软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2014,25(7):1541−1556 [doi: 10.13328/j.cnki.jos.004451]
©中国科学院软件研究所版权所有.
E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

应用累积系数确认的网络编码机会路由协议*

王伟平, 陈小专, 鲁鸣鸣, 王建新

(中南大学 信息科学与工程学院,湖南 长沙 410083)

通讯作者: 王建新, E-mail: jxwang@mail.csu.edu.cn

摘 要: 在无线 mesh 网络中,机会路由通过高效使用无线传输的广播特性显著地提高了无线网络的吞吐量.引入网络编码,使得机会路由协议可以避免复杂的调度,更加易于实现.然而,网络编码的引入给机会路由协议带来新的问题:转发节点应该发送多少编码包?MORE 等协议依据平均链路状况信息来预计节点转发编码包数目的方法,无法准确判定发送的冗余.以 CCACK 为代表的研究采用逐跳反馈的方式来减少编码包的冗余发送.首先,针对采用正交向量确认的 CCACK 机制进行分析,说明了 CCACK 尽管可以减少确认开销,减少误判,但却带来了"信息空间已覆盖而无法正交"的漏判问题.在此基础上,提出了一种基于累积编码系数反馈确认的网络编码机会路由协议 CFACK.该确认机制中转发节点通过侦听下游节点的编码系数向量,并与来自上游节点的编码系数向量进行相关性分析,从而获知下游节点信息是否覆盖自身信息.证明了在无差错网络环境下该确认机制不存在误判和漏判的可能,同时,在有差错网络环境下对该确认机制的有效性进行了分析.结果表明:在一般节点分布情况下,利用额外的一次携带确认,可以确保 90%以上的准确性.仿真测试结果表明:CFACK 相比 CCACK,显著提高了网络的吞吐量,平均提高率为 72.2%,同时在编码计算、存储和包头开销上都少于 CCACK.

关键词: 机会路由;网络编码;无线 mesh 网络

中图法分类号: TP393

中文引用格式: 王伟平,陈小专,鲁鸣鸣,王建新.应用累积系数确认的网络编码机会路由协议.软件学报,2014,25(7):1541−1556. http://www.jos.org.cn/1000-9825/4451.htm

英文引用格式: Wang WP, Chen XZ, Lu MM, Wang JX. Network coding based opportunistic routing using cumulative coding coefficient feedback acknowledgments. Ruan Jian Xue Bao/Journal of Software, 2014,25(7):1541−1556 (in Chinese). http://www.jos.org.cn/1000-9825/4451.htm

Network Coding Based Opportunistic Routing Using Cumulative Coding Coefficient Feedback Acknowledgments

WANG Wei-Ping, CHEN Xiao-Zhuan, LU Ming-Ming, WANG Jian-Xin

(School of Information Science and Engineering, Central South University, Changsha 410083, China)

Corresponding author: WANG Jian-Xin, E-mail: jxwang@mail.csu.edu.cn

Abstract: Opportunistic routing (OR) significantly improves transmission reliability and network throughput in wireless mesh networks by taking advantage of the broadcast nature of the wireless medium. With network coding (NC), OR can be implemented in a simple and practical way without resorting to a complicated scheduling. With the introduction of NC, how to reduce redundant transmission of coded packets becomes a very important problem in OR protocol. MORE, *et al*. protocols estimate the expected number of transmissions for each forwarder based on periodic measurements of the average link loss rates. However, these approaches may suffer severe performance degradation in dynamic wireless environments. Recently, some studies, known as CCACK, employ orthogonal vector as feedback to reduce redundant transmission of coded packets. The analysis of CCACK scheme indicates that the false-positive probability is reduced at the cost of increasing the false-negative probability, which results in unnecessary packets transmission. This paper proposes a NC-based

* 基金项目: 国家自然科学基金(61173169, 61202494); 教育部新世纪优秀人才计划(NCET-10-0798)
收稿时间: 2012-04-16; 修改时间: 2012-09-12; 定稿时间: 2013-07-01

© 中国科学院软件研究所 http://www.jos.org.cn

OR protocol, named CFACK, based on cumulative coding coefficient feedback acknowledgement. In this scheme, the coding vectors piggybacked in coded packets are used as feedback information, and each forwarder overhears coding vectors sent by downstream nodes. Through correlation analysis between coding vectors from upstream nodes and downstream ones each forwarder knows whether its knowledge space is covered by its downstream nodes. This paper proves that CFACK is completely free from any false-positive and false-negative problem in reliable network. The efficiency of CFACK in unreliable network is also analyzed, and the result shows that in random topologies embedding an extra ACK vector in each packet can guarantee 90% accuracy. Evaluation shows that, compared with CCACK, CFACK significantly improves throughput by reducing unnecessary packet transmission, with average improvements of 72.2%. Furthermore, the overheads of encoding computation, storage, and header of CFACK are less than that of CCACK.

Key words: opportunistic routing; network coding; wireless mesh network

由于无线网络中链路的不可靠性,与传统路由相比,机会路由可以显著提高吞吐量[1].Chachulski 等人将随机网络编码引入机会路由协议,提出了 MORE 协议[2].源节点发送数据包的随机线性组合,每个转发节点通过随机组合的方式进行组合并对所收到的数据包进行编码,然后转发,目的节点收到足够的数据包就能解码得到原始数据包.网络编码的引入,使得编码数据包对于解码获得原始数据包来说都是等价的,无需确切得知哪个数据包丢失,只需到达目的节点的编码包数达到一定数量就能解码,从而获得了比传统机会路由更高的吞吐量.但机会路由中同样的信息可能由多个转发节点参与转发,带来了传输信息的冗余.这里需要解决的关键问题是:每个转发节点应该转发多少个编码包,才能保证数据包的正常到达,同时又将冗余发送降低到最小.

以图 1 为例来说明该问题.源节点 S 拥有 A 和 B 两个转发节点.S 将原始包 P_1,P_2 和 P_3 组合成 $P_1+2P_2+3P_3$,$P_1+2P_2+P_3$ 和 $P_1+P_2+P_3$ 这 3 个编码包发送.3 个编码包的编码系数可表示成编码向量,分别为(1,2,3),(1,2,1)和(1,1,1).编码包(1,2,3)和(1,1,1)被节点 A 接收,编码包(1,2,1)和(1,1,1)被节点 B 接收.这时,虽然下游节点 A 和 B 单独都没有收到足够多的信息量(即线性独立编码包)来代替 S 完成数据发送任务,但两个节点合起来已收到足够的信息量,可以替代 S 完成数据的发送任务.此时,S 已没必要继续发送编码数据包.然而,问题是如何让 S 节点知道这一情况,停止发送冗余的编码包.

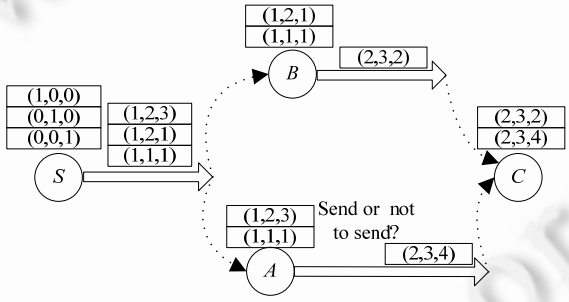


Fig.1 Important of knowing how many coded packets to transmit

图 1 节点需要知道发送多少编码包的重要性

为了降低发送冗余,MORE 协议中,源节点利用转发节点到目的节点的 ETX 值[3]预先计算出每个转发节点的传输信誉值,该值可以体现出数据包经由该节点到目的节点传输的代价,以此来决定转发的概率.这样,并非每个收到数据包的节点都进行转发,从而减少了冗余、提高了吞吐量.但这种预计算的方法仍然不能保证目标节点能够接收到足够的编码包进行解码,因此,MORE 中,当目的节点能够解码时,发送 ACK 确认,源节点一直保持发包直到收到 ACK 才停止当前批次的编码包的发送.预计算的方法可以减少冗余但无法消除冗余,同时,由于 ACK 反馈的过程中源节点和中间节点一直保持发送状态,因此也造成了一定的包冗余.

一些工作基于 MORE 中采用的预计算传输信誉的方法,同时在传输和编码机制上进行了改进,提高了网络的吞吐量.CodeOR 通过将“块”分解为尺寸更小的“段”,并使用滑动窗口机制允许多个“段”的数据同时在网络上传输,减小了解码延迟,而且利用了大规模网络产生的延迟带宽,提高了吞吐量[4].SlideOR 把在线编码和 TCP Vegas 的思想结合扩展到了无线多跳网络当中,通过流式编码机制,显著提高了网络的性能[5].MIXIT 利用无线

© 中国科学院软件研究所 http://www.jos.org.cn

链路符号出错的概率比包出错的概率要低这一特点,通过把 MORE 基于包的编码改成基于符号级的编码,从而提高网络的性能[6].文献[7]则不再采用基于块的网络编码,而将编码窗口滑动的机制与机会路由协议结合,分析了解码窗口大小的分布,并且通过在源端限制注入到网络中包的数目,从而控制解码窗口的大小.CodePipe 是基于随机线性网络编码和机会路由的可靠多播路由协议[8],该协议将流内和流间编码结合在一起,提高了多播的性能.

基于预先计算传输信誉值来减少冗余的方法依赖于对链路丢失率的精确和实时的测量.由于丢包率是通过周期性的探测和广播来进行估计的,探测的频率越高,精确度越高,但开销也就越大.为了减少该开销,MORE 的作者只在实验一开始收集丢包率和计算信誉值.然而文献[9,10]的研究表明:尽管链路参数在一个比较长的时期内是稳定的,但是当有背景流存在时,就变得相当敏感.例如,在文献[9]中作者观察到:在拥有 14 个节点的测试床,两个节点之间发送 100 个 ping 包(1packet/s)会导致 10%的链路的 ETT(expected transmission time)测量值增长 200%甚至更多.更糟糕的是,两个节点之间 TCP 一分钟的数据传输会导致 55%的链路的 ETT 值增长 300%甚至更多.

因此,这种预先计算传输信誉值的方法往往不能体现网络的即时状态.

一些研究则不建立在预计算的基础上,而采用逐跳确认的方法及时阻止上一跳节点的冗余发送.

SOR 给每个编码包分配一个序列号,通过 ACKMap 实现逐跳确认[11].但由于编码包是多个数据包的混合,通过序列号并不能判断出编码包之间的线性关系,因此,ACKMap 逐跳确认机制会存在漏判,产生不必要编码包的传输.

CCACK[12]提出了利用正交向量来进行累积编码确认的机制.上游节点通过正交向量运算知道下游转发节点是否已经接收到了充足的信息量,从而及时停止编码包的发送,减少不必要编码包的传输.CCACK 相对于 MORE,其吞吐量有了较大的增长,但由于正交向量是对编码向量空间的有损压缩,存在误判的概率.为了降低正交向量误判的概率,CCACK 捎带的只是下游节点部分向量的正交向量,这样,正交向量就不能完整表达下游节点编码向量之间的线性关系,从而导致 CCACK 存在许多漏判的情况,产生一些不必要编码包的发送.

本文指出了 CCACK 机制中存在的漏判问题,分析了漏判的原因和概率.从线性相关性本质着手,提出了一种直接利用下游节点发送数据包携带的编码向量进行累积确认的方法,证明了该方法的正确性.同时,针对不可靠网络环境,提出了采用两次携带确认的机制,提高了该方法的有效性.仿真结果表明:该方法相对于 CCACK,吞吐量获得了显著的提高,同时降低了协议开销.

1 CCACK 反馈机制存在的问题分析

按照 CCACK 机制,任意节点保存 3 个队列,分别是:新数据包编码向量空间 B_v 保存来自上游节点的新数据包的编码向量,即 B_v 中所有向量都是线性无关的;接收到数据包的编码向量空间 B_u 保存所有来自上游节点编码包的编码向量;输出数据包编码向量空间 B_w 保存节点发送出去的编码包编码向量.为了方便表示,我们分别用 B_v^i,B_u^i,B_w^i 表示节点 i 对应的 3 个编码向量空间,用 $R(E)$表示向量空间 E 的秩,即 E 中的线性无关的向量的个数.

在 CCACK 中,采用了正交向量携带确认机制.由于正交向量对编码向量线性相关性的判断只是一种近似的判断,故存在假阳性问题,即上游节点依据正交反馈的判定认为下游节点已经覆盖自己的认知空间,而实际上并没有,因而出现误判.这种误判会导致节点少发一些包,最终目标节点可能没有接收到足够的包,不能解码.

为了减少误判,CCACK 通过引入 M 个 H 矩阵使得假阳性问题(误判)出现的概率从 $\frac{1}{2^8}$ 降到 $\left(\frac{1}{2^8}\right)^M$.CCACK 机制中,每次从接收向量空间 B_u 中选取 $\left(\frac{K}{M}-1\right)$ 个向量 u,K 表示编码块的大小,计算出同时满足以下条件的 Z 向量:

$$\forall j=1,\ldots,M,u\cdot H_j Z^T=0 \tag{1}$$

为了方便描述,当等式(1)成立时,称其为 u 与 Z 满足 H-正交.

© 中国科学院软件研究所 http://www.jos.org.cn

上游节点处,当计算出编码向量 x 与某一反馈向量 Z 满足 H-正交时,认为该向量通过了 H_tests.当 B_u 和 B_w 中通过 H_tests 的编码向量组的秩和 B_v 的秩相等时,节点认为下游节点已经覆盖其认知空间,停止发包.只有收到来自上游节点新的数据包(即与原 B_v 中的向量线性不相关)时,才能触使其再次发包.

然而,这样会导致对于某个 Z 向量,接收向量空间中只有 $\left(\frac{K}{M}-1\right)$ 个向量与 Z 满足 H-正交.CCACK 这种依据反馈信息正交的判定机制尽管减少了误判率,但存在漏判的可能,使得本来某节点的下游节点收到的信息已经覆盖了该节点的信息,但该节点依据正交判定认为没有完全覆盖.这会导致发送冗余的数据包.

我们以图 2 为例来说明 CCACK 存在的漏判问题.

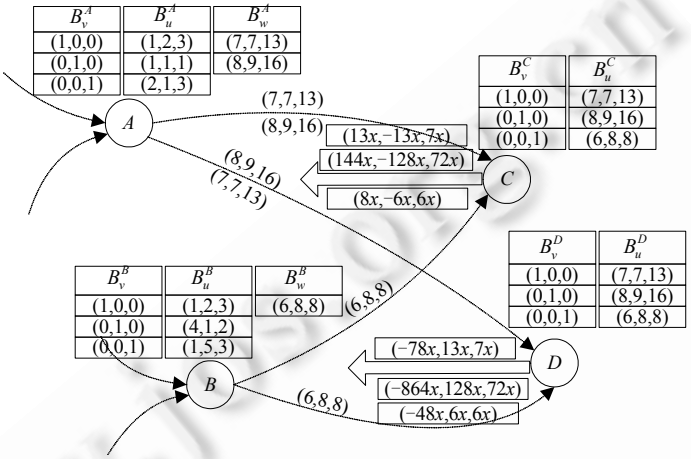


Fig.2 A typical scenario of an NC-based OR protocol to illustrate the false-negative problem in CCACK

图 2 CCACK 反馈机制存在的漏判问题举例

假设图 2 中节点 A 和节点 B 的下一跳节点为节点 C 和 D,节点 A 与 B 之间不存在直接传输链路.假设 K=3, M=2,那么 C 节点在计算正交反馈向量时,正交向量一次最多与一个向量正交.假设节点 C 和节点 D 都接收到来自 A 的两个编码包(7,7,13),(8,9,16)和来自节点 B 的一个编码包(6,8,8).

依据 CCACK 的反馈机制,我们假设节点 C 的 H 矩阵为

$$H_{c1}=\begin{Bmatrix}1,0,0\\0,2,0\\0,0,1\end{Bmatrix},H_{c2}=\begin{Bmatrix}1,0,0\\0,3,0\\0,0,2\end{Bmatrix};$$

假设节点 D 的 H 矩阵为

$$H_{d1}=\begin{Bmatrix}1,0,0\\0,2,0\\0,0,4\end{Bmatrix},H_{d2}=\begin{Bmatrix}1,0,0\\0,3,0\\0,0,3\end{Bmatrix}.$$

那么节点 C 分别生成对应的 3 个反馈向量为(13x,−13x,7x),(144x,−128x,72x)和(8x,−6x,6x).若取 x=1,则节点 C 反馈的 3 个向量分别是 Z_c=(13,−13,7), Z_c'=(144,−128,72)和 Z_c''=(8,−6,6).节点 A 收到反馈向量后,分别用 B_u^A 和 B_w^A 中的向量与 Z_c, Z_c', Z_c'' 计算判定是否满足 H-正交条件.例如,针对 B_u^A 中的向量(1,2,3),分别计算得到:

$$(1,2,3)\cdot H_{c1}\cdot Z_c^T=-18,(1,2,3)\cdot H_{c2}\cdot Z_c^T=-23,$$
$$(1,2,3)\cdot H_{c1}\cdot (Z_c')^T=-152,(1,2,3)\cdot H_{c2}\cdot (Z_c')^T=-192,$$
$$(1,2,3)\cdot H_{c1}\cdot (Z_c'')^T=2,(1,2,3)\cdot H_{c2}\cdot (Z_c'')^T=8.$$

表明(1,2,3)与节点 C 的任意反馈向量都不满足 H-正交.同样可以计算得到 B_u^A 中其他向量(2,1,3),(1,1,1)与

© 中国科学院软件研究所 http://www.jos.org.cn

节点 C 的反馈向量也都不满足 H-正交.而针对 B_w^A 中的向量(8,9,16)和(7,7,13),计算得到:

$$(8,9,16)\cdot H_{c1}\cdot (Z_c')^T=0,(8,9,16)\cdot H_{c2}\cdot (Z_c')^T=0,$$

$$(7,7,13)\cdot H_{c1}\cdot Z_c^T=0,(7,7,13)\cdot H_{c2}\cdot Z_c^T=0.$$

表明(8,9,16)与 Z_c' 满足 H-正交,而(7,7,13)与 Z_c 满足 H-正交.

同样,节点 D 分别生成对应的 3 个反馈向量为$(-78x,13x,7x)$,$(-864x,128x,72x)$和$(-48x,6x,6x)$.若取 x=1,则 C 节点反馈的是 Z_d=(−78,13,7), Z_d' =(−864,128,72)和 Z_d'' =(−48,6,6).节点 A 收到反馈后,分别用 B_u^A 和 B_w^A 中的向量与 Z_d, Z_d', Z_d'' 计算判定是否满足 H-正交条件.同样可以计算出 B_u^A 中向量(1,2,3),(2,1,3),(1,1,1)与反馈向量都不满足 H-正交, B_w^A 中向量(8,9,16)与 Z_d' 满足 H-正交,即 $(8,9,16)\cdot H_{d1}\cdot (Z_d')^T=0,(8,9,16)\cdot H_{d2}\cdot (Z_d')^T=0$; 向量(7,7,13)与 Z_d 满足 H-正交,即 $(7,7,13)\cdot H_{d1}\cdot Z_d^T=0,(7,7,13)\cdot H_{d2}\cdot Z_d^T=0$.

根据上述计算判定,对于节点 A 来说, B_u^A 和 B_w^A 中与下游节点反馈向量 H-正交的个数是 2,而节点 A 本身新数据包队列向量的秩为 $R(B_v^A)=3$,因此,节点 A 认为下游节点 C 和 D 并未完全覆盖其认知空间,其相对下游节点还有一个新信息.

同样,节点 B 计算时只有向量(6,8,8)与 Z_c'' 是 H-正交,而 $R(B_v)$=3,故 B 节点同样认为下游节点 C 和 D 未完全覆盖其认知空间,还有两个新信息.

事实上,这个例子中节点 C 和 D 其实都已经完全覆盖了节点 A 和 B 的认知空间,这样就造成节点 A 一个多余编码包的发送,节点 B 两个多余编码包的发送.

上述例子说明,CCACK 机制中采用正交向量反馈的方式存在漏判的可能性,从而出现“信息空间已覆盖而无法正交”的问题.

接下来我们分析 CCACK 机制的漏判概率.为了方便表示,我们称节点 i 的发送向量,即存在于 B_w^i 中的向量,为节点 i 的 F 向量;称存在于 B_u^i 中,已能被下一跳节点的接收向量线性表示,但不包含在下一跳节点接收向量集合中的向量,为节点 i 的 U 向量.

通过分析与证明,我们得到定理 1(定理 1 的证明详见附录).

定理 1. CCACK 中,任意节点的 F 向量与下一跳节点的任一反馈向量 Z 满足 H-正交的概率大于 $\frac{1}{M}-\frac{1}{K}$,即:最多只需要收到 $\frac{MK}{K-M}$ 个来自其下一跳的反馈向量 Z,就能得到确认.而任意节点的 U 向量,与下一跳节点的任一反馈向量 Z 满足 H-正交的概率是 $\sum_{j\in V}C_{r_j}^{t_j}\left(\frac{1}{256}\right)^{r-t_j}+\left(1-\sum_{j\in V}C_{r_j}^{t_j}\left(\frac{1}{256}\right)^{r-t_j}\right)\left(\frac{1}{256}\right)^{M}$,其中,$M$ 是反馈正交向量产生时采用的 H 矩阵个数,K 是编码块的大小,V 为下一跳节点的集合.

$$r=Rank\left(\bigcup_{j\in V}B_u^j\right),r_j=Rank(B_u^j),t_j=\min\left(r_j,\frac{K}{M}-1\right).$$

举例来说明 F 向量和 U 向量被确认的情况:当某个节点有 5 个下一跳节点,K=32,M=4,r=32,r_j=10 时,该节点的 F 向量最多只需收到来自下一跳节点的 5 个反馈 Z 向量,就能判定该向量与 Z 向量 H-正交,从而得到确认.而该节点的 U 向量与任意反馈向量 Z 满足 H-正交的概率是

$$5\times C_{10}^7\times\left(\frac{1}{2^8}\right)^{32}+\left(1-5\times C_{10}^7\times\left(\frac{1}{2^8}\right)^{32}\right)\left(\frac{1}{2^8}\right)^4\approx\left(\frac{1}{2^8}\right)^4\approx 2.33\times 10^{-10}.$$

可见,U 向量的漏判概率几乎是 100%.

从图 2 的例子我们也可以看出:节点 A 的 F 向量(8,9,16),(7,7,13)、节点 B 的 F 向量(6,8,8)都能与反馈向量满足 H-正交,而节点 A 的 U 向量(1,2,3),(2,1,3),(1,1,1)以及节点 B 的 U 向量(1,2,3),(4,2,1),(1,5,3)都不能与反馈向量满足 H-正交.在这个例子中,尽管节点 A,B 的 U 向量此时已经完全能由下一跳节点 C 和 D 的接收向量线性表示,但由于漏判将导致 A,B 继续发送冗余编码包.这个例子中,A 与 B 的向量空间相同,将导致多一倍的发送量.

© 中国科学院软件研究所 http://www.jos.org.cn

2 累积编码系数反馈确认机制

2.1 基本思想

由于 CCACK 会导致节点不必要编码包的发送,降低了协议的性能,因此,我们考虑如何把转发节点的认知空间完全反馈到上游节点.由于下游节点的认知空间完全反映在编码系数上,最直接的办法是每次发送编码包时,节点都把其所收到编码包对应的所有线性无关的编码向量捎带在数据包当中,但这样开销太大,而且直接捎带编码向量的话,不适用于高丢包率的无线 mesh 网络.

我们观察到:下游节点在发送数据包时本身携带的用于解码的编码向量就能反映其认知空间,如果上游节点每次都能侦听到下游节点发送的编码包,并将这些编码包的编码向量累积在一起,自然就能得到下游转发节点的认知空间.节点就可以确定下游转发节点的认知空间是否覆盖其认知空间,是否需要停止发包.这种采用编码向量的随机线性编码作为反馈信息的做法有两个主要的好处:① 不存在误判的情形;② 转发的编码包中本身就有编码向量的随机线性编码(即与编码包对应的编码系数),不需要额外的信息.由于该方法是通过累积下游节点发送包中的编码向量来构成确认的,所以我们称其为累积反馈确认机制.

以图 3 场景为例,节点 A 和 B 为节点 S 的转发节点,节点 C 为节点 A 和 B 的共同转发节点.S 节点通过侦听节点 A 和 B 发送的编码数据包可以知道,其下游节点 A 和 B 覆盖的认知空间为向量(5,5,4),(6,7,5)和(2,4,2)组成的矩阵 M_1.把节点 S 的向量(1,0,0),(0,1,0),(0,0,1)与向量(5,5,4),(6,7,5),(2,4,2)组成矩阵 M_2,通过比较矩阵 M_1 和 M_2 的秩,即可知道节点 S 相对于其下游节点 A 和 B 是否还有新信息.这个例子中,由于 $R(M_1)=2$,$R(M_2)=3$,所以判定节点 S 相对于其下游节点 A 和 B 还有一个新信息.

同样,节点 B 的下游节点 C 覆盖的认知空间是由向量(8,11,7)和(10,15,9)组成的矩阵 M_3.节点 B 把向量(3,1,2),(1,2,1)与向量(8,11,7),(10,15,9)一起组成矩阵 M_4,通过比较矩阵 M_3 和 M_4 的秩,可以知道 $R(M_3)=2$,$R(M_4)=2$,所以节点 B 判定其下游节点 C 已经覆盖其认知空间,故节点 B 停止发送编码包,直到收到其上游节点 S 的新的线性独立数据包,才再次编码发包.

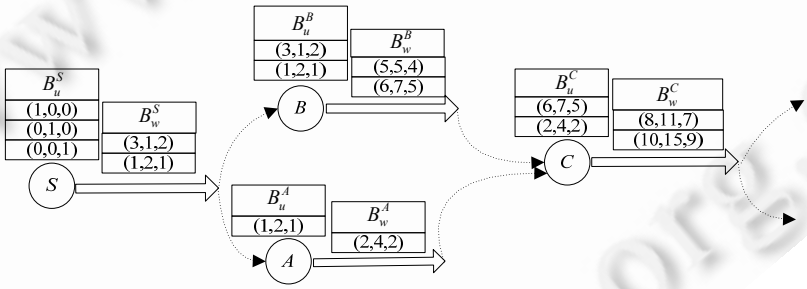


Fig.3 An example of cumulative coding coefficient acknowledgement

图 3 基于累积编码系数确认示例

2.2 累积编码系数反馈机制及其正确性证明

2.2.1 累积编码系数反馈机制

为了方便描述,给出以下符号的定义:M_u^i 是由节点 i 接收到的来自上游节点编码包的编码系数向量组成的向量组;M_w^i 表示由节点 i 发送的所有编码包的编码系数向量组;M_f^i 表示由节点 i 侦听到的所有下游节点发送的编码包的编码系数向量组;M_{op}^i 表示由节点 i 侦听到的所有上游节点和下游节点发送的编码包的编码系数向量组,即 $M_{op}^i = M_u^i \cup M_f^i$.

采用累积反馈机制的任意节点 i,通过比较 $R(M_f^i)$ 和 $R(M_{op}^i)$ 的值来决定是否停止发包:当 $R(M_f^i) = R(M_{op}^i)$ 时,停止编码包的发送;当 $R(M_f^i) < R(M_{op}^i)$ 时,节点认为其相对下游转发节点还有新数据包,即下游转发节点还

© 中国科学院软件研究所 http://www.jos.org.cn

没有完全覆盖其认知空间,继续发编码包.

由于向量组 M_{op}^i 包含了向量组 M_f^i 中的所有向量,所以不存在 $R(M_f^i)>R(M_{op}^i)$ 的情况.

2.2.2 正确性证明

下面,我们通过向量组的线性相关性理论来证明累积编码系数确认机制的正确性.

为了证明累积反馈机制的正确性,我们只需证明:(1) 当 $R(M_f^i)=R(M_{op}^i)$ 时,节点 i 随机产生的任意编码向量都会与转发节点集 V 中的所有节点 j 的已接收向量的并集 $\bigcup_{j\in v}M_u^j$ 线性相关;(2) 当 $R(M_f^i)<R(M_{op}^i)$ 时,至少存在一个由节点 i 产生的编码向量与下游节点发送向量线性无关.

前者说明了 $R(M_f^i)=R(M_{op}^i)$ 时,下一跳节点的信息肯定已经覆盖了节点 i 的信息,可以作为停止发送的充分条件,不存在误判(误判是指事实没有覆盖而被判定为覆盖,造成后继节点无法解码);后者说明了只要 $R(M_f^i)<R(M_{op}^i)$,下一跳节点的发送信息肯定没有完全覆盖节点 i 的信息,应继续发送,不存在漏判(漏判是指事实已经覆盖而被判定为未覆盖,造成冗余包的发送).

以下是证明过程.

由于 M_f^i 表示由节点 i 侦听到的所有下游节点发送包的编码系数向量组,而 M_{op}^i 表示由节点 i 收到以及侦听到的所有上游节点和下游节点发送包的编码系数向量组,$M_{op}^i=M_u^i\cup M_f^i$.根据秩比对的原理容易推知:当 $R(M_f^i)=R(M_{op}^i)$ 时,表示下游节点的发送向量已经覆盖了上游节点的接收向量,而下游节点的发送向量是由 $\bigcup_{j\in v}M_u^j$ 随机线性编码产生的,所以有 $\bigcup_{j\in v}M_u^j$ 已经覆盖了节点 i 的信息;同样容易证明:当 $R(M_f^i)<R(M_{op}^i)$ 时,下游节点的发送向量不能完全覆盖上游节点的已接收向量.

由上述证明可知:在侦听不丢包的情况下,采用累积编码系数反馈确认机制不存在误判和漏判情况.

2.3 不可靠网络环境下累积编码系数反馈机制有效性分析

累积编码系数反馈机制在可靠的网络环境下,即当后继节点发包时,前一跳节点都能够侦听到的情况下,没有漏判情况.如图 4 所示场景,$A\rightarrow B\rightarrow C$ 形成传输链路,在可靠的环境下,B 发送编码包,A 都可以侦听到,从而可以准确判断下一跳的信息空间是否覆盖自身.但实际情况中,无线 mesh 网络的信号易受多方面因素的影响,使得无线网络的链路状态随着时间而不断变化,经常出现高丢包率以及链路丢包率不对称的现象.在 Roofnet[13]上,超过半数的链路的丢包率高达 30%.同时,这种链路丢包率的不一致,会导致采用累积反馈确认机制的节点接收到的反馈信息数目不够.

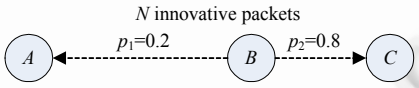


Fig.4 Simple linear topology scenario

图 4 简单直线拓扑场景

同样以图 4 为例,假设链路 $B\rightarrow A$ 的到达率为 p_1,链路 $B\rightarrow C$ 的到达率为 p_2,节点 B 接收到来自节点 A 的新信息(一条信息如果与节点之前接收到的所有信息都线性独立的话,那么其相对节点来说为新信息)数目为 n 条,那么节点 A 需要侦听到从 B 发出的 n 条信息.在如图 4 所示的场景中,节点 B 把 n 条信息成功发送到节点 C 所需要的期望传输次数为 n/p_2,而节点 A 能够成功侦听到的反馈信息条数为 np_1/p_2.显然,当 $p_1\geqslant p_2$ 时,节点 A 能够接收足够的反馈信息;但是,当 $p_1<p_2$ 时,节点 A 能够接收反馈信息的数目就不够.图 4 中,当 $n=4$,$p_1=0.2$,$p_2=0.8$ 时,节点 A 能侦听到的信息条数期望值为 1,远少于其所需要的数目 4.

显然,这种情况将导致节点 A 无法全部侦听到节点 B 发送的数据包,从而导致漏判.接下来,通过分析在存在丢包的情况下,节点需要反馈信息数目与其实际侦听到的数据包数目的比值来看丢包对累积确认机制有效性的影响.

© 中国科学院软件研究所 http://www.jos.org.cn

定义节点 i 的 *ETX* 值为节点 i 依据最优路径投递一个包到目的节点 d 所需的期望传输次数[3].当转发节点接收到包后,依据 *ETX* 值最小的原则选择节点转发该包.假设网络中共有节点数目为 N,按照 *ETX* 值升序从小到大顺序给节点编号,$i<j$ 表示节点 i 比节点 j 更靠近目标节点,即节点 i 的 *ETX* 值小于节点 j.设 *ETX* 值比源节点小的节点数目为 N',编号范围为 1~N'.源节点编号为 N'+1.

我们采用与文献[14,15]同样的假设,即假设不同节点的接收事件是独立的,分析一个包从源节点投递到目标节点的过程.定义以下变量:

- ε_{ij} 表示从节点 i 直接给节点 j 发包的丢包率,当节点 i 和 j 之间没有链路时,ε_{ij}=1;
- Z_i 表示为了将一个数据包从源节点成功传输到目标节点,节点 i 必须完成的最少期望传输次数;
- L_j 表示节点 j 必须成功转发的数据包数目.

因为节点 j 收到某个数据包,只有当所有 *ETX* 值比其小的节点都没有收到该包时,它才必须转发该包.该事件的发生概率为 $\prod_{k<j}\varepsilon_{ij}$,所以节点 j 必须转发的数据包数目 L_j 可以表示成:

$$L_j=\sum_{i>j}\left(Z_i\times(1-\varepsilon_{ij})\prod_{k<j}\varepsilon_{ik}\right) \tag{2}$$

显然,由于源节点产生数据包,所以 $L_{N'+1}$=1.

现在我们来考虑节点 j 必须完成的最少传输次数.节点 j 传输数据包,当一个 *ETX* 值比它小的节点接收到该包时,称为成功传输.那么节点 j 向前传输 1 次,成功发送该包的概率为 $(1-\prod_{k<j}\varepsilon_{jk})$. 由于节点 j 必须成功转发的数据包数目为 L_j,那么需要的发送次数 Z_j 为

$$Z_j=\frac{L_j}{(1-\prod_{k<j}\varepsilon_{jk})} \tag{3}$$

节点 j 需要的反馈信息条数,即为它接收到的包的数目,那么节点 j 需要反馈的信息条数 S_j 表示为

$$S_j=\sum_{k>j}(Z_k\times(1-\varepsilon_{kj})) \tag{4}$$

节点 j 能够接收到的反馈信息条数,就是其能侦听到所有 *ETX* 值比其小的节点发送的包的数目,节点 j 实际能够收到的反馈信息的条数为

$$R_j=\sum_{i<j}(Z_i\times(1-\varepsilon_{ij})) \tag{5}$$

在已知任意ε_{ij} 值的情况下,我们以源节点初始发送值 $L_{N'+1}$=1 开始,利用公式(2)~公式(5),可以计算出 S_j 和 R_j 的值.具体过程是:由 $L_{N'+1}$=1 计算得到 $Z_{N'+1}$,然后迭代计算 $L_{N'},Z_{N'},L_{N'-1},Z_{N'-1},\ldots,L_1,Z_1$,最后分别计算出 S_j 和 R_j 的值.

为了考察一般分布情况下的ε_{ij} 值,随机地把 50 个静态节点部署到 1000m×1000m 区域,利用第 4.1 节所介绍的传播模型根据节点 i 和 j 之间的距离计算出节点之间包的到达率 p_{ij}(传播模型中的功率衰减因子β取值为 2),从而得到ε_{ij}=1−p_{ij}.然后,根据公式(2)~公式(5)分别计算出各个节点需要的反馈信息的条数 S_j 与接收到的反馈信息的条数 R_j 的比值.定义 $\eta_j=\frac{S_j}{R_j}$,图 5 中的理论曲线为依据随机产生的 110 个拓扑中的ε_{ij} 计算得到的η值的累积分布曲线.而由于实际机会路由协议中在选择下一跳节点时并未用到所有 *ETX* 比自己小的点,通常只是选择若干个 *ETX* 相对较小的点作为下一跳,这与理论分析略有不同.因此,我们在测试模拟结果时采用了 *ETX* 值相对较小的 8 个节点作为下一跳邻居完成了相应模拟测试,得到了图 5 所示的模拟曲线.

从图中可以看出:理论分析和模拟实验得到的结论基本上是一致的,$\eta\leqslant 1$ 的节点数目约占总节点数的 71%,这就意味着,如果我们只是单纯地采用侦听下游节点编码包中的编码向量作为反馈,当下游节点发送时,对于一般分布的网络拓扑,只有约 70%的下游发送包可以被上游节点侦听到,这会造成约 30%的漏判.

从曲线中可以看出:$\eta\leqslant 2$ 的节点比例超过了 90%.这给了我们一个启示:在编码系数反馈机制中,如果下游节点在发送编码包时除了携带该编码包自身的编码系数向量以外,从收到的数据包随机线性生成一个编码系

© 中国科学院软件研究所 http://www.jos.org.cn

数向量额外捎带在该编码包中,那么上游节点每侦听到一个来自下游转发节点的编码包,就相当于接收到两条反馈信息,这样能够使 90%以上的节点侦听到足够的反馈信息.

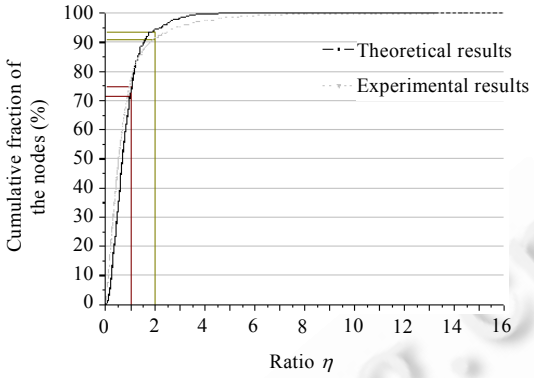


Fig.5 Cumulative distribution function of the ratio η

图 5 比值η累积分布图

3 CFACK 的设计

基于上述反馈机制的启示,我们提出了基于累积编码系数反馈确认机制的机会路由协议——CFACK (cumulative feedback acknowledgment).

3.1 CFACK概述

CFACK 的源节点和转发节点采用基于块的流内随机线性编码.每 32 个原始数据包组成一个编码块.线性组合的随机系数取自 2^8 大小的伽罗瓦域.

源节点将同一编码块中的原始数据包随机产生了 K 个编码包,$K\geqslant 32$.每当接收到上游节点的新数据包,转发节点对所收到的编码包进行随机线性网络编码,生成新的编码包进行传输.

发送过程中,源节点和中间节点保持侦听状态,采用累积编码系数反馈确认机制,依据侦听到的编码系数判定下一跳节点是否覆盖自身数据空间以停止发送.当收到目的点的块确认信息后,停止当前块的发送,开始新块的发送.

目标节点利用高斯消元对编码包进行解码,还原出原始数据包,并向源节点发送对编码块的确认包.

3.2 CFACK的编码包发送与控制机制

基于第 2.3 节得到的启示,在 CFACK 中,采用 2 倍的累积反馈确认机制,即每次发送的编码包除了自身的编码向量外,同时还包含了一个携带编码向量,该向量是由接收向量随机线性运算得到.

任意节点保存 3 个编码向量矩阵 M_u,M_f,M_{op}.源节点或者中继节点每当收到新的数据包或者来自上游节点的新编码包时,将其编码向量加入矩阵 M_u 和 M_{op},如果节点处于停止发包状态,触发节点重新开始发包.每当侦听到下游节点的编码包时,将其编码向量和携带编码向量加入矩阵 M_f 和 M_{op},然后比较矩阵 $R(M_f)$ 和 $R(M_{op})$ 的大小:当 $R(M_f)<R(M_{op})$ 时,说明下游节点还未接收到足够多的编码包,继续编码包的发送;当 $R(M_f)=R(M_{op})$ 时,说明下游节点已经覆盖其认知空间,该节点将停止发送,直到从上游节点收到新的信息.

为了进一步避免漏判的出现,减少冗余发送,任意节点当收到来自上游节点编码包与已接收的编码包线性相关时,我们就近似认为这是由于上游节点没有收到足够的反馈信息造成的,由该节点直接发送一个显式反馈包给上游节点,反馈包中捎带两个由节点的接收向量随机线性运算产生的编码向量,我们将这种反馈包称为 LACK 包.

图 6 给出了中间节点处理包的过程.

© 中国科学院软件研究所 http://www.jos.org.cn

```
Algorithm 1. Packet processing procedure.
//sending a coded packet P at a forwarding node
if tx_state_=STATE_TX then
   construct a coded packet P by randomly linearly combination of all received coded packets;
   construct an ACK vector by randomly linearly combination of all vectors in Mu and embed it in packet P;
   send packet P;
end if

//upon receiving an ACK at an intermediate node
 stop transmitting packets from batch i;
 forward this ACK to upstream nodes on the shortest path

//upon receiving an LACK at an intermediate node
if LACK is comes from downstream nodes then
   store all coding vectors of LACK packet in vector space
   Mf and Mop;
   if R(Mf)=R(Mop) then
      set tx_state_=STATE_IDLE
   end if
end if

//upon receiving a data packet P at a forwarding node
if P is comes from upstream nodes then
   if P is innovative then
      store the coding vector of the packet in the vector space
   Mu and Mop
      set tx_state_=STATE_TX
   else
      discard the packet and send a LACK for batch i
   end if
else
   store both the coding vector and the ACK vector in the packet
in vector space Mf and Mop;
end if
if R(Mf)=R(Mop) then
   set tx_state_=STATE_IDLE
end if
```

Fig.6 Packet processing procedure at intermediate node

图 6 中间转发节点包处理流程算法

3.3 转发节点的选择

每个节点定期地测量相邻节点的链路丢包率,通过一次源节点到目的节点的广播,每个节点可以计算得到自己到目的节点的 *ETX* 值[3].当某个节点选择转发节点时,依据邻居节点到目标节点的 *ETX* 值,选择其中 *ETX* 较小的 m 个节点作为转发节点集.为了避免太多的转发节点导致通信介质的竞争过于激烈,CFACK 中选择 m=8.

3.4 编码块的确认

目标节点把接收到的编码包加入到当前编码块的解码矩阵中,当目标节点接收到足够多的新编码包,能够解码成功时,就立即向源节点发送一个 ACK 包.

为了加快 ACK 包的投递,使用最短路径进行发送.在所有节点中,ACK 包的优先级高于数据包.为了保证 ACK 的可靠性,在 MAC 层采用单播,通过 MAC 层提供的可靠机制进行可靠性保证.

源节点接收到当前编码块的 ACK 包时,立即停止当前编码块数据包的发送.如果传输任务没有完成,源节点进行下一个编码块数据包的发送.其他所有节点一旦侦听到 ACK 包或者接收到新的编码块的编码包,立即停止当前编码块编码数据包的发送.

4 仿真结果与分析

为了分析和比较 CFACK 和 CCACK 的性能表现,我们采用 NS2 的扩展 Nsclick[16−18]进行了仿真测试.

4.1 仿真环境

仿真实验基于对数常态跟踪传播模型[19].假设 d_{ij} 和 p_{ij} 分别表示节点 i 与节点 j 之间的距离以及链路的包

© 中国科学院软件研究所 http://www.jos.org.cn

到达率,p_{ij} 可以近似地表示成 d_{ij} 的函数如下:

$$p_{ij}=\begin{cases}1-\left(\left(\dfrac{d_{ij}}{R}\right)^{2\beta}\right)\times\dfrac{1}{2}, & \text{if } d_{ij}\leqslant R\\ \left(\left(\dfrac{2R-d_{ij}}{R}\right)^{2\beta}\right)\times\dfrac{1}{2}, & \text{if } R<d_{ij}\leqslant 2R\\ 0, & \text{otherwise}\end{cases} \tag{6}$$

其中,β是功率衰减因子,仿真中β取值为 2.上述模型中,当两个节点间的距离 d_{ij} 为 R 时,链路丢包率 P_{ij} 为 0.5.因此,R 取值为最大通信距离的一半.

仿真中,随机地把 50 个静态节点部署到 1000m×1000m 区域构成测试网络.模拟环境是基于标准的 802.11b.节点的通信范围为 250m,载波监听范围为 550m.我们设置 R=125m,当 $d_{ij}\geqslant 2R$ 时,p_{ij}=0.所有的路由协议固定操作在比特率为 11Mb/s 的链路上.包的负载大小为 1.4KB,包头的大小取决于路由协议本身.编码块的大小固定为 32.我们把 RTS/CTS 握手协议设置为不启用.源节点传输 17MB 大小的文件到目标节点.在模拟实验中,只有当两个节点之间的链路到达率不低于 0.1 时,才被当作邻居节点.

4.2 仿真结果分析

在测试中,我们随机生成了 110 种网络拓扑,每种网络拓扑下运行 12 次,记录了 CCACK 和 CFACK 两个协议在每种拓扑下的平均吞吐量.图 7 是 CFACK 和 CCACK 在 110 种拓扑中平均吞吐量的累积分布图.从图中我们可以看出,CFACK 的平均吞吐量优于 CCACK.以吞吐量为 100kbps 为例,CCACK 机制中吞吐量大于 100kbps 的情况约为 20%,而 CFACK 机制得到的吞吐量大于 100kbps 的情况约为 40%.所有情况求平均值,CCACK 和 CFACK 的平均吞吐量分别为 69.62kbps 和 100.697kbps.

图 8 是 110 种随机拓扑中 CFACK 对于 CCACK 的相对吞吐量提高量α的累积分布图,定义:

$$\alpha_i=\frac{T^i_{\text{CFACK}}-T^i_{\text{CCACK}}}{T^i_{\text{CCACK}}}\times 100\%,$$

其中,T^i_{CCACK} 和 T^i_{CFACK} 分别是 CCACK 和 CFACK 在第 i 种随机拓扑的平均吞吐量.在测试的 110 种拓扑中,其中的 107 个拓扑中,CFACK 的性能优于 CCACK.CFACK 相对 CCACK 的平均吞吐量提高率为 72.2%.

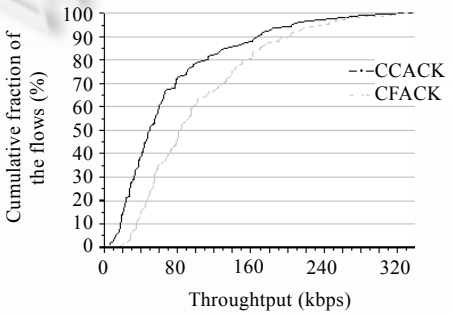


Fig.7 Cumulative distribution function of throughputs achieved with CCACK and CFACK

图 7 CCACK 与 CFACK 平均吞吐量累积分布图

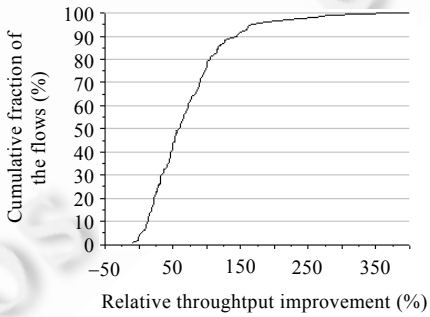


Fig.8 Cumulative distribution function of relative throughput improvement of CFACK over CCACK

图 8 CFACK 相对 CCACK 平均吞吐量提高量α累积分布图

图 9 所示为在测试的 110 种随机拓扑中 CFACK 相对 CCACK 的吞吐量提高量,从图中我们可以很直观地看出:大部分场景下,CFACK 相对于 CCACK 的提高率在 70%左右.110 种随机拓扑中,只有 3 种拓扑吞吐量提高量为负值.通过分析拓扑图我们发现,这是由于存在少量源节点到目标节点路由单一、跳数很少的场景所致.在这种场景下,源节点到目标节点只存在单条路径路由,在这种情况下,CCACK 的反馈向量都由同一个上一跳节

© 中国科学院软件研究所 http://www.jos.org.cn

点的发送向量产生,所以总能满足正交条件,不会发生漏判情况.同时,由于 CCACK 中单个正交反馈向量一次可以确认多个向量,而单个 CFACK 反馈信息包只能确认两个向量,因此在不发生漏判的场景下,CCACK 确认效率略高于 CFACK.可以看出,在测试的 110 种场景中,最差的情况是 CFACK 吞吐量较 CCACK 降低 11.2%.

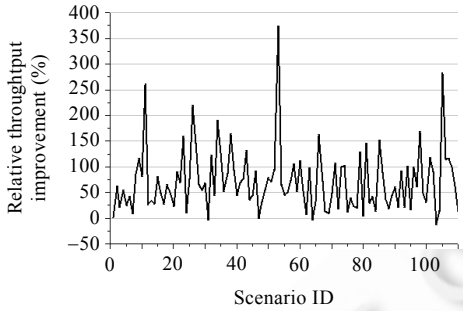
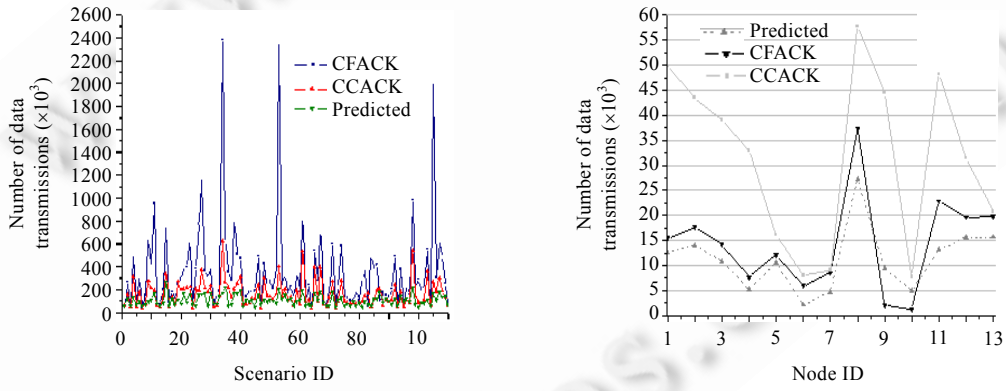


Fig.9 Relative throughput improvement of CFACK over CCACK in each topology

图 9 每个场景 CFACK 相对 CCACK 平均吞吐量提高量

为了进一步说明 CFACK 相对于 CCACK 吞吐量提高这么多的原因,图 10(a)描述了 110 种拓扑中每种拓扑场景下源节点发送 17MB 大小的文件时,CCACK,CFACK 中节点总发包次数与依据预测得到的总的发包次数的比较.图中的 Predicted 是基于 *ETX* 的离线计算算法预测的总发包次数,该值可以表示将一个数据包从源节点成功传输到目标节点的过程中,网络中每个节点上必须进行的发包次数的理论最优值[2].我们观察到,CCACK 在所有 110 种场景中,发包次数绝大多数都多于预测的次数.实际发包数目基本上都是预测数目的两倍,有时甚至达到了 6~7 倍.这意味着正是由于 CCACK 减少了与正交向量正交的编码向量个数,降低了不同下游节点的编码向量的相关性,从而导致一些场景正交向量无法进行判断,产生了许多不必要编码包的发送.



(a) 每个场景节点总发包次数 (b) 其中某个场景每个节点的发包次数

Fig.10 Total number of data transmissions with CCACK, CFACK and predicted number of transmissions

图 10 CCACK,CFACK 以及预测发包总数比较

相反地,CFACK 在所有场景下的发包数目都少于 CCACK,在大部分场景下,CFACK 的发包数目都与预测值比较接近.在一些场景下,甚至会低于预测值.这主要是由于 CFACK 中采用的 2 倍累积反馈确认,提高了判定的正确性,导致了冗余发送的减少.正是由于 CFACK 相对 CCACK 减少了节点的发包次数,所以能够大幅度提高网络的吞吐量.

图 10(b)是某场景下转发节点发包数目的分布图.节点是按照其相对目标节点的 *ETX* 距离排序的.例如,节点 1 是源节点,节点 13 是最靠近目标节点的转发节点.在下游节点接收到足够的新颖信息后,由于 CFACK 能够

© 中国科学院软件研究所 http://www.jos.org.cn

比 CCACK 更加及时,使得上游节点停止发包,因此所有 13 个节点的发包数目都比 CCACK 要少.

4.3 CFACK的开销分析

最后,我们对比 CFACK 和 CCACK 的开销.类似文献[2],对比了 3 种类型的开销:编码开销、存储开销和包头开销.

(1) 编码计算开销

直觉上,CFACK 的编码开销比 CCACK 小很多,因为在发包的时候 CCACK 构建正交向量的过程比 CFACK 构建编码向量的过程要复杂得多.为了验证该直觉,我们测试了 110 个模拟场景中每个节点在发包和收包过程中对每个包的各种操作的复杂性.假设编码块的大小为 32,即编码包中包含了 32 个原始包的信息,每个编码包的负载为 1.4KB.以 GF(2^8)上的乘法运算作为单位开销,表 1 给出了各种运算操作次数的平均值和标准方差值(标记 * 表示发包过程中操作,标记※表示包的接收过程中的操作).

Table 1 Coding overhead of CFACK and CCACK in terms of GF(2^8) multiplication

表 1 CFACK 和 CCACK 在 GF(2^8)上乘法运算编码开销比较

Operation	Avg. (CCACK/CFACK)	Std. Dev. (CCACK/CFACK)
Coded pkt construction in src*	45824/45824	0/0
Coded pkt construction in FNs*	29119/26752	16924/15146
Orthogonal vector construction*	11464/–	3381/–
ACK vector construction*	–/598	–/338
Independence check※	401/407	286/273
H_tests※	435/–	267/–
Rank of H pkts in $B_u \cup B_w$※	314/–	269/–
Rank of M_f※	–/345	–257
Rank of M_{op}※	–/362	–/264

在 CFACK 和 CCACK 中,发包过程的计算开销包括线性编码包的计算产生和反馈信息的产生两部分.前者的计算开销基本是相同的;但从后者反馈信息的产生来看,CCACK 在构建正交向量时,平均需要进行 11 464 次乘法运算,而 CFACK 产生一个编码向量,平均只需进行 598 次乘法运算.所以,CFACK 的开销明显小于 CCACK.测试结果表明:在中间转发节点(FNs)的发包过程的计算开销上,CFACK 的计算量比 CCACK 降低了 48.4%;即使在源节点上,CFACK 的计算量也比 CCACK 降低了 23.4%.

在收包操作中,CCACK 中 *H_tests* 和判断 $B_u \cup B_w$ 的秩时需要乘法运算分别是 435 和 314 次,而 CFACK 判断 M_f 和 M_{op} 的秩分别需要 345 和 362 次乘法运算.所以,CFACK 和 CCACK 的收包操作开销差不多.

(2) 存储开销

与 MORE 一样,CCACK 和 CFACK 的路由器分别在 B_v 和 M_u 缓存中对流的新数据包进行操作,而且都需要一个 64KB 大小的查询表来减少 GF(2^8)乘法运算开销.包的大小为 1400bytes,那么 B_v 和 M_u 的大小都为 44.8KB.在 CCACK 机制中,由于 B_u 和 B_w 中的向量数目与链路状况和数据发送有关,因此在 CCACK 的实现上,设置的存储开销是 2×5×32×32=10KB[12].而 CFACK 中 M_f 和 M_{op} 总共只需 2×32×32=2KB 固定大小的存储空间,因为它们保存的都只是新的编码向量.

(3) 包头开销

CFACK 和 CCACK 在实现过程中同样在数据包中加入了一个包含 K 个分量的 ACK 向量,其中,K 为编码块的大小,每个分量是 8 位.实验中,K 取值 32.在实现过程中,CCACK 和 CFACK 的包头大小是一样的.

5 结束语

本文分析指出:基于正交向量反馈的 CCACK 机制尽管减少了误判概率,但存在较高的漏判概率.我们提出了一种基于累积编码系数反馈确认的网络编码机会路由协议 CFACK,该协议利用侦听到的下游节点发送编码包中的编码向量来进行累积确认,在确保收听到下游编码包的情况下,该机制不存在误判和漏判.仿真结果表明:CFACK 相对于 CCACK,显著提高了网络的吞吐量,平均提高率达到了 72.2%.同时,在编码、存储和包头的开

© 中国科学院软件研究所 http://www.jos.org.cn

销相对 CCACK 要小.

References:

[1] Chachulski S, Jennings M, Katti S, Katabi D. Trading structure for randomness in wireless opportunistic routing. ACM SIGCOMM Computer Communication Review, 2007,37(4):169−180. [doi: 10.1145/1282427.1282400]

[2] Biswas S, Morris R. ExOR: Opportunistic multi-hop routing for wireless networks. ACM SIGCOMM Computer Communication Review, 2005,35(4):133−144. [doi: 10.1145/1090191.1080108]

[3] De Couto DS, Aguayo D, Bicket J, Morris R. A high-throughput path metric for multi-hop wireless routing. Wireless Networks, 2005,11(4):419−434. [doi: 10.1007/s11276-005-1766-z]

[4] Lin Y, Li B, Liang B. CodeOR: Opportunistic routing in wireless mesh networks with segmented network coding. In: Sarac K, ed. Proc. of the ICNP 2008. IEEE, 2008. 13−22. [doi: 10.1109/ICNP.2008.4697020]

[5] Lin Y, Liang B, Li B. SlideOR: Online opportunistic network coding in wireless mesh networks. In: Boutaba R, Chatterjee M, eds. Proc. of the IEEE INFOCOM 2010. IEEE, 2010. 1−5. [doi: 10.1109/INFCOM.2010.5462249]

[6] Katti S, Katabi D, Balakrishnan B, Medard M. Symbol-Level network coding for wireless mesh networks. ACM SIGCOMM Computer Communication Review, 2008,38(4):401−412. [doi: 10.1145/1402946.1403004]

[7] Chen C, Dong C, Wu F, Wang H. Improving unsegmented network coding for opportunistic routing in wireless mesh network. In: Sibille A, ed. Proc. of the WCNC 2012. IEEE, 2012. 1847−1852. [doi: 10.1109/WCNC.2012.6214086]

[8] Li P, Guo S, Yu S, Vasilakos AV. CodePipe: An opportunistic feeding and routing protocol for reliable multicast with pipelined network coding. In: Berry R, Wang C, McNair J, eds. Proc. of the IEEE INFOCOM 2012. IEEE, 2012. 100−108. [doi: 10.1109/INFCOM.2012.6195456]

[9] Bicket J, Aguayo D, Biswas S, Morris R. Architecture and evaluation of an unplanned 802.11b mesh network. In: Proc. of the ACM MOBICOM 2005. New York: ACM Press, 2005. 31−42. [doi: 10.1145/1080829.1080833]

[10] Zhang X, Li B. Dice: A game theoretic framework for wireless multipath network coding. In: Proc. of the ACM MobiHoc 2008. New York: ACM Press, 2008. 293−302. [doi: 10.1145/1374618.1374658]

[11] Lee PPC, Misra V, Rubenstein D. On the robustness of wireless opportunistic routing toward inaccurate link-level measurements. In: Das D, ed. Proc. of the COMSNETS 2010. IEEE, 2010. 1−10. [doi: 10.1109/COMSNETS.2010.5431992]

[12] Koutsonikolas D, Wang CC, Hu YC. CCACK: Efficient network coding based opportunistic routing through cumulative coded acknowledgments. In: Boutaba R, Chatterjee M, eds. Proc. of the IEEE INFOCOM 2010. IEEE, 2010. [doi: 10.1109/INFCOM.2010.5462125]

[13] Aguayo D, Bicket J, Biswas S, Judd G, Morris R. Link-Level measurements from an 802.11b mesh network. ACM SIGCOMM Computer Communication Review, 2004,34(4):121−132. [doi: 10.1145/1030194.1015482]

[14] Reis C, Mahajan R, Rodrig M, Wetherall D, Zahorjan J. Measurement-Based models of delivery and interference in static wireless networks. ACM SIGCOMM Computer Communication Review, 2006,36(4):51−62. [doi: 10.1145/1151659.1159921]

[15] Miu A, Balakrishnan H, Koksal CE. Improving loss resilience with multi-radio diversity in wireless networks. In: Proc. of the ACM MOBICOM 2005. New York: ACM Press, 2005. 16−30. [doi: 10.1145/1080829.1080832]

[16] Letor N, De Cleyn P, Blondia C. Enabling cross layer design: Adding the MadWifi extensions to Nsclick. In: Alouf S, ed. Proc. of the 2nd Int'l Conf. on Performance Evaluation Methodologies and Tools. ICST, 2007. 19.

[17] Neufeld M, Schelle G, Grunwald G. Nsclick user manual. Technical Report, CO 80309, Boulder: University of Colorado, 2003.

[18] Kohler E, Morris R, Chen R, Jannotti J, Kaashoek MF. The click modular router. ACM Trans. on Computer Systems (TOCS), 2000, 18(3):263−297. [doi: 10.1145/354871.354874]

[19] Network simulator-ns-2. http://www.isi.edu/nsnam/ns/

附录:CCACK 机制中漏判概率的理论分析

引理 1. 假设 y,z 是两个具有相同元素个数的向量,X 是一个向量集合,X 中任意向量也与 y,z 具有相同的元素个数.假设 X 与 z 满足 H-正交(即集合 X 中任意向量与 z 满足 H-正交),当向量 $y\in X$ 或当 y 可以由向量空间 X

© 中国科学院软件研究所 http://www.jos.org.cn

线性表示时,y 与 z 满足 H-正交.

证明:因为对于$\forall x_i \in X$,都有 $x_i H_j Z^T=0$,显然有:当 $y \in X$,y 与 Z 满足 H-正交;

而当 $y=\sum_{i=1}^{m}\alpha_i x_i$ (其中,m 为向量空间 X 的大小),依然有 $yH_j \cdot Z=\sum_{i=1}^{m}\alpha_i x_i H_j \cdot Z=0$.命题得证. □

引理 2. 设 X 是向量集合,z 是某个确定向量,X 与 z 满足 H-正交.当任意向量 $y \notin X$ 且 y 不能由 X 线性表示时,y 与 z 满足 H-正交的概率小于 $\left(\frac{1}{2^8}\right)^M$ (假设线性编码系数选取区域为 GF(2^8)有限域,X 中任意向量与向量 y,z 具有相同的元素个数 K,M 为 H 向量的个数).

证明:具备 K 个元素的向量,且元素的值在 GF(2^8)有限域的向量的总数目是$(2^8)^K$.

容易推知:这些向量中能够由向量空间 X 线性表示的向量 q 的总数目为$(2^8)^{R(X)}$,其中,$0 \leqslant R(X) \leqslant K$.

因为 X 与 z 满足 H-正交,由引理 1 可知,所有 q 都满足 $qH_jZ^T=0$.

同样容易推知:对于某个 z 和 H_j,共存在有$(2^8)^{K-1}$ 个向量 p 与 z 满足 $pH_jZ^T=0$.

所以,共有$(2^8)^k-(2^8)^{R(X)}$个向量不能由 X 线性表示,其中,能满足 $pH_jZ^T=0$ 的向量个数是$(2^8)^{k-1}-(2^8)^{R(X)}$.

所以,对于不能有 X 线性表示的向量 y,满足 $yH_jZ^T=0$ 的概率是

$$p=\frac{(256)^{K-1}-(256)^{R(X)}}{(256)^K-(256)^{R(X)}}<\frac{1}{2^8}.$$

因为 H-正交需要同时满足 M 个 $yH_jZ^T=0(1 \leqslant j \leqslant M)$这样的等式,所以 y 与 Z 满足 H-正交的概率小于 $\left(\frac{1}{2^8}\right)^M$.

命题得证. □

引理 3. 令向量 y 能由向量空间 $X=\bigcup_{1 \leqslant i \leqslant m} X_i$ 线性表示,其中,X_i 为 X 的子向量空间,令 $r=R(X), r_i=R(X_i), t_i=\min\left(r_i,\left(\frac{K}{M}-1\right)\right)$,则 y 能由子向量空间 X_i 中任意包含 r_i 个向量的部分向量空间线性表示的概率 P 为

$$P=\sum_{X_i \in X} C_{r_i}^{t_i}\left(\frac{1}{256}\right)^{r-t_i}.$$

证明:y 可以由向量空间 X 线性表示,则 y 向量共有$(256)^r$种可能.

能由其中任意 X_i 中任意 t_i 个向量线性表示的可能为 $C_{r_i}^{t_i}(256)^{t_i}$.

$$P=\frac{\sum_{X_i \in X} C_{r_i}^{t_i}(256)^{t_i}}{(256)^r}=\sum_{X_i \in X} C_{r_i}^{t_i}\left(\frac{1}{256}\right)^{r-t_i}.$$ □

接下来,我们由引理 1~引理 3 来说明 CCACK 机制存在漏判的问题.

定理 1-1. 采用 CCACK 机制,发送路径上任意节点 i 的发送向量空间 B_w^i 中的向量与 i 的下一跳节点产生的某一个反馈向量 Z 满足 H-正交的概率大于 $\frac{1}{M}-\frac{1}{K}$.其中,M 是反馈正交向量产生时采用的 H 矩阵个数,K 是编码块的大小.

证明:设 i 的下一跳节点集合为 V,因为 B_w^i 中的任意向量 $y \in B_u^j (j \in V)$,所以只要任意 j 产生 Z 时,所取到的部分接收向量空间中包含了 y,则由引理 1 可知,y 与 Z 满足 H-正交.

令任意 j 选取 $\min\left(r_j,\frac{K}{M}-1\right)$ 个向量中包含 y 的概率为 ω,则 $\omega=\frac{\min\left(r_j,\frac{K}{M}-1\right)}{r_j}$.

因为 r_j 的最大值为 K,所以 $\omega>\frac{1}{M}-\frac{1}{K}=\frac{K-M}{MK}$.

推论 1-1. 任意节点的发送向量,最多只需要收到 $\frac{MK}{K-M}$ 个来自其下一跳节点的反馈 Z,就能得到确认.

© 中国科学院软件研究所 http://www.jos.org.cn

当 K=32,M=4 时,最多只需收到来自下一跳节点的 5 个 Z 向量,就能判定 H-正交,从而得到确认.

定理 1-2. 令节点 i 的下一跳节点集合为 V,采用 CCACK 机制,发送路径上任意节点 i 的 B_u^i 中存在的某个向量 y 能被 V 的整个接收向量空间 $\bigcup_{j\in V} B_u^j$ 线性表示,并且 $y\notin \bigcup_{j\in V} B_u^j$,则 y 与 V 产生的所有可能的 Z 中的任意一个满足 H-正交的概率是

$$\sum_{j\in V} C_{r_j}^{t_j}\left(\frac{1}{256}\right)^{r-t_j}+\left(1-\sum_{j\in V} C_{r_j}^{t_j}\left(\frac{1}{256}\right)^{r-t_j}\right)\left(\frac{1}{256}\right)^{M},$$

其中,$r=Rank\left(\bigcup_{j\in V} B_u^j\right)$,$r_j=Rank(B_u^j)$,$t_j=\min\left(r_j,\frac{K}{M}-1\right)$,$M$ 是 H_j 的个数.

证明:设 B_u^i 中存在向量 y,能由 $\bigcup_{j\in V} B_u^j$ 线性表示,由引理 3 可知,y 可由 B_u^j 中的任意 t_j 个部分向量线性表示的概率为 $p=\sum_{j\in V} C_{r_j}^{t_j}\left(\frac{1}{256}\right)^{r-t_j}$,其中,$r_j=Rank(B_u^j)$,$t_j=\min\left(r_j,\frac{K}{M}-1\right)$.

依据引理 1,这种情况下,y 能与 $j\in V$ 产生的 Z 满足 H-正交.

显然,y 不能由 B_u^j 中的任意 t_j 个部分向量线性表示的概率为 1–p.依据引理 2,这部分向量与 Z 满足 H-正交的概率即为误判的概率 $\left(\frac{1}{2^8}\right)^M$.

所以,y 整体满足 H-正交的概率是 $\mu=p+(1-p)\left(\frac{1}{2^8}\right)^M$.得证. □

为了衡量μ的大小,令 V 中有 5 个节点,当 K=32,M=4,r=32,r_j=10 时:

$$\mu=5\times C_{10}^{7}\times\left(\frac{1}{2^8}\right)^{32}+\left(1-5\times C_{10}^{7}\times\left(\frac{1}{2^8}\right)^{32}\right)\left(\frac{1}{2^8}\right)^{4}\approx\left(\frac{1}{2^8}\right)^{4}\approx 2.33\times10^{-10}.$$

可见,可以得到 H-正交的概率是非常低的.

推论 1-2. 任意节点的存在于 B_u 中但不包含在下一跳节点接收向量中的向量,即使已能被下一跳节点的接收向量空间表示,也只能以很小的概率与反馈向量 z 满足 H-正交,即,将以近似于 100%的概率出现漏判.

由定理 1-1、定理 1-2、推论 1-1、推论 1-2,定理 1 可以得证.

王伟平(1969－),女,江苏苏州人,博士,教授,博士生导师,主要研究领域为网络编码,网络优化理论,网络信息安全.
E-mail: wpwang@csu.edu.cn

陈小专(1985－),男,硕士,主要研究领域为基于网络编码的路由算法.
E-mail: xzchen@csu.edu.cn

鲁鸣鸣(1978－),男,博士,副教授,CCF 会员,主要研究领域为无线网络,移动计算.
E-mail: mingminglu@csu.edu.cn

王建新(1969－),男,博士,教授,博士生导师,CCF 高级会员,主要研究领域为计算机算法,网络优化理论,生物信息学.
E-mail: jxwang@mail.csu.edu.cn

© 中国科学院软件研究所 http://www.jos.org.cn