


國立交通大學

網路工程研究所

碩士論文

使用網路編碼技術之機會型非同  
步資訊傳播演算法



Opportunistic Asynchronous  
Information Dissemination - A  
Forwarding Algorithm Using Network  
Coding

研究生：王湘博

指導教授：曾煜棋 教授、易志偉 教授

中華民國九十八年六月

使用網路編碼技術之機會型非同步資訊傳播演算法  
Opportunistic Asynchronous Information Dissemination - A Forwarding  
Algorithm Using Network Coding

研究生：王湘博

Student：Hsiang-Po Wang

指導教授：曾煜棋、易志偉

Advisor：Yu-Chee Tseng、

Chih-Wei Yi

國立交通大學  
網路工程研究所  
碩士論文



A Thesis  
Submitted to Institute of Network Engineering  
College of Computer Science  
National Chiao Tung University  
in partial Fulfillment of the Requirements  
for the Degree of  
Master  
in  
Computer Science

June 2009

Hsinchu, Taiwan, Republic of China

中華民國九十八年六月

# 使用網路編碼技術之機會型非同步資訊傳播演算法

學生：王湘博


指導教授：曾煜棋 教授

易志偉 教授

國立交通大學

網路工程研究所

## 摘要



COPE 是由 Katti 等人提出的一種分散式、區域性、嘗錯式網路編碼演算法。在這篇論文裡我們提出一種新的通用性轉送演算法，此演算法適用於無線網路環境。我們稱這種演算法叫 *Opportunistic Asynchronous Information Dissemination*，我們用英文縮寫 OASIS 表示。在 OASIS 裡面，我們繼承了 COPE 的兩項特性，*opportunistic listening* 以及 *opportunistic coding*，並且，我們更進一步發展新的技術，我們稱為 *opportunistic information dissemination*。在這個新技術下，我們嘗試混入更多的原始封包到一個編碼封包裡，即使這些混入的封包不是他們 1-hop neighbor 所需要的。當這些 1-hop neighbor 收到這些封包時，他們會將這些封包放入他們的資訊儲存槽，這些封包可以提供更多資訊，以利未來的編碼及解碼的用途。除此之外，由於資訊儲存槽有大小的限制，我們觀察到有效管理資訊儲存槽可以更進一步的提升網路的吞吐量。除了仿效 COPE 週期性的清除資訊儲存槽內的過期封包，我們更進一步利用該封包的參照次數來幫助我們管理資訊儲存槽。為此我們設計新的儲存槽管理演算法。由實驗結果得知，在平均上，

OASIS 可以達到的網路吞吐量約為傳統單點傳輸的 2.15 倍，且是 COPE 的 1.22 倍。

關鍵字：網路編碼，無線網路，OASIS，COPE



# Opportunistic Asynchronous Information Dissemination - A Forwarding Algorithm Using Network Coding

Student : Hsiang-Po Wang

Advisors : Dr. Yu-Chee Tseng

Dr. Chih-Wei Yi

Institute of Network Engineering

National Chiao Tung University



COPE, proposed by Katti et al. , is a distributed and localized network coding heuristic for wireless networks. In this thesis, we propose a generic forwarding algorithm for wireless networks, called *Opportunistic Asynchronous Information Dissemination* (OASIS). OASIS not only inherits two features from COPE, *opportunistic listening* and *opportunistic coding*, but also introduces a new feature, opportunistic information dissemination, which aggressively codes as many plain packets as possible into a packet even if those packets are not immediately necessary for neighbors. The neighbors who receive these packets would put these packets into information pool. And these packets would help the future coding and decoding process. In addition, we observe that we can further improve the network throughput if we can properly manage the information pool. We not only periodically clear the out-of-date packets as COPE does but also include a new factor, reference times, to

help us manage information pool. Simulation results show that the throughput achieved by OASIS is about 2.15 times than Unicast and 1.22 times than COPE on average.

Keywords : network coding, wireless networks, OASIS, COPE



## 誌 謝

首先，誠摯的感謝我的兩位指導教授，曾煜棋教授以及易志偉教授，在兩年來給我很多的指導跟鼓勵，並且提供良好的實驗室環境，讓我得以順利完成此篇論文。

此外，我由衷的感謝我的指導學長莊宜達，在研究上提供很多建議與指導。另外，也感謝HSCC與NOL的全體同學，在這兩年來給我的鼓勵與幫助。

最後，我要感謝我的父母以及所有關心我的人對我付出的關懷與期許，使我在挫折的時候可以再度站起來，度過最困難的時光。真的謝謝大家，因為有你們的幫忙跟鼓勵，讓我可以再在碩士兩年的生涯學到很多也經歷很多，留下許多美好的回憶。謝謝!

王湘博 於

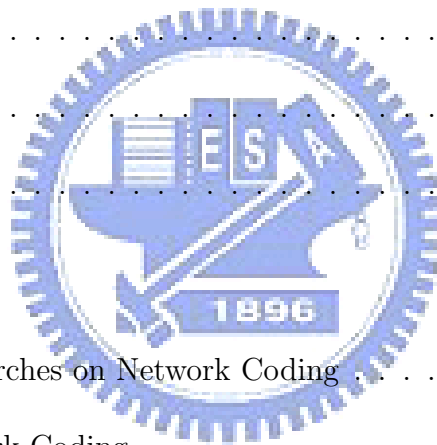
國立交通大學網路工程研究所碩士班

中華民國九十八年六月



# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Wireless Networks . . . . .	1
1.2	Network Coding . . . . .	4
1.3	Motivation . . . . .	6
1.4	Organization . . . . .	9
<b>2</b>	<b>Related Works</b>	<b>11</b>
2.1	Theoretical Researches on Network Coding . . . . .	11
2.2	Benefits of Network Coding . . . . .	15
2.3	Implementation on Network Coding . . . . .	18
<b>3</b>	<b>The OASIS Algorithm</b>	<b>23</b>
3.1	The Proposed Algorithm . . . . .	23
3.2	Implementation Details . . . . .	30
3.3	The OASIS Process Flow Chart . . . . .	34
3.4	Information Pool Management . . . . .	37



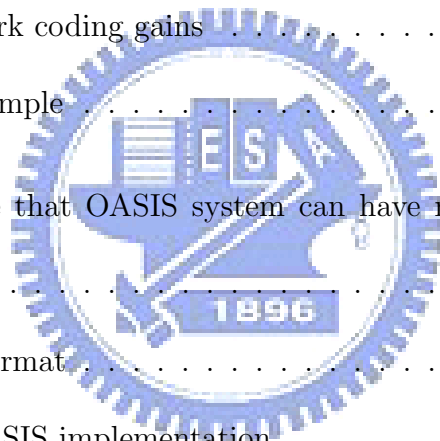


4	Simulation Results	41
5	Conclusions	49



# List of Figures

1.1	Butterfly example . . . . .	5
1.2	Alice and Bob scenario . . . . .	7
2.1	Example of network coding gains . . . . .	15
2.2	"X"-Topology example . . . . .	21
3.1	A counterexample that OASIS system can have more benefits than COPE architecture . . . . .	25
3.2	OASIS's header format . . . . .	31
3.3	Flow chart for OASIS implementation . . . . .	35
4.1	Simulation topology . . . . .	42
4.2	offered traffic load vs. throughput . . . . .	44
4.3	offered traffic load vs. end-to-end delay . . . . .	45
4.4	Normalized reference times under different offered traffic load . . . . .	46



# List of Tables


4.1 Simulation parameters . . . . .	43
-------------------------------------	----



# Chapter 1

## Introduction

### 1.1 Wireless Networks



With the advance of wireless communication technology, wireless networks play important roles in today's modern life. Wireless networks bring us much convenience. With the mobility brought by wireless networks, we can access Internet without the limit of wired links. We can move from one place to another place while we access Internet at the same time. In traditional wired networks, the case may be different. We access Internet rely on the wired links. Each computer communicates with each other using wired links through routers or end systems. There is no mobility among wired networks and computers must be located on fixed locations. People who want to access Internet have to go to some places located with computers. However, it is not convenient. Thus, in the recent researches, wireless devices have been invented and wireless networks become more and more popular around the world.

Wireless networks can be generally categorized as infrastructure wireless networks and ad hoc networks. In the infrastructure wireless network, there are four components: *access point*, *wireless medium*, *station*, and *distribution system*. An access point serves a group of stations within its service range which form a basic service set. And each basic service set owns its unique ID. Using this ID, we can identify different basic service sets in the networks. Access points work as a bridge between stations and traditional wired networks. Stations can access the Internet via wireless medium through the access points which serve the basic service sets. The distribution system is usually a wired mesh network (e.g. Ethernet) which connects access points, other distribution systems, and traditional networks to cover a larger range called extended service set. Each extended service set also owns its unique ID which is used to identify this extended service set. The infrastructure networks have a major drawback, poor survivability, because if an access point is destroyed by any artificial or natural manner, such as typhoons, earthquake, and power failure, mobile nodes can not communicate with others, and then the basic service set which is served by the access point paralyzes. In view of this, ad hoc network has no base station, in other words, ad hoc network does not contain any component acted like an access point. Thus, it is better survival than infrastructure network.

Ad hoc network, contrary to infrastructure network, is a self-organized network without the aid of any infrastructure. Ad hoc nodes, different from the devices in the traditional wired networks or the infrastructure wireless networks, have mobility and play both as routers and end systems. Further, the topology changes frequently due to the mobility of ad hoc nodes.

Two ad hoc nodes can communicate with each other if they are within the transmission range of each other. While two ad hoc nodes are out of the transmission range of each other, they can communicate with each other through the aid of intermediate nodes. Moreover, the ad hoc network can be formed by itself and has well survivability as comparing to infrastructure wireless network.

However, there are still many problems suffered by wireless networks. Most of the protocols and designs for wireless environment are extended from wired networks. For example, wireless channels are used as point-to-point links, routing adopts shortest path protocol, and using retransmission mechanism provides reliability. All of these may work well on wired environment, but less so for the unreliable and unpredictable wireless medium.

The wireless networks are totally different with wired networks. Wireless medium is a broadcast medium, while wired medium is a unicast medium. Nodes transmitting in wireless environment may interfere with each other; however, there is no interference in wired environment. Further, Wireless networks were designed for supporting mobility and portability, whereas wired networks were usually static. Thus, wireless links usually have the features high bit error rate, unreliable, and unpredictable. Most of the designs for wired networks are not fit for wireless networks because of the characteristics described above. As a result, current wireless networks suffer from many problems such as low throughput, dead spots, and inadequate mobility support.

At the first sight, the characteristics of wireless networks may all seem disadvantageous; however, a novel design may put it into another case. In the current design of wireless

networks, there is a specific node as the next-hop to relay packets, and when the packet is lost, the sender has to retransmit the packet. In this case, the reliability relies on retransmission, however, this is not efficient and has poor reliability. Well utilizing the broadcast nature of wireless networks, we can improve the network reliability. When a node broadcasts a packet, it is very likely that more than one node receives this packet. The nodes which are not the next-hop can work as the real next-hop and help to relay the packet. In this situation, when a packet is lost, there are other next-hops which may relay the packet successfully. The reliability relies on not retransmission but the multi-path next-hops. The property is called path diversity and has been explored in the literature [1, 2].

In addition, the capacity of wireless networks is limited by radio frequency bandwidth, and what's worse, wireless communication suffers the problem of radio frequency interference that further reduces network capacity especially in dense random access networks. Therefore, recent researches focus on how to improve the limit of network capacity and increase the network throughput. Network coding is a novel mechanism proposed to solve above problems. It well utilizes the broadcast nature of wireless networks and encodes multiple information into a single packet [3]. A review of network coding and its possible Internet applications can be found in [4].

## 1.2 Network Coding

The concept of network coding was first proposed in the pioneering work by Ahlswede *et al.* [3] in which they showed that multicast capacity can be increased by properly mixing

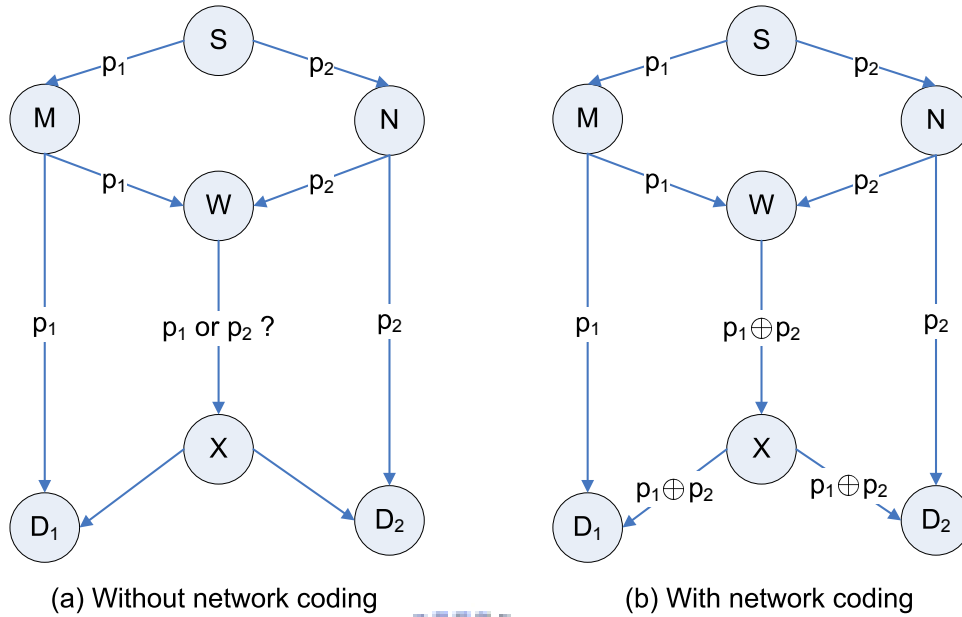


Figure 1.1: Butterfly example

information from different sources at intermediate nodes. There is a famous example in [3], the butterfly network, to illustrate the idea of network coding. In Fig. 1.1, we describe two different network models. One of these two models transmits data flows without network coding (Fig. 1.1(a)) and the other transmits data flows with network coding (Fig. 1.1(b)). The capacity of each link is one bit per unit time, and there is only one source node generating data flow. Data flows are transmitted to destinations through multi-hop relays. Note that we assume that there is no processing delay at the intermediate nodes in both models. In both models, we generate same traffic patterns in which  $S$  multicasts  $p_1$  and  $p_2$  to both  $D_1$  and  $D_2$ . In Fig. 1.1(a), a network model without network coding is depicted. The only solution is that  $S$  lets links  $\overline{SM}$ ,  $\overline{MW}$ , and  $\overline{MD_1}$  carry  $p_1$  and links  $\overline{SN}$ ,  $\overline{NW}$ , and  $\overline{ND_2}$  carry  $p_2$ . It is easy to know that the bottleneck will happen at the intermediate nodes  $W$



and  $X$ , because the intermediate nodes  $W$  and  $X$  can just relay a data bit either  $p_1$  or  $p_2$  per unit time. Hence, it leads the average network throughput to be  $\frac{3}{2}$  data bits per unit time. Fig. 1.1(b) describes the network model with network coding. The difference between Fig. 1.1(a) and Fig. 1.1(b) is that when the intermediate node  $W$  receives both data bits  $p_1$  and  $p_2$ , it combines the two bits into a data bit by calculating  $p_1 \oplus p_2$  and forwards it to the destinations  $D_1$  and  $D_2$ . The destination  $D_1$  can receive data bit  $p_1$  from link  $\overline{MD_1}$ , and extract  $p_2$  by calculating  $p_1 \oplus (p_1 \oplus p_2)$ , where  $p_1 \oplus p_2$  is received from link  $\overline{XD_1}$  at the same time. In the same way,  $D_2$  can receive both  $p_1$  and  $p_2$  at the same time. Thus, the average network throughput becomes 2 data bits per unit time and this result is better than the model without network coding. This example shows the advantage of network coding.

### 1.3 Motivation

In a network adopting network coding, nodes belonging to more than one routing path may encode multiple information from different sources into a packet and then multicast the coded packet to corresponding routing paths. Receivers can recover the original information by decoding the coded packet with the packets received before. In other words, network coding utilizes path diversity to increase network throughput. In order to make uses of network coding, the underlying networks should be with two properties: path diversity and broadcast nature. Fortunately, for wireless networks, the broadcast nature of wireless communication provides an environment to implement network coding schemes.

Here we give a possible network coding scenario in wireless networks. In Fig. 1.2, there

are one relay node Charlie, that could be a wireless AP, and two workstations Alice and Bob.

Alice and Bob need Charlie to relay messages for each other. Now, assume Alice has a packet

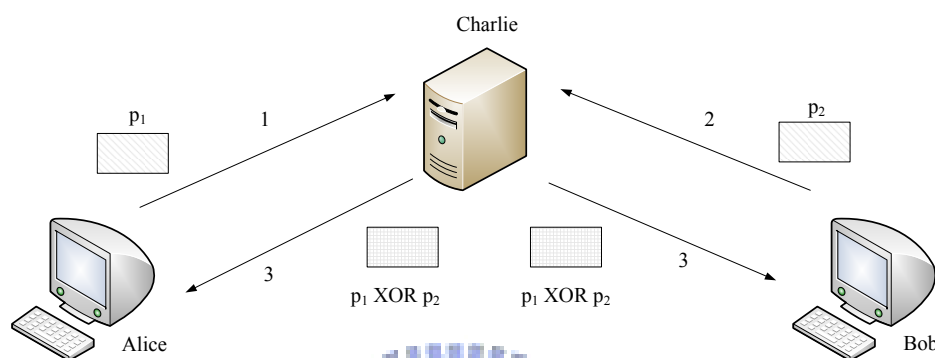


Figure 1.2: Alice and Bob scenario

$p_1$  for Bob, and Bob has a packet  $p_2$  for Alice. One possible transmission schedule is like this: Alice and Bob respectively transmit  $p_1$  and  $p_2$  to Charlie, and then Charlie relays  $p_1$  to Bob and  $p_2$  to Alice. It takes 4 time slots to complete the task. However, there is a smart way for Charlie to relay packets: After Charlie receives  $p_1$  and  $p_2$ , instead of unicasting  $p_1$  and  $p_2$ , he broadcasts  $p_1 \oplus p_2$ . Since Alice has  $p_1$ , she can find  $p_2$  by calculating  $p_1 \oplus (p_1 \oplus p_2)$ . Similarly, since Bob has  $p_2$ , he can find  $p_1$  by calculating  $p_2 \oplus (p_1 \oplus p_2)$ . So, totally only 3 time slots are needed, and the total latency is reduced by  $1/4$ . Actually, such scenarios can be found frequently in wireless networks, e.g., WiFi networks, Mesh networks, ad hoc networks, etc.. In general, network throughput can be improved by network coding. The

best part is such ideas can be implemented in a localized and distributed manner. This is an important edge for wireless ad hoc networks.

In [5], Katti et al. proposed a localized wireless network coding heuristic, called COPE, that adopts the idea illustrated in the previous example. In COPE, nodes do not need the information of the global network topology. Instead, each node learns its 1-hop neighbors' information by receiving reports which are piggybacked in packets. As one node wants to transmit a packet over wireless channels, it greedily mixes as many packets in its output queue as possible by the XOR operation if each packet is the only unknown one among all XORed packets to the corresponding packet's next-hop, and then the mixed packet is broadcasted. To prevent ambiguity, in what follows, if necessary, original packets before being XORed are called *plain* packets and mixed packets to be transmitted over wireless channels are called *coded* packets.

In the coding procedure of COPE, packets can be mixed into a coded packet only if all plain packets can be decoded by their corresponding next-hops. Behind COPE, the most fundamental data is the knowledge of packets possessed by its 1-hop neighbors. These packets are stored in a pool which is maintained by each 1-hop neighbor. COPE assumes that each node can learn the knowledge by overhearing or from piggyback information. However, COPE can be improved if we can find other means to increase the number of packets possessed by its 1-hop neighbors. Here we propose that besides the original heuristics used in COPE, we further encode packets that can be decoded by some non-next-hop nodes. By this means, nodes can possess more plentiful packets in their pool. Although these packets

may not be intended by certain node, however, they are really helpful to decoding procedure. Therefore, in following transmission, there is a good possibility of coding more plain packets, and network throughput can be further improved. We call this new feature *opportunistic information dissemination* which further codes as many plain packets as possible into coded packets even if those packets are not intended by their corresponding next-hops.

In addition, we further observe that if we have an effective approach to manage the pools when the pools are full we can increase the utility of the packets stored in the pools. Because the pools have the limit of size, the new coming packets may be dropped if the pool is full. A management approach is to make a decision about which packets have higher priority to be replaced so that the new coming packets can be kept. COPE uses time as the factor to decide if a packet should be removed. Once the packets stay in the pool over the time limit (default  $T = 0.5s$ ), the packets are removed from the pool. However, there is a probability that a packet is removed while it has not used yet. To increase the utility of each packet in the pool, we further introduce a new factor, reference times, into the management approach. We first replace the packets which have been referred and then replace the packets which stay in the pool over the time limit. Therefore, the utility of the packets can be increased.

## 1.4 Organization

The rest of this thesis is organized as follows. In chapter 2, we introduce some basic concepts of network coding and review related previous works. The proposed Opportunistic Asynchronous Information Dissemination (OASIS) algorithm is presented in chapter 3. Simulation

results and analyses are given in chapter 4. Finally, the conclusions and future works are drawn in chapter 5.



# Chapter 2

## Related Works

### 2.1 Theoretical Researches on Network Coding

After the pioneering work by Ahlswede et al. [3], a lot of works focused on coding packets based on network topology to improve network capacity. Li et al. [6] proposed a linear coding scheme for multicast traffic that can achieve the max-flow from the source to each receiving node which is the maximum capacity bound [7]. In [8], Koetter and Médard proposed polynomial time encoding and decoding algorithms that were then extended to random coding by Ho et al. [9].

In [10], Gkantsidis proposed a randomized and distributed large content distribution solution that adopts the concept of network coding and perfectly utilizes network broadcasting. When a source node wants to share a large file to other nodes, the source node divides the file into several equal size segments and broadcasts enough number of linearly independent

combinations of these segments into the network. To increase the variations of linear combinations, instead of purely relaying received combinations, intermediate nodes may broadcast a randomized linear combination of received combinations. After one node receives enough linearly independent combinations, it can recover the original file. In [11], Keller brings an idea of thinking transmit  $M$  information packets as orthonormal basis of a  $M$ -dimensional vector space. In every transmission, each node maintains its subspace of this  $M$ -dimensional vector space. They call the subspace as knowledge space. When a node collects enough information, i.e. the dimension of its knowledge space becomes  $M$ , this node can immediately obtain all  $M$  information packets. In [12], Kumar et al. proposed a broadcasting scheme over erasure channels with perfect replies. They extend the concept of Keller's idea and define a new term, called virtual queue, as the subspace of sender's knowledge space. Using the virtual queues the sender can record the information that receivers still need to receive for recovering the information packets. Further, they introduce a new notion — *Seeing* a packet by a node and classify the packet buffer management mechanisms into two categories: *drop-when-decoded* and *drop-when-seen*. The authors define that a node has seen a packet  $p$  if it receives a packet of linear combination of the form  $p+q$ , where  $q$  is also a linear combination involving only packets that arrived after  $p$  at the sender. In drop-when-decoded scheme, for a specified receiver, a packet can be dropped if and only if the packet can be decoded by all receivers after the arrival of the packet. Let  $T$  be the time of an arbitrary arrival in a steady state spending in the physical queue before departure, excluding the slot in which the arrival occurs. The packet will not depart until each virtual queue has become empty

at least once since its arrival. Let  $D_j$  be the time until the next emptying of the  $j_{th}$  virtual queue after the new arrival. Then,  $T = \max_j D_j$  and  $E[T] \geq E[D_j]$ . As for drop-when-seen scheme, the first unseen packet of each virtual queue is chosen to be encoded. As a packet is seen by a receiver, the packet is removed from the virtual queue of the receiver. A packet can be removed from the physical queue after it is seen by all receivers since that is enough for the sender to transmit only packets beyond  $p$ . However, to improve the broadcast performance, we need to make sure every receiver can obtain useful information from the coded packet. The problem is how to choose the coefficients for encoding. Kumar designs a coding algorithm to pick proper coefficients such that all receivers can see their oldest unseen packet accordingly in each transmission. Note that the definition of the virtual queue in this paper is different with our definition in following section. We will define our terms later.

In the sensor networks, Dimakis and Dan et al. exploit the network coding approach to achieve the efficiency data storage, collection and dissemination [13, 14]. Recent researches show that network coding in the specific unicast topologies can make the better throughput than traditional transmission [15][16]. Most of the previous works described focus on the coding algorithms, there are also some theoretical works on the analysis of the impact of network coding on network throughput. In [17], the authors developed the theoretical foundation for the analysis of the throughput capacity of wireless networks. The main result shows that in a randomly placed wireless networks with  $n$  nodes which are capable of transmitting  $W$  bits per seconds and have fixed transmission range, the throughput of each node can reach for a randomly chosen destination is  $\Theta(\frac{W}{\sqrt{n \log n}})$  bit per seconds. If the node's position, trans-



mission ranges, and the traffic patterns are all designed optimally, the throughput of each node can reach is  $\Theta(\frac{W}{\sqrt{n}})$  bit per seconds. These results give a good consideration to the network designers. Since then, the throughput capacity of random wireless networks has been studied extensively in the literatures[18, 19]. In the random wireless network, Lu et. al. [20] show that network coding on the physical-layer can improve the throughput capacity substantially, achieve the minimum delay and provide confidentiality. They also derived tighter bounds in two-dimensional random wireless network with unicast traffic, which is uniformly distributed among all nodes. Except for adapting network coding on network-layer, MAC-layer and physical-layer, David et. al. [21] proposed a modification of IEEE 802.11 back-off mechanism using feedback approach to improve network coding throughput over TCP data flow. It XORs forward TCP data packets and reverse TCP data packets to improve throughput. In [22], the authors design a distributed algorithm to approach optimal performance and further show that network coding can offer a constant factor of benefits under the fixed network, such as the circular network and the grid network. From the above works, it has shown that network coding can indeed improve the throughput of the multicast networks, wireless networks, TCP data flows, and so on. To implement network coding into practical, the network models should possess two properties, which are path diversity and multicast.

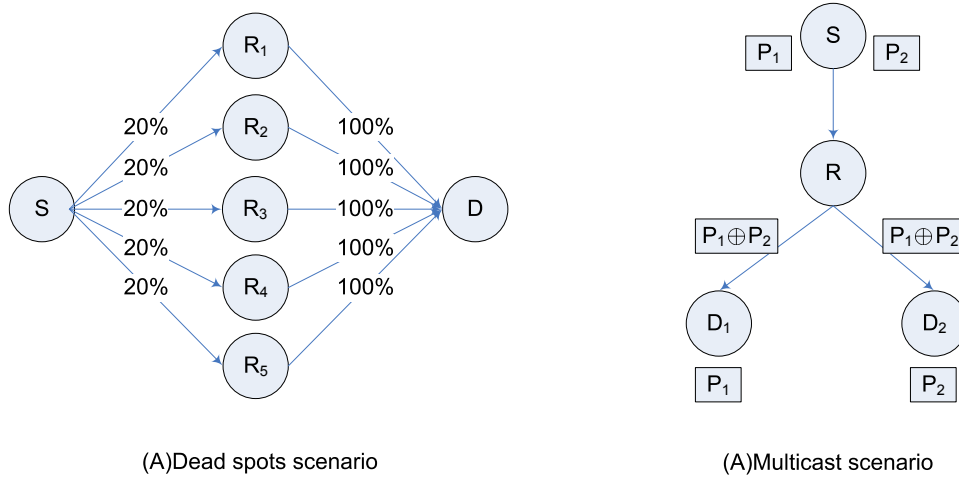


Figure 2.1: Example of network coding gains

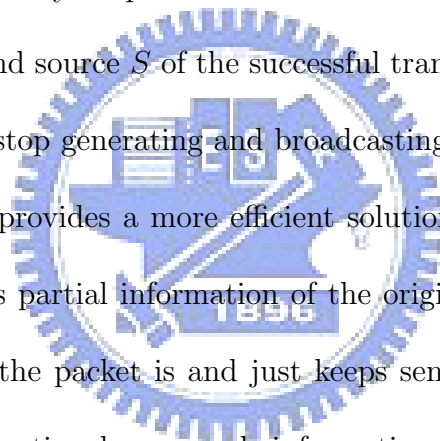
## 2.2 Benefits of Network Coding

In [23], Fragouli et al. estimate several network coding techniques, and analyze the several benefits that may be brought from network coding. Improvement on the network throughput is the most intuitive, and just like we depicted in previous section, the butterfly network. In this section, we present two examples with Fig. 2.1, which are depicted in [23]. In Fig. 2.1(a), we depict the dead spots scenario in wireless environment, and then we depict the multicast with the scenario of possible retransmission in Fig. 2.1(b). These two examples show that efficient reliability can be achieved by network coding techniques.

In 2.1(a), the successful delivery rate is labeled on the links.  $S$  is the source node and it has packets to destination  $D$ , but  $D$  is not within the  $S$ 's transmission range. The nearby

nodes  $R_i$ , where  $i$  is from 1 to 5, have the responsibility to relay packets to  $D$ . The links between  $R_i$  and  $D$  have 100% successful probability, however, there are only 20% between  $S$  and  $R_i$ .  $S$  is in a dead spot case, in which case all links to its nearby nodes have low delivery rate. In the traditional 802.11 networks,  $S$  must choose the best path to  $D$ . In this case, there is no special choice because each path has equal delivery rate. The total transmission times on average are 6 times per packet, that is, 5 times for  $S$  transmitting the packet to designate relay node and once for the relay node to  $D$ . We now consider another routing protocol called opportunistic routing protocol. Instead of unicasting packets to a designate relay node,  $S$  well uses the path diversity of wireless networks and broadcasts the packets to multiple relay nodes. The relay nodes who receive these packets work as designate relay node and relay the packets to  $D$ . In this way, the successful delivery rates from  $S$  to the relay nodes  $R_i$  increases from 20% to  $(1 - 0.8^5) \times 100\% = 67.2\%$ . This is the probability of at least one relay node receiving the packet since the nodes who receive the packet would relay the packet to  $D$  as designate relay node dose. Thus, the total transmission times on average decreases from 6 times per packet to 2.5 times per packet (1.5 times for  $S$  to relay nodes and once for relay nodes to  $D$ ). Comparing with the traditional 802.11 routing, the networks with opportunistic routing improve the throughput about 2.4 times in this case. However, there are some problems behind opportunistic routing protocol. That is, broadcast storm may happen and large amount of duplicate packets may be generated because each relay node attempts to transmit the same packets. This will waste the wireless bandwidth. The combination of path diversity and network coding provides an elegant solution to solve this problem. Assume

$S$  has  $n$  packets to be transmitted. In each transmission,  $S$  generates a linear combination of the form  $p'_i = \sum_{j=1}^n c_{ij}p_j$ , and broadcasts it to its relay nodes. The nodes who receive  $p'_i$  can participate in relaying packets to  $D$ , and they generate a new linear combination of the packets received before. The purpose of this step is to increase the variations of linear combinations such that every transmission surely carries partial information of the original  $n$  packets. As a result, it will decrease the number of duplicate packets flooding in the network because each coded packet surely carries useful information. Once the destination  $D$  receives any  $n$  packets with  $n$  linearly independent coefficient vectors,  $D$  can decode and recover the original  $n$  packets by simple matrix inversion. And then  $D$  will schedule an ACK to notify the relay nodes and source  $S$  of the successful transmission.  $S$  and the relay nodes who receive the ACK will stop generating and broadcasting a new linear combination. The approach described above provides a more efficient solution to reliability. That is because each coded packet contains partial information of the original packets. The sender has no need to care about which the packet is and just keeps sending linear combinations to the destination until the destination has enough information to decode the original packets. In other words, without network coding, the sender should exactly know which packets the destination misses, and retransmit them to the destination. This mechanism relies on feedbacks between source and destination; however, this will significantly consumes wireless bandwidth to communicate feedbacks reliably in a unreliable wireless environment. Contrary, with network coding, we just need one ACK as feedback to notify the sender of the successful transmission. It is more bandwidth-saving and more efficient. Above idea is adopted and



implemented by MORE protocol which is proposed in [24], and we will further introduce it in next section.

In 2.1(b), we describe a multicast scenario.  $S$  wants to multicast a video stream or data stream to both  $D_1$  and  $D_2$ .  $S$  first transmits  $p_1$  and  $p_2$  to relay node  $R$ , and then  $R$  broadcasts  $p_1$  and  $p_2$  to  $D_1$  and  $D_2$ . Because the wireless receptions of different nodes is highly independent according to the analysis in [25, 26], it is possible that when  $R$  broadcasts  $p_1$  and  $p_2$ ,  $D_1$  only receives  $p_1$  while  $D_2$  only receives  $p_2$ . In this situation,  $R$  should retransmit  $p_1$  to  $D_2$  and  $p_2$  to  $D_1$  respectively. Using network coding,  $R$  can simply broadcast  $p_1 \oplus p_2$  and then both  $D_1$  and  $D_2$  can recover their corresponding losses in a single transmission (this is just like we do in the butterfly example). This approach provides a more efficient reliability than traditional retransmission mechanism.

## 2.3 Implementation on Network Coding

In the wireless networks, broadcast nature of wireless communication provides a good environment to implement network coding schemes. Because of the broadcast nature of wireless medium, receivers can receive packets if they are within the transmission range of the sender. Moreover, when a wireless device transmits packets to the next-hops, there is high probability for 1-hop neighbors to overhear packets. The broadcast nature can satisfy both path diversity and multicast condition that are required for implementing network coding. Hence, the wireless environment is applicable for the implementation of network coding.

Recently, researchers focus on the possibility of implement network coding on wireless networks such as ad-hoc networks, mesh networks, WiFi, etc. In the presence of omnidirectional antenna, the problem of minimizing communication costs can be formulated by linear programming and solved in a distributed manner [27].

In [24], Chachulski et al. implement a MAC-independent opportunistic routing protocol called MORE. The MORE architecture inserts a MORE-layer between IP-layer and network-layer. It well exploits the natures of wireless networks, that is, broadcast nature of the wireless medium, path diversity, and the large amount of data redundancy. Source nodes divide a file into batches of  $K$  segments and transmit the linear combination of the  $K$  segments. MORE-layer header is used to record the coefficient vector, and the built in forwarding list which is a list of intermediate nodes that are responsible for forwarding the coded segment. The nodes which have the responsibility for forwarding the coded segments randomly pick coefficient vectors to compute a new linear combination of the segments received so far and rebroadcast this coded segment into networks. At the destinations, once receiving the  $K$  segments which have  $K$  linear independent coefficient vectors, the node can recover the original batch by using simple matrix multiplication:

$$\begin{bmatrix} p_1 \\ \cdot \\ \cdot \\ p_K \end{bmatrix} = \begin{bmatrix} c_{11} & \cdot & \cdot & c_{1K} \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ c_{K1} & \cdot & \cdot & c_{KK} \end{bmatrix}^{-1} \begin{bmatrix} p'_1 \\ \cdot \\ \cdot \\ p'_2 \end{bmatrix} \quad (2.1)$$

, where  $p_i$  represent the segment before linear coding in current transmitting batch, and

$p'_i$  represent the coded segment whose coefficient vectors is  $\vec{c}_i = \langle c_{i1}, \dots, c_{iK} \rangle$ , for all  $i$  from 1 to  $K$ . MORE is implemented on a 20-node testbed and it demonstrates the great improvement on network throughput. The coding style that MORE exploits is usually called intra-flow network coding because the segments which are coded into a coded segment are all sent to the same destination.

Oppose to MORE, in [5], Katti et al. proposed a new network coding technique for wireless mesh networks, called COPE, which is implemented in a coding layer also inserted between the IP and MAC layers like MORE and features in opportunistic coding in a single transmission to increase the system throughput. COPE introduced two features of wireless network coding:

- *Opportunistic listening*: Due to the broadcast nature of wireless communications, nodes can overhear packets sent by their neighbors. In order to make use of this feature, nodes are set in a promiscuous mode and use the overheard information to decode a coded packet.
- *Opportunistic coding*: A node may have many ways to learn the information of what its neighbors have heard so far, such as periodical reports sent by the neighbors or predicting based on routing protocol. While sending packets, a node uses this information to code the payload of multiple packets by the XOR operation when each receiver has sufficient information to decode the coded packet. To distinguish the difference coding style from MORE, this kind of coding style is called inter-flow network coding because the coded packets are mixed by the packets sent to different destinations.

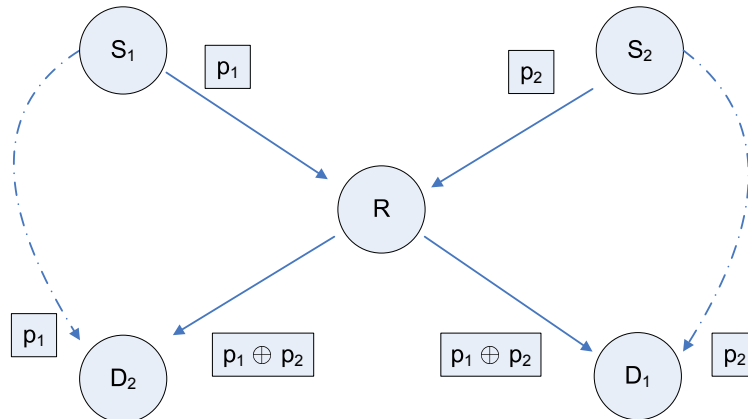


Figure 2.2: "X"-Topology example

COPE demonstrates the possibilities of putting network coding into practice. An example extended from Alice-and-Bob scenario can give us an intuition of the great benefits brings from network coding. In the real networks, the probability of the occurrence of the Alice-and-Bob scenario is quite low. However, One can always assume that the "X"-topology depicted in Fig. 2.2 exists in the real networks. In fact, we can view the real networks as the combination of many small "X"-topology networks. In Fig. 2.2,  $R$  represents the mobile router where two information flows intersect at. There are two source  $S_1$  and  $S_2$  that have data for destination  $D_1$  and  $D_2$ , respectively. Both  $S_1$  and  $S_2$  need the help of  $R$  to forward their packets. However,  $D_1$  is within the transmission range of  $S_2$ , and  $D_2$  is in the transmission range of  $S_1$ . Because of the broadcast nature of wireless network,  $D_1$  can overhear the packet  $p_2$  sent by  $S_2$ , and  $D_2$  can overhear the packet  $p_1$  sent by  $S_1$ . This situation provides the great opportunities for using network coding techniques. A simple



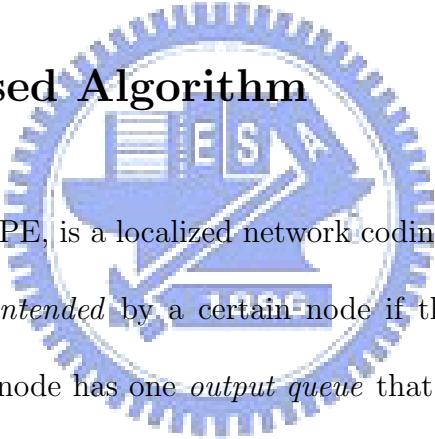
way is XORing  $p_1$  and  $p_2$  at the intermediate node  $R$ , and let  $R$  broadcast the XORed packet of  $p_1$  and  $p_2$ . Both  $D_1$  and  $D_2$  can extract the packets that they intend because both of them have possessed the partial information (i.e.  $p_1$  or  $p_2$ ) of the XORed packet. The total latency is reduced by 1/4 comparing with original unicast transmission. The authors will exploit the property of this "X-topology" and experiment on three topologies: Alice-and-Bob, "X"-topology, and the cross topology. It really demonstrates the great utility of network coding.

In this thesis, we adopt these two features, *Opportunistic listening* and *Opportunistic coding*, from COPE and further introduce a new feature in our work. We have two observations from COPE. First, COPE always codes packets that are intended by certain neighbors. Second, each node maintains an information pool to record the overheard packets and uses the packets in the information pool to decode packets. Base on the above observations, a packet may not be XORed into a coded packet if 1) it is not intended by a certain next-hop or 2) the packet can not decode for a receiver by decoding procedure due to the lack of sufficient information in the receiver's information pool. If the first observation is relieved from COPE, there may have more information disseminated to the neighbors of a node and then increase the probability of coding more packets in the future transmission. Therefore, we introduce a new feature, called *opportunistic information dissemination*, which tries to XOR as many packets as possible into a coded packet even if some packets are not intended by the receivers.

# Chapter 3

## The OASIS Algorithm

### 3.1 The Proposed Algorithm



OASIS, an extension of COPE, is a localized network coding protocol for wireless networks. We say a plain packet is *intended* by a certain node if this node is the next-hop in the route of this packet. Each node has one *output queue* that contains all plain packets to be transmitted. For simplicity, we denote the output queue as *queue* in the following sections. Each node also maintains per-neighbor *virtual queues*, each of which is corresponding to a 1-hop neighbor. A virtual queue contains a subsequence of packets of the *queue* and all of these packets are intended by the corresponding neighbor. Let  $virtualQueue_w$  denotes the virtual queue maintained by a certain node for its 1-hop neighbor  $w$ . Besides, each node also maintains an *information pool* that records a copy of packets that has been received or sent before. The information pool is for the use of decoding coded packets. The more

packets in the neighbors' information pools, the higher probability one can encode packets to these neighbors. The reason is that while we encode packets to the intended neighbors, we must ensure that all of the intended neighbors can decode the coded packet, and this relies on what packets the neighbors have in their information pools. In the following sections, we use *infoPool* as a shorthand for the information pool and *infoPool<sub>v</sub>* to indicate the *infoPool* of the specified node *v*. A node *knows* a packet means that this packet is in its *infoPool*. In other words, the node has ever received, overheard or sent this packet before. A node can decode a plain packet from a coded packet by decoding procedure. Besides the above data structures, a node maintains a *knowledge database* which records the packet list from its 1-hop neighbors' information pool. We denote *knowledge<sub>w</sub>* as the knowledge database maintained by a certain node for its neighbor *w*. This packet list can be obtained by information exchange, e.g. periodical control message or piggyback information. As one node does encoding decision, it needs to make sure that receivers can decode the coded packets. Therefore, the knowledge about neighbor's *infoPool* is a key to ensure the coded packets can be correctly decoded. We will describe the decoding procedure formally later.

We can improve the performance of network coding by enriching the *infoPool*. OASIS aggressively encodes as many plain packets as possible into a coded packet even if some plain packets are not intended by the receivers. The reason for doing so is to increase the number of packets in the *infoPool* of neighbors such that we have higher probability to encode more packets in the following transmission.

Here we give an example to illustrate the possible benefit of OASIS in Fig. 3.1. There

are 3 nodes,  $n_1$ ,  $n_2$  and  $n_3$ , and a relay node  $R$ .  $R$  has packets  $p_1$ ,  $p_2$ , and  $p_3$  in its *queue* for  $n_1$ ,  $n_2$ , and  $n_3$ , respectively.  $R$  will classify the packets to corresponding virtual queues, e.g. put  $p_1$  into *virtualQueue* $_{v_1}$ ,  $p_2$  into *virtualQueue* $_{v_2}$ , and  $p_3$  into *virtualQueue* $_{v_3}$ . Besides,  $n_1$ ,  $n_2$ , and  $n_3$  have already received or overheard  $p_2$ ,  $p_3$ , and  $p_1$ , respectively and store them in their *infoPools* accordingly.  $R$  can record the corresponding information in its knowledge databases *knowledge* $_{n_1}$ , *knowledge* $_{n_2}$ , and *knowledge* $_{n_3}$ , respectively, and use these information to encode packets. The content of virtual queues and knowledge databases are shown in Fig 3.1. According to COPE protocol, relay node  $R$  will unicast  $p_1$  to  $n_1$  since

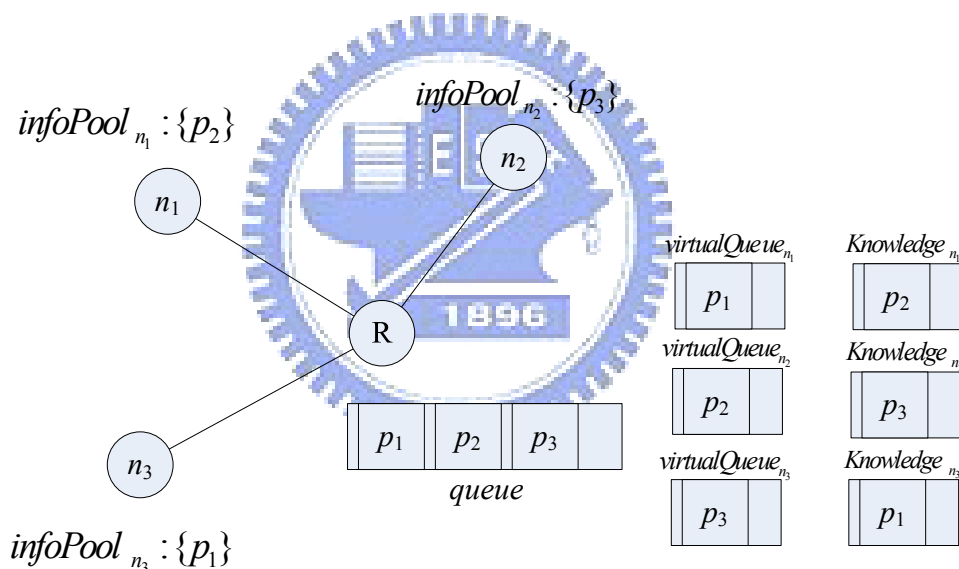


Figure 3.1: A counterexample that OASIS system can have more benefits than COPE architecture

$p_1$  is at the head of the *queue* and no other plain packets can be encoded together. After  $p_1$  is transmitted,  $n_1$  and  $n_2$  can update their *infoPools*. Therefore, the content of *infoPool* for

$n_1$  becomes  $\{p_1, p_2\}$  and that for  $n_2$  becomes  $\{p_1, p_3\}$ , respectively. Later,  $p_2$  is unicasted to  $n_2$  for the same reason of  $p_1$ . The content of *infoPool* for  $n_2$  is updated to  $\{p_1, p_2, p_3\}$  and that for  $n_3$  is updated to  $\{p_1, p_2\}$ . Finally,  $p_3$  is unicasted to  $n_3$ . Therefore, COPE executes three unicasts sequentially and needs three time slots to complete the task. There does not exist any encoding events if we follow the COPE protocol. Therefore, in this case, COPE performs like the ordinary unicast. However, there is a coding schedule that only needs two time slots to complete the same task. Let  $R$  transmit  $p_1 \oplus p_2$  and  $p_2 \oplus p_3$  sequentially. After  $p_1 \oplus p_2$  is transmitted, then,  $n_1$  can decode  $p_1$  from  $p_1 \oplus p_2$  by calculating  $p_2 \oplus (p_1 \oplus p_2)$  since  $n_1$  has already known  $p_2$ . Although  $p_2$  is not intended by  $n_3$ ,  $n_3$  can decode  $p_2$  from  $p_1 \oplus p_2$  by the similar calculation like  $n_1$  does and increase the content of its *infoPool*. Therefore, the content of *infoPool* for  $n_3$  is updated to  $\{p_1, p_2\}$ . In the following transmission,  $p_2$  and  $p_3$  can be encoded together since both  $n_2$  and  $n_3$  have enough information  $p_3$  and  $p_2$  to decode their intended packets,  $p_2$  and  $p_3$ , respectively. Finally, all  $n_1$ ,  $n_2$  and  $n_3$  can successfully decode their intended packets. In this case, the throughput of  $R$  is improved from 1 (packet/time slot) to 1.5 (packet/time slot). As we saw from the above example, if we consider not only the intended packets in the coding procedure but also the packets that may increase the opportunity of encoding more packets in the future (e.g. we transmit  $p_2$  to  $n_3$  in the first transmission), we can gracefully make use of the network coding technique and efficiently increase the system throughput.

OASIS is a two-phase greedy coding algorithm. In the first phase, the algorithm encodes as many intended plain packets as possible into the coded packet such that all receivers of

the coded packet can successfully decode their intended plain packet. This phase is similar to COPE. In the second phase, we implement the *opportunistic information dissemination* concept. Without affecting the decoding process of the plain packets encoded in the previous phase, the algorithm further encodes as many unintended plain packets as possible into the coded packet. This can increase common knowledge among neighbors (i.e.  $infoPool_{n_i}$  and  $knowledge_{n_i}$ ) and therefore is helpful for following encoding processes and potentially increases network throughput. The details of the coding procedure is presented in PROCEDURE 1.

As one node is going to transmit a packet, it first dequeues a (plain) packet  $p$  from its *queue* and assigns it to coded packet  $codedPacket$ . The node records the next-hop of  $p$  in  $nextHop$  that keeps a list for all receiving nodes in this coding procedure. Then, for all its 1-hop neighbors  $w$  from its 1-hop neighbor list excluding all next-hops  $neighbor/nextHop$ , it dequeues a packet  $q$  from the virtual queue  $virtualQueue_w$  and checks whether for each node  $v$  in  $nextHop \cup \{w\}$  can decode  $codedPacket \oplus q$ . If  $codedPacket \oplus q$  can be decoded by all intended nodes ( $nextHop \cup \{w\}$ ), then update  $nextHop$  and  $codedPacket$ , and remove  $q$  from *queue*. This is the first phase of the coding procedure. The heuristic is greedily encoding as many intended packets as possible in a transmission. In the second phase of the coding procedure, for each packet  $r$  in the *queue*, if there exists a node  $w$  belonging to  $neighbor/nextHop$  such that for each node  $v$  in the  $nextHop$  can decode its intended packet and the node  $w$  can decode the packet  $r$  from  $codedPacket \oplus q$ , then update  $codedPacket$  and  $nextHop$ . Note that, in order to reserve the transmission order,  $r$  can not be intended

packet for  $w$  since we have already coded all packets that are intended for certain receivers in the transmission order in first phase. Therefore,  $r$  is not removed from the *queue* since  $r$  is just used to increase  $w$ 's *infoPool*. In the second phase, opportunistic information dissemination is a greedy heuristic for increasing the common knowledge for the sake of future transmissions.

A node decodes intended packet from the *codedPacket* by the following decoding procedure as shown in PROCEDURE 2. The decoding procedure is simple and similar to that in COPE architecture. A node can decode the plain packet for itself from the *codedPacket* with  $n$  plain packets by using XOR operation if it has exactly these  $n - 1$  plain packets excluding its intended one in its *infoPool*. For example, assume that *codedPacket* is encoded with  $n$  packets,  $p_1, p_2, \dots, p_n$ , and  $v$  is intend  $p_1$ . Suppose that  $v$  has already known  $p_2, \dots, p_n$ . Then  $v$  can decode  $p_1$  by simply XOR all  $p_2, \dots, p_n$  with *codedPacket*. If a node receives, overhears or decodes a plain packet, then it checks whether this packet is intended by itself. If this plain packet is intended by this node, then the node stores this packet to the *queue* for later relay if it is the relay node or sends to upper layer for further processing if it is the destination. If this plain packet is not intended by this node, the node stores this packet into the *infoPool* for future decoding purposes. The intuitive idea is that the more plentiful the neighbors' *infoPools* are the higher probability that a node can encode more plain packets. That is why we incorporate the opportunistic information dissemination into the second phase of the coding procedure.

---

**PROCEDURE 1** *OASIS\_Coding* (*queue*, *neighbor*)

---

**Require:** *queue*: the output queue; *neighbor*: the 1-hop neighbor list.

dequeue  $p$  from *queue*.

$codedPacket = p$

$nextHop = \{\text{the next hop of } p\}$

Phase 1:

**for**  $\forall w \in neighbor/nextHop$  **do**

    let  $q$  denote the first packet in  $virtualQueue_w$

**if**  $\forall v \in nextHop \cup \{w\}$ ,  $v$  can decode  $codedPacket \oplus q$  **then**

$codedPacket = codedPacket \oplus q$

$nextHop = nextHop \cup \{w\}$

        remove  $q$  from *queue* and also  $virtualQueue_w$

**end if**

**end for**

Phase 2:

**for**  $\forall$  packet  $r \in queue$  **do**

**if**  $\exists w \in neighbor/nextHop$  such that  $w$  and  $\forall v \in nextHop$  can decode  $codedPacket \oplus r$

**then**

$codedPacket = codedPacket \oplus r$

$nextHop = nextHop \cup \{w\}$

**end if**

**end for**

**return**  $codedPacket, nextHop$





---

**PROCEDURE 2** *OASIS\_decoding* ( $v, codedPacket$ )

---

**Require:**  $codedPacket = \{p_1, p_2, \dots, p_n\}$ ;  $v$ : the receiving node

$v$  tries to decode the packet  $p_x$ , where  $x \in \{1, 2, \dots, n\}$ .

$decodedPacket = codedPacket$

**if**  $v$  knows  $\forall p_i \in codedPacket / \{p_x\}$  in  $infoPool_v$  **then**

**for**  $\forall p_i$  **do**

$decodedPacket = p_i \oplus decodePacket$

**end for**

$p_x = decodedPacket$

**if**  $v$  is the next hop of  $p_x$  **then**

  put  $p_x$  into the  $queue_v$

**else**

  put  $p_x$  into the  $infoPool_v$  for future decoding purpose

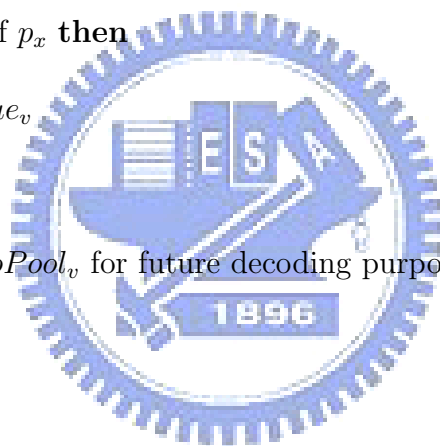
**end if**

**else**

  drop the  $codedPacket$

**end if**

---



## 3.2 Implementation Details

OASIS improves COPE with aggressively adding more information during a single transmission. This chapter describes some implementation details about packet formats, information

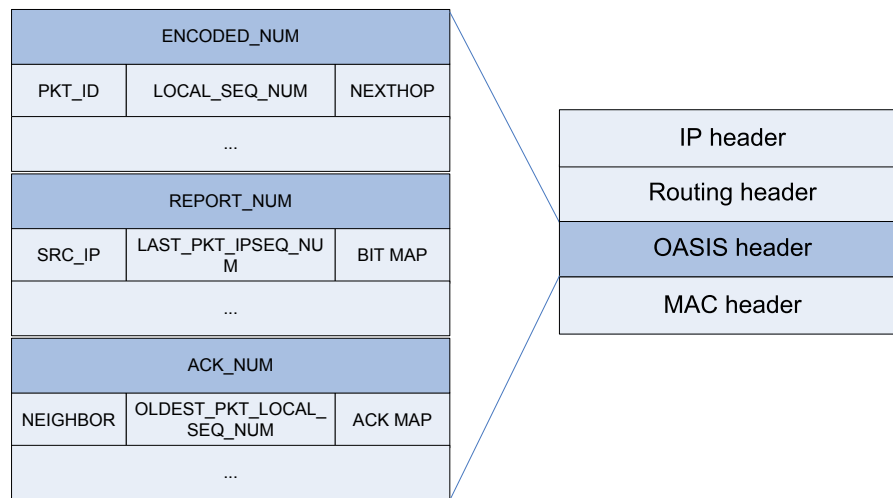


Figure 3.2: OASIS's header format

reports and packet acknowledgements. OASIS adds a variable-length header in each packet between IP and MAC header as shown in Fig. 3.2. The OASIS's header consists of three blocks:

- Information about the coded packets:** The first block records all plain packets that are encoded in this transmission. It starts with ENCODED\_NUM which represents the number of plain packets that are XORed together. The ENCODED\_NUM field is followed by a variable length list of entries that contains the information of all plain packets. Each entry in the list represents a plain packet and it contains three fields to record the information of the plain packet. First field, PKT\_ID, records the packet's unique ID, which is a 32-bit hash of the packet's source IP address and IP

sequence number. The packet ID is implemented as in COPE. This is followed by LOCAL\_SEQ\_NUM in the second field. Each node maintains a per-neighbor 16-bit counter, called Neighbor Seqno Counter. Whenever the node sends a plain packet to that neighbor, the according counter is increased by 1 and its value is assigned to the plain packet as a local sequence number, LOCAL\_SEQ\_NUM. The two neighbors use this sequence number to identify the packet. Finally in the third field, NEXTHOP, the MAC address of the next-hop of this plain packet, is used to identify whether a receiving node intends to any plain packet which are XORed into the coded packet, in which case it decodes the packet, and processes it further.

- **Reports of infoPools for increasing knowledge database:** The reports of *infoPools* is the second block in the OASIS's header. As shown in the Fig. 3.2, the report information block starts with the number of the report entries brought with this packet called REPORT\_NUM. Each entry firstly indicates the source IP address (SRC\_IP) and is followed by LAST\_PKT\_IPSEQ\_NUM, the IP sequence number of the last packet, and a bit-map that records recently received packets from that source. For example, a report entry of the form {140.113.158.101, 30, 00010001} represents that the neighbor reports the packets with source IP address 140.113.158.101, and it includes three reported packets with IP sequence number 30, 29, and 25. The reported information is important since nodes in a realistic wireless environment need this information to update their knowledge databases and perform coding process. This block is implemented as in COPE. Note that since the size of the *infoPool* is limited, the

packet will be dropped when there is no free space to store the packet. It leaves a problem that how we can manage the *infoPool* effectively such that the *infoPool* can store more useful packets and increase the utility of each packet in *infoPool*. We describe our approach later and compare it with the approach adopted by COPE.

- Cumulative acknowledgement for reliable transmission:** It is possibly for more than one packet are transmitted in a single transmission under network coding paradigm. However, the traditional RTS/CTS/DATA/ACK mechanism in MAC can only protect the ongoing packet (i.e. the first packet in *queue*). To ensure other packets coded together in a XORed packet with minimum protection, we use a modified cumulative ACKs from COPE. The ACK block starts with ACK\_NUM, the number of ACK entries in this packet. The first field of an ACK entry is the MAC address of the neighbor (NEIGHBOR), followed by a local sequence number of the oldest packet (OLDEST\_PKT\_LOCAL\_SEQ\_NUM) to indicate where the ACK starts. This is followed by a cumulative ACK bit-map indicating previously received and missing packets. For example, an entry of the form  $\{00 - 10 - 5A - 81 - CA - FD, 30, 10001000\}$  represents that the targeted neighbor to be ACKed has MAC address  $00 - 10 - 5A - 81 - CA - FD$ , and the packets to be ACKed have local sequence numbers 30, 31, and 35. As we mentioned before, the local sequence number of a plain packet is used to identify the plain packet between the two neighbors. Thus, a node can schedule retransmission or drop the packets from *queue* according to the local sequence numbers of the plain packets recorded in ACK block that it receives. Note

that, we ACK starting from the oldest packet, that is different with COPE. The reason for doing so is to make sure that every received packet can be ACKed at least once. Unlike COPE, it always ACKs the latest 9 packets and has the probability to miss ACKing some of the older packets, our strategy has stronger reliability and can avoid the waste of bandwidth caused by large amount of retransmission because of missing ACK. To avoid too many redundant ACKs, every plain packet is ACKed for no more than 2 times.

### 3.3 The OASIS Process Flow Chart

In Fig. 3.3, we depict the flow chart of OASIS implementation. Fig. 3.3 (a) describes the process flow on sender side, and Fig. 3.3 (b) describes the process flow on receiver side. On the sender side, shown in Fig.3.3 (a), when MAC is ready to send, the sender dequeues the first packet in its *queue*. And then, the two phases coding procedure of OASIS is executed. As we mentioned before, in the first phase of the procedure, we focus on coding as many intended plain packets as possible. In the second phase, we incorporate the feature, opportunistic information dissemination, into coding procedure for the purpose of increasing the content of neighbors' *infoPool*. After finishing the coding procedure, we add information about the coded packet if the result packet is encoded. And then, the node adds reports of its *infoPool* to inform all of its 1-hop neighbors to update their knowledge databases. Before the packet can depart from this node, the ACK entries are added to the header to ACK received packets to its neighbors who sent these packets.

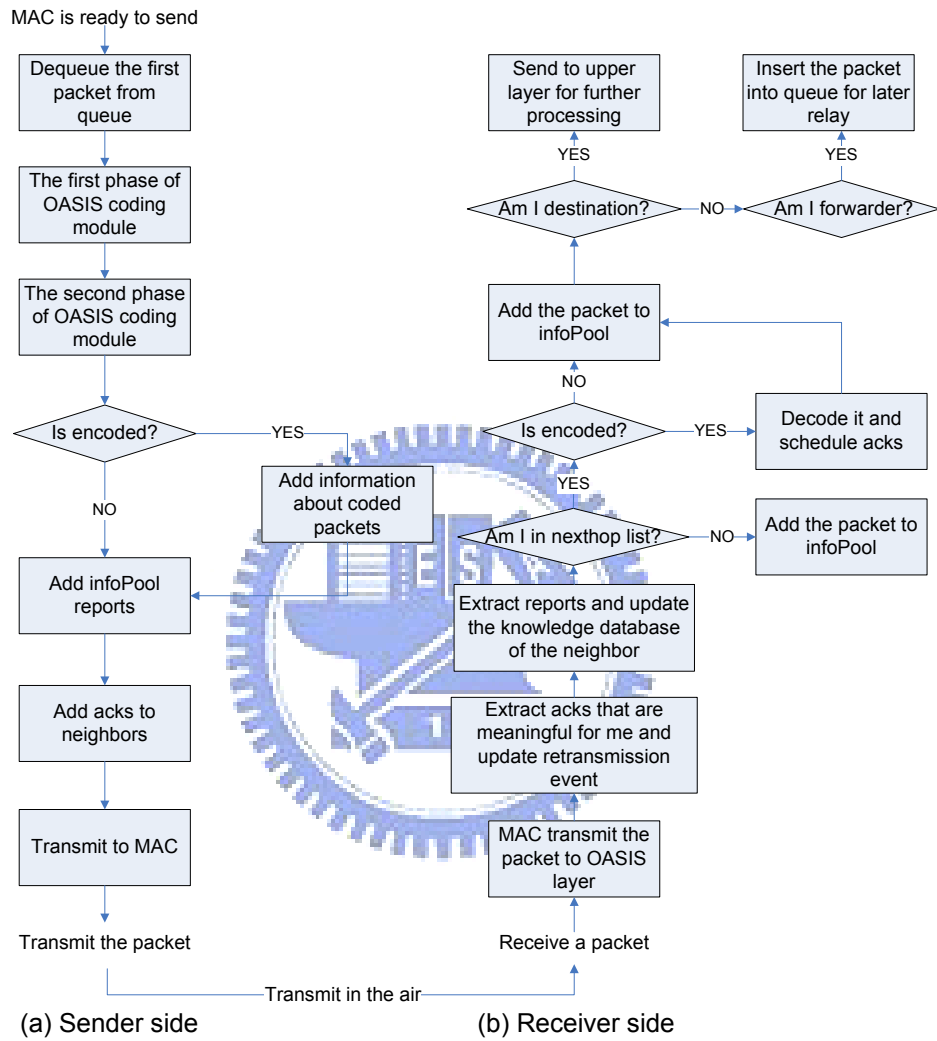


Figure 3.3: Flow chart for OASIS implementation

On the receiver side, when a packet arrives, MAC will process this packet and then transmit it onto OASIS layer. The receiving node first extracts ACKs that are meaningful for it, and updates the retransmission events or drops the ACKed packets. And then, the node extracts all report entries to update the knowledge database of this neighbor, which is the view to the neighbor's *infoPool*. After extracting useful information for it from the header, the node can check whether it has the responsibility to receive this packet from the *nextHop* list which is recorded in the OASIS header. If the node is not in the *nextHop* list, then it simply puts the packet into its *infoPool*. If the node is in the *nextHop* list, it further checks if the packet is coded. If it is, then OASIS decoding procedure is invoked, and the node tries to decode the XORed packet using the plain packets stored in its *infoPool* to retrieve the intended packet for itself. After successful decoding, the node immediately schedules ACK events to notify the sender of the successful transmission. After retrieving the intended packet, the node now checks that whether it is the destination of this packet or not. If it is the destination, the node sends the packet to upper layer for further processing. Otherwise, if the node is a forwarder which has the responsibility to forward this packet, it inserts this packet into the *queue* for later relay. If the packet is not coded, the node simply puts the packet into *infoPool*, and processes the packet in the same way as a coded packet.

### 3.4 Information Pool Management

As we mentioned before, one of the keys to make the network coding practical is information pool. Due to the limited size of *infoPool*, how to manage *infoPool* is an important issue. If we can manage *infoPool* effectively, then we can effectively utilize the benefit of OASIS protocol.

In [5], this issue is not be discussed much. The original approach that COPE adopts is buffering the received packet for a period time (the default value is set to  $T = 0.5s$ ). In this way, the size of *infoPool* may be small; however, it does not well utilize the received packets. That is, the received packets may be dropped before they can be used to decode plain packets.

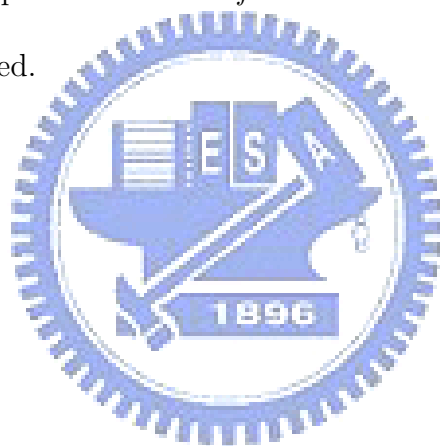
We further observe a situation that once a packet in the *infoPool* has been used to decode a plain packet from a coded packet, then the probability of the packet being used again is really low. We call that this packet is referred. If the packets in the *infoPools* of the neighbors of a certain sender have been referred in a certain transmission, then which implicitly means that this sender has mixed all of these packets into a coded packet. In addition, there must have some potential receivers intending for these packets. If all these potential receivers successfully decode their intended packets, then the sender will drop all of these packets and never send them again. In other words, there is small probability for these packets in the neighbors' *infoPools* being referred again since these packets have already removed from the sender's *queue*. The only situation that the packets are sent again is the occurrence of the retransmission events. However, there are rarely such events in our



experiments. The conditional probability of packets referred twice under that are referred once is no more than 2% as shown in Fig. 4.4. As the above observation, we propose a new approach to manage *infoPool*. The detail of the management algorithm is depicted in PROCEDURE 3.

In our proposed algorithm depicted in PROCEDURE 3, we use not only the factor adopted by COPE, the time that a packet stays in the *infoPool*, but also a new factor, that is the number of times that a packet is referred. In addition, we consider that a packet which has been referred has lower probability to be referred again than which has long staying time. Thus, in our algorithm, we give the higher priority for a packet being replaced to the packets which have been referred. A node  $v$  inserts a plain packet  $p$  into its *infoPool<sub>v</sub>* when it generates, relays and overhears a plain packet or decodes a plain packet from coded packet. It first checks whether *infoPool<sub>v</sub>* is full or not. If the *infoPool<sub>v</sub>* is not full,  $v$  simply put  $p$  into *infoPool<sub>v</sub>*. If *infoPool<sub>v</sub>* is full, then one packet must be remove from the *infoPool* to free a slot for  $p$ . We first check that if there exists a packet  $q$  belonging to *infoPool<sub>v</sub>* such that the inequality,  $q.refTimes > REFER\_TIMES\_LIMIT$ , can be held, where  $q.refTimes$  represents the reference times of  $q$  and  $REFER\_TIMES\_LIMIT$  is defined to be the threshold of reference times that a packet can be referred. Here, we set the value of  $REFER\_TIMES\_LIMIT$  to be 1, that is, we can replace a packet once it has been referred. If there is such packet  $q$ , then we replace  $q$  with the new packet  $p$  and finish the management procedure. If there is no such packet in *infoPool<sub>v</sub>*, then we further check that if there exists a packet  $q$  belonging to *infoPool<sub>v</sub>* such that the inequality,

$q.stayTime > STAY\_TIME\_LIMIT$ , can be held, where  $q.staytime$  represents the time that  $q$  stays in  $infoPool_v$  and  $STAY\_TIME\_LIMIT$  is defined to be the threshold of the time that a packet can stay in  $infoPool_v$ . Here, we set the value of  $STAY\_TIME\_LIMIT$  to be  $0.5s$ , which is the same setup with the default value of COPE. If there is such packet  $q$ , we replace  $q$  with the new packet  $p$  and finish the management procedure. Finally, if both check described above are failed, we randomly choose a packet  $q$  belonging to  $infoPool_v$ , and replace  $q$  with the new packet  $p$ . The reason for doing so is that we think that a new coming packet has more potential to be used in the future. That is because, in a network with high traffic load, the packets in the  $infoPool$  are soon out of date. Thus, frequently updating  $infoPool$  is needed.



---

**PROCEDURE 3** *OASIS\_infoPool\_management* ( $v, infoPool$ )

---

**Require:** node  $v$  receives a packet  $p$ , and want to insert it into  $infoPool_v$

**if**  $infoPool_v$  is full **then**

**if**  $\exists q \in infoPool_v$ , and  $q.refTimes > REFER\_TIMES\_LIMIT$  **then**

replace  $q$  with  $p$

**else if**  $\exists q \in infoPool_v$ , and  $q.stayTime > STAY\_TIME\_LIMIT$  **then**

replace  $q$  with  $p$

**else**

randomly pick  $q \in infoPool_v$

replace  $q$  with  $p$

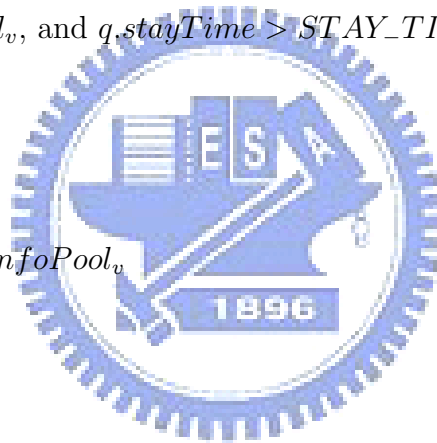
**end if**

**else**

put  $p$  into  $infoPool_v$

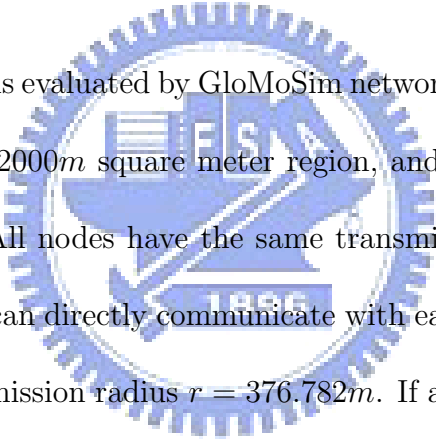
**end if**

---



# Chapter 4

## Simulation Results



The performance of OASIS is evaluated by GloMoSim network simulator [28]. The simulation environment is a  $2000m \times 2000m$  square meter region, and 16 nodes are placed in a  $4 \times 4$  grid as shown in Fig.4.1. All nodes have the same transmission (and collision) range with  $r = 376.782m$ . Two nodes can directly communicate with each other if the distance between them is less than the transmission radius  $r = 376.782m$ . If a packet is retransmitted 2 times and still can't go through the link, the packet will be dropped. Note that in COPE or OASIS, packets are transmitted in multicast way. If collisions happen at some nodes, we only need to retransmit those collided plain packets and run OASIS based on the updated *queues*. We use cross-like CBR traffic flows in our simulation: nodes in the grid sides independently generate CBR traffic to the node in the other side based on Poisson arrival with packet generating rate varied between  $0.037Mbps$  to  $0.37Mbps$ . For example, in Fig.4.1, only the node with deep blue color can generate CBR traffic flows and transmit data to the other

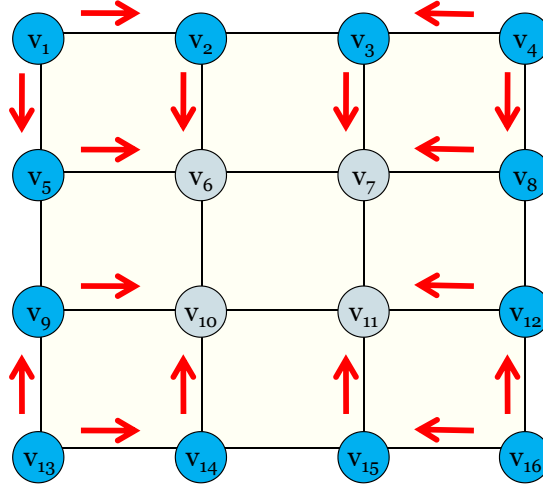


Figure 4.1: Simulation topology

sides. The flows cross at the intermediate nodes, thus we call that cross-like traffic flows. The CBR data packet size is 512 bytes. The details of simulation parameters are depicted in Table 4.1.

In the simulation, we compare the performance of classical unicast forwarding (denoted by Unicast), COPE-like protocol [5] (denoted by COPE), and OASIS protocol (denoted by OASIS). Note that, in the simulation setup, we have a little difference on the setup of parameters for COPE. In the original COPE protocol, they disable RTS/CTS/ACK mechanisms; however, we use RTS/CTS/ACK mechanism to provide more reliable transmissions in our experiment. Therefore, we say that is a COPE-like protocol.

We first investigate the impact of packet generating rate on the network throughput. The network throughput is defined as the average of total number of bits received by destinations per second in the network. The classical graph for packet generating rate vs. network throughput where IP output queue size equals to 100 is depicted in Fig. 4.2. Note that, x-

Table 4.1: Simulation parameters

Parameter type	Parameter value
Simulation duration	2000 sec
Deployment region	$2000m \times 2000m$
Transmission/collision radius	$376.782m$
Number of nodes	16
Packet generating rate per flow	$0.0371089 \sim 0.3710928 \text{Mbps}$
MAC protocol	IEEE 802.11
Propagation path-loss model	TWO-RAY
Channel bandwidth	$6 \text{Mbps}$
CBR data sessions	16
IP output queue size	100 packets
Maximal retransmission times	2

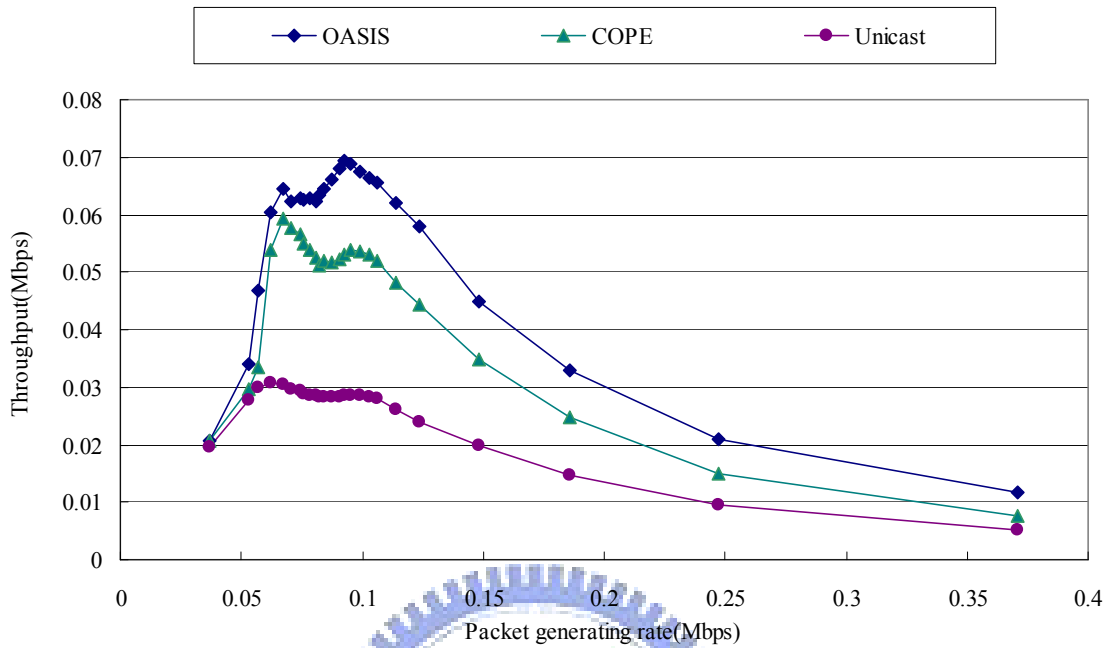


Figure 4.2: offered traffic load vs. throughput

axis and y-axis represent for the packet generating rate and throughput respectively. Three curves, diamond line, triangle line and circle line represent for the throughput for OASIS, COPE and Unicast, respectively.

As we expected, OASIS has higher throughput than COPE and Unicast. The peak throughputs are at (0.092773, 0.069408), (0.067471, 0.059225), and (0.061849, 0.030831) for OASIS, COPE, and Unicast, respectively. Meanwhile, we can see that the throughputs for OASIS and COPE are 2.25 times and 1.92 times, respectively, of the throughput for Unicast. On average, the throughputs are 0.054787, 0.044768, and 0.02551 for OASIS, COPE, and Unicast, respectively. Therefore, the throughputs for OASIS and COPE are about 2.15 times

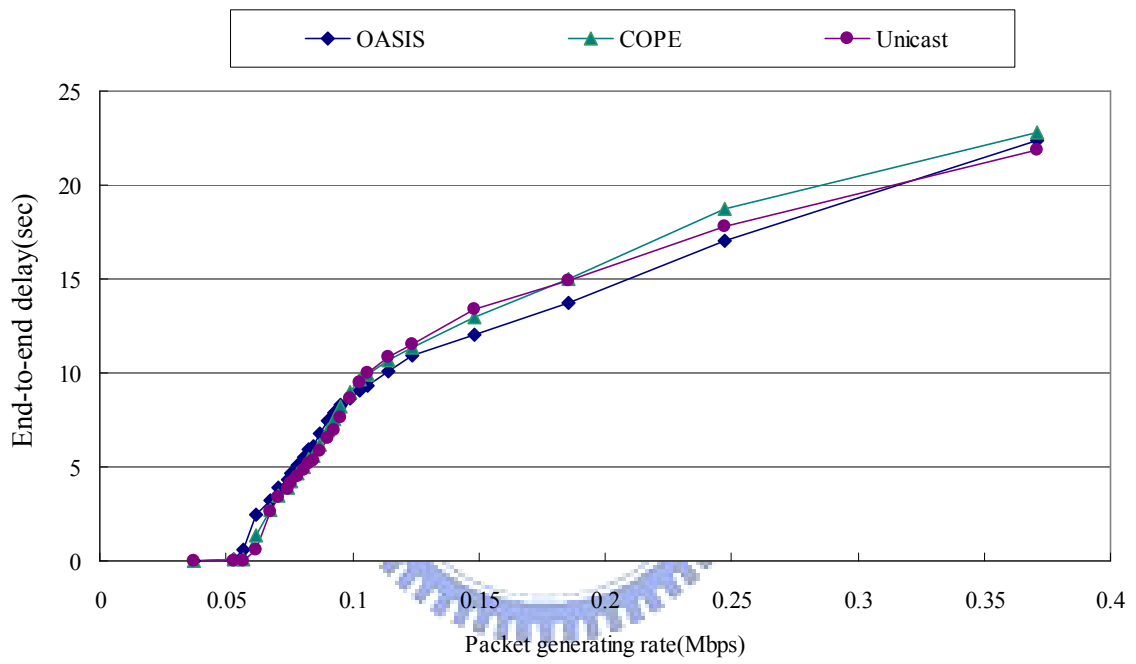


Figure 4.3: offered traffic load vs. end-to-end delay



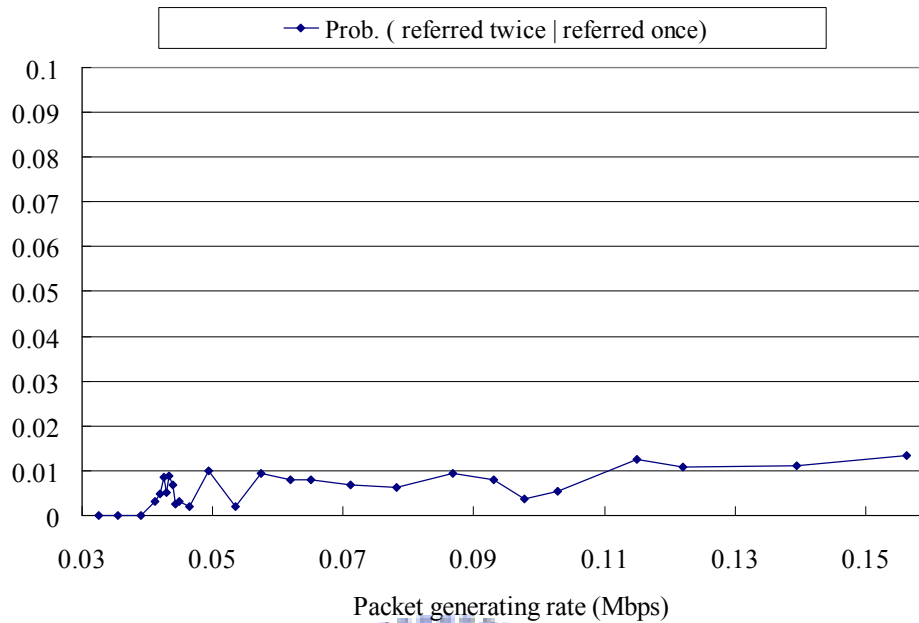


Figure 4.4: Normalized reference times under different offered traffic load

and 1.75 times than Unicast on average, respectively. In addition, the throughput for OASIS is 1.22 times than the throughput achieved by COPE on average.

The throughput trend of the three curves is reasonable. While the packet generating rate increases, the network becomes heavy. The network throughput can increase with the increase of packet generating rate if the network has the ability to afford the increasing traffic. Once the traffic becomes too heavy, the network can not consume the large amount of generated packets. The IP output queue can not buffer packets any more, and then many packets will be dropped. Moreover, the network traffic is too heavy such that many collision events occur during the packet transmissions. As a result, network throughput decreases with the increase of packet generating rate. The occurrence of peak throughput represents the ability that a network can afford the traffic. Thus, the peak throughput of both OASIS

and COPE appear in the up-right of the peek throughput of Unicast. Unicast has the worst throughput since it never utilizes broadcast property of wireless communications. In each transmission, there is at most one packet can be delivered. However, network coding schemes have the possibility to deliver more than one packets in each transmission. Moreover, the situation that OASIS outperforms COPE has two major reasons. First, it greedily forwards information that is not necessary but may be helpful for the future transmissions. This possibility was shown in the example depicted in Fig. 3.1. Second, it introduces a new scheme of information pool management that increases the utility of packets in the *infoPools*. Thus, we get a better result in the experiment.

Fig. 4.3 illustrates the packet generating rate vs. end-to-end delay. The end-to-end delay is defined as the total time experience as the time a packet generated from the source to the time this packet received by the destination. As shown in this figure, we can say that the network coding schemes have not too much overhead in the end-to-end delay but significantly increase the total system throughput.

Fig. 4.4 illustrates the packet generating rate vs. the conditional probability of packets in *infoPool* being referred twice under that are referred once. In this experiment, we collect the statistics of the number of packets in the *infoPool* which are referred once and the number of packets which are referred twice. And we calculate the conditional probability with different packet generating rate:

$$\begin{aligned} & \Pr(\text{a packet is referred twice} \mid \text{a packet is referred once}) \\ &= \frac{\text{Num of packets which are referred twice}}{\text{Num of packets which are referred once}} \end{aligned}$$

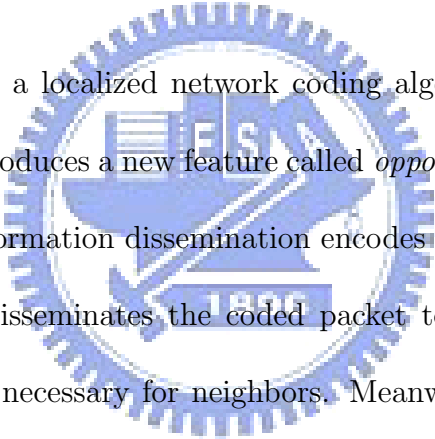
As shown in the figure, the probability of a packet in the *infoPool* being referred twice is no more than 2%. The largest value is 0.013311 while packet generating rate equal to 0.15625 Mbps. The result really makes sense as we mentioned in the previous section. And it is helpful for us to design a new management approach.

As a summary for the above simulation evaluation, the simulation results shows that the proposed OASIS that introduces opportunistic information dissemination indeed increases more throughput than COPE and the information pool management plays a significant role in network coding transmission. Moreover, the simulation results show that the network coding implementation of OASIS and COPE do not append too much overhead in the end-to-end delay.



# Chapter 5

## Conclusions



In this thesis, we proposed a localized network coding algorithm called OASIS that is an extension of COPE and introduces a new feature called *opportunistic information dissemination*. The opportunistic information dissemination encodes as many information as possible into a coded packet and disseminates the coded packet to neighbors even if some information is not immediately necessary for neighbors. Meanwhile, a simple information pool management is introduced to deal with the limitation on the information pool size. In the information pool management, a new factor of reference times is introduced. When the *infoPool* is full, we first replace the packet which has been referred. If there is no such packet, then we replace the packet which stays in *infoPool* over the specified time limit. If there are no such packets described above, we randomly choose a packet to be replaced. The simulation results show that the network throughput achieved by OASIS is 2.15 times than classical unicast forwarding and 1.22 times than COPE. The simulation results also show

that it takes a few overhead on end-to-end delay to implement OASIS. So, overall speaking, OASIS is an offerable localized network coding heuristic for wireless networks.

In this work, we consider the implementation issues of design a network coding protocol in a realistic wireless environment. However, there are still many things need to be considered when implement network coding into network protocol suit, such as more advance information pool management mechanisms, ACK mechanisms, and so on. All this are possible research topics for future works.



# Bibliography

- [1] S. Biswas and R. Morris, “Opportunistic routing in multi-hop wireless networks,” *SIGCOMM Comput. Commun. Rev.*, vol. 34, no. 1, pp. 69–74, 2004.
- [2] S. DIGGAVI, N. AL-DHAHIR, A. STAMOULIS, and A. CALDERBANK, “Great expectations: the value of spatial diversity in wireless networks,” *Proceedings of the IEEE*, vol. 92, no. 2, pp. 219–270, Feb 2004.
- [3] R. Ahlswede, N. Cai, S.-Y. R. Li, and R. W. Yeung, “Network information flow,” *IEEE Transactions on Information Theory*, vol. 46, no. 4, pp. 1204–1216, Jul 2000.
- [4] P. Chou and Y. Wu, “Network coding for the internet and wireless networks,” *IEEE Signal Processing Magazine*, vol. 24, no. 5, pp. 77–85, September 2007.
- [5] 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, June 2008.
- [6] S.-Y. R. Li, R. W. Yeung, and N. Cai, “Linear network coding,” *IEEE TRANSACTIONS ON INFORMATION THEORY*, vol. 49, no. 2, pp. 371–381, February 2003.

- [7] S. Fong and R. Yeung, “Variable-rate linear network coding,” in *Information Theory Workshop (ITW '06)*, Oct. 2006, pp. 409–412.
- [8] R. Koetter and M. Médard, “An algebraic approach to network coding,” *IEEE/ACM Transactions on Networking*, vol. 11, no. 5, pp. 782–795, Oct. 2003.
- [9] T. Ho, R. Koetter, M. Médard, D. R. Karger, and M. Effros, “The benefits of coding over routing in a randomized setting,” in *IEEE International Symposium on Information Theory (ISIT 2003)*, 29 June–4 July 2003, p. 442.
- [10] C. Gkantsidis and P. Rodriguez, “Network coding for large scale content distribution,” in *Proceedings of the 24th Annual Joint Conference of the IEEE Computer and Communications Societies (IEEE INFOCOM 2005)*, vol. 4, March 2005, pp. 2235–2245.
- [11] L. Keller, E. Drinea, and C. Fragouli, “Online broadcasting with network coding.” IEEE, Jan. 2008, pp. 1–6.
- [12] J. Kumar Sundararajan, D. Shah, and M. Medard, “Arq for network coding.” IEEE, July 2008, pp. 1651–1655.
- [13] A. Dimakis, V. Prabhakaran, and K. Ramchandran, “Decentralized erasure codes for distributed networked storage,” *Information Theory, IEEE Transactions on*, vol. 52, no. 6, pp. 2809–2816, June 2006.
- [14] D. Wang, Q. Zhang, and J. Liu, “Partial network coding: Theory and application for continuous sensor data collection,” June 2006, pp. 93–101.

- [15] T. Ho and R. Koetter, “Online incremental network coding for multiple unicasts,” in *DIMACS Working Group on Network Coding*, 26-28 January 2005.
- [16] Z. Li and B. Li, “Network coding: the case of multiple unicast sessions,” in *Proceedings of the 42nd Allerton Annual Conference on Communication, Control, and Computing*, 2004.
- [17] P. Gupta and P. Kumar, “The capacity of wireless networks,” *Information Theory, IEEE Transactions on*, vol. 46, no. 2, pp. 388–404, Mar 2000.
- [18] S. Toumpis and A. Goldsmith, “Large wireless networks under fading, mobility, and delay constraints,” *INFOCOM 2004. Twenty-third Annual Joint Conference of the IEEE Computer and Communications Societies*, vol. 1, pp. –619, March 2004.
- [19] J. Liu, D. Goeckel, and D. Towsley, “Bounds on the gain of network coding and broadcasting in wireless networks,” *INFOCOM 2007. 26th IEEE International Conference on Computer Communications. IEEE*, pp. 724–732, May 2007.
- [20] K. Lu, S. Fu, and Y. Qian, “Capacity of random wireless networks: Impact of physical-layer network coding,” *Communications, 2008. ICC '08. IEEE International Conference on*, pp. 3903–3907, May 2008.
- [21] P. Samuel David and A. Kumar, “Network coding for tcp throughput enhancement over a multi-hop wireless network,” *Communication Systems Software and Middleware and Workshops, 2008. COMSWARE 2008. 3rd International Conference on*, pp. 224–233, Jan. 2008.



- [22] C. Fragouli, J. Widmer, and J.-Y. Le Boudec, “Efficient broadcasting using network coding,” *Networking, IEEE/ACM Transactions on*, vol. 16, no. 2, pp. 450–463, April 2008.
- [23] C. Fragouli, D. Katabi, A. Markopoulou, M. Medard, and H. Rahul, “Wireless network coding: Opportunities & challenges,” Oct. 2007, pp. 1–8.
- [24] S. Chachulski, M. Jennings, S. Katti, and D. Katabi, “Trading structure for randomness in wireless opportunistic routing,” in *SIGCOMM '07: Proceedings of the 2007 conference on Applications, technologies, architectures, and protocols for computer communications*. New York, NY, USA: ACM, 2007, pp. 169–180.
- [25] A. Miu, H. Balakrishnan, and C. E. Koksal, “Improving loss resilience with multi-radio diversity in wireless networks,” in *MobiCom '05: Proceedings of the 11th annual international conference on Mobile computing and networking*. New York, NY, USA: ACM, 2005, pp. 16–30.
- [26] C. Reis, R. Mahajan, M. Rodrig, D. Wetherall, and J. Zahorjan, “Measurement-based models of delivery and interference in static wireless networks,” in *SIGCOMM '06: Proceedings of the 2006 conference on Applications, technologies, architectures, and protocols for computer communications*. New York, NY, USA: ACM, 2006, pp. 51–62.
- [27] D. Lun, N. Ratnakar, R. Koetter, M. Médard, E. Ahmed, and H. Lee, “Achieving minimum-cost multicast: a decentralized approach based on network coding,” in *Pro-*

*ceedings of the 24th Annual Joint Conference of the IEEE Computer and Communications Societies (IEEE INFOCOM 2005)*, vol. 3, March 2005, pp. 1607–1617.

- [28] X. Zeng, R. Bagrodia, and M. Gerla, “Glomosim: a library for parallel simulation of large-scale wireless networks,” in *in Workshop on Parallel and Distributed Simulation*, 1998, pp. 154–161.

