

國立交通大學

網路工程研究所

碩 士 論 文

利用網路編碼增進網格網路的效能

Performance Improvement by Network Coding on Grid
Network

研 究 生：甘効原

指導教授：易志偉 教授

中 華 民 國 九 十 七 年 十 一 月

利用網路編碼增進網格網路的效能
Performance Improvement by Network Coding on Grid Network

研究生：甘効原

Student : Hsiao-Yuan Kan

指導教授：易志偉

Advisor : 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

November 2008

Hsinchu, Taiwan, Republic of China

中華民國九十七年十一月

利用網路編碼增進網格網路的效能

學生：甘効原

指導教授：易志偉

網路工程研究所

國立交通大學

摘要

近年來無線隨意繞徑協定開始利用地理資訊在大型的無線隨意網路中進行繞徑。其中一個廣為人知的地理資訊繞徑協定—網格繞徑協定—目前已有許多的應用架構於此之上。然而，隨著科技的進步和裝置價格越來越便宜，使得一個裝置將同時負責越來越多的工作，以及無線隨意網路的裝置密度會越來越高，如此會造成無線網格網路負載將越來越重，因此如何增進無線網格網路傳輸效能將會是一個重要的議題。有人提出一個技術—網路編碼，網路編碼主要用來增進網路傳輸效能、網路頻寬等。由於網格網路有多點傳播和路徑分岔的特性，所以網路編碼適用於網格網路之中。

在此篇論文中，我們利用網路編碼演算法—XOR 編碼演算法，建構於貪婪網格繞徑協定上面—來增進網格網路的傳輸效能。我們假設網格網路中，每個節點在每個時間槽會以某個機率產生封包，並以隨機媒體存取控制(Random MAC)的方式單點傳播封包。每個封包的容量一樣而且來源點和目的點都是均勻分布在網格網路中。在網格網路達到穩定狀態的模擬實驗中(執行 500000 時間槽)，XOR 編碼演算法可以增進網格網路吞吐量 74.0%，點對點延遲 55.7%。當節點緩衝存儲器儲存容量小於 30 個封包量時，儲存容量對網格網路的吞吐量有明顯的改變，但超過 30 個封包後，增加儲存容量對吞吐量的影響即不顯著。在經過一系列的各種

不同的網路環境實驗模擬和分析後，我們可以得到以下結論：在網格網路中使用網路編碼傳遞封包時，不但可以增加網路吞吐量並且可以降低點對點延遲。在未來我們將會對這作相關的理論分析，並把網路編碼應用到三角形和六角形等網路拓撲，以了解網路編碼和網路拓撲的關係。

關鍵字

網格網路，網路編碼，XOR 編碼，效能增進。



Performance Improvement by Network Coding on Grid Networks

Student: Hsiao-Yuan Kan

Advisor: Dr. Chih-Wei Yi

Institute of Network Engineering

National Chiao Tung University

Abstract

In the recent years, geographic information is exploited for routing in large scale ad hoc networks. One of most famous geographic-based routing protocol, GRID routing, is applied to many applications nowadays. Moreover, with the advance of information technology and cost down in wireless devices, each device can take many more tasks and the device density becomes even denser than before. As, the traffic load becomes heavier and heavier, and how to efficiently utilize the limited wireless resource becomes the most important issue. Recently, a novel technique, network coding, is proposed to improve network throughput and bandwidth. Due to the broadcast nature and path diversity of wireless networks, network coding is applicable to grid networks.

In this thesis, to improve the throughput of grid network, a network coding algorithm, XOR encoding, is exploited in the grid networks with greedy grid routing. We assume that each node unicasts packets with packet generating rate and transmits

packets followed Random MAC model in the grid network. Each packet has the same size, and the source and destination pairs are randomly chosen and distributed uniformly over the grid network. Under the stable state run for 500000 time slots, the XOR encoding algorithm improves network throughput and end-to-end delay about 74.0% and 55.7%, respectively. Meanwhile, the network throughput increases when the buffer size is smaller than 30 and slows down when the buffer size exceeds 30. As a summary of the series simulation analyses, it can be concluded that the network coding can both increase network throughput and reduce end-to-end delay for the grid network under different network environments. In the future, we will do some theoretical analysis of grid networks with network coding and exploits network coding to delauney network, hexagonal network, etc. to know the relationship between network topologies and network coding.



Keywords

Grid network, network coding, XOR encoding, performance improvement.

Contents

1	Introduction	1
1.1	Background of Wireless Ad Hoc Network	1
1.2	Motivation of Thesis and Simulation Environment	2
1.3	Main Results and Organization	6
2	Related Work	8
2.1	Background of Grid Network	8
2.2	Traffic Load of Grid Cell	13
2.3	Background of Network Coding	14
3	Network Coding Algorithm and Implementation	18
3.1	<i>XOR</i> Coding Algorithm	18
3.2	Network Model and Assumptions	24
3.3	Network Coding Implementation	26
4	Simulation Analysis	31
4.1	Simulation Settings and Scenarios	31
4.2	Distribution of Traffic Load of Grid Cell	33
4.3	The Impact of Packet Generating Rate	36
4.3.1	Throughput	37

4.3.2	Packet Loss	39
4.3.3	End-to-End Delay	42
4.4	The Impact of Buffer Size	44
4.4.1	Packet Loss	45
4.4.2	Throughput	47
5	Conclusion	50



List of Figures

1.1	Alice and Bob scenario	4
2.1	Neighboring cells of cell 1 and neighbor nodes of node a	10
2.2	(a) Routing table utilizes grid IDs to route packets and this offers more resilient route maintenance. (b) GRID routing protocol has highly resilient for node mobility, because a new leader takes over the relaying jobs when the current one wanders out of the grid.	11
2.3	Traffic distribution over celss for 60×60 grids	14
2.4	Butterfly example	15
3.1	Let the primitive packets to be encoded maximal numbers in each transmission.	23
3.2	Flow chart of transmitter side for encoding.	27
3.3	Flow chart of receiver for decoding.	29
3.4	Flow chart of transmitter side for ACK.	30
4.1	Distribution of number of times for transmission using <i>Unicast</i>	34
4.2	Distribution of number of times for transmission using <i>XOR</i> coding algorithm.	35
4.3	Packet generating rate v.s. Throughput	37
4.4	Packet generating rate v.s. Packet loss	39
4.5	Packet generating rate v.s. Packet loss caused by insufficient buffer size	40

4.6	Packet generating rate v.s. End-to-End delay	42
4.7	Buffer size v.s. Packet loss (at arrival 0.0333 packet/time slot)	45
4.8	Buffer size v.s. Packet loss (at arrival 0.0167 packet/time slot)	45
4.9	Buffer size v.s. Packet loss (at arrival 0.0083 packet/time slot).	46
4.10	Buffer size v.s. Throughput (with packet generating rate 0.0333)	47
4.11	Buffer size v.s. Throughput (with packet generating rate 0.0167)	48
4.12	Buffer size v.s. Throughput (with packet generating rate 0.0083)	49



List of Tables

4.1	Simulation parameters with 500000 primitive packets.	34
4.2	Simulation parameters in 500000 time slot.	36
4.3	Average throughput and ratio of throughput in <i>XOR</i> coding algorithm to that in <i>Unicast</i> .(Saturation point indicates packet-generating rate 0.014286 at the maximal throughput in the curve of the <i>XOR</i> coding algorithm.)	37
4.4	Average number of primitive packets in the buffer. (Saturation point is maximal throughput in the curve of the <i>XOR</i> coding algorithm at packet generating rate 0.014286.)	37
4.5	Ratio of packet loss caused by buffer overflow to total packet loss.	40
4.6	Average end-to-end delay and ratio of that in <i>XOR</i> coding algorithm to that in <i>Unicast</i> . (Saturation point indicates packet generating rate 0.025 at the maximal end-to-end delay in the curve of the <i>XOR</i> coding algorithm.)	42
4.7	Average number of primitive packets in the buffer. (Saturation point indicates packet generating rate 0.025 at the maximal end-to-end delay in the curve of the <i>XOR</i> coding algorithm.)	42
4.8	Simulation parameters in 500000 time slot.	44
4.9	Decrease of packet loss by increase of per buffer size.	46
4.10	Ratio of throughput increase in buffer size 30 (or 80) to that in buffer size 10 (or 30).	49

Chapter 1

Introduction

1.1 Background of Wireless Ad Hoc Network

With the advance of the communication technology, the Internet is popular around the world and plays a more and more important role in our daily lives. In the traditional network, computers communicate with each other using a wire link through routers or end systems, and must be located in fixed positions due to lack of mobility. In the past, when people wanted to access the Internet, they had to find fixed locations equipped with computers, but it was considered very inconvenient. Hence, in recent years, researchers invented wireless devices and developed wireless network, which can be generally categorized as infrastructure wireless network and ad hoc network, namely independent basic service set network.

Three components involve in the infrastructure wireless network: access point, wireless distribution system and mobile node. Mobile nodes access the Internet with wireless medium through access points which act like bridges, and an access point can only serve mobile nodes, which are its one-hop neighbors. Thus, devices within the service range of an access point form a small wireless network, namely a basic service set with a unique ID. Wireless distribution systems are utilized to connect access points, wireless distribution systems or traditional

networks, so that a mobile node can communicate with other mobile nodes in other basic service sets or computers in the Internet through wireless distribution systems. In other words, wireless distribution systems can extend a basic service set to many other systems, and the so-called extended service set, also has a unique ID. However, the infrastructure network has a major drawback, which is its poor survivability. Once an access point is destroyed in any artificial or natural manner, such as by typhoon, earthquake, power failure and so on, the mobile nodes would be unable to communicate with others, and consequently the basic service set served by it would be paralyzed. In view of this, an ad hoc network does not require a base station, in other words, it does not contain any component acting as an access point. Thus, it has more survivability than the infrastructure network.

An ad hoc network, contrary to a infrastructure network, is a self-organized network without the aid of any infrastructure. Ad hoc nodes, different from the devices in the traditional wire network or the infrastructure wireless network, have mobility and act both as routers and end systems, and the topology changes frequently due to the mobility of ad hoc nodes. Two ad hoc nodes can directly communicate with each other if they are within the transmission range of each other, but they can still communicate with the aid of intermediate nodes while they are not. Moreover, an ad hoc network can be formed by itself and has good survivability as compared to a infrastructure wireless network. Because of the frequent changes in the network topology due to the movement of ad hoc nodes, the strategy of routing protocols in the ad hoc network, unlike that of the wire network and the infrastructure wireless network, has a great impact on the performance and is extensively studied by the researchers in recent years.

1.2 Motivation of Thesis and Simulation Environment

In recent years, there are many proposed routing protocols, which can be categorized into topology-based and geographic-based routing protocols. Among the literatures [1,2], Li and

Jain showed that topology-based routing protocols are not scalable because of the enormous overhead for maintenance. To develop a scalable routing protocol, geographic-based routing protocols are proposed, such as GRID routing protocol (GRID) [3], Grid Location Service [1], and Geographic Perimeter Stateless Routing protocol (GPSR) [4]. Although GPS and some other positioning systems are becoming popular in recent years, different positioning systems have different positioning accuracy, and therefore geographic routing protocols requiring highly accurate geographic information are not practical. GRID routing protocol, one of the famous geographic routing protocols, has much tolerance to inaccurate location information and is easy to implement with hierarchical architecture. Hence, on the practical side, GRID routing protocol has high potential to be implemented in the practical environment in the future. With more applications widely used over grid networks and lower prices for wireless devices, the network density would also increase. Moreover, the traffic load would also increase in a heavy situation since there are a large number of transmissions going on at the same time. Therefore, how to efficiently transmit data over wireless network with limited wireless resources becomes a critical issue.

In recent years, a novel technique, network coding, is proposed to improve network throughput by mixing different flow packets into a single packet. To utilize the network coding over the underlying networks, it is required that the network possesses two characteristics: multicast and path diversity. Because wireless devices transmit packets through broadcasting, and each node acts as an intermediate node for some data flows, it is clear that the wireless network meets the two above-mentioned requirements.

To provide a simple explanation of how network coding improves throughput, it is illustrated with the well-known Alice-and-Bob scenario [5] in Figure 1.1. In this figure, three network devices, the relay station, Alice and Bob are involved in the process. The relay station is a device necessary for Alice and Bob to exchange their information. Suppose that Alice wants to transmit one packet P_1 to Bob, and at the same time Bob wants to trans-

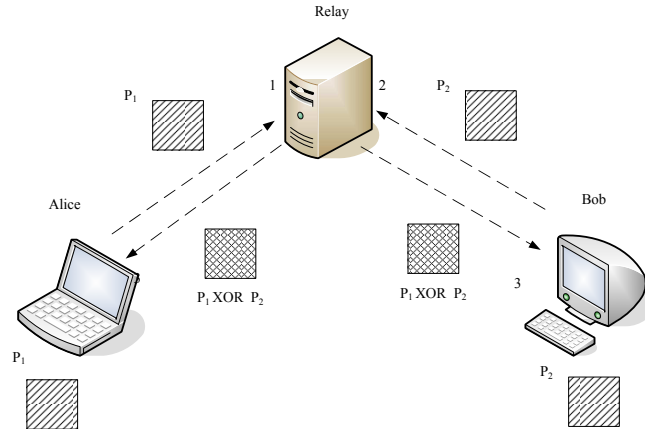


Figure 1.1: Alice and Bob scenario

mit one packet P_2 to Alice. One possible transmission schedule without applying network coding is as follows. First of all, Alice transmits P_1 to the relay station, which relays P_1 to Bob. The number of transmission required for Alice to transmit packets P_1 to Bob is 2. On the other hand, while transmitting P_2 to Alice, Bob has a transmission schedule similar to Alice'. Therefore, if Alice and Bob exchange their own one packet with each other, the total number of transmission for Alice and Bob would be 4. However, if network coding is adopted in this scenario, Alice would transmit packet P_1 to the relay station in the first place. At the same time, Bob also transmits packet P_2 to the relay station. After receiving packets P_1 and P_2 , the relay station encodes packets P_1 and P_2 with XOR operation into an coded packet $P_1 \oplus P_2$ and broadcasts it to its one hop neighbors, Alice and Bob, instead of simply unicasting packets P_1 and P_2 . Alice and Bob will then receive the coded packet $P_1 \oplus P_2$. Because Alice has packet P_1 and Bob has packet P_2 , they can extract their desired packet by calculating $(P_1 \oplus P_2) \oplus P_1$ or $(P_1 \oplus P_2) \oplus P_2$. Therefore, the total number of transmission is 3, and the total latency for exchanging packets is reduced by $\frac{1}{4}$. In this scenario, the total number of transmission with network coding is less than that with unicasting. In other words, a network device using network coding can transmit more than one content

of information to its one hop neighbors during a single transmission. However, it can only transmit a single content of information during a single transmission while using traditional unicasting. Therefore, in general, network coding can improve network throughput.

Katti et al. [5] proposed a localized wireless network coding heuristic, called COPE, that adopts the idea illustrated in the previous example. Each node in COPE is informed of knowledge of its one hop neighbors without knowing the global topology of the entire network. Each node obtains knowledge of one hop neighbors by overhearing packets transmitted by its one hop neighbor or piggyback information in the successive transmissions. To prevent ambiguity, in the following discussion the original packets before encoding are called primitive packets, and packets after encoding are called coded packets. While a node is ready to transmit a primitive packet over the wireless channel, it encodes as many primitive packets in its buffer as possible into one coded packet by XOR operation and broadcasts the coded packet to their nexthops if all these primitive packets of their nexthops are capable of decoding their demand packets. A node overhears a coded packet and stores it into its information pool. After receiving a packet, a node tries to extract its demand packet by decoding it with packets stored in its information pool and buffer. Thus, packets in the information pool are the fundamental knowledge for decoding the incoming packets. Because there is high potential to implement ad hoc networks with grid networks, which routes packets by GRID routing protocol, we can utilize the network coding technique as a powerful tool to efficiently make use of the wireless resource in the real environment as the network traffic load becomes heavier and heavier in the future. Furthermore, we can investigate the impact of network coding on the performance of the network throughput. The mechanism of the network coding used in this work is referred to as COPE [5].

The environment of simulation used in this work is described as follows. Each ad hoc node in the grid network operates in the synchronized TDMA-Similar mode and performs slotted random MAC, called Rand-MAC in this thesis. For easy simulation analysis, each

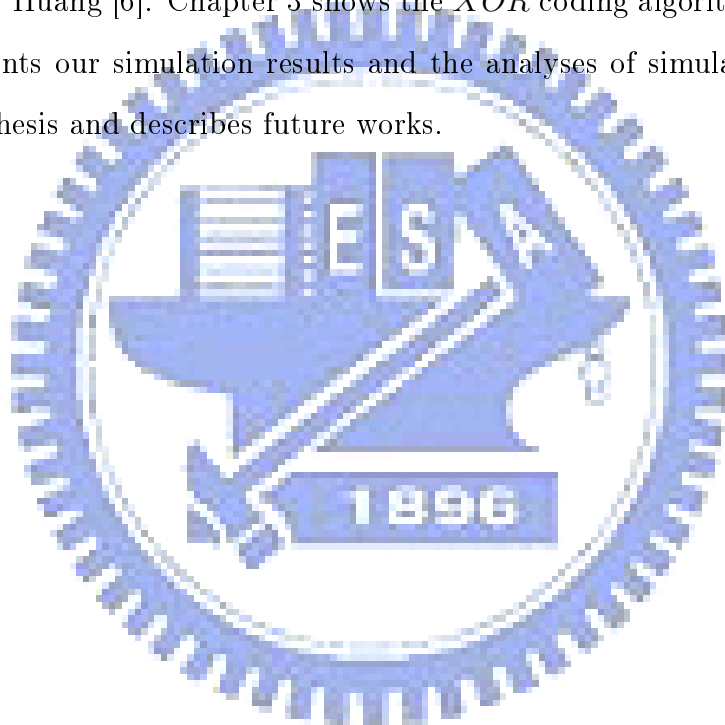
grid cell contains only one ad hoc node, which works all the time without crash. Nodes located in neighboring cells can communicate with each other, and thus the deliverability of each node is asymptotic and almost surely guaranteed. Moreover, the bandwidth of each link in the grid networks is the same, so it is assumed that the payload of each packet is one unit. The grid network model has collisions and retransmissions and utilizes greedy grid unicast routing, called *Unicast* in this thesis. COPE [5] suggested that packet knowledge can be collected by piggybacking information in successive transmission and assumed that there exists such mechanisms without too much overhead. The same assumption is adopted in this work. Actually, each node needs to know whether previous transmissions are successfully received and decoded by its neighbors and which packets has been overheard by its one hop neighbors. These are important implementation issues. However, in this work we do not really discuss and handle related issues, which will be left to our future works.

1.3 Main Results and Organization

In their previous work, Huang et al. [6] derived the theoretical value and the distribution of traffic load in each grid cell, and found that the traffic load distribution of the grid networks resembles a hill under the ideal network environment, i.e. without collisions and retransmissions. The hot spot occurs at the center of the deployment region. Simulation analyses and evaluations are conducted in this work. The evaluations are focused on the throughput improvement by the network coding, namely the *XOR* coding algorithm as compared to the greedy grid unicast routing, called *Unicast* in this work. According to our simulation, the information exchange using *XOR* coding algorithm can improve the throughput of the grid network efficiently as compared with *Unicast*. For each grid cell, the traffic load distribution is similar to that of the theoretical value, except for the less traffic load due to collision and packet loss. Our simulations show the throughput using the *XOR* coding algorithm is better than that using *Unicast*. In the stable state of grid network

(During 500000 time slots, the 10×10 grid network achieves a stable state.), the average improvement rate of throughput and end-to-end delay using the *XOR* coding algorithm is about 74.0% and 55.7%. Thus, our simulation shows that the *XOR* coding algorithm can improve the throughput of the grid network and reduce end-to-end delay.

The rest of this thesis will be organized as follows. Chapter 2 introduces related works, including GRID routing protocol, Huang's work, which presents the theoretical value of traffic load of grid network, as well as related works on network coding. Our simulation is based on GRID routing protocol [3] which is compared to the preliminary traffic load analysis done by Huang [6]. Chapter 3 shows the *XOR* coding algorithm used in this thesis. Chapter 4 presents our simulation results and the analyses of simulation data. Chapter 5 concludes this thesis and describes future works.



Chapter 2

Related Work

2.1 Background of Grid Network

Ad hoc nodes act both as routers and end systems, and an ad hoc network can be organized by itself. The topology of the traditional wire network does not change, because devices are fixed to specific locations. In the infrastructure wireless network, there is a base station, namely access point, which serves its one-hop neighbor mobile nodes in its basic service set. If mobile nodes roam inside the radio range of its access point, they can access the network; otherwise, they are disconnected from the network. Hence, the topology of the infrastructure wireless network of mobile nodes can change, but its access point always stays in the same place. In addition, mobile nodes communicate with others in the network directly through its access point, but not through any mobile nodes in the same basic service set. However, each ad hoc node has mobility and communicates with others through many intermediate ones. Therefore, in the ad hoc network, unlike the traditional wire network and the infrastructure wireless network, information transmitted by an ad hoc node may be relayed through many intermediate ones, and the entire network topology can change while passing on information. Thus, routing protocols utilized for the traditional wire network and infrastructure wireless

network are not applicable for the ad hoc network.

In recent years, many routing protocols of the ad hoc network are proposed, and can be generally categorized into topology-based and geographic-based routing protocols. In the topology-based routing protocols, such as Dynamic Destination-Sequenced Distance-Vector Routing (DSDV) [7], Optimized Link State Routing (OLSR) [8], Topology Broadcast based on Reverse-Path Forwarding (TBRPF) [9, 10], Dynamic Source Routing (DSR) [11] and Ad hoc On Demand Distance Vector (AODV) [12], the logical link information in the network is utilized to determine the packet forwarding route. Recent researches [1, 2] showed that these topology-based routing protocols without geographical information are not scalable due to serious maintenance overhead. To decrease the maintenance overhead of topology-based routing protocols, researchers proposed scalable geographic-based routing protocols, such as GRID routing protocol [3], Grid Location Service [1], and Geographic Perimeter Stateless Routing protocol (GPSR) [4]. Although many researchers have made great progress on positioning accuracy of global positioning systems (GPS) and other positioning systems in recent years, nowadays location errors still exist in all location estimating engines. Moreover, in the realistic point of view, some geographic routing protocols relying on accurate positioning devices to route packets cannot work well and be implemented in the realistic environment. However, in the ad hoc network, there is one famous routing protocol, GRID routing protocol, which has much location error tolerance and is scalable to be implemented in the practical environment due to the hierarchical architecture. Today many applications, such as agriculture monitoring in paddy fields, tracking of an object, gathering data and others, are based on this protocol. In short, GRID routing protocol has high potential to be applied to many spheres in the realistic environment in the future.

GRID routing protocol, which is reactive, geographic-based and fully location-aware, is proposed by Tseng et al. [3]. It exploited location information about route discovery, packet relay, and route maintenance, and assumed that each node equipped with a GPS device

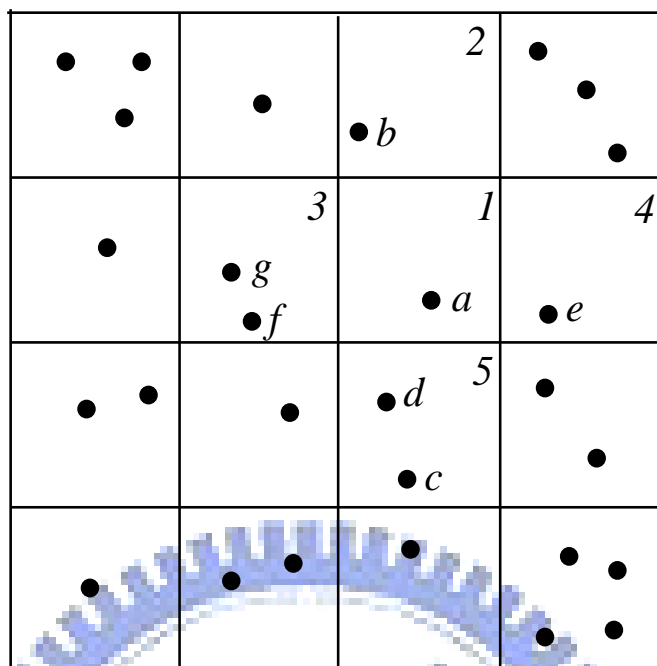


Figure 2.1: Neighboring cells of cell 1 and neighbor nodes of node a .

knows its location in the network. GRID tessellates the geographic area of the MANET into the 2D logic equal square areas called grid cells, and assigns a tuple (x, y) to each grid cell as its xy-coordinate. Two grid cells are called neighboring grids if they share a common edge, and two nodes are called neighboring nodes if they are located in neighboring cells and are within each other's transmission range. For instance, in Figure 2.1, the grid cells 2, 3, 4, 5 are neighbor cells of the grid cell 1. In each grid cell, the node closest to the geographic center of that grid cell, is elected as the leader of a grid cell among the other ad hoc nodes. A leader acts as a gateway and takes responsibility for routing packets in a grid-by-grid manner, while other non-leader ad hoc nodes do not. The leader of a grid cell takes responsibility for many tasks, such as forwarding route requests to neighboring grids, propagating data packets to neighboring grids, and maintaining routes which pass the grid in which it resides. Therefore, for power consumption, it would be better to choose only one node as the leader rather than to choose more than one. While a node moves closer to the geographic center of a grid cell

than its current leader, it would be elected the new leader. As a consequence, many leaders may reside in a grid cell at the same time. In this case, when a leader gets a packet from another leader closer to the geographic center of the grid cell, it turns into a nonleader and stops forwarding any packets. While a leader roams out of its grid cell, a new leader would be elected and the old leader would pass its routing table to the new one by broadcasting. If an active leader exists in a grid cell, the grid cell is considered alive; otherwise, it would be considered crashed, and another ad hoc node in this grid cell will be elected as the new leader.

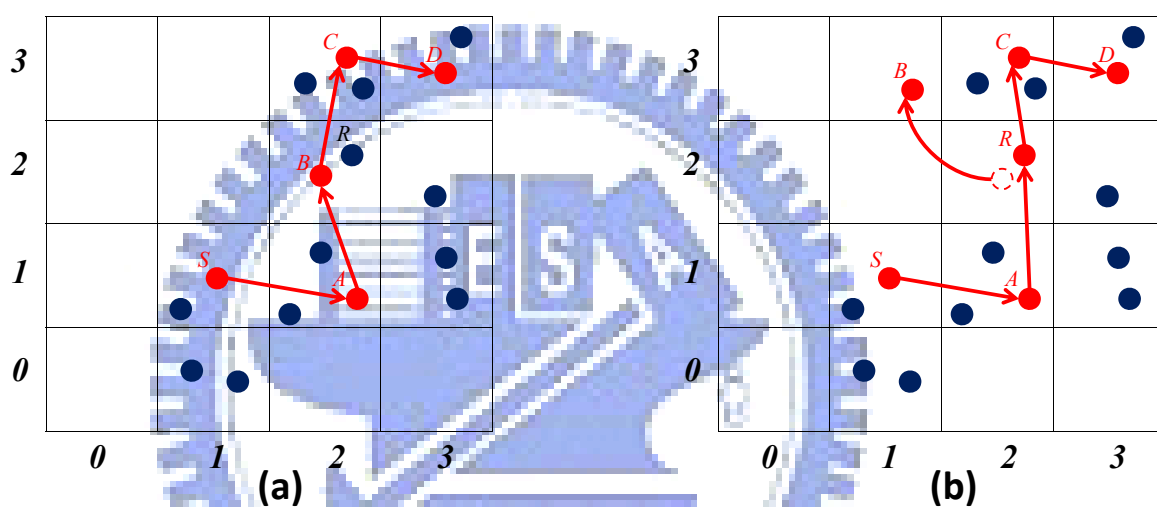


Figure 2.2: (a) Routing table utilizes grid IDs to route packets and this offers more resilient route maintenance. (b) GRID routing protocol has highly resilient for node mobility, because a new leader takes over the relaying jobs when the current one wanders out of the grid.

Three tasks, forwarding route requests to neighboring grids, propagating data packets to neighboring grids, and maintaining routes which pass through the grid it resides, are carried out by a leader. If leaders and nonleaders can communicate with neighboring nodes, it is possible that the leader and nonleaders send out a large number of redundant request packets during route discovery for one source node. This situation causes poor routing performances and wastes network resources. To avoid such shortcomings, only leaders are

allowed to communicate with neighboring nodes. With this filtering mechanism, the GRID routing protocol is insensitive to high host density and can efficiently route packets with less network resources in high density environments. Because only the leaders can communicate with neighboring nodes, they are responsible for route discovery. When a source node is ready to transmit a packet, first of all, a request packet is transmitted to its leader, and the leader then relays this request to neighboring nodes in an on-demand manner. A route discovery from the leader to a destination in the GRID routing protocol works in a similar way as the AODV mechanism. Finally, a destination receives a request packet from its leader and replies to the source following a corresponding reverse routing path. A routing table records the next grid ID which heads to the destination, rather than a host ID, because they offer stronger, more resilient route maintenance. For example, in Figure 2.2(a), node B registers grid $(2,2)$ instead of the address of node B . This mechanism renders routing highly resilient to node mobility. While node B roams out of grid $(2,2)$, a newly elected leader, node R , takes over the relaying job. Long routing paths are often lost due to node's mobility, however routing packets following grid ID overcome this drawback. Thus, GRID routing protocol can sustain longer than other protocols, which are less vulnerable to node mobility in the long route.

Because GRID partitions the network into several grid cells and routes packets in a grid-by-grid manner, a network using GRID routing protocol is called a grid network. To facilitate the analysis, we confine the receiver to the ad hoc nodes in the neighbor cells of the sending node. In other words, the sender in the grid cell can only communicate in four directions. In Figure 2.1, the ad hoc node a can communicate with the ad hoc nodes b, c, d, e, f, g . In order for the deliverability to be asymptotic and almost surely guaranteed, we assume that the transmission radius of the ad hoc nodes in the network is $\sqrt{5}$ times of the grid size or larger for every node to reach any node in neighboring cells [6]. Hence, a node in a grid cell can communicate directly with a node in any of the four neighbor cells. In this thesis, our

simulation on the grid network are aimed at facilitating future analyses.

2.2 Traffic Load of Grid Cell

A previous work [6], has derived the traffic load sent or relayed by any particular cell in the deployment region by applying grid routing. In the network model, nodes are represented by a Poisson random point process with mean n over the unit-area square region D . The deployment region D is tessellated into N^2 equal sized square cells. Each cell is given a grid coordinate (i, j) , where $1 \leq i, j \leq N$. The routing distance between two nodes is measured by $L1$ grid distance, also known as the Manhattan distance. In other words, if node u is in the cell (i_u, j_u) and node v is in the cell (i_v, j_v) , the routing distance between them is given by $|i_u - i_v| + |j_u - j_v|$. It is assumed that each node knows the cell in which it is located by utilizing geometric information. Each node has the same probability of being a sender and every source node has equal probability of choosing any other nodes as destinations. The network has no packet collisions and retransmissions. Let $(i_s, j_s, i_0, j_0, i_d, j_d)$ denote the percentage of traffic going from cell (i_s, j_s) through cell (i_0, j_0) to cell (i_d, j_d) , then

$$f(i_s, j_s, i_0, j_0, i_d, j_d) = \frac{\binom{|i_0 - i_s| + |j_0 - j_s|}{|i_0 - i_s|} \binom{|i_d - i_0| + |j_d - j_0|}{|i_0 - i_d|}}{\binom{|i_d - i_s| + |j_d - j_s|}{|i_d - i_s|}}$$

, where (i_s, j_s) is the cell containing the source node and (i_d, j_d) is the cell containing the destination node. The total traffic flow in the network is one with a random traffic pattern. It shows that the traffic loads of grid cell (i_0, j_0) is

$$f(i_0, j_0) = \frac{2}{N^4} \left(\sum_{i_s=1}^{i_0} \sum_{j_s=1}^{j_0} \sum_{i_d=i_0}^N \sum_{j_d=j_0}^N + \sum_{i_s=i_0}^N \sum_{j_s=1}^{j_0} \sum_{i_d=1}^{i_0} \sum_{j_d=j_0}^N \right) f(i_s, j_s, i_0, j_0, i_d, j_d) \\ - \frac{2}{N^4} (i_0(N - i_0 + 1) + j_0(N - j_0 + 1)) + \frac{1}{N^4}$$

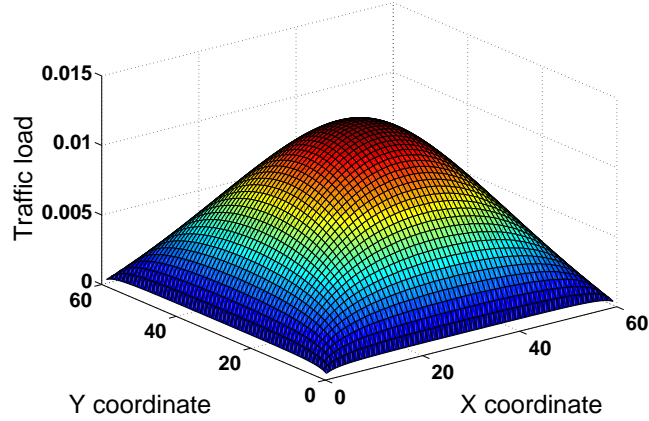


Figure 2.3: Traffic distribution over cells for 60×60 grids

, where $f(i_0, j_0)$ is the traffic load of a cell (i_0, j_0) . Hence, the hot spot occurs at the center of the deployment region, and Figure 2.3 shows the traffic load distribution over 60×60 grid cells.

2.3 Background of Network Coding

As mentioned in the previous section, the grid network is practical, scalable, simple implemented, and will be applied to many fields in the future. In general, most grid network applications, such as agriculture monitoring in paddy fields, tracking of an object and others, are served by multi-hops communication. Moreover, the advancement of technology has led to great progress in computers, and as more kinds of network applications are invented in the upcoming years, the computers can not only become cheaper but would also be able to take up many more tasks than in the past. The grid network density and the utility rate of network communications can be increased enormously year by year. Since the grid network throughput and capacity directly affect the efficiency in services involving grid network applications, the improvement of both is a critical issue. Fortunately, a novel and powerful tool, network coding, is proposed by Ahlswede et al. [13] in recent years to improve network

throughput and capacity, which seems to have provided a solution to the above-mentioned problem.

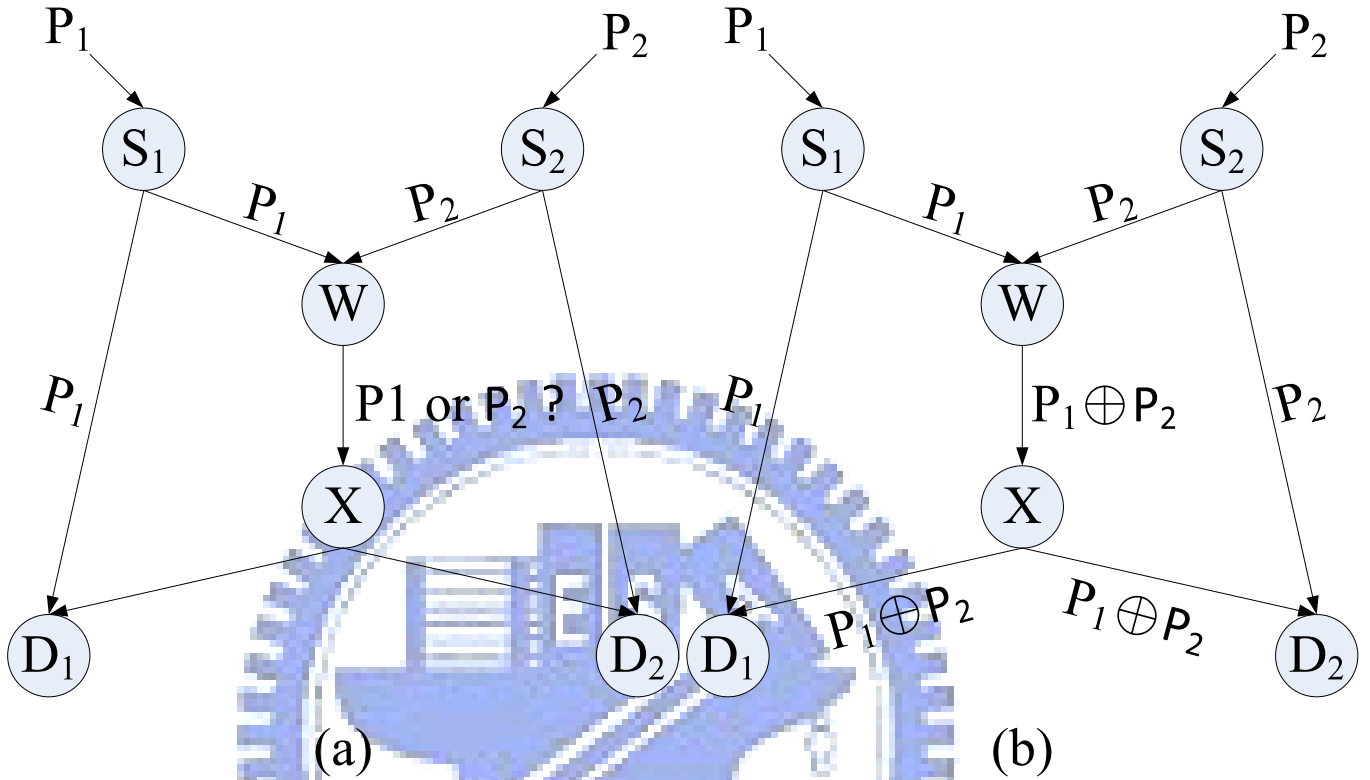


Figure 2.4: Butterfly example

The butterfly network is a well-known example [13] for illustrating the idea underlying network coding. In Figures 2.4(a) and 2.4(b), it is assumed that the capacity of each link is one message stream during one unit of time, and the middle routers W and X only forward received message stream. S_1 and S_2 represent two source nodes, which prepare for transmission message stream of both P_1 and P_2 to both destinations D_1 and D_2 . In Figure 2.4(a), in the network without network coding, it is easy to predict that the bottleneck will happen at the middle routers W and X , because the middle routers W and X can only relay one message stream P_1 or P_2 during one unit of time. Hence, it leads to the conclusion that the network throughput without network coding is $\frac{3}{2}$ message stream per unit of time. Figure

2.4(b) describes the network with network coding. When the middle router W receives the message streams of both P_1 and P_2 , it combines them into $P_1 \oplus P_2$ by mixing two message of bits and forwards it to the destinations D_1 and D_2 . The destinations D_1 and D_2 receive the message stream of both P_1 and P_2 by extracting its message stream from $P_1 \oplus P_2$ using one of received message stream of P_1 and P_2 . Thus, the network throughput with network coding is 2 message stream per unit of time, and is better than that without network coding.

Network coding was first proposed in the pioneering work by Ahlswede et al. [13] in which they showed that multicast capacity can be increased by properly mixing information from different sources at intermediate nodes. Following Ahlswede's work, a large number of works are focused on coding packets based on network topology to improve network capacity. Li et al. [14] extend the work and show that a linear coding scheme for multicast traffic that can achieve the maxflow from the source to each receiving node that is the maximum capacity bound [15]. In [16], polynomial time encoding and decoding algorithms are presented by Koetter and Me'dard and are extended to random coding by Ho et al. [17]. In the sensor network, Dimakis and Dan et al. [18, 19] exploit the network coding approach to achieve efficient data storage, collection, and dissemination. Recent researches show that network coding in specific unicast topologies can induce better throughput than in traditional transmissions [20–22]. Moreover, [5, 23] indicate that implementing network coding using *XOR* coding on the MAC layer of IEEE 802.11 can effectively improve end-to-end unicast throughput.

Most of the works described above focus on the coding algorithms. There are also some theoretic works on analyses of the impact of network coding on network throughput. In [24], the authors developed the theoretical foundation for analyses of the throughput capacity of wireless network. The main results are two-fold. First, the throughput capacity of two-dimensional arbitrary wireless networks is in the order of $O(\sqrt{n})$, where n is the number of nodes in the network; and second, for random wireless network, the throughput capacity will

scale with $O(\frac{n}{\log n})$. Since such results were published, the throughput capacity of random wireless networks has been studied extensively in the literatures [25–27]. In the random wireless network, Lu et al. [28] show that network coding on the physical-layer can improve the throughput capacity substantially, minimize delay and provide confidentiality. It also derives tighter bounds in two-dimensional random wireless networks with unicast traffic, which is uniformly distributed among all nodes. Except for adapting network coding to networks, MAC, physical layers, David et al. [29] proposed a modification for TCP of IEEE 802.11 back-off mechanism using the feedback approach. It XORs the forwarding flow of TCP data packets and reverses the flowing TCP data packets to improve the throughput. In [30], the authors show that the total number of transmissions with network coding, as compared to that without network coding, is a constant factor under the fixed network, such as the circular network and the grid network. According to related works, network coding is a powerful tool which, unquestionably, can improve the network throughput of multicasts, broadcasts, TCP mechanisms, and other network transmission mechanisms. To implement network coding, the network should include two properties: path diversity and multicast.

In wireless networks, the broadcast nature of wireless communications provides an environment for implementing network coding schemes. In other words, packets are transmitted by a transmitter over the air interface. Thus, wireless devices can receive the packets while located within the radio range of the transmitter. Moreover, while a wireless device transmits packets to the nexthops by broadcasting, it is very likely for one-hop neighbor devices to overhear packets. The broadcasting feature can satisfy both requirements for implementing network coding: path diversity and multicast. Hence, the environment of the wireless grid network is applicable for implementing network coding.

Chapter 3

Network Coding Algorithm and Implementation

3.1 *XOR* Coding Algorithm

XOR coding algorithm uses basic xor for its major operations, and has been implemented between the MAC layer and the IP layer in the practical network environment. Thus, this thesis uses the *XOR* coding algorithm referred to in the COPE [5]. COPE suggested that packet knowledge can be collected by piggybacking information in ACK and assumed that there exists such ACK mechanisms without too much implementation cost. The same assumption is adopted in this work. Each node needs to know whether previous transmissions are successfully received and decoded by its neighbors and which packets has been overheard by its one hop neighbors. These are important implementation issues. However, like COPE, we do not really discuss and handle related issues in this paper and leave them for future studies. The *XOR* coding algorithm works in the following procedure, but the details of which are depicted in PROCEDURE 1. Each node maintains two buffers: one buffer stores all the primitive packets ready for transmission and the other, the information pool, stores

undemanded coded packets and primitive packets. In general, primitive packets in the buffer are categorized into virtual queues by different nexthops, and the number of virtual queues is the same as that of one-hop neighbor nodes. The node maintains a virtual queue that are sequence of the buffers and indicates which packet is demanded by its neighbor. In other words, primitive packets in different virtual queues are transmitted to different nexthops. Due to the broadcast nature of wireless devices, one-hop neighbor nodes have many opportunities to overhear coded packets. Information pools store undemanded primitive or coded packets overheard as fundamental knowledge for decoding coded packets, and buffers store demanded primitive packets. Let $buffer$ denote the *buffer*, and $virtualQueue_w$ the virtual queue for neighbor w .

As one node has a transmission opportunity, it first dequeues a primitive packet p from its buffer $buffer$ and assigns it to coded packet $codedPacket$. The node records the nexthop of p into $nextHop$ to keep a list of all the possible receiving nodes in this coding procedure. Then, for all its one-hop neighbors w from its one-hop neighbor list $neighbor$, it extracts a primitive packet q from the virtual queue $virtualQueue_w$ and checks whether for all nodes v in $nextHop \cup w$ can successfully retrieve their demanded primitive packets after coding the content of this packet into $codedPacket$. If there exists any node that is unable to successfully retrieve its demanded primitive packet, it takes out another primitive packet q' from the virtual queue $virtualQueue_w$ to substitute for primitive packet q and re-runs the previous step, until new coded packet $codedPacket$ can be retrieved successfully by all nodes v in $nextHop \cup w$ or until there is no unverified primitive packet in the virtual queue $virtualQueue_w$. The content of a primitive packet is firstly encoded into $codedPacket$ if its receivers can both retrieve it from $codedPacket$ and the content of the retrieved primitive packet is the desired one for its receiver. If q can be encoded into $codedPacket$, $codedPacket$ is encoded with q and $nextHop$ is updated to $nextHop \cup w$. The coding procedure for which the greedy heuristic is to increase as many receiving nodes as possible.

A node retrieves information from the coded packet followed by the decoding procedure as shown in Procedure 2. Each node maintains an information pool *infoPool* that records a copy of the information of a packet it has received, sent out or retrieved from the coded packet. A node can retrieve information from the coded packet *codedPacket* with n primitive packets by XOR operation if it has exactly the same $n - 1$ primitive packets in its *infoPool*. The coding and decoding procedures of the *XOR* coding algorithm are described in the algorithm 1 and the algorithm 2. While a transmitter has a transmission opportunity and is in preparation of selecting primitive packets from its virtual queues, four conditions should be considered while deciding on what kind of primitive packets should be selected. The four conditions are as follows:

PROCEDURE 1 *XOR_encoding* (*buffer*, *neighbor*)

Require: *buffer* stores primitive packet and *neighbor* is a list of one hop neighbors.

dequeue a packet p from *buffer*.

the coded packet $codedPacket = p$

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

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

extract a primitive packet q from $virtualQueue_w$

while $\exists v \in nextHop \cup \{w\}$, v cannot decode the coded packet p && $virtualQueue_w$ has unverified primitive packet **do**

extract a primitive packet q from $virtualQueue_w$

end while

$codedPacket = codedPacket \oplus q$

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

remove q from *buffer*

end for

return *codedPacket*

PROCEDURE 2 *XOR_decoding*($v, codedPacket$)

Require: node v that wants to decode the received coded packet $codedPacket$

if $codedpacket$ is unscramble **then**

extract the primitive packet q from coded packet $codedPacket$

if v is the next hop of q **then**

put q into the $buffer_v$

else

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

end if

else

put $codedpacket$ into the $infoPool_v$

end if

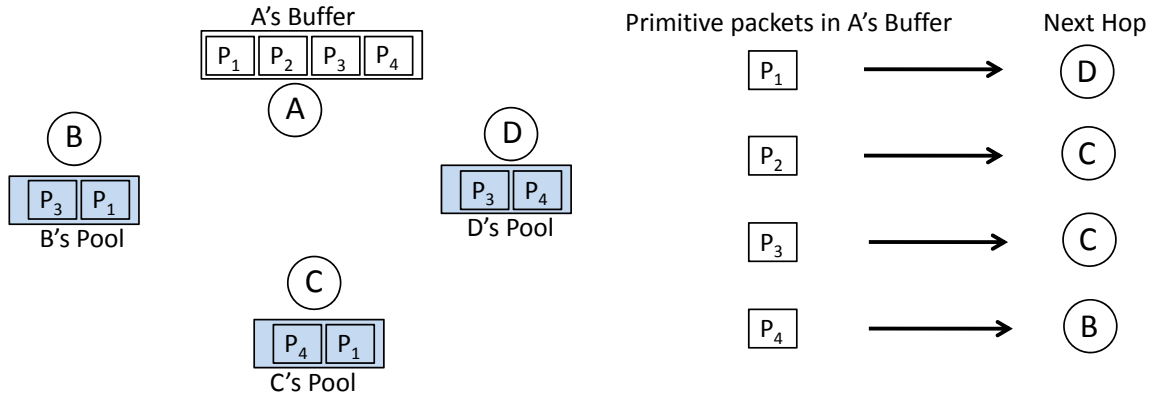
1. Any of the primitive packets' nexthops are not the same. If a transmitter selects the primitive packets P_1, P_2, \dots, P_n and two or more primitive packets P_i, P_{i+1}, \dots, P_j are transmitted to the same nexthop N , when nexthop N receives the coded packet $P' = P_1 \oplus P_2 \oplus \dots \oplus P_n$, it cannot recover primitive packets $P_i \oplus P_{i+1} \oplus \dots \oplus P_j$ from coded packet P' , and $P_i \oplus P_{i+1} \oplus \dots \oplus P_j$ is still a scramble for a receiver. Therefore, path diversity must be satisfied.
2. All the receivers R_1, R_2, \dots, R_n have the packets $P' = P_1 \oplus P_2 \oplus \dots \oplus P_{i-1} \oplus P_{i+1} \oplus \dots \oplus P_n$ XORed from their information pools, if a transmitter selects the primitive packets P_1, P_2, \dots, P_n to be XORed together. In other words, all the receivers R_1, R_2, \dots, R_n have the packets $P_1 \oplus P_2 \oplus \dots \oplus P_{i-1} \oplus P_{i+1} \oplus \dots \oplus P_n$, meaning that all the receivers can recover their demanded primitive packet P_i by calculating $P_i = (P_1 \oplus P_2 \oplus \dots \oplus P_n) \oplus (P_1 \oplus P_2 \oplus \dots \oplus P_{i-1} \oplus P_{i+1} \oplus \dots \oplus P_n)$.
3. Encode the maximum number of primitive packets in each transmission. If a transmitter encodes a large number of primitive packets and broadcasts the coded packets

to its one-hop neighbor nodes, a large number of nexthops would receive them and may recover its demanded primitive packet in one single transmission. Thus, the more primitive packets are encoded in each transmission the more efficient the network throughput can become.

4. The primitive packets selected to be encoded together should have the similar bit length. Should any difference in bit length exist between primitive packets selected by a transmitter, the shorter ones are padded with trailing zeros to their data until their bit length is as long as the largest one's. Then, a transmitter can begin to encode them by the *XOR* coding algorithm and broadcast the coded packets to one-hop neighbors. In such a transmission, some bandwidths are wasted because useless padded zeroes are also transmitted, the process of which consumes the network capacity. Thus, enlarging the shorter primitive packets by padding zeros wastes the bandwidths of the network. Therefore, by selecting primitive packets with similar lengths to be encoded, the efficiency of network bandwidth use can be boosted.

The following example (in [5]) shows how the *XOR* coding algorithm works and gains benefits, and the previous conditions avail. In Figure 3.1(a), node *A* has packets P_1, P_2, P_3, P_4 in its buffer and its one-hop neighbors *B, C, D* overhear some packets. It is assumed that *A* knows which packets one-hop neighbors have. When *A* has a transmission opportunity, if *A* picks up packets P_2 and P_3 to be encoded into $P_2 \oplus P_3$ and transmits it, *C* receives it but can neither extract packet P_2 nor packet P_3 with packets P_1 and P_4 in its information pool. Since packets P_2 and P_3 have the same nextstop, they are not able to be selected to be mixed at the same time. Therefore, any of the primitive packets' nexthops are not the same. (see the previous condition 1).

If *A* picks up packets P_1, P_2 and P_3 , which are to be encoded into $P_1 \oplus P_2 \oplus P_3$, which is received by *D*, *D* would decode $P_1 \oplus P_2 \oplus P_3$ into $P_1 \oplus P_2 = (P_1 \oplus P_2 \oplus P_3) \oplus P_3$. But it cannot extract packet P_1 with packets P_2 . Only when *D* has one more packet P_2 can it recover P_1



(a) A can encode primitive packets to transmit. (b) Nexthops of primitive packets in A's buffer.

Figure 3.1: Let the primitive packets to be encoded maximal numbers in each transmission.

from $P_1 \oplus P_2 \oplus P_3$ with packets P_2 and P_3 . Therefore, all the receivers R_1, R_2, \dots, R_n have the packets $P' = P_1 \oplus P_2 \oplus \dots \oplus P_{i-1} \oplus P_{i+1} \oplus \dots \oplus P_n$ XORed from their information pools, if a transmitter selects the primitive packets P_1, P_2, \dots, P_n to be XORed together (see the previous condition 2). This certainly ensures that all nexthops can recover their demanded primitive packets from the received coded packet.

Because B and D have packet P_3 , and none of the nodes has packet P_2 , packet P_3 is more likely to be decoded and is better than packet P_2 . Therefore, four conditions may occur in terms of packet selection. First, if A selects packets P_1 and P_3 to be encoded, because C has packet P_1 , and D has packet P_3 , C and D can recover their demanded packets from $P_1 \oplus P_3$. Thus, A delivers two packets in one transmission. Second, if A selects packets P_1 and P_4 to be encoded, because B has packet P_1 , and D has packet P_4 , B and D can recover their demanded packets, and A also delivers two packets in this one transmission. Third, if A selects packets P_3 and P_4 , the same result would occur, and B and C can extract their demanded packets, whereas A delivers two packets in one transmission. Finally, if A selects packets P_1, P_3 and P_4 to be encoded, because B has packets P_1 and P_3 , C has

packets P_1 and P_4 , and D has packets P_3 and P_4 , each one can recover its demanded packet from $P_1 \oplus P_2 \oplus P_3$. In this case, A delivers three packets in one transmission. Thus, the more primitive packets to be encoded in each transmission, the better it is for network throughput (see condition 3 mentioned above). Therefore, four approaches should be taken while selecting which primitive packets to be encoded.

3.2 Network Model and Assumptions

In this work, the assumptions of our grid network model is consistent with that of the previous work [6], except for its ignorance of collisions, retransmissions and any traffic pattern, and listed in the following.

- Every grid cell has only one node, which presents the leader of the grid cell and works all the time without crash, and all nodes are synchronized.
- Each node can directly communicate with other nodes in neighboring cells, and while routing packets, local minimal situation never happens here.
- The payload of each packet is the same, and in our simulation, we assume that the payload of each packet is one unit. In other words, this guarantees that the bandwidth of each link is the same.
- The grid network model, different from that in the previous paper, has collisions and retransmissions, and utilizes greedy grid unicast routing, *Unicast*, with which packets are randomly transmitted via the shortest route to neighbor cells.
- Each node needs to know whether previous transmissions are successfully received and decoded by its neighbors and which packets have been overheard by its one hop neighbors.

In this thesis, we assume that the grid network operates in a synchronized-TDMA-manner, and all the ad hoc nodes communicate with each other in Rand-MAC mode. The following is to introduce our simulation network mechanism in detail. In our simulation, time is divided into time slots similar to TDMA system. During each time slot, each node generates one packet with the same probability, called packet generating rate, if its buffer has enough storage. The source and destination pairs of each node are generated randomly and uniformly distributed over all nodes. The operation of Rand-MAC, random MAC, in each node is simple. If one or more packets have transmission opportunities, a node would wait until the beginning of the next time slot, when it has the probability, called Rand-MAC probability, to decide whether to transmit one single packet at each time slot. Only one single packet can be completely transmitted in each time slot. In other words, a node spends one complete single time slot on transmitting one packet from a transmitter to a nexthop.

Collision is defined as the following. Due to the broadcast nature of wireless communications, collision occurs, when two or more of its one-hop neighbors transmit packets during the same time slot. If a node receives two or more packets from different transmitters at the same time slot, it receives either none or only one single packet among these. Thus, when there is no collision, a node would spend one time slot on a successful transmission of one packet. If a node fails to transmit one packet, it does not receive an ACK from a receiver before the end of time slot, and it inserts the packet into its buffer. When the failed transmission packet is taken out next time, a node retransmits it with the Rand-MAC probability to decide whether to retransmit it.

Our simulation network mechanism sets a time limit on retransmission, called maximal retransmission time. If the retransmission of a packet by a transmitter exceeds the maximal retransmission time, it would be discarded by the transmitter. Thus, a node retransmits one packet, until successful retransmission or the times of transmission exceed the maximal retransmission time. In addition, packet loss may occur by another situation. If a buffer in a

receiver does not have enough storage, it would discard incoming packets. Therefore, packet loss is caused by two factors, one being retransmission exceeding the maximal retransmission time and the other being buffer overflow.

3.3 Network Coding Implementation

COPE architecture is taken from the paper [5] and adopted in our work. Our *XOR* coding algorithm is built on *Unicast*. Each node maintains two buffers: one buffer stores demanded primitive packets and the other, the information pool, stores undemanded packets. A node classifies primitive packets into virtual queue by different nexthops. Due to each grid cell with the existence of one node and limitation on the transmission range of nodes, each grid cell maintains at most four virtual queues.

In the system, during each time slot, each packet is generated with the same packet generating rate, if a buffer has enough storage. Each node is prepared to transmit one packet with the Rand-MAC probability in this time slot, if it has primitive packets in its buffer. If one node has transmission opportunities in this time slot, the procedures of the *XOR* coding algorithm can be carried out. The procedures in our work are divided into three parts: the coding procedures of a transmitter, the decoding procedures of a receiver, and the retransmission procedures of a transmitter.

In the first part, a transmitter takes out four primitive packets in four virtual queues by the *XOR* coding algorithm (as described in the previous section), if its buffer has primitive packets. Before encoding them, a transmitter would check whether coded packets are retrieved successfully by all nexthops. As mentioned in the previous section, a node takes out as many primitive packets in four virtual queue as possible, which is 1-4 primitive packets due to existence of four grid neighbors. In the second part, the primitive packets taken out in the previous step are encoded, primitive packet IDs are inserted into the header of the coded packet, and finally, the coded packet is transmitted to all the nexthops by broadcast.

A transmitter sets up a timer to wait for ACKs from all the nexthops. If one neighbor node has collision, it cannot receive a coded packet from a transmitter; otherwise, it would be able to receive the packet. The step described above is depicted in Figure 3.2.

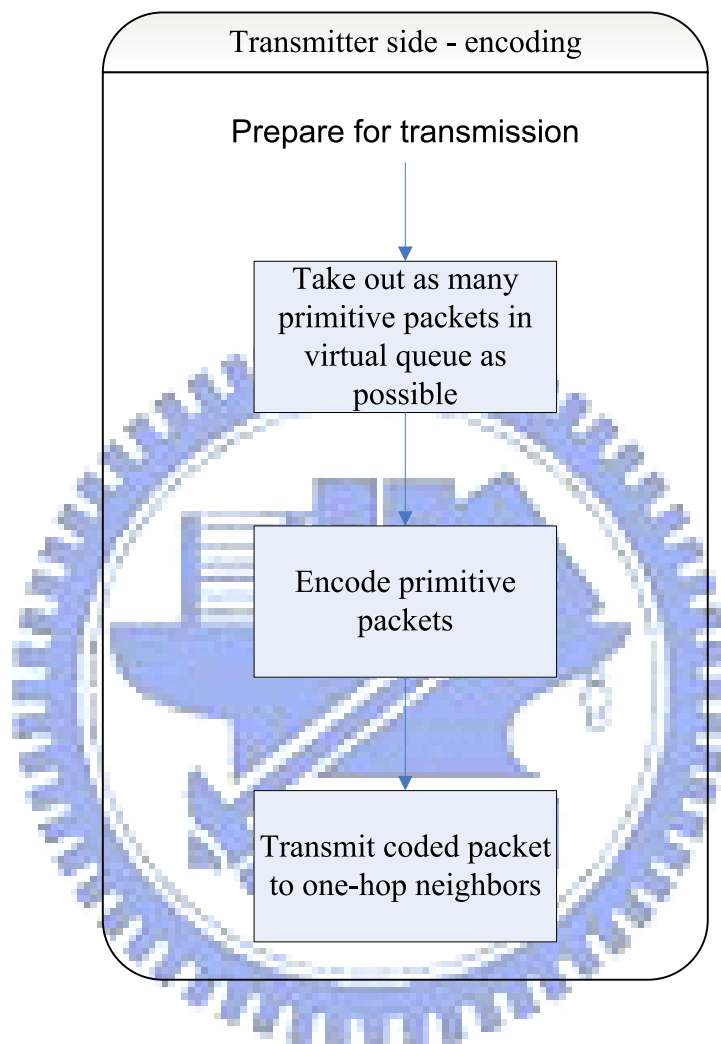


Figure 3.2: Flow chart of transmitter side for encoding.

Decoding procedures of a receiver are described below. When a coded packet arrives at a one-hop neighbor node, it checks whether it is a nexthop. If not, and if the information pool has enough storage, the coded packet is stored into the information pool. If the information pool is full of coded packets, it would be discarded. Each coded packet stays in the information pool within fixed time slots, namely packet alive time. In each time slot, a

node maintains its information pool by discarding coded packets over the packet alive time. Thus, a node discards coded packets in cases of information pool overflow or when packet alive time is passed. If a nexthop receives a coded packet, it tries to retrieve its primitive packet by checking IDs in the header of that the coded packet. If a coded packet is scramble, the node would take it as a coded packet received by a none-nexthop. Otherwise, a nexthop recovers a primitive packet and stores it into its buffer when there is enough storage. If there is no storage in the buffer, the primitive packet would be discarded. This is one factor contributing to packet loss. Then, primitive packets are categorized into virtual queues by different nexthops. If the information pool has enough storage, the primitive packets would also be stored into it; if not, the packets would be discarded. And it responds an ACK with a packet ID to the transmitter. The decoding procedures of a coded packet conducted by a receiver is depicted in Figure 3.3.

Finally, there is the retransmission procedures of a transmitter. When an ACK arrives to a transmitter within the allotted time, it removes the primitive packet corresponding to the packet ID from its buffer. However, when the timer expires, the transmitter would discard the primitive packets selected within the encoding procedures and exceeding the maximal retransmission time. This procedures is described in Figure 3.4.

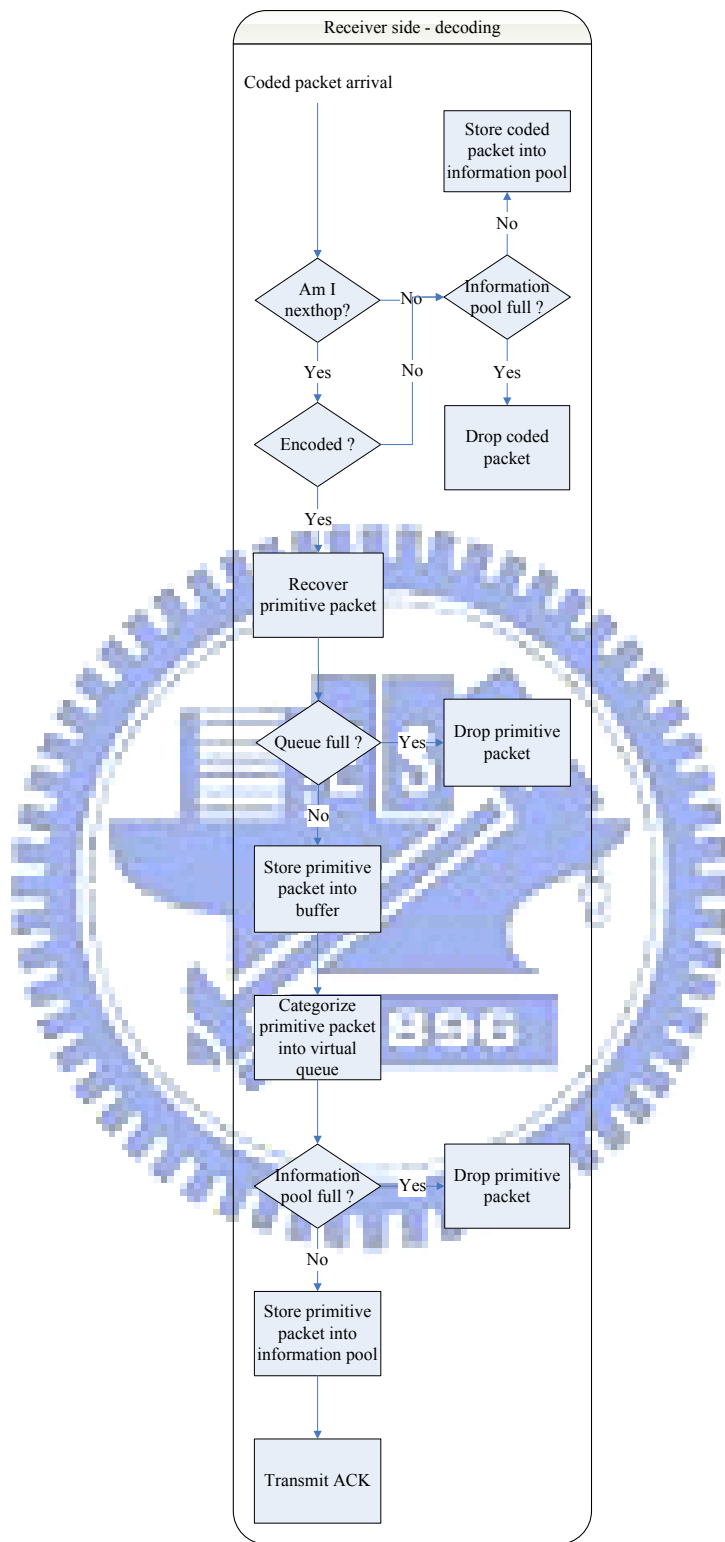


Figure 3.3: Flow chart of receiver for decoding.

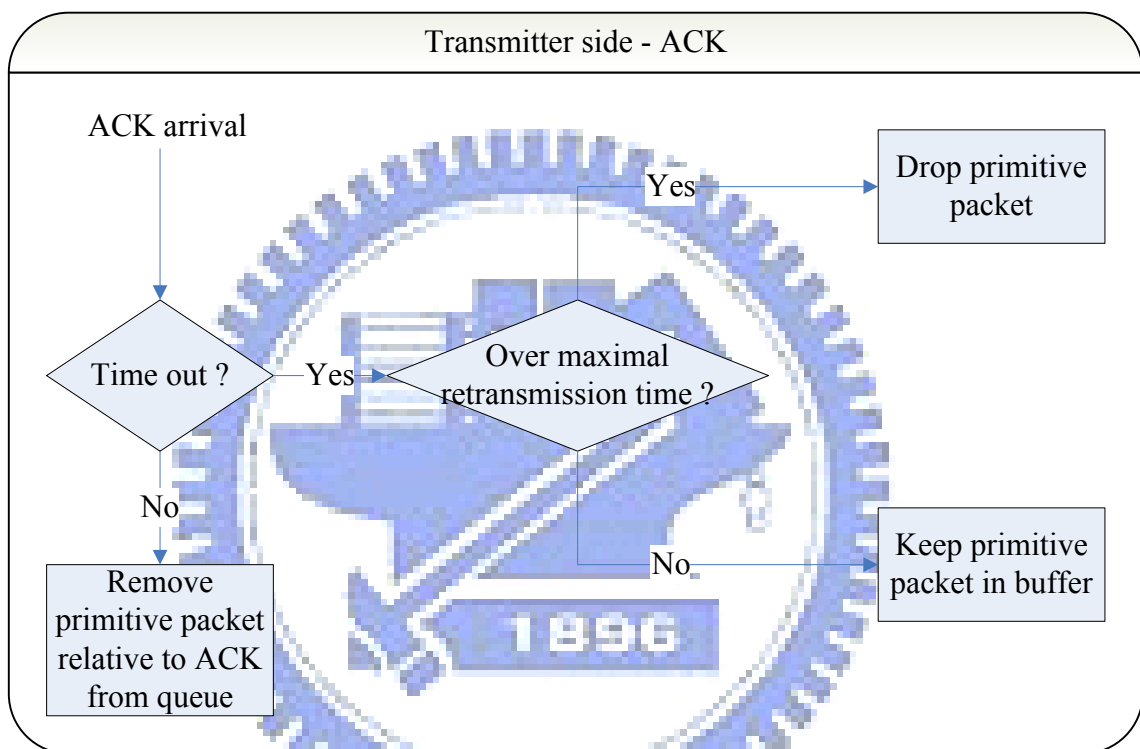


Figure 3.4: Flow chart of transmitter side for ACK.

Chapter 4

Simulation Analysis

4.1 Simulation Settings and Scenarios

Each experiment written in Java program language runs on 50 tasks and the results are averaged. We evaluate the *XOR* coding algorithm as compared with *Unicast*. Parameters of our simulation are introduced below and most of them, such as packet generating rate, Rand-MAC probability, maximal retransmission time and packet alive time, have been presented in the previous sections and re-introduced here.

- Packet generating rate: The probability is utilized while deciding whether to generate one primitive packet during each time slot, if a node has enough buffer storage. Otherwise, it does not generate any primitive packet.
- Rand-MAC probability: The probability is utilized while deciding whether a node has transmission opportunities in each time slot, if a node has packets in its buffer.
- Maximal retransmission time: The maximal times are for the retransmission of one primitive packet. If a packet is retransmitted over the maximal retransmission time, it would be discarded by a node.

- Buffer size: The maximal number of primitive packets that can be stored by a buffer. When the buffer is full of primitive packets, the incoming ones would overflow the buffer, until the buffer regains storage by sending out or discarding packets.
- Information pool size: The maximal number of coded packets that can be stored by an information pool. When an information pool is full of packets, it would discard incoming ones, until there is enough storage. A node has an information pool manager to maintain its information pool.
- Maximal packet alive time: The maximal time slot during which each packet can be stored in an information pool. The pool manager would discard coded packets kept in its information pool over the maximal packet alive time.

The following parameters are used to evaluate the *XOR* coding algorithm as compared to *Unicast*:

- End-to-end delay: The average time slots are spent by per primitive packet on delivering to its destination. When an algorithm has a larger end-to-end delay, it would spend more time slots on relaying. Thus, in this respect, the less time slot end-to-end delay is, the faster packets can be delivered.
- Throughput: The average number of primitive packets delivered to their destinations within one time slot. If an algorithm has better throughput, the network would have a better consumption of packets. In this respect, we know how many packets can arrive at their destinations within one time slot.
- Number of times for transmission: The times for one node to transmit packets. If the number of times for transmission is fewer, the nodes would transmit packets less frequently. In this respect, while transmitting the same number of packets, networks with fewer number of times for transmission are more efficient.

- Packet loss: The number of primitive packets discarded by buffer overflow or retransmitted over the maximal retransmission time. Packet loss is caused by two factors: buffer overflow and collision. If a network has much packet loss, the network obliquely would have poor throughput. In this respect, we clearly understand how much packet loss would have impact on the network throughput through buffer overflow and collision.

We evaluate *XOR* coding algorithm in three scenarios. One has 500000 primitive packets successfully transmitted under light traffic load (in the section 4.2). Another scenario (in the section 4.3) has the same buffer size, 60 primitive packets in each node, and 10×10 grid networks in a stable state by running 500000 time slots. Last scenario (in the section 4.4) has 0.0083, 0.0167 and 0.0333 as its three packet generating rates, and its 10×10 grid networks are also in a stable state. Because the graph of packet generating rate V.S. throughput (in Figure 4.3) is a curve, 0.0083, 0.0167 and 0.0333 is used to present the rise of throughput, the maximal throughput, and the decline of throughput.

4.2 Distribution of Traffic Load of Grid Cell

In the following simulation, the 20×20 grid networks with the *XOR* coding algorithm or *Unicast* randomly produce 500000 primitive packets, which are distributed uniformly over all nodes. It runs with perfect, successful transmissions without any packet loss. Our simulation parameters are described in detail in Table 4.1. Figures 4.1 and 4.2 show the distribution of the number of times for transmission in each grid cell with the *XOR* coding algorithm and *Unicast*.

Table 4.1: Simulation parameters with 500000 primitive packets.

Parameters type	Parameter value
Width of grid network \times Height of grid network	20×20
Packet generating rate per node	0.01 packet/ time slot
Rand-MAC probability	0.25 packet/time slot
Buffer size	1024 packets
Information pool size	8192 packets
Maximal retransmission times	4 times
Maximal packet alive time	10 time slots
Number of primitive packets delivered	500000 packets

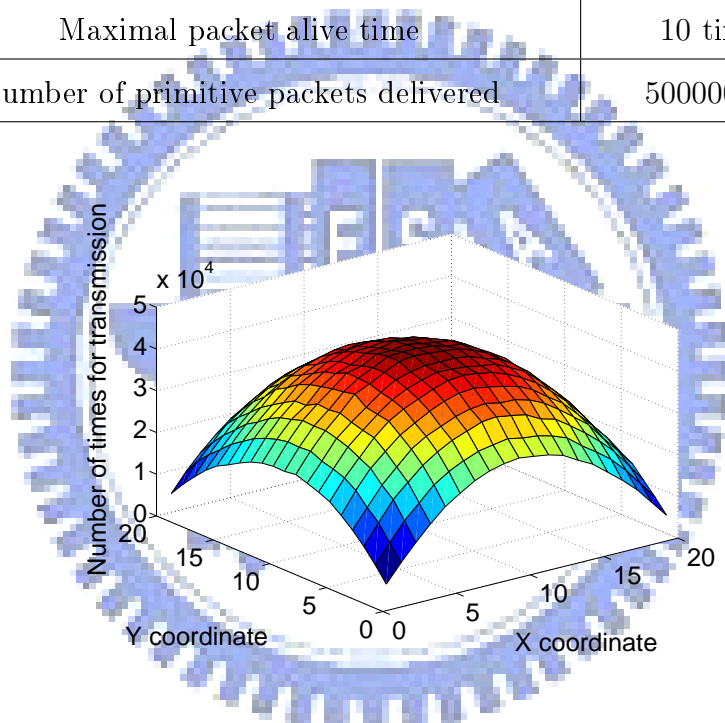


Figure 4.1: Distribution of number of times for transmission using *Unicast*.

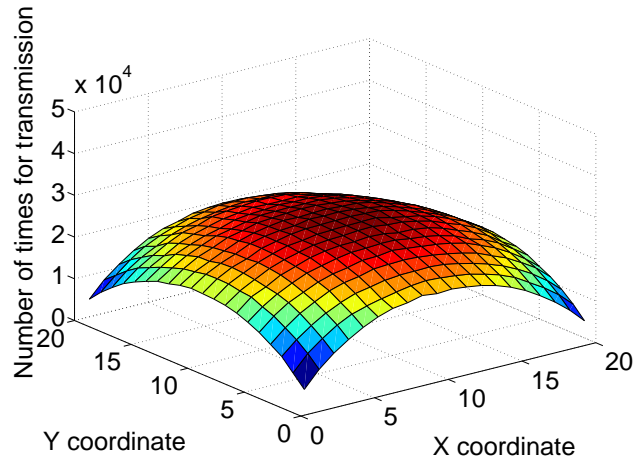


Figure 4.2: Distribution of number of times for transmission using *XOR* coding algorithm.

Figure 4.1 and Figure 4.2 indicate that the number of times for transmitting 500000 primitive packets in the 20×20 grid network using the *XOR* coding algorithm is fewer than that using *Unicast* (The ratio of that using the *XOR* coding algorithm to that using *Unicast* is about 0.6987. In other words, by contract, the *XOR* coding algorithm improves the number of times for transmission in the grid network about 30.0%). Thus, the distribution of the number of times for transmission is flatter using the *XOR* coding algorithm than that using *Unicast*. The maximal number of times for transmission is 26489.2 and 43064.4 for the *XOR* coding algorithm and *Unicast*, so the *XOR* coding algorithm improves the number by about 39.5% as compared with *Unicast*. With successful transmissions of 500000 primitive packets, the ratio of time slots consumed while using the *XOR* coding algorithm to that while using *Unicast* is about 0.7233. It shows that the *XOR* coding algorithm spends fewer time slots on delivering all packets than *Unicast*. Consequently, the *XOR* coding algorithm has less end-to-end delay. Since shorter time is used for totally successful transmissions, collisions caused by the *XOR* coding algorithm are fewer than those caused by *Unicast*. (The ratio of collisions caused by the *XOR* coding algorithm to that by *Unicast* is about 0.6038, which means the ratio is improved by 40.0%.) Finally, in the grid network, the *XOR* coding algorithm is more capable of improving the throughput than *Unicast*, and

can improve the throughput by about 44.4% as compared to *Unicast*. According to the previous simulation analysis, it can be concluded that grid networks using the *XOR* coding algorithm can cut down on the use of time slots, decrease collision occurrences, increase the total network throughput and reduce end-to-end delay as compared with *Unicast*.

4.3 The Impact of Packet Generating Rate

Table 4.2: Simulation parameters in 500000 time slot.

Parameter type	Parameter value
Width of grid network \times Height of grid network	10×10
Packet generating rate per node	$0.005 \sim 0.1$ packet/time slot
Ran-MAC probability	0.1 packet/time slot
Buffer size	60 packets
Information pool size	600 packets
Maximal retransmission time	5 times
Maximal packet alive time	10 time slots
Simulation duration	500000 time slots

Here is a Table 4.2, which shows the simulation parameters in details. Each experiment also runs on 50 tasks, and the results are averaged. In the section 4.3, in each node, the size of the output buffer and that of the information pool are 60 and 600 respectively. We analyze the impact of the packet generating rates from the aspects of throughput, packet loss and end-to-end delay time in the grid network using the *XOR* coding algorithm as compared with *Unicast*.

4.3.1 Throughput

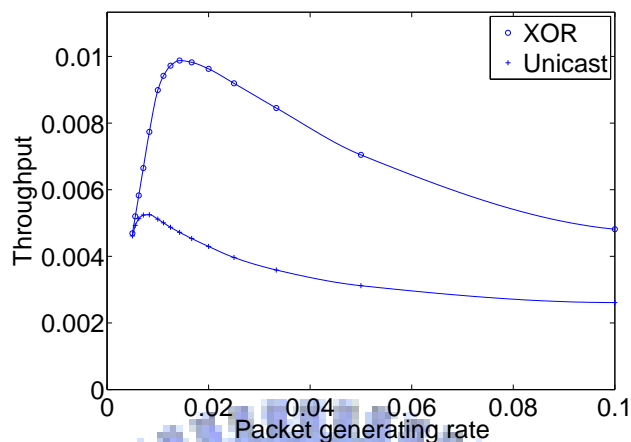


Figure 4.3: Packet generating rate v.s. Throughput

Table 4.3: Average throughput and ratio of throughput in *XOR* coding algorithm to that in *Unicast*. (Saturation point indicates packet generating rate 0.014286 at the maximal throughput in the curve of the *XOR* coding algorithm.)

Packet generating rate p	<i>XOR</i>	<i>Unicast</i>	<i>XOR/Unicast</i>
$p \leq \text{saturation point}$	0.007280	0.005020	1.450227
saturation point $< p$	0.008159	0.003686	2.213481
Overall	0.007805	0.004466	1.747443

Table 4.4: Average number of primitive packets in the buffer. (Saturation point is maximal throughput in the curve of the *XOR* coding algorithm at packet generating rate 0.014286.)

Packet generating rate p	<i>XOR</i>	<i>Unicast</i>
$p \leq \text{saturation point}$	5.475	10.750
saturation point $< p$	43.458	46.756

Figure 4.3 shows that the grid network throughput using the *XOR* coding algorithm is better than that using *Unicast*. At a packet generating rate smaller than the saturation point (It indicates packet generating rate 0.014286 at the maximal throughput in the curve of the *XOR* coding algorithm), the *XOR* coding algorithm improves the average throughput by about 45.0% (see Table 4.3); moreover, at a packet generating rate larger than that point, the average throughput can be improved by up to 121.3%. It is obvious that the *XOR* coding algorithm at a packet generating rate larger than the saturation point can improve the throughput much better. At a packet generating rate smaller than that point, the buffer would have fewer primitive packets on average. Table 4.4 shows that a buffer averagely has 5.475 and 43.45833 primitive packets, respectively at packet generating rate smaller and larger than that point. Hence, at little packet generating rate smaller than that point, cases in which no primitive packets are available for transmission would occur more frequently even with the transmission permission. The *XOR* coding algorithm also has less combination choices. It leads to coded packets combined with less primitive packets. Thus, in one transmission, a node using the *XOR* coding algorithm transmits less primitive packets at packet generating rate smaller than the saturation point. Therefore, the average throughput improvement is better at packet generating rate larger than the saturation point. In this simulation, the average throughput improvement using the *XOR* coding algorithm is about 74.7%.

At packet generating rates smaller than the saturation point, the throughput increases by the packet generating rate. However, at packet generating rates larger than that point, it decreases through packet loss due to buffer overflow. With the packet generating rate smaller than the saturation point, the number of primitive packets transmitted increases by the packet generating rate owing to more combination choices and less buffer emptiness, until the throughput arrives at its maximum value. Hence, the throughput increases along with the increase of the packet generating rate. However, at packet generating rates larger

than that point, the throughput decreases along with the increase of the packet generating rate. In Table 4.4 , at packet generating rates larger than that point, 43.45833 and 46.756 primitive packets exist in the buffer on average, respectively in the *XOR* coding algorithm and in *Unicast*. At packet generating rates larger than that point, the possibility for a buffer to function without storage increases as the packet generating rate increases. Consequently, buffer overflow leads to more and more packet loss, and the throughput decreases along with the increase of the packet generating rate. Because of better throughput with the *XOR* coding algorithm, the grid network can consume more packets and can withstand larger packet generating rates. Thus, the maximal throughput of the *XOR* coding algorithm occurs at larger packet generating rate as compared with *Unicast*. Thus, it is certain that the *XOR* coding algorithm can effectively improve the grid network throughput.

4.3.2 Packet Loss

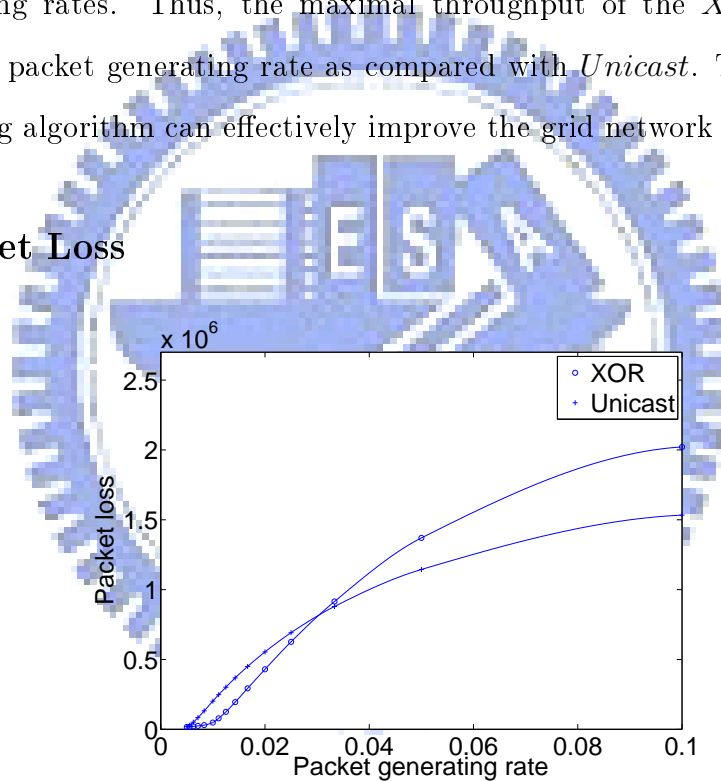


Figure 4.4: Packet generating rate v.s. Packet loss

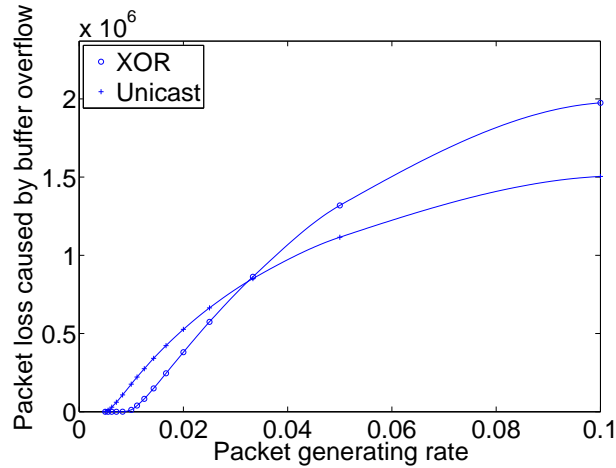


Figure 4.5: Packet generating rate v.s. Packet loss caused by insufficient buffer size

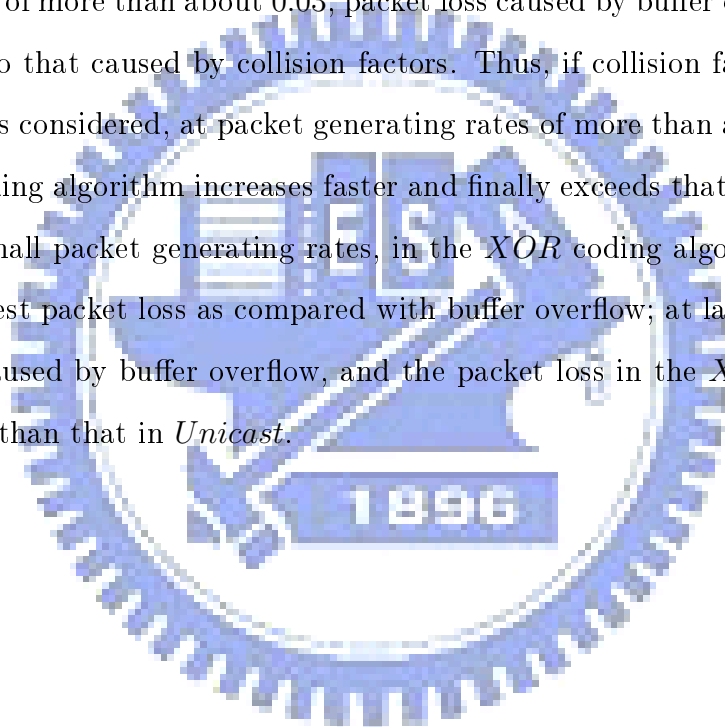
Table 4.5: Ratio of packet loss caused by buffer overflow to total packet loss.

Packet generating rate p	XOR	$Unicast$
$p \leq 0.01$	0.076566	0.745196
$0.01 < p < 0.03$	0.842201	0.938083
$0.03 \leq p$	0.965153	0.975923

Figure 4.4 shows that at packet generating rate smaller than about 0.03, packet loss while using the XOR coding algorithm is less than that while using $Unicast$, whereas at packet generating rate more than 0.03, the packet loss while using the former is more than that while using the latter. Figure 4.5 indicates that at packet generating rate smaller than about 0.01, buffer overflow in the XOR coding algorithm causes less packet loss than that in $Unicast$. This is because the XOR coding algorithm consumes more packets owing to better throughput than $Unicast$, and thus buffer overflow occurs rarely in the XOR coding algorithm. Moreover, Table 4.5 implies that in the XOR coding algorithm, collisions cause most of the packet loss at packet generating rate smaller than about 0.01. However, using $Unicast$, at packet generating rates smaller than about 0.01, packet loss is caused by buffer

overflow and collisions (see Table 4.5). Thus, grid networks in the *XOR* coding algorithm have less packet loss than in *Unicast* at packet generating rate smaller than about 0.01.

Because a node using the *XOR* coding algorithm transmits more packets than using *Unicast*, as more buffers are full, packet loss would increase faster in *XOR* coding algorithm than in *Unicast*. Figure 4.4 and Figure 4.5 show a rapid packet loss increase due to buffer overflow. Between packet generating rates of about 0.01 ~ 0.03, packet loss in the *XOR* coding algorithm is not as much as that in *Unicast*, although packet loss caused by buffer overflow in the *XOR* coding algorithm increases faster. Table 4.5 indicates that at packet generating rates of more than about 0.03, packet loss caused by buffer overflow is the greatest in comparison to that caused by collision factors. Thus, if collision factors are ignored and buffer overflow is considered, at packet generating rates of more than about 0.03, packet loss in the *XOR* coding algorithm increases faster and finally exceeds that using *Unicast*. As we have seen, at small packet generating rates, in the *XOR* coding algorithm, collision brings about the greatest packet loss as compared with buffer overflow; at larger rates, most of the packet loss is caused by buffer overflow, and the packet loss in the *XOR* coding algorithm increases faster than that in *Unicast*.



4.3.3 End-to-End Delay

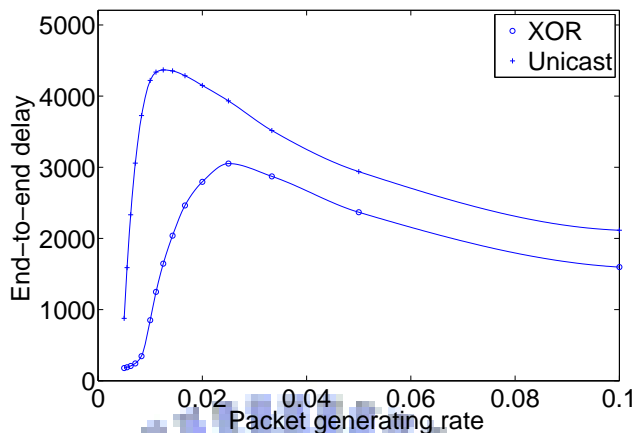


Figure 4.6: Packet generating rate v.s. End-to-End delay

Table 4.6: Average end-to-end delay and ratio of that in *XOR* coding algorithm to that in *Unicast*. (Saturation point indicates packet generating rate 0.025 at the maximal end-to-end delay in the curve of the *XOR* coding algorithm.)

Packet generating rate p	<i>XOR</i>	<i>Unicast</i>	<i>XOR/Unicast</i>
$p \leq \text{saturation point}$	613.9348	3063.915	0.200376
saturation point $< p$	2524.959	3490.063	0.723471
Overall	1473.365	3320.373	0.443735

Table 4.7: Average number of primitive packets in the buffer. (Saturation point indicates packet generating rate 0.025 at the maximal end-to-end delay in the curve of the *XOR* coding algorithm.)

Packet generating rate p	<i>XOR</i>	<i>Unicast</i>
$p \leq \text{saturation point}$	14.60	19.75857
saturation point $< p$	52.49	51.29429

Figure 4.6 shows that the grid network using the *XOR* coding algorithm can improve end-to-end delay better than that using *Unicast*. At packet generating rates smaller than the saturation point (It indicates the packet generating rate of 0.025 at the maximal end-to-end delay in the curve of the *XOR* coding algorithm.), the *XOR* coding algorithm improves the end-to-end delay by about 80.0%; at packet generating rates larger than that point, the delay is improved by about 28.7%. As we see in Table 4.5 and Table 4.7, buffer overflow causes most of the packet loss. Therefore, no matter in the *XOR* algorithm or in *Unicast*, much more packets would spend much time waiting for transmissions and are lost on the way to their destinations at packet generating rates larger than that point. Also, at packet generating rates larger than that point, the average end-to-end delay would be improved to a much lesser extent.

Either in the *XOR* coding algorithm or in *Unicast*, end-to-end delay increases along with the increase of the packet generating rates smaller than the saturation point. At a rate larger than that point, delay decreases along with the increase of packet loss through buffer overflow and packet service time (see Table 4.5). The increase of the packet generating rate leads to more and more packets stored in the buffers. In this case, packets would spend much more time waiting for transmission. The time it takes increases along with the increase of packet generating rate, for packets delivered to their destinations. Thus, end-to-end delay increases while packet generating rate is smaller than the saturation point. Table 4.7 shows that buffers have averagely 52.49 and 51.29 packets, respectively in the *XOR* coding algorithm and in *Unicast*, and each buffer is almost full. As packet generating rate achieves some levels, buffer fullness occurs often and causes a large number of packets to be discarded on the way. Thus, it leads that most packets being able to arrive at their destinations, travel on shorter and shorter routing paths because if the increase of packet generating rate larger than the saturation point. Hence, the end-to-end delay is decreased by the packet generating rate. No matter what packet generating rate is, the *XOR* algorithm can surely improve end-

to-end delay as compared with *Unicast*. In this simulation, the average end-to-end delay improvement made by the use of the *XOR* coding algorithm is about 55.7%.

4.4 The Impact of Buffer Size

Table 4.8: Simulation parameters in 500000 time slot.

Parameter type	Parameter value
Width of grid network \times Height of grid network	10×10
Packet generating rate per node	0.0167, 0.0333, 0.00833 packet/time slot
Ran-MAC probability	0.1 packet/time slot
Buffer size	10 \sim 60 packets
Information pool size	100 \sim 600 packets
Maximal retransmission time	5 times
Maximal packet alive time	10 time slots
Simulation duration	500000 time slots

Table 4.8 describes the simulation parameters in detail. Each experiment also runs on 50 tasks, and the results are averaged. In the section 4.4, each experiment with variable buffer sizes runs at the packet generating rates. We analysis the impact of variable buffer sizes on throughput and packet loss. To clearly know the impact of variable buffer size, packet generating rates of 0.0167, 0.0333 and 0.00833 are selected to represent respectively the rise of throughput, the maximal throughput and the decline of throughput, each in turn representing high traffic load, middle traffic load, and low traffic load.

4.4.1 Packet Loss

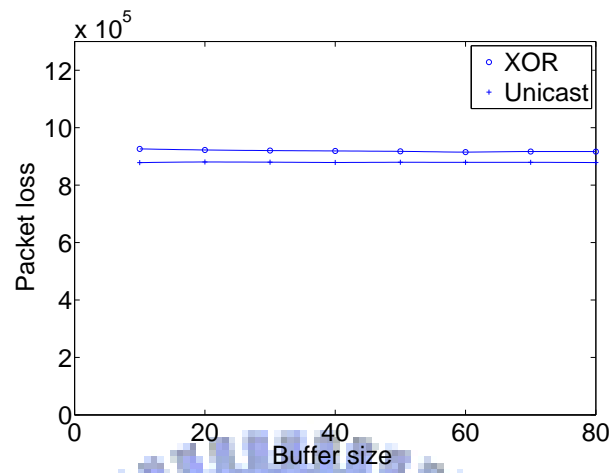


Figure 4.7: Buffer size v.s. Packet loss (at arrival 0.0333 packet/time slot)

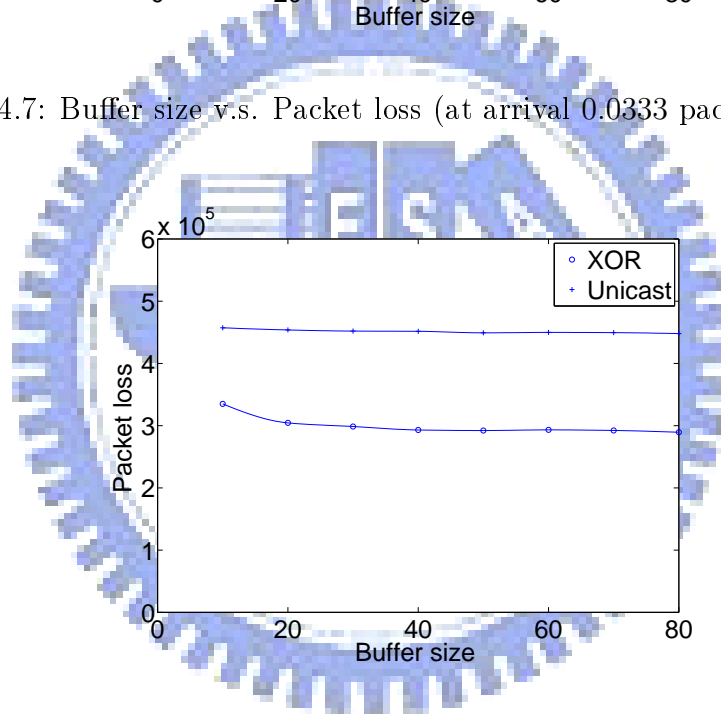


Figure 4.8: Buffer size v.s. Packet loss (at arrival 0.0167 packet/time slot)

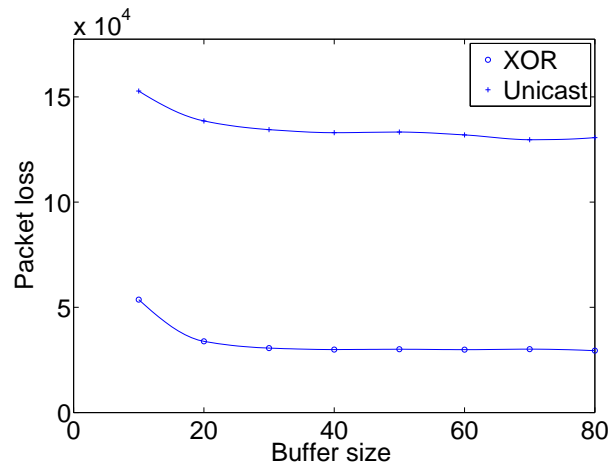


Figure 4.9: Buffer size v.s. Packet loss (at arrival 0.0083 packet/time slot).

Table 4.9: Decrease of packet loss by increase of per buffer size.

Packet generating rate	Buffer size 10 ~ 30	Buffer size 30 ~ 80
0.0083 - <i>XOR</i>	1241.2	19.38
0.0167 - <i>XOR</i>	2080.55	193.02
0.0333 - <i>XOR</i>	468.8	72.62
0.0083 - <i>Unicast</i>	1033.75	81.32
0.0167- <i>Unicast</i>	314.7	84.76
0.0333- <i>Unicast</i>	-61.65	30.38

Figures 4.7, 4.8 and 4.9 show that packet loss decreases as buffer size increases. Table 4.9 shows that using *Unicast*, the increase of per buffer size from 10 ~ 30 averagely reduces packet loss by 1033.75 and 314.7, respectively at packet generating rates of 0.0083 and 0.0167. However, the buffer size increased from 30 ~ 80 averagely reduces packet loss by 81.32 and 84.76, respectively at packet generating rates of 0.0083 and 0.0167. We know that packet loss is improved by the increase of per buffer size, much more by the increase from 10 ~ 30 than by that from 30 ~ 80 at packet generating rates of 0.0083 and 0.01667. Using the

XOR coding algorithm, Table 4.9 shows that the impact of the packet loss improved by increasing the buffer size from 10 ~ 30 is more obvious than that improved by the increase from 30 ~ 80. As the buffer size exceeds 30, the decrease trend of packet loss slows down and ceases rapidly. Overall, no matter whether the *XOR* coding algorithm is utilized or not, this result suggests that the optimal buffer size is 30, and it is concluded that packet loss is decreased by the increase of buffer size.

4.4.2 Throughput

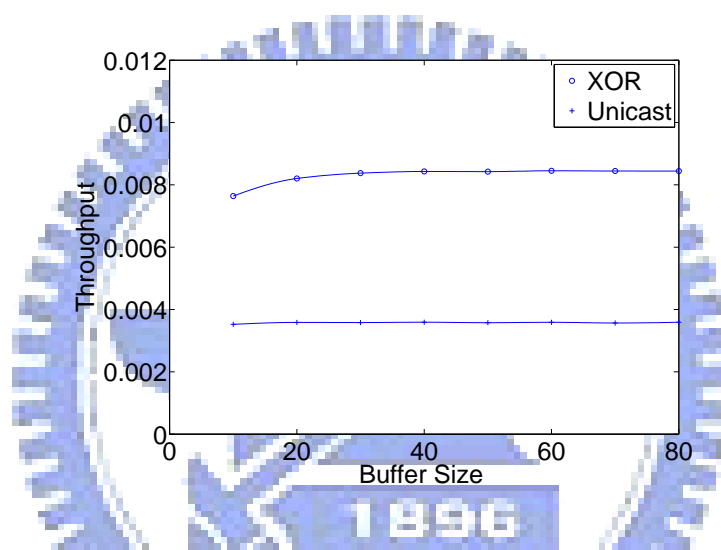


Figure 4.10: Buffer size v.s. Throughput (with packet generating rate 0.0333)

Figures 4.10, 4.11 and 4.12 demonstrate that the throughput of the grid network increases along with buffer size and increases faster from the buffer size 10 ~ 30 than that from the buffer size 30 ~ 80. With or without the *XOR* coding algorithm, as the previous subsection 4.4.1 mentioned, packet loss decreases faster from the buffer size 10 ~ 30 than that from the buffer size 30 ~ 80 (see Figures 4.7, 4.8 and 4.9). Thus, the throughput increases faster with the increase of per buffer size from 10 ~ 30 than with the increase of per buffer size from 30 ~ 80. Table 4.10 shows that, in the *XOR* coding algorithm, the increase of per buffer size from 10 ~ 30 improves throughput by about 7.18%, 12.73% and 9.63%, respectively at packet

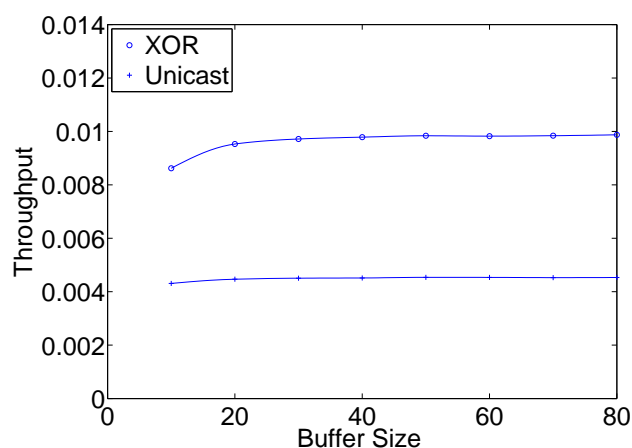
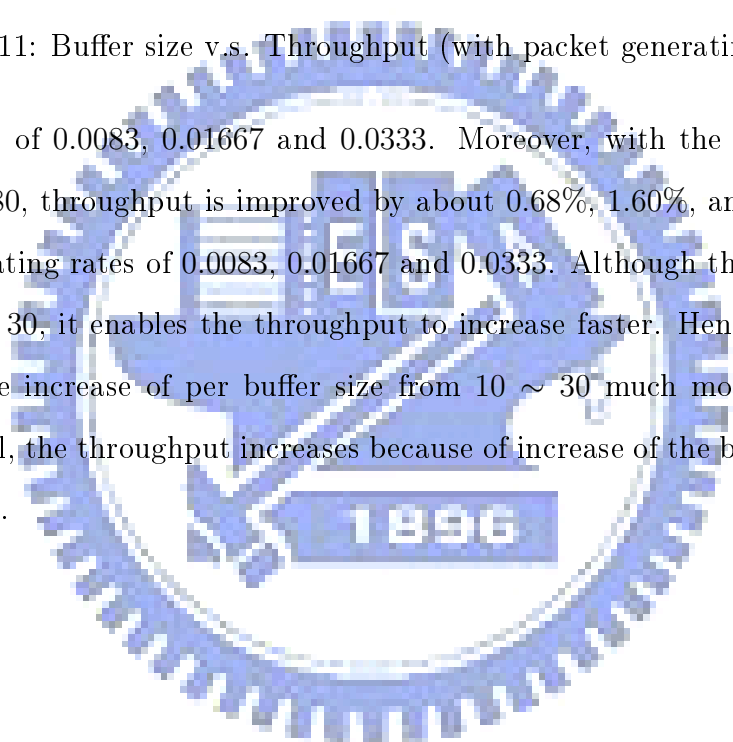


Figure 4.11: Buffer size v.s. Throughput (with packet generating rate 0.0167)

generating rates of 0.0083, 0.01667 and 0.0333. Moreover, with the increase of per buffer size from 30 ~ 80, throughput is improved by about 0.68%, 1.60%, and 0.79%, respectively at packet generating rates of 0.0083, 0.01667 and 0.0333. Although the buffer size increases little from 10 ~ 30, it enables the throughput to increase faster. Hence, throughput is also improved by the increase of per buffer size from 10 ~ 30 much more than by that from 30 ~ 80. Overall, the throughput increases because of increase of the buffer size with limited storage resource.



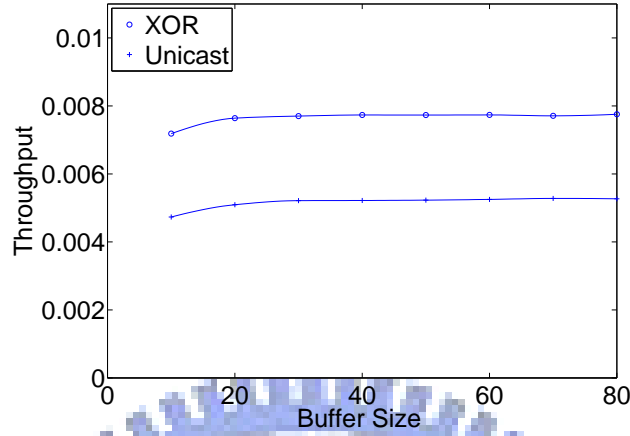


Figure 4.12: Buffer size v.s. Throughput (with packet generating rate 0.0083)

Table 4.10: Ratio of throughput increase in buffer size 30 (or 80) to that in buffer size 10 (or 30).

Packet generating rate	Buffer size 30 to 10	Buffer size 80 to 30
0.0083 - <i>XOR</i>	1.071839	1.006863
0.0167 - <i>XOR</i>	1.127344	1.016077
0.0333 - <i>XOR</i>	1.096358	1.007988
0.0083 - <i>Unicast</i>	1.101762	1.010305
0.0167 - <i>Unicast</i>	1.045612	1.005792
0.0333 - <i>Unicast</i>	1.016299	1.002703

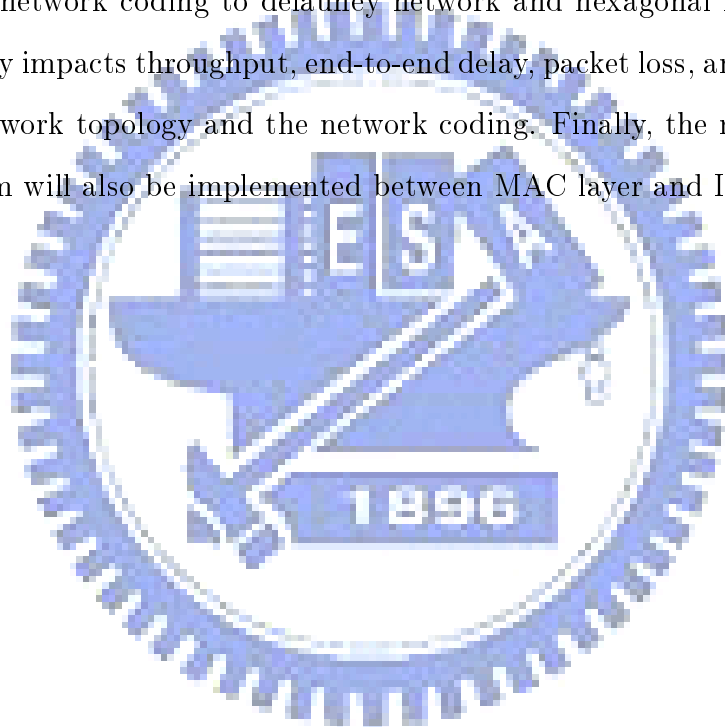
Chapter 5

Conclusion

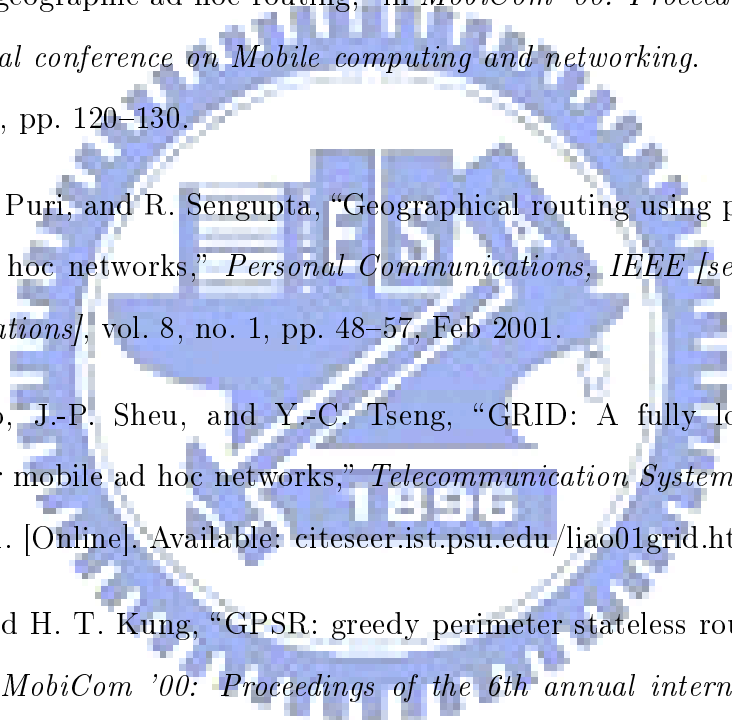
In this thesis, we explore the network coding technique, the *XOR* coding algorithm, to improve the throughput of the grid network, and to examine how packet generating rate and buffer size impact the network throughput, packet loss and end-to-end delay. In our simulation, the *XOR* coding algorithm, being built on greedy grid unicast routing, namely *Unicast*, is evaluated with *Unicast*. Under the light traffic load, 500000 to be exact transmitted, which are generated randomly with their source and destination pairs distributed uniformly over the 20×20 grid network, the distribution of the traffic load of each grid in the *XOR* coding algorithm or in *Unicast* is like a hill, the results of which are similar to , but flatter than those of a previous paper [6] due to collisions and retransmissions. In the 10×10 grid network, which is in a stable state and runs with the same number of time slots, namely 500000 time slots, at packet generating rates of 0.005 0.1, the *XOR* coding algorithm improves the throughput and end-to-end delay by about 74.7% and 55.7% on average as compared with *Unicast*. With respect to variable buffer size, the throughput increases along with the buffer size, and increases faster when the buffer size is between 10 30. On the contrary, packet loss decreases along with the buffer size, and decreases faster when the buffer size is between 10 30. However, as the buffer size exceeds 30, the increase of

the throughput and the decrease of packet loss slows down and ceases rapidly. Thus, with limited resources, this result suggests that 30 is the optimal buffer size. As a summary of the series of simulation analyses, it can be concluded that the network coding can both increase network throughput and reduce end-to-end delay for the grid network under different network environments.

In the future, we will perform several theoretical analyses, such as those on throughput, end-to-end delay, packet loss, with queuing model, and try to find the theoretical traffic load of each grid cell in the collisions and retransmissions network model. Moreover, we will extend the network coding to delauney network and hexagonal network, to know how network topology impacts throughput, end-to-end delay, packet loss, and find the relationship between the network topology and the network coding. Finally, the random linear network coding algorithm will also be implemented between MAC layer and IP layer in the realistic environment.



Bibliography

- 
- [1] J. Li, J. Jannotti, D. S. J. D. Couto, D. R. Karger, and R. Morris, “A scalable location service for geographic ad hoc routing,” in *MobiCom '00: Proceedings of the 6th annual international conference on Mobile computing and networking*. New York, NY, USA: ACM, 2000, pp. 120–130.
- [2] R. Jain, A. Puri, and R. Sengupta, “Geographical routing using partial information for wireless ad hoc networks,” *Personal Communications, IEEE [see also IEEE Wireless Communications]*, vol. 8, no. 1, pp. 48–57, Feb 2001.
- [3] W.-H. Liao, J.-P. Sheu, and Y.-C. Tseng, “GRID: A fully location-aware routing protocol for mobile ad hoc networks,” *Telecommunication Systems*, vol. 18, no. 1-3, pp. 37–60, 2001. [Online]. Available: citeseer.ist.psu.edu/liao01grid.html
- [4] B. Karp and H. T. Kung, “GPSR: greedy perimeter stateless routing for wireless networks,” in *MobiCom '00: Proceedings of the 6th annual international conference on Mobile computing and networking*. New York, NY, USA: ACM, 2000, pp. 243–254.
- [5] S. Katti, H. Rahul, W. Hu, D. Katabi, M. Medard, 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] C.-W. Huang and C.-W. Yi, “The critical grid size and transmission radius for local-minimum-free grid routing in wireless ad hoc networks,” *Master Thesis of Institute of*

Network Engineering, College of Computer Science, National Chiao-Tung University, 2007.

- [7] C. Perkins and P. Bhagwat, “Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computers,” in *ACM SIGCOMM’94 Conference on Communications Architectures, Protocols and Applications*, 1994, pp. 234–244. [Online]. Available: citeseer.ist.psu.edu/perkins94highly.html
- [8] P. Jacquet, P. Muhlethaler, T. Clausen, A. Laouiti, A. Qayyum, and L. Viennot, “Optimized link state routing protocol for ad hoc networks,” *Multi Topic Conference, 2001. IEEE INMIC 2001. Technology for the 21st Century. Proceedings. IEEE International*, pp. 62–68, 2001.
- [9] R. Ogier, “Topology dissemination based on reversepath forwarding (TBRPF),” February 2004. [Online]. Available: citeseer.ist.psu.edu/ogier04topology.html
- [10] B. Bellur and R. Ogier, “A reliable, efficient topology broadcast protocol for dynamic networks,” *INFOCOM ’99. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, vol. 1, pp. 178–186 vol.1, Mar 1999.
- [11] Y. H. D. Johnson and D. Maltz, “RFC 4728 the dynamic source routing protocol (DSR) for mobile ad hoc networks for IPv4,” 2007.
- [12] C. Perkins and E. Royer, “Ad-hoc on-demand distance vector routing,” *Mobile Computing Systems and Applications, 1999. Proceedings. WMCSA ’99. Second IEEE Workshop on*, pp. 90–100, Feb 1999.
- [13] R. Ahlswede, N. Cai, S.-Y. Li, and R. Yeung, “Network information flow,” *Information Theory, IEEE Transactions on*, vol. 46, no. 4, pp. 1204–1216, Jul 2000.
- [14] S.-Y. Li, R. Yeung, and N. Cai, “Linear network coding,” *Information Theory, IEEE Transactions on*, vol. 49, no. 2, pp. 371–381, Feb. 2003.

- [15] S. Fong and R. Yeung, "Variable-rate linear network coding," *Information Theory Workshop, 2006. ITW '06 Chengdu. IEEE*, pp. 409–412, Oct. 2006.
- [16] R. Koetter and M. Medard, "An algebraic approach to network coding," *IEEE/ACM Transactions on Networking*, vol. 11, no. 5, pp. 782–795, Oct. 2003.
- [17] T. Ho, R. Koetter, M. Medard, D. Karger, and M. Effros, "The benefits of coding over routing in a randomized setting," *IEEE International Symposium on Information Theory, 2003*, p. 442, June-4 July 2003.
- [18] 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.
- [19] D. Wang, Q. Zhang, and J. Liu, "Partial network coding: Theory and application for continuous sensor data collection," *Quality of Service, 2006. IWQoS 2006. 14th IEEE International Workshop on*, pp. 93–101, June 2006.
- [20] T. Ho and R. Koetter., "Online incremental network coding for multiple unicasts," in *In DIMACS Working Group on Network Coding, 2005*.
- [21] P. A. C. Y. Wu and S. Y. Kung, "Online incremental network coding for multiple unicasts," in *Technical Report MSR-TR-2004-78, Microsoft Research*, 2004.
- [22] Z. Li and B. Li, "Network coding: the case of multiple unicast sessions," in *in Proceedings of the 42nd Allerton Annual Conference on Communication, Control, and Computing*, 2004.
- [23] 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.

- [24] P. Gupta and P. Kumar, "The capacity of wireless networks," *Information Theory, IEEE Transactions on*, vol. 46, no. 2, pp. 388–404, Mar 2000.
- [25] 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.
- [26] L.-L. Xie and P. Kumar, "A network information theory for wireless communication: scaling laws and optimal operation," *Information Theory, IEEE Transactions on*, vol. 50, no. 5, pp. 748–767, May 2004.
- [27] 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.
- [28] 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.
- [29] 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.
- [30] 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.