Spring 5-31-2011

Efficient packet delivery in modern communication networks

Nan Wang
New Jersey Institute of Technology

Copyright Warning & Restrictions

The copyright law of the United States (Title 17, United States Code) governs the making of photocopies or other reproductions of copyrighted material.

Under certain conditions specified in the law, libraries and archives are authorized to furnish a photocopy or other reproduction. One of these specified conditions is that the photocopy or reproduction is not to be "used for any purpose other than private study, scholarship, or research." If a, user makes a request for, or later uses, a photocopy or reproduction for purposes in excess of "fair use" that user may be liable for copyright infringement,

This institution reserves the right to refuse to accept a copying order if, in its judgment, fulfillment of the order would involve violation of copyright law.

Please Note: The author retains the copyright while the New Jersey Institute of Technology reserves the right to distribute this thesis or dissertation

Printing note: If you do not wish to print this page, then select "Pages from: first page # to: last page #" on the print dialog screen

ABSTRACT

EFFICIENT PACKET DELIVERY IN MODERN COMMUNICATION NETWORKS

by
Nan Wang

Modern communication networks are often designed for diverse applications, such as voice, data and video. Packet-switching is often adapted in today's networks to transmit multiple types of traffic. In packet-switching networks, network performance is directly affected by how the networks handle their packets. This work addresses the packet-handling issues from the following two aspects: Quality of Service (QoS) and network coding.

QoS has been a well-addressed issue in the study of IP-based networks. Generally, nodes in a network need to be informed of the state of each communication link in order to make intelligent decisions to route packets according to their QoS demands. The link state can, however, change rapidly in a network; therefore, nodes would have to receive frequent link state updates in order to maintain the latest link state information at all times. Frequent link state updating is resource-consuming and hence impractical in network design. Therefore, there is a trade-off between the link state updating frequency and the QoS routing performance. It is necessary to design a link state update algorithm that utilizes less frequent link state updates to achieve a high degree of satisfaction in QoS performance. The first part of this work addresses this link state update problem and provides two solutions: ROSE and Smart Packet Marking. ROSE is a class-based link state update algorithm, in which the class boundaries are

designed based on the statistical data of users' QoS requests. By doing so, link state update is triggered only when certain necessary conditions are met. For example, if the available bandwidth of a link is fluctuating within a range that is higher than the highest possible bandwidth request, there is no need to update the state of this link. Smart Packet Marking utilizes a similar concept like ROSE, except that the link state information is carried in the probing packet sent in conjunction with each connection request instead of through link state updates.

The second part of this work addresses the packet-handling issue by means of network coding. Instead of the traditional store-and-forward approach, network coding allows intermediate nodes in a multi-hop path to code multiple packets into one in order to reduce bandwidth consumption. The coded packet can later be decoded by its recipients to retrieve the original plain packet. Network coding is found to be beneficial in many network applications. This dissertation makes contributions in network coding in two areas: peer-to-peer file sharing and wireless ad-hoc networks. The benefit of network coding in peer-to-peer file sharing networks is analyzed, and a network coding algorithm – Downloader-Initiated Random Linear Network Coding (DRLNC) – is proposed. DLRNC shifts the coding decision from the seeders to the leechers; by doing so it solves the "collision" problem without increasing the field size. In wireless network coding, this work addresses the implementation difficulty pertaining to MAC layer scheduling. To achieve the ideal network coding gain in wireless networks, it requires perfect MAC layer scheduling. This dissertation first provides an algorithm to solve the ideal-case MAC layer scheduling problem. Since the ideal MAC layer schedule is often difficult to realize,

a practical approach is then proposed to increase the network coding performance by modifying the ACK packets in the 802.11 MAC.

EFFICIENT PACKET DELIVERY IN MODERN COMMUNICATION NETWORKS

by
Nan Wang

A Dissertation
Submitted to the Faculty of
New Jersey Institute of Technology
in Partial Fulfillment of the Requirements for the Degree of
Doctor of Philosophy in Electrical Engineering

Department of Electrical and Computer Engineering

May 2011

APPROVAL PAGE

EFFICIENT PACKET DELIVERY IN MODERN COMMUNICATION NETWORKS

Nan Wang

Dr. Nirwan Ansari, Dissertation Advisor — Date
Professor of Electrical and Computer Engineering, NJIT

Dr. Reza Curtmola, Committee Member — Date
Assistant Professor of Computer Science, NJIT

Dr. Sui-hoi E. Hou, Committee Member — Date
Associate Professor of Electrical and Computer Engineering, NJIT

Dr. Roberto Rojas-Cessa, Committee Member — Date
Associate Professor of Electrical and Computer Engineering, NJIT

Dr. Osvaldo Simeone, Committee Member — Date
Assistant Professor of Electrical and Computer Engineering, NJIT

BIOGRAPHICAL SKETCH

Author: Nan Wang

Degree: Doctor of Philosophy

Date: May 2011

Undergraduate and Graduate Education:

- Doctor of Philosophy in Electrical Engineering
  New Jersey Institute of Technology, Newark, NJ, 2011
- Master of Science in Electrical Engineering,
  New Jersey Institute of Technology, Newark, NJ, 1998
- Bachelor of Science in Electrical Engineering,
  National Chen-Kung University, Tainan, Taiwan, 1995

Major: Electrical Engineering

Presentations and Publications:

Publications:

N. Ansari, G. Cheng and N. Wang, "ROSE II for Updating Additive Link State Information," *Proc. IEEE International Conference on Communications (ICC 2006)*, Istanbul, Turkey, June 11-15, 2006, pp. 676-680.

N. Wang, N. Ansari, and R. Rojas-cessa, "Improving the Accuracy of EEAC-SV with Smart Packet Marking," *Proc. IEEE GLOBECOM 2006*, San Francisco, Nov. 27 – Dec. 1, 2006.

N. Ansari, G. Cheng, and N. Wang, "Routing-Oriented update SchEme (ROSE) for Link State Updating," *IEEE Transactions on Communications*, Vol. 56, No. 6, pp. 948-956, June 2008..

N. Wang and N. Ansari, "Identifying the Network Coding Opportunity," *Proc. 2010 IEEE Sarnoff Symposium*, Princeton, NJ, Apr. 12 -14, 2010.

N. Wang and N. Ansari, "Downloader-Initiated Random Linear Network Coding for P2P File Sharing," *IEEE Systems Journal*, Vol. 5, No.1, pp. 61-69, Mar. 2011.

Presentations:

N. Wang, N. Ansari, and R. Rojas-cessa, "Improving the Accuracy of EEAC-SV with Smart Packet Marking," presented at GLOBECOM '06, San Francisco, CA, November 27-December 1, 2006.

N. Wang and N. Ansari, "Identifying the Network Coding Opportunity," presented at IEEE Sarnoff Symposium 2010, Princeton, NJ, USA. April 12-14, 2010.

ACKNOWLEDGMENT

I would like to take this opportunity to express my thanks to those who have helped me with various aspects of conducting research and the writing of this dissertation.

First and foremost, I would like to thank my advisor, Dr. Nirwan Ansari, for his support, encouragement, and endless source of ideas. His breadth of knowledge, his enthusiasm for research, and his vision for the future have guided me. I am especially grateful for the countless hours he has spent with me, discussing everything from research to reading my papers. My life has been enriched professionally, intellectually, and personally by working with Dr. Ansari.

I also would like to acknowledge my other committee members, Dr. Reza Curtmola, Dr. Edwin Hou, Dr. Roberto Rojas-Cessa, and Dr. Osvaldo Simeone for serving on my dissertation committee and for their help and advices to my work.

A special thank to Dr. Robert Li, director of the Institute of Network Coding at the Chinese University of Hong Kong, for his willingness of sharing his research ideas on network coding.

I am greatly thankful to my labmate, Gang Cheng, for his collaboration during this research and contributions to this dissertation. I have had the good fortune to meet many helpful and friendly people during my residence at NJIT: Si Yin, Jingjing Zhang, Tao Han, Thomas Lo, and many other names I cannot put here one by one.

TABLE OF CONTENTS

TABLE OF CONTENTS
(Continued)

Chapter **Page**

TABLE OF CONTENTS
(Continued)

Chapter **Page**

TABLE OF CONTENTS
(Continued)

Chapter **Page**

TABLE OF CONTENTS
(Continued)

Chapter **Page**

CHAPTER 1

INTRODUCTION

Today's networks are designed for diverse applications such as voice, data, and video. It is observed that today's networks are evolving towards the following two directions: first, packet switching is replacing the old-fashioned circuit switching, and second, wireless communication devices are gaining popularity with wider applications.

Packet switching is the key to integrate various types of traffic onto one same network while still maintaining efficient use of network resources. The Internet Protocol (IP) has become the de facto packet-switching network layer protocol of choice, due to the great popularity of the Internet. Because of its success, more networks are adopting IP as the network layer protocol, which also implies that more networks are becoming packet-switching. For example, newly designed networks, such as the 4G wireless network, are starting to adopt the Internet Protocol for the sake of interoperability with the existing networks.

Wireless networks have become new market favors because they extend the boundaries of traditional wired networks, provide greater accessibility to users, and support at least some degree of mobility. Widely deployed wireless networks include cellular networks and WiFi networks. In addition, a wireless ad hoc network can further provide easy and fast deployment because it does not require physical wired connections among its nodes. These are great features for sensor networks and disaster-recovery area networks. With all these benefits offered by wireless networks, wireless networks will gain greater penetration in people's lives in the future.

New challenges arise as today's networks evolve. For starters, traffics from various applications, such as voice, video or data, have different characteristics, and therefore require different quality of service (QoS). When engineers attempt to put all these traffics in the same network, can it be ensured that the network can provide proper QoS to each type of traffic? Second, in a wireless network, its access medium – radio frequency waves – is often a limited resource subject to bandwidth regulations and interferences. Can the network bandwidth be utilized efficiently while coping with interference?

The list of challenges is lengthy. The works presented in this dissertation focus on the aforementioned two challenges. The approaches are to tackle these problems from the packet level. Since all traffics are delivered in packets, it is how the switches and routers handle packets that essentially determines the performance of the networks. The contributions of this dissertation consist of two parts. The first part is related to link state updates. Link state update algorithms determine how the link state information is disseminated to network routers; based on this information, the routers decide how the packets should be routed. Therefore, a good link state update algorithm is essential for good QoS performance. This dissertation offers new solutions for link state update that increase the accuracy of updating link state information. The second part of the contributions is related to network coding, particularly, in peer-to-peer networks and wireless networks. In peer-to-peer network, the findings and solutions of how network coding improves the efficiency of a peer-to-peer file sharing network is presented. In the area of wireless network coding, the issues of MAC layer scheduling is addressed. Wireless network coding exploits the broadcast nature of wireless communications to

make more efficient use of wireless medium, and therefore improve overall throughput of the network. A major challenge of wireless network coding is the MAC layer scheduling. The reason that network coding increases the throughput is that it allows multiple packets to be encoded together, thus reducing the number of transmissions. Ideally, the MAC layer must schedule the packet transmissions in an order that takes full advantages of all coding opportunities. This dissertation provides an algorithm to find the ideal MAC layer schedule, given the network topology and route information. The ideal schedule, however, is not easy to implement in practice. Therefore, following the ideal solution, a MAC layer scheduling scheme that utilizes the ACK packets to inform the neighboring nodes of potential coding opportunities and gives the nodes with matching packets higher priorities to transmit those packets is proposed. This proposed approach increases the chance of packet encodings (as compared to the traditional contention-based MAC layer), and thus improves the practical performance of wireless network coding.

The rest of this dissertation is organized as follows. Chapters 2, 3, and 4 are dedicated to link state update algorithms pertaining to quality of service; Chapter 5 presents the proposed method of utilizing network coding in peer-to-peer file sharing networks; Chapter 6 and 7 discusses network coding in wireless network environments; and finally, Chapter 8 presents the conclusions and proposes future research directions.

CHAPTER 2

UPDATING LINK STATE INFORMATION

2.1 Introduction

In order to provide Quality of Service (QoS) in the broadband integrated network, the network has to be aware of all the QoS parameters in every node. Based on this knowledge, the network can determine the best routing strategy to find the path that satisfies the QoS requirements of each connection and achieves the overall network resource efficiency [1]. In order to provide the knowledge of all the QoS parameters in each node, each node itself must employ some scheme to report its own QoS parameters; this process is referred to as “link state update”. Since the network resource is precious (e.g., limited bandwidth), it is not practical to have all nodes reporting precise QoS parameters all the time; this would result in a tremendous overhead. Therefore, an effective link state update algorithm is necessary to provision QoS. Link state update determines the behavior of how each node updates its status to the entire network, including when to update and what to update.

A widely used link state update protocol, OSPF [2], recommends the link state to be updated once every 30 minutes. However, because of the highly dynamic nature of the traffic, updating in such a long time interval will result in stale/outdated link state parameters. This will compromise the efficiency of QoS routing. Several other link state update policies, such as threshold, equal class and exponential class based update policies [3], have been proposed. In the threshold policy, an update is triggered when the difference between the current value and the previously updated value of a certain

parameter exceeds a threshold. That is, given a threshold value τ, an update occurs when $|b_c - b_0| > \tau$, where b_0 is the previously updated value and b_c is the current value of a QoS parameter. In the equal class and the exponential class based update policies, the values of QoS parameters are divided into classes. An update is triggered when the current value of a QoS parameter changes from one class to another. For example, in a two-class situation, if the range (interval) of the first class is $(0, b_1)$, and the range of the second class is (b_1, b_2), then an update will happen when b_c changes from $0 < b_c < b_1$ to $b_1 < b_c < b_2$, or vice versa. What separates the equal class based link state update policy from the exponential class based link state policy is the choice of the boundaries, or in other words, the partitioning of each class. In the equal class based link state update, the class of a QoS parameter is partitioned into equal-sized intervals, for example, $(0, B)$ $(B, 2B)$, $(2B, 3B)$, …, etc.. In the exponential class based update, the QoS parameter is partitioned into unequal-sized ranges, $(0, B)$, $(B, (f+1)B)$, $((f+1)B, (f^2+f+1)B)$, …, whose sizes grow geometrically by a factor of *f*, where *B* is a predefined constant.

No matter which link state update policy is adopted by the network, it is unavoidable that the QoS parameters of each node known to the entire network might not be exactly accurate at any given time, due to the staleness and coarse classes. As a result, false routing is inevitable. Some works [4], [5], [6] have been done trying to reduce the impact of stale or inaccurate link state information. Recently, a probability-based link state update referred to as “Routing-Oriented update SchEme” (ROSE) has been proposed by Cheng and Ansari [7] to minimize the probability of false routing. As reviewed above, most works reported in literature only consider concave link state metrics, e.g., bandwidth. Nonetheless, it is not enough to provide an efficient solution by

studying the concave metrics alone because of the different nature of additive and concave link state metrics. Based on the previous work, ROSE [7], in which concave metrics were considered, an effective link state update policy for additive metrics is formulated, referred to as ROSE II. The key difference between ROSE II and previously reported works lies in the fact that ROSE II takes the property of additive QoS metrics into account: ROSE II is proposed for the purpose of efficient QoS routing under additive constraints. As a result, it achieves the goal of providing accurate link state information with a low protocol overhead. Via theoretical analysis and simulations, it is shown that ROSE II greatly outperforms the state of the art. The rest of this chapter is organized as follows. Section 2.2 describes the properties of additive constraints. Then, Section 2.3 describes the proposed efficient link state information update scheme, ROSE II. The simulation results are presented in Section 2.4. Finally, concluding remarks are given in Section 2.5.

2.2 Properties of Additive Constraints

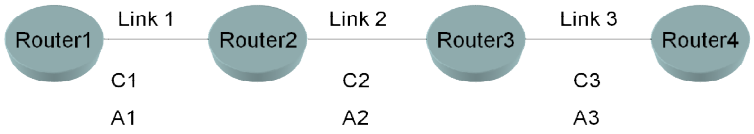
Most of the QoS constraints can be categorized into the following three types: concave, additive, and multiplicative. Multiplicative constraints can be converted into additive constraints by using the logarithm operator. Therefore, only concave and additive constraints are considered in QoS routing. A concave constraint works as follows: in the case of a multi-link end-to-end path, as long as the smallest QoS parameter among all the links is larger than the corresponding QoS constraint, then this path is considered to have met the QoS requirement. Bandwidth is a typical example of a concave constraint. An additive constraint works as follows: in the case of a multi-link end-to-end path, the sum

of all the QoS parameters along the path has to be less than the corresponding QoS constraint for this path to be considered to have met the QoS requirement. Delay is a typical example of an additive constraint. In Figure 2.1, the path consists of 3 links: Link 1, Link 2, and Link 3. Each of these links has a concave QoS parameter C_1, C_2, and C_3, respectively, and an additive QoS parameter A_1, A_2, and A_3, respectively. If a connection imposes QoS constraints C0 and A0, then the path is deemed acceptable if $\min\{C_1, C_2, C_3\} \geq C_0$, and $A_1 + A_2 + A_3 \leq A_0$.

One of the special characteristics of an additive constraint is that, from a per-link point of view, the QoS requirement of each link is related to the QoS behavior of all the other links in the same path. Consider Link 2 in Figure 2.1 as an example: Link 2 will be accepted if $A_2 \leq A_0 - (A_1 + A_3)$. Similarly, in an *m*-link path, a link among these *m* links, say Link n $(n \leq m)$, is acceptable if

$$A_n \leq A_0 - \sum_{j=1, j \neq n}^{m} A_j \tag{2.1}$$

Because of such special characteristic, a single link cannot make decision whether to accept or reject a connection purely based on its own additive link state metrics. Those aforementioned current link state update schemes (threshold, equal class, and exponential class updates) do not take this property into consideration; therefore, they are not optimized for handling additive QoS metrics. ROSE II does take such property into consideration which enables it to manage additive metrics more efficiently, as it will be illustrated in the next section.



Concave QoS parameters of Link 1, 2, and 3 = C1, C2, and C3.
Additive QoS parameters of Link 1, 2, and 3 = A1, A2, and A3,

The path meets concave constraint if *min{C1, C2, C3}>=C0.*
The path meets additive constraint if *A1+A2+A3<=A0.*
C0 and *A0* are the imposed concave and additive constraints.

Figure 2.1 Illustration of concave and additive constraints.

2.3 Stochastic Approach of Link State Update on Additive QoS Parameters

Consider a network composed of core and edge nodes. Assume that the edge nodes make the decision of accepting or rejecting a connection and are responsible for path finding. False routing occurs when an incorrect decision is made by the edge nodes owing to the inaccurate link state information: a connection request is rejected that should be accommodated by the network – also known as false negative – or a connection request should be declined but is accepted – also known as false positive. Generally, in a class-based link state update scheme, when each link updates its QoS parameter using k classes, the ranges of the classes can be expressed as: $[0, B_1]$, $[B_1, B_2]$, $[B_2, B_3]$, …, $[B_{k-1}, B_k]$, where 0 and B_1 ~ B_k are the boundaries of classes. (Without loss of generality, it is assumed the class boundaries of all links are identical.) Let $B_1^{adv}, B_2^{adv}, \ldots, B_k^{adv}$ be the advertised value of the QoS parameter when its actual value falls into $[0, B_1]$, $[B_1, B_2]$, …, $[B_{k-1}, B_k]$, respectively. In other words, the exact QoS measurement is quantized into coarse ranges. Taking the delay parameter as an example,

instead of reporting the exact delay measurement, a link reports its quantized value B_2^{adv} (for instance) if its actual delay measurement falls in between B_1 and B_2. Owing to the quantization, it is inevitable that the edge nodes will obtain inaccurate link state information and result in false routing. False routing results in degraded user satisfaction and/or waste of network resources. Link state update scheme must be designed to reduce the occurrences of false routing.

Continuing from equation (2.1), assume that for each link j ($j \leq m$), its additive QoS parameter A_j is always within a range of $[0, A_j^{MAX}]$. This is a reasonable assumption because, using the delay parameter as an example, the delay of each link can only reach a certain amount (i.e., queue full) before the link will reject any connection attempts all together. Without loss of generality, it is further assumed that $A_j^{MAX} = A^{MAX} \ \forall j$, that is, the maximum value of A_j is the same for all links. In addition, the requested QoS constraint A0 can also be assumed to have a finite range of values. Therefore, on a per-link basis, the decision of whether to accept or reject the link depends on whether this link's current QoS parameter A_n is less than or greater than $\left(A_0 - \sum_{j=1, j\neq m}^{m} A_j\right)$.

Let $C = \left(A_0 - \sum_{j=1, j\neq m}^{m} A_j\right)$, which will be referred to as the "accept/reject criterion" throughout the rest of this chapter, then the decision making is essentially:

$$\begin{cases} Accept\ if\ A_n \leq C \\ Reject\ if\ A_n > C \end{cases} \tag{2.2}$$

By treating all the QoS parameters that C is composed of as random variables (that is, from A_0 to A_m, except A_n), C is simply the sum of m random variables.

According to the central limit theorem, when *m* becomes large, the accept/reject criterion of a link can be approximated by a normal distribution.

Based on equation (2.2), the problem of updating additive link state information can be greatly simplified in the sense that the probability density function (pdf) of the accept/reject criterion *C* can be estimated. The pdf of *C* is solely related to the average hop count, A^{MAX} and A_0, of which the information is readily available to estimate the pdf. With the estimated pdf, a class based update policy can be designed to minimize the probability of false routing. Let B_{adv}^nbe the advertised value of a QoS parameter by link *n* ($B_{adv}^n \in \{B_1^{adv}, \dots, B_k^{adv}\}$, where *k* is the number of classes). With the above class-based update scheme, according to equation (2.2), a link *n* will be accepted if $B_{adv}^n \leq C$, and will be rejected otherwise. Therefore, a false positive will occur when $B_{adv}^n \leq C$ but $A_n > C$, where A_n is the actual value of the QoS parameter of link *n*. On the other hand, a false negative will occur when $B_{adv}^n > C$ but $A_n \leq C$.

Assume that when Link *n* reports a QoS parameter $B_{adv}^n = B_i^{adv}$, the actual parameter A_n is randomly distributed from B_{i-1} (note $B_0 = 0$) to B_i with a pdf $p(x)$. Given a value $C = \tau$ ($B_i^{adv} < \tau < B_i$), the probability of false positive can be expressed as:

$$Pr\{false\ positive | C = \tau, B_{adv}^n = B_i^{adv}\} = \int_{\tau}^{B_i} p(x)dx\,, B_i^{adv} < \tau < B_i \qquad (2.3)$$

Equation (2.3) is essentially the probability of A_n falling in between τ and B_i, given that $C = \tau$ and $B_i^{adv} < \tau < B_i$. Let $f_c(\tau)$ be the pdf of the accept/reject criterion C, then equation (2.3) can be expanded as:

$$Pr\{false\ positive | B_{adv}^n = B_i^{adv}\} = \int_{B_i^{adv}}^{B_i} \int_{\tau}^{B_i} p(x)dx\, f_c(\tau)d\tau \tag{2.4}$$

Similarly, given a value $C = \tau$, but $B_{i-1} < \tau < B_i^{adv}$, the false negative probability can be written in the same manner as in equation (2.3):

$$Pr\{false\ negative | C = \tau\} = \int_{B_{i-1}}^{\tau} p(x)dx,\ B_{i-1} < \tau < B_i^{adv},$$

and

$$Pr\{false\ negative | B_{adv}^n = B_i^{adv}\} = \int_{B_{i-1}}^{B_i^{adv}} \int_{B_{i-1}}^{\tau} p(x)dx\, f_c(\tau)d\tau \tag{2.5}$$

Therefore, the probability of false routing (both false positive and negative) when the advertised parameter is B_i^{adv} is then:

$$Pr\{false\ routing | B_{adv}^n = B_i^{adv}\} = \int_{B_i^{adv}}^{B_i} \int_{\tau}^{B_i} p(x)dx\, f_c(\tau)d\tau + \int_{B_{i-1}}^{B_i^{adv}} \int_{B_{i-1}}^{\tau} p(x)dx\, f_c(\tau)d\tau \tag{2.6}$$

Since there are *k* classes, the overall probability of false routing is the sum of the probability of false routing within each class multiplied by the probability of the occurrence of that class:

$$\sum_{i=1}^{k} Pr\{false\ routing | B_{adv}^{n} = B_i^{adv}\} \cdot Pr\{B_{adv}^{n} = B_i^{adv}\} \tag{2.7}$$

The distribution of *p(x)*, which may be estimated, is assumed to be uniformly distributed in (B_{i-1}, B_i). The estimation of *p(x)* is beyond the scope of this chapter and will be addressed in the future work. Instead, the worst-case scenario is presented because a uniformly distributed random variable yields the highest entropy, i.e., the most uncertainty, in any finite range. With this assumption, equation (2.6) can be rewritten as:

$$Pr\{false\ routing | B_{adv}^{n} = B_i^{adv}\}$$

$$= \int_{B_i^{adv}}^{B_i} \frac{B_i - \tau}{B_i - B_{i-1}} f_c(\tau) d\tau + \int_{B_{i-1}}^{B_i^{adv}} \frac{\tau - B_{i-1}}{B_i - B_{i-1}} f_c(\tau) d\tau \tag{2.8}$$

Equation (2.7) involves the probability distribution of the advertised parameter B_i^{adv}. Assuming that A_n is uniformly distributed between 0 and A^{MAX}, then $Pr\{B_{adv}^{n} = B_i^{adv}\} = \frac{B_i - B_{i-1}}{A^{MAX}}$ for all *i*=1,2,…, *k*.

So the objective becomes:

$\min \sum_{i=1}^{k} Pr\{false\ routing | B_{adv}^{n} = B_i^{adv}\} \cdot \frac{B_i - B_{i-1}}{A^{MAX}}$.

Substituting equation (2.8) into the above equation:

$$\min_{B_j} \sum_{i=1}^{k} \left(\int_{B_i^{adv}}^{B_i} \frac{B_i - \tau}{B_i - B_{i-1}} f_c(\tau) d\tau + \int_{B_{i-1}}^{B_i^{adv}} \frac{\tau - B_{i-1}}{B_i - B_{i-1}} f_c(\tau) d\tau \right) \cdot \frac{B_i - B_{i-1}}{A^{MAX}} \tag{2.9}$$

Let $f\left(B_1, \ldots, B_k, B_1^{adv}, \ldots, B_k^{adv}\right)$

$= \sum_{i=1}^{k} \left(\int_{B_i^{adv}}^{B_i} \frac{B_i - \tau}{B_i - B_{i-1}} f_c(\tau) d\tau + \int_{B_{i-1}}^{B_i^{adv}} \frac{\tau - B_{i-1}}{B_i - B_{i-1}} f_c(\tau) d\tau \right) \cdot \frac{B_i - B_{i-1}}{A^{MAX}}$

$= \frac{1}{A^{MAX}} \cdot \sum_{i=1}^{k} \left(\int_{B_i^{adv}}^{B_i} (B_i - \tau) f_c(\tau) d\tau + \int_{B_{i-1}}^{B_i^{adv}} (\tau - B_{i-1}) f_c(\tau) d\tau \right)$

The objective is to find B_i and B_i^{adv} that minimize *f*. Therefore,

$$\frac{\partial f}{\partial B_i} = 0 \Rightarrow \int_{B_i^{adv}}^{B_i} f_C(\tau) - \int_{B_i}^{B_{i+1}^{adv}} f_C(\tau) = 0 \tag{2.10}$$

and

$$\frac{\partial f}{\partial B_i^{adv}} = 0 e^{i\theta} \Rightarrow \frac{\partial}{\partial B_i^{adv}} \left(\int_{B_i^{adv}}^{B_i} (B_i - \tau) f_C(\tau) d\tau + \int_{B_{i-1}}^{B_i^{adv}} (\tau - B_{i-1}) f_C(\tau) d\tau \right) = 0$$

$$\Rightarrow B_i^{adv} = \frac{B_i - B_{i-1}}{2} \quad \text{for} \quad i=1,2,\ldots, k, \text{ and } B_0 = 0 \tag{2.11}$$

From equations (2.10) and (2.11), in a *k*-class case, B_i can be derived from the following equation:

$$\int_{B_{i-1}}^{B_i} f_c(\tau) d\tau = \int_{B_i}^{B_{i+1}} f_c(\tau) d\tau = \frac{1}{k} \tag{2.12}$$

Note that if $f_c(\tau)$ is uniformly distributed, the result is identical to the equal class update policy. It can be interpreted as that equal class update policy is a special case of ROSE II.

2.4 Simulations

The simulation presented in this section uses delay as the additive QoS constraint. Denote D_{MAX} as the maximum amount of delay a link can experience in the network (e.g., queue full), i.e., $A^{MAX} = D_{MAX}$. The accept/reject criterion C is simulated as normally distributed with mean $= 0.3 \cdot D_{MAX}$, and variance $= 3 \cdot D_{MAX}$. Since the actual delay distribution of each link in the network is not known, in the simulations, the actual delay of the link, D_{actual} , is modeled as uniformly distributed in $(0, D_{MAX})$ – which represents the highest uncertainty (entropy). One hundred thousand connection setup attempts were made, each time with a different value of accept/reject criterion C and a different D_{actual} . The class boundaries and their corresponding advertised delay D_i^{adv} (which is equivalent to B_i^{adv} in equation (2.11)) were calculated according to equations (2.11) and (2.12). D_{MAX} is fixed at 1000 units.

Figure 2.2 shows the results of the simulation when the number of classes varies from 3 to 12. As one can see, when the number of classes increases, the probability of false routing from either ROSE II or equal-class updates decreases. This is due to the fact that the more classes, the more accurate link state information the network can obtain. However, regardless the number of classes, ROSE II always performs better than equal-class update, especially when the number of classes is small because equal-class update does not take the accept/reject criterion C into consideration.

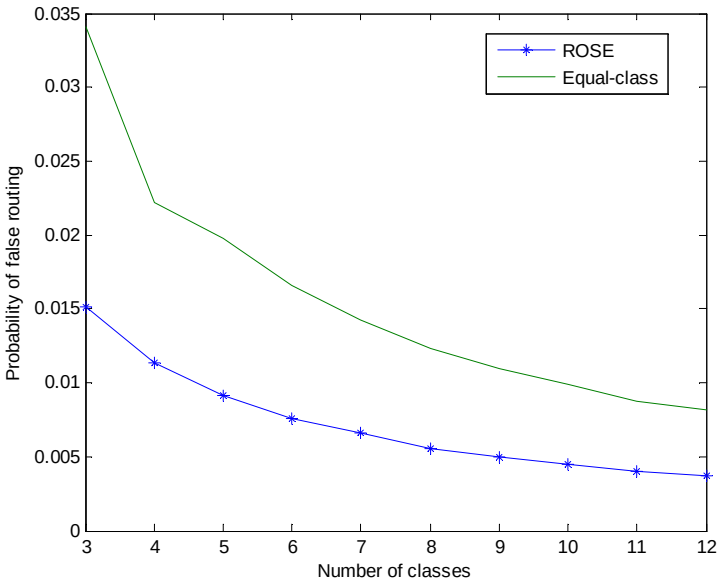


Figure 2.2 Probability of false routing with varying number of classes. (ROSE II vs. equal class.)

Figure 2.3 shows the results where the number of classes is fixed to 5 but the variance of the accept/reject criterion is varying from 1000 to 10000. From this figure, one can see that when the variance of accept/reject criterion *C* increases, the probability of false routing increases. However, ROSE II still performs better than equal-class, especially when the variance is low. Since ROSE II takes the probability distribution of the accept/reject criterion into consideration, the lower variance means the accept/reject criterion is more predictable, hence yielding better performance for ROSE II.

Figure 2.4 compares the performance between ROSE II and exponential-class update. Here, the number of classes is 5 and the variance of the accept/reject criterion is $3 \cdot D_{max}$. The factor f in exponential update varies from 1.1 to 2.0. One can see that the probability of false routing with ROSE II remains almost constant because the change of f does not affect ROSE II. On the other hand, the performance of exponential-class update

decays slightly as the value of *f* increases. Exponential-class update performs well when the QoS parameter is exponentially distributed or approximately so, because it assigns finer class ranges at lower value. Since this simulation is based on uniformly distributed QoS parameter and the accept/reject criterion is normal-distributed, which is a likely condition that additive QoS constraints would encounter, ROSE II is the better solution.

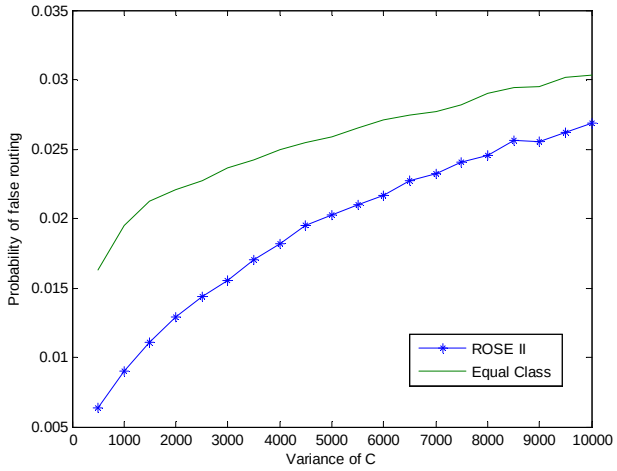


Figure 2.3 Probability of false routing with varying variance of *C*. (ROSE II vs. equal class.)

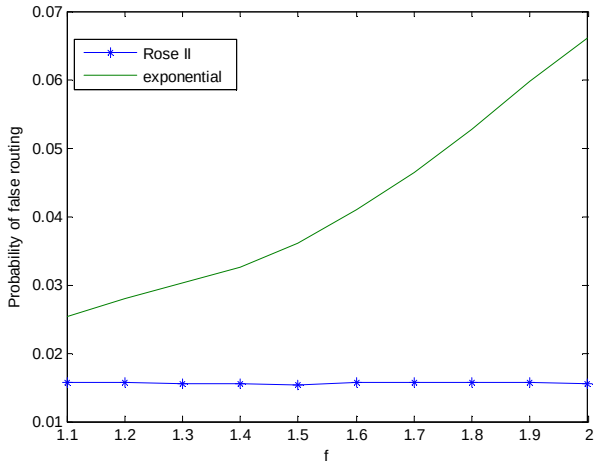


Figure 2.4 Probability of false routing with varying *f*. (ROSE II vs. exponential class.)

2.5 Summary

In this chapter, the behavior of additive QoS parameters has been investigated, such as delay. Owing to the unique property of additive QoS metrics, each link cannot make the decision of whether to accept or reject a connection request based on its own QoS parameter individually. Instead, the decision depends on the QoS parameters of all the links along the potential path. By the central limit theorem, on a per-link basis, the QoS criterion of accepting or rejecting a connection can be approximated as a normal-distributed random variable when the QoS constraint is additive. In this case, one can design the class ranges accordingly, and apply the ROSE II algorithm based on equations (2.11) and (2.12) to minimize the probability of false routing. Through extensive simulations, the results demonstrate that ROSE II performs better than equal-class update and exponential-class update with additive constraints.[1]

[1] ROSE has been demonstrated to outperform other update schemes for the concave constraints [7].

CHAPTER 3

SMART PACKET MARKING

TO IMPROVE THE ACCURACY OF EEAC_SV

3.1 Introduction

In order to provide Quality of Service (QoS) for the internet traffic, two fundamental frameworks have been proposed: Integrated Service (Intserv) [8] and Differentiated Service (Diffserv) [9]. Intserv along with RSVP [10] aims to offer flexible QoS guarantees to each individual flow, but suffers from the scalability limitation. Diffserv tries to overcome the scalability problem by coarsely aggregating traffic flows with various QoS requirements into a finite number of service classes, such as Expedited Forwarding (EF), Assured Forwarding (AF), and Best Effort (BE). All traffic flows in the same class experience the same QoS. The inability of Diffserv to provision QoS in fine granularity is its major drawback, thus potentially resulting in the wastage of network resources. Recently, an Explicit Endpoint Admission Control with Service Vector (EEAC-SV) has been proposed in [11] to provide finer granularity in QoS provisioning while maintaining the scalability of Diffserv. In EEAC-SV, each node maintains a finite number of service classes, and allows an end-to-end connection to utilize different service classes in different routers along the path. Finer QoS granularity results in better network utilization, and hence reduced costs [11].

Two phases are conducted during the life span of an end-to-end connection in EEAC-SV: probing phase and data transmission phase. In the probing phase, a probing packet is sent from the host that initiates the connection along the pre-selected route

Each router in the path marks the probing packet with its explicit QoS performance measurements (e.g., bandwidth, delay, etc.) of each service class. When the probing packet is sent back to this end host, based on the marked QoS information in the probing packet, it calculates the best strategy to utilize the network resource: find the least cost service vector that satisfies the QoS requirements. A service vector is defined as a selection of service classes in all the routers along the path. Once a service vector is selected, the data transmission phase begins.

The calculation of the service vector is a key step in EEAC-SV, and its accuracy highly depends on the packet marking scheme. However, there have not been many studies related to the efficiency of packet marking. In the framework proposed in [11], three congestion levels corresponding to the random early detection (RED) algorithm [12] are mapped into the delay measurements which are marked into the probing packets. Further studies on the probing packet marking mechanism are needed. First, the number of congestion levels should be in the form of 2^n in order to fully utilize the marking field in the probing packet, where *n* is a positive integer. For example, if the total number of congestion levels is five, three bits in the marking field are needed to represent the congestion levels from 1 to 5 even though the three bits are able to represent eight levels. To fully utilize the three bits, eight congestion levels should be adopted that will provide finer resolution of the congestion levels. Second, the choice of boundaries of each level can also affect the efficiency of the service vector calculation; and third, there are various types of QoS constraints (i.e., concave and additive) which have different properties and need to be considered separately. In this chapter, the packet marking mechanism is further investigated and an efficient packet marking solution that is suitable for various

types of QoS constraints is developed. The work presented in this chapter assumes that for any given QoS constraint, the probability distribution of users' requirement is known *a priori*. That is, taking the delay constraint as an example, the percentage of users that request network connections with end-to-end delay smaller than x is assumed to be known, where x is in the unit of time. This assumption can be justified by the fact that the probability distributions of user requirements can be obtained from user profiles and from the past experience of network operations. Furthermore, it is assumed that the probability distributions of all QoS performance measurements of each service class in each node are known. This probability distribution can be derived from the operation history of each node. With the above assumptions, the Smart Packet Marking mechanism is proposed. This mechanism takes advantage of the knowledge of the probability distributions of users' QoS requests and the network's QoS performance measurements to improve the efficiency of service vector calculation.

The rest of this chapter is organized as follows: Section 3.2 describes the general model of probing packet marking in EEAC-SV and points out its potential challenges. In Section 3.3 introduces the proposed Smart Packet Marking scheme. Section 3.4 presents the simulation results, and Section 3.5 the conclusion.

3.2 Modeling the Probing Packet Marking Mechanism

In EEAC-SV, a probing packet is sent along the path before the end-to-end connection is established. Each node along the path marks the probing packet with its own QoS measurements of each service class. Marking the packet with the precise measurements of all QoS parameters can result in tremendous amount of network overhead, i.e., the

probing packet has to be gigantic in order to carry detailed information. Therefore, in reality, the actual QoS measurement is coarsely quantized into a few levels. A network node marks a probing packet with the quantized level of a certain QoS parameter that this node can provision. When the probing packet is sent back to the sender, according to the markings in the probing packet, an end-to-end path that satisfies the user's QoS requirements is set up if such path exists, or the connection request is rejected if the network is deemed unable to satisfy the end user's QoS request. If the connection is rejected, the user can drop the request, make another request later, or try a different path. The user's reaction to the connection rejection is beyond the scope of this dissertation. In the case that the network is able to support the requested connection, a service vector is chosen based on the marking in the probing packet. The calculation of the service vector can be performed by edge routers or end hosts. For simplicity, in this chapter, the end hosts are referred to as the entities that perform the service vector calculation. The service vector defines which service class the connection uses at each node. This procedure allows the end-to-end connection to select different service classes at different nodes to satisfy the end-to-end QoS requirements while minimizing the cost.

Since the calculation of the service vector is based on the information in the probing packets, and the markings in probing packets only carry coarsely quantized values of each node's QoS measurements, it is possible that the chosen service vector is not optimized. When the service vector is not optimized, two situations can occur: (1) the chosen service vector cannot actually satisfy the user's QoS requirement, or (2) the chosen service vector can satisfy the user's QoS requirement at an unnecessarily higher cost. Note that the first situation also includes the condition where the connection is

falsely accepted by the network while the network is unable to support its QoS at all (i.e., even the highest-cost path is unable to satisfy the QoS requirement). The second situation includes the condition where the connection is falsely rejected by the network while there should be a service vector that can support the connection. In this chapter, situation (1) is referred to as a false positive since a service vector is falsely accepted, and situation (2) is referred to as a false negative since a service vector is falsely rejected, or when the user is forced to choose a service vector with a higher-than-necessary cost. The term "false routing" [7] [2] will be used in this chapter to refer to both false positive and false negative. The probability of false routing will be used as the measure of the efficiency of packet marking and service vector calculation.

3.3 Smart Packet Marking

The Smart Packet Marking mechanism is proposed in this section, which can reduce the probability of false routing. This mechanism takes the assumption that, for a given QoS parameter, the probability distribution of users' QoS requirement is known, denoted by pdf *p(x)*, and the probability distributions of the QoS measurements of each service class in each node are also known, denoted by pdf $q_c^l(y)$,where *l* is the node number and *c* denotes the service class (e.g., BE, AF, and EF). For simplicity, further assume that $q_c^1(y) = q_c^2(y) = \ldots = q_c(y)$ for all nodes. In packet marking, let *k* be the total number of

[2] "False routing" refers to the situation that the network makes an incorrect decision of accepting or rejecting a connection with respect to the actually available resources. In this paper, it is assumed that the only cause of false routing is the inaccurate QoS information reported by each node.

levels of quantization in a service class *c*, and B_i^c be the boundary of each level *(i=0,…, k)*, and $B_0^c = 0$. Then, for each QoS parameter in each class, if its actual measurement is *y*, the node will mark the probing packet as level *m* if $B_{m-1}^c \le y \le B_m^c$ $(1 \le m \le k)$. When the marked probing packet is sent back to the end user, it carries the information of the QoS performance levels of all service classes in all nodes. For example, a probing packet that has been marked by three nodes would have the following matrix:

$$\text{Marked information} = \begin{bmatrix} \left[Q_{BE}^1, Q_{AF}^1, Q_{EF}^1 \right] \\ \left[Q_{BE}^2, Q_{AF}^2, Q_{EF}^2 \right] \\ \left[Q_{BE}^3, Q_{AF}^3, Q_{EF}^3 \right] \end{bmatrix},$$

where $Q_{BE}^1 = 0,1,\ldots,k$ is the delay level of the BE class currently measured at node 1, Q_{EF}^2 is the delay level of the EF class measured at node 2, etc.. Note that the above matrix only carries the information of one type of QoS performance measurement (e.g., delay), each different type of QoS is denoted by a separate matrix.

For each level, an "advertised value" $\overline{B_m^c}$ $(B_{m-1}^c \le \overline{B_m^c} \le B_m^c)$, known to the end user *a priori*, is used in the calculation of the service vector. The Smart Packet Marking is to design the boundaries B_m^c and advertised value $\overline{B_m^c}$ based on *p(x)* and $q_c(y)$ to minimize the probability of false routing.

To analyze the probability of false routing, it is necessary to scrutinize two major types of QoS constraints: concave and additive constraints (There is another type of QoS constraints: multiplicative, which can be converted into additive constraints by using the logarithm operation. Therefore, only concave and additive constraints need to be addressed). Consider the case where an end-to-end path traverses through *m* links, and

each link has a certain QoS parameter measurement (e.g., delay or bandwidth) Q_i *(i=1,..,m),* then this end-to-end path satisfies the user's QoS requirement x if $min\{Q_1, Q_2, \ldots, Q_m\} \geq x$ when the QoS constraint is concave, or if $\sum_{i=1}^{m} Q_i \leq x$ when the QoS constraint is additive. Concave and additive constraints exhibit different properties in the analysis of the probability of false routing, and therefore they have to be considered separately.

Concave constraints:

On a per-link basis, let y be the actual QoS measurement of class c in this link. Then, when y falls between the interval $[B_{m-1}^c, B_m^c]$ (i.e., level m), the probability of false positive for concave constraints can be expressed by the following:

$$Pr\{False\ Positive\} = \int_{B_{m-1}^c}^{\overline{B_m^c}} \int_{B_{m-1}^c}^{x} q_c(y)dy \cdot p(x)dx . \tag{3.1}$$

This is the probability that $\overline{B_m^c}$ is greater than the user's request x, but the actual QoS measurement y is smaller than x. The connection will be accepted into class c of this link, but this class is actually unable to satisfy the request.

Similarly, the probability of false negative is:

$$Pr\{False\ Negative\} = \int_{\overline{B_m^c}}^{B_m^c} \int_{x}^{B_m^c} q_c(y)dy \cdot p(x)dx \tag{3.2}$$

In this case, $\overline{B_m^c}$ is smaller than the user's request *x*, but the actual QoS measurement *y* is greater than *x*. The connection will be denied from using class *c* of this link while this class is actually able to satisfy the requirement.

Then, the probability of false routing at level *m* is

$$\int_{B_{m-1}^c}^{\overline{B_m^c}} \int_{B_{m-1}^c}^{x} q_c(y)dy \cdot p(x)dx + \int_{\overline{B_m^c}}^{B_m^c} \int_{x}^{B_m^c} q_c(y)dy \cdot p(x)dx \tag{3.3}$$

Additive constraints:

A unique property of additive constraints is that the decision of whether a certain class of a link can support the QoS requirement cannot be made based solely on this class's QoS measurement; it involves the QoS measurements of all other links along the path. Therefore, from a single link's point of view, whether class *c* can be selected to support the connection relies on whether the advertised QoS performance $\overline{B_m^c}$ is greater or smaller than $x - \sum\{\text{advertised values of all other nodes}\}$, where *x* is the user's requirement. Since it is assumed that the probability distributions of user's request *x* and the QoS measurement *y* are known, let $C = x - \sum\{\text{advertised values of all other nodes}\}$, then *C* can be viewed as the sum of multiple random variables. By applying the Central Limit Theorem, the probability distribution of *C* can be approximated by a Gaussian distribution whose mean and variance can be derived from *p(x)* and $q_c(y)$.

Let $f_C(x)$ be the pdf of *C*, on a per-link basis, the probability of false positive for additive constraints can be expressed by the following:

$$Pr\{False\ Positive\} = \int_{B_{m-1}^c}^{\overline{B_m^c}} \int_{B_{m-1}^c}^{x} q_c(y)dy \cdot f_C(x)dx \tag{3.4}$$

Similarly, the probability of false negative is:

$$Pr\{False\ Negative\} = \int_{\overline{B_m^c}}^{B_m^c} \int_{x}^{B_m^c} q_c(y)dy \cdot f_C(x)dx \tag{3.5}$$

The probability of false routing at level *m* is then

$$\int_{B_{m-1}^c}^{\overline{B_m^c}} \int_{B_{m-1}^c}^{x} q_c(y)dy \cdot f_C(x)dx + \int_{\overline{B_m^c}}^{B_m^c} \int_{x}^{B_m^c} q_c(y)dy \cdot f_C(x)dx \tag{3.6}$$

Thus far, the probabilities of false routing have been derived for each congestion level for both concave and additive constraints. Recall that *k* is the total number of levels, and so the overall probability of false routing in each class is

$$Pr\{False\ Routing\ \} = \sum_{m=1}^{k} Pr\{False\ Routing\ ,\ level\ =\ m\} \tag{3.7}$$

The objective is to design B_m^c and $\overline{B_m^c}$ such that the probability of false routing is minimized. Let *g=Pr{False Routing}*. In order to find the minimum probability of false routing, one needs to find B_m^c and $\overline{B_m^c}$ *(m=1,2,…,k)* such that $\frac{\partial g}{\partial B_m^c} = 0$ and $\frac{\partial g}{\partial \overline{B_m^c}} = 0$. Therefore, for concave constraints:

$$\frac{\partial g}{\partial B_m^c} = 0 \Rightarrow \int_{\overline{B_m^c}}^{B_m^c} p(x)dx - \int_{B_m^c}^{\overline{B_{m+1}^c}} p(x)dx = 0 \tag{3.8}$$

$$\frac{\partial g}{\partial \overline{B_m^c}} = 0 \Rightarrow \int_{B_{m-1}^c}^{\overline{B_m^c}} q_c(y)dy - \int_{\overline{B_m^c}}^{B_m^c} q_c(y)dy = 0 \tag{3.9}$$

For additive constraints:

$$\frac{\partial g}{\partial B_m^c} = 0 \Rightarrow \int_{\overline{B_m^c}}^{B_m^c} f_C(x)dx - \int_{B_m^c}^{\overline{B_{m+1}^c}} f_C(x)dx = 0 \tag{3.10}$$

$$\frac{\partial g}{\partial \overline{B_m^c}} = 0 \Rightarrow \int_{B_{m-1}^c}^{\overline{B_m^c}} q_c(y)dy - \int_{\overline{B_m^c}}^{B_m^c} q_c(y)dy = 0 \tag{3.11}$$

From equations (3.8) to (3.11), B_m^c and $\overline{B_m^c}$ can be solved for. Although solving these equations may require a considerable amount of computation, this computation only needs to be executed once before the implementation of the Smart Packet Marking scheme, since $q_c(y)$, $f_C(x)$, and $p(x)$ do not vary dramatically with time. Once the solutions for B_m^c and $\overline{B_m^c}$ are found, they can be plugged into the routers. Then, the routers will simply mark the probing packets according to these boundary values and the end hosts will calculate the service vectors according to $\overline{B_m^c}$'s.

3.4 Simulations

To demonstrate the efficiency of the Smart Packet Marking scheme, this section compares its performance with other packet marking schemes that do not take stochastic information of user's requests and network QoS measurements into consideration. In the following simulations, an Equal-boundary Packet Marking scheme is being compared with the Smart Packet Marking scheme. Unlike Smart Packet Marking, the Equal-boundary Packet Marking (referred to as Equal Packet Marking hereafter) scheme designs its boundaries in a way such that the differences between each adjacent boundary pairs are equal, that is, $B_m^c - B_{m-1}^c = B_{m+1}^c - B_m^c$ for all m. In other words, the quantization is uniform. This packet marking scheme corresponds to the "Equal Class Update" scheme in the study of link state update [7] [3], and it is therefore chosen here for comparison purposes. It is assumed that the network offers three service classes: BE, AF, and EF, with the cost of each class equals to 1, 2, and 3, respectively. Each end to end connection traverses through 5 links (hop count = 5). Both concave and additive constraints are considered in the simulation: for concave constraints, bandwidth is chosen as the QoS parameter; for additive constraints, delay is used as the QoS parameter. In each simulation, 100,000 connection attempts are made, and the propagation time of probing packets are assumed to be negligible. The probability of false routing is calculated from the total number of false routing occurrences divided by the total number of connection attempts, while the occurrence of false routing is as defined in Section II.

A. Concave Constraint – Bandwidth:

The bandwidth capacity of each class is 100,000 units, and is equal in every link. The occupied bandwidth is exponentially distributed with means equal to 20,000 units in EF, 30,000 units in AF, and 40,000 units in BE. The user requested bandwidth is Gaussian distributed with the mean of 40,000 units and variance of 5×10^8. The total number of levels varies from 4 to 8, and is the same for each class and each link. Figure 3.1 shows the comparison of probability of false routing between Smart Packet Marking and Equal Packet Marking with various numbers of levels.

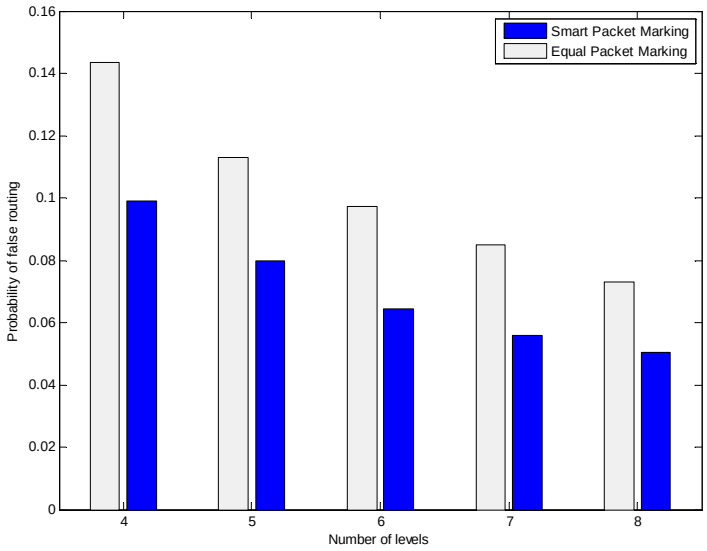


Figure 3.1 Probability of false routing for the concave (bandwidth) constraint. Equal Packet Marking vs. Smart Packet Marking. The number of quantization levels varies from 4 to 8.

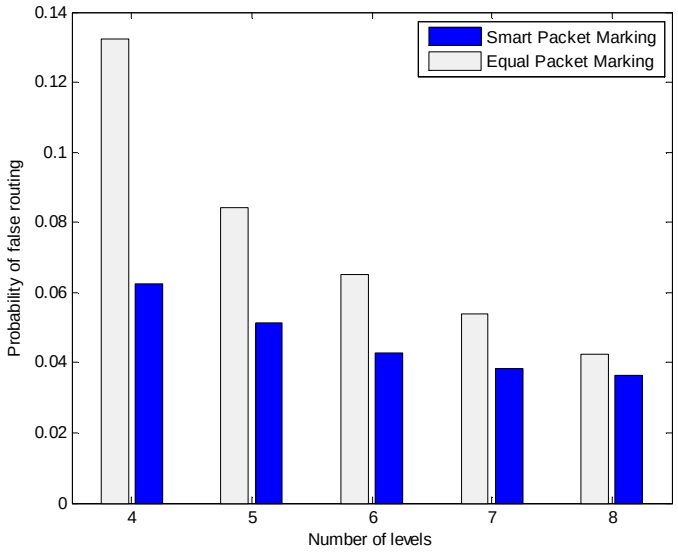


Figure 3.2 Probability of false routing for the additive (delay) constraint. Equal Packet Marking vs. Smart Packet Marking. The number of quantization levels varies from 4 to 8.

B. Additive Constraint – Delay:

The maximum delay of each class is 100,000 units and is equal in every link. The actual delay measurement is exponentially distributed with means equal to 8,000 units in EF, 10,000 units in AF, and 12,000 units in BE. The user requested delay is Gaussian distributed with the mean of 50,000 units and variance of 1×10^8. Figure 3.2 shows the comparison of probability of false routing between Smart Packet Marking and Equal Packet Marking with the delay constraint. Again, the total number of levels varies from 4 to 8, equally applied to all classes in all links.

As one can see, Smart Packet Marking yields much lower probability of false routing in most of the cases, especially when there are fewer levels. Recall that the total number of levels corresponds to the number of bits used in packet marking; fewer levels means smaller probing packets and less overhead. Therefore, the results indicate that

even with only two bits dedicated for packet marking for each class, which corresponds to four levels, Smart Packet Marking can effectively reduce the probability of false routing for both concave and additive cases.

C. Varying the Variance

The previous subsection has shown the effect of changing the number of levels on the performance of packet marking. This subsection further investigates the effect of changing the variance by running simulations with a fixed number of levels and various variances. In the case of bandwidth (concave) constraint, five scenarios are simulated. In the first scenario, the mean bandwidth occupancies of EF, AF, and BE are 5,000, 15,000, and 35,000, respectively. With each increment in the scenario index, the mean in each class increases by 5000 units. Note that, since the bandwidth occupancy is exponentially distributed, the variance of the available bandwidth is the square of the mean; increasing the mean will increase the variance automatically. The number of levels is fixed at 4, and the rest of the conditions remain the same as in the previous simulations. The results are presented in Figure 3.3.

In the case of the delay (additive) constraint, five scenarios are adopted as well. The mean of delay measurements increases by 2,000 units for each scenario: from 6,000 to 14,000 in EF, from 8,000 to 16,000 in AF, and from 10,000 to 18,000 in BE. Since all delay measurements are assumed to be exponentially distributed, the variance therefore increases accordingly. The number of levels is fixed at 4 as before. The results are presented in Figure 3.4.

From Figures 3.3 and 3.4, one can see that a higher variance of QoS measurements will cause a higher probability of false routing in both packet marking schemes for both concave and additive constraints. The increment is even more significant in the case of the delay constraint since the algorithm of choosing the service vector for an additive constraint involves the sum of the QoS behavior of all links. Since a small variance increment in each link will result in a big variance increment in the sum, the overall impact becomes more pronounced. One interesting note is that even though Equal Packet Marking does not take any stochastic information into consideration, its performance is still affected by the change in variance. However, regardless of the variance chosen in the above simulations, Smart Packet Marking always outperforms Equal Packet Marking.

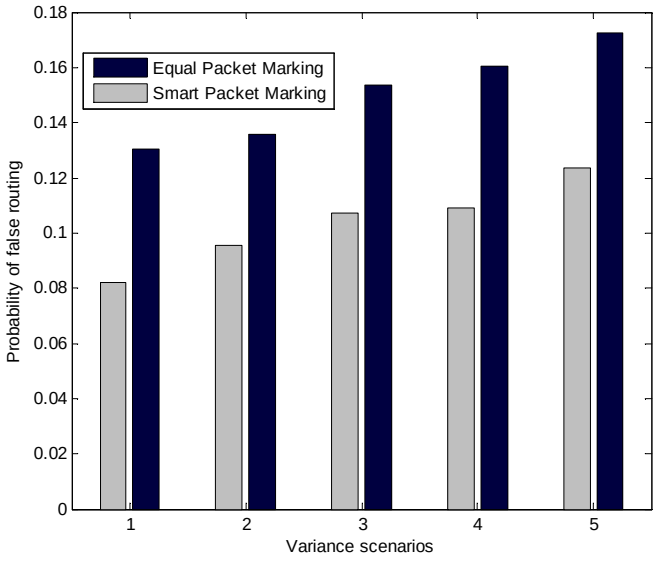


Figure 3.3 Probability of false routing for the concave (bandwidth) constraint. Equal Packet Marking vs. Smart Packet Marking. The variance of bandwidth measurements increases with each scenario.

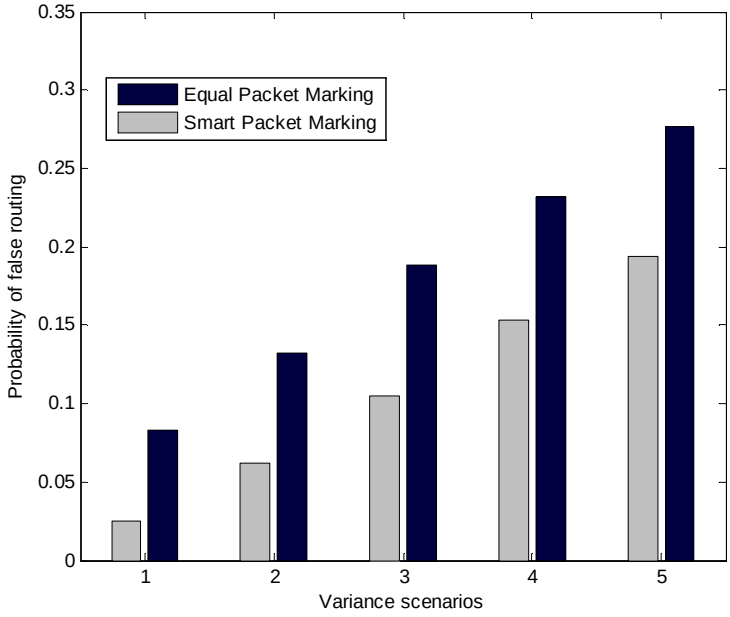


Figure 3.4 Probability of false routing for the additive (delay) constraint. Equal Packet Marking vs. Smart Packet Marking. The variance of delay measurements increases with each scenario.

3.5 Summary

EEAC-SV is a solution that allows the network to provision QoS in finer granularity, thus increasing network utilization and reducing cost. Marking the probing packet is an important element in EEAC-SV. The mathematical analysis indicates that the performance of EEAC-SV can be improved with a properly designed packet marking scheme. In this chapter, a probing packet marking mechanism, Smart Packet Marking, has been proposed based on a rigorous mathematical analysis. Extensive simulations demonstrated that, with the knowledge of the probability distribution functions of users' QoS requirements and network's QoS performance, one can design the quantization boundaries of probing packet marking that minimizes the probability of false routing by

finding the optimal service vector. The computation, though with some degree of complexity, only needs to be executed once before the implementation. Therefore, with the same amount of network overhead, Smart Packet Marking achieves maximum efficiency in the calculation of the service vector, hence improving the performance of EEAC-SV.

CHAPTER 4

ROUTING-ORIENTED UPDATE SCHEME (ROSE) FOR LINK STATE UPDATING

4.1 Introduction

The ability to provide Quality of Service (QoS) is a necessity for the next generation integrated networks. Today, QoS routing has become the fundamental focus of study. The goal of QoS routing is to find a path that satisfies multiple QoS constraints while maximizing the network utilization and minimizing users' costs. QoS routing in general consists of two critical issues: link state dissemination and route selection [1]. The link state dissemination addresses how the link state information is exchanged throughout the network while the route selection elaborates on how to find the optimal path given the available link state information. Many works have addressed the issue of route selection [13]-[18]. This chapter concentrates on the issue of link state dissemination.

The purpose of link state dissemination is to provide the knowledge of QoS status of all the links to the routing devices (e.g., routers) in a network. Based on this knowledge, the network can then determine the best route for any given end-to-end connection to meet its QoS requirements and utilize the overall network resource efficiently. In order to provide the knowledge of all the QoS parameters of each link, each link itself must employ some scheme to report its own QoS parameters, referred to as "link state update". Generally, it is impractical to assume that routing devices have accurate link state information of all links at all time, because this would require rapid link state updates from all links, hence, consuming a large amount of network resource.

Therefore, an effective link state update algorithm is necessary to provision QoS. Link state update determines the behavior of how each node updates its status to the entire network, including when to update and how to update.

A widely used link state update protocol, OSPF [2], which has also been adopted in many types of networks such as optical networks [19], recommends the link state to be updated once every 30 minutes. However, because of the highly dynamic nature of the traffic, updating in such a long time interval will result in stale/outdated link state parameters. This will compromise the efficiency of QoS routing. Several other link state update policies, such as threshold, equal class and exponential class based update policies [20], have been proposed. In the threshold policy, an update is triggered when the difference between the current value and the previously updated value of a certain parameter exceeds a threshold. That is, given a threshold value τ, an update occurs when $|b_c - b_0| > \tau$, where b_0 is the previously updated value and b_c is the current value of a QoS parameter. In the equal-class and the exponential-class based update policies, the values of QoS parameters are divided into classes. An update is triggered when the current value of a QoS parameter changes from one class to another. For example, in a two-class situation, if the range (interval) of the first class is $(0, b_1)$, and the range of the second class is (b_1, b_2), then an update will happen when b_c changes from $0 < b_c < b_1$ to $b_1 < b_c < b_2$, or vice versa.

What separates the equal class based link state update policy from the exponential class based link state policy is the choice of the boundaries, or in other words, the partitioning of each class. In the equal class based link state update, the class of a QoS parameter is partitioned into equal-sized intervals, for example, *(0,B), (B, 2B), (2B,*

3B),…, etc.. In the exponential class based update, the classes are partitioned into unequal-sized ranges, *(0,B), (B, (f+1)B), ((f+1)B,(f^2+f+1)B),*…, etc., whose sizes grow geometrically by a factor of *f*, where *B* is a predefined constant.

No matter which link state update policy a network adopts, it is unavoidable that the QoS parameters of each node known to the entire network might not be exactly accurate at any given time, due to the staleness and coarse classes. As a result, false routing is inevitable. Some works have been done in analyzing the effect of stale or inaccurate link state information, and attempting to reduce its impact. In [4], extensive simulations were made to uncover the effects of the stale link state information and random fluctuations in the traffic load on the routing and setup overheads. In [5]-[6], the effects of the stale link state information on QoS routing algorithms were demonstrated through simulations by varying the link state update interval. A combination of the periodic and triggered link state update is considered in [21]. Instead of using the link capacities or instantaneous available bandwidth values, Li *et al.* [22] used a stochastic metric, Available Bandwidth Index (ABI), and extended BGP to perform the bandwidth advertising.

In this chapter, for the purpose of saving network resources and reducing the staleness of link state information, a new link state information update scheme is introduced: Routing-Oriented update SchEme (ROSE)[3]. The uniqueness of ROSE is that it takes the QoS requirements of applications and the network QoS behavior into account. As reviewed above, most of the existing link state update schemes do not consider both

[3] Preliminary results of ROSE have been presented in [7] and [23].

the statistical distributions of the actual user's QoS requirements and the network's QoS behavior. In fact, this research has discovered that the knowledge of the distribution of the user's QoS requirements and the history of network's QoS behavior can greatly improve the efficiency of link state update and the accuracy of QoS routing. The statistical distribution of the user's QoS requirement and the network's QoS behavior can be obtained from the network operation history. The key concept of ROSE is to utilize these statistical distributions and design a class-based link state update scheme that is able to provide the most helpful link state information for the connection setup processes, hence yielding better performance than other existing link state update schemes. Via theoretical analysis and simulations, ROSE is shown to greatly outperform the state of the art.

The rest of this chapter is organized as follows. Section 4.2 describes the properties of various types of QoS constraints. Section 4.3 defines the term "false routing" and the cost of false routing. Then, Section 4.4 proposes an efficient link state information update scheme, ROSE. The simulation results are presented in Section 4.5. Finally, concluding remarks are given in Section 4.6.

4.2 Properties of QoS Constraints

Most of the QoS constraints (e.g., bandwidth, delay) can be categorized into the following three types: concave, additive, and multiplicative. Multiplicative constraints can be converted into additive constraints by using the logarithm operator. Therefore, only concave and additive constraints are considered in the study of QoS routing. A

concave constraint works as follows: in the case of a multi-link end-to-end path, as long as the smallest (or largest) QoS parameter among all the links is larger (or smaller) than the corresponding QoS requirement, then this path is considered acceptable. Bandwidth is a typical example of the concave constraint. An additive constraint works as follows: in the case of a multi-link end-to-end path, the *sum* of all the QoS parameters along the path has to be less than the corresponding QoS requirement in order for this path to be acceptable. Delay is a typical example of the additive constraint. In Figure 4.1, the path consists of 3 links: link 1, 2, and 3. Each of these links has a concave QoS parameter C_1, C_2, and C_3, respectively, and an additive QoS parameter A_1, A_2, and A_3, respectively. If a connection imposes QoS constraints C_0 and A_0, then the path is deemed acceptable if $min\{C_1, C_2, C_3\} \geq C_0$, and $A_1 + A_2 + A_3 \leq A_0$

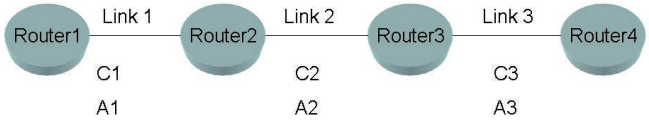


Figure 4.1 Illustration of concave and additive constraints: Concave QoS parameters of link 1, 2, and 3 = *C1*, *C2*, and *C3*. Additive QoS parameters of link 1, 2, and 3 = *A1*, *A2*, and *A3*. The path is acceptable if $\min\{C1, C2, C3\} \geq C0$ and $A1 + A2 + A3 \leq A0$, where C0 and A0 are required concave and additive constraints.

One of the special characteristics of an additive constraint is that, from a per-link point of view, the QoS requirement of each link is related to the QoS behavior of all the other links in the same path. Taking link 2 in Figure 4.1 as an example, link 2 will be accepted if $A_2 \leq A_0 - (A_1 + A_3)$. Similarly, in an *m*-link path, a link among these *m* links, $l_j \ (j \leq m)$, is acceptable if $A_j \leq A_0 - \sum_{i=1, i \neq j}^{m} A_i$.

Therefore, from the perspective of a single link, it cannot make the decision whether to accept or reject a connection purely based on its own additive link state metrics.

As one can see, the concave constraints have quite different properties from those of additive constraints; therefore, they have to be considered separately when designing a link state update scheme. Those aforementioned current link state update schemes (threshold, equal class, and exponential class updates) do not take the difference of these properties into consideration. The contents in the next few sections will show how ROSE can cope with both concave and additive constraints better than the current link state update schemes.

4.3 False Routing

Ideally, when a connection request with certain QoS requirements is made to the network, the network's routing mechanism will accept this request and set up the connection if there are enough resources in the network to support the required QoS, and reject the request otherwise. However, in the real situation, since the routing mechanism does not always have the accurate link state information, it is unavoidable that some connections will be falsely accepted when the network actually cannot meet the QoS requirements while some other connections will be falsely rejected when the network actually has enough resource to support the QoS requirements. In this chapter, an instance of the first situation – a connection is falsely accepted – is referred to as a "false positive", and an instance of the second situation – a connection is falsely rejected – is referred to as a "false negative". Both false positives and false negatives constitute the definition of

"false routing". In other words, "false routing" has occurred as long as either a false positive or a false negative occurs.

False positives can jeopardize user's satisfaction since users are experiencing poor QoS in this situation. Meanwhile, false negatives can cause the under-utilization of network resources by rejecting the connections that should have been accepted. Therefore, both false positives and negatives are considered undesired situations. One can argue that one situation is more severe or, in other words, more costly, than the other. To reflect this concern, instead of simply gauge the performance of QoS routing by the probability of false routing, one should compare the "cost of false routing" for more realistic evaluation. A cost factor is used in ROSE, and therefore ROSE is not only capable of minimizing the occurrence of overall false routing, but also minimizing the overall cost of false routing. Throughout the rest of this chapter, the cost of false routing will be used as the measure of the efficiency of various link state update schemes.

4.4 Route-Oriented Update SchEme (ROSE)

This section describes the new class-based link state update scheme, ROSE. The fundamental concept of ROSE is to utilize the statistical distribution of the user's QoS request and the network's QoS behavior in order to design an efficient class-based link state update scheme. The distribution of the user's QoS request can be obtained from the user profile (for example, *x*% of the connections requires *y* bps of bandwidth). The distribution of the network's QoS behavior can be derived from observing the operation history. Taking delay as an example, many reports have studied the delay measurements of various traffic types [24][25]. Reference [25] has proposed a method to measure the

single-hop delay, represented as the frequency histogram of delay. Reference [26] directly indicates that the queue length of a bottlenecked link is likely to be Gaussian distributed as long as there is a large number of TCP sessions on this link at any given time. Since queuing delay is a major contributor of the end to end delay and possesses the most dynamic nature, the distribution of queue length can also be used to derive the pdf of single-hop delay. The subject of Internet measurements, which is a readily pursued research, is beyond the scope of this dissertation. In this chapter, the pdf's of the user's request and network's QoS behavior are assumed to be known for the purpose of illustrating the ROSE algorithm.

Consider a network composed of m links, denoted by the graph $G(V, E)$, and assume there is a routing device (either distributed or centralized) that makes the decision of whether to accept a connection request and finds the end-to-end paths that can provide the appropriate QoS to all accepted connections. The routing device makes the decision based on the link state information acquired via link state update. Generally, in a class-based link state update scheme, each link updates its QoS parameter by using a finite number of classes; here, let k be the number of classes. For a given QoS parameter, its value is assumed within a finite range (for example, the available bandwidth of a link can only be ranged from 0 up to the full link capacity). Under these assumptions, with k classes, the ranges of the respective classes can be expressed as: $[B_{min}, B_1]$, $[B_1, B_2]$, $[B_2, B_3]$,…, $[B_{k-1}, B_{MAX}]$, where B_{min} and B_{MAX} are the minimum and maximum of the QoS parameter, and B_1, B_2 ,…, B_{k-1} are the boundaries of classes. To simplify the notations, let $B_0 = B_{min}$ and $B_k = B_{MAX}$ throughout the rest of the chapter. For each class, there is also a representative value which is "advertised" by the link to the routing device as if it is the

exact value, denoted by $B_1^{adv}, B_2^{adv}, \ldots, B_k^{adv}$. For instance, if the available bandwidth of a link $l_j \in E$ falls in the range of *[B_1, B_2]* (class 2), then the link state update message will "advertise" that the available bandwidth of l_j is B_2^{adv}. In short, the links update their QoS status in a quantized manner. Figure 4.2 illustrates the concept of class-based link state update. The routing device then makes the routing decision based on the advertised values from all the links. However, owing to quantization, false routing is inevitable. This can be illustrated by continuing the above example: When link l_j reports its available bandwidth as B_2^{adv}, the true value can be anywhere from B_1 to B_2. Therefore, if a connection attempting to utilize link *j* requests *x* amount of bandwidth and $B_1 < x < B_2^{adv}$, the routing device will accept this request. However, it is possible that the actual available bandwidth of link l_j is less than *x* but greater than B_1, and therefore incurring a false positive. On the other hand, if $B_2^{adv} < x < B_2$ and the actual available bandwidth of link l_j is greater than *x* but less than B_2, a false negative will occur (refer to Figure 4.2). The goal of ROSE is to design the class boundaries and the advertised values intelligently to minimize the cost of false routing. Owing to the different properties of concave constraints and additive constraints as described in Section 2, they have to be considered separately in the design of ROSE.

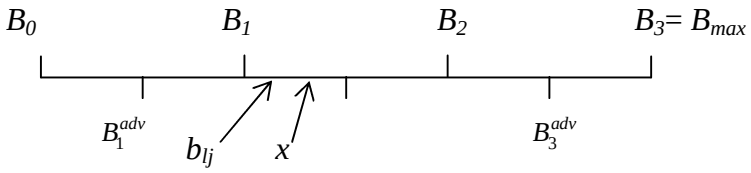


Figure 4.2 Illustration of class boundaries and advertised values. In the concave case, a false positive occurs when $B_1 < b_{l_j} < x < B_2^{adv}$.

A. Concave QoS Constraints:

The analysis in this section starts with a single concave QoS metric – bandwidth. When a connection requests *x* amount of bandwidth from link l_j, the connection will be accepted if $x < B^{adv}(l_j)$, where $B^{adv}(l_j)$ denotes the advertised available bandwidth of link l_j, and will be rejected otherwise. Assume that the actual available bandwidth of l_j, b_{l_j}, is within the range of class *n* ($1 \le n \le k$, *k* is the total number of classes), then $B^{adv}(l_j) = B_n^{adv}$. A false positive occurs when $x < B^{adv}(l_j)$ but $b_{l_j} < x$ (the actual bandwidth is less than the requested bandwidth). For this condition to hold, the following has to be true: $B_{n-1} < b_{l_j} < x < B^{adv}(l_j)$. Recall that it is assumed that the statistical information of the user's QoS requirements and the network's QoS behavior are known, from which one can derive their corresponding probability density functions (pdf). Therefore, *x* and b_{l_j} can be simply treated as random variables. Let $q(x)$ be the pdf of the user's request *x* and $p(b)$ be the pdf of the actual available bandwidth b_{l_j}, then the probability of false positive can be written as:

$$Pr\{False\ Positive,\ class=n\} = \int_{B_{n-1}}^{B_n^{adv}} \int_{b}^{B_n^{adv}} q(\tau)d\tau \cdot p(b)\,db \qquad (4.1)$$

Similarly, the probability of a false negative is:

$$Pr\{False\ Negative,\ class=n\} = \int_{B_n^{adv}}^{B_n} \int_{B_n^{adv}}^{b} q(\tau) d\tau \cdot p(b)\, db \tag{4.2}$$

Equation (4.1) represents the situation of $B_{n-1} < b_{l_j} < x < B^{adv}(l_j)$, and equation (4.2) the situation of $B^{adv}(l_j) < x < b_{l_j} < B_n$. Note that equations (4.1) and (4.2) are *not* conditional probabilities; they describe the probability of false positive/negative AND the current class is *n*. Therefore, the overall probability of a false positive is

$$Pr\{False\ Positive\} = \sum_{n=1}^{k} Pr\{False\ Positive,\ class=n\} \tag{4.3}$$

and the probability of a false negative is

$$Pr\{False\ Negative\} = \sum_{n=1}^{k} Pr\{False\ Negative,\ class=n\} \tag{4.4}$$

Since the severity of a false positive and a false negative might not be equal, let c_p be the cost of a false positive and c_n be that of a false negative, then the total cost of false routing *C* can be written as:

$$C = c_p \cdot Pr\{False\ Positive\} + c_n \cdot Pr\{False\ Negative\}$$

$$= c_p \sum_{n=1}^{k} \int_{B_{n-1}}^{B_n^{adv}} \int_{b}^{B_n^{adv}} q(\tau) d\tau \cdot p(b)\, db + c_n \sum_{n=1}^{k} \int_{B_n^{adv}}^{B_n} \int_{B_n^{adv}}^{b} q(\tau) d\tau \cdot p(b)\, db \tag{4.5}$$

In order to minimize *C* with respect to B_n and B_n^{adv}, one needs to find the solutions to the following equations:

$$\frac{\partial C}{\partial B_n} = 0 \Rightarrow c_n \int_{B_n^{adv}}^{B_n} q(\tau) d\tau - c_p \int_{B_n}^{B_{n+1}^{adv}} q(\tau) d\tau = 0 \tag{4.6}$$

$$\frac{\partial C}{\partial B_n^{adv}} = 0 \Rightarrow c_p \int_{B_{n-1}}^{B_n^{adv}} p(\tau) d\tau - c_n \int_{B_n^{adv}}^{B_n} p(\tau) d\tau = 0 \tag{4.7}$$

B. Additive Constraints:

In the analysis of additive constraints, delay is chosen as an example for the rest of the chapter. A unique property of additive constraints is that the decision of whether a link l_j can support the QoS requirement cannot be made based solely on this link's QoS measurement; it involves the QoS measurements of all other links along the path. Therefore, from the routing device's point of view, the decision of whether to select link l_j depends on whether

$$B^{adv}\left(l_j\right) < x - \sum_{i \in path, i \neq j} B^{adv}\left(l_i\right) \tag{4.8}$$

In other words, the decision is made based on whether the advertised delay of l_j is less than the user's request *x* subtracted by the sum of the advertised delays of all other links in the potential path. Again, since it is assumed the statistical information of *x* (request) and b_{l_j} (actual delay in link l_j) is available, *x* and $B^{adv}\left(l_j\right)$ can be treated as

random variables, where the pdf of $B^{adv}(l_j)$ can be derived from the pdf of b_{l_j}. Then, the right half of equation (4.8) can be viewed as the sum of random variables. Let *S*= $x - \sum_{i \in path, i \neq j} B^{adv}(l_i)$, and $f_S(s)$ be the pdf of *S*. Essentially, *S* is the criterion of whether the connection will be accepted to utilize link l_j. Therefore, *S* will be referred to as the "accept/reject criterion" in this chapter. Applying the Central Limit Theorem, $f_S(s)$ can be approximated by Gaussian distribution whose mean and variance can be derived from the pdf of *x* and $B^{adv}(l_j)$. Note that the mean and variance are affected by the number of hops in a connection. To simplify this problem, ROSE adopts the average hop count in a network to estimate $f_S(s)$. As it will show in the simulations, this simplified estimation still produces better performance for ROSE than equal-class and exponential-class link state updates.

Assume that the actual delay of link l_j falls in class *n*, i.e., its advertised delay $B^{adv}(l_j) = B_n^{adv}$. On the per-link basis, a false positive occurs when $B^{adv}(l_j) < S < b_{l_j} < B_n$, and a false negative occurs when $B_{n-1} < b_{l_j} < S < B^{adv}(l_j)$. Therefore, the probability of a false positive and a false negative can be written as:

$$Pr\{False\ Positive,\ class=n\} = \int_{B_n^{adv}}^{B_n} \int_{B_n^{adv}}^{b} f_S(s)ds \cdot p(b)db \tag{4.9}$$

$$Pr\{False\ Negative,\ class=n\} = \int_{B_{n-1}}^{B_n^{adv}} \int_{b}^{B_n^{adv}} f_S(s)ds \cdot p(b)db \tag{4.10}$$

where $p(b)$ is the pdf of the actual delay distribution of l_j.

From equations (4.9) and (4.10), following the same procedure as in the analysis for concave constraints, the overall cost of false routing can be obtained:

$$C = c_p \cdot Pr\{False\ Positive\} + c_n \cdot Pr\{False\ Negative\}$$

$$= c_p \sum_{n=1}^{k} \int_{B_n^{adv}}^{B_n} \int_{B_n^{adv}}^{b} f_S(s)ds \cdot p(b)db + c_n \sum_{n=1}^{k} \int_{B_{n-1}}^{B_n^{adv}} \int_{b}^{B_n^{adv}} f_S(s)ds \cdot p(b)db \quad (4.11)$$

Again, to find B_n and B_n^{adv} ($n=1,\ldots,k$), one needs to solve the following equations:

$$\frac{\partial C}{\partial B_n} = 0 \Rightarrow c_p \int_{B_n^{adv}}^{B_n} f_s(s)ds - c_n \int_{B_n}^{B_{n+1}^{adv}} f_s(s)ds = 0 \quad (4.12)$$

$$\frac{\partial C}{\partial B_n^{adv}} = 0 \Rightarrow c_n \int_{B_{n-1}}^{B_n^{adv}} f_s(s)ds - c_p \int_{B_n^{adv}}^{B_n} f_s(s)ds = 0 \quad (4.13)$$

Solving equations (4.6)-(4.7) and (4.12)-(4.13) requires a certain degree of computational complexity. However, the advantage of ROSE is that once the boundaries of the classes (B_n's) and their respective advertised values (B_n^{adv}'s) are solved, they can be simply plugged into each corresponding router so that the routers will perform link state update accordingly. In a network where the traffic pattern varies at different time of the day, the traffic pattern can be first categorized into different types for different time periods (such as peak-hour/off-peak-hour traffic, etc.), then each of them will have a separate set of B_n's and B_n^{adv}'s which will be in effect during its corresponding time period. As long as the traffic of the same type does not change drastically from day to

day, (that is, say, every workday's traffic pattern between 9am to 11am is similar) the values of B_n's and B_n^{adv}'s do not need to be recalculated. Therefore, when the network is in operation, aside from applying different B_n's and B_n^{adv}'s at different time periods of the day, ROSE will not incur additional computational overhead than equal-class or exponential-class link state updates.

4.5 Simulations

This section evaluates the performance of ROSE by comparing it with the existing class-based update policies in [20]. For completeness, the equal class based and exponential class based update policies are briefly defined in the following paragraphs.

Definition 1: Equal class based update policy [20] is characterized by a constant *B* which is used to partition the available bandwidth or delay operating region of a link into multiple equal size classes: *(0,B), (B, 2B), (2B, 3B),…,* etc. An update is triggered when the available bandwidth on an interface changes to a class that is different from the one at the time of the previous update.

Definition 2: Exponential class based update policy [22] is characterized by two constants *B* and *f (f>1)* which are used to define unequal size classes: *(0,B), (B, (f+1)B), ((f+1)B, (f^2+f+1)B),…,* etc. An update is triggered when a class boundary is crossed.

4.5.1 Concave Constraint: Bandwidth

The network topology used in the simulation is a 32-node network. Two performance indices are adopted for the purpose of comparison: the update rate (average number of

updates in a unit time) and the false routing probability of connections, which are respectively defined below:

$$\text{Update rate} = \frac{\text{Total number of updates}}{(\text{Total simulation time}) \cdot (\text{Number of links})}, \text{ and}$$

$$\text{False routing probability} = \frac{\text{number of falsely-routed connections}}{\text{number of connection requests}}.$$

The arrivals of connection requests are generated by a Poisson process with arrival rate $\lambda = 1$ and the duration of each connection is derived from the standard Pareto distribution with α =2.5 (the cumulative distribution of the standard Pareto distribution is $F(x) = 1 - (\beta / x)^{\alpha}$, where α is the shape parameter and β is the scale parameter). Hence, the average duration of a connection is $d = \alpha\beta / (\alpha - 1)$ (the mean of the standard Pareto distribution). Upon the acceptance and the end of a connection, the available bandwidth is re-computed. The bandwidth requested by each connection is uniformly distributed in [b_{min}, b_{max}], that is, $q(x) \sim u(b_{\min}, b_{\max})$ in equations (4.1) and (4.2) (do not confuse this with the actual available bandwidth which is distributed in [*0, C*], where *C* is the link capacity.)

Without loss of generality, it is assumed the costs of a false positive and a negative are equal. Note that for a single class based link state update policy, the larger number of the classes the bandwidth is partitioned into, the more accurate the link state information is, implying the lower false blocking probability of connections, while the more sensitive it is to the fluctuation of the available bandwidth, thus resulting in a larger update rate. Hence, one can claim that policy 1 outperforms policy 2 if and only if, for any given number of classes used for policy 2, an appropriate number of classes can

always be found for policy 1 such that it achieves better performance in terms of both the update rate and false routing probability of connections. By extensive simulations, it is found that the link state update policy proposed in this chapter outperforms the equal and exponential class based link state update policies for any given number of classes. This section presents the simulation results of the cases that the numbers of classes of the equal class based update policy is 10 (*B=0.1C*), and for the exponential class based update policy, *B=0.05C* and *f=2* (the number of classes is 5). In the two simulations, [b_{min}, b_{max}] is set as [0, 0.05*C*] and [0.05*C*, 0.1*C*], and the number of classes of ROSE are 3 and 4, respectively.

The first step of the proposed link state update policy is to compute the classes which partition the bandwidth. Since it is assumed the requested bandwidth is uniformly distributed in the simulations and the costs of false positive and negative are equal, equations (4.6) and (4.7) can be solved as:

$$B_n = b_{min} + \frac{n \cdot (b_{max} - b_{min})}{k}$$

and

$$B_n^{adv} = \frac{(b_{max} - b_{min})}{2}$$

where *k* is the number of classes in ROSE.

Hence, the class based update policy adopted in the simulations is obtained. Figures 4.3-4.6 illustrate the simulation results, in which *Beta* denotes the scale parameter β . In both

simulations, the proposed link state update policy achieves much better performance than others, i.e., the proposed link state update policy achieves lower false routing probabilities with lower update rates than others, implying that the proposed link state update is more practical than the equal and exponential class based link state update policies in terms of the update rate and false blocking probability of connections.

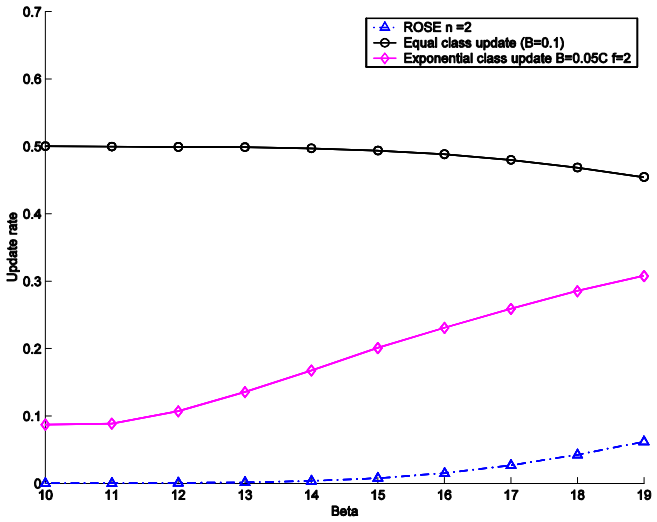


Figure 4.3 Update rate when [b_{min}, b_{max}] = [0, 0.05C]

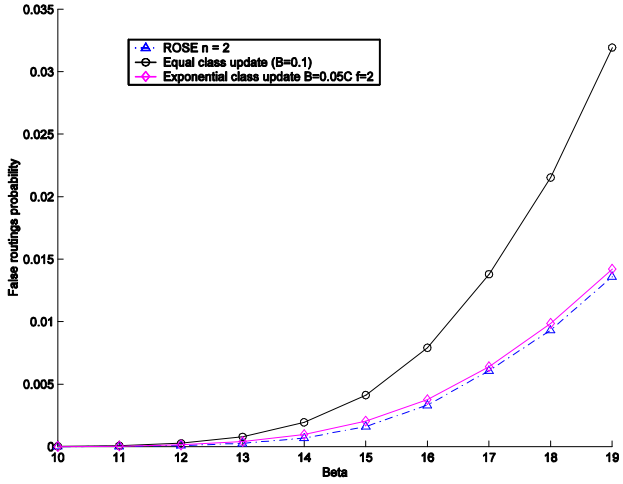


Figure 4.4 False routing probability when [bmin, bmax] = [0, 0.05C]

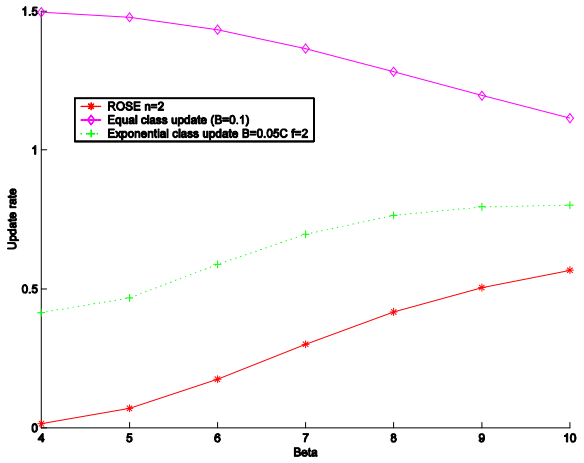


Figure 4.5 Update rate when [bmin, bmax] = [0.05C, 0.10C]

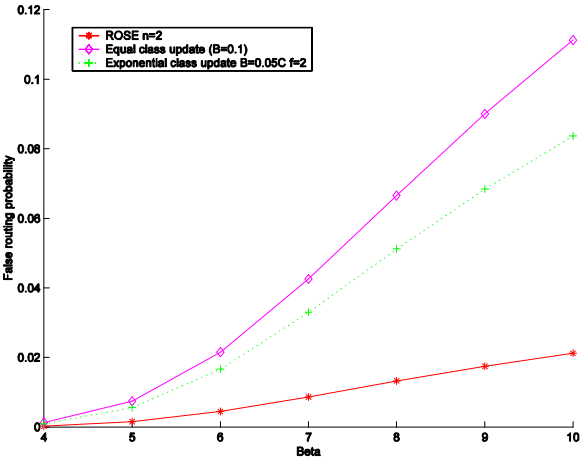


Figure 4.6 False routing probability when [bmin, bmax] = [0.05C, 0.10C]

4.5.2 Concave Constraint with Error in PDF Estimation

Since ROSE relies on the estimation of the pdf's of the user's request and the network's QoS behavior, it is important to examine the impact of erroneous estimation (in other words, fault tolerance). Here, bandwidth is used for illustrative purposes. In Figure 4.7, error is introduced in measuring the mean of the user's bandwidth request: the actual distribution is assumed to be $q(x) \sim N(0.3C, 0.02C^2)$ while the estimated pdf is

$\bar{q}(x) \sim N\left(0.3C*(1+error), 0.02C^2\right)$. In Figure 4.8, the error resides in measuring the variance of user's bandwidth request distribution. The incorrectly estimated pdf is $\bar{q}(x) \sim N\left(0.3C, 0.02C^2*(1+error)\right)$. For both experiments, the network's actual available bandwidth distribution is assumed to be exponentially distributed. The resulting probability of false routing is compared with that of the equal-class update. From these experiments, the ROSE algorithm exhibits a good degree of fault tolerance.

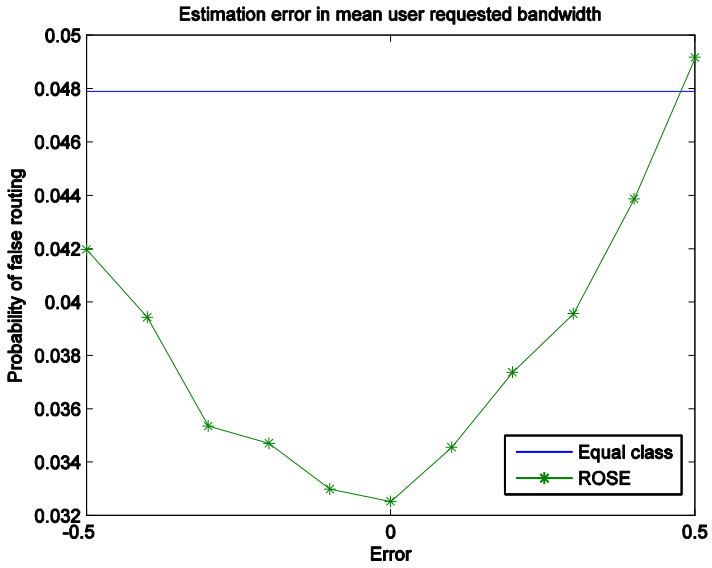


Figure 4.7 False routing probability when there is error in measuring user's mean bandwidth request. Actual request pdf $q(x) \sim N\left(0.3C, 0.02C^2\right)$.

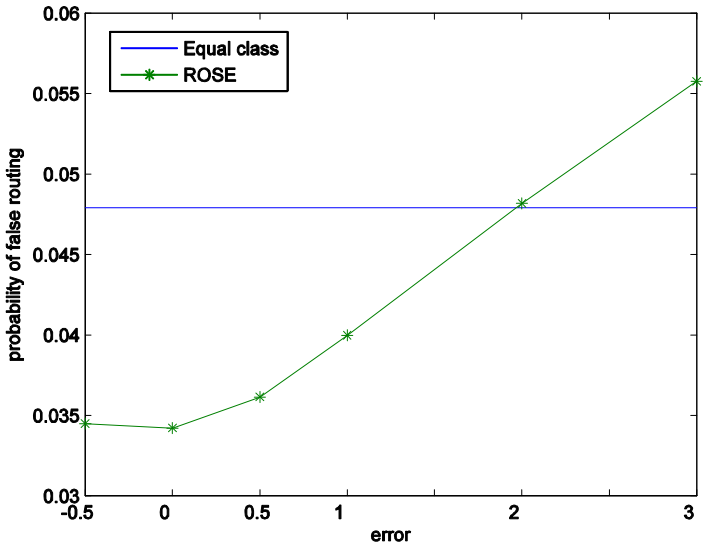


Figure 4.8 False routing probability when there is error in measuring user's bandwidth request variance. Actual request pdf $q(x) \sim N\left(0.3C, 0.02C^2\right)$.

4.5.3 Additive Constraint: Delay

Let D_{MAX} be the maximum amount of delay a link can experience in the network (e.g., queue full). The accept/reject criterion S (recall that $S = x - \sum_{i \in path, i \neq j} B^{adv}(l_i)$) is simulated as normally distributed with mean $= 0.3 \cdot D_{MAX}$, and variance $= 3 \cdot D_{MAX}$. The actual delay distribution of D_{actual} is approximated as exponentially distributed in the simulation. One hundred thousand connection setup attempts were made, each time with a different value of accept/reject criterion S and a different D_{actual}. The class boundaries and their corresponding advertised delay B_n^{adv} were calculated according to equations (4.12) and (4.13). D_{MAX} is fixed at 1000 units.

Figure 4.9 shows the results of the simulation when the number of classes varies from 3 to 12. As one can see, when the number of classes increases, the probability of

false routing from either ROSE or equal-class updates decreases. This is due to the fact that the more classes, the more accurate link state information the network can obtain. However, regardless of the number of classes, ROSE always performs better than equal-class update, especially when the number of classes is small because equal-class update does not take the accept/reject criterion *C* into consideration.

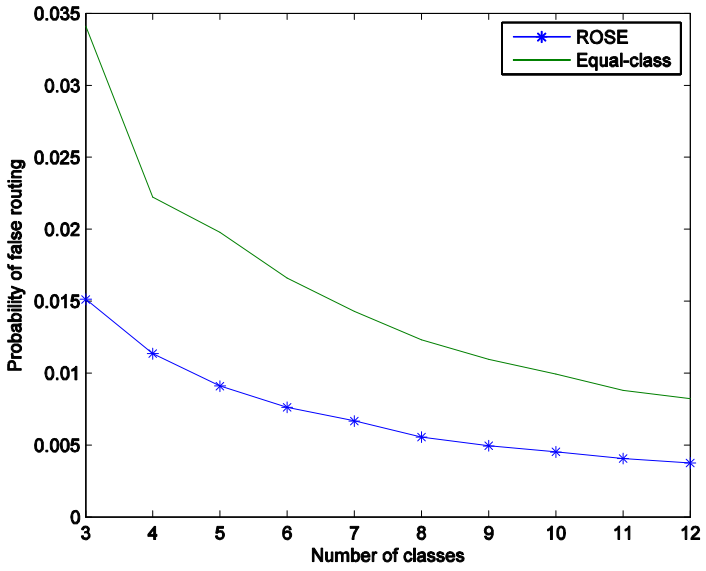


Figure 4.9 Probability of false routing with varying number of classes. (ROSE vs. equal class.)

Figure 4.10 shows the results where the number of classes is fixed to 5 but the variance of *S* is varying from 1000 to 10000. From this figure, one can see that when the variance of accept/reject criterion *S* increases, the probability of false routing increases. However, ROSE still performs better than equal-class and exponential class updates, especially when the variance is low. Since ROSE takes the probability distribution of the accept/reject criterion into consideration, the lower variance means the accept/reject criterion is more predictable, hence yielding better performance for ROSE.

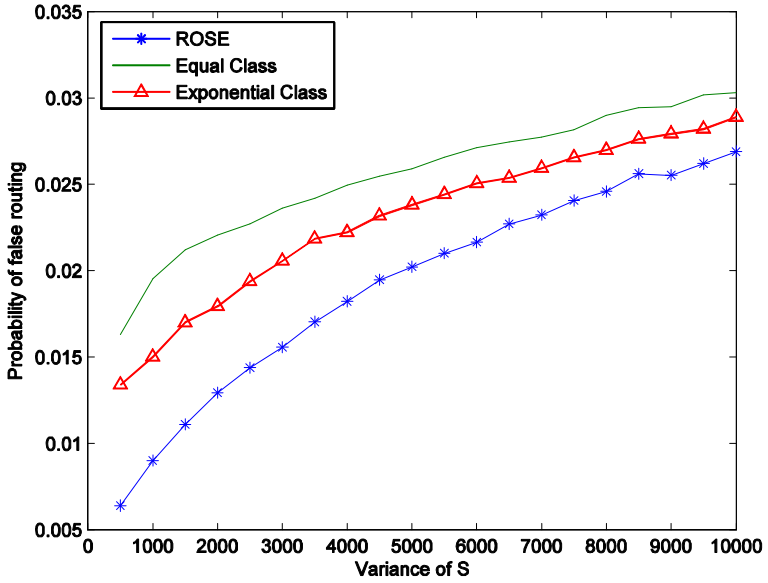


Figure 4.10 Probability of false routing with varying variance of *S*. (ROSE versus exponential class and equal class.)

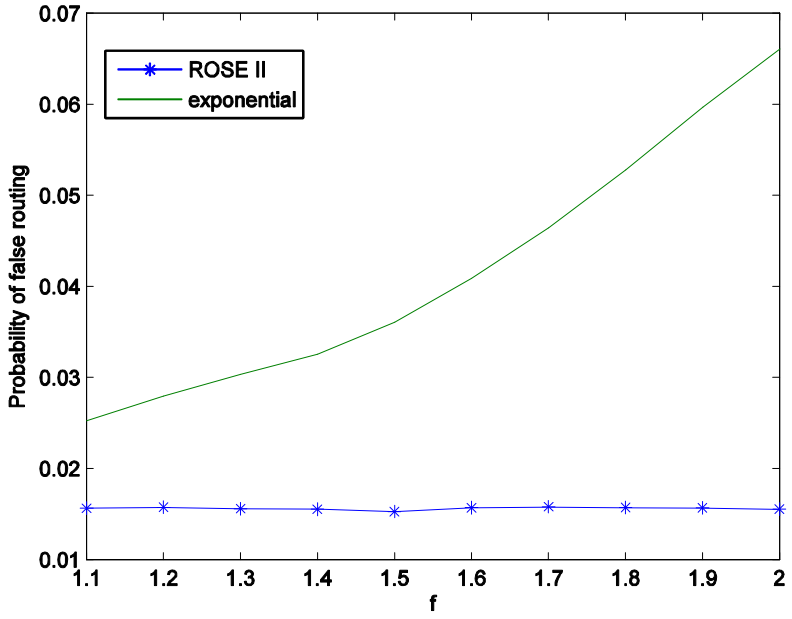


Figure 4.11 Probability of false routing with varying *f* (ROSE vs. exponential class).

Figure 4.11 compares the performance between ROSE and exponential-class update. Here, the number of classes is 5 and the variance of the accept/reject criterion is $3 \cdot D_{MAX}$. The factor *f* in exponential update varies from 1.1 to 2.0. One can see that the probability of false routing with ROSE remains almost constant because the change of *f* does not affect ROSE. However, the performance of exponential-class update decays slightly as the value of *f* increases. Exponential-class update can be viewed as a special case of ROSE in which all QoS parameters are exponentially distributed; in such case the ROSE algorithm would also yield class sizes (optimized) resembling those of exponential-class update. Nevertheless, the simulation result indicates that ROSE still performs better than exponential-class update even under exponentially distributed additive QoS parameters. This is because, for additive constraints, even if the QoS parameter of an individual link is exponentially distributed, the accept/reject criterion *S* is not. Therefore, the merit of ROSE is clearly revealed here.

4.5.4 Additive Constraints with Various Hop Counts

As it has been previously pointed out, the pdf estimation of the accept/reject criterion *S* is based on the average hop count in the network. Obviously, ROSE serves well for the connections with the hop count equal to the average hop count. However, it is important to observe the impact to the other connections with different numbers of hops. For this purpose, a network is simulated in which the delay distribution is exponentially distributed with mean 8×10^3 units and variance 6.4×10^7 unit^2. The user's request is Gaussian distributed with mean 10^5 units and variance 10^8 unit^2. The average hop count is set as 5, and therefore by applying the Central Limit Theorem, f_S, the pdf of *S*, can be

approximated by Gaussian distribution with mean $6x10^3$ units and variance of $4.2x10^8$ $unit^2$.

To observe the effect of ROSE on the connections with various hop counts away from the average, simulations are run over the connections with hop counts from as low as 2 up to 8. The performance is compared with equal-class update and exponential class update, as presented in Figure 4.12. From the result, one can see that ROSE still performs better than both exponential-class and equal-class updates for different hop counts. It is interesting to notice that the larger number of hops a connection traverses, the higher chance of false routing it suffers. This is due to the fact that the inaccuracy of the link state information will accumulate from link to link in the case of additive constraints.

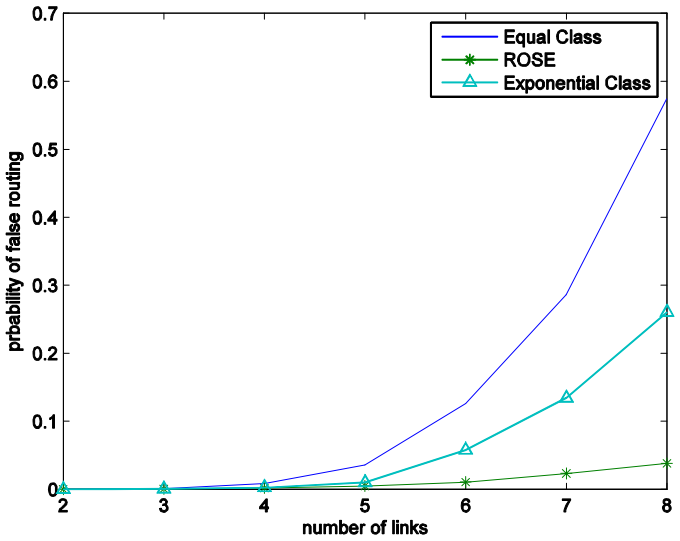


Figure 4.12 Probability of false routing with various hop count. Average hop count =5. (ROSE versus exponential class and equal class.)

4.5.5 Discussion

These simulation results demonstrate that ROSE yields lower probability of false routing than equal class update and exponential update in most of the scenarios. More importantly, ROSE also shows reasonable fault-tolerance even when the estimation of pdf is not accurate. In most of the simulations, the network QoS parameters are exponentially distributed while the user's request is normally distributed, but note that ROSE is applicable to different types of pdf's. The key here is to estimate the pdf's and solve Equations (4.12) and (4.13); the more accurate the estimation, the better the performance of ROSE.

4.6 Summary

In this chapter, it has been demonstrated that the statistical distribution of the user's QoS requirements and networks QoS measurements can be exploited to efficiently and effectively update link state information. An efficient link state update policy is proposed, referred to as ROSE. Through theoretical analysis and extensive simulations, it has shown that ROSE greatly outperforms its contenders which do not incorporate the statistical information, i.e., ROSE achieves a much lower false routing probability and reduces the cost of false routing without significantly increasing the network overhead. Furthermore, ROSE can not only be applied to networks with various types of traffic and user requests, but is also capable of handling the dynamic nature of modern network traffic. ROSE can be the fundamental building block for QoS link state update in the next generation network.

CHAPTER 5

DOWNLOADER-INITIATED RANDOM LINEAR NETWORK CODING FOR PEER-TO-PEER FILE SHARING

5.1 Introduction

Peer-to-Peer (P2P) file sharing networks have gained enormous popularity in recent years. Several well-known P2P file sharing networks include Gnutella, eDonkey, the Kazaa network, and BitTorrent. With the growing availability of broadband internet access, more people are running P2P file sharing applications on their computers. As a result, P2P file sharing has dominated the internet traffic [27]. The purpose of a file sharing network is to distribute a single file or a bundle of files from a given set of nodes to the other nodes in the network that are interested in obtaining such file(s). Traditionally, file distribution is done by a client-server model in most of the networks. Such client-server model suffers from the scalability problem: when the number of users who concurrently want to download a certain file grows, the increasing demand will eventually saturate the server(s)' capacity(ies). As a result, users will experience slow downloading speed (long downloading time) or, even worse, will not be able to gain accesses to the server(s) at all.

The introduction of P2P file sharing has solved this type of scalability problem by allowing every participating client who is downloading data from the network to act as a server that uploads information as well. Therefore, there are no longer clients and servers in such a model, only peers instead. The more users downloading a file, the more users uploading this file, and hence the more upload bandwidth. While the scalability is a

known advantage of P2P, it also brings new issues which require further investigation. One of the issues is how to efficiently distribute data among peers such that each peer can obtain the entire data as quickly as possible and the system is less susceptible to peer churns [28], [29], [30].

Among all the proposed solutions, one is to utilize network coding in P2P file sharing networks [31]. The concept of network coding was first introduced by Ahlswede *et al*. in 2000 [32]. Later on, Li *et al*. [33] proposed a well defined linear network coding algorithm. The fundamental difference between traditional networks and those with network coding is that, in traditional networks, when information is sent from the source node to the destination node, the intermediate nodes (if any) along the transmission route simply forward the information to the next node along the path: they do not alter the information nor impose any coding on this information; whereas in network coding, the intermediate nodes are allowed to apply coding on the information they are forwarding, implying that the out-going message(s) can be different from the incoming message(s) at any given intermediate node. This approach has been shown to improve the overall efficiency in multicast in a wired network, and increase the throughput in a wireless network [34], [35]. It can also provide network security against wiretapping [50], [51]. Avalanche [36] is a project which attempts to apply network coding for file sharing.

Generally, when network coding is used in peer-to-peer file sharing networks, instead of sending a particular piece of data, the peer now sends a *linear combination* of pieces to other peers. When a peer receives enough *linearly-independent* pieces of data, it will be able to reconstruct the original file. In the early stage, the benefit of network coding in a P2P file sharing network was still unclear [37]. Today, there have been

studies supporting the idea that network coding can indeed improve the efficiency of such P2P file sharing networks [38]-[40]. Deb *et al.* [41]-[43] and Mosk-Aoyama *et al.* [44] presented a series of analytical results on the performance improvements benefited by network coding in generic data-distribution networks by using gossip-based approach. The main difference between the gossip-based approach and a contemporary file sharing network is that the nodes in a file sharing network are aware of the contents of certain other peers while those in a gossip based network have no such information at all – nodes simply make random connection with each other and see whether there is useful information to exchange. Even so, the results from Deb *et al.* lead us to believe that network coding can indeed improve the performance of a P2P file sharing network.

In this chapter, therefore, the work is focused on further investigating the effectiveness of network coding in the file sharing P2P network model and offering solutions for improvements. The first part of the chapter provides a mathematical analysis on the performances of P2P networks with and without network coding. In order to adopt random linear network coding at the sending node, it is necessary to increase the field size in the binary operation in order to avoid possible "collisions" [31]. A "collision" is the incident when one peer has innovative pieces that another peer needs, but this innovative information is lost after it is linearly combined with other pieces. In order to avoid this collision, the field size of the binary operation has to be increased. However, increasing the field size introduces more overhead to network coding. Therefore, in the second part of this chapter, a "Downloader-Initiated Random Linear Network Coding (DRLNC)" scheme is proposed for the purpose of avoiding the collisions without increasing the overhead.

The rest of the chapter is organized as follows: Section 5.2 describes the general concept of P2P file sharing networks, defines the terminologies used in this chapter, and provides an overview on how network coding is applied in P2P file sharing networks. Section 5.3 presents the mathematical performance analysis of P2P file sharing networks, with and without network coding. Section 5.4 discusses the DRLNC scheme: its concept and advantages. Section 5.5 provides the results of the simulations to substantiate the analysis. Section 5.6 presents the summary.

5.2 Linear Network Coding for P2P File Sharing Networks

The purpose of a file sharing network is to distribute a single file or a bundle of files from a given node in the network to other nodes in the network that are interested in obtaining such file(s). In this chapter, the node that has the original data to be distributed is called the source node, and the nodes that are acquiring the data are called the downloading nodes. Obviously, this process of distributing data can be done by the conventional client-server protocols, such as FTP. However, the drawback of these client-server protocols is the lack of scalability: each time when a downloading node requests for the original data, the source node has to upload the entire file(s). For example, if the original data has a size of D and there are n downloading nodes, the source has to upload $D \times n$ amount of data through its upload link. Therefore, when n becomes large, the source node's network resource (i.e., upload capacity) can be exhausted. As a result, the average downloading time becomes large as the downloading nodes have to wait for their connections. The P2P file sharing network is an innovative way to solve the scalability problem encountered in the client-server mode. Its basic approach is to allow all downloading nodes (which

would be considered as clients in FTP) to behave as uploading nodes (which would be considered as servers in FTP) as well. All nodes can distribute any part of the original data they have to any other nodes that have not yet obtained this part. In the ideal scenario, the source node only has to upload the original data once, i.e., $D \times 1$, and then the rest of the downloading nodes can distribute the data among themselves until all nodes have respectively obtained the entire original data.

Terminologies:

- **Source node**: the node that contains the entire original data to be distributed (a.k.a. "seeder" in some protocols).
- **Original data**: the entire file or bundle of files that are to be obtained by other nodes.
- **Upload**: the action of transferring a piece of data a node already has to another node. A node performing an upload does not increase the amount of its local data.
- **Download**: the action of receiving a piece of data from other nodes. A node performing a download generally increases the amount of its local data.
- **Downloader**: the node that is the receiving end during a data transmission.
- **Uploader**: the node that is the sending end during a data transmission.
- **Downloading nodes**: the nodes which attempt to obtain the original data (a.k.a. "leechers" in some protocols).
- **Uploading nodes**: the nodes that are uploading part of the original data to other downloading nodes. Note that any node in a P2P file sharing network can potentially be an uploading node.
- **Upload-complete**: the situation when the information that has been uploaded from the source node is sufficient to reconstruct the original data. Theoretically, when upload-complete is reached, the source node no longer has to upload anymore information: the downloading nodes can exchange information among themselves to obtain the original data.

5.2.1 P2P File Sharing Network without Network Coding

Currently, in most of the file sharing protocols, the original data to be distributed is chopped into a number of small equal-sized pieces, referred to as “chunks” in this chapter. Each time when an upload/download occurs, a chunk is sent from one node to the other. Nodes attempt to download the chunks that they do not have, and upload the chunks they have to other nodes when requested. When a downloading node has collected all the chunks, it can then reconstruct the original file. Any node can leave the network at any given time, either before or after it receives the complete original data. Later in this chapter, the P2P file sharing network without network coding will also be referred to as “conventional P2P file sharing network”.

5.2.2 P2P File Sharing Network with Network Coding

A P2P file sharing network with network coding operates in similar ways to those without network coding as described above. The only difference is that, instead of uploading a specific chunk of the original data each time, a node now uploads a “linear combination” of multiple chunks to the downloading nodes along with the “coefficient” of this linear combination. When a node receives enough linearly independent combinations, it will be able to reconstruct the original data. At first sight, this approach seems to add extra overhead in transmission – the “coefficient” information. However, the extra overhead can improve the overall efficiency of the P2P file sharing network by overcoming some obstacles encountered by the conventional P2P file sharing networks, as will be described in the next section.

5.3 Performance Analysis – Coding vs. No Coding

For a conventional P2P file sharing network to achieve the ideal efficiency, all downloading nodes have to download chunks from the source node – and later on from each other – in a perfectly cooperative manner. The following example illustrates this point: Consider a network with 1 source node S and 10 downloading nodes (n_1 to n_{10}). The original data is divided into 10 chunks (k_1 to k_{10}). Keep in mind that in any such network, total uploads=total downloads. Therefore, when there are 10 downloading nodes attempting to acquire the 10-chunk data, the total download needed is $10\times10=100$ chunks. Ideally, the burden of providing these 100 chunks worth of downloads should be evenly shared by all nodes (load-balancing). In an ideal situation, the source node only needs to offer 10 chunks of upload, and the remaining 10 nodes upload 9 chunks each. So this leads to $10\times1+9\times10=100$ total chunks being downloaded.

Obviously, the nodes have to avoid downloading the same piece from the same source as their peers to achieve this perfect load-balancing. Therefore, in most of the conventional P2P file sharing networks, they adopt the “rarest-first” algorithm [45], which dictates the nodes to download the chunks that are currently owned by the fewest nodes first. This requires the downloading nodes to be aware of which chunks other peers currently have. If all peers have this information from all other peers, the “global-rarest-first” algorithm can be carried out properly. However, owing to the large size of most of the P2P file sharing networks, it is often impossible for all nodes to be aware of the information of all other nodes in the entire network. Therefore, most P2P file sharing networks use the “local-rarest-first” algorithm instead. In “local-rarest-first”, each node is only aware of a limited number of other peer nodes – which are referred to as its

"neighbor nodes". When a node is requesting a download, it will look for the chunk that is owned by the fewest nodes within its neighborhood. However, even though "local-rarest-first" is a reasonable alternative as opposed to "global-rarest-first" when global information is not available, the overall efficiency of such a P2P file sharing network will be less desirable than the "global-rarest-first" since it is less likely for all nodes to act cooperatively.

However, even "global-rarest-first" in a conventional P2P file sharing network has its limitations. For example, if at a given moment, there are 2 chunks that are both the rarest and there are two nodes attempting to download these chunks, there is no guarantee that each of these two nodes would select different rarest chunks to download. The application of network coding in a P2P file sharing network is an attempt to solve this kind of problems exhibited in the conventional file sharing.

5.3.1 Modeling the P2P Network

To analyze how much network coding can improve the performance of a P2P file sharing network, consider a P2P file sharing network with the following model:

1. The original data is divided into k chunks, $k_1, k_2, \ldots, k_k$.
2. There is one node with the original data (1 source node).
3. There are n nodes participating in the download (n downloading nodes).
4. All nodes are aware of the states of other nodes. This means the global-rarest-first algorithm is possible for conventional P2P networks.
5. Each time, there can be C simultaneous uploads from a node.
6. Without loss of generality, the network is assumed homogeneous, that is, the amount of time needed to download a chunk is the same for all nodes.

Based on this model, the performance of a P2P file sharing network without network coding is compared with one with network coding.

5.3.2 Performance Metrics

To evaluate the performance, one needs to observe how many "rounds" of uploads the source node has to perform until all downloading nodes have enough information to reconstruct the original file among themselves – in other words, the "upload-complete" state is reached. In the work presented in [41] and [43], the performance metric was "how many rounds does the network take for all nodes to obtain all the information". However, in a P2P file sharing network, the "upload-complete" state is a more important factor for the following two reasons: (1) the state of upload-complete marks the moment after which the original source node is no longer needed in the network. The sooner the upload-complete is reached, the sooner the source node can take off the shared file so the computer resources (disk space, upload bandwidth) can be used for other tasks, and the less likely there would be "dead files" (files that will never be completely downloaded). (2) The fewer rounds to reach upload-complete lead to fewer uploads from the source node and more uploads from the downloading nodes (leechers). This means better load-balancing – which is one of the original purposes of the P2P network: to solve the scalability problem of the client-server model. In this work, the "round" is defined as: since each node can upload to C different nodes simultaneously, each "round" refers to C such uploads. Note that not all these C uploads would be uploading different chunks (since some of the C downloaders might have requested the same chunk). Obviously, since the original data has k chunks, it takes at least $\lceil k/C \rceil$ rounds of uploads to reach

"upload-complete". So the question is: what is the probability that, in reality, upload-complete is achieved after $\lceil k/C \rceil$ rounds? This probability is greatly affected by whether the P2P file sharing network adopts network coding, which is analyzed in the next section.

5.3.3 P2P file Sharing Network with Network Coding

The model used in this chapter for a peer-to-peer file sharing network with network coding is described as the following: each time when a node uploads information, it sends a *linear combination* of chunks to the downloader. The linear combination is done in a finite base field $\mathbf{F}_q$. Without loss of generality, this analysis sets the field size to 2. (The effect of choosing a larger field size is discussed in Section 5.4.) Using the 10-chunk example (k=10*)* given in the previous section, if the uploader is sending a linear combination of chunks k_2, k_5, and k_{10}, then it sends $k_2 \oplus k_5 \oplus k_{10}$ along with a binary coefficient vector [0 1 0 0 1 0 0 0 0 1] (the 2nd, 5th and 10th elements are 1, and others 0). Here, the operator $\oplus$ means "modulo 2 addition" of the entire contents of the chunks. A node will have complete information to reconstruct the original data once it has collected 10 (=k) linearly independent binary coefficient vectors. That is, these 10 vectors can form a matrix whose rank=10 in the binary space.

Now return to the question that is being investigated: given a k-chunk original file, and each time the number of simultaneous uploads is C, what is the probability that "upload-complete" is reached after $\lceil k/C \rceil$ rounds of uploads?

First, taking a simple case when $k = m \times C$, where m is a positive integer. Therefore, $\lceil k/C \rceil = m$ and $m \times C = k$. Under this condition, after m rounds of uploads

from the source node, there will be *k linear combinations* – or *k* vectors – that have been downloaded by the downloading nodes. If these *k* vectors are all linearly independent in the binary space, then "upload-complete" is reached. So, the probability in question is essentially: "what is the probability that *k* random vectors (size $1\times k$, binary) can form a full-ranked $k\times k$ matrix in the binary space?"

To analyze this problem, consider the following facts:

1. The total number of possible linear combinations is 2^k-1. Since only binary space is considered, the coefficients (vector) of any linear combination are either one or zero. Excluding the zero vector, there are 2^k-1 possible linear combinations (vectors).

2. For *k* vectors to form a full-ranked $k\times k$ matrix in the binary space, each vector has to be linearly independent from the previous ones. So, for the *j*th vector to be linearly independent, it cannot fall into the vector space spanned by the previous *j-1* vectors. This leaves $\left(2^k-1\right)-\left(2^{j-1}-1\right)=2^k-2^{j-1}$ possibilities for the *j*th vector.

Therefore, the probability that *k* randomly selected nonzero binary vectors can form a full-ranked $k\times k$ matrix is

$$\left(2^k-1\right)^{-(k-1)}\prod_{j=1}^{k-1}\left(2^k-2^j\right) \tag{5.1}$$

Next, let's look at a more generic case when *k* is not a multiple of *C*, i.e., $k=m\times C+a$, where *m* and *a* are positive integers and $a<C$. This case becomes trickier since $(m+1)\times C$ linear combinations have been uploaded after $\lceil k/C\rceil=m+1$ rounds of uploads. That is, there can be $(C-a)$ vectors which are not linearly independent. The probability of $(m+1)\times C$ vectors to form a *k*-ranked $(m+1)\times C$ by *k* matrix is

$$\frac{\prod_{i=1}^{k-1}\left(2^k-2^i\right)}{\left(2^k-1\right)^{(m+1)C-1}}\cdot\left[\sum_{j_1=1}^{k-(C-a)}\sum_{j_2=j_1}^{k-(C-a)+1}\cdots\sum_{j_{C-a}=j_{C-a-1}}^{k}\left(2^{j_1}-1\right)\left(2^{j_2-1}-1\right)\cdots\left(2^{j_{C-a}-C-a+1}-1\right)\right] \tag{5.2}$$

To further explain equation (5.2): take $(C-a)=1$ as an example; among these $(m+1)\times C$ vectors, one of them can be linearly dependent from its predecessors. The conditional probability that the *j*th vector is linearly dependent with the previous (*j-1*) vectors, given that previous (*j-1*) vectors are linearly independent, is $\left(2^{j-1}-1\right)/\left(2^k-1\right)$. Therefore, the probability that only the *j*th vector is linearly dependent from its predecessors is

$$\frac{\prod_{i=1}^{k-1}\left(2^k-2^i\right)}{\left(2^k-1\right)^{(m+1)C-1}}\cdot\left(2^{j-1}-1\right) \tag{5.2a}$$

and *j* can vary from 2 to $(m+1)\times C$; therefore, the probability that there is exactly one vector that is linearly dependent to its predecessors among a series of $(m+1)\times C$ vectors is

$$\frac{\prod_{i=1}^{k-1}\left(2^k-2^i\right)}{\left(2^k-1\right)^{(m+1)C-1}}\cdot\sum_{j=2}^{k}\left(2^{j-1}-1\right)=\frac{\prod_{i=1}^{k-1}\left(2^k-2^i\right)}{\left(2^k-1\right)^{(m+1)C-1}}\cdot\sum_{j=1}^{k-(C-a)}\left(2^j-1\right) \tag{5.2b}$$

When *(C-a)* is greater than 1, more vectors can be linearly independent from their predecessors, thus resulting in the nested summation in equation (5.2).

Next, the case of the conventional P2P file sharing network is explored.

5.3.4 Conventional P2P File Sharing Network

In the analysis of conventional P2P file sharing networks, first again taking a simple case where $k = m \times C$. For all downloading nodes to be able to reconstruct the original data among themselves after m rounds of uploads, all the $m \times C$ chunks that have been uploaded from the source node have to be different from each other. Recall the previously described model of the conventional P2P file sharing network; the downloading nodes are aware of which chunks have been downloaded by other peers so they will not request the same chunk from the source – assuming global-rarest-first. However, as mentioned previously, global-rarest-first does not prevent two nodes from downloading the same chunk at the same time. Therefore, the cause of inefficiency lies in the situation when not all C chunks are different in one round of uploads. So, the probability that $m \times C$ uploads from the source will provide k distinct chunks is essentially the probability that all C chunks being uploaded in each round are distinct.

So for the first round's C chunks to be distinct, the probability is

$$\frac{k!}{k^C \cdot (k-C)!} \tag{5.3}$$

For the second round's C chunks to be distinct, the probability is

$$\frac{(k-C)!}{(k-C)^C \cdot (k-2C)!} \tag{5.4}$$

Since there are *m* rounds of uploads, the probability that all uploads thus far are distinct (hence "upload-complete") is:

$$k! \cdot \prod_{i=0}^{m} \frac{1}{\left(k - i \cdot C\right)^{C}} \tag{5.5}$$

Next, the analysis takes on a more generic case where *k* is not a multiple of *C*, i.e., $k = m \times C + a$, where *m* and *a* are positive integers and $a < C$. It is more difficult to formulate the probability that all *k* chunks have been uploaded by the source node after $(m+1)$ rounds of uploads, since now there can be up to $C - a$ uploaded chunks among all $(m+1) \times C$ uploads to be the same as any of their predecessors; and the probability of them being the same as their predecessors changes from round to round. Even so, it can still provide a loose upper bound for this probability:

Assuming that, for the first $(m-1)$ rounds of uploads, these $(m-1) \times C$ uploaded chunks are distinct from each other with high probability, therefore, one only needs to focus on the uploads of the last *C*+*a* chunks. This means the question can be reduced to: with $C + a$ chunks not yet uploaded, what is the probability that all of these chunks will be uploaded in the minimum number of rounds (two in this case) of uploads? This can be formulated by using the same logic:

$$\sum_{n=0}^{C-a} \{ [\frac{(C-a)!}{(C+a)^{C} (C+n)!} \cdot \sum_{i_{1}=1}^{C-n} \sum_{i_{2}=1}^{i_{1}} \cdots \sum_{i_{n}=1}^{i_{n-1}} i_{1} \times i_{2} \times \cdots \times i_{n}]$$

$$\cdot [\frac{(a+n)!}{(a+n)^{C}} \cdot \sum_{i_{1}=1}^{a+n} \sum_{i_{2}=1}^{i_{1}} \cdots \sum_{i_{C-a-n}=1}^{i_{C-a-n-1}} i_{1} \times i_{2} \times \cdots \times i_{n}] \} \tag{5.5}$$

To understand how equation (5.6) is derived, take a simple example when C=4 and a=2. The loose upper-bound assumes the first $(m-1)$ rounds of uploads all contain distinct chunks, and therefore after $(m-1)$ rounds there will be $C+a$=4+2=6 chunks from the original data yet to be uploaded. So, in order to have all remaining 6 chunks uploaded in the next two rounds, either one of the following has to be true: 1) the m-th round contains four distinct chunks (four distinct out of four total), and the $(m+1)$-th round contains two distinct chunks (two distinct out of four total), 2) three distinct chunks in the m-th round, and three distinct chunks in the $(m+1)$-th round, or 3) two distinct chunks in the m-th round, and four distinct chunks in the $(m+1)$-th round. The index n in equation (5.6) represents "the number of non-distinct chunks in the m-th round".

In order to further investigate the probability besides calculating the upper bounds, Figures. 5.1(a) and 5.1(b) are presented here to illustrate the same question from a different point of view: what is the probability that upload-complete is reached at the x-th round? After conducting some experiments based on the same mathematical model (the chunks that have been uploaded in the previous rounds will not be uploaded again, but within each round there can be non-distinct chunks uploaded), the statistical results are shown in these two figures. One can see that the probability to reach upload-complete at the $(m+1)$th round is low. In general, it takes about (m+2) to (m+4) rounds to reach upload-complete.

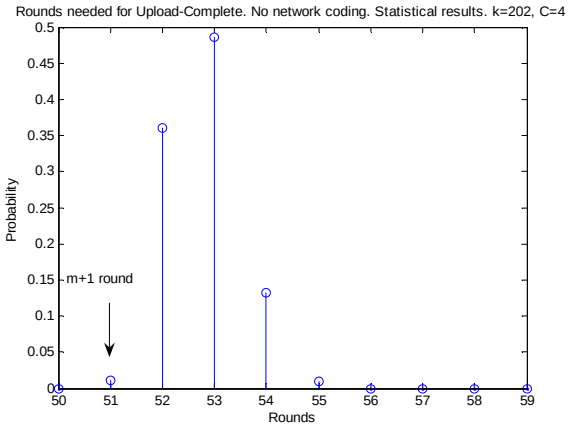


Figure 5.1(a)

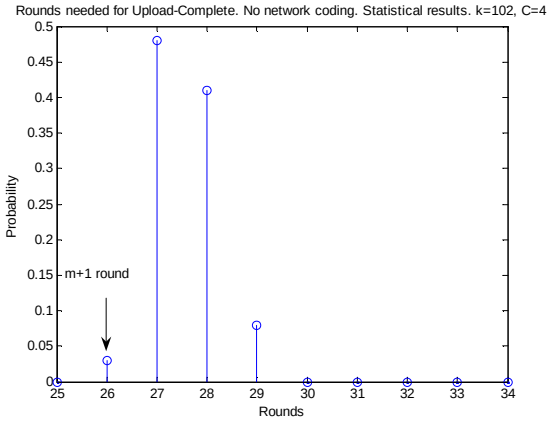


Figure 5.1(b)

Figure 5.1 Statistical results of rounds needed for upload-complete for (a) k=202, C=4. (b) k=102, C=4.

5.3.5 The Comparison: Conventional vs. Network Coding

For the simple case of $k = m \times C$, compare equation (5.1) with equation (5.5). Equation (5.1) converges to approximately 28% as k becomes large. The value of equation (5.5) depends on C; if C=4 as in the common bittorrent protocol [46], equation (5.5)

approaches zero when k becomes large. Interestingly, if C=1, equation (5.5)=1 regardless of k. This makes sense since C=1 means that there is only one chunk uploaded in each round from the source; so, there is no chance that two nodes would download the same chunk at the same time. However, in a practical network, even with C set to be 1, it still cannot guarantee that two different nodes would not request the same chunk to download from the source node. For instance, a node "A" is downloading chunk x from node "B" at time t. Meanwhile, node "C" has made a request to node "B" for chunk x. At the next moment t_2, "B" will have forwarded chunk x to A. At this moment, ideally node "C" should realize that "A" has already obtained chunk x, and therefore chunk x is no longer the rarest piece and "C" should withdraw the request from "B". However, if "B" starts to upload to "C" before "C" has made this discovery, then the duplication would still occur. In general, as long as there is a chance that two nodes would be downloading the same chunk from the source, equation (5.5) approaches zero when k is large.

For the generic case, compare equation (5.2) with equation (5.6). Interestingly, when k is large (>100), the value of equation (5.2) only depends on the value of a and C. For illustrative purposes, C is set as 4 and the effect of a is observed. When a=1, equation (5.2) is about 88%; when a=2, it is about 77%; and when a=3, it is 57%. The reason the smaller a yielding higher probability of "upload-complete" is that: with a fixed C, when C-a is larger (smaller a), it means there are more "allowable rooms" for uploads to be linearly dependent, and therefore the chance of "upload-complete" is higher. For equation (5.6), since it is a loose upper bound, its value also only depends on a and C. Again, if C=4, when a=1, equation (5.6) yields 30%; when a=2, it is 21%; when a=3, it decreases to 9.5%.

As one can see from Table 5.1, according to the analytical results, P2P file sharing networks can reach upload-complete in fewer rounds of uploads with high probability when it adopts network coding. In the next section, the efficiency of network coding in P2P file sharing networks is further investigated.

Table 5.1 Probability of Reaching Upload-complete after *(m+1)* Rounds. Generic Case: $k = m \times c + a$

	a=1	a=2	a=3
No network coding	<30%	<21%	<9.5%
Network Coding	88%	77%	57%

5.4 Downloader-Initiated Random Linear Network Coding

The previous section has provided mathematical analysis substantiating that P2P file sharing networks with network coding perform better than that of the conventional ones. When network coding is implemented in a peer-to-peer network with small field size (such as 2), it is possible that a "collision" can occur when pieces of data is linearly combined. A generic way to combat this is to increase the field size. However, increasing the field size means the size of coefficient vectors is increased. Since coefficient vectors need to be sent along each download and each content information exchange, adopting larger field size imposes more overhead to the network coding scheme. This section is dedicated to further discuss this problem and propose the Downloader-initiated Random Linear Network Coding, a new network coding algorithm designed for P2P file sharing networks that avoid the collisions without increasing the field size.

5.4.1 Random Linear Network Coding

In traditional network coding algorithms (not limited to those proposed for P2P networks), the encoding is done by the sending nodes. Gkantsidis *et al.* [31] showed, through simulations/experiments, that by applying random linear network coding in a P2P network, the efficiency can be improved. They also compared the performances of *a*) network coding only at the source node, and *b*) network coding at all nodes, and found that network coding in all nodes has better performance. Their findings are consistent with the analysis in Section 5.3. Since network coding improves the efficiency of a downloading node in distributing its data to peers just as well as it does to the source node, allowing network coding in all nodes is expected to yield better performance.

5.4.2 The "Collision"

Interestingly, when network coding is carried out in the sending nodes, there is a possibility that even though a node (say, node A) has innovative information to another node (say, node B), after a random linear combination process at node A, the resultant might no longer be innovative to B.

To illustrate this point, refer to the model described in Section 5.3: Consider node A and node B; both are downloading nodes in a P2P file sharing network with network coding. At some point during the transmission, node A has received two linear combinations, i.e., two vectors v_1^A=[1 1 0 0 1] and v_2^A=[1 0 1 1 0] (original data = 5 chunks), and node B has received one vector v_1^B=[0 1 1 1 1]. At this point, node B can be benefited by downloading either of the two vectors node A has because either v_1^A or v_2^A is linearly independent of v_1^B. However, because of random network coding, node A might

decide to distribute $v_1^A \oplus v_2^A$ instead. In such case, node B will not consider node A having useful information (since $v_1^A \oplus v_2^A = v_1^B$) and will not download from node A, thus causing a missed opportunity of downloading. Gkantsidis and Rodriguez [31] pointed out that this kind of "collision" can be reduced to a negligible level by increasing the field size to 2^{16}. However, such increase in the field size directly impacts the size of the coefficient vectors; a field size of 2^{16} requires 16 bits to represent each coefficient, instead of just 1 bit when the field size is 2. Coefficient vectors and coefficient matrices are exchanged frequently among peers. For example, according to the bittorrent protocol, peers must exchange the bitmaps of the pieces they currently have when nodes first connect with their neighbors (handshake), and keep their neighbors updated upon the reception of each additional piece. With the implementation of network coding, the bitmaps are replaced by the coefficient matrices and each update of the coefficient matrices corresponds to broadcasting a coefficient vector to all the neighbors. Therefore, larger coefficient vectors dramatically increase the overhead. In the next subsection a "Downloader-Initiated Random Linear Network Coding" scheme is proposed for the purpose of avoiding such collisions using only a field size of 2 – the minimum required to implement network coding.

5.4.3 Downloader-Initiated Random Linear Network Coding

The "Downloader-Initiated Random Linear Network Coding" (DRLNC) algorithm is based on the following rules:

1. The original data is chopped into *k* chunks of equal size. The larger the original data, the larger the *k.*
2. Each time when upload occurs, a linear combination (modulo 2) of these *k* chunks is

sent to the downloader.

3. Along with this linear combination, a "coefficient vector" is also sent to the downloader, informing the downloader the linear combination it is receiving.

4. A downloading node keeps a "coefficient matrix" of which the rows correspond to the coefficient vectors it has received.

5. The coefficient matrix at the source node is a $k \times k$ identity matrix.

6. Each node in the network selects a limited number of other nodes as its neighbors. In this chapter, this number ranges from 10 to 30.

7. A downloading node keeps a copy of the current coefficient matrix of each of its neighbors. A node informs the update of its coefficient matrix to its neighbors whenever it receives an innovative linear combination.

8. A downloading node examines its neighbors' coefficient matrices. If the downloading node deems a neighbor node has innovative information, it then makes a request to this neighbor to download. When a downloading node makes a request to download from its neighbor node, it randomly picks a linear combination among those that carry useful information, and requests such combination explicitly from this neighbor node.

9. When a downloading node receives enough information such that its coefficient matrix has a rank of *k*, it then reconstructs the original data, and changes its coefficient matrix to a $k \times k$ identity matrix like the source node.

Rule #8 above is the major difference between the proposed network coding algorithm and other existing ones. It eliminates the "collision" situation completely. To further explain how rule #8 works, here is an example:

Node A currently has the coefficient matrix:

$$M_A = \begin{pmatrix} 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 \end{pmatrix}$$

Node B currently has the coefficient matrix:

$$M_B = \begin{pmatrix} 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 \end{pmatrix}$$

Node B sees node A's matrix M_A and realizes that it can increase the rank of its own matrix (M_B) by downloading either the 2nd, 3rd, or 4th row from M_A. However, node B also sees if it downloads a linear combination of row 2 and row 3 from M_A, it will not improve the rank of M_B. Therefore, node B will randomly select a linear combination of row 2, 3, and 4, but excluding the combination of row $2 \oplus 3$. Node B then explicitly requests node A to upload the combination selected by node B. For example, if node B chooses a linear combination of row $3 \oplus 4$, it sends a request to node A asking node A to send the linear combination of row 3 and row 4 in A's matrix to B.

The exchange of the coefficient matrices among neighbor nodes imposes communication overhead, but it does not increase the complexity of the protocol, as some conventional P2P protocol [46] already allows nodes to exchange chunk information before the data transmission occurs. The size of a coefficient matrix is small as compared to the memory of a computer. For example, a 1000-chunk file would have a coefficient matrix of 125 kilobytes. Therefore, it would not be a major concern for a node to keep the coefficient matrices of all its neighbor nodes.

The proposed algorithm does require some computational overhead in the downloading nodes, such as calculating whether a neighbor node has innovative information. However, the extra computation required here as opposed to other network coding algorithms is simply the calculation of the rank of a binary matrix in the binary space. The benefit of this extra computation can be significant as illustrated in the next section.

5.5 Simulations

In this section the performances of the proposed DRLNC are compared with those of the traditional network coding where the random linear coding is performed at the uploading nodes.

The network parameters considered in these simulations are: N – the total number of nodes; Filesize – the number of chunks in the original data; C – the number of simultaneous uploads, and neighborhood size – the number of neighbor nodes of each node. Each simulation starts with one source node with the complete original data to be distributed and ends when all nodes have obtained the entire original data. The field size is set to 2 for both DLRNC and the traditional network coding. The performance metric is the number of rounds each simulation takes – the fewer rounds to distribute the original data to the entire network, the better the performance. Readers are reminded not to confuse this performance metric with the one used in Section 5.3 – where the performance metric was defined as "the rounds needed for upload-complete". The difference is that the comparison offered in Section 5.3 is the analytical results between network coding and no network coding at all; in this section, the comparison is the simulation results between traditional network coding and DRLNC.

The first comparison is presented in Figure 5.2, where N=100, C=4, and neighborhood size=10. The effect of changing the Filesize is observed. Naturally, the larger the Filesize the more rounds it will take to distribute the data. However, it is clear that the downloader-initiated random linear network coding performs better than the traditional network coding, especially when the original data has more chunks.

Figure 5.3 illustrates how the neighborhood size would influence the performances. One can see that the downloader-initiated random linear network coding performs better than the traditional network coding for all different numbers of neighbor nodes. The margin is larger when the neighborhood size is smaller. This makes sense because when a downloader has fewer neighbor nodes, it has fewer random linear combinations to choose from, and therefore it is more difficult to avoid the "collision" situation. The fact that the downloader-initiated random network coding performs better with smaller neighborhood size is a significant advantage: this means the nodes do not have to exchange information about their coefficient matrices with a large number of neighbor nodes in order to achieve the same performance as the traditional network coding.

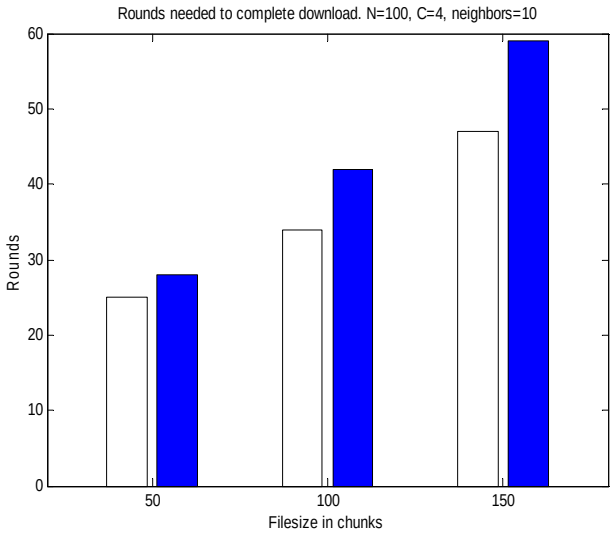


Figure 5.2 Performance comparison: traditional network coding vs. downloader-initiated network coding. Network size=100 nodes, neighborhood size=10 nodes. *C*=4. Filesize changes from 50 to 150.

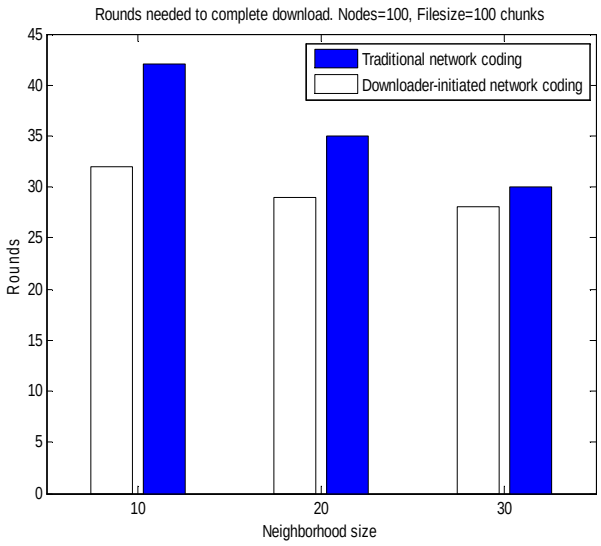


Figure 5.3 Performance comparison: traditional network coding vs. downloader-initiated network coding. Network size=100 nodes, Filesize=100 chunks, C=4, neighborhood size=10 to 30.

In Figure 5.4, the simulation sets *N*=100, Filesize=100 chunks, neighborhood size=10 nodes, and changes *C* from 3 to 7. As expected, it takes fewer rounds to complete the data distribution when *C* (number of simultaneous uploads) is larger, i.e., more upload-download bandwidth is used. Figure 5.5 shows comparison performances with various network sizes (*N* from 60 to 120) while keeping other parameters unchanged. Figure 5.4 and Figure 5.5 indicate that the advantage of DRLNC is not sensitive to the change of *C* or the total network size: it takes 10 to 15 percent fewer rounds to complete the download than the traditional network coding in all scenarios.

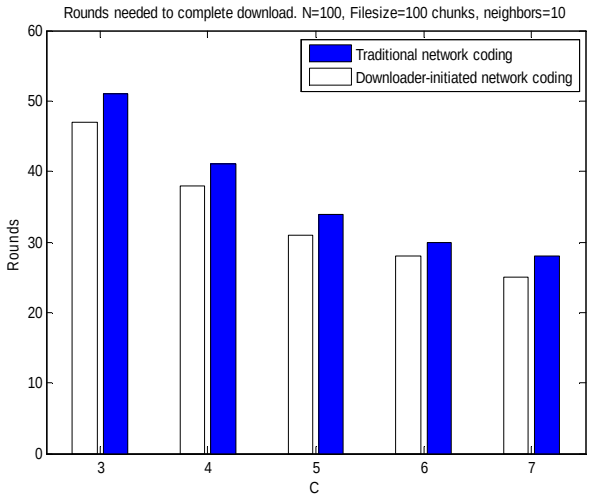


Figure 5.4 Performance comparison: traditional network coding vs. downloader-initiated network coding. Network size=100 nodes, *Filesize*=100 chunks, neighborhood size=10, *C*=3 to 7.

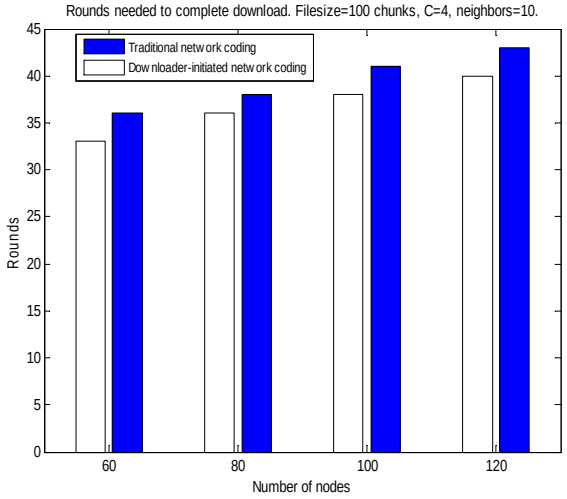


Figure 5.5 Performance comparison: traditional network coding vs. downloader-initiated network coding. *Filesize*=100 chunks, *C*=4, neighborhood size=10, Network size=60 to 120 nodes.

Table 5.2 compares the overhead incurred by exchanging the coefficient matrices/vectors in a network. DRLNC using field size of 2 is compared with traditional network coding using field size of 8. Each entry represents the total bytes used by one

peer to exchange coefficient matrices/vectors with their neighbors until all nodes have completed downloads. Both approaches achieve similar performances in terms of rounds needed to complete download, but DRLNC requires less overhead.

Table 5.2 Overhead Comparison: DRLNC vs. Traditional Network Coding, Neighbors=10, Total Number of Nodes=100

	Filesize=50	Filesize=100	Filesize =150
DRLNC	3.465MB	12.99MB	28.22MB
Traditional network coding	24.75MB	99.90MB	222.8MB

In summary, from these simulation results one can conclude that downloader-initiated random linear network coding generally performs better than the traditional network coding. Its margin of improvement is less sensitive to the size of the network and the value of C, but more relevant to the file size and the neighborhood size; the larger the file size and/or the smaller the neighborhood size, the bigger the margin.

5.6 Summary

In this chapter, the effectiveness of network coding for P2P file sharing has been investigated with mathematical analysis, and the Downloader-initiated Random Linear Network Coding is proposed to further improve the performance of network coding in the environment of P2P file-sharing networks. The mathematical models presented in this chapter are specifically modeled after the contemporary file-sharing protocols. The results from these analyses indicate that network coding can indeed help a P2P file sharing network distribute data more effectively. This work has also pointed out the “collision” problem in the traditional network coding. Previously proposed solution of

this problem is to increase the field size, which leads to a larger overhead. This dissertation proposed the Downloader-Initiated Random Linear Network Coding algorithm to mitigate this problem without increasing the field size. Through extensive simulations, the proposed scheme has been demonstrated to outperform that of existing P2P network coding algorithms without additional overhead.

CHAPTER 6

INDENTIFYING THE NETWORK CODING OPPORTUNITY IN WIRELESS AD HOC NETWORKS

6.1 Introduction

In recent years, network coding has been widely studied for various networking applications. The concept of network coding was first introduced in 2000 [32]. Later on, Li *et al*. [33] proposed a well defined linear network coding algorithm. Network coding has been shown to improve the overall efficiency in multicast in a wired network, and increase the throughput in wireless networks [35]. While the throughput of unicast traffic in a wired network cannot be benefited by network coding, it is not so in a wireless environment. Katti *et al.* [34] proposed the COPE network coding scheme which performs coding at the packet level on unicast traffic in a wireless network, and showed that the coding gain can vary depending on the topology. Based on the concept of COPE, there have been various studies on how to take advantage of wireless network topologies to design practical and effective network coding schemes [47]-[48]. For a network coding scheme to be practical, it must be scalable and computationally economical; for it to be effective, it must yield high coding gains. Based on these concerns, this dissertation proposes a network coding algorithm referred to as "Identifying the Coding OPportunity" (ICOP) for unicast packets. ICOP makes two major contributions. First, ICOP simplifies the MAC Layer scheduling and saves communication overhead: ICOP proactively calculates the coding configurations and their corresponding MAC layer scheduling. Each node that participates in network coding will encode, decode, and send the

appropriate packets according to a pre-determined schedule; nodes do not make coding decisions "on the flight", and so there is no need for extra communication overhead. Second, ICOP takes the traffic pattern information into consideration to improve the practical coding gain: In COPE, the theoretical coding gain was calculated under the assumption that all participating nodes have packets to send whenever the MAC layer network-coding timeslot arrives. In practice, if any of such nodes does not have a packet to send in this timeslot, the practical coding gain will be less than the theoretical coding gain. By considering the traffic pattern, ICOP can look for the coding configuration that accommodates the most amount of traffic. Furthermore, ICOP maintains its scalability by breaking down a large network into small "coding cells", as will be described in detail in Section 6.3. The rest of this chapter is organized as follows: Section 6.2 briefly reviews the network coding mechanism in a wireless network and how it can improve the throughput; Section 6.3 introduces the proposed ICOP algorithm; Section 6.4 presents simulation and comparison results, and the conclusion is given in Section 6.5.

6.2 The Network Coding Opportunity

Consider a wireless network, $G(V, E)$, where V is the set of all wireless nodes and E is the set of wireless links. Let A_i ($i=1,\ldots, m$; m is the total number of nodes in the network) be a set of nodes that are one-hop neighbors of node i, i.e., $A_i = \{V_k : \forall\, k \;\exists\; V_k$ is a one-hop neighbor of $V_i\}$. The concept of "network coding opportunity" is illustrated by the two examples shown in Figure 6.1. In the network on the left in Figure 6.1, $A_1=\{V_4, V_r\}$, $A_2=\{V_3, V_r\}$, $A_3=\{V_2, V_r\}$, and $A_4=\{V_1, V_r\}$; one possible *coding configuration* is indicated by the arrows in the figure: V_3 has a packet $p_{3\text{-}1}$ to be delivered to V_1, and V_4 has

a packet $p_{4\text{-}2}$ to be delivered to V_2. The transmission ranges of these nodes are not long enough to reach their respective destinations directly, and so both packets have to go through the relay node. Without loss of generality, further assume that the packets have equal length and their transmission times are the same – one timeslot. With the conventional store-and-forward method, it would take 4 timeslots to complete the delivery of these two packets. On the other hand, using network coding, the two packets can be delivered in 3 timeslots: At timeslot 1, V_3 transmits $p_{3\text{-}1}$ to V_r. Meanwhile, V_2 which is within the transmission range of V_3 overhears this packet and stores it in its buffer. At timeslot 2, V_4 transmits $p_{4\text{-}2}$ to V_r while V_1, in a similar position as V_2, overhears it and stores it in its buffer. At timeslot 3, V_r transmits the XOR'ed packet $p_{3\text{-}1} \oplus p_{4\text{-}2}$ which can be received by all other four nodes. After the third timeslot, both V_1 and V_2 can decode the XOR'ed packet using their previously overheard packets to obtain $p_{3\text{-}1}$ and $p_{4\text{-}2}$, respectively. Comparing this scheme with the traditional store-and-forward method which would take 4 timeslots to deliver the same two packets, the coding gain is (theoretically) 4/3=1.33. The "overhearing" process is possible only because of the broadcast nature of wireless transmission, commonly referred to as "opportunistic listening" in the wireless network coding literature. The network on the right in Figure 6.1 (star topology) is a special case where each node's transmission range includes every node in the network *except* the node that is directly opposite to the relay node. That is, A_1={ V_2 ,V_3 ,V_5 ,V_6 ,V_7}, and so on. Here, V_7 is the relay node. According to COPE, the packets from each node (1,2, … , 6) destined to their respective opposite nodes (1 to 4, 2 to 5, and so on; note that the arrows are bi-directional) can be coded in the following manner: from timeslot 1 to 6, nodes 1 to 6 transmit their packets to the relay node in turn;

at timeslot 7, node 7 transmits the XOR'ed packet of all these 6 packets. At the end of the 7^{th} timeslot, all six nodes (1 through 6) will receive their designated packets. In this case, the theoretical coding gain is 12/7 (12 timeslots needed for store-and-forward). Obviously, this kind of special case does not exist often since it requires all nodes to be positioned near perfectly. However, when it exists, it produces higher coding gain than other ordinary scenarios because twice the number of packets can be coded together. In this chapter, this kind of special case is referred to as "*full-duplex*" since every sending node is also a receiving node in its role of network coding.

Note that the coding gains mentioned in this section are considered "theoretical" because they can only be achieved if there are appropriate packets to be encoded. The details of coding gains is further investigated in the following sections.

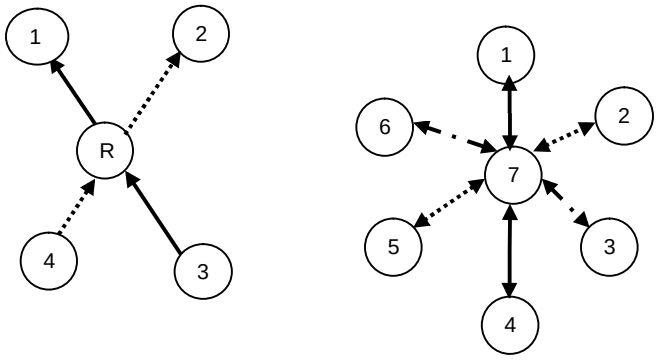


Figure 6.1 Left: X topology, where $A_1=\{V_4, V_R\}$, $A_2=\{V_3, V_R\}$, $A_3=\{V_2, V_R\}$, and $A_4=\{V_1, V_R\}$. Right: star topology, where $A_1=\{ V_2 ,V_3 ,V_5 ,V_6 ,V_7\}$, $A_2=\{ V_1 ,V_3 ,V_4 ,V_6 ,V_7\}$, $A_3=\{ V_1 ,V_2 ,V_4 ,V_5 ,V_7\}$, $A_4=\{ V_2 ,V_3 ,V_5 ,V_6 ,V_7\}$, $A_5=\{ V_1 ,V_3 ,V_4 ,V_6 ,V_7\}$, $A_6=\{ V_1 ,V_2 ,V_4 ,V_5 ,V_7\}$, and $A_7=\{ V_1 , V_2 ,V_3 ,V_4 ,V_5 ,V_6\}$. Opportunistic listening is allowed.

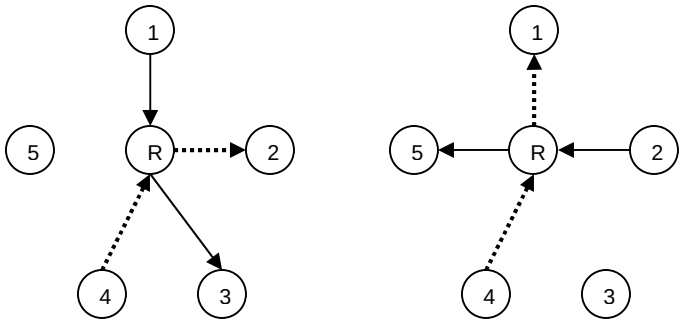


Figure 6.2 Two possible configurations in the 5-node network. A_1={ V_2 ,V_5 ,V_R}, A_2={ V_1 ,V_3 , V_R }, A_3={ V_2 ,V_4 ,V_R }, A_4={ V_3 ,V_5 ,V_R }, A_5={ V_1 ,V_4 ,V_R } and A_R={ V_1 , V_2 ,V_3 ,V_4 ,V_5}. Opportunistic listening is allowed.

6.3 Identifying the Coding Opportunity (ICOP) Algorithm

The theoretical coding gains mentioned in Section 6.2 are only related to the topology. To achieve these gains in practice, all sending nodes must have packets to send to their respective receivers whenever the MAC layer assigns timeslots to these nodes for network coding. If any of these nodes does not have a packet to send, the theoretical network coding gain cannot be achieved. To illustrate this point, refer to the six-node network in Figure 6.2. If node 1 has a packet, $p_{1\text{-}3}$, to be delivered to node 3, and node 4 has a packet, $p_{4\text{-}2}$, to be delivered node 2, then network coding can be performed as described in the previous section. However, if $p_{1\text{-}3}$ has been transmitted to the relay node but node 4 does not have a $p_{4\text{-}2}$ to send, then the relay node might not have a suitable packet to be XOR'ed with $p_{1\text{-}3}$. At this moment, the relay node can choose to wait a few timeslots for $p_{4\text{-}2}$ to arrive, or simply forward $p_{1\text{-}3}$ as is to node 3 in the very next timeslot. Waiting for $p_{4\text{-}2}$ increases delay in delivering $p_{1\text{-}3}$, while forwarding $p_{1\text{-}3}$ as is sacrifices the coding gain; either of these two choices can cause undesirable impact. Another choice for the relay

node is to look among some other packets currently available that are suitable to be XOR'ed with $p_{1\text{-}3}$. For instance, if at the same instance node 3 has a packet to be delivered to node 5, then node 3 can cease the channel and transmit $p_{3\text{-}5}$ to the relay node. Then, the relay node can transmit $p_{1\text{-}3} \oplus p_{3\text{-}5}$ at the very next timeslot, and nodes 3 and 5 can decode. However, this maneuver requires the network nodes to make intelligent decisions "on the flight"; this not only complicates the MAC layer scheduling, i.e., somehow the MAC layer has to know that $p_{3\text{-}5}$ can be coded and so it grants node 3 the channel, but also calls for extra communication overhead, i.e., nodes 3 and 5 have to be informed to decode $p_{1\text{-}3} \oplus p_{3\text{-}5}$ with some of their previously overheard packets while other nodes will disregard this coded packet. Based on the above reasons, this dissertation proposes ICOP, a network coding algorithm that does not require the nodes to make coding decisions on the flight, and yet still maintains network coding efficiency. ICOP instructs the nodes which packets to encode and which packets to decode, and how to decode before the actual transmission starts. ICOP also informs the MAC layer to designate appropriate scheduling for packets that are network coded or going to be network coded. During the actual transmission, each node would simply follow the pre-determined instruction to send and receive packets, and so no extra communication overhead is needed. The tradeoff of using this pre-determined coding method is the sacrifice of some coding gain. For example, with the same scenario mentioned above (Figure 6.2 left), assume ICOP has designated a network coding configuration which involves $p_{1\text{-}3}$, $p_{4\text{-}2}$, and $p_{1\text{-}3} \oplus p_{4\text{-}2}$. Then, in the absence of $p_{4\text{-}2}$, the relay node will have no choice but to send $p_{1\text{-}3}$ as is at the timeslot in which it was supposed to send $p_{1\text{-}3} \oplus p_{4\text{-}2}$, which leads to no coding gain in this round of transmissions. Therefore, in order to maintain high coding gains, one has to reduce the occurrence of the

absence of packet(s) to be coded. ICOP attempts to do so by taking the traffic pattern into consideration and uses the traffic information to find the coding configurations that can potentially encode the most amount of packets.

To explain how ICOP works in detail, refer to the network in Figure 6.2. In this network, each node (1, 2,… ,5) can hear the transmissions from the relay node and its two adjacent nodes. This topology contains many possible *coding configurations*, for example: 1) node 1 to 3, and 4 to 2, 2) node 2 to 5, 4 to 1,etc. ICOP is designed to choose the configuration that can potentially code the most number of packets, and hence yield the highest practical coding gain. It is assumed that (1) the one-hop neighbors of each node is known, (2) the traffic pattern in the network is known *a priori*, and (3) the one-hop neighbor information of each node is disseminated to all other nodes in the network. Besides the network, $G(V,E)$, the traffic pattern is also considered, which is denoted by $\Lambda=\{\lambda(i,j): i \neq j, \forall V_i, V_j \in V\}$ where $\lambda(i,j)$ is the total traffic rate from node *i* to node *j*, including any transit traffic (traffic that is not originated from nor destined to *i* or *j*). Essentially, ICOP is to find the optimal coding configuration based upon the graph $G(V,E)$ and Λ. The algorithm consists of two phases:

1: Relay node selection – the node with the largest number of neighbor nodes will be selected as the relay node. In a large network where there is more than one relay node, each relay node and its one-hop neighbors form a "*coding cell*". This is an important step that makes ICOP scalable.

2: Coding opportunity search – based on the relay node(s) selected in the previous phase, the algorithm uses $G(V,E)$ and Λ to calculate the optimal coding configuration *within each cell*.

The remainder of this section describes the algorithm in detail:

Phase 1: Relay Node Selection

Step 1: The algorithm will simply pick the node with the largest number of neighbor nodes as the relay node because it potentially provides the most coding opportunity. When there are two or more nodes tied with the highest number of neighbor nodes, the algorithm will randomly select one node as the relay node.

Step 2: Once a relay node is determined, this relay node and all its one-hop neighbors form a "*coding cell*". A temporary subgraph of *G*, say, $G^1(V^1, A^1)$, will be generated by removing this coding cell from *G*.

Step 3: The algorithm will repeat Steps 1 and 2 to pick the second relay node based on G^1, and then generate a new subgraph G^2 by removing the second coding cell from G^1.

The algorithm repeats Steps 1 and 2 recursively until the latest subgraph generated contains no nodes with more than 1 neighbor, i.e., no more coding cell can be created.

Phase 2: Coding Opportunity Search

Once the relay nodes are selected, the coding opportunities within each coding cell are searched. Note that in the proposed algorithm the searching process only has local-significance within the coding cell. In other words, if there are *n* relay nodes in the entire network (*n* coding cells), then the search process computes the coding opportunities as if these are *n* separate networks. The algorithm of searching the coding opportunities within each coding cell is described below:

Step 1: Pick any node from the one-hop neighbors of the relay node. Denote the relay node as node *r*, and then pick any node $V_{i(1)}$ from A_r.

Step 2: Pick any node that is in A_r but not in $A_{i(1)}$. That is, pick $V_{j(1)} \in \left\{A_r \setminus A_{i(1)}\right\}$. Assume $V_{i(1)}$ is the sending node and $V_{j(1)}$ is the receiving node.

Step 3: Pick another node $V_{i(2)} \in A_r \cap A_{j(1)}$ as another sending node.

Step 4: Pick a node $V_{j(2)} \in A_r \cap \left\{A_{i(1)} \setminus A_{i(2)}\right\}$ as the receiving node.

Step 5: Repeat Steps 3 and 4; pick sending node(s) $V_{i(n)}$ such that $V_{i(n)} \in A_r \cap \left\{\bigcap_{k=1}^{n-1} A_{j(k)}\right\}$ and receiving node(s) such that $V_{j(n)} \in A_r \cap \left\{\bigcap_{k=1}^{n-1} A_{i(k)} - A_{i(n)}\right\}$, until no more such sending/receiving pair can be found. At this point, all the sending/receiving pairs found so far are considered as one "configuration", denoted by two *lists* of nodes $\mathbb{S} = \left\{V_{i(1)}, V_{i(2)}, ..., V_{i(n)}\right\}$ and $\mathbb{R} = \left\{V_{j(1)}, V_{j(2)}, ..., V_{j(n)}\right\}$, where $\mathbb{S}$ represents the list of *n* sending nodes and $\mathbb{R}$ represents the list of *n* receiving nodes. Each pair $(V_{i(a)}, V_{j(a)})$ is a sending/receiving pair. Note there must be an even number of sending/receiving pairs to code unicast packets (i.e., *n* is an even number), and each two consecutive pairs [$(V_{i(a)}, V_{j(a)})$,$(V_{i(a+1)}, V_{j(a+1)})$] (*a=1,3,5,7,…,n-1*) form a *coding pair*, implying that the packets from these two pairs will be XOR'ed at the relay node.

Step 6: Calculate the total traffic rate among all the *coding pairs* found at the end of Step 5. That is, total traffic rate $\lambda_{total} = \sum_{a=0}^{n/2-1} \min\left(\lambda(V_{i(2a)}, V_{j(2a)}), \lambda(V_{i(2a+1)}, V_{j(2a+1)})\right)$.

Step 7: Look for all other possible coding configurations around this relay node. Eventually, select the configuration that yields the maximum λ_{total}.

Step 8: Look for the special "full-duplex" opportunity. Full-duplex coding is possible only if the following condition is met: for every sending/receiving pair, its

sending node and receiving node share the same one-hop neighbors around the relay node. That is, $A_r \cap A_{i(k)} = A_r \cap A_{j(k)}$ for $k=1,\ldots, n$, where n is the total number of sending/receiving pairs in this configuration.

To further understand the second phase of the algorithm, refer to Figures. 6.3 and 6.4. Essentially, ICOP attempts to construct a tree diagram as in Figure 6.3. (The tree shown here is built upon the 6-node network in Figure 6.2. For clarity, the marks for the bottom layer under nodes 2 to 5 are omitted.) Each *path* from top to bottom represents a valid coding configuration in the network. For example, the left-most path 1-3-4-2 represents the coding configuration where node 1 and 4 sending and node 3 and 2 receiving (referred to as one *coding pair*: 1-4 and 3-2). The weight of each coding pair is min ($\lambda(i_1,j_1)$, $\lambda(i_2,j_2)$) where i_1 and i_2 are the two sending nodes; j_1 and j_2 are the two receiving nodes. ICOP picks the path that has the highest total weight as the optimal configuration. Figure 6.4 is the flow chart of the algorithm from Step 1 to Step 7 in the second phase of the algorithm. The process of finding a sender or a receiver follows the same rule stated in Step 5 above.

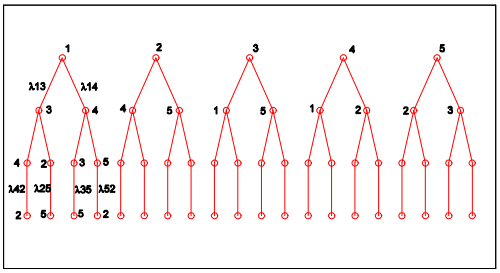


Figure 6.3 Tree diagram of possible coding configurations for the network in Figure 6.2.

Once ICOP has chosen a coding configuration, the network then operates accordingly: all the packets that traverse between each *coding pair* will be sent with

network coding; all other packets will be sent in the conventional mean. When the network topology and/or the network traffic pattern changes, ICOP will have to use the newly changed information to recalculate the coding configurations in order to maintain the network coding efficiency. It is worth mentioning that MAC layer scheduling is another important issue in wireless network coding. ICOP coordinates the MAC layer scheduling within a coding cell to ensure that nodes transmit their packets in the right sequence, i.e., each node with a native packet to send transmits it in turns, and then the relay node transmits the encoded packet.

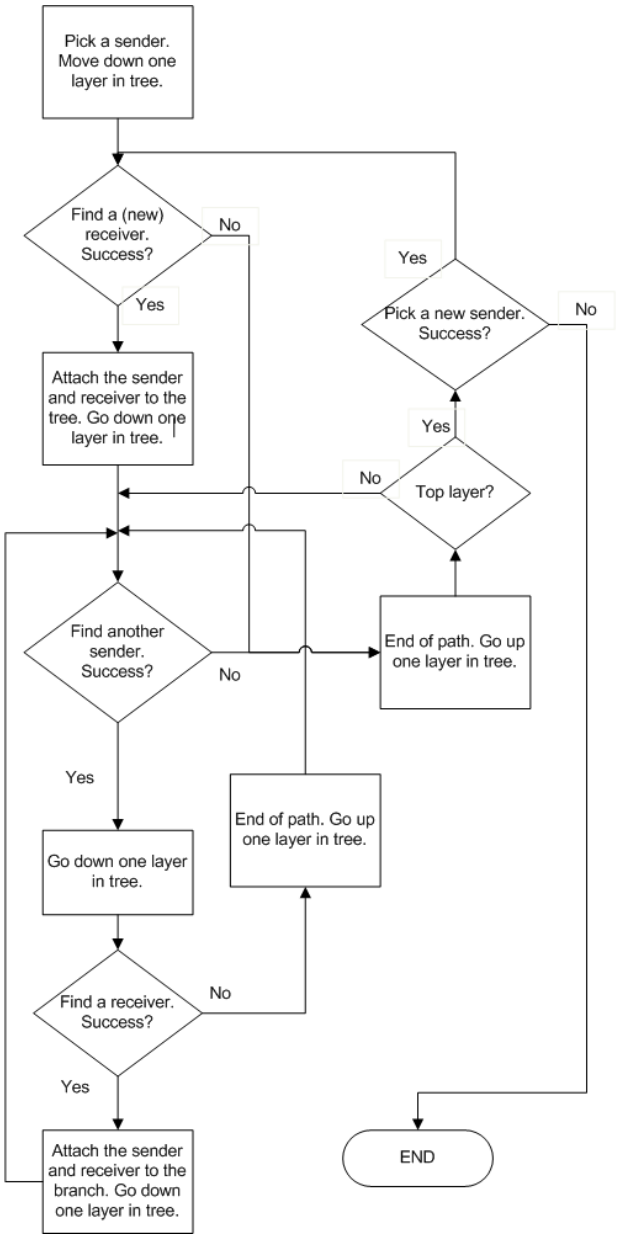


Figure 6.4 Flow chart of the second phase of ICOP algorithm (steps 1 to 7).

6.4 Simulations

The ICOP algorithm is illustrated via two examples. The first example is a small 9-node wireless ad-hoc network as shown in Figure 6.5. The transmission range of each node is 70. For illustration purposes, $\lambda(i,j)$ is simplified as $\lambda(i,j) = \lambda(i)$ for all j, i.e., the traffic rate

from node *i* to any other node *j* in the network is the same. As a result, instead of a 9 by 9 matrix for Λ, there is one simplified 9 by 1 matrix Λ=[0.95, 0.23, 0.61, 0.49, 0.89, 0.76, 0.46, 0.02, 0.82]. Figure 6.5 shows the outcome of ICOP: node 1 is selected as the relay node since it has the largest number of neighbors (all but node 2). Then, the seven nodes (from node 3 to 9) around node 1 are potential candidates for network coding. After considering the traffic rate matrix Λ, ICOP selects three pairs around node 1 to perform network coding: nodes 3-8, 6-7, and 9-4. Note that this is only a half-duplex configuration, i.e., in timeslot 1-4, nodes 3, 6, and 9 are the sending nodes while nodes 8, 7, and 4 are the receiving nodes, respectively; then, in the next 4 timeslots the nodes swap their sending/receiving roles. The theoretical coding gain is 6/4=1.5.

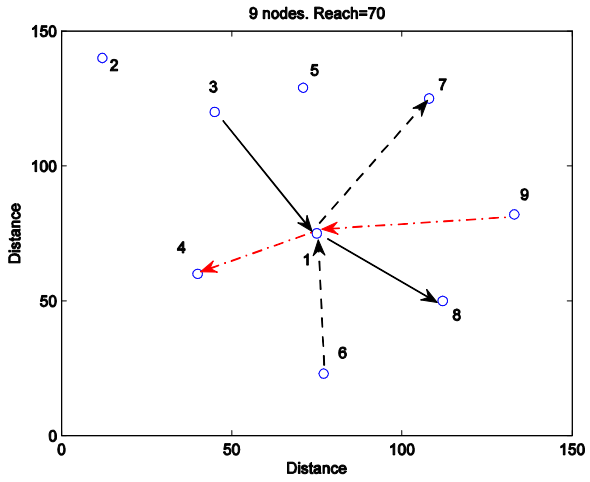


Figure 6.5 The 9-node example.

Next, the practical coding gains between ICOP and COPE is compared using the 6-node network featured in Figure 6.2. Each node from 1 to 5 generates packets randomly designated to the other nodes except the relay node. If the packet is designated to an adjacent node, e.g., from node 2 to node 3, the packet will be sent directly without the

involvement of the relay node. Otherwise, the packet is first sent to the relay node. The relay node will then either store-and-forward this packet to its destination or XOR it with other packets, depending on the pre-calculated configuration from ICOP. The operation of the relay node in both ICOP and COPE is set to minimize delays. If the relay node does not have the appropriate packets to encode when it obtains access to the channel, it will forward the packet as is. To avoid collision, only one node can transmit a packet at any given time; each packet is assumed to have the same length, and hence requires the same transmission time (one unit time). In the simulation, the total packet arrival rate is 0.4/unit time. Two pairs of non-adjacent nodes (i.e., nodes 1-4 and 2-5) are assigned to have higher bi-directional traffic rates within the pair than the rest of the nodes. In other words, node pair 1-4 and node pair 2-5 are the two "busy routes" that have more packets to be delivered than the rest of the routes. Denote the traffic rates on these two busy routes as $\lambda(1,4)$ and $\lambda(2,5)$. Without loss of generality, further assume that $\lambda(1,4) = \lambda(2,5) = \lambda_B$. Let λ_{avg} represent the traffic rate of each of the non-busy routes, then the traffic rate of the busy route can be quantified using a factor f such that $\lambda_B = f \cdot \lambda_{avg}$. Note that f only affects the ratio of the traffic rates between a busy route and a non-busy route; it does not affect the overall arrival rate of 0.4.

To evaluate the performances, recall that when more packets can be XOR'ed together at once, the higher is the coding gain. One can observe the operations of ICOP and COPE over a period of time and see how many original packets, on average, are XOR'ed into one coded packet, that is, $\frac{\text{Total number of original packets}}{\text{Total number of coded packets}}$. The higher this ratio means the better performance. Figure 6.6 shows the result of the simulation. One can see that ICOP allows more packets to be coded together as f increases. Under COPE,

however, this ratio is independent from the change of *f*. Since ICOP knows the route between nodes 1-4 and the route between nodes 2-5 have higher traffic rates, it will pick nodes 1-4 and 2-5 as the two coding pairs. On the other hand, COPE does not utilize the traffic rate information, and therefore the coding chances at the relay node depends heavily on the timing of each node's channel access, which can be random at times. This explains why ICOP performs better when the busy routes carry heavier traffic while the performance of COPE remains unchanged.

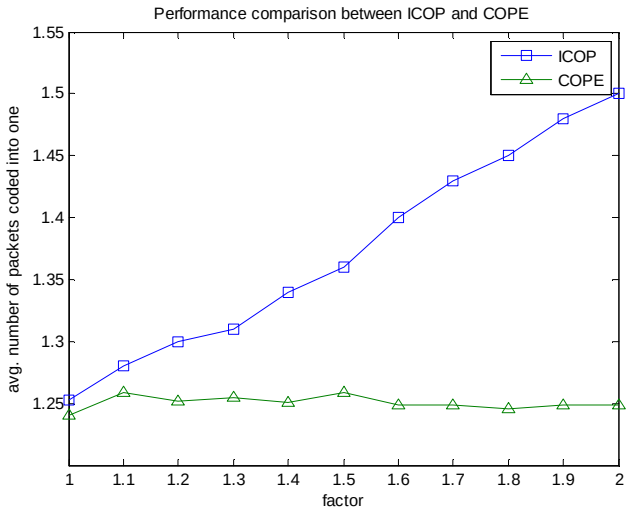


Figure 6.6 Performance Comparison, ICOP vs. COPE in the 6-node network.

6.5 Summary

This chapter has proposed a practical algorithm, ICOP, to find the network coding opportunity. The ICOP algorithm is able to look for the most effective network coding opportunities when both the topology and traffic pattern are known. The fact that ICOP takes into consideration of the traffic pattern gives its unique advantage over the plain COPE algorithm such that ICOP is able to offer high network coding gain without

complicated MAC layer scheduling and extra communication overhead. The simulations indicate that ICOP yields higher practical coding gains when there are certain known busy routes in the network. ICOP can recognize the busy routes and design the coding configurations that fit into these routes, and hence allow statistically more packets to be coded together. ICOP is particularly useful in wireless networks with slow-varying, non-evenly distributed traffic patterns, such as the wireless mesh networks. With fast-varying network conditions, ICOP can still maintain its advantage by recalculating the coding configurations more frequently.

CHAPTER 7

MAC LAYER SCHEDULING FOR WIRELESS NETWORK CODING: THEORY AND PRACTICE

7.1 Introduction

Network coding has gained wide interests in modern network researches. Network coding enables intermediate nodes along a path to encode and decode the packets as opposed to the traditional store-and-forward method. By doing so, it allows a multicast to achieve its maximum throughput. Li *et al* [32] demonstrated the concept of network coding using the butterfly network. Since then, it is also discovered that network coding can improve the throughput in wireless networks. COPE [34] was proposed to apply network coding to wireless networks by exploiting the broadcast nature of wireless communications. The fundamental concept of COPE is illustrated in Figure 7.1, the Alice-Bob topology. Alice has a packet to send to Bob, and Bob has a packet to send to Alice. Alice and Bob, however, are out of each other's transmission range, and therefore their communication paths have to go through the intermediate node (node R). In the traditional store-and-forward network, it would take four transmissions to deliver these two packets (one packet from each user to the other). However, COPE proposed that by using network coding, the node R can XOR both Alice and Bob's packets together and then broadcast it, so that the intended receivers will be able to decode this XOR'ed packet and extract the original information. In this case, it now only takes three transmissions to deliver these two packets. Since the proposal of COPE, many works have been done to address more open issues following COPE. These issues can be classified into the following three

categories: 1. Coding opportunity discovery: how do the nodes know which packets can be coded together and how to decode? 2. Code-aware routing: can the routing algorithm be improved so that it will create more network coding opportunities, given the same set of flows? 3. Code-aware MAC scheduling: How can the MAC layer be scheduled such that the packets are delivered in the right sequences so that they can be network-coded?

Many literatures have addressed the first issue (coding opportunity discovery). Generally, nodes discover the coding opportunity by collecting its neighborhood topology and routes information [34], [47], [55]. Le *et al.* [52] further explored the coding opportunity beyond the two-hop limitation in COPE. In terms of the second issue – code-aware routing – Le *et al.* [52] proposed a joint "coding+routing discovery" algorithm. Sengupta *et al.* [53] used linear programming to solve the multi-path code-aware routing problem. Wu *et al.* [54] proposed a new routing metric that takes the coding opportunity into consideration when calculating the routes: links with coding opportunities would be assigned reduced weight than those without. The focus of this chapter is on the third issue: MAC layer scheduling. The challenge in MAC layer scheduling is to find a practical, distributed scheduling scheme that improve the network coding performance as compared to the conventional, contention-based MAC protocol such as 802.11. In the original proposal of COPE, MAC layer scheduling was not particularly addressed; the nodes would simply code the packets opportunistically. That is, when an intermediate node gains access to the channel, it will perform network coding only if there is a matching packet in its buffer that can be coded with the head-of-the-line packet. Otherwise, it will send a native (uncoded) packet. With this scheme, many coding opportunities can be lost. For illustrative purposes, consider the Alice-Bob

topology. Assume that Alice has forwarded her packet (destined to Bob) to node R, while Bob still has not transmitted his packet. At this moment, node R has Alice's packet in its buffer, but not Bob's. If node R gains access to the channel, it will simply forward Alice's packet to Bob in the native form. However, one can see that, ideally, node R should "wait" for Bob to send his packet first so that node R can XOR these two packets. The conventional MAC scheduling is not aware of this, and therefore whether Bob gains access before node R is random. It is easy to design an ideal MAC scheduling for this simple case; however, it is not so obvious when the network is large and when there are many unicast sessions. This chapter proposes an algorithm that can find the theoretical optimal solution of MAC layer scheduling and its corresponding throughput given any network topology and routes. The optimal solution, nonetheless, is often not easy to be implemented practically with today's MAC protocols. Therefore, in the second part of this chapter, a distributed MAC layer scheduling scheme is proposed, namely, "Identifying the Coding Opportunity" (ICOP). ICOP improves the network coding performance (in terms of throughput) over the conventional, contention-based MAC layer scheduling.

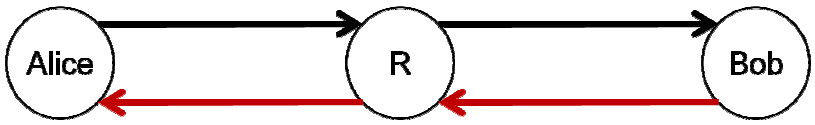


Figure 7.1 The Alice-Bob topology.

7.2 Related Works

The pioneer work by Li *et al.* [32] first showed that network coding can improve the throughput of a wired network for multicasts. It demonstrated that with network coding, the maximum throughput predicted by the max-flow-min-cut theorem can be achieved.

For wireless applications, Katti *et al.* [34] proposed a scheme, referred to as COPE, which utilizes network coding in wireless networks by exploiting the broadcast nature of wireless transmission. The concept of COPE is best illustrated in the Alice-Bob topology as shown in Figure 7.1, and is described in the previous section. Many works have followed COPE, addressing various issues on network coding in wireless networks [66]. These works can be roughly categorized into three areas: first, network coding opportunity discovery; second, network-coding-aware routing; third, network-coding-aware MAC layer scheduling. In the area of coding opportunity discovery, many works have addressed this issue from various aspects, such as topology and transmission power [34],[47],[64],[48],[68]. In [59] and [67], the impact of overhearing probability on COPE is studied. In the area of code-aware routing, the main concern is to route packets such that more coding opportunities will be created [53],[54],[60]. Since network coding has changed the traditional interference-avoidance-oriented routing, the traffic now can be routed “closer to” each other to create more coding opportunities. Sengupta *et al.* [53] used linear programming to find the theoretical optimal solutions on code-aware routing. Wu *et al.* [54] proposed a routing algorithm that assigns modified weights to each link based on its previous-hop and next-hop information. Zhang *et al* [58] proposed BEND, an opportunistic MAC layer mixing method without fixed routes. References [56] and [57] studied the impact of varying channel data rates on network coding. When a node

broadcasts a packet to its neighboring nodes, it can only use the worst data rate among all the intended recipients. These works study the trade-off between the gain of coding two packets into one and the loss of having to broadcast the coded packet at a lower data rate.

7.3 Finding the Optimal MAC Schedule: A Theoretical Solution

7.3.1 Problem Formulation

Given a wireless network represented in a directed graph $G(V, E)$, where V is the set of all nodes and E is the set off all directed edges. An edge $e_{i,j}$ belongs to E if and only if node j is within node i's transmission range. In other words, node i can transmit a packet directly to node j without any relay. Assume there is only unicast traffic, and let N be the total number of unicast sessions, each session with source node S_i and destination node D_i ($i=1,2,\ldots, N$).

Assumptions:

(1) The route for each unicast session is already determined. Further assume that only single path routing is allowed.

(2) All source nodes always have a packet to send.

(3) All packets have the same length.

(4) All channels have the same rate and are lossless, and it takes one unit time to transmit one packet.

Given the above conditions, the following two questions are considered:

(1) What is the optimal throughput that the network can achieve using network coding (with or without opportunistic listening)? Here, the throughput is defined as "the number of packets (from all flows) delivered successfully to the destination per unit time".

(2) What is the MAC layer schedule that yields the maximum throughput?

7.3.2 Tackling the Problem

Denote f_i,, $i=1,2,\ldots N$, as the unicast sessions in this network, and each f_i has a path $\wp_{f_i}$. Denote $H(f_i)$ as the hop count of flow f_i; then, $\wp_{f_i}$can be represented by an $H(f_i)$*-tuple*, that is, $\wp_{f_i} = \{e_{d,j1}, e_{j1,j2}, \ldots, e_{j(H(fi)-1),s}\}$, where d is the source node, s is the destination node, and $j_1, j_2,\ldots j_{(H(fi)-1)}$ are the intermediate nodes along the path. Each $e_{j,k} \in \wp_{f_i}$ is a "hop" in $\wp_{f_i}$. Note that it is possible that a link $e_{j,k}$ belongs to multiple $\wp_{f_i}$'s (because multiple flows can traverse a link). In order to differentiate the hops from different paths that share the same link, it is necessary to introduce a new notation, $h_{i,k}$, which denotes the k*-th* hop of flow f_i. For example, in Figure 7.2, $\wp_{f_3} = \{e_{7,5}, e_{5,4}, e_{4,3}\}$, and so $h_{3,2}=e_{5,4}$.

Because of wireless transmissions, links that are within the interference ranges of each other should not be activated simultaneously. For the same reason, "hops" that are within the interference ranges of each other should not be activated simultaneously either. This means one can construct a conflict graph from the hops-basis. From the conflict graph, all the possible independent sets $I_l=\{h_{i,j}: h_{i,j}$ can be activated simultaneously$\}$can be obtained. Note that I_l here are not limited to only the maximal independent sets[4]; any set of hop(s) that can be activated simultaneously is a legitimate I_l, including those of a single hop. Let L be the total number of independent sets. Without network coding, I_l must be composed of hops that are out of interference range of each other. However, with network coding, a broadcast of a coded packet can be viewed as a "simultaneous activation" of the hops that each of the original (native) packets would have traversed through. As a result, two hops that were used to be conflicted with each other due to

[4] A maximal independent set is one that becomes a non-independent set if any additional vertex is included.

interference may no longer interfere with each other because of network coding. In other words, network coding can erase some edges in the conflict graph, and therefore increase the total number of independent sets.

These independent sets, I_l (l=1,2,.., L), are very important in the algorithm because a MAC layer schedule is essentially to determine when to activate which one of these independent sets. Let $A(I_l)$ ($0 \leq A(I_l) \leq 1$) be the fraction of time that this independent set is activated (i.e., all hops in this set is activated simultaneously), then $\sum_{l=1}^{L} A(I_l) \leq 1$. Let $A(h_{i,j})$ be the fraction of time that hop $h_{i,j}$ is activated, and then $A(h_{i,j}) = \sum_{l:h_{i,j} \in I_l} A(I_l)$. Now, the MAC layer scheduling problem can be reduced into an optimization problem:

$max\ \lambda = \sum_{i=1}^{N} A(h_{i,1})$

subject to

$$\sum_{l=1}^{L} A(I_l) \leq 1 \tag{7.1}$$

$$A(h_{i,1}) = A(h_{m,1})\ \forall i = 1,2,\dots,N;\ m = 1,2,\dots,N; i \neq m \text{ (fairness)} \tag{7.2}$$

$$\forall i = 1,2,\dots,N, A(h_{i,j}) = A(h_{i,m})\ (j,m = 1,2,\dots,H(f_i); j \neq m) \tag{7.3}$$

$$A(I_l) \geq 0,\ \forall l = 1,2,\dots,L \tag{7.4}$$

Constraint (7.1) states that the sum of the fractions of time that each independent set is activated cannot exceed 1. Constraint (7.2) ensures the fairness of each flow, such that each flow experiences the same packet rate. In the situation where some flows require more bandwidth than others, constraint (7.2) will be modified and possibly expanded to multiple constraints, as will be shown in one of the examples. Constraint

(7.3) is the information conservation law that simply dictates that the packet rates of all the hops in the same path must be the same.

The inputs of the function λ are $A(h_{i,1}), \forall i = 1,2,\dots,N$; however, it is the values of $A(I_l)$ that are of interest. $A(h_{i,1})$ can therefore, be substituted with $A(I_l)$ using the relation $A(h_{i,j}) = \sum_{l:h_{i,j}\in I_l} A(I_l)$.

The solution gives the values of $A(I_l)$, $l = 1,2,\dots,L$, indicating how much fraction of time each independent set should be active in order to achieve the optimal throughput. The value of function λ at this optimal point is the ideal throughput. The next step is to convert this solution to the actual schedule.

7.3.3 Actual Scheduling

Lemma 1: With the same set of unicasts, the optimal MAC layer schedule is periodical.

Proof: The exact solution to a linear programming problem is always a rational number as long as the coefficients of the function and constrains are rational [62]. Therefore, $A(I_l)$ is always a rational number for all l. Let p_l/q_l represent the fractional expression of $A(I_l)$, and let τ be the least common multiple (LCM) of q_l, and then the schedule can be sufficiently described in τ steps (time slots); the schedule repeats itself after τ steps. Therefore, the optimal MAC layer schedule is periodical. ■

From Lemma 1, one can conclude that values of $A(I_l)$ can be expressed in the fractional form p_l/q_l, and their lowest common denominator τ is the period of the schedule. Rewrite $A(I_l) = \rho_l/\tau$, where $\rho_l/\tau = p_l/q$. In order to satisfy the solutions of $A(I_l)$, the optimal schedule should consist of τ steps, among which ρ_l steps must be dedicated to activate all the hops in the independent set I_l. However, the next question is

how these τ steps should be put in order. It turns out, for the sole purpose of maximizing the throughput, any permutation of these τ steps will do. However, different permutations do impact some other performance factors such as packet jitter and the number of packets that need to be buffered in the intermediate nodes. Finding the optimal permutation based on minimizing jitter or minimizing buffers is out of the scope of this dissertation, and it will be addressed in the future work.

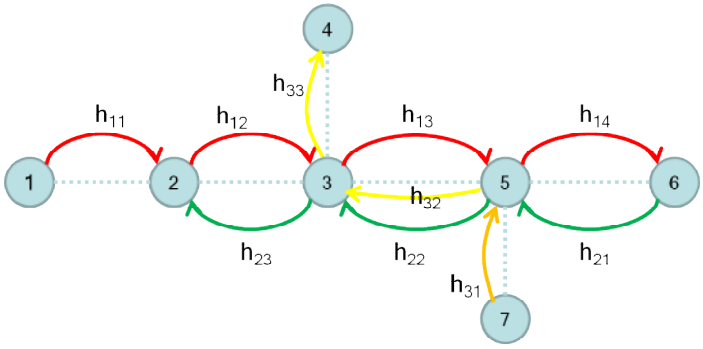


Figure 7.2 Seven-nodes three-flows example

7.3.4 Example

This section presents an example to illustrate the proposed algorithm. Refer to the network shown in Figure 7.2, which consists of seven nodes and three unicast flows, f_1 , f_2 ,and f_3 . The paths of the flows are $\wp_{f_1} = \{e_{1,2}, e_{2,3}, e_{3,5}, e_{5,6}\}$, $\wp_{f_2} = \{e_{6,5}, e_{5,3}, e_{3,2}\}$, and $\wp_{f_3} = \{e_{7,5}, e_{5,3}, e_{3,4}\}$. These edges are then mapped into hops: $h_{1,1} = e_{1,2}$, $h_{1,2} = e_{2,3}$, and so on. Assuming the interference range is two-hop, then, fifteen independent sets can be found: $I_1 = \{h_{1,1}, h_{1,4}\}$, $I_2 = \{h_{1,1}, h_{2,1}\}$, $I_3 = \{h_{1,1}, h_{3,1}\}$, $I_4 = \{h_{1,3}, h_{2,3}\}$, $I_5 = \{h_{1,4}, h_{2,2}\}$, $I_6 = \{h_{1,1}\}$, $I_7 = \{h_{1,2}\}$, $I_8 = \{h_{1,3}\}$, $I_9 = \{h_{1,4}\}$,

$I_{10} = \{h_{2,1}\}$, $I_{11} = \{h_{2,2}\}$, $I_{12} = \{h_{2,3}\}$, $I_{13} = \{h_{3,1}\}$, n$I_{14} = \{h_{3,2}\}$, and $I_{15} = \{h_{3,3}\}$. One can observe that I_4 and I_5 are made possible only because of network coding.

The solution of the optimization is: $A(I_2) = A(I_3) = A(I_{10}) = A(I_{13}) = 1/14$, $A(I_4) = A(I_5) = A(I_7) = A(I_{14}) = A(I_{15}) = 2/14$, and zero activation time for the remaining independent sets. The maximum throughput is $\lambda = 3/7$. From the solution, one can conclude that the optimal schedule has a period of 14 steps (time slots), within these 14 steps each source node injects two packets into the network, and each flow receives two packets at its destination.

Table 7.1 Optimal Schedule for the Network in Figure 7.3

Step	Activated hops
1	$h_{1,1}, h_{2,1}$
2	$h_{1,2}$
3	$h_{1,3}, h_{2,3}$
4	$h_{1,4}, h_{2,2}$
5	$h_{1,1}, h_{3,1}$
6	$h_{3,2}$
7	$h_{3,3}$
8	$h_{1,2}$
9	$h_{2,1}$
10	$h_{1,3}, h_{2,3}$
11	$h_{1,4}, h_{2,2}$
12	$h_{3,1}$
13	$h_{3,2}$
14	$h_{3,3}$

Table 7.1 is one possible optimal schedule. Note that Steps 3 and 4 represent the steps where network coding is performed at node 3 and 4, respectively, and so are Steps 10 and 11.

7.4 A Practical Scheme: Identifying the Coding Opportunity

Finding the optimal solution is NP-hard as it involves with finding all the independent sets. Furthermore, even after the optimal solution is obtained, it is still difficult to implement the solution in the actual networks because a centralized coordination for all nodes is required. Therefore, a practical, distributed MAC layer scheduling scheme is necessary. The conventional MAC protocol (such as 802.11), though easy to implement, is not suitable for network coding. In this section, a new distributed MAC layer scheme -- ICOP – is proposed that will asymptotically improve the throughput of a wireless network with network coding comparing to those using the conventional MAC.

7.4.1 The Shortcoming of Conventional MAC

In the conventional MAC protocol such as 802.11, nodes compete against each other to access the wireless medium. Which nodes get the access at which time is essentially determined by chance. The potential benefit of network coding may not be realized with this type of MAC scheme. Take a simple example from the Alice-Bob topology. Alice has a packet to send to Bob while Bob has a packet to send to Alice. In the beginning, either Alice or Bob would have a 50% chance to gain access to the channel (since they are the only two nodes with packets to send). Assume that Alice won the contention and sent her packet to the relay node. Now at the second time slot, it is the relay node and

Bob who are competing for the channel. If Bob wins the contention and send his packet to the relay node, then the relay node can network-code these two packets together and broadcast the coded packet to Alice and Bob when it gains the access. However, if it is the relay node that gains the access before Bob, then the relay node would simply forward the packet it had received from Alice to Bob. Obviously in such case, a potential coding opportunity is missed. Such missed coding opportunities mar the benefits of network coding, and therefore should be avoided.

7.4.2 Identifying the Coding Opportunity (ICOP)

This section describes the proposed scheme "Identifying the Coding Opportunity" (ICOP). ICOP is designed to reduce the occurrence of missing coding opportunities as described above. The main idea of ICOP is to utilize the ACK packet sent by an intermediate node to inform its neighboring nodes about the existence of a coding opportunity, and gives the node(s) with the matching packet with a higher priority to access the channel in the next time frame, assuming that all nodes are aware of network coding, and have the knowledge of which packets could be coded together (such knowledge can be obtained from the neighborhood information). When an intermediate node receives a packet that can be coded with another packet that is already stored in its buffer, then the intermediate node will send a regular ACK. However, when an intermediate node receives a packet that could be coded, but it does not have a matching packet in its buffer, then the node will send a modified ACK that informs its surrounding nodes of such coding opportunity. The modified ACK contains the addresses of its previous hop and the next hop, and its "type" field indicates that this is a modified ACK

[61]. The surrounding nodes that overheard this ACK packet can use these two addresses to determine whether itself has a matching packet in its local buffer. The surrounding node that has the matching packet will then set its back off timer shorter than a threshold *t*, while other nodes set their back off timers larger than *t*. When the node with the matching packet gains access to the channel, it will send the matching packet from its buffer, even if this packet is not at the head of line. This gives the node with matching packet higher priority to deliver the matching packet to the intermediate node, thus reducing the occurrence of missing coding opportunities. The only differences between ICOP and the conventional MAC are the modified ACK packet and the setting of back off timer. The simulation indicates that this slight modification produces significant improvement, as will be shown in the next section.

7.5 Simulations Results

In this section provides the simulation results of ICOP, and compare them with those of the conventional MAC scheme. The simulations in this section are carried out on a stand-alone discrete-time simulator built for this dissertation using Matlab. The wireless channels are lossless and each node is equipped with a tail drop buffer in the size of 100 packets. The simulation begins with the basic Alice-Bob topology with two flows, and compare the throughputs achieved by ICOP and by conventional MAC. The throughput is calculated by $\frac{number\ of\ packets\ arrived}{total\ unit\ time}$. Unless otherwise specified, the number of packets arrived include those from all flows. The simulation is run with various packet rates at the source node. At the first run, each of the source nodes (Alice and Bob) generates packets at the rate of 1/5 (one packet per five unit times, where one unit time is

defined as the time needed to transmit one packet). Once the packet is generated, it is temporarily stored in a FIFO buffer waiting to be transmitted. The source packet rate is increased with each run until it reaches one. The rate of one means the source nodes always have packets to send. Figure 7.3 shows the throughput comparison between ICOP and the conventional MAC. Obviously, the ideal throughput in this topology is 2/3, which means each flow delivers one packet in every three unit time, and thus the ideal packet generation rate is 1/3. This is reflected in Figure 7.3, as the throughput stays at the same level (~2/3) once the packet generation rate reaches 1/3. Interestingly, there does not seem much difference between the throughputs achieved by ICOP and by conventional MAC. When further comparing the fairness of each flow (i.e., the throughput per flow), ICOP and conventional MAC yield similar fairness. However, when the simulation is conducted using the network in Figure 7.2, ICOP shows noticeable advantage over the conventional MAC.

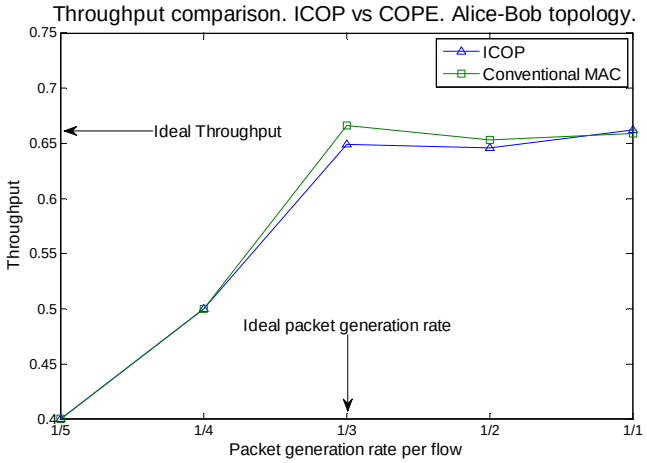


Figure 7.3 Throughput comparison: ICOP vs. conventional MAC; Alice-Bob topology.

The packet generation rate starts at 1/10, and gradually increases to 1. Figure 7.5 shows the throughput comparison between ICOP and conventional MAC using the

seven-node, three-flow network as shown in Figure 7.2. From the optimization algorithm presented in Section 7.3, it is known that the ideal throughput is 3/7, which means each flow delivers one packet in every seven unit time, and thus the ideal packet generation rate is 1/7. In this figure, one can see that when the packet generation rate is below 1/7, both ICOP and the conventional MAC perform the same since the network is not heavily loaded. However, when the packet generation rate exceeds the ideal 1/7, the throughput degrades dramatically with the conventional MAC, while the throughput of ICOP does not decrease as much even when the packet generation rate reaches one. In terms of fairness, ICOP and the conventional MAC both yield similar performances. (Refer to Figure 7.5 and Figure 7.6). Finally, the frequencies of coding with ICOP and the conventional MAC are compared in Figure 7.6. Generally, the more frequent the network coding occurs, the fewer transmissions are required to deliver the same amount of packets. The frequency of coding is measured as $\frac{\text{total number of coded packets}}{total\ unit\ time}$. As one can see, when the packet generation rate exceeds the ideal level, the number of packets being coded with ICOP is far more than those with the conventional MAC.

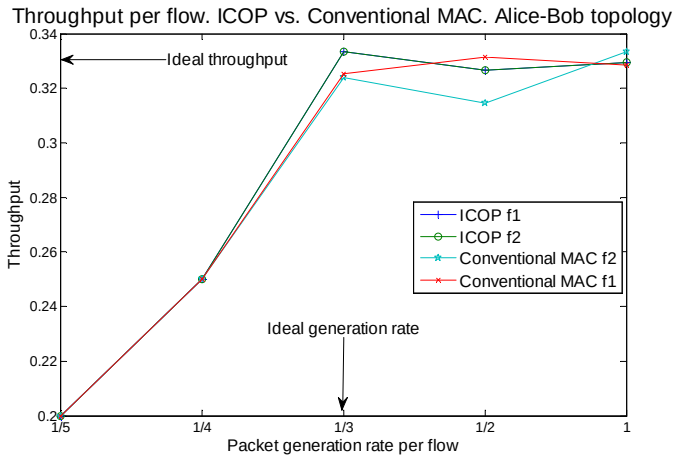


Figure 7.4 Per-flow throughput: ICOP and conventional MAC.

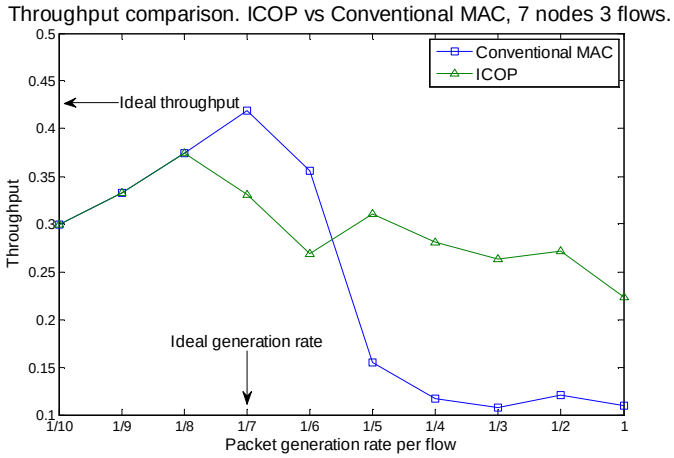


Figure 7.5 Throughput comparison: ICOP vs. conventional MAC; 7 nodes and 3 flows.

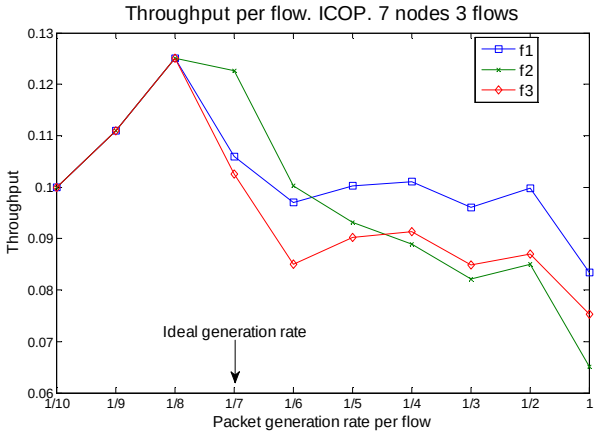


Figure 7.6 Per-flow throughput. ICOP.

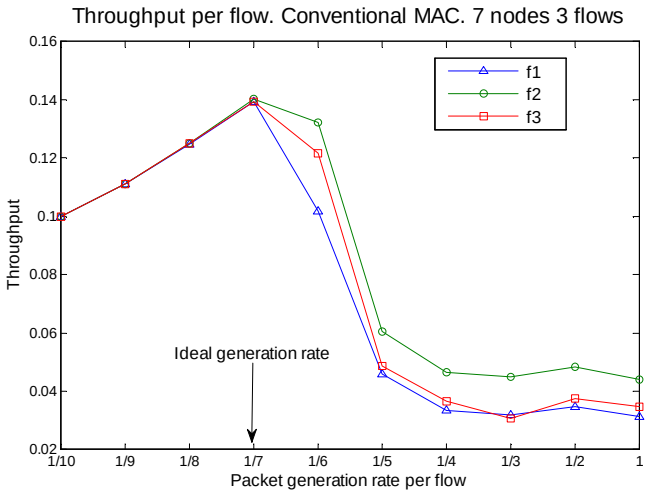


Figure 7.7 Per-flow throughput. Conventional MAC.

From the results of the 7-node network, one can see that when the packet generation rates at the source nodes are below the ideal level, the conventional MAC and ICOP perform equally well in terms of throughput. This is because the network is lightly-loaded, and therefore can handle all the traffic easily. However, if the packet arrival rates at the source nodes exceed the ideal throughput, then the throughput of the conventional MAC degrades significantly while ICOP's performance remains steady despite of the increased packet arrival rates. These results imply that ICOP can be particularly advantageous over the conventional MAC in networks where the upper layer flow control is not perfect; this is often the case with today's flow control mechanism in TCP [65].

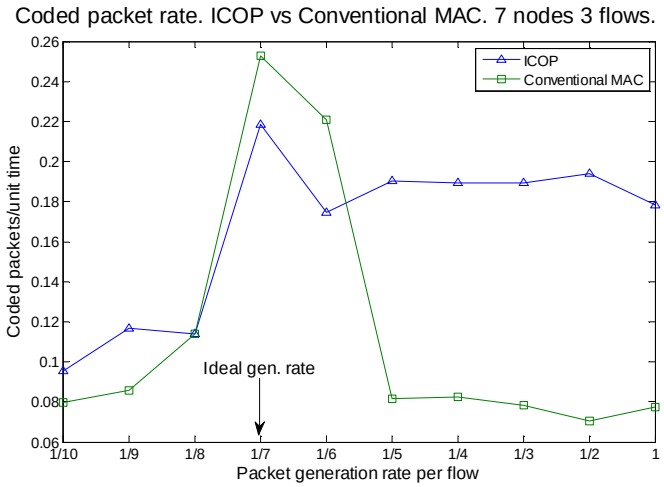


Figure 7.8 Coding occurrence comparison.

7.6 Summary

In wireless network coding, the MAC layer scheduling can affect the actual throughput gain. This chapter has proposed an algorithm that finds the optimal MAC layer schedule and the optimal throughput. Even though finding the optimal schedule is NP-hard, the proposed algorithm provides an upper bound of the achievable throughput. This chapter has also proposed a distributed MAC layer scheduling scheme, ICOP, which utilizes a modified ACK packet to inform surrounding nodes of the network coding opportunity. The simulations show that when the network is heavily loaded without perfect flow control, ICOP can achieve better throughput than the conventional MAC.

CHAPTER 8

CONCLUSIONS AND FUTURE WORKS

Most of the problems and challenges faced by today's networks can be reduced to a common root: how packets shall be delivered. The works presented in this dissertation have contributed in subjects including link state updates in the area of QoS routing, network coding in peer-to-peer file sharing, and network coding in wireless networks. With the growing popularity of wireless networks, network coding in wireless networks has drawn much attention. The ICOP proposed in Chapter 6 is a first step towards finding a practical solution to implement network coding in wireless networks. One of the challenges of wireless network coding is MAC layer scheduling. In order to increase the number of coded packets (hence increase the network coding gain), MAC layer has to deliver the right packet at the right time so the matching packets can be coded together at the relay node. The ideal MAC layer schedule can be found by using the algorithm described in Chapter 7. However, to practically implement the ideal schedule is difficult. This dissertation therefore, proposed a practical network-coding-aware MAC layer scheduling scheme that will better support the network coding algorithm as compared to the current MAC layer protocol such as 802.11.

In summary, the contributions of this dissertation are:

(1) Link state update: proposed ROSE, a class-based link state update scheme that utilizes the statistical information of user's QoS requests and the links' actual QoS states. ROSE takes this statistical information into account to design the class boundaries to minimize the cost of false routing.

(2) Peer-to-peer file sharing networks: proposed DRLNC, a network coding scheme that shifts the coding decisions from the uploaders to the downloaders. DRLNC eliminates the collision problem without increasing the field size.

(3) Wireless network coding: this dissertation has addressed the MAC layer scheduling problem on both theoretical and practical grounds. In the theoretical work, it has proposed an algorithm to solve the MAC layer scheduling problem. In the practical work, it has proposed ICOP, a MAC layer protocol that utilizes the ACK packets to deliver extra network-coding-related information so that matching packets can have better chances to be coded at intermediate nodes.

The future works will be focused on MAC layer scheduling for wireless network coding, which includes upgrading ICOP to accommodate variable channel rates and handle packet losses.

REFERENCES

[1] S. Chen and K. Nahrstedt, "An over view of quality of service routing for next-generation high-speed network: problems and solutions," *IEEE Network*, vol. 12, no. 6, pp. 64-79, Nov. 1998.

[2] J. Moy, "OSPF version 2," RFC2328, IETF, 1998.

[3] G. Apostolopoulos, R. Guerin, S Kamat, and S. Tripathi, "Quality-of-service based routing: A performance perspective," *Proceedings of ACM SIGCOMM 1998*, Vancouver, vol. 28, pp.17-28.

[4] A. Shaikh, J. Rexford, and K. G. Shin, "Evaluating the impact of stale link state on quality-of-service routing," *IEEE/ACM Transactions on Networking*, vol. 9, no. 2, pp.162-176, Apr. 2001.

[5] Q. Ma and P. Steenkiste, "Quality-of-service routing for traffic with performance guarantees," *Proceedings of IFIP Int. Workshop Quality of Service 1997*, pp. 115-126.

[6] L. Breslau, D. Estrin, and L. Zhang, "A simulation study of adaptive source routing in integrated service networks," Computer Science Department, University of Southern California, Tech, Rep. 93-551, 1993.

[7] G. Cheng and N. Ansari, "ROSE: a novel link state information update scheme for QoS routing," *Proceedings 2005 Workshop on High Performance Switching and Routing (HPSR 2005)*, pp. 24 -28, May 12-14, 2005.

[8] R. Braden, D. Clark, and S. Shenker, "Integrated Services in the Internet Architecture: an Overview," RFC1633, June 1994.

[9] S. Blake, D. Black, M. Calson, E. Davies, Z. Wang, and W. Weiss, "An Architecture for Differentiated Services," RFC2475, Dec. 1998.

[10] Y. Bernet, P. Ford, R. Yavatkar, F. Baker, L. Zhang, M. Speer, R. Braden, B. Davie, J. Wroclawski, and E. Felstaine, "A framework for integrated services operation over DiffServ networks," RFC 2998, Nov. 2000.

[11] J. Yang, J. Ye, S. Papavassiliou, and N. Ansari, "A Flexible and Distributed Architecture for Adaptive End-to-End QoS Provisioning in Next-Generation Networks," *IEEE Journal on Selected Areas in Communications*, vol. 23, no. 2, pp. 321-333, Feb. 2004.

[12] S. Floyd and V. Jacobson, "Random early detection for congestion avoidance," *IEEE/ACM Trans on Networking*, vol. 1, no. 4, pp. 397–413, Jul. 1993.

[13] G. Cheng and N. Ansari, "On Multiple Additively Constraint Path Selection," *IEE Proceedings on Communications*, vol.149, no. 5, pp. 237-241, Oct. 2002.

[14] R. Guerin and A. Orda, "QoS based routing in networks with inaccurate information: theory and algorithms," *Proc. IEEE INFOCOM'97*, Kobe, Japan, Apr. 7-11, 1997, vol. 1, pp. 75-83.

[15] T. Korkmaz, M. Krunz, and S. Tragoudas, "An efficient algorithm for finding a path subject to two additive constraints," *Proc. ACM SIGMETRICS'2000*, Santa Clara, California, June 18 - 21, 2000, pp. 318-327.

[16] L. Gang and K. G. Ramakrishnan, "A*prune: an algorithm for finding k shortest paths subject to multiple constraints," *Proc. IEEE INFOCOM'2001*, Anchorage, Alaska, Apr. 22-26, 2001, vol. 2, pp. 743-749.

[17] S. Chen and K. Nahrstedt, "Distributed QoS routing with imprecise state information," *Proc. 7th Intl. Conf. Computer Communications and Networks*, Lafayette, Louisiana, Oct. 12-15, 1998, pp. 614-621.

[18] Z. Wang and J. Crowcroft, "Quality of Service routing for supporting multimedia applications," *IEEE Journal on Selected Areas on Communications*, vol. 14, no. 7, pp. 1228-1234, Sep. 1996.

[19] S. Sengupta, D. Saha, and S. Chaudhuri , "Analysis of enhanced OSPF for routing lightpath in optical mesh networks," *Proc. IEEE ICC'02*, vol. 5, pp. 2865-2869, 2002.

[20] G. Apostolopoulos, R. Guerin, S Kamat, and S. Tripathi, "Quality-of- service based routing: A performance perspective," *Proc. ACM SIGCOMM 1998*, Vancouver, British Columbia, Canada, *Aug. 31 – Sep. 04, 1998*, vol. 28, pp.17-28.

[21] M. Peyravian and R. Onvural, "Algorithm for efficient generation of link-state updates in ATM networks," *Computer Networks and ISDN Systems*, vol. 29, pp. 237-247, 1997.

[22] X. Li, L. K. Shan, W. Jun, and N. Klara, "QoS Extension to BGP," *Proc. IEEE Intl. Conf. Network Protocols (ICNP'02)*, Paris, France, Nov. 12-15, 2002, pp. 100-109.

[23] N. Wang, G. Cheng, and N. Ansari, "ROSE II: The Case of Additive Metrics for Updating Additive Link State Information," *Proc. ICC 2006*, Istanbul, Turkey, May 11-15, 2006, pp. 676-680.

[24] J. Ding, M. Kirkpatrick, and E.H.-M. Sha, "QoS measures and implementations based on various models for real-time communications," *Application-Specific Systems and Software Engineering Technology, 2000. Proceedings. 3rd IEEE Symposium on,* 24-25 March 2000, pp.125 – 129.

[25] L. Angrisani, G. Ventre, L. Peluso, and A. Tedesco, "Measurement of processing and queuing delays introduced by an open-source router in a single-hop network," *IEEE Transactions on Instrumentation and Measurement*, vol. 55, no. 4, pp. 1065 – 1076, Aug. 2006.

[26] G. Appenzeller, I. Keslassy, and N. McKeown, "Sizing Router Buffers," *Computer Communication Review*, vol. 34, no. 4, pp. 281-292, 2004.

[27] M. Garetto, D.R. Figueiredo, R. Gaeta, and M. Sereno, "A modeling framework to understand the tussle between ISPs and peer-to-peer file-sharing users," *Performance Evaluation*, vol. 64, no. 9-12, pp. 819-837, October, 2007.

[28] W. Sang and D. Qiu, "On the efficiency of peer-to-peer file sharing," *IEEE International Conference on Multimedia and Expo, 2007,* pp.32 – 35, July 2007.

[29] J. Zhang, L. Cheng, and X. Zhu, "An improved strategy of piece selection in P2P," *Control and Decision Conference, 2009. CCDC '09*, pp. 4457 – 4460, June 2009.

[30] D. Stutzbach and R. Rejaie, "Understanding churn in peer-to-peer networks." *Proceedings of the 6th ACM SIGCOMM conference on Internet measurement, IMC*, Oct. 2006, pp. 189-201.

[31] C. Gkantsidis and P. Rodriguez, "Network coding for large scale content distribution," *IEEE Infocom*, Miami, FL, 2005, vol. 4, pp. 2235-2245.

[32] R. Ahlswede, N. Cai, S. R. Li, and R. W. Yeung, "Network information flow," *IEEE Transactions on Information Theory*, vol. 46, no. 4, pp. 1204-1216, Jul. 2000.

[33] S. Li, R. Yeung, and N. Cai, "Linear network coding," *IEEE Transactions on Information Theory*, vol. 49, no. 2, pp. 371-381, Feb. 2003.

[34] S. Katti, H. Rahul, W. Hu, D. Katabi, M. Médard, and J. Crowcroft, "XORs in the air: practical wireless network coding," *IEEE/ACM Transactions on Networking*, vol. 16, No. 3, pp. 497-510, March. 2008.

[35] C. Fragouli, D. Katabi, A. Markopoulou, M. Medard, and H. Rahul, "Wireless network coding: opportunities & challenges," *2007 IEEE Military Communications Conference (MILCOM 2007)*, pp. 1 – 8.

[36] http://research.microsoft.com/camsys/avalanche/

[37] D. M. Chiu, R.W Yeung, J. Huang, and B. Fan, "Can network coding help in P2P networks?" *2006 4th International Symposium on Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks,* Apr. 2006, pp.1 – 5.

[38] D. Qiu, and R. Srikant, "Modeling and performance analysis of BitTorrent-like peer-to-peer networks," *Proceedings of the 2004 conference on Applications, technologies, architectures, and protocols for computer communications, ACM Sigcomm*, pp. 367-378.

[39] C. Gkantsidis, J. Miller, and P. Rodriguez, "Anatomy of a P2P content distribution system with network coding," *IPTPS'06,* Santa Barbara, CA, February 2006.

[40] C. Gkantsidis, J. Miller, and P. Rodriguez, "Comprehensive view of a live network coding P2P system," *Proceedings of the 6th ACM SIGCOMM conference on Internet measurement*, Oct 2006, pp. 177-188.

[41] S. Deb and M. Médard, "Algebraic gossip: A network coding approach to optimal multiple rumor mongering," *Proceedings of the 42nd Annual Allerton Conference on Communication, Control, and Computing,* Sep. 2004.

[42] S. Deb, M. Médard, and C, Choute, "On random network coding based information dissemination," *Proceedings, International Symposiums on Information Theory 2005, ISIT 2005.* Sept. 2005. pp. 278 - 282

[43] S. Deb, M. Médard, and C, Choute, "Algebraic gossip: a network coding approach to optimal multiple rumor mongering," *IEEE Transactions on Information Theory,*, vol. 52, no. 6, pp. 2486-2507, 2006.

[44] D. Mosk-Aoyama and D. Shah, "Information dissemination via network coding." *IEEE International Symposium on information Theory - Proceedings*, art. no. 4036267, pp. 1748-1752, 2006.

[45] A. Legout, G. Urvoy-Keller, and P. Michiardi, "Rarest first and choke algorithms are enough," *Proceedings of the 6th ACM SIGCOMM Internet Measurement Conference, IMC*, Oct. 2006, pp. 203-216.

[46] J. Fonseca, B. Reza, and L. Fjeldsted. BitTorrent protocol - BTP/1.0 (2005) http://jonas.nitro.dk/bittorrent/bittorrent-rfc.html.

[47] S. Omiwade, R. Zheng, and C. Hua, "Butterflies in the Mesh: Lightweight Localized Wireless Network Coding," *Proceedings - 2008 4th Workshop on Network Coding, Theory, and Applications, NetCod 2008*, Hong Kong, Jan. 2008, pp. 1-6.

[48] Q. Dong, J. Wu, W. Hu, and J. Crowcroft, “Practical Network Coding in Wireless Networks,” *Proceedings of the 13th annual ACM international conference on Mobile computing and networking, MobiCom 2007*, Montréal, Sep. 2007, pp. 306-309.

[49] J. Zhang, Y.P. Chen, and I. Marsic, “Network Coding Via Opportunistic Forwarding in Wireless Mesh Networks,” 2008 IEEE Wireless Communications and Networking Conference, Las Vegas, Apr. 2008, pp. 1775-1780.

[50] N. Cai and R. W. Yeung, “Secure network coding on a wiretap network,” *IEEE Transactions on Information Theory*, vol. 57, no. 1, pp. 424-435, 2011.

[51] N. Cai and T. Chan, “Theory of secure network coding,” *Proceedings of the IEEE*,vol. 99, no. 3, pp. 421-437, 2011.

[52] J. Le, J. C.S. Lui, and D. Chiu, “DCAR: Distributed Coding-Aware Routing in wireless networks,” *IEEE Transactions on Mobile Computing*, vol. 9, no. 4, pp. 596-608. Aug. 2009.

[53] S. Sengupta, S. Rayanchu, and S. Banerjee, “An analysis of wireless network coding for unicast sessions: The case for coding-aware routing,” in *Proc. INFOCOM*. Anchorage, AK: IEEE, May 2007, pp.1028-1036.

[54] Y. Wu, S. M. Das, and R. Chandra, “Routing with a markovian metric to promote local mixing,” *Proc. IEEE INFOCOM 2007 Minisymposium*. Anchorage, Alaska: IEEE, May 2007, pp. 2381-2385.

[55] Y. Wu, J. Padhye, R. Chandra, V. Padmanabhan, and P. A. Chou, “The local mixing problem,” *Proc. Information Theory and Applications Workshop*, San Diego, CA: Univ. of California, San Diego, Feb. 2006.

[56] P. Chaporkar and A. Proutiere, “Adaptive network coding and scheduling for maximizing throughput in wireless networks,” *Proceedings of the 13th annual ACM international conference on Mobile computing and networking*, MobiCom '07, September 09-14, 2007, Montréal, Québec, Canada, pp. 135-146.

[57] H. Yomo and P. Popovski, “Opportunistic scheduling for wireless network coding,” *IEEE International Conference on Communications, ICC '07*. Glasgow, Jun. 2007, pp. 5610-5615.

[58] J. Zhang, Y. P. Chen, and I. Marsic, “MAC-layer proactive mixing for network coding in multi-hop wireless networks,” *The International Journal of Computer and Telecommunications Networking*, vol. 54, no. 2, pp. 196-207. Feb. 2010.

[59] K. Chi, X. Jiang and S. Horiguchi, "Network coding opportunity analysis of COPE in multihop wireless networks," *IEEE Wireless Communications and Networking Conference, WCNC 2008*. Las Vegas, Apr. 2008, pp. 2858 – 2863.

[60] X. Jiao, X. Wang, and X. Zhou, "Active Network Coding based High-Throughput Optimizing Routing for Wireless Ad Hoc Networks," *Wireless Communications, Networking and Mobile Computing, 2008. WiCOM '08. 4th International Conference on*, Dalian, Nov. 2008, pp. 1-5.

[61] IEEE 208.11 standard. 2007.

[62] D. L. Applegate, W. Cook, S. Dash, and D.G. Espinoza, "Exact solutions to linear programming problems," *Operation Research Letters*, vol. 35, no. 6, pp. 693-699, Nov. 2007.

[63] A. Khreishah, C. Wang, and N. Shroff, "Cross-layer optimization for wireless multihop networks with pairwise intersession network coding," *IEEE Journal on Selected Areas in Communications*, vol. 27, no. 5, pp. 606-621, June 2009.

[64] N. Wang and N. Ansari, "Identifying the network coding opportunity," *2010 IEEE Sarnoff Symposium*, Princeton, Apr. 2010, pp. 1-5.

[65] H. Roh and J. Lee, "Network coding-aware flow control in wireless ad-hoc networks," *IEEE Wireless Communications and Networking Conference, WCNC 2009.* Budapest, Apr. 2009, pp. 1-6.

[66] Y. Wu, "Network coding for wireless networks," Microsoft Research Technical Report MSR-TR-2007-90, 2007.

[67] Y. Kim and G. Veciana, "Is rate adaptation beneficial for inter-session network coding?" *IEEE Journal on Selected Areas in Communications,* vol. 27, no. 5, June, 2009.

[68] T. Cui, L. Chen, and T. Ho, "Energy efficient opportunistic network coding for wireless networks," *IEEE INFORCOM 2008. The 27th Conference on Computer Communications*. Phoenix, Apr, 2008, pp. 361-365.