesome/write_with_LaTeX
[5] 下载地址：https://github.com/dustincys/PlutoThesis/releases
[6] 下载地址：https://github.com/jiec827/njustThesis
[7] 下载地址：https://github.com/chenzewei01/bjtuThesis

参考文献

[1] Dah Ming Chiu, Raj Jain. Analysis of the Increase and Decrease Algorithms for Congestion Avoidance in Computer Networks[J]. Computer Networks & Isdn Systems. 1989, **17**(1):1–14

[2] YE TIAN, KAI XU, , NIRWAN ANSARI. TCP in Wireless Environments: Problems and Solutions[C]. IEEE Radio Communications • March 2005. IEEE, 2005

[3] J. K. Sundararajan, D. Shah, M. Medard, M. Mitzenmacher, J. Barros. Network Coding Meets TCP[C]. Proc. IEEE INFOCOM 2009. 2009, 280–288

[4] H. Balakrishnan, V. N. Padmanabhan, S. Seshan, R. H. Katz. A comparison of mechanisms for improving TCP performance over wireless links[J]. IEEE/ACM Transactions on Networking. Dec. 1997, **5**(6):756–769

[5] C. Barakat, E. Altman, W. Dabbous. On TCP performance in a heterogeneous network: a survey[J]. IEEE Communications Magazine. Jan 2000, **38**(1):40–46

[6] T. R. Henderson, R. H. Katz. Transport protocols for Internet-compatible satellite networks[J]. IEEE Journal on Selected Areas in Communications. Feb 1999, **17**(2):326–344

[7] Tsaoussidis Vassilis, Matta Ibrahim. Open issues on TCP for mobile computing[J]. Wireless Communications & Mobile Computing. 2010, **2**(1):3–20

[8] J. J. Alcaraz, F. Cerdan, J. Garcia-Haro. Optimizing TCP and RLC interaction in the UMTS radio access network[J]. IEEE Network. March 2006, **20**(2):56–64

[9] F. Vacirca, A. De Vebductis, A. Baiocchi. Optimal design of hybrid FEC/ARQ schemes for TCP over wireless links with Rayleigh fading[J]. IEEE Transactions on Mobile Computing. April 2006, **5**(4):289–302

[10] M. Assaad, D. Zeghlache. Cross-Layer design in HSDPA system to reduce the TCP effect[J]. IEEE Journal on Selected Areas in Communications. March 2006, **24**(3):614–625

[11] F Vacirca, A De Vendictis, A Todini, A Baiocchi. On the effects of ARQ mechanisms on TCP performance in wireless environments[C]. Global Telecommunications Conference, 2003. GLOBECOM '03. IEEE. 2003, 671–675 Vol.2

[12] T. Mahmoodi, V. Friderikos, O. Holland, A. H. Aghvami. Cross-Layer Design to Improve Wireless TCP Performance with Link-Layer Adaptation[C]. 2007 IEEE 66th Vehicular Technology Conference. 2007, 1504–1508

[13] M Allman, D Glover, L Sanchez. Enhancing TCP Over Satellite Channels using Standard Mechanisms[M]. RFC Editor, 1999

[14] A. Bakre, B. R. Badrinath. I-TCP: indirect TCP for mobile hosts[C]. International Conference on Distributed Computing Systems. 1995, 136–143

[15] Hari Balakrishnan, Srinivasan Seshan, Randy H. Katz. Improving reliable transport and handoff performance in cellular wireless networks[J]. Wireless Networks. 1995, **1**(4):469–481

[16] Gergö Buchholcz, Adam Gricser, Thomas Ziegler, Tien Van Do. Explicit loss notification to improve TCP performance over wireless networks[C]. IEEE International Conference on High Speed Networks and Multimedia Communications. Springer, 2003, 481–492

[17] Lawrence S. Brakmo, Larry L. Peterson. TCP Vegas: End to end congestion avoidance on a global Internet[J]. IEEE Journal on selected Areas in communications. 1995, **13**(8):1465–1480

[18] K Ramakrishnan, S Floyd, D Black. RFC 3168[J]. The addition of explicit congestion notification (ECN) to IP. 2001

[19] R. Ahlswede, Ning Cai, S. Y. R. Li, R. W. Yeung. Network information flow[J]. IEEE Transactions on Information Theory. Jul. 2000, **46**(4):1204–1216

[20] Philip A Chou, Yunnan Wu, Kamal Jain. Practical network coding[C]. Proceedings of the annual Allerton conference on communication control and computing. The University; 1998, 2003, vol. 41, 40–49

[21] Jérôme Lacan, Emmanuel Lochin. On-the-Fly Coding to Enable Full Reliability Without Retransmission[J]. CoRR. 2008, **abs/0809.4576**

[22] J. K. Sundararajan, D. Shah, M. Medard. ARQ for network coding[C]. 2008 IEEE International Symposium on Information Theory. 2008, 1651–1655

[23] J. K. Sundararajan, D. Shah, M. Medard, S. Jakubczak, M. Mitzenmacher, J. Barros. Network Coding Meets TCP: Theory and Implementation[J]. Proceedings of the IEEE. Mar. 2011, **99**(3):490–512

[24] 梅达尔. 网络编码基础与应用 [M]. 机械工业出版社, 2014

[25] James F Kurose. Computer networking: A top-down approach featuring the internet, 3/E[M]. Pearson Education India, 2005

[26] Luigi Rizzo. Effective erasure codes for reliable computer communication protocols[J]. ACM SIGCOMM computer communication review. 1997, **27**(2):24–36

[27] R. Koetter, M. Medard. An algebraic approach to network coding[J]. IEEE/ACM Transactions on Networking. Oct 2003, **11**(5):782–795

[28] A. Fujimura, S. Y. Oh, M. Gerla. Network coding vs. erasure coding: Reliable multicast in ad hoc networks[C]. MILCOM 2008 - 2008 IEEE Military Communications Conference. 2008, 1–7

[29] Szymon Chachulski, Michael Jennings, Sachin Katti, Dina Katabi. Trading structure for randomness in wireless opportunistic routing[M], vol. 37. ACM, 2007

[30] Tracey Ho, Muriel Medard, Jun Shi, Michelle Effros, David R Karger. On randomized network coding[C]. Proceedings of the Annual Allerton Conference on Communication Control and Computing. Citeseer, 2003, vol. 41, 11–20

[31] Chien-Chia Chen, Chieh-Ning Lien, Uichin Lee, Soon Y Oh, Mario Gerla. CodeCast: network coding based multicast in MANETs[C]. Demos of the 10th international workshop on mobile computing systems and applications (HotMobile 2009). 2009

[32] J. s. Park, M. Gerla, D. S. Lun, Y. Yi, M. Medard. Codecast: a network-coding-based ad hoc multicast protocol[J]. IEEE Wireless Communications. October 2006, **13**(5):76–81

[33] Chien-Chia Chen, Soon Y Oh, Phillip Tao, Mario Gerla, MY Sanadidi. Pipeline network coding for multicast streams[C]. Proceedings of the 5th International Conference on Mobile Computing and Ubiquitous Networking (ICMU), Seattle, USA. 2010

[34] Desmond S. Lun, Muriel Médard, Ralf Koetter, Michelle Effros. On coding for reliable communication over packet networks[J]. Physical Communication. 2008, **1**(1):3–20

[35] C. Fragouli, D. Lun, M. Medard, P. Pakzad. On Feedback for Network Coding[C]. 2007 41st Annual Conference on Information Sciences and Systems. 2007, 248–252

[36] 张宗橙. 纠错编码原理和应用 [M]. 电子工业出版社, 2003

附录 A

攻读硕士学位期间发表的论文及科研成果

1. 发表学术论文

[1] **姜维**, 诸葛亮, 赵云, 等. 基于战争大数据的九伐中原与六出祁山的战略决策分析 [J]. 中国战争信息物理, 2015, 40(13): 520-530.（SCI 收录号: , IF=0.16）

[2] **姜维**, 诸葛亮, 刘禅, 等. 基于神经网络的子午谷奇谋仿真研究 [J]. 战争仿真学报, 2015, 19(4): 324-330.（EI 收录号：）

[3] W. Jiang, G. L. Zhu. , et al. Improved $\mathcal{H}_\infty$ Robust Control Design for Mu-Niu-Liu-Ma System[C]. Intelligent System Design and Applications, Chengdu, 2015. EI 收录号：）

2. 申请国家专利

[1] 诸葛亮, **姜维**. 一种木牛流马的制造方式及其使用方法. 中国, 20152006387.1[P]. xxxx-xx-xx.

3. 参与科研项目

[1] **学生第一主研**, 高效粮食载运工具基础科学问题研究与样机研制, 蜀自然科学基金项目. 课题编号: XXXX.